Efficient Link Cuts in Online Social Networks

Junjun Ruan[†] Jing Deng[†*] George A. Amariucai[‡] Shuangqing Wei[§]
[†]Dept. of Computer Science, University of North Carolina at Greensboro, Greensboro, NC, USA.
[‡]Dept. of Electrical and Computer Engineering, Iowa State University, IA, USA
[§]Dept. of Electrical and Computer Engineering, Louisiana State University, LA, USA
[*]Corresponding author, email: jing.deng@uncg.edu

***Abstract*— Due to the huge popularity of online social networks, many researchers focus on adding links, e.g., link prediction to help friend recommendation. So far, no research has been performed on link cuts. However, the spread of malware and misinformation can cause havoc and hence it is interesting to see how to cut links such that malware and misinformation will not run rampant. In fact, many online social networks can be modeled as undirected graphs. In this paper, we investigate different strategies to cut links among different users in undirected graphs so that the speed of virus and misinformation spread can be slowed down the most or even cut off. Two measures are chosen to evaluate the performance of these strategies: Average Inverse of Shortest Path Length (AIPL) and Rumor Saturation Rate (RSR). AIPL measures the communication efficiency of the whole graph while RSR checks the percentage of users receiving information within a certain time interval.**

I. Introduction

Online social networks such as Facebook and Twitter make news dissemination faster than ever. However, viruses, malware, misinformation (such as rumors) will also propagate quickly in such networks. Such a propagation speed would be hard to imagine in the old days of flyers and/or floppy-drive virus infections. The connections in online social networks can be modeled as graphs consisting of many nodes (users) together with a set of edges. These graphs can be either directed or undirected, depending on whether the edges are directional or not. For example, Twitter's user graph is usually modeled as a directed graph while Facebook's user graph is an undirected one. When two users become friends on Facebook, for instance, a new edge will be added on the graph. A graph is connected when there is a path between every pair of nodes [1]. In general, more edges in a graph will likely lead to higher speed of (mis)information spread, and vice versa. A natural solution to address the spread of misinformation or rumor is to cut some edges. The question is which of the edges we should cut to minimize the spread of misinformation or rumor.

The spread of rumor is certainly correlated to network connectivity, which is one of the basic concepts in graph theory. There are many definitions of connectivity, from average shortest-path length (APL), network diameter, to algebraic connectivity [2]. In fact, one of such checks the minimum number of elements (nodes or edges) that need to be removed to disconnect the remaining nodes from each other, thus splitting the graph [1].

In this work, we focus on the problem of how to choose edges to cut. A cut is defined as the removal of an edge.[1] With the removal of some edges, the new graph will be less connected and thus it will take longer time for rumor to reach nodes. More specifically, a rumor will reach fewer nodes within a certain delay, limiting its impact. And the goal here is to lower such impacts the most with efficient cuts.

We design an algorithm named CDegree Cut, which selects some of the edges to cut mainly based on node degree comparison. Here, degree is simply defined as the number of edges connected to the node. The selection is interestingly not fixed upon the order of the node degree. Instead, it depends on how much delay (defined as the number of hops that a rumor is allowed to travel) before which time we measure a so-called Rumor Saturation Rate, RSR. If the delay is larger than APL, the so-called low-low cuts would perform the best. Instead, if the delay is shorter, the so-called high-random cuts should be chosen.

The remainder of this paper is structured as follows. Section II presents the related work; Section III defines CDegree Cut; Section IV compares the performance of CDegree Cut with random cuts based on real-life Facebook data; finally, Section V concludes this paper.

II. Related Work

In researches related to link changes, many researchers have investigated the link prediction problem. For example, Backstrom and Leskovec [3] assigned weights to the edges and identified the heavy weights indicating the occurrence of new links. Based on such an observation, they developed an algorithm to predict potential links with supervised random walks.

The link change problem also plays an important role in recommender systems for online shopping website such as Amazon. Much work has been reported recently in this field. Sarwar et al. proposed an item-based collaborative filtering recommendation algorithm by identifying relationships between different items. The proposed technique was reported to have outperformed traditional user-based algorithm in producing higher quality recommendations and more recommen-

[1]Note that we use "cut" with a broad meaning of stopping rumor spread, but not necessary meaning the physical removal of the edge thus cutting the flow of "good" information at the same time. For example, within the financial/manpower budget, it might be beneficial to notify/monitor a limited number of links/nodes of the rumor spread. In this sense, a "cut" is to only stop the flow of rumor information on the edge by the means of, for instance, selective filtering.

dations per second for millions of users [4]. Huang exploited the connection between link prediction and graph topological structure, analyzed generalized clustering coefficients, and designed a cycle formation model for link prediction [5].

The behavior and damage of the rumor/misinformation spread in different networks have also been investigated. In online social networks, reducing connectivity can slow down or even stop rumor spreading. Chierichetti et al. studied the performance of rumor spreading in the classic preferential attachment model and compared the efficiency of disseminating information among different strategies: the standard PUSH-PULL, PUSH, and PULL strategies [6]. In email networks, viruses can be transmitted quickly through attachments. An email network is a graph with email addressbooks as sources and edges representing communication. Newman et al. presented techniques to prevent virus infection by analyzing how they spread [7].

In citation networks, nodes represent papers, edges represent citations. Therefore, if one paper cites another, one directional edge would be added between these two. Hummon and Doreian developed a new algorithm to analyze a citation network describing the development of DNA theory. The selected papers are identified through their structural connectivity in the network [8].

Recently a study on the analysis of connectivity damage to a graph has been performed by Cartledge and Nelson. Their motivation was that traditional methods such as the size of the largest connected component do not reflect a graph's connectivity especially when it is disconnected, instead they presented a new metric: average inverse path lengths (AIPLs) [9]. This metric can be used to measure the influence caused by adding new edges in the graph.

Hayel and Zhu [10] studied attacked links as well as link recovery on network resilience. They modeled link failures as well as link recovery as a Markovian process and computed the algebraic connectivity of the dynamic network. An optimized recovery strategy has been developed through the analysis.

Compared to these related work, our work differs in the sense that we focus on the selection of links (edges) to remove so that misinformation spread will slow down the most. We investigate different strategies to choose links for removal. To the best of our knowledge, this has never been investigated.

III. CDegree Cut

In general, the selection of edges to cut can be separated into two steps: deciding which node's links to cut and deciding which link from the chosen node to cut. While there are many different selection criterion in making these two decisions, we focus on a natural node property: node degree, defined as the number of neighbors that the node has. While there are many different other criterion for selection, we believe that node degree is a natural indication of how quickly a node can spread the rumor further. Our study below shows that, even in such a simple selection process, some results are counter-intuitive.

In the first decision, we can see that there are four different strategies: high-degree, medium-degree, low-degree, random.[2]

High-degree selection is to choose the node with the highest node degree. Similarly, medium-degree and low-degree selections are based on node degree being medium/lowest among all nodes. Random selection is just randomly picking one node. The second decision again can be made with four different strategies: high-degree, medium-degree, low-degree, random, among all edges from the chosen node.

Combining these two decisions, we would have 16 different strategies. Two examples are high-high and low-low selections. In the high-high selection, we choose the node with highest degree and then sort all neighbors of the node based on the neighbors' degrees from high to low. Edges will be chosen from the list in the same order. In the low-low selection, the selection is basically the opposite. The node with the lowest degree will be chosen first. Then all neighbors of the node will be sorted based on their degrees from low to high. And cuts are performed from the sorted list.

There is another parameter that will impact computation overhead, called L. L is the number of edges to cut before sorting is performed again. Since cutting the edges will change node degrees, sorting is needed in order to ensure that all further selections are made accurately. Therefore, L is the "knob" to tune how strictly the chosen algorithm is followed. Two extreme cases are $L = 1$ and $L = \infty$. When $L = 1$, sorting is performed after every cut, rendering high overhead. When $L = \infty$, no sorting will be performed. All edges from a chosen node will be cut until K cuts are made. In fact, when L is greater than the maximum degree of all nodes, these 16 strategies collapse to 4 until the last round of edge selections as all links from a chosen node will be cut before L links are exhausted.

Note that it may seem that the removal of these edges are sequential, i.e., cutting one edge after another. In fact, these are all cut at once and we are interested in finding the best set of edges to cut so that rumor will spread slowest in the new graph.

