SCIENTIFIC REPORTS

OPEN

Received: 19 April 2017
Accepted: 14 August 2017
Published online: 06 September 2017

An efficient algorithm for finding all possible input nodes for controlling complex networks

Xizhe Zhang[1,2], Jianfei Han[2] & Weixiong Zhang[3,4]

Understanding structural controllability of a complex network requires to identify a Minimum Input nodes Set (*MIS*) of the network. Finding an *MIS* is known to be equivalent to computing a maximum matching of the network, where the unmatched nodes constitute an *MIS*. However, maximum matching is often not unique for a network, and finding all possible input nodes, the union of all *MIS*s, may provide deep insights to the controllability of the network. Here we present an efficient enumerative algorithm for the problem. The main idea is to modify a maximum matching algorithm to make it efficient for finding all possible input nodes by computing only one *MIS*. The algorithm can also output a set of substituting nodes for each input node in the *MIS*, so that any node in the set can replace the latter. We rigorously proved the correctness of the new algorithm and evaluated its performance on synthetic and large real networks. The experimental results showed that the new algorithm ran several orders of magnitude faster than an existing method on large real networks.

Controlling complex networks[1–3] is of great importance in many applications, such as social, biological and technical networks. For example, it has been shown that understanding network controllability can help identify genes responding to viral infection[4] and genes related to cancer[5], as well as assist analyzing metabolic process[6].

A network is said to be controllable if it can be driven from any initial state to a desirable state in finite steps by exerting external control signals on some selected nodes[1], which are called driver nodes[7], input nodes[8] or control nodes[9]. The controllability of a network can be determined by Kalman's controllability rank condition[1] if the weight of each edge is known, or Lin's structural controllability theory[10] if only a zero or non-zero value of each edge's weight is known. As edge weights of many real networks cannot be precisely measured and are often unknown, the structural controllability theory has been widely adopted recently to analyze biological systems, such as protein interaction network[4, 5, 11], signaling pathway network[12], and gene regulatory network[13, 14].

Based on the structural controllability theory, one approach[7–9, 15, 16] assumes that a node can control only one of its outgoing neighbor. Therefore, input nodes of a network can be inferred by finding a maximum matching of the network, which is consisted of the set of maximum edges that do not share nodes[3]. The unmatched nodes related to a maximum matching constitute a Minimum Input nodes Set, or *MIS*. However, this approach can be only applied to directed networks. Another approach[5, 17, 18] assumes that a node can control all of its outgoing edges independently, that is, a node can output different signals for each edge. Under this assumption, a minimum node dominate set (*MDS*) can be used to control the network[18]. This approach may be more reasonable for artificial networks and can be applied to undirected networks.

Based on above frameworks, extensive works have been done about control principles of complex networks. It has been shown that the size of an *MIS* is closely related to the node degree distribution of the network[7]. Interestingly, the fraction of input nodes is primarily determined by the nodes of low in- and out-degrees[16]. The concepts of input nodes and *MIS* are also extensively used in analyzing many biological networks, e.g., identifying important proteins in biological networks[4, 5], analyzing interbank networks[19], and increasing the effectiveness of selective modulation of brain networks[20].

Unfortunately, maximum matching is not unique for most networks[21] (Fig. 1), so there may exist numerous *MIS*s. Although these *MIS*s have the same size, they may have different input nodes. We call a node in an *MIS* a

[1]Key Laboratory of Medical Image Computing of Northeastern University, Ministry of Education, Shenyang, China. [2]School of Computer Science and Engineering, Northeastern University, Shenyang, Liaoning, China. [3]College of Math and Computer Science, Institute for Systems Biology, Jianghan University, Wuhan, 430056, China. [4]Department of Computer Science and Engineering, Washington University, Saint Louis, Missouri, USA. Correspondence and requests for materials should be addressed to X.Z. (email: zhangxizhe@mail.neu.edu.cn)

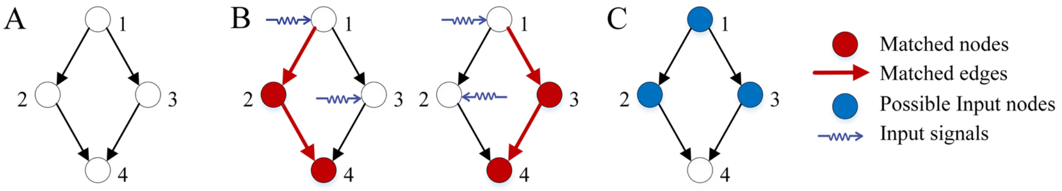

Figure 1. An example network with two *MIS*s. (**A**) An example network; (**B**) two *MIS* of the network $D_1 = \{1, 3\}$, $D_2 = \{1, 2\}$; (**C**) all possible input nodes of the network, which form the union of both *MIS*s.

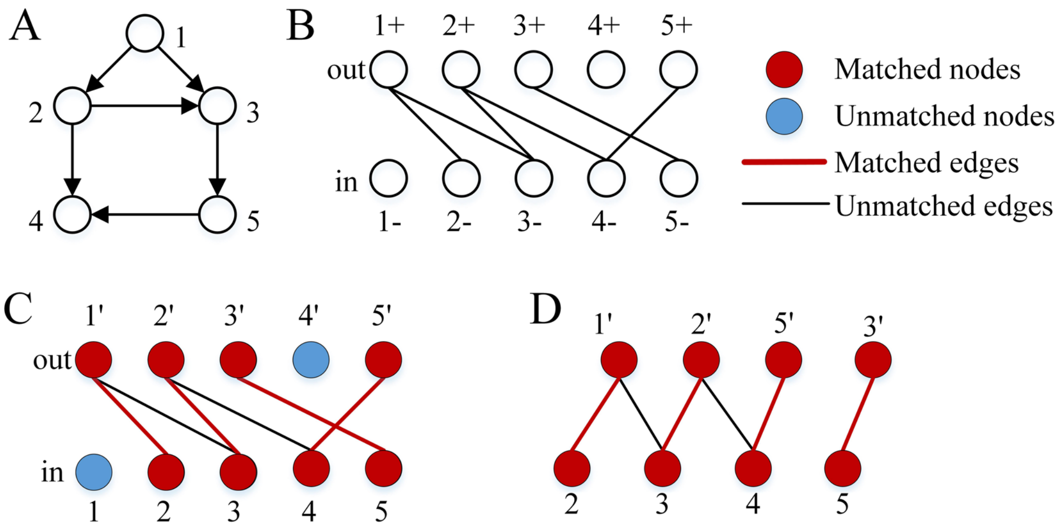

Figure 2. An example of a maximum matching of a network. (**A**) A directed network; (**B**) its corresponding bipartite graph; (**C**) A maximum matching of the bipartite graph. An unmatched node in the in-set is an input node; (**D**) Two alternating paths corresponding to the maximum matching in (**C**).

possible input node. Apparently, all possible input nodes are the union of all *MIS*s. It is important to find all possible input nodes for understanding the controllability of a complex network. For example, finding all possible input nodes may help understand the roles of nodes in control systems[22], design suitable MISs under different constraints[8], and identify critical genes on signaling pathways[14]. Finding all possible input nodes is essentially an enumeration problem. While enumeration problems have been extensively studied, such as for enumerating all maximum matchings[23, 24] or all maximally-matchable edges[25], there are very few works on how to find all unmatched nodes. To solve this problem, a previous approach[22] first computes a maximum matching and then assess if an unmatched node is a possible input node by removing it to test if its removal may result in a larger maximum matching. The computational complexity of finding a maximum matching is $O(N^{1/2}L)$ and the evaluation process is $O(NL)$ on a network of N nodes and L edges, for a total complexity of $O(NL)$ for this method.

We developed an efficient algorithm for finding all possible input nodes of a network. We proved that all possible input nodes could be identified by a simple modification to a maximum matching algorithm. The complexity of our algorithm is $O(N^{1/2}L)$, which is the same as the complexity of the maximum matching algorithm. Because our algorithm does not need to evaluate every node of the network, it runs several orders of magnitude faster than the previous method[22] on large networks. Furthermore, our algorithm can also output the substituting nodes set for each input nodes. Because some input nodes may not be suitable to be inputted control signals due to some economic or technical constraints, these substituting nodes can be used to replace the original input nodes and obtain new *MIS*s with some input nodes replaced.

Method

Consider a directed network $G(V, E)$ over a set of nodes V and a set of edges E. To find an *MIS* of a directed network $G(V, E)$, we first convert $G(V, E)$ to an equivalent undirected bipartite graph $B(V^{in}, V^{out}, E)$ (Fig. 2A,B). The bipartite graph is built by splitting the node set V into two node sets V^{in} and V^{out}, where a node n in G is converted to two nodes n^{in} and n^{out} in B, and nodes n^{in} and n^{out} are, respectively, connected to the in-edges and out-edges of node n.

Now consider maximum matching. A *matching* is a set of edges that share no common node. A node is called a *matched node* if it is connected to a matching edge, or *unmatched node*, otherwise. A matching with the maximum number of edges is called a *maximum matching*. In an undirected bipartite graph, an *alternating path* is a path whose edges are alternate in and out matching. An *augmenting path* is an alternating path whose two end nodes are unmatched nodes. Based on the Berge theorem[26], a matching M^* is a maximum matching if there is no

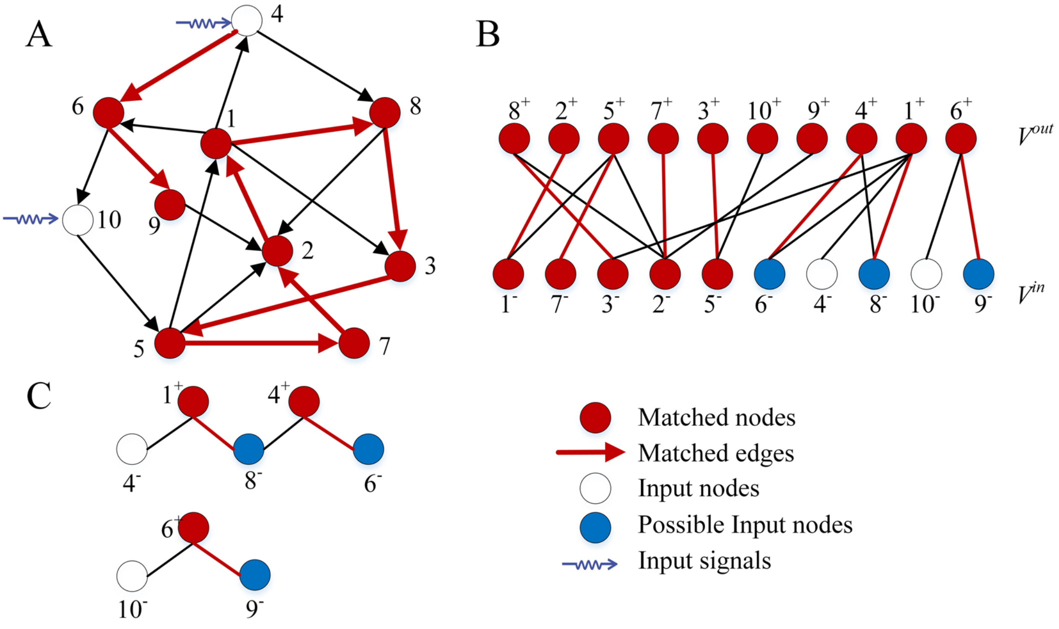

Figure 3. An example of the process of algorithm *All_Input*(*G*). (**A**) A sample network and its maximum matching (red edges) (**B**) the corresponding bipartite graph; (**C**) In the last step of the algorithm, we search for alternating paths from unmatched nodes. The nodes on the alternating paths in V^{in} are nodes {8, 6, 9}, and the input nodes are {4, 10}. Therefore, all possible input nodes of the network are {4, 6, 8, 9, 10}. The substituted nodes of input node 4 are node 8 and node 6, and those of input node 10 is node 9. Therefore, if node 4 or 10 is not suitable to be inputted control signals, we can use the corresponding substituted nodes to input control signals and regain full control of the network.

augmenting path in $B(v_1, v_2, E)$ with respect to M^*. The input nodes are the unmatched nodes in V^{in} corresponding to a maximum matching of bipartite graph $B(V^{in}, V^{out}, E)$. The unmatched nodes in V^{in} corresponding to any maximum matching form an MIS of *G*. If there exists a perfect matching in the network, the *MIS* can be any node of the network based on the Minimum Input Theorem[7].

Because maximum matching is not unique for most networks, there may exist many *MISs*. The union of all *MISs* contains all possible input nodes. We now show that to find all possible input nodes of a network, we only need to compute one maximum matching without enumerating all *MISs* or evaluating all matched nodes as done before in ref. [22].

Theorem 1: Given a directed network *G* and a maximum matching *M* of *G*, a node *n* is a possible input node if it satisfies one of the following two conditions:

1. Node *n* is an input node related to *M*;
2. The in-node n^{in} of the bipartite graph *B* can be reached from an input node m^{in} related to *M* through an alternating path p_{nm}.

Proof: It is sufficient to prove condition 2.

Sufficiency. Suppose that node *n* satisfies condition 2. Apparently, the length of p_{nm} must be even because both node n^{in} and m^{in} are in the set V^{in} of bipartite graph *B*. Therefore, the alternating path p_{nm} must start with an unmatched edge connected to m^{in} and end with a matched edge connected to node n^{in}. Change the types of all edges of p_{nm}, i.e., change the matched edges to unmatched and the unmatched edges to matched, then the new path p'_{nm} is still an alternating path. Consequently, we get a new maximum matching *M'*. Clearly, the node n^{in} is not matched by *M'* and m^{in} is matched by *M'*. Therefore, node *n* is an input node of *MIS* $D' = D\text{-}\{m\} + \{n\}$.

Necessity. Let node n^{in} be matched in *M* and be unreachable by any input node related to *M*. Suppose that node n^{in} is not matched by a maximum matching *M'*. Node n^{in} must have at least one in-edge because it is matched by *M*. Therefore, there must be an alternating path p_{nm} related to *M'* which starts with unmatched node n^{in} and end with a matched node m^{in}. Now consider the path p_{nm} under the maximum matching *M*. The length of p_{nm} must be even because nodes n^{in} and m^{in} are both in the set V^{in}. Therefore, the alternating path p_{nm} must end with an unmatched node m^{in} related to *M* because n^{in} is matched by *M*. This contradicts the fact that n^{in} cannot be reached by any input node related to *M*, which completes the proof.

The above proof implies an important fact that any of the nodes reachable from an input node can be used to replace the original input node and obtain a new *MIS*. Therefore, we have the following corollary:

Corollary 1: Consider an *MIS D* and one of its input node $m \in D$, let K_m be the set of the nodes that satisfied condition 2 of Theorem 1. For every node $n \in K_m$, $D' = D - \{m\} + \{n\}$ is an *MIS*.

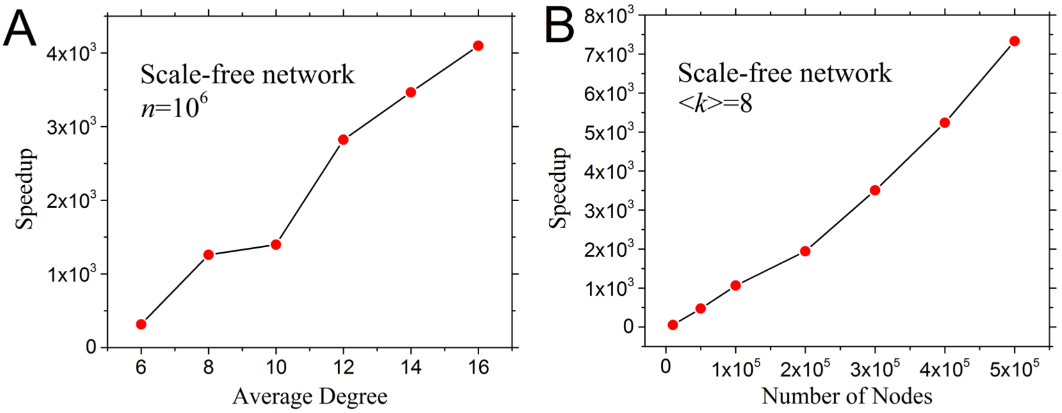

Figure 4. Speedup of our algorithm as compared to the previous algorithm[22]. (**A**) Speedup as a function of the average degree when $n = 10^6$; (**B**) Speedup as a function of the number of nodes when average degree $< k > = 8$. The networks are generated based on Scale-Free Network model[29] with $r^{in} = r^{out} = 3$.

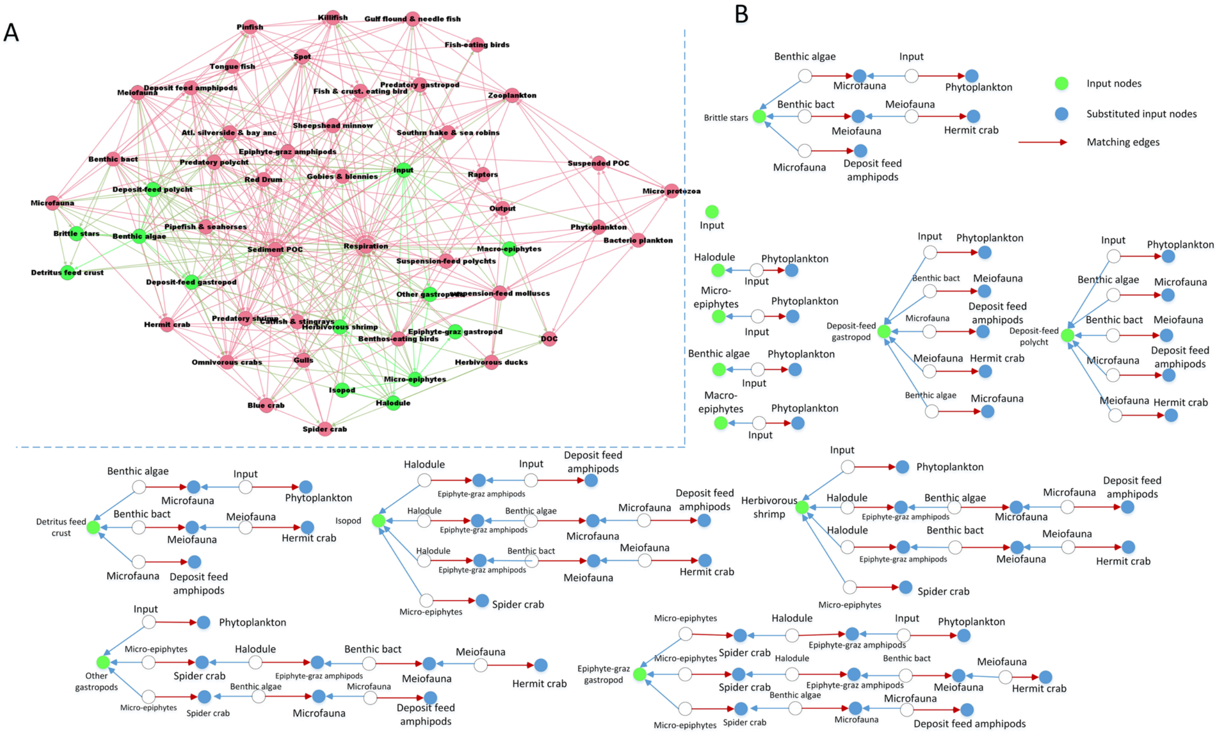

Figure 5. Input nodes and their substituted set of St. Marks foodweb[43]. (**A**) Network topology of St. Marks foodweb. The input nodes are green nodes. (**B**) All 13 Input nodes and their substituted nodes. We showed the alternating paths which connected the input nodes and their substituted nodes. The red edges are the matching edges.

Based on Theorem 1, the proof to the corollary is trivial. Corollary 1 suggests how to find the substituting nodes for an input node. For example, in Fig. 3, the substituting nodes of input node 4 are node 8 and node 6. If node 4 is not suitable for accepting control signals, we can use either node 8 or node 6 to substitute node 4, and the new *MIS* would be {8, 10} or {6, 10}.

The significance of Theorem 1 and Corollary 1 is that all possible input nodes can be identified by some alternating paths of the input nodes of any given *MIS*. This observation leads to a novel two-step approach to identification of all possible input nodes, i.e., we first compute an *MIS* and then consider all of its alternating paths. Moreover, these two steps can be combined using a simple modification to the Hopcroft–Karp maximum matching algorithm for undirected graphs[27]. The basic idea of the Hopcroft–Karp algorithm is to iteratively find all augmenting paths corresponding to the matching *M* at hand, and then to derive a larger matching *M*'. A maximum matching is obtained when no augmenting path can be founded. The last step of the algorithm is exactly

Network		*L*	n_{pd}	t_{new}*(sec)*	t_{old}*(sec)*	*Speedup*
$n = 10^6$	$<k> = 6$	300000	0.444	0.343	108.1	315.1
	$<k> = 8$	400000	0.396	0.546	687.9	1260.1
	$<k> = 10$	500000	0.124	0.858	1198.6	1396.9
	$<k> = 12$	600000	0.039	0.826	2330.6	2821.5
	$<k> = 14$	700000	0.018	0.889	3080.3	3464.9
	$<k> = 16$	800000	0.008	0.952	3900.1	4096.8
$<k> = 8$	$n = 10^5$	40000	0.332	0.047	2.5	52.1
	$n = 5 * 10^5$	200000	0.388	0.218	102.9	472.0
	$n = 10^6$	400000	0.411	0.530	562.9	1062.2
	$n = 2 * 10^6$	800000	0.397	1.435	2784.6	1940.5
	$n = 3 * 10^6$	1200000	0.397	2.106	7381.4	3504.9
	$n = 4 * 10^6$	1600000	0.399	2.777	14550.0	5239.5
	$n = 5 * 10^6$	2000000	0.395	3.276	24012.7	7329.9

Table 1. Comparison of the execution time of some synthetic networks. For each network, we show its average degree $<k>$, number of nodes (N) and edges (L), destiny of all possible input nodes N_{pd}, the execution time of our method t_{new}, the execution time of previous algorithm[22] t_{old}, and the speedup ratio.

Type	**Name**	*N*	*L*	n_{pd}	t_{new} *(sec)*	t_{old} *(sec)*	*Speedup*
Biological	*E. Coli*[30]	423	578	0.730	0.001	0.016	16.0
	TRN-Yeast-1[31]	4441	12873	0.999	0.015	0.062	4.1
	TRN-Yeast-2[32]	688	1079	0.920	0.001	0.015	15.0
	Human PPI[33]	6339	34814	0.585	0.032	1.295	40.5
Trust	Slashdot0902[34]	82168	948464	0.912	0.421	1568.3	3725.2
	Slashdot0811[34]	77360	905468	0.910	0.234	1388.9	5935.5
	WikiVote[35]	7115	103689	0.666	0.047	2.044	43.5
	SciMet[36]	3084	10416	0.661	0.015	0.187	12.5
	Kohonen[37]	4470	12731	0.669	0.016	0.172	10.8
Internet	p2p-1[38]	10876	39994	0.911	0.062	2.746	44.3
	p2p-2[38]	8846	31839	0.926	0.046	1.732	37.6
	p2p-3[38]	8717	31525	0.933	0.031	1.700	54.8
Product co-purchasing	Amazon0302[39]	262111	1234877	0.177	1.685	14119.4	8379.5
	Amazon0312[39]	400727	3200440	0.127	9.344	36696	3927.2
	Amazon0505[39]	410236	3356824	0.915	7.519	45453	6045.2
	Amazon0601[39]	403394	3387388	0.053	2.886	49293	17080.1
Social network	Twitter[40]	81306	1768149	0.800	2.230	2532.5	1135.6
	Higgs_Twitter[41]	456626	14855842	0.297	12.589	66445.2	5278.0
	UClonline[42]	1899	20296	0.819	0.016	0.296	18.5
	Facebook_348[40]	572	6384	0.612	0.001	0.062	62.0

Table 2. Comparison of the execution time of some real networks. For each network, we show its type, name, number of nodes (N) and edges (L), density of all possible input nodes n_{pd}, the execution time of our method t_{new}, the execution time of previous algorithm[22] t_{old}, and the speedup ratio.

to look for all alternating paths starting from the input nodes of the maximum matching. Therefore, all possible input nodes can be obtained in the last step of Hopcroft–Karp algorithm based on Theorem 1.

The above idea and steps can be formulated in Algorithm *All_Input*(G) for finding all possible input nodes in network G, which is listed as follows:

All_Input(G):

1. For a directed network $G(V, E)$, let $B(V^{out}, V^{in}, E)$ be its corresponding bipartite graph; let the initial matching $M = \text{null}$;
2. Find all the alternating paths of all unmatched nodes in V^{in}, denoted as $AP = \{P_1, P_2 \ldots P_n\}$, and let the nodes of AP in V^{in} as candidate results;
3. If AP contain augmenting paths, expand all augmenting paths and obtain a new matching M'; clear all candidate nodes, $M = M'$; return to step 2;
4. If AP contain no augmenting path, the candidate nodes are all possible input nodes, and the set of the unmatched nodes is an *MIS* of G.

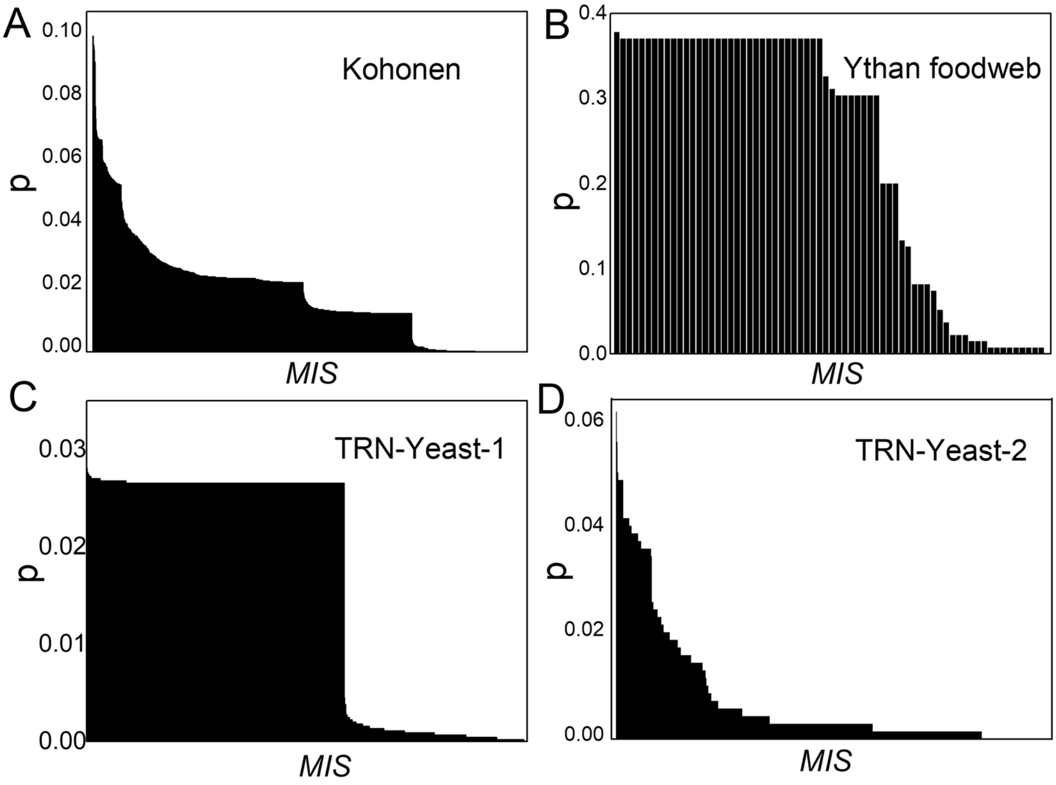

Figure 6. Percentage of substituted nodes for each input node of real networks. The networks we used are Kohonen[37], TRN-Yeast-1[31], Ythan foodweb[44] and TRN-Yeast-2[31]. The vertical axis represents the percentage of substituted nodes $p_i = s_i/N$, where s_i is the number of substituted nodes of input node i, and N is the total number of the nodes. The horizontal axis shows an *MIS*, in which the input nodes are sorted based on s_i by descending order.

Figure 3 illustrates an example of *All_Input*(*G*) on a small network. The time complexity of the above algorithm is the same as that of the Hopcroft-Karp algorithm, which is $O(N^{1/2}L)$.

Result

To assess the efficiency of the new algorithm, which was coded in JAVA, we compared it with the previous algorithm[22]. The source code of our algorithm and the previous algorithm is available in the supplementary information. The comparison was done on a Windows 7 workstation with a quad-core Intel i7-3770 processor of 3.9 GHz and 32GB DDR3 1600MHz memory.

We considered 13 synthetic networks, in which the number of nodes n varied from 10^5 to 5×10^6 and the average degree $<k>$ varied from 6 to 16. Networks were generated with Gephi[28] based on the Scale-Free Network model[29]. The experimental results on these networks showed that our algorithm outputted the same results set yet significantly outperformed previous algorithm[22] (Fig. 4). With a small network with $n = 10^5$, our algorithm achieved 52x speedup compared to the previous algorithm[22]. With a larger network with $n = 5 \times 10^6$, our algorithm achieved 7330x speedup with the execution time being only 3.276 seconds. Note that the speedup increases with the average degree.$<k>$ (Fig. 4A), which indicates that our algorithm has better performance on dense networks. The details of the results are listed in Table 1.

Next, we evaluated the performance of the algorithm on some real networks. These networks were selected based on their diversity of topological structures. These networks include biological networks, social networks, and Internet networks. The size of these networks varied from very small (*E. Coli* network, 423 nodes) to very large (Amazon network, 4×10^6 nodes). The results shown in Table 2 indicated that our algorithm also significantly outperformed the previous algorithm. On large networks, such as Amazon or Twitter, the results can be obtained within 10 seconds, resulting in an almost 10^4x speedup compared to the previous algorithm.

As we have proven in corollary 1, our algorithm can also output the substituting nodes for each input node. Figure 5A shows an example of St. Marks foodweb[43], which has 13 input nodes in an *MIS*. We computed the substituting nodes for each input node and showed the alternating paths between them in Fig. 5B. Interestingly, the size of the substituting nodes set of each input node is different, indicating some input nodes are more robust in controlling the network. The experimental results on other real networks are similar (Fig. 6). Note that some input nodes have the same number of substituting nodes because they are linked to the same set of substituting nodes, e.g., those of TRN-Yeast-1[31] and Ythan foodweb[44] in Fig. 6.

Conclusion

We developed an efficient algorithm for finding all possible input nodes for controlling complex networks. We proved that all possible input nodes can be efficiently identified along with computing an *MIS* without increasing the overall complexity beyond finding the *MIS*. Therefore, our algorithm offers a significant speedup over the previous algorithm on both synthetic networks and many large real networks. Furthermore, our algorithm can also output the substituted nodes set for each input node. It means that once we computed an *MIS*, we can immediate obtain all the substituting nodes for the *MIS*. Thanks to its efficiency, the new algorithm makes it possible to study controllability of large real-world networks and will have many potential applications in diverse areas.

Data availability. All data generated or analyzed during this study are included in this article (and its Supplementary Information files).

References

1. Kalman, R. E. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control* **1**, 152–192 (1963).
2. Luenberger, D. Introduction to dynamic systems: theory, models, and applications. *Proceedings of the IEEE* **69** (1979).
3. Murota, K. *Matrices and matroids for systems analysis*. Vol. 20 (Springer Science & Business Media, 2009).
4. Vinayagam, A. *et al*. Controllability analysis of the directed human protein interaction network identifies disease genes and drug targets. *Proc Natl Acad Sci USA* **113**, 4976–4981, doi:10.1073/pnas.1603992113 (2016).
5. Wuchty, S. Controllability in protein interaction networks. *Proceedings of the National Academy of Sciences* **111**, 7156–7160 (2014).
6. Basler, G., Nikoloski, Z., Larhlimi, A., Barabási, A.-L. & Liu, Y.-Y. Control of fluxes in metabolic networks. *Genome research* **26**, 956–968 (2016).
7. Liu, Y.-Y., Slotine, J.-J. & Barabási, A.-L. Controllability of complex networks. *Nature* **473**, 167–173 (2011).
8. Zhang, X., Lv, T. & Pu, Y. Input graph: the hidden geometry in controlling complex networks. *Scientific reports* **6** (2016).
9. Ruths, J. & Ruths, D. Control profiles of complex networks. *Science* **343**, 1373–1376 (2014).
10. Lin, C.-T. Structural controllability. *IEEE Transactions on Automatic Control* **19**, 201–208 (1974).
11. Wuchty, S., Boltz, T. & Küçük-McGinty, H. Links between critical proteins drive the controllability of protein interaction networks. *Proteomics* (2017).
12. Kawakami, E. *et al*. Network analyses based on comprehensive molecular interaction maps reveal robust control structures in yeast stress response pathways. *npj Systems Biology and Applications* **2**, 15018 (2016).
13. Wang, B. *et al*. Diversified control paths: A significant way disease genes perturb the human regulatory network. *PloS one* **10**, e0135491 (2015).
14. Ravindran, V., Sunitha, V. & