CSE 575: Statistical Machine Learning

Final Project Report

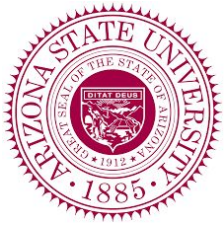
on

Graph Learning - Information Dissemination

by Group 3

Group members:

Aneesh M Gangadhar	1212890677
Anish Oswal	1217914776
Arvind Hasti	1217130408
Sahithi Gurram	1217415680
Srihitha Reddy Vuyyuru	1215180798

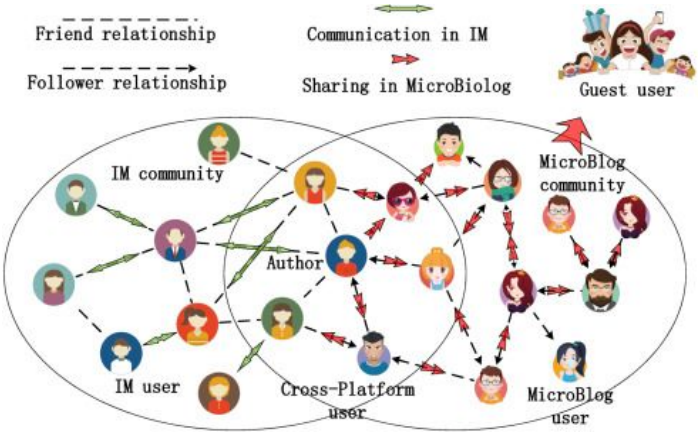


1. ABSTRACT:

Information dissemination is the means by which information is distributed in the real world at large. Information dissemination can be viewed as a graph problem. Our project deals with controlling the propagation of entities such as memes, information, viruses, news, etc in a graph network. In this report, we focus on providing an overview about the methodologies we implemented in our project to control or increase the dissemination of such entities. We ran our algorithms mentioned in this paper on both small and large datasets and also analysed the results to see which algorithm gives better performance for the datasets we have chosen.

2. INTRODUCTION:

Graph theory can solve a majority of computational problems in the real world. Any system can be assumed to be a graph topology because every system is based on some relations. The information dissemination problem can also be viewed as a graph problem. For example, any devices like laptops, phones or servers can be viewed as nodes and we can draw an edge between any two nodes that can interact (eg: send and receive messages) with each other. Managing the dissemination of an entity like meme, virus, etc on a large graph is a challenging problem with applications in various settings and disciplines. In its generality, the propagating entity can be many different things, such as a meme, a virus, an idea, a new product, a blog, a gif etc. The propagation of an entity can be affected by both topology and the properties of the entity like its speed, stickiness or the duration of the infection of a node. Our project focuses on the topology since we assume that we cannot alter the properties of the propagating entity. The goal of our project is to control entity dissemination, i.e either reduce/contain dissemination or enhance/increase dissemination. For example, a blogger might want to increase the dissemination of his/her blog where as decrease the dissemination of a virus. The aim of our project is to implement methodologies to achieve this objective.



3. PROBLEM DESCRIPTION:

Two cases arise in controlling information dissemination: reduce/contain information flow and increase/enhance dissemination.

3.1. Reduce/Contain Dissemination:

To reduce the information dissemination in a graph, the two approaches are to remove either nodes or edges from the graph. Both approaches have their applications in different situations. For example, in social-media networks, we cannot just remove nodes(users) in most of the cases. So we use edge deletion. In other cases such as containing virus propagation, it is effective to use node deletion. The Netshield algorithm removes the k nodes and Netmelt algorithm removes k edges from the graph.

3.1.1. NetShield Algorithm:

Given a graph, this algorithm finds the k best nodes to be removed to minimize the dissemination in the remaining nodes of the graph.

This is the core problem for many applications:

a. In a computer network, we want to find the k best nodes to be removed to minimize the spread of malware.
b. In a law-enforcement setting, given a network of criminals, we want to neutralize or remove the nodes that will maximally scatter the graph.

To compute the k nodes to be removed, we need a measure of vulnerability of the graph, a measure of shield-value for a set of k nodes. To obtain that, Netshield algorithm is used.

3.1.2. NetMelt Algorithm:

This algorithm contains the dissemination by removing a given number of edges, i.e. deleting a set of k edges from the graph to minimize the infected population. For example, we can consider the distribution of malware over a social network. Deleting user accounts may not be desirable, but deleting edges (‘unfriending’ people) may be more acceptable. We implemented both the methods mentioned above and analyzed which method provides better results for the chosen datasets in the results and evaluations section.

3.2. Increase/Enhance Dissemination:

This algorithm enhances the dissemination by adding a given number of edges. Specifically, we want to add a set of k new edges into the graph to maximize the population that adopts the information. For example, we could extend the social network scenario using the recent ‘Arab spring’ which often used Facebook and Twitter for coordinating events: we may want to maximize the spread of a potential piece of information.

After we analyze the results, we will integrate the above algorithms into an ensemble system

which would be capable of determining actions that would need to be taken in order to either increase or decrease the flow of information, given the graph.

4. DATASET DESCRIPTION:

We used three different datasets:

- Karate Dataset - A small dataset with 34 nodes and 57 edges. We used this data set for visualizations[5].
- AS Graph Dataset - A large dataset with nearly 12000 nodes and 25000 edges used for measuring performance and evaluations[4].
- Facebook Dataset - A large dataset with 4039 nodes, 88235 edges used for measuring performance and evaluations[6].

The input in all these datasets are edges.

5. METHODOLOGIES USED:

5.1. REDUCE/CONTAIN DISSEMINATION

5.1.1. k-Node deletion (NetShield)

5.1.1.1. Measure vulnerability:

The first eigenvalue (eqn. 1) of the adjacency matrix is used as vulnerability score. The larger eigenvalue is, the more vulnerable the whole graph is.

$$V(\mathbf{G}) \triangleq \lambda \tag{1}$$

The first eigenvalue λ gives a good measurement of the vulnerability of the graph. As per the epidemic thresholds from immunology, the epidemic threshold is inversely proportional to the first eigenvalue. So a graph with larger eigenvalue has a lower threshold and an epidemic is more likely.

5.1.1.2. Measure Shield-value:

Shield-value quantifies the impact of deletion of k nodes to the vulnerability of the rest of the graph. One of the choices is to use the drop in eigenvalue. But it is computationally expensive to do that. So an approximate and efficient shield-value score is calculated using eqn(2). A set of nodes has higher shield-value if each of them has high eigen-score and they are dissimilar with each other.

$$Sv(S) = \sum_{i \in S} 2\lambda \mathbf{u}(i)^2 - \sum_{i,j \in S} \mathbf{A}(i,j)\mathbf{u}(i)\mathbf{u}(j) \tag{2}$$

5.1.1.3. The Netshield Algorithm:

Initially we compute the first eigenvalue and the corresponding eigenvector and the shield-value of each individual node. Then in each iteration, we greedily select one or more nodes into set S according to shield-value score.

Input: the adjacency matrix **A** and an integer *k*

Output: a set *S* with *k* nodes

Algorithm:

1. compute the first eigen-value λ of **A**; let u be the corresponding eigen-vector **u**(j) (j = 1, ..., n);
2. initialize *S* to be empty;
3. **for** j = 1 to n **do**
4. $\mathbf{v}(j) = (2 \cdot \lambda - \mathbf{A}(j, j)) \cdot \mathbf{u}(j)\ 2$;
5. **end for**
6. **for** iter = 1 to *k* do
7. let $\mathbf{B} = \mathbf{A}(:, S)$;
8. let $\mathbf{b} = \mathbf{B} \cdot \mathbf{u}(S)$;
9. **for** j = 1 to n **do**
10. **if** $j \in S$ **then**
11. let score(j) = −1;
12. **else**
13. let $\text{score}(j) = \mathbf{v}(j) - 2 \cdot \mathbf{b}(j) \cdot \mathbf{u}(j)$;
14. **end if**
15. **end for**
16. let i = argmax j score(j), add i to set *S*;
17. **end for**
18. return *S*.

The straight-forward methods such as Com-Eigs and Com-Eval are computationally intractable. The speed up of Netshield algorithm over both Com-Eigs and Com-Eval is very high, several orders of magnitude and the speed up increases with increase in size of the graph.

5.1.2. k-Edge deletion (NetMelt)

The edge deletion method works on the concept of deleting k important edges which could contain information dissemination. The edges that are to be deleted are selected by eigenvalue optimization.

Edge deletion over Node deletion

To describe the importance of edge deletion over node deletion, we consider a real situation of a

social networking site, where individuals' accounts are considered as nodes and their relationship with each other account as edge. In a circumstance of spread of malware, unfriending (deleting edges) would be a better way to reduce the spread of malware than deleting the individual accounts (nodes). Hence, edge deletion is an efficient way of suppressing the information dissemination in certain kinds of data sets.

Adjacency matrix

In graph theory and computer science, an adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

Eigenvalues and Eigenvectors calculation

The eigenvalues of a matrix A is calculated by solving the equation,

$$A - \lambda I = 0$$

Where λ is the eigenvalues and I is the identity matrix of the size of A.

The corresponding eigenvectors of the corresponding eigenvalues are calculated by

$$A\nu = \lambda\nu$$

Where ν is the corresponding eigenvector of the eigenvalue λ

Concept of line graph

Given a directed graph **A**, it's directed line graph L(**A**) is a graph such that each node of L(**A**) represents an edge of **A**, and there is an edge from a node e_1 to e_2 in L(**A**) if and only if for the corresponding edges (i_1 , j_1) and (i_2 , j_2) in **A**, $j_1 = i_2$.

Eigenvalue optimization as a solution

The leading eigenvalue of the adjacency matrix serves as the single sufficient parameter in the calculation of the epidemic threshold (a metric that quantifies infection of virus In real networks.

$$\tau = 1/\lambda_{1,A}$$

Where $\lambda_{1,A}$ is the leading eigenvalue of the adjacency matrix. τ is the epidemic threshold.

Net-Melt

The Net Melt algorithm is based on decreasing the leading k edges which would cause the maximum decrease in the the leading eigenvalue. The eigenvalues and the corresponding left and right eigenvectors of the Adjacency matrix is computed. The leading eigenvalue is located and its corresponding left and right eigenvector is considered. The minimum value of the left eigenvector if negative, the entire vector is multiplied by -1. The same thing is repeated for the right eigenvector as well. The scores of the m edges is computed by multiplying **u** and **v** element wise. The top *k* scores are computed and their edges are considered for deletion.

Net-Melt algorithm

Input: adjacency matrix **A**, number of edges to be deleted *k*

Output : the deleted *k* edges.

Algorithm:

1. The eigenvalues and vectors of the adjacency matrix are computed.
2. The leading eigenvalue is located and the corresponding left and right eigenvectors of the leading eigenvalue are computed and they are designated as **u** and **v** respectively.
3. **if** $min(\mathbf{u}) < 0$

   $\mathbf{u} = -\mathbf{u}$

   end if
4. **If** $min(\mathbf{v}) < 0$

   $\mathbf{v} = -\mathbf{v}$

   end if
5. For each edge (e), index from 1 to m in the line graph, with (i,j) index in the original matrix, **do**

   $score(e) = \mathbf{u}(i).\mathbf{v}(j)$

   end for
6. Return the top *k* edges with maximum scores.

Algorithm complexity:

Time complexity - **O**(*mk* + *n*)

Space Complexity - **O**(*m* + *n* + *k*)

5.2. INCREASE/SPREAD DISSEMINATION

5.2.1. k-Edge Addition (NetGel)

The edge addition method is based on the concept of adding k important edges to the graph which increase the information dissemination. The edges to be added are selected on the basis of maximizing the eigenvalue.

The core idea behind the algorithm is an eigenvalue optimization problem which maximizes the leading eigenvalue λ.

Eigenscores

Eigenscores are the left(u) and right(v) eigenvectors that correspond to the leading eigenvalue. They are always more than or equal to zero.

Net-Gel Algorithm:

Input: Adjacency matrix **A**, number of edges to be added *k*

Output: The added *k* edges.

Algorithm:

1. The eigenvalues and their corresponding left(**u**) and right(**v**) eigenvectors are computed corresponding to the leading eigenvalue(**u**, $\mathbf{v} \geq 0$).
2. The maximum in-degree and out-degree of the adjacency matrix is calculated respectively,
3. Then the subset of the *k* + in-degree nodes with the highest left eigenscores **u** is found and indexed.
4. Similarly, the subset of the *k* + out-degree nodes with the highest right eigenscores **v** is found and indexed separately.
5. **for** each edge in the in-degree and out-degree nodes represented by (i,j) respectively **do**

   score(e) = **u***(i)* × **v***(j)* and further indexed.

   end for
6. Return the top *k* edges which are non-existing and have the highest scores among the edges.

Algorithm Complexity:

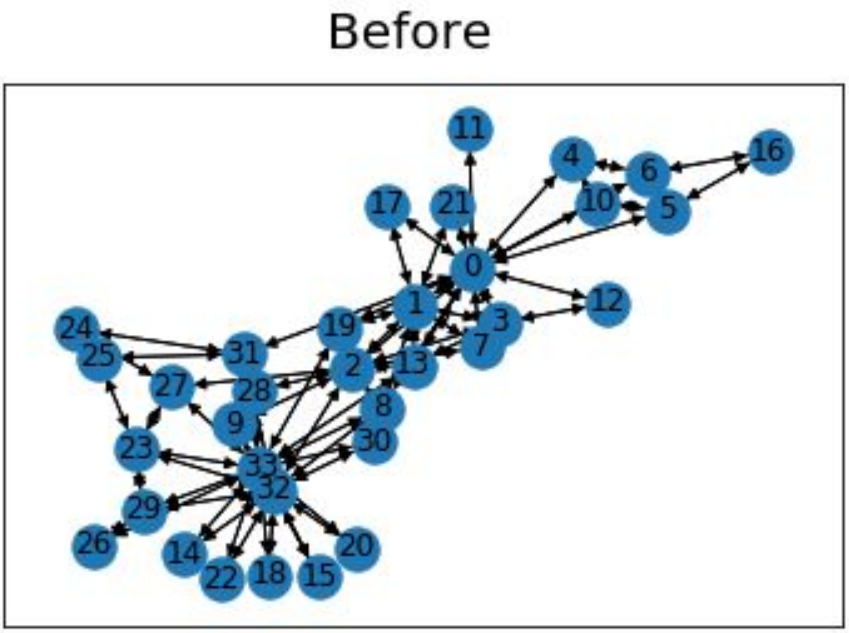
Time complexity - $\mathbf{O}(m + nt + kt^2)$

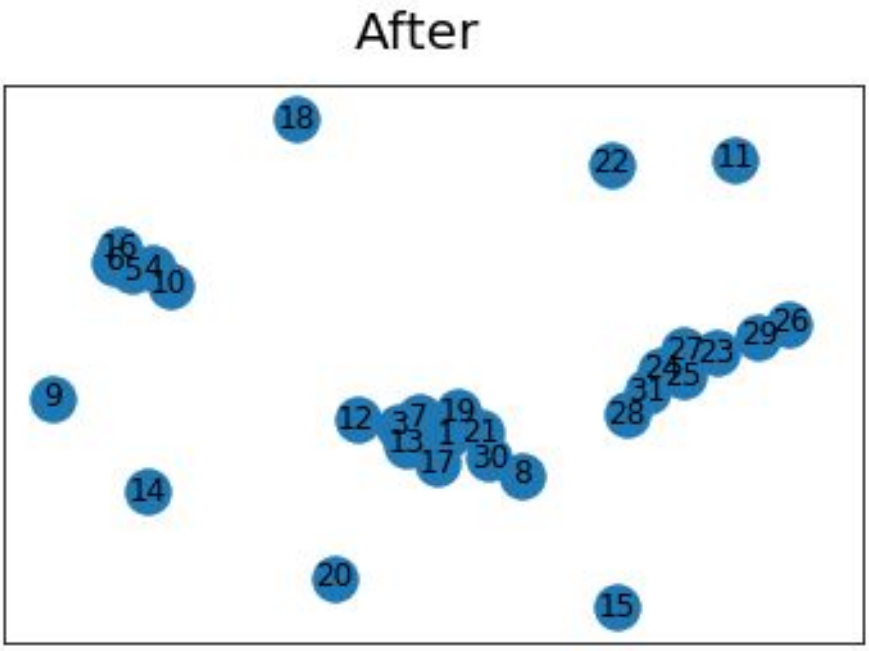
Space complexity - $\mathbf{O}(m + n + t^2)$

6. RESULTS AND EVALUATIONS (also LARGE DATASETS)

6.1. K-node Deletion - Netshield

We have run the netshield algorithm for the Karate dataset with k=4 and obtained the results and the corresponding plot as follows:

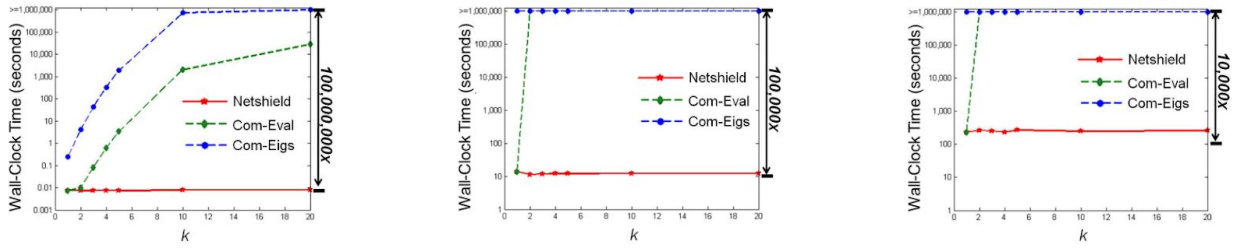




The nodes 0, 2, 32, 33 are removed.

We have used the max-flow parameter as performance evaluation metric for the netshield algorithm. The max-flow values before and after removing k=4 nodes are 32 and 29 respectively.

Comparison of Netshield algorithm with Com-eigs and Com-eval for the different datasets: karate, Authorship network and netflix data. (Netshield is producing best results in all datasets)



(a) *Karate* (b) *AA* (c) *NetFlix*

NetShield

6.2 K-Edge Deletion - Netmelt:

The results of running the net-melt algorithm for the karate dataset with k = 5 and k = 10 respectively and the obtained results and the corresponding plot are as follows:

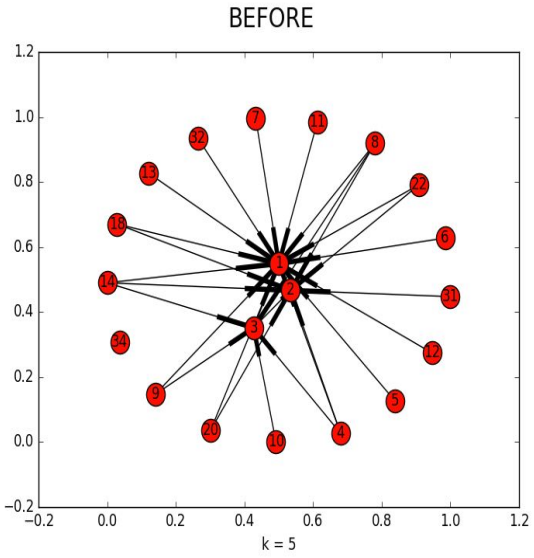


Fig: Graph for Karate Dataset before deleting edges

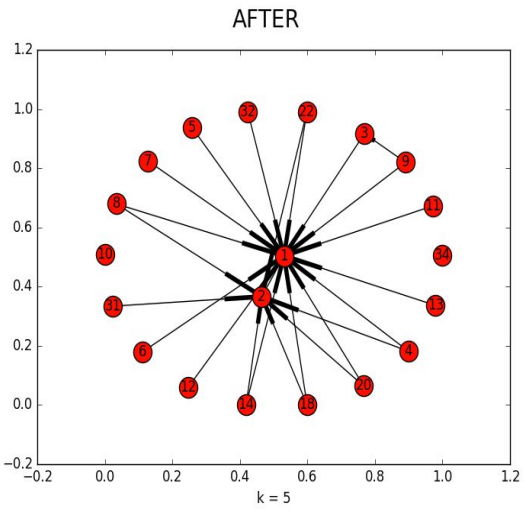


Fig: Graph after removing k = 5 edges

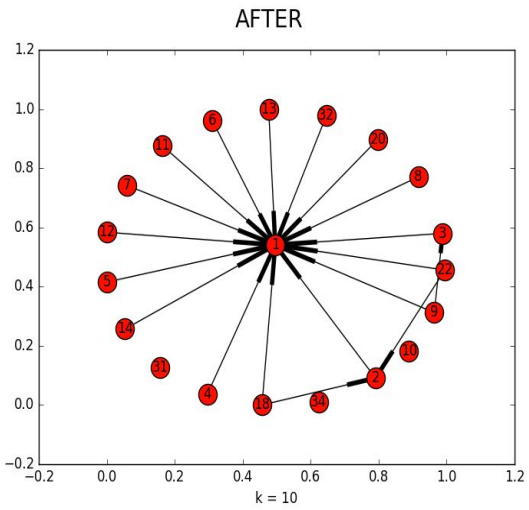


Fig: Graph after removing k = 10 edges

We computed maxflow of the graph before and after deleting K-edges.
Max flow before deleting 10 edges: 32
Max flow after deleting 10 edges: 25
Max flow is reducing when we delete edges from the given graph.

6.3 K-Edge Addition - Netgel:

Karate Dataset:

The results of running the net-get algorithm for the karate dataset with k = 5 and k = 10 respectively and the obtained results and the corresponding plot are as follows:

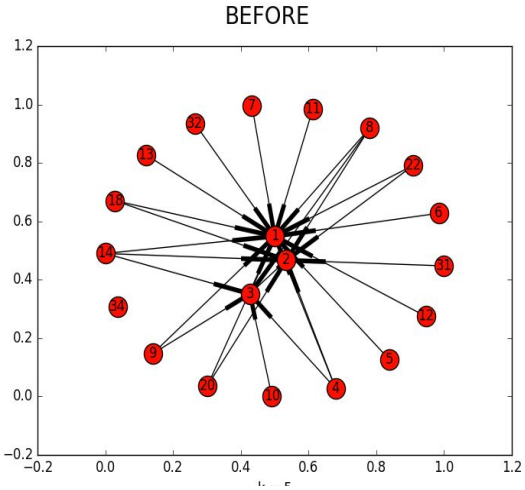


Fig. Graph for Karate Dataset before adding edges

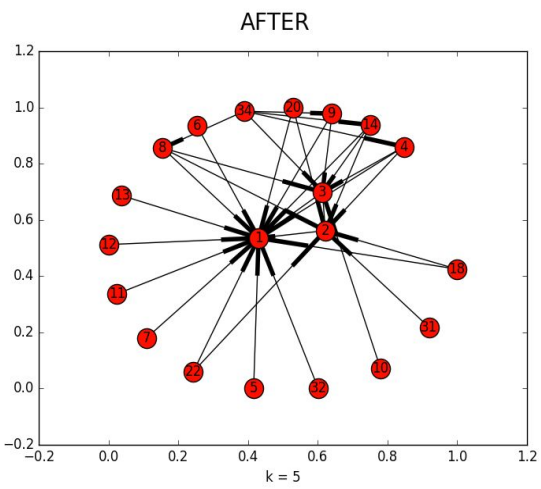


Fig: Graph after adding k = 5 edges.

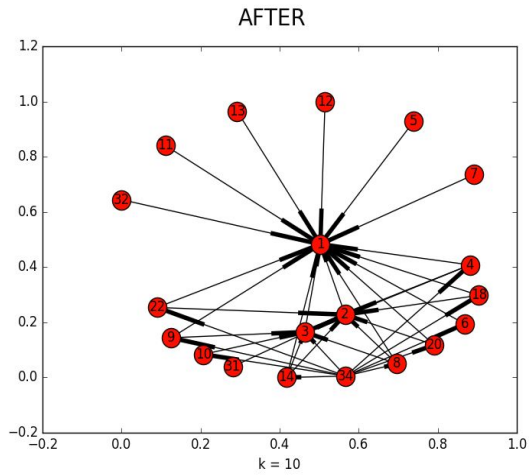
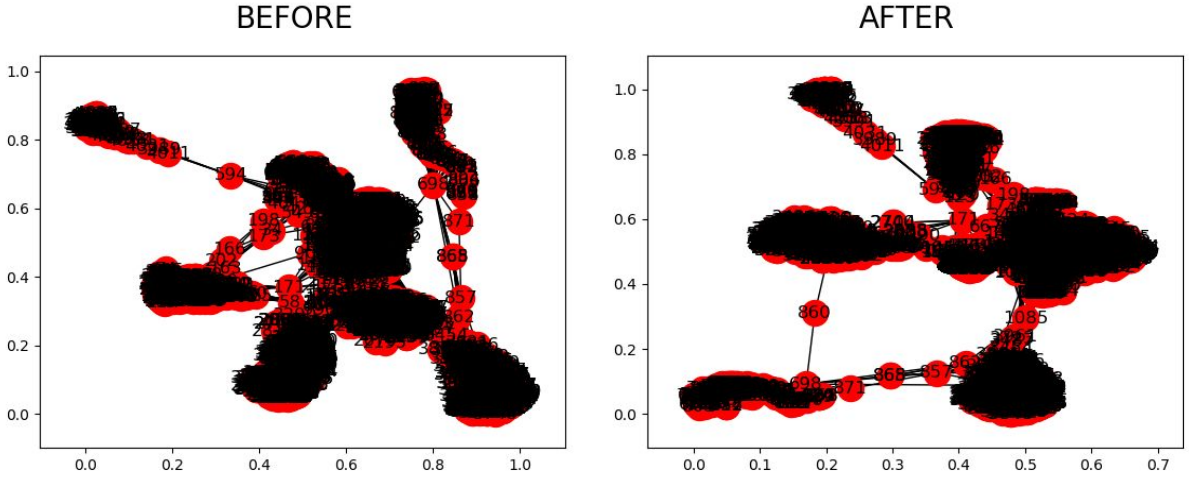


Fig. Graph after adding k=10 edges.

We computed maxflow of the graph before and after adding K-edges.
Max flow before adding 10 edges: 32
Max flow after adding 10 edges: 45
Max flow is increasing when we add edges to the given graph.

Facebook Dataset:

The results of running the net-gel algorithm for the Facebook dataset with K = 10,000 and the obtained results and the corresponding plot are as follows for before adding and after adding 10,000 edges to the large dataset graph; thus increasing the information dissemination in our graph:



k = 10,000

7. RELATED WORK

In literature, there are many node importance measurements including betweenness centrality based on both shortest path[7] or random walks, PageRank[8], HITS[9] and coreness score. The concepts of immunization in virus propagation and epidemic thresholds involve power-law graphs, study of heuristics[10] of immunization policies. The spectral graph analysis pioneered from Fiedler's seminal work[11]. Representative follow-up works use the eigen-vectors of the graph. The closest related work to the k-edge deletion algorithm[12] proposes a convex optimization based approach to approximately minimize the leading eigenvalue of the graph. However, the method is based on semi-definite programming and does not scale to large graphs.

8. CONCLUSION

We studied vulnerability of real graphs with an objective to understand and identify the actions that need to be taken to increase or decrease the information dissemination within the graph. We studied a novel definition for shield-value score and an efficient and scalable algorithm (Netshield) to find a set of nodes with the highest shield-value score. We studied the plots of the graphs before and after removing the nodes and observed that max-flow (evaluation metric) reduced accordingly. We observe that for a large family of information dissimilation processes, the problem boils down to the eigenvalue optimization problem. The leading eigenvalue is used as a key parameter for both netmelt and netgel. On real large graphs, we observe that edge deletion is more effective than node deletion since it operates on the edge level. Both netmelt and netgel scale to large graphs. We also observe that the max-flow reduces significantly when we use netmelt.

9. FUTURE WORK

For the netshield algorithm, the 'k' has to be supplied by the user because it is application dependent. However, we can use some heuristic approaches to find optimal k for the given graph. Since the 'k' value would depend on the requirement (increase or decrease dissemination) it can also be treated as a machine learning problem where the values of 'k' can be learned for different types of graph for their respective requirements.

10. REFERENCES:

[1]Tong, H.; Prakash, B. A.; Eliassi-Rad, T.; Faloutsos, M.; and Faloutsos, C. 2012. Gelling, and melting, large graphs by edge manipulation. In Proceedings of the ACM International Conference on Information and Knowledge Management, 245–254.

[2]H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In ICDM, pages 1091–1096, 2010.

[3]Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. SRDS, 2003.

[4]http://topology.eecs.umich.edu/data.html
[5]http://networkrepository.com/soc-karate.php
[6]https://snap.stanford.edu/data/egonets-Facebook.html
[7]L. C. Freeman. A set of measures of centrality based on betweenness. Sociometry, pages 35–41, 1977.
[8]L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).
[9]J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In ACM-SIAM Symposium on Discrete Algorithms, 1998.
[10]R. Cohen, S. Havlin, and D. ben Avraham. Efficient immunization strategies for computer networks and populations. Physical Review Letters, 91(24), Dec. 2003.
[11]M. Fiedler. Algebraic connectivity of graphs. 1973.
[12]A. N. Bishop and I. Shames. Link operations for slowing the spread of disease in complex networks. EPL, 95(1), 2011.