A. Details of CDegree Cut

Suppose we are given an undirected graph G and need to find K edges to cut in order to lower the speed of rumor spread. We define two functions: $f()$ for the choice of which node's links to cut and $g()$ for the choice of which links of a chosen node to cut. The two functions can be executed according to an input strategy and the updated graph. For example, if we choose high-random cuts, function $f()$ will rank nodes' degrees from high to low and choose the highest one's index each time. Then the second highest-degree node is chosen, etc., until L links are cut, at which time the list is updated based on the new degrees. Similarly, function $g()$

[2]Note that more complicated selections are possible. For instance, it might be better to select those nodes with the largest set of m-hop neighbors, where m is a system-defined parameter, such that they would have higher impact in distributing the (mis)information. Since such methods would require much more computation overhead, we do not consider them here in this work.

in high-random cuts will select edges from the chosen node randomly.

The pseudo-code of the procedure is shown in Algorithm 1 from step 6 to 12. For clarity of discussion, we treat the undirected graph as a directed graph with one edge in each direction in our connectivity matrix. Therefore, cutting one edge in the undirected G means cutting two edges in both directions at the same time. After each cut is performed, we add 1 to the count of cuts that have been made. We apply steps 6 - 12 before resorting degrees of the updated graph unless L numbers of edges from current node are cut or K edges have been cut or all edges of the currently selected node have been cut.

input : G: an $N \times N$ symmetric matrix represents a graph with N nodes
L: number of edges to cut before re-sorting;
K: total number of edges to cut;
$schema$: choose which strategy to apply (high, medium, low, random);
output: $\mathcal{C}$: set of edges to cut

```
1  countK ← 0;
2  while countK < K do // not enough K cuts
3      countL ← 0;
4      nodeLeft ← f(G, schema);
5      connList ← g(G, schema, nodeLeft);
6      for i ← 1 to length(connList) do
7          nodeRight ← connList(i);
8          G(nodeLeft, nodeRight) ← 0,
9          G(nodeRight, nodeLeft) ← 0;
10         C ← (nodeLeft, nodeRight);
11         countK ← countK + 1;
12         countL ← countL + 1;
13         if countL == L then          // L cuts?
14             break;
15         end
16         if countK == K then          // K cuts?
17             break;
18         end
19     end
20 end
```

Algorithm 1: CDegree Cut

B. Discussions on Schema Selection

We discuss our schema selection in $f()$ and $g()$ in the following.

Intuitively, in order to reduce network connectivity quickly, the graph should be cut to be as sparse as possible. Cutting the links of the most popular nodes, i.e., the nodes with the highest node-degrees, can be a good choice, but we need to be careful of how many links should be cut. If cutting the link of two popular nodes can split the graph into two subgraphs, then cutting it would be helpful. However, online social networks are usually highly connected [11] and it is difficult to split the graph. Then randomly cutting some of the edges or even all the edges of a popular node would be useful, as it will cost all other nodes more steps to transmit information among themselves. This method is called high-random cuts. On the other hand, if we want to isolate a few nodes quickly, i.e., with a relatively small number of cuts, from the graph, cutting the links of least popular nodes can be a good choice. This is because of their small number of edges. The method is called low-low cuts.[3]

The intrinsic question is which method would be the most efficient. If we want to see how many nodes can be affected by the source rumor information in just a few steps (or delays, if we model the propagation of misinformation on each link as one unit time), high-random cuts would be a good choice because such hubs can be quickly dismantled. However, if a longer delay is allowed, low-low cuts would be more efficient. The reason is the following: with a large allowable delay, misinformation will most likely reach throughout the network except those isolated nodes. Therefore, the best solution is to isolate some nodes with low-low cuts.

IV. Performance Evaluation

In this section, we evaluate different strategies in CDegree Cut with different L values. Our evaluations are based on a subgraph of real-life Facebook snapshot, obtained from SNAP [12]. It is an undirected graph and consists of 4,039 nodes and 88,234 edges. Each node in the graph represents a user and each edge stands for a relationship between two nodes. The network diameter (maximum undirected shortest path length) of the graph is 8, the average shortest-path length (APL) value is 3.7, and AIPL value is 0.3066. The highest node degree is 1,045. The top-10 highest and lowest degrees are shown on Tables I and II. The maximum number of edges to cut is set to $K = 6,000$, about 7% of the number of edges on the graph. We use a baseline algorithm called Random Cut, which simply picks edges randomly to cut. Each experiment has been repeated 40 times with different seeds.

TABLE I
THE TOP TEN HIGHEST DEGREES OF THE ORIGINAL FACEBOOK GRAPH

Ranking	1	2	3	4	5
Degree	1045	792	755	547	347
Ranking	6	7	8	9	10
Degree	294	291	254	245	235

TABLE II
THE TOP TEN NON-ZERO LOWEST DEGREES OF THE ORIGINAL FACEBOOK GRAPH

Ranking	1	2	3	4	5
Degree	1	1	1	1	1
Ranking	6	7	8	9	10
Degree	1	1	1	1	1

Before we present our results, we first introduce performance metrics.

[3]Note that, because of the small number of neighbors, low-high, low-medium, low-low, and low-random are similar disregard of L.

A. Performance Metrics

We mainly focused on two performance metrics: Average Inverse of Shortest Path Length and Rumor Saturation Rate.

Average Inverse of Shortest Path Length (AIPL)- In graph theory, the shortest path between any two nodes is an interesting and well-investigated problem. We choose breadth-first search (BFS) to compute the shortest path from any node to other nodes in the undirected graph and add all these path lengths for all nodes in the graph. Then the sum is divided by the number of nodes. The result is usually called average shortest-path length (APL), defined as the average number of steps along the shortest paths for all possible vertex pairs on the graph [13]. However, the APL measurement cannot handle partitioned graphs, on which some pairs of nodes are infinite distance from each other [14]. For this consideration, we use the inverse of the distance, since $1/\infty$ is simply 0. The average of all such inverse path lengths is called average inverse of shortest path length (AIPL). It can be proven that the range of AIPL is (0,1), where 0 means there is no edge in the graph and 1 means that the graph is fully connected.

Rumor Saturation Rate (RSR)- AIPL gives us the mean value of the inverse lengths of the shortest paths between all possible pairs of vertices in G, but it still does not tell us how quickly (mis)information can spread. Therefore, we look at a new performance metric called rumor saturation rate (RSR), which can be obtained through experimental settings. In order to find out RSR, S number of nodes are chosen as the sources of the same misinformation (these are called "rumor sources"). In each unit time, rumor is spread from all those nodes carrying it so far to all their neighbors. Such a procedure continues until D unit times. RSR is defined as the number of nodes that have seen the rumor D unit times later divided by the total number of nodes, N. Because of the random selections, RSR needs to be measured through repeated Monte Carlo random experiments and it is a function of (S, D).

B. Experimental Results

We first investigated the effects of L. With $K = 6,000$, we tested high-high cuts for different L values and presented the results in Figure 1. As can be seen from Figure 1, AIPL decreases as K increases, first rather quickly and then the rate of AIPL decrease slowing down. Comparing the results of different L's, we observe that $L = 1$ and 20 have rather similar performance. $L = 500$ and 1,045[4] are similarly better than $L = 1$ and 20. Therefore, a large L actually will result into a better performance, with lower overhead (of re-sorting the new degrees). Similar comparison results have been observed for other schemes for both AIPL and RSR. Hence, we conclude that L can be chosen as a relatively large value such that the overhead of re-sorting is low. In the following experiments, we chose $L = 1,045$.

In Figure 2, we showed the AIPL values of different cut methods for different K values. The two random methods, random cuts and random-high cuts, ended with similar results

[4]Note that this is the maximum node degree in the graph (see Table I).

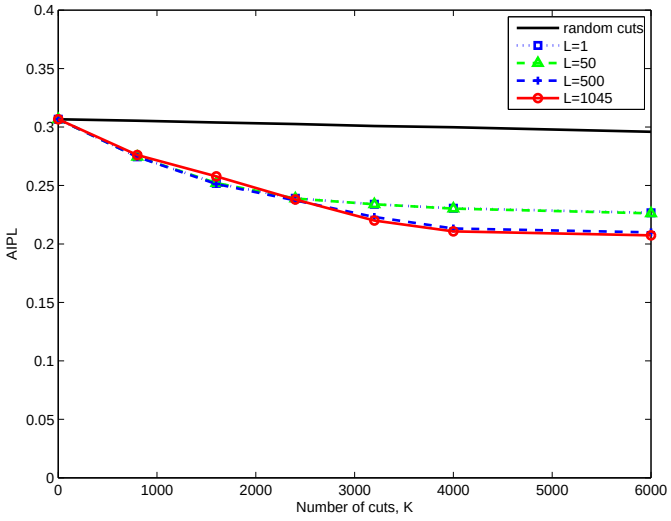

Fig. 1. AIPL with high-high cuts in different L values. Performance is better with larger L's, which also have lower overhead. Random cut results are there for comparison purpose.

