The robustness of sparse network under limited attack capacity*

Xiao-Juan Wang(王小娟)[1], Mei Song(宋梅)[1,†], Lei Jin(金磊)[1], and Zhen Wang(王珍)[2]

[1] *Electronic Engineering Institute, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*

[2] *Electronic Engineering Institute of PLA, Hefei* 230000, *China*

(Received 13 January 2017; revised manuscript received 21 April 2017; published online 10 July 2017)

The paper studies the robustness of the network in terms of the network structure. We define a strongly dominated relation between nodes and then we use the relation to merge the network. Based on that, we design a dominated clustering algorithm aiming at finding the critical nodes in the network. Furthermore, this merging process is lossless which means the original structure of the network is kept. In order to realize the visulization of the network, we also apply the lossy consolidation to the network based on detection of the community structures. Simulation results show that compared with six existed centrality algorithms, our algorithm performs better when the attack capacity is limited. The simulations also illustrate our algorithm does better in assortative scale-free networks.

Keywords: robustness, dominated relation, merge, lossless

PACS: 89.75.Ck, 89.75.Fb **DOI:** 10.1088/1674-1056/26/8/088901

1. Introduction

A large number of complex systems can be represented as networks. The research scope of the complex network includes biology,[1,2] social network,[3,4] communication,[5,6] and so on. The robustness of the network is significative to be focused on. It has been proved that the large-scale real network is robust to the random attack while it shows vulnerability to the targeted attack. Most real networks are sparse networks. Sparse networks[7–10] are those with small number of short loops and edges. So, in complex network, we focus on the researches of sparse networks.

Generally, the network will be broken after deleting a part of nodes chosen by centrality algorithms. The existing centrality algorithms[11,12] include the degree algorithm,[13] the betweenness algorithm,[14] the closeness algorithm,[15] the k-core algorithm,[16] and so on. In real networks, the number of nodes in the network is large, but the attack capacity is limited, that is to say, we can just attack a small part of nodes in the network. Under this occasion, deleting the nodes chosen by the algorithms above has no big influence on the network, which means the existing centrality algorithms perform not so well when the attack capacity is limited. So in this paper we tend to design an effective centrality algorithm to figure out the critical nodes when the attack capacity is limited.

The process of deleting nodes can be seen as the process of percolation in the network. Latora[17] studied the vulnerability and protection of infrastructure networks in which he put forward a framework to define critical damages, critical improvements and structural vulnerability. Karrer[18] reformulated percolation as a message passing process and gave effective way to calculate the size of the percolating cluster and the average cluster size in sparse network. Agliari[19] analysed bond percolation process on correlated random network and showed different evolution of the network between stochastic and deterministic process. Azimitafreshi[20] studied k-core percolation in multilayer networks and presented an analytical framework which enables us to describe the nature of the transitions. Liu[21] figured out influential nodes via SIR model and used dynamic centrality to measure the importance of the nodes. Tian[22] focused on phase transition during percolation. The explosive percolation is found in finite networks. However, the explosive percolation will not appear after limited nodes deleted. Therefore, the existing centrality algorithms are not effective when the attack capacity is low.

In this paper, we tend to pay attention on the network structure and design a novel algorithm to measure the centrality of nodes. There are vast previous researches on network structure. Kanevsky[23] gave fast algorithm to find out minimum cut vertex set in the network from which we can merge the network iteratively. Rosvall[24] used maps of information flow to detect the community structure in the network. We can use the community structure to merge the network and the nodes which have swallowed many other nodes are considered to be important to the network. Wang[25] detected the community structure using nonnegative matrix factorization which has powerful interpretability and close relationship between clustering methods. Singh[26] found community structure in sparse network using random walks with reluctant backtracking operators. Liu[27] defined three properties of community and proposed a simple and novel framework to detect community by network topology. However, if we merge the network by just using the exiting community detection algorithm, the

*Project supported by the National Natural Science Foundation of China (Grant No. 61471055).

†Corresponding author. E-mail: songmbupt@126.com

© 2017 Chinese Physical Society and IOP Publishing Ltd

critical nodes perform poorly during the process of attack.

In this research we define a dominated relation between nodes based on the cut vertex in the network. Then we merge the network using the strongly dominated relation and we call this process as the lossless consolidation. We design a dominated clustering algorithm to figure out the critical nodes by the weights which can be obtained by the merging process. Simulation results show the effectiveness of our algorithm in sparse real networks when the attack capacity is limited. Our algorithm performs better in assortative scale-free network compared with other centrality algorithms.

The rest of this paper is arranged as follows. Section 2 gives the indicators for the network vulnerability and shows the limitation of the existing algorithms. We define the strongly dominated relation and put forward the dominated clustering algorithm in Section 3. In Section 4, the simulation result of deleting nodes is shown. Section 5 concludes the paper.

2. Topology vulnerability

2.1. Indicators for vulnerability

The indicators for measuring the vulnerability of networks are abundant. This paper uses the Largest Connected Component (LCC) to evaluate the communication capability of the network currently, in other words, the robustness of networks. Furthermore, we use the number of components (the isolated nodes are excluded) and the scale of small components to evaluate the broken degree of the network. Then we can write down the definition of vulnerability indicators as follows:

(i) The scale of the LCC

$$S = \{n_j | \text{iff} \exists n_i \in S \Lambda e_{ij} = 1\}. \tag{1}$$

Here we use S to represent the number of nodes in the LCC of the network. n_i and n_j represent node i and node j. e_{ij} is the edge between the node i and node j. The LCC is the carrier of the network traffic. Therefore, when the network is under attack, we can use the number of nodes in the LCC to evaluate the overall connected situation of the network. S is also an import indicator to indicate whether the network is failed.

(ii) The number of connected components

$$C = \{C_j | \exists n_1, \ldots, n_k \in C_j, k > 1\}. \tag{2}$$

After deleting several nodes in the network, the connected components will appear in the network. Considering the exist of the isolated nodes will impact the study results, we exclude the isolated nodes from the connected components.

(iii) The mean scale of small connected components

$$M = \frac{\text{Sum}(C_j)}{N(C_j)}, \quad C_j \in C. \tag{3}$$

Small connected components are the set of connected components except the LCC. Here we use $\text{Sum}(C_j)$ to represent the number of nodes in small connected components. Then $N(C_j)$ is the number of small connected components.

2.2. The limitation of existing algorithms

Generally, the way to find out the critical nodes is to use node centrality algorithms. Different centrality indicators measure the nodes value from different angles. If we attack the network with different centrality algorithms, it will show different attack performances. Here we give some existing node centrality indicators. First, we introduce the degree centrality which is known as the most common used indicator.[13]

$$C_{\text{d}}(i) = k(i). \tag{4}$$

Here i represents the node i in the network. Degree centrality indicates the value of the node by using the number of edges connected to it. Then the betweenness centrality[28] which calculates the number of the shortest paths which pass the node is shown as

$$C_{\text{b}}(i) = \frac{2\sum_{s<t} g_{st,t}/n_{st}}{n(n-1)}, \tag{5}$$

where $g_{st,i}$ is the number of the shortest paths from node s to node t passing node i. n_{st} is the total number of the shortest paths from node s to node t. n is the total number of nodes. After that we introduce the closeness centrality which measures the distance of the node to other nodes in the network. The value of the closeness centrality[11] shows the broadcasting time length of information in the network.

$$C_{\text{c}}(i) = \sum_{s<t} d_{st}, \tag{6}$$

where d_{st} represents the distance from node s to node t. Eigenvector centrality[29] uses the eigenvectors of the adjacent matrix to indicate the importance of nodes

$$C_{\text{e}}(i) = \lambda^{-1} \sum_{j=1}^{n} a_{ij} e_j, \tag{7}$$

where A is the adjacent matrix. If there is an edge between node i and node j, we set $a_{ij} = 1$. Otherwise, $a_{ij} = 0$. Besides $(\lambda_1, \lambda_2, \ldots, \lambda_n)$ are eigenvalues of A, and the corresponding eigenvectors are $(e_1, e_2, \ldots, e_n)$. We also use Pagerank centrality[30] to indicate the node's value which is defined as

$$C_{\text{p}}(i) = \frac{1-q}{n} - q\sum_{j} \frac{C_{\text{p}}(j)}{k(j)}. \tag{8}$$

Here $k(j)$ is the number of adjacent nodes of node j. q is the transition probability. What is more, the k-core centrality is an important indicator, which deletes the nodes of degree k and sets the centrality of the deleted nodes as k ($k = 1, 2, 3, \ldots$). In this paper, we use $C_k(i)$ to represent k-core centrality[12] of the node i.

However, the existing centrality algorithms do not always perform well, especially under the limited attack capacity. Figure 1 shows the 20 most important nodes of different centrality algorithms in the Router network. From Fig. 1 we can find that different centrality algorithms select different important nodes, but there are also some nodes chosen by two or more algorithms. It is shown that the centrality algorithms measure the node importance from various aspects. However, the simulations will show the shortness of the existing centrality algorithms in the following part. In the percolation process of the network, we delete the nodes according to the order of the centrality. In simulations, we observed that the phase transition will not appear until at least 5 percents of the nodes are deleted. While deleting 5 percents of the nodes in the real networks may exceed the attack capacity. Furthermore, the nodes centrality just measure the importance of the single node, ignoring the linkage effect between nodes. Thus we need to design an algorithm which considers the connection between nodes and indicates the importance of the nodes as well.

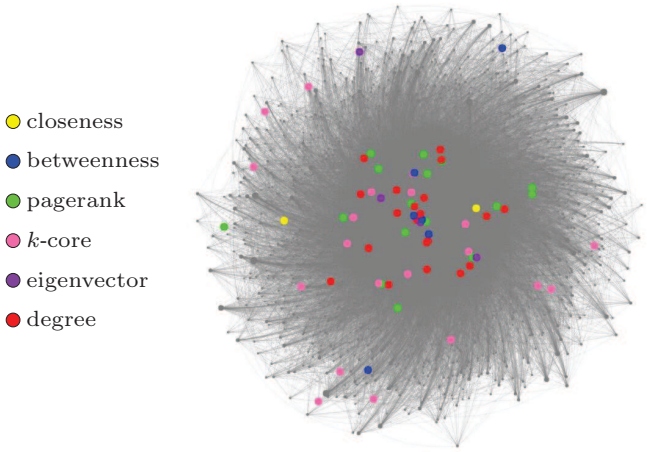


Fig. 1. (color online) The visualization of nodes selected by different algorithms.

2.3. Network dataset

In order to show the different performances of the centrality algorithms, we do simulations on the real networks and the Barabasi Albert network with different assortativities. Table 1 and 2 give networks[31–36] properties in our simulations.

Table 1. The real networks.

Networks	n	m	$\langle k\rangle$	ρ	C	$k_{\max}$	$k_{\min}$	D
Router	23441	32049	2.73	−0.18484	0.00152	123	1	0.000117
Yeast	2361	6646	5.63	−0.09910	0.13006	64	1	0.002386
Erdos	6927	11850	3.42	−0.11259	0.12390	507	1	0.000494
Powergrid	4941	6594	2.67	0.00347	0.08010	19	1	0.000540
Geom	9343	11898	2.55	0.24259	0.32024	102	0	0.000273
Polblogs	1490	16715	22.44	−0.22123	0.26265	351	0	0.015067
Roget	1022	3648	7.14	0.17912	0.14985	28	0	0.006992

Table 2. The artificial networks.

Networks	n	m	$\langle k\rangle$	ρ	C	$k_{\max}$	$k_{\min}$	D
BA1	1000	1996	3.992	−0.3	0.00802	55	2	0.003996
BA2	1000	1996	3.992	−0.1	0.01970	55	2	0.003996
BA3	1000	1996	3.992	0	0.01353	55	2	0.003996
BA4	1000	1996	3.992	0.1	0.01450	55	2	0.003996
BA5	1000	1996	3.992	0.3	0.01617	55	2	0.003996
BA6	1000	3984	7.968	−0.3	0.00028	131	4	0.007976
BA7	1000	3984	7.968	−0.1	0.04067	131	4	0.007976
BA8	1000	3984	7.968	0	0.03377	131	4	0.007976
BA9	1000	3984	7.968	0.1	0.03735	131	4	0.007976
BA10	1000	3984	7.968	0.3	0.11466	131	4	0.007976

Here n is the number of nodes in the network. m is the number of edges, $\langle k\rangle$ is the mean degree, and ρ is assortativity coefficient. C represents clustering coefficient. $k_{\max}$ is the largest node degree and $k_{\min}$ is the smallest node degree in the network. D measures the density of the network which is defined as

$$D = m/\binom{2}{n}. \tag{9}$$

We can find that for sparse network, D is very small. The density of networks relative. If the network is fully connected ($D = 1$), our algorithm is disabled. However, at this time, it is useless to analyze the robustness of centrality. For most networks, D is small. These networks are sparse network.

We also compare performance in artificial network. BA networks are scale-free networks. Initially we have m_0 nodes fully connected. At each time step, we introduce a node with m ($m < m_0$) edges, and the m edges are connected to other nodes with probability $k_i/\sum_j k_j$. After that we can use structural algorithm to change the assortativity of the network. Assortativity[37] coefficient is defined as

$$\rho = \frac{1}{\sigma_q^2}\left[\sum_{e_{ij}} k_i k_j - \mu_q^2\right], \tag{10}$$

where μ_q is the sum of degree and σ_q is the variance of degree. For structural algorithm, we randomly choose two links $a \sim b$, $c \sim d$ at each time. If $k_a k_c + k_b k_d > k_a k_b + k_c k_d$, we change the connections to $a \sim c$, $b \sim d$. Obviously in this algorithm, the degree distribution does not change. By repeating this process, we can get different networks with different ρ.

3. Network mergence based on dominated relation

3.1. The dominated relation

In this paper, we tend to analyze the network using a new relation between nodes which is called as the dominated relation. For an undirected network $G(V,E)$, we use V to represent

the vertex set and E to represent the edge set. To fully understand the dominated relation between nodes, we give definitions below.

Definition 1 An undirected network is k-cut, if the network is unconnected after at least removing k nodes.

Lemma 1 If there are nodes whose degree is 1, then the network is 1-cut.

Definition 2 If the network is k-cut, then the minimum cut of the network is the set of k nodes, and if the k nodes are deleted the network is not fully connected.

Lemma 2 The minimum cut of the network is not unique.

In the 1-cut network, node a is one of the nodes whose degree is 1. Node b is connected with a. From the theories above, we can define node a is dominated by node b, in other words, there is a dominated relation between nodes a and b. We also define node b as the minimum cut of the network. Then we can merge the dominated nodes whose degree is one. This process can be repeated several times until there is no node whose degree is 1.

Then we focus on the situation when the network is 2-cut. In this situation we can say the nodes which connect only to the minimum cut are dominated by the minimum cut of the network. However, this relation is not reliable for deciding the critical vertex. In this paper we define a strongly dominated relation between nodes. Suppose one minimum cut of the network is (a,b), the dominated nodes of (a,b) are $(c_1,c_2,\ldots,c_k)$. We use $p(a)$, $p(b)$, $p(c_i)$ to represent the profit of just deleting one node a, b or c. (The profit of nodes represents the degree of damage to the network) $p(a,b)$ is the profit from deleting nodes a and b. If there is

$$p(a,b) > \mathrm{Max}(p(a),p(b),p(c_i)),\ i \in [1,k]. \tag{11}$$

Then we call (a,b) and $(c_1,c_2,\ldots,c_k)$ have strongly dominated relation between them. Furthermore, we extend this strongly dominated relation to k-cut situation. Then we have

$$p(a_1,a_2,\ldots,a_k) > \mathrm{Max}(p(a_i),p(b_j)),\ i \in [1,k],\ j \in [1,m]. \tag{12}$$

Here $a_1,a_2,\ldots,a_k$ is the minimum cut set and $b_1,b_2,\ldots,b_m$ are dominated nodes.

3.2. The dominated clustering algorithm

Using the strongly dominated relation between nodes, we can design dominated clustering algorithm (DCA) to find out the critical vertex. To find the important nodes is to find the nodes with high profit. In our algorithm, we merge nodes step by step, and use two lists to save the node information (value and size). Value is the profit of the node and size keeps the index of the other nodes merged by the node.

Generally, we suppose there are one-degree nodes in the network. So from the Lemma 1 we can gain the network is 1-cut. At the first step, we initially set the value of every node is 1, and the size of it just keep the index of itself. After that, we delete the dominated nodes and their values are added to the corresponding minimum cut nodes' value. The value of the deleted nodes is set to be 0. According to Lemma 2, this process will be repeated until the network has no one-degree nodes.

Then we tend to deal with 2-cut situation. (a,b) are minimum cut and $(c_1,c_2,\ldots,c_k)$ are dominated nodes. From formula (11), we obtain

$$\frac{\left(a\cdot v\times a\cdot s+b\cdot v\times b\cdot s+\sum_{i=1}^{k} c_i\cdot v\times c_i\cdot s\right)}{a\cdot s+b\cdot s} > \mathrm{Max}(a\cdot v,b\cdot v,c_i\cdot v)\ i\in[1,k]. \tag{13}$$

Here we use v to represent value and s to represent the number of nodes in size. If the formula (11) is satisfied, we delete b and $(c_1,c_2,\ldots,c_k)$ and set the value to zero. Then we connect node a to the nodes that node b used to be connected with. The value of a is set as the profit of (a,b). The profit is

$$p(a,b)=\frac{\left(a\cdot v\times a\cdot s+b\cdot v\times b\cdot s+\sum_{i=1}^{k} c_i\cdot v\times c_i\cdot s\right)}{a\cdot s+b\cdot s}. \tag{14}$$

During the process of dealing with 2-cut situation, the network may have new one degree nodes. We just deal with the nodes as the method of 1-cut. When the network is k-cut, we can extend our algorithm to k-cut. $a_1,a_2,\ldots,a_k$ is minimum cut set and $b_1,b_2,\ldots,b_m$ are dominated nodes.

$$\frac{\left(\sum_{i=1}^{k} a_i\cdot v\times a_i\cdot s+\sum_{j=1}^{m} b_j\cdot v\times b_j\cdot s\right)}{\sum_{i=1}^{k} a_i\cdot s+\sum_{j=1}^{m} b_j\cdot s} > \mathrm{Max}(a_i\cdot v,b_j\cdot v),i\in[1,k]. \tag{15}$$

At the merging step, we merge the k minimum cut nodes as 2-cut. The value is set as

$$p(a_1,a_2,\ldots,a_k)=\frac{\left(\sum_{i=1}^{k} a_i\cdot v\times a_i\cdot s+\sum_{j=1}^{m} b_j\cdot v\times b_j\cdot s\right)}{\sum_{i=1}^{k} a_i\cdot s+\sum_{j=1}^{m} b_j\cdot s}. \tag{16}$$

Figure 2 shows several special cases for 1-cut, 2-cut, and 3-cut. We also show the merging method of each case.

Overall our algorithm is arranged as:

Step 1 Initialize the data struct value and size.

Step 2 Deal with 1-cut situation iteratively until there is no one degree node.

Step 3 Deal with 2-cut, 3-cut, and so on.

Step 4 Sort the node value. The node indexes in the size of node with high value are critical nodes for our algorithm.

From formula (15), we can find that when k is larger the strongly dominated relation is harder to be satisfied. Generally, for real networks, when we deal with k-cut situation, $k=4$ is enough. In this process, the information of the network is lossless, so we call this process as the lossless consolidation.

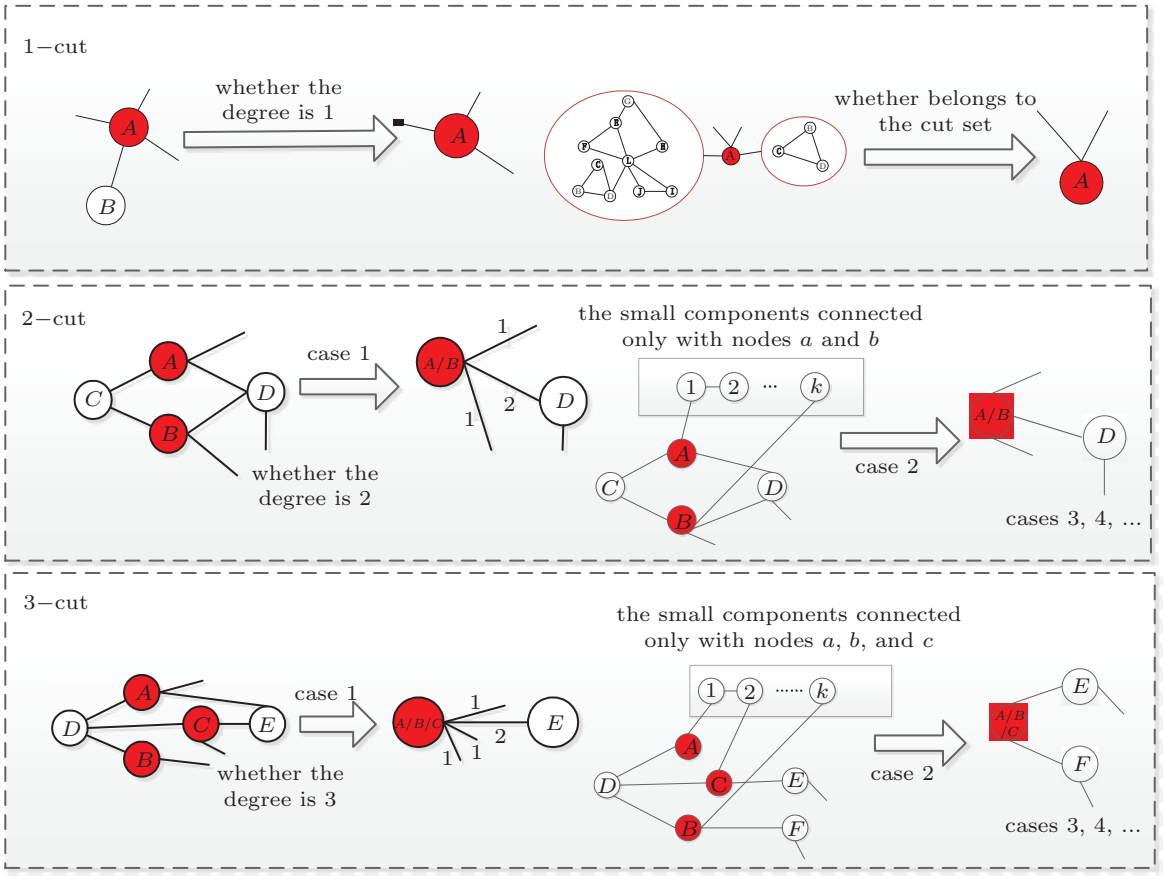


Fig. 2. (color online) The node centrality of different algorithms.

In Fig. 3, here is a small network with 52 nodes and 87 edges. In Step 1 of our algorithm, we find all the nodes with one degree, and then we merge the one-degree nodes into the connected nodes. At this moment, there exists no one-degree vertex in the network. In Step 2, firstly we find all the two-degree nodes. Then we focus on the nodes connected to these two-degree nodes and merge them into one node. For instance, we merge node A and the other nodes in the same circle into node A. Now we notice the existence of one-degree nodes in the network, so we repeat Step 1. Actually, Step 3 is the same as Step 1. At this point, we turn our attention to the cut point in the network. In Step 4, we find the one cut points, for instance, node C, and merge the other connected nodes into the one cut points, which is actually merge the nodes in the red circle into node C. In Step 5, two cut points are found and the nodes they connect are merged. For example, the nodes in the red circle are merged into node D. In this process, we should judge whether the nodes to be merged meet the combination condition before every merging.

The characteristics of our algorithm is that the algorithm focus on local structure of the network through cut nodes. We understand the network structures more clearly. The existing centralities can be divided into three classes which are neighborhood-based centralities, path-based centralities, and iterative refinement centralities. Neighborhood-based centralities are degree, k-core, H-index, and so on. These centralities focus on neighbor nodes of the central nodes. For path-based centralities, there are betweenness, closeness, and Subgraph centrality. These centralities concentrate on the shortest paths between nodes. For iterative refinement centralities, there are eigenvector, pagerank, HITs, and so on. These centralities are based on that the influence of a node is not only determined by the number of neighbor nodes, but also determined by the influence of its neighbor nodes. So in these centralities, every node gets support from its neighbor nodes. However, there is no centrality which concentrates on local dominated relations between nodes. In our algorithm, we calculate the node centrality using node mergence. This is also a characteristic of our algorithm.

After using the lossless consolidation, we can also do the lossy consolidation. This means the network will be further merged after applying our algorithm. The significance of the lossy consolidation is the visualization of the network. The real network has too many nodes to show the details of the network. Thus we want to merge the nodes first, and then the struct of the network will be easy to be shown. In lossy consolidation, firstly, we do community detection in the network after mergence. In every community, there are nodes which

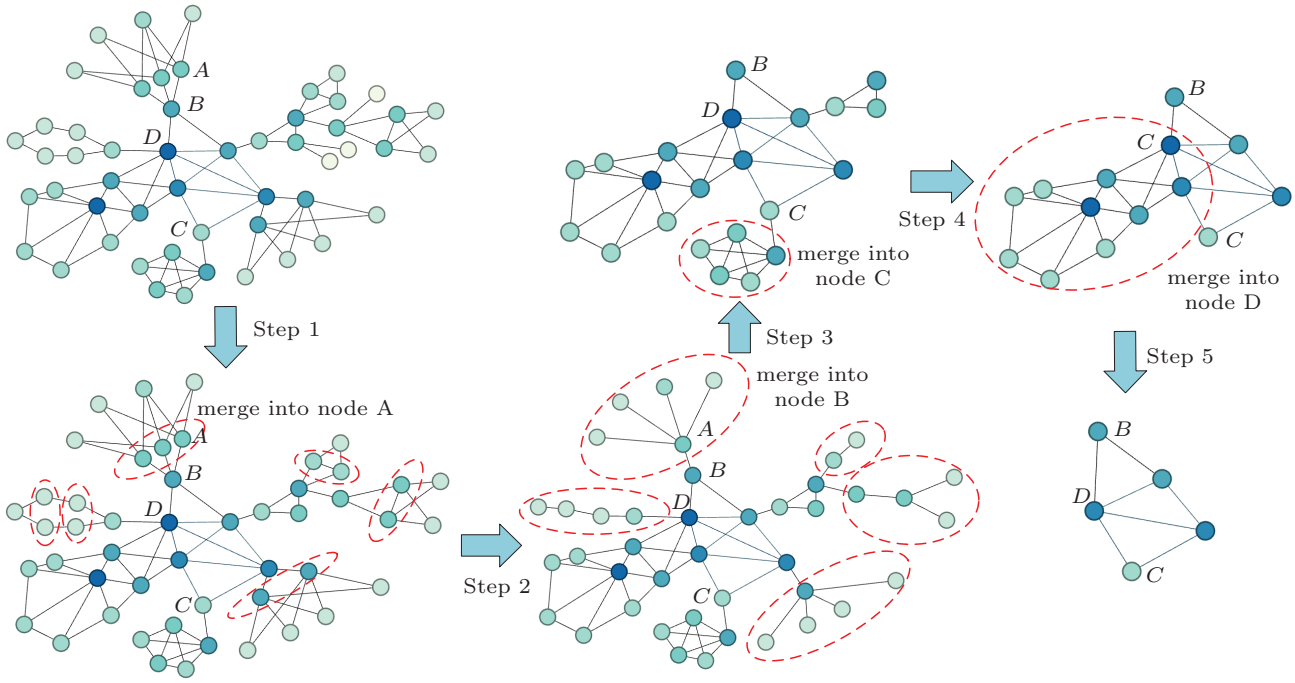


Fig. 3. (color online) The schematic diagram of mergence process.

are connected with other nodes in other community. We call these nodes as gate nodes. In a community C, $n_1, n_2, \ldots, n_m$ are nodes which are just connected with nodes in C. $g_1, g_2, \ldots, g_k$ are gate nodes of the network. So the profit of $g_1, g_2, \ldots, g_k$ is

$$p(g_1, g_2, \ldots, g_k) = \frac{\left(\sum_{i=1}^{m} n_i \cdot v \times n_i \cdot s + \sum_{i=1}^{k} g_i \cdot v \times g_i \cdot s\right)}{\sum_{i=1}^{m} n_i \cdot s + \sum_{i=1}^{k} g_i \cdot s}. \tag{17}$$

In every community, we keep the gate nodes. For nodes $n_1, n_2, \ldots, n_m$, if there is $p(n_i) > p(g_1, g_2, \ldots, g_k)$, we keep the node n_i. Otherwise, we delete it. Obviously, this process dose not satisfy the strongly dominated relation and the information of the network is missing, so we call this process as the lossy consolidation. The network after this process is easy for visualization.

3.3. The merging performance of networks

To show the efficiency of our algorithm, we do simulation in the dataset. Figure 4(a) is the visualization of router network which has 23441 nodes. Figure 4(b) is the router network after mergence which has 1164 nodes. We can use the network with 1164 nodes and merge data to represent the whole network. It is easy to observe that figure 4(a) is too complicated to show the structure of the network, while figure 4(b) shows the details of the network.

Table 3 shows the merging performance of different networks, and we can find that the dominated clustering algorithm can effectively merge the network. From lossless consolidation we can obtain strong dominated relation in the network. Then we use the node weight of the network after lossless consolidation to find out critical nodes.

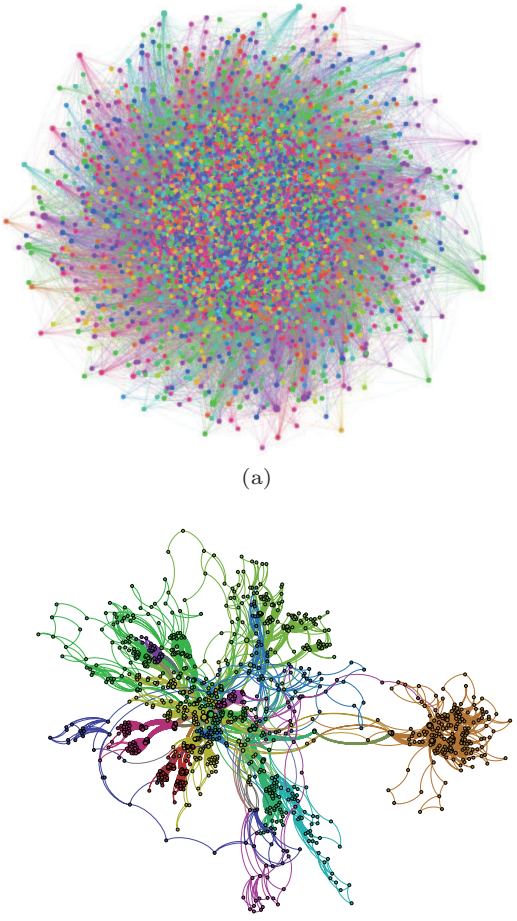


Fig. 4. (color online) Comparison of theoretical value and simulation value.

Table 3. The merging performance of different networks.

Networks (size)	Lossless	Lossy	Networks (size)	Lossless	Lossy
Router (23441)	1164	1151	BA3 (1000)	282	211
Yeast (2362)	1091	1081	BA4 (1000)	278	241
Erdos (6927)	1774	429	BA5 (1000)	473	320
Powergrid (4941)	750	692	BA6 (1000)	638	5
geom (9343)	1299	1180	BA7 (1000)	1	1
polblogs (1490)	959	951	BA8 (1000)	401	150
Roget (1022)	804	745	BA9 (1000)	575	233
BA1 (1000)	252	7	BA10 (1000)	498	7
BA2 (1000)	11	8			

4. Simulation

4.1. The correlation analysis on algorithms

To verify the effectiveness of the algorithm, we utilize Spyder on the python platform to achieve the DCA algorithm we proposed. We compared the simulation results among different algorithms in different networks. In the simulation we used the Package networkx. We can use *nx.degree*(), *nx.betweenness_centrality*(), *nx.pagerank*(), *nx.closeness_centrality*(), *nx.eigenvector_centrality*(), *and nx.k_core*() to calculate six centralities. In order to guarantee the accuracy of the simulation results, we expand the scale of the networks and repeat the experiments to reduce the error. The simulation results are the average of the experimental data.

In this part, we tend to analyze the correlation between the dominated clustering algorithm and existing algorithms. Table 4 shows the top ten nodes obtained by our algorithm and their value and rank in other centrality algorithms. (In Table 4, the first item is value and the second item is rank). We can find that the nodes chosen by our algorithm rank different in other algorithms, which shows our algorithm measure the importance of the nodes in a new angle. From simulations later, we can prove that the important nodes in our algorithm are conducive to network attack under the limited capacity.

Table 4. The artificial networks.

Rank	Degree	Betweenness	Closeness	*k*-core	Eigenvector	Pagerank
1	63/37	0.0843/0	0.169/42	10/167	26514/58	0.00069/102
2	35/190	0.0563/11	0.169/42	10/167	14386/1050	0.00034/369
3	58/46	0.0535/61	0.147/670	10/167	19588/402	0.00083/54
4	55/53	0.0767/67	0.146/716	10/167	18106/470	0.00078/70
5	39/148	0.0279/126	0.145/876	4/1390	10972/1437	0.00054/188
6	36/176	0.0250/183	0.144/952	4/1390	10740/1444	0.00050/206
7	34/205	0.0205/187	0.144/952	7/875	11630/1200	0.00046/231
8	37/171	0.0324/70	0.094/19085	4/1390	16100/870	0.00064/129
9	14/1074	0.0317/83	0.094/19085	11/1	19838/365	0.00021/737
10	31/242	0.0117/25	0.155/431	3/2867	9622/1900	0.00045/243

4.2. Comparision on attack performance

Figure 5 shows comparison between dominated clustering algorithm and other centrality algorithms. Figure 5(a)

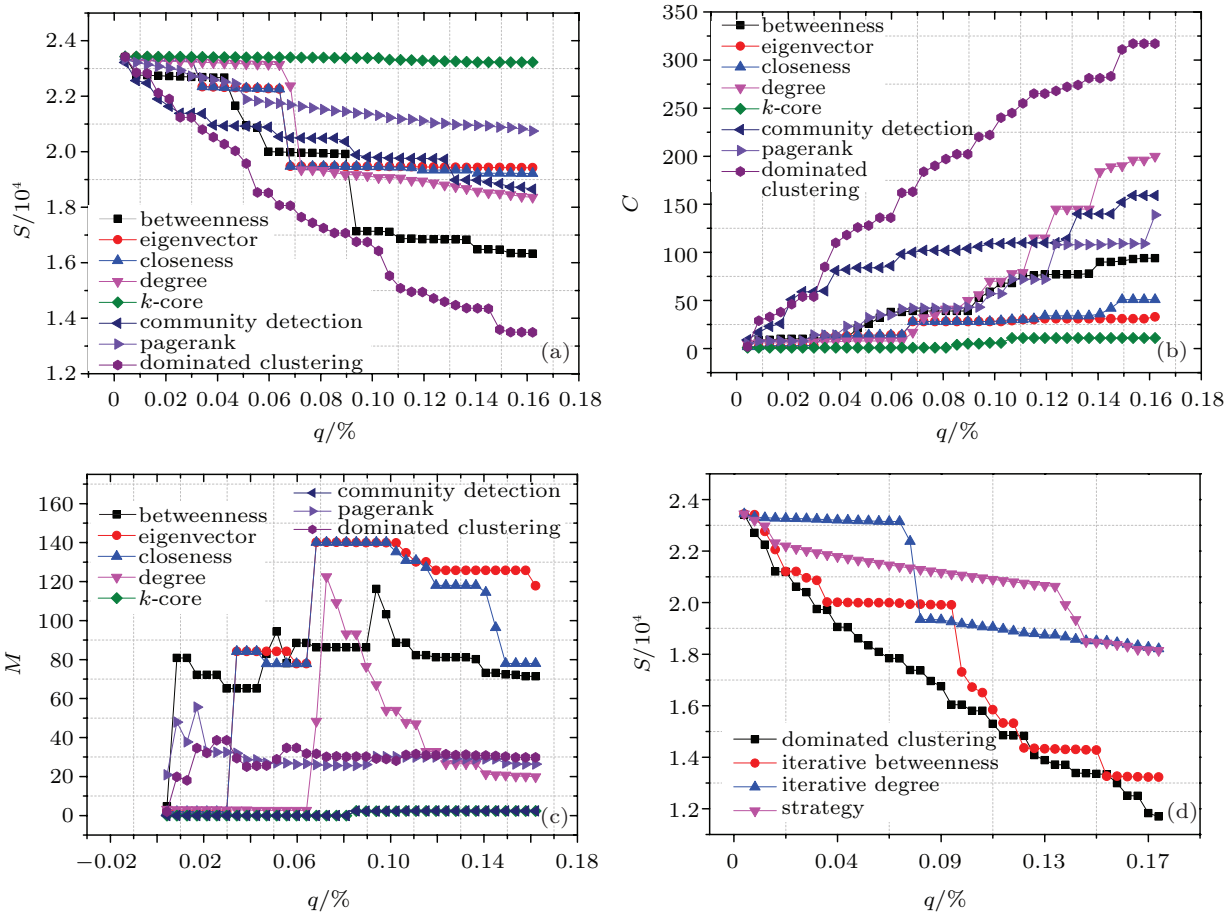


Fig. 5. (color online) The comparison of algorithms in router network.

measures the change of S during the process of deleting nodes. After deleting 0.18 percent of the nodes, dominated clustering algorithm performs best. S declines steadily. Figure 5(b) shows the change of C. We can find that dominated clustering algorithm generates more clusters than other algorithms. It means that the dominated clustering algorithm damages the network more thoroughly. Figure 5(b) is the change of M. The dominated clustering algorithm produces more clusters. The S of the dominated clustering algorithm is stable at 30 or so. We also use community detection algorithm (Greedy algorithm) to cluster nodes. Then we use the value of the nodes to show the importance of the nodes. From Fig. 5 we find that if we just use community detection algorithm, the performance of the attack is poor.

If we just attack the network using single indicator, this approach is too naive. There have been many existing researches describing this process as Information Maximum Problem (IMP). Given network $G(V,E)$, the target is to find a set S to maximize function $f(S)$. In this paper, $f(S)$ represents the degree of damage to the network which is related with LCC. Here we use a heuristic algorithm.[38] We obtain the remove nodes by adaptive recalculation, that is, to choose the node with the largest centrality at first, and then recalculate the centrality after removing the chosen node. Then we choose degree and betweenness to do this because of better performance. Greedy algorithm can also be used for choosing nodes. We choose top n nodes in every centralities as a strategy set. In every step of removing nodes, we choose the node which maximizes $f(S)$.

In the Fig. 5(d), we find if we remove the nodes with recalculating betweenness, the attack performance can be better. This process costs so much time because of the calculate complexity of betweenness. However, our algorithm still performs best when the attack capacity is limited.

In this section, we pay more attention on the performances of the algorithms under real networks. Tables 5–7 show the LCC of the network after deleting 0.4%, 0.7%, 1% nodes respectively. Each table gives the LCC of the seven real networks after deleting certain percent nodes based on six centrality algorithms and DCA we proposed. The lighted number is the minimum of the row, in other words, the algorithm the lighted number corresponding to has the best performance under the certain situation. From three tables, it can be derived that DCA we proposed is more effective when the attacked nodes number is small, alternatively, the attack to the network has a limited capacity. When the attacked nodes number is allowed to be larger, the advantage of our algorithm becomes not so obvious. We also noticed that DCA has a better performance when the network is sparser.

Table 5. The performance in the real networks ($q = 0.4\%$)

Network	Betweenness	Eigenvector	Closeness	Degree	k-core	Pagerank	DCA
Router (23441)	9998	18027	16225	12866	19129	14070	**9762**
Yeast (2362)	2151	2192	2165	2164	2207	2151	**2128**
Erdos (6927)	5625	6214	6186	5828	6384	5653	**5587**
Powergird (4941)	4914	4911	4826	4826	4917	4803	**4425**
Geom (7344)	3354	3426	3416	3410	3515	3340	**3086**
Polblogs (1490)	1184	1210	1209	1184	1216	1184	**1168**
Roget (1022)	1189	1212	1214	1196	1218	1189	**1178**

Table 6. The performance in the real networks ($q = 0.7\%$).

Network	Betweenness	Eigenvector	Closeness	Degree	k-core	Pagerank	DCA
Router (23441)	**4527**	15687	14924	9064	17324	10176	8068
Yeast (2362)	2121	2172	2132	2118	2196	2107	**2069**
Erdos (6927)	4993	5785	5718	5163	6010	5052	**4889**
Powergird (4941)	4892	4857	4766	4766	4859	4706	**4343**
Geom (7344)	3168	3355	3289	3251	3459	3158	**2870**
Polblogs (1490)	1168	1205	1174	1174	1212	1174	**1151**
Roget (1022)	987	987	987	987	983	987	**971**

Table 7. The performance in the real networks ($q = 1\%$).

Network	Betweenness	Eigenvector	Closeness	Degree	k-core	Pagerank	DCA
Router (23441)	**2449**	14611	14614	6067	15050	7901	7111
Yeast (2362)	2070	2158	2106	2075	2185	2065	**2029**
Erdos (6927)	**4392**	5433	5273	4666	5700	4433	4404
Powergird (4941)	4861	4852	4650	4650	4821	4592	**3991**
Geom (7344)	2976	3249	3168	3081	3322	2981	**2686**
Polblogs (1490)	1151	1200	1165	1164	1202	1160	**1129**
Roget (1022)	983	984	984	984	980	983	**962**

To show more detailed information about our algorithm, we display the results of deleting nodes from 0 to 1 in router networks in Fig. 6. We can find that when the attack capacity is not limited, the attack performance is similar for betweenness, degree, pagerank, and our algorithm. We can also find there is percolation phase in the process of deleting nodes. Furthermore, our algorithm focuses on the limited attack capacity. When the proportion of deleting nodes is below 1%, our algorithm is obviously better than other networks. In reality, it is impossible to have infinite attack capacity. For large networks, 1% of nodes may already have hundreds of nodes. So our algorithm is more practical.

After that, we study the comparison between the attack performance of the dominated clustering algorithm (DCA) we proposed and that of other centrality algorithms. Different from the simulations before, this section focuses on the performance in the BA networks with different assortativity coefficients. To simplify the simulation and make the figures clearer, here, we choose the degree algorithm (DA) to represent the centrality algorithms because of its popularity and good performance.

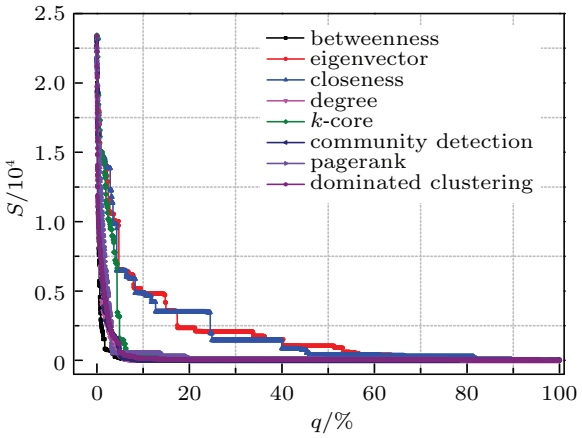


Fig. 6. (color online) The attack process of router network.

The simulation results are shown in Fig. 7. The networks in this section are all BA networks with 1000 nodes. The networks in Figs. 7(a) and 7(b) have 4000 edges while the networks in Figs. 7(c) and 7(d) have 2000 edges. Figures 7(a) and 7(c) give the comparison under the networks more disassortative, while for Figs. 7(b) and 7(d), the networks are more assortative. Generally, figure 7 shows DCA performs better

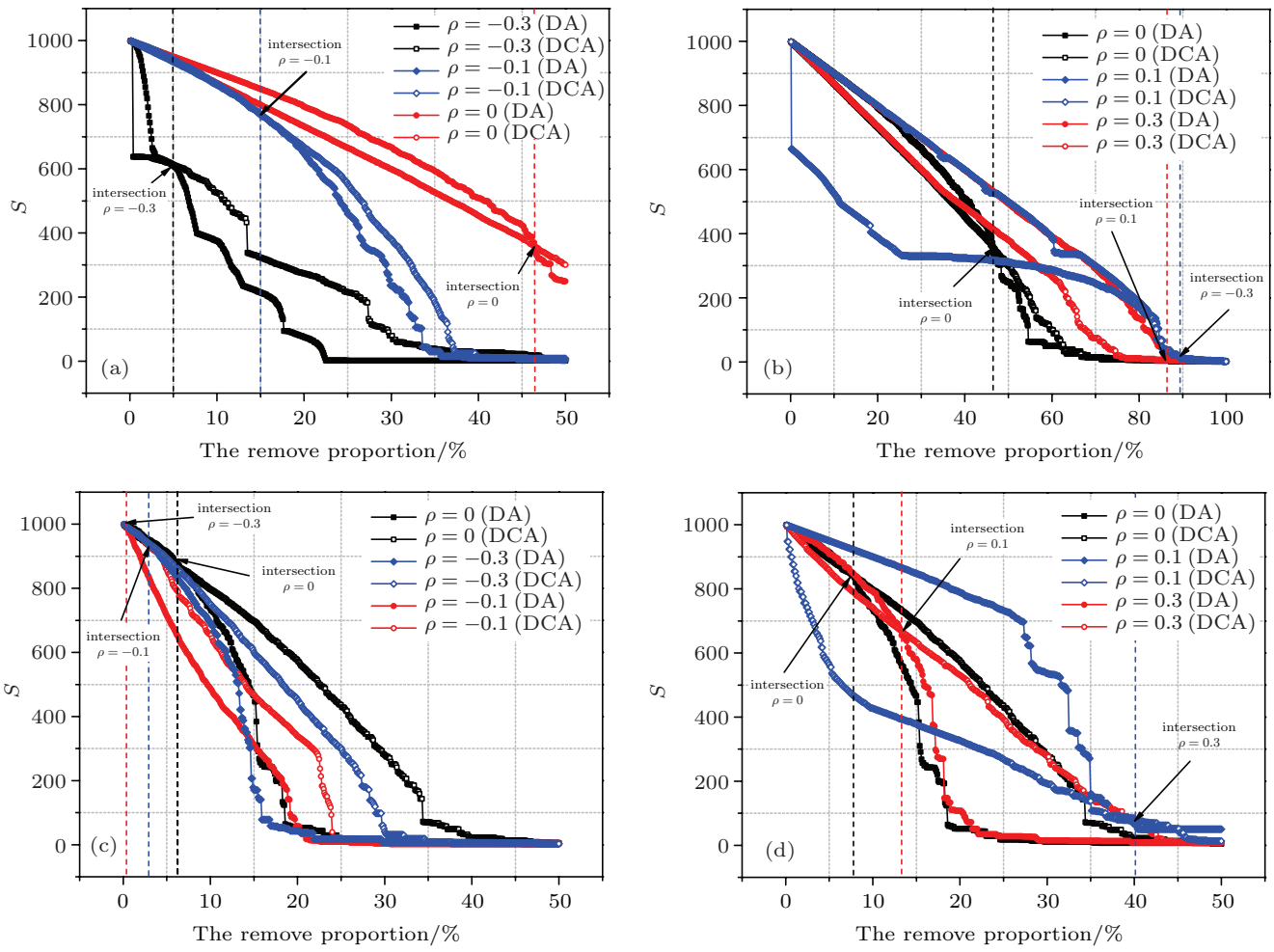


Fig. 7. (color online) The comparison of different performance in BA network. (a) Disassortative BA networks with 4000 edges, (b) assortative BA networks with 4000 edges, (c) disassortative BA networks with 2000 edges, (d) assortative BA networks with 2000 edges.

than DA when a few nodes are deleted. With the number of the deleted nodes increases, the performances of the two algorithms meet an intersection. Moreover, when the network is more assortative, the time they meet will be later, that is to say, the intersection will move right. That results further illustrate that the DCA we proposed is more useful for the network attack under the limited capacity and performance is better when the networks are more assortative.

5. Conclusion

In this paper, we focus on the dominated relation between nodes in the network. Based on minimum cut set of the network, we define a strongly dominated relation between nodes. After that, we can merge the network based on strongly dominated relation. Besides, the merging process is lossless. Because this process just merge the nodes which are dominated nodes. Then we design the dominated clustering algorithm using effective data structure to realize this process. With DCA we can find the critical nodes in the network. We also design an algorithm of lossy consolidation which is helpful in visulization. In simulation we compare the performance of our algorithm with other centrality algorithms. Simulation results show that when the attack capacity is limited, DCA performs best in the real networks. Furthermore, DCA is effective when the network is sparse. From simulation in scale-free network, we can obtain that DCA works better in assortative scale-free network.

References

[1] Cai Q and Liu J 2016 *Sci. Rep.* **6** 35904
[2] Baldassano S N and Bassett D S 2016 *Sci. Rep.* **6** 26087
[3] Zhang Y, Zhou S, Guan J, *et al.* 2011 *Phys. Rev. E* **87** 1079
[4] A.L Barabasi, H Jeong, Z Nda, *et al.* 2015 *Veterinary Surgery* **6** 66
[5] Gao J, Liu Y Y, D'Souza R M, *et al.* 2014 *Nat. Commu.* **5** 5415
[6] Ito S and Sagawa T 2013 *Phys. Rev. Lett.* **111** 81
[7] Mitrović M and Tadić B 2009 *Phys. Rev. E* **80** 026123
[8] Karrer B, Newman M E and Zdeborová L 2014 *Phys. Rev. Lett.* **113** 208702
[9] Decelle A, Krzakala F, Moore C, *et al.* 2011 *Phys. Rev. Lett.* **107** 065701
[10] Luccioli S, Olmi S, Politi A, *et al.* 2012 *Phys. Rev. Lett.* **109** 138103
[11] Kitsak M, Gallos L K, Havlin S, *et al.* 2010 *Nat. Phys.* **6** 888
[12] Dorogovtsev S N, Goltsev A V and Mendes J F 2006 *Phys. Rev. Lett.* **96** 185
[13] Elbirt B 2007 "The nature of networks: A structural census of degree centrality across multiple network sizes and edge densities", Ph. D. Dissertation (Buffalo: State University of New York)
[14] Freeman L C 1977 *Sociometry* **40** 35
[15] Freeman L C 1978 *Social Networks* **1** 215
[16] Batagelj V and Zaversnik M 2003 *Comput. Sci.* **1** 34
[17] Latora V and Marchiori M 2005 *Phys. Rev. E* **71** 015103
[18] Karrer B, Newman M E and Zdeborov L 2014 *Phys. Rev. Lett.* **113** 208702
[19] Agliari E, Cioli C and Guadagnini E 2011 *Phys. Rev. E* **84** 247
[20] Azimitafreshi N, Gmezgardes J and Dorogovtsev S N 2014 *Phys. Rev. E* **90** 032816
[21] Liu J G, Lin J H, Guo Q, *et al.* 2015 *Sci. Rep.* **6** 032812
[22] Tian L and Shi D N 2012 *Phys. Lett. A* **376** 286
[23] Kanevsky A 1993 *Networks* **23** 533
[24] Rosvall M and Bergstrom C T 2007 *Proceedings of the National Academy of Sciences USA* 1118
[25] Wang F, Li T, Wang X, *et al.* 2011 *Data Mining and Knowledge Discovery* **22** 493
[26] Singh A and Humphries M D 2015 *Sci. Rep.* **5** 8828
[27] Liu W, Pellegrini M and Wang X 2014 *Sci. Rep.* **4** 5739
[28] Brandes U 2008 *Social Networks* **30** 136
[29] Bonacich P 1987 *American Journal of Sociology* **92** 1170
[30] Langville A N and Meyer C D 2005 *Siam Review* **47** 135
[31] Bu D, Zhao Y, Cai L, *et al.* 2003 *Nucleic Acids Research* **31** 2443
[32] http://vlado.fmf.uni-lj.si/pub/networks/data/
[33] Watts D J and Strogatz S H 1998 *Nature* **393** 440
[34] Adamic L, Glance N, Adamic L, *et al.* 2005 *IL Nuovo Cimento A* **62** 127
[35] Roget P M and Dutch R A 1982 *Roget's Thesaurus of English Words and Phrases* (**Longman**) p. 90
[36] Knuth D E 1993 *The Stanford Graph Base: a platform for combinatorial computing* (Reading: Addison-Wesley) pp. 41–43
[37] Newman M E 2002 *Phys. Rev. Lett.* **89** 111
[38] Holme P, Kim B J, Yoon C N and Han S K 2002 *Phys. Rev. E* **65** 56109
[39] Kempe D, Kleinberg J and Tardos E 2003 *Proceedings of the* 9*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* , August 24–27, 2003, Washington, DC, USA, pp. 137–146