Ranking-free Attacks to Complex Networks

WANG Tianyu[1,2], SUN Xiaoqian[1,2*], WANDELT Sebastian[1,2*]
1. School of Electronic and Information Engineering, Beihang University, Beijing 100191, P. R. China
2. National Key Laboratory of CNS/ATM, Beijing 100191, P. R. China
E-mail: wangtianyu@buaa.edu.cn, sunxq@buaa.edu.cn, wandelt@informatik.hu-berlin.de

Abstract: Many real-world systems can be seen as complex networks, where nodes denote elements in the systems and links denote relationships between elements. Since most of the real-world systems are vulnerable under disruptions, it is necessary to perform research on network robustness. However, finding the optimal attack on networks is a NP-hard problem, thus many methods based on ranking certain network metrics are proposed, for their low computational time complexity. Although strategies can be obtained quickly according to the ranking methods, the quality of the strategies is unsatisfying, because no single metric can represent all the properties of a network. In this paper, we propose a ranking-free method to attack networks. The main idea of our method is to identify components, which are likely to be in the final network after attack. Our research targets a trade-off between expensive optimal attacks and fast ranking-based heuristics. The evaluations, both on random networks and real-world networks, show that our method behaves better than popular ranking methods.

Key Words: Complex Networks, Ranking-free Attack Strategy, Component Identification

1 Introduction

During recent years, research on complex networks has been a hot topic in many fields, such as social [1], traffic [2], biological [3] and economic [4] networks. Some researchers excavate the structure of the communities in the networks [5] to analysis the group behavior of elements, especially for the researches on social networks. The robustness of networks recently stimulated researchers interest, given that a rather small error or an intended attack may heavily damage a real-world network. For example, there are several power system blackouts, the most famous of which occurred in East America and Canada on August 28, 2003, resulting from only 4% of the power grid crashing [6]. Among the researches on the network robustness, most methods are based on ranking nodes in terms of certain aspect of property. After ranking nodes, attacking methods simply select nodes according to the importance in a decreasing order. The reason for researchers ranking nodes is that if one wants to attack a network on a fixed number of nodes, computing the optimal strategy is computationally expensive, and ranking methods can obtain a relative good strategy with usually much lower time complexity.

Among the current researches on network robustness, there are several popular methods to find the vulnerability of the network. Four of the most popular methods are degree, betweenness, closeness and pagerank. Degree is the simplest metric of networks denoting a node's number of neighbors. Betweenness of a node is the proportion of the shortest paths through this node relative to the number of total shortest paths in the network. Closeness of a node is the reciprocal of the average distance between this node and all other nodes in the network. All the details of these three methods introduced above can be seen in [7]. Pagerank is a dynamic ranking method where each node assigns its value to all its neighbors according to the neighbors' values, for details see [8].

A new method called Collective Influence (CI) was presented recently to rank importance of nodes [9]. This method defines a node's importance as $CI_l(i) = (k_i - 1)\sum_{j\in\partial Ball(i,l)}(k_j - 1)$, where i is the start node, k_i means the degree of node i and l is the distance between node i and node j. $Ball(i,l)$ means the set of all the neighbors, which are l steps away from node i. Nodes with higher CI values are more important in terms of the collective influence.

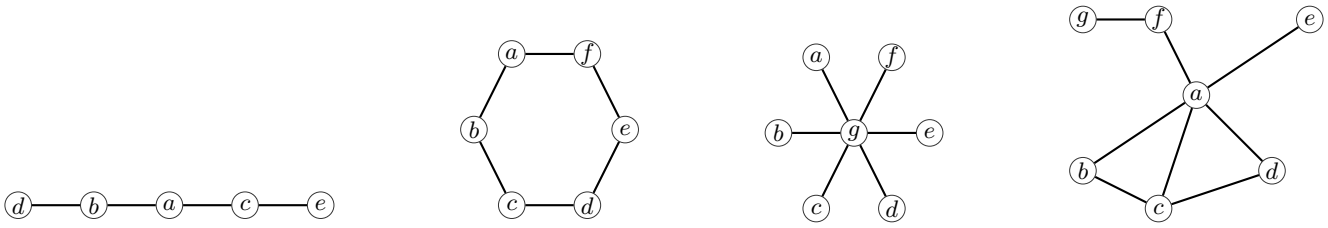
Different metrics usually induce different attacking strategies. Moreover, when the number of attacked nodes increases, shorter optimal attacks are not necessarily prefixes of longer optimal attacks. However, when we restrict ourselves to rankings, we can only have a version of importance order of nodes for one network, independently of the length of the attack. We conclude that methods based on ranking network metrics have their merits, but also several limitations when analyzing the robustness of a network.

In this light, we propose a method to detect vulnerability of the network without solely ranking the nodes according to a given network metric. We search the optimal attacks on networks by counting out some components, which are likely in the final networks after an attack of a given length. Our method exploits the fact that attack resistance in networks is usually measured by the size of giant component (GC_size). Evaluations both on random networks and real-world networks are presented in our study, comparing with popular ranking methods. The results show that our method can find better attacks than the other popular methods, at the price of having a longer running time.

This paper are organized as follows: In Section 2, we discuss the limitations of ranking-based methods and introduce our novel ranking-free method; In Section 3, we evaluate our method on three types of random networks and three real-world networks, compared with some popular ranking methods; Section 4 presents the conclusion and future work.

2 Methodology

In this section, we introduce our method. Before introducing our method, we show the limitations of ranking-based methods on four small example networks. In Figure 1, we present four standard networks: a line network, a ring network, a star network and a mesh network. Table 1 shows the importance of the nodes in each network in Figure 1 according to three popular ranking methods. For the line network, if we want to attack one node, node a is the best choice ac-

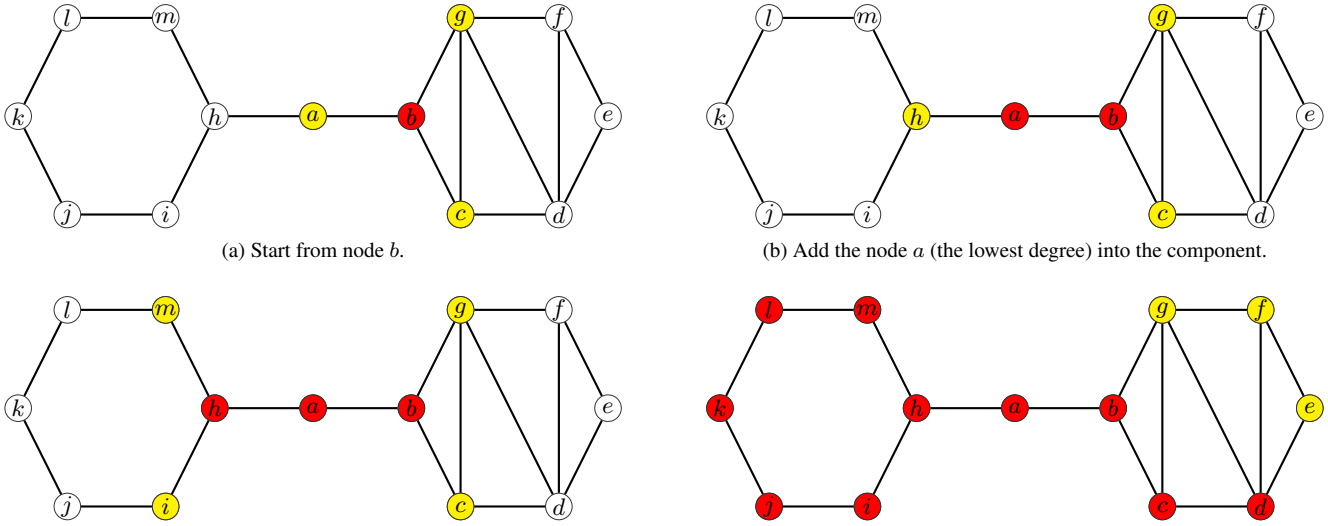


(a) Line network (b) Ring network (c) Star network (d) Mesh network

Fig. 1: Four types of example networks: line network, ring network, star network and mesh network.

Network	degree	betweenness	CI ($d = 2$)
Line network	$a = b = c > d = e$	$a > b = c > d = e$	$b = c > a = d = e$
Ring network	$a = b = c = d = e = f$	$a = b = c = d = e = f$	$a = b = c = d = e = f$
Star network	$g > a = b = c = d = e = f$	$g > a = b = c = d = e = f$	$a = b = c = d = e = f = g$
Mesh network	$a > c > b = d > f > e = g$	$a > c = f > b = d = e = g$	$f > b = c = d > a = e = g$

Table 1: Importance of each nodes in example networks calculated by 3 popular methods.



(a) Start from node b. (b) Add the node a (the lowest degree) into the component.

(c) When there are more than one nodes having the lowest degree, we choose one from them according the order recorded in the method (here it is node i). (d) When the set union of the component and its direct neighbors covers the whole network, the search for the given initial node (here: b) stops.

Fig. 2: Four typical statuses of example attack on a network while counting out components. The red nodes means those in the component; the yellow ones are the component's neighbors (candidate choices).

cording to betweenness, but it is not the best one based on CI ($d = 2$). In the ring network, all nodes in the network are equally important no matter which method we use, since the nodes cannot be distinguished based on graph-theoretic properties. As for the star network, if we want to remove one node, we cannot find difference on the basis of CI ($d = 2$) method, since every node has the same importance in the network. When it comes to mesh network, node a and node c are the top two nodes according to degree and betweenness methods, but they have the lowest status in terms of CI ($d = 2$). Therefore, if we want to remove one node, we will choose node a according to the first two methods, but we would choose node f based on CI ($d = 2$). And if the number of attacked nodes is two, degree and betweenness' choice should be node a and node c, while CI ($d = 2$) will choose node f and node b, which is absolutely different from that of those two methods. Another thing we should note is that the theoretically best choices for a network are always not a super-set of less-number attacks when the number of attacked nodes varies. However, if we want to attack four nodes totally, the choice obtained according one ranking method must be a combination of that of three attacked nodes and one new nodes in the network, for the reason that once a network is given, the importance of each nodes in the network is fixed after executing an order-based prefix of an attack.

Considering the limitations of ranking-based methods on attacking networks, we design a method to attack networks without explicitly exploiting rankings of node importance only. Inspired by the measurement criteria of attacks on a network, size of the giant component, we intend to find sets of nodes that can divide the network into specific components. When the number of attacked nodes varies, we search different sets of nodes for the final attack choice. This makes our method's attack strategy different from ranking-based techniques.

In Figure 2, we present an example for our method to count out components. Let us assume, we want to identify interesting components, starting from a random node; here we use b. In the Figure 2 (a), if we start from node b, we put

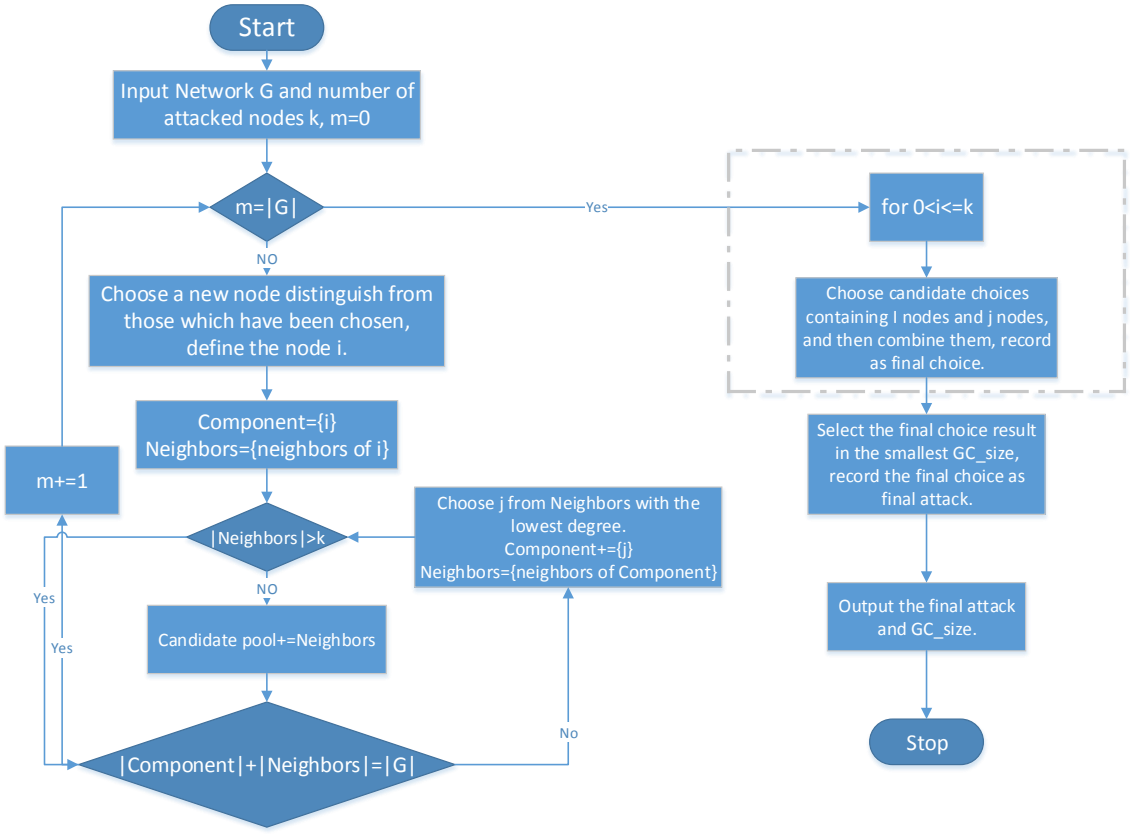


Fig. 3: Flow chart of our method. The left side shows the first step of our method; The right side shows the second step. The dashed frame denotes the scope of the for loop, and when they are finished, the flow chart goes on.

it into a set that means a component (filled in red). Then we make another set to hold the component's neighbors (filled in yellow). After we store the information of the component and its neighbors, we choose one neighbor with the lowest degree from all the neighbors (in this figure, it is node a). The intuition is that this node in the network is particular sensitive to an attack. The situation is depicted in Figure 2 (b). After we add node a into the component, the component's neighbors change. Then we record the new information of the current component and its neighbors. When there is more than one node having the lowest degree, as shown in Figure 2 (c), we deterministically choose the node with a smaller index. We repeat the steps above until the union of the component and its neighbor nodes cover the whole network, a situation which can be seen in Figure 2 (d).

Each set of neighbors, discovered during the previously described search phase, stands for a candidate choice. Cutting out the neighbors will yield the components discovered up to that point. Thus far, we have constructed a set of candidate choices from one (random) node, and each candidate choice contains one or more nodes. We repeat all the steps above for each node in the network. After we have constructed all the candidate choices, we need to combine some choices to be a final attack strategy. If we want attack k nodes in total, we should consider all the candidate choices, which consist of less than or equal to k nodes. Then we choose appropriate subattacks according to the decomposition of k. For instance, decomposition of k could be $k = k$, $k = (k-1)+1$, $k = (k-2)+2$, $k = (k-2)+1+1$ etc; we potentially need to consider many combinations. To simplify this process, and investigate the potential of our technique, we only consider to divide k into 2 parts in our method, which can significantly decrease the computational complexity. Our experiments below show that this simplification already helps to identify significantly better attacks than ranking-based methods.

Overall, our method has two steps which are explained in detail below:

1) **Make a candidate choice pool:** We first construct a set named candidate choice pool to hold all the sets of components' neighbors, whose quantities are less than or equal to k. The elements of the candidate choice pool are used to combine the final attack, which contains k nodes. From a start node, we add the node into an empty set named component, then we have a component consisting of only one node. Besides, we record the neighbors of the component (node) into a set, and put the set into the candidate choice pool. It is obvious that if we remove the set of nodes in the candidate choice pool, we can at least obtain the component we got beforehand. Next, we choose the neighbor with the lowest degree from the component's neighbors and put it into the component set. After we extend the component, we recheck the new component's neighbors, and put them into a new set, add it into the candidate choice pool. The same as above, if we remove nodes in the new component's neighbors, we can at least obtain the new component. The reason why we choose the neigh-

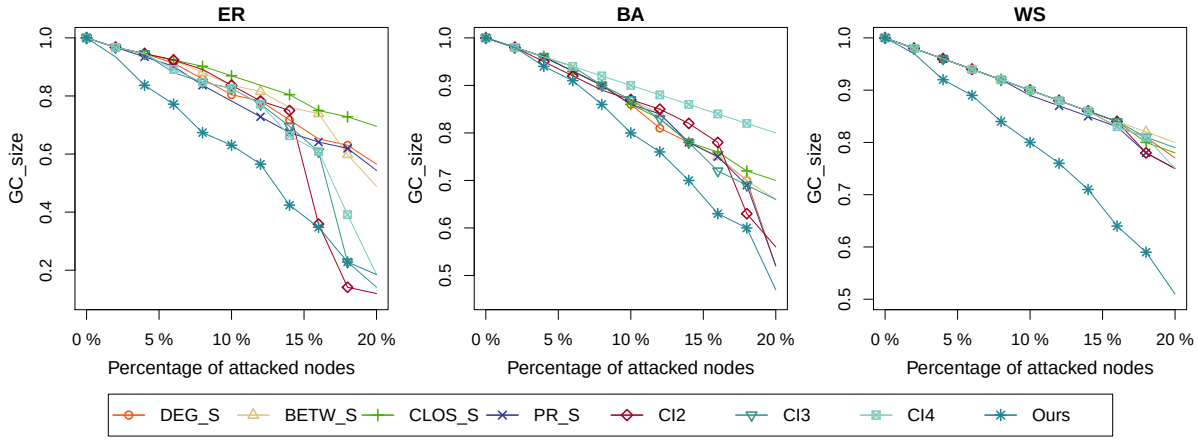


Fig. 4: The size of giant component (GC_size) of three random networks calculated by our method and others' after first 20% attack; ER stands for Erdös-Renyi network; BA represents Barabasi-Albert network; WS means Watts-Strogatz network.

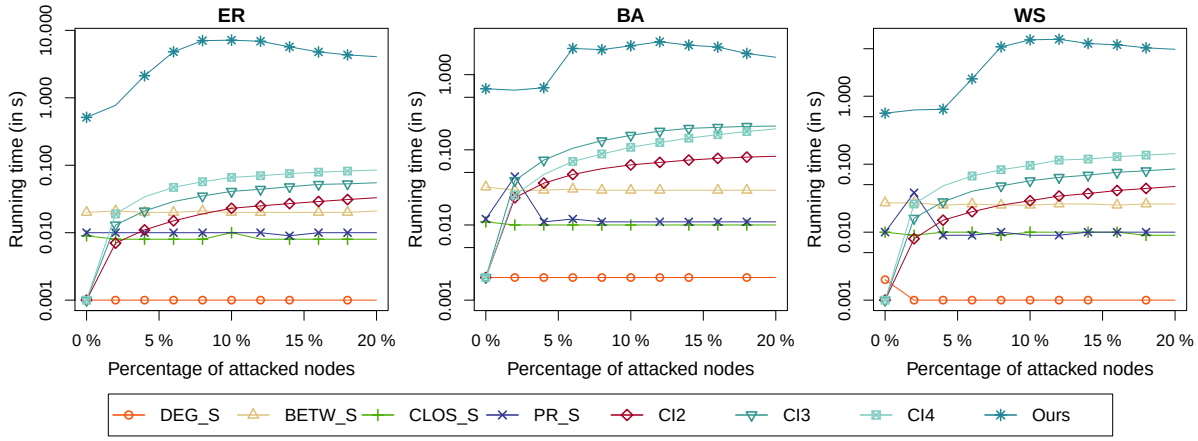


Fig. 5: Running time for 3 random networks in log scale calculated by our method and others' after first 20% attack; ER stands for Erdös-Renyi network; BA represents Barabasi-Albert network; WS means Watts-Strogatz network.

bor with the lowest degree here is that though ranking methods behave poorly in attacking networks, we believe this kind of nodes are more likely to be in the final network after attack. We repeat all the procedures of the start node on each node in the network.

2) **Combine some candidate choices into a final attack strategy:** After step 1, we have got a candidate choice pool consisting of all the candidate choices we have detected. We choose those choices containing less than or equal to k nodes. Since considering all the decomposition of k is computationally hard, we only concern to divide k into two parts in our method. For each addend, we choose a candidate choice consisting of the same number of nodes, and combine them. Every combination could be a final attack strategy, and then we record GC_size after attack of each strategy. The best strategy is that with the lowest GC_size.

To visualize our method, we present the flow chart in Figure 3. The left side of the flow chart represents the first step, and the right side corresponds to the second step.

3 Evaluation

In this section, we will present our method's effectiveness on three types of random networks and three different real-world networks, compared with ranking-based techniques. Both, the size of the giant component (GC-size) and running time, are shown in the evaluations, as the percentage of attacked nodes increases from 0% to 20%; thus the focus of our evaluation is on finding good initial attacks, which is what network scientists are usually interested in. We perform all experiments on a laptop with Intel(R) Core(TM) i7-4610M CPU @ 3.00GHz processor and 4.00 GB main memory in Python 3.4.1 on Fedora 23.

3.1 Evaluation on random networks

In this part, we evaluate our method on three types of random networks with 100 nodes. Before presenting the results of the experiments, we will introduce the properties of these random networks first.

1) Erdös-Renyi (ER) network. To obtain an ER network, 2 parameters are needed, including the number of the total nodes n and the probability to make a new link p. The probability of ER networks means that for each pair of nodes in the network, a link between them exist at the possibility of p. Therefore, in the ER network, there are approximately $p * C_n^2$ links. A larger p stands for better robustness of the network.

2) Barabasi-Albert (BA) networks. In this type of random network, there are also 2 variables to get a specific

Network	n	other parameters
Erdös-Renyi network	100	$p = 0.03$
Barabasi-Albert network	100	$l = 3$
Watts-Strogatz network	100	$k = 5, p = 0.3$

Table 2: Parameter settings for the 3 random networks used in our study.

network, containing the number of nodes n and the number of each new node's links l. In this model of networks, when we add a new node to an initial network, the node choose l nodes in the network to make links, and the l nodes are selected with a linear preference of their degree (nodes with more degree are more likely to be linked). Following this rule, BA networks have power-law degree distributions. Similar to ER networks, larger l means larger density.

3) Watts-Strogatz (WS) network. Different from the first 2 types of networks, WS networks have 3 parameters: the number of nodes n, the initial number of each node's degree k and the probability of each node rewiring p. WS networks with larger k have better robustness, which is similar to BA networks. Different p can lead to different degree distributions.

Based on all above and consideration of the networks' robustness, we set parameters of each networks as seen in Table 2.

In Figure 4 we show the relative size of the giant component after attacking a fraction between 0% and 20% of the network. We find that our method's results are mostly better than ranking-based methods'. In ER network, our method's result is better than the others' in first 16%, and when the percentage of attacked nodes come to 18% and 20%, CI ($d = 2$) is a bit better than ours. We can see that when the percentage is 15%, the results of all other methods can only make the GC_size about 70%, while our method can lead GC_size to be about 40%. When it comes to BA network and WS network, our method behaves better than all other methods for the first 20% attack. For BA network, CI ($d = 4$) behaves worst among all the 8 methods, and the reason may be the small number of the nodes in the network. When the size of the network goes up, CI ($d = 4$) may be much better. Among the other 6 methods, there is no one method has absolute advantage than the others for the first 20% attack. As for the WS network, our method behaves much better than the others. For the current settings, all other methods can only decrease the GC_size to about 80%, among which 75% is the best result, while our method can make GC_size 51%. Taking all above into account, our method can attack networks more seriously when percentage of attacked nodes is not too large.

In Figure 5, all methods' running time is presented. The running time of our method is longer than the other methods'. However, considering the good performance and the small-scale of the total time, our method is worth to being used when networks are attacked. Overall, this study presents initial results and shows that careful design of an attacking strategy outperforms ranking-based methods. The running time is the major limitation, which should be addressed in the future.

3.2 Evaluation on real-world networks

Network	nodes	links	average degree	density
dolphins	62	159	5.129	0.041
football	115	613	10.661	0.046
karate	34	78	4.588	0.067

Table 3: Basic statistics of 3 real-world networks in our study, including number of nodes, number of links, average degree and density.

To evaluate the effectiveness of our method, we apply our method to 3 real-world networks. Table 3 shows the basic statistics of these networks including their number of nodes, number of links, average degree and density. In each network, we attack nodes from 0% to 20% of the total nodes, and compare the GC_size as well as the running time. Other 7 methods we use in the random network evaluation are also used in this part.

In Figure 6, we present the variety of GC_size as the percentage of attacked nodes increase from 0% to 20% in 3 real-world networks. For dolphins network, our method behaves better than all the other methods except CI ($d = 3$) is better than ours on 3 points. Our method can make the GC_size 29%, while most of other methods can only decrease the GC_size to about 70%. The results for football networks are much more interesting. All other methods can only make GC_size 80% when the percentage of attacked nodes is 20%, while our method can lead to a component containing 74% of the total nodes. Our method's good result on this network can also show the perfect performance of our method on networks with good robustness. As for karate network, during the first 20% attack, our method's result is always better than the others', while CI ($d = 3$) and CI ($d = 4$) behaves badly in this network. When 20% of nodes are attacked, our method can result in a component containing 14% of nodes.

In Figure 7, the running time of our method in these 3 real-world networks is similar to that of the random networks. Our method needs more time than the others. It is interesting for the football network, our method's running time is close to that of CI ($d = 3$) and CI ($d = 4$) when percentage of attacked nodes is about 15%, and goes up afterwards sharply. It is considerable that even when the percentage of attacked nodes is pretty small, our method also needs a lot of time compared with other methods. This would be solved in future work.

4 Conclusions and Future work

In this paper, we propose a ranking-free network attack method. Originally, we locate the weak points of a network by counting out components instead of attacking according to the order of nodes' importance calculated through a certain network metric. We evaluate our method on 3 types of random networks and 3 real-world networks in terms of the size of giant component and running time, while varying the percentage of attacked nodes between 0% and 20%. Our method, compared to 7 ranking-based techniques, behaves well in all the networks in terms of the size of giant component, while there may be other methods performing better than ours on only a few points. As for the running time, our method needs more time than the other popular methods. Overall, our method can provide a more effective strategy to attack networks at the cost of increased running

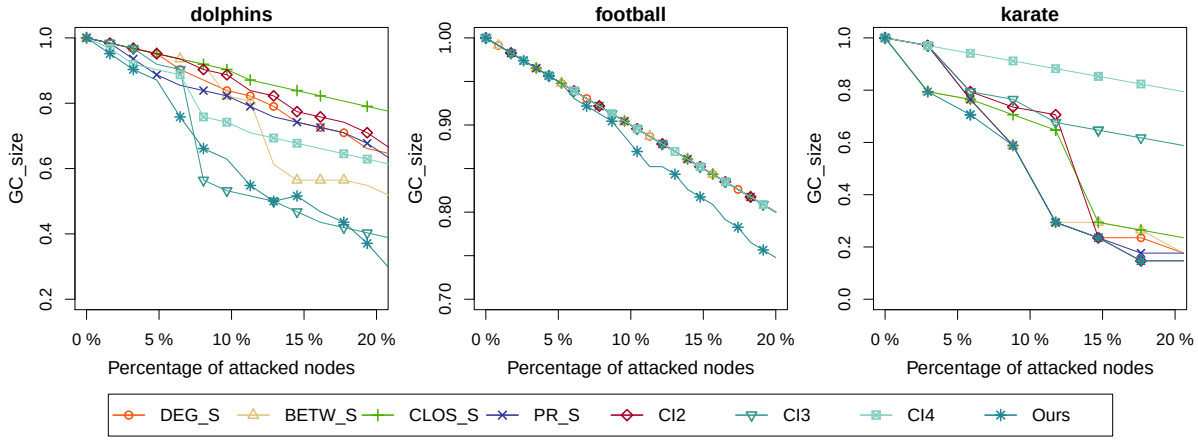


Fig. 6: The size of giant component (GC_size) of 3 real-world networks calculated by our method and 7 others for percentage of attacked nodes from 0% to 20%; ER stands for Erdös-Renyi network; BA represents Barabasi-Albert network; WS means Watts-Strogatz network.

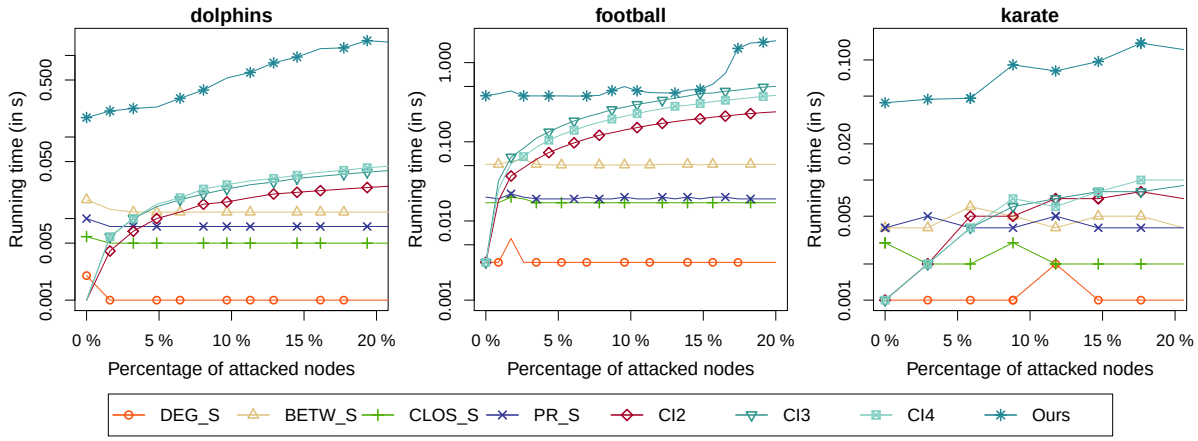


Fig. 7: Running time for 3 real-world networks in log scale calculated by our method and 7 others for percentage of attacked nodes from 0% to 20%; ER stands for Erdös-Renyi network; BA represents Barabasi-Albert network; WS means Watts-Strogatz network.

time. We would like to underline that although our method's running time is longer than the other popular ranking methods', it is much shorter than that of the searching process of the optimal strategy.

In future work, we would improve our method regarding several aspects. First, we could count out components more efficiently as of now. In the current method, we count out components in the same process on every node in the network, but a huge number of the same components would be recorded; Second, one should decrease the number of the candidate choices. One option is to delete some candidate choices that are impossible in the final network after attack; Last but not least, we should decide how many parts k should be divided into intelligently; instead of fixing the number as in this preliminary evaluation.

References

[1] P. Walker, S. Fooshee, and I. Davidson, "Complex interactions in social and event network analysis," in *Social Computing, Behavioral-Cultural Modeling, and Prediction*, ser. Lecture Notes in Computer Science, N. Agarwal, K. Xu, and N. Osgood, Eds. Springer International Publishing, 2015, vol. 9021, pp. 440–445. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-16268-3_56

[2] X. Sun and S. Wandelt, "Network similarity analysis of air navigation route systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 70, pp. 416–434, 2014.

[3] S. Sinha, "From network structure to dynamics and back again: Relating dynamical stability and connection topology in biological complex systems," in *Dynamics On and Of Complex Networks*, ser. Modeling and Simulation in Science, Engineering and Technology, N. Ganguly, A. Deutsch, and A. Mukherjee, Eds. Birkhuser Boston, 2009, pp. 3–17. [Online]. Available: http://dx.doi.org/10.1007/978-0-8176-4751-3_1

[4] E. Estrada and N. Hatano, "Communicability and communities in complex socio-economic networks," in *Econophysics Approaches to Large-Scale Business Data and Financial Crisis*, M. Takayasu, T. Watanabe, and H. Takayasu, Eds. Springer Japan, 2010, pp. 271–288. [Online]. Available: http://dx.doi.org/10.1007/978-4-431-53853-0_14

[5] R. Guimera, S. Mossa, A. Turtschi, and L. A. N. Amaral, "The worldwide air transportation network: Anomalous centrality, community structure, and cities global roles," *PNAS*, vol. 102, no. 22, pp. 7794–7799, 2005.

[6] S. Mei, X. Zhang, and M. Cao, "Foundation of soc in power systems," in *Power Grid Complexity*. Springer Berlin Heidelberg, 2011, pp. 95–132. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16211-4_3

[7] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks: Structure and dynamics," *Physics Reports*, vol. 424, no. 4-5, pp. 175–308, Feb. 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037015730500462X

[8] S. Weinberg, "A model of leptons," *Physical review letters*, vol. 19, no. 21, p. 1264, 1967.

[9] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, 2015.