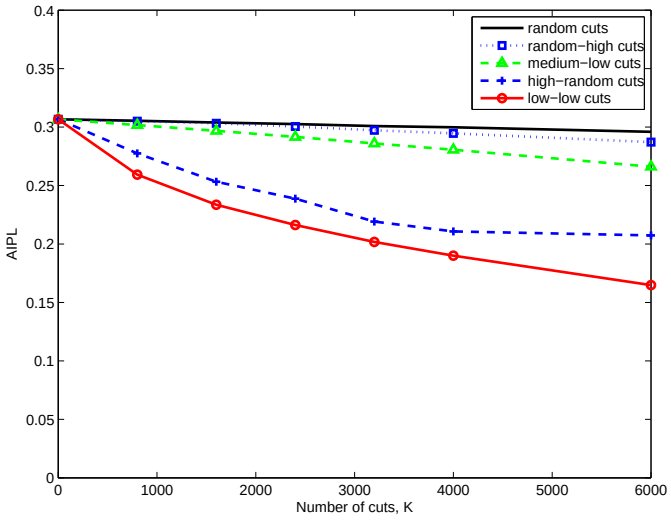

Fig. 2. AIPL comparisons of different cut methods. Low-low cuts seem to be the best method among all shown.

of very slow AIPL decrease. Medium-low cuts and high-random cuts are better, but the best performance belongs to low-low cuts. Therefore, if we were to conclude based solely on AIPL comparisons, low-low cuts would be a clear winner among the different schemes shown in Figure 2. In fact, all other schemes are outperformed by low-low cut in our results but we only showed these schemes for better clarity. Interestingly, as to be demonstrated with RSR results, such is not always the case.

Figures 3, 4, 5, and 6 demonstrated different behaviors of different cut schemes under various S and D values. Note that S represents the number of rumor sources and D is the delay before RSR is measured. We chose these D values based on the APL value of the graph. In Figures 3 and 4, i.e., when

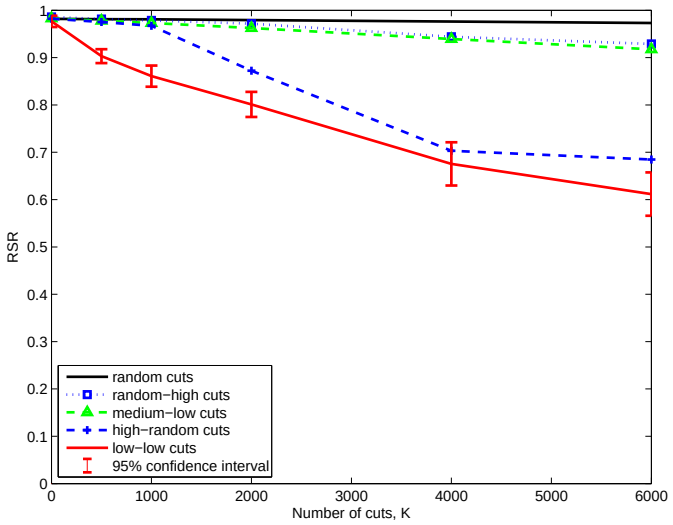

Fig. 3. RSR comparisons for (S, D) = (2, 5). S represents the number of rumor sources and D is the delay before RSR is measured. Low-low cut method is the best in this graph.

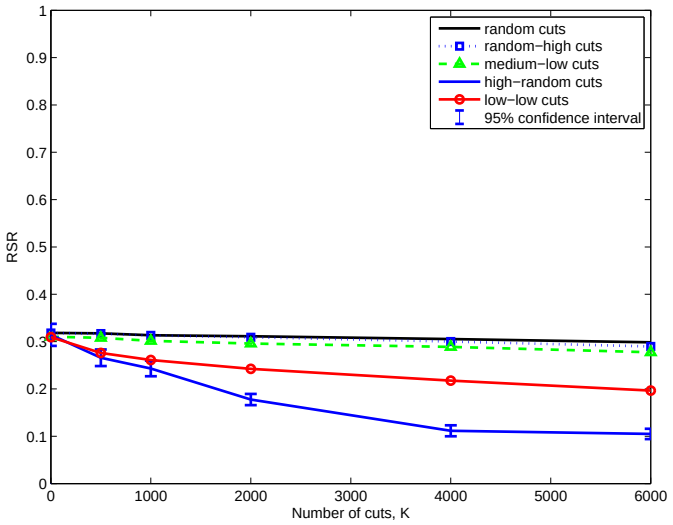

Fig. 5. RSR comparisons for (S, D) = (2, 2). S represents the number of rumor sources and D is the delay before RSR is measured. High-random cut method is the best in this graph.

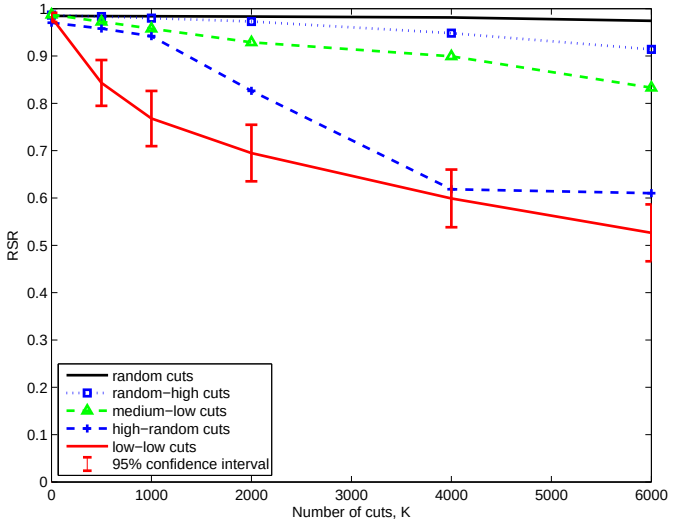

Fig. 4. RSR comparisons for (S, D) = (1, 6). S represents the number of rumor sources and D is the delay before RSR is measured. Low-low cut method is the best in this graph.

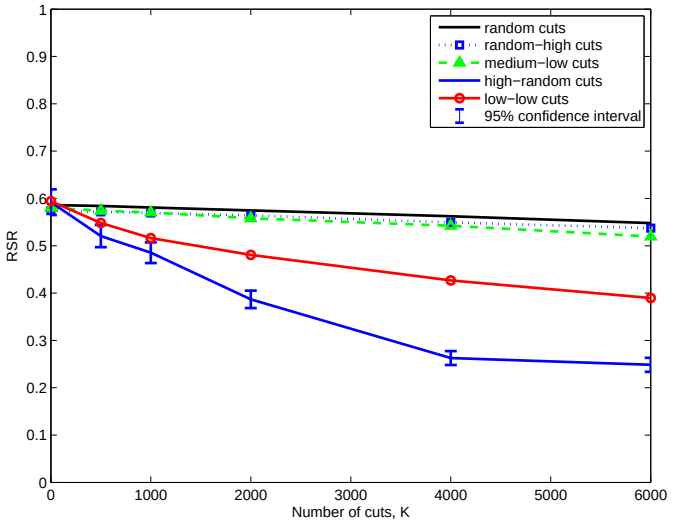

Fig. 6. RSR comparisons for (S, D) = (5, 2). S represents the number of rumor sources and D is the delay before RSR is measured. High-random cut method is the best in this graph.

D is relatively large, low-low cuts remain the best among all schemes. However, as shown in Figures 5 and 6, i.e., when D is rather small, high-random cuts outperforms other schemes. An intuitive explanation is in order: when D approaches APL, rumors will reach a majority of the nodes in the graph and it would be better to cut away some nodes so that they are isolated; However, when D is smaller than APL, it would be better to cut of some of the major connectors of the graph (those high-degree nodes) in order to lower the rumor spread rate.

Comparing the results from Figures 2, 3, 4, 5, and 6, we can see that AIPL metric has its limits in measuring network connectivity. In fact, AIPL would sometimes provide misleading indication of network connectivity. While RSR requires repeated Monte Carlo experiments, it provides interesting insights into how fast (mis)information could spread.

As we discussed, low-high, low-medium, low-low, low-random schemes work similarly because of the small number of links on such nodes. One would naturally wonder which among high-high, high-medium, high-low, and high-random cuts perform best (note that, with a large L, these schemes operate the same except the last round of edge selections). Figure 7 shows the comparison of these schemes. From Figure 7, we can conclude that these schemes are similar in performance. Considering computational overhead, high-

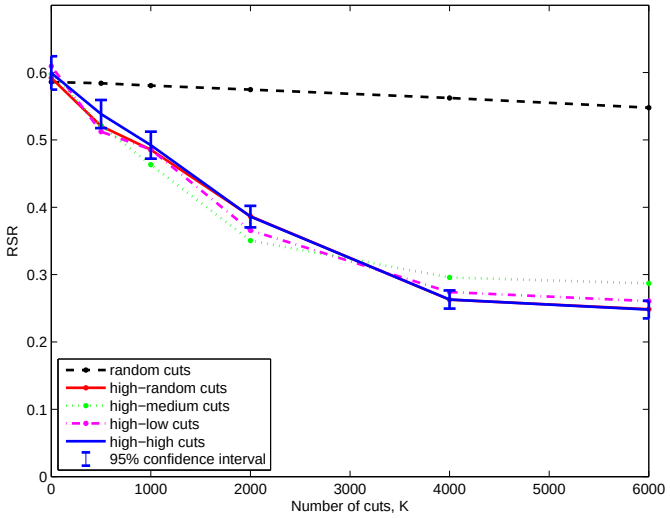

Fig. 7. RSR for (S, D) = (5, 2). High-high, high-medium, high-low, and high-random schemes are shown to have similar performance with $L = 1,045$.

random scheme works fine as no sorting is needed.

V. Conclusions

In this paper, we have investigated the problem of efficient link cuts in large online social networks to lower the speed of misinformation spread. We have designed an algorithm called CDegree Cut to choose such links. CDegree Cut has gone through extensive experimental evaluations through real-life Facebook data with two different performance metrics: AIPL and RSR. We have found that when the delay is larger than APL, low-low cuts should be used; if the delay is shorter than APL, high-random cuts should be chosen. Another interesting observation is the ambiguity of AIPL results. Instead, a more computation-intensive RSR measurement can help to provide more accurate insights in the comparison of different schemes.

In this work, we focused on the issue of link selection under the assumption that the (mis)information sources are unknown. In the scenario that such are known, the design issues would be different. A natural selection would be cutting those links around the sources within the cut budget.

While the cut algorithm designed here may not be the optimum strategy, we argue that it is among the best that only incur low overhead, which is a critical characteristics in any practical algorithm to be applied in large online social networks.

In our future work, we will develop an analytical model to study the behavior of different link cut selections. Different rumor propagation models will also be studied as well.

Acknowledgment

This work is in part supported by NSF grant CCF-1320428.

References

[1] Reinhard Diestel, *Graph Theory*, Number 173 in Graduate Texts in Mathematics. Springer, 1997.

[2] Miroslav Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[3] Lars Backstrom and Jure Leskovec, "Supervised random walks: predicting and recommending links in social networks," in *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 2011, pp. 635–644.

[4] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[5] Zan Huang, "Link prediction based on graph topology: The predictive value of generalized clustering coefficient," *Available at SSRN 1634014*, 2010.

[6] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi, "Rumor spreading in social networks," in *Automata, Languages and Programming*, pp. 375–386. Springer, 2009.

[7] Mark EJ Newman, Stephanie Forrest, and Justin Balthrop, "Email networks and the spread of computer viruses," *Physical Review E*, vol. 66, no. 3, pp. 035101, 2002.

[8] Norman P Hummon and Patrick Dereian, "Connectivity in a citation network: The development of dna theory," *Social Networks*, vol. 11, no. 1, pp. 39–63, 1989.

[9] Charles L Cartledge and Michael L Nelson, "Connectivity damage to a graph by the removal of an edge or a vertex," *arXiv preprint arXiv:1103.3075*, 2011.

[10] Y. Hayel and Quanyan Zhu, "Resilient and secure network design for cyber attack-induced cascading link failures in critical infrastructures," in *Information Sciences and Systems (CISS), 2015 49th Annual Conference on*, March 2015, pp. 1–3.

[11] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna, "Four degrees of separation," in *Proceedings of the 4th Annual ACM Web Science Conference*, New York, NY, USA, 2012, WebSci '12, pp. 33–42, ACM.

[12] Jure Leskovec and Julian J Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[13] Chia-Chen Yen, Mi-Yen Yeh, and Ming-Syan Chen, "An efficient approach to updating closeness centrality and average path length in dynamic networks," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 867–876.

[14] Jameson Watts and Kenneth W Koput, "Supple networks: Preferential attachment by diversity in nascent social graphs," *Available at SSRN 2420764*, 2014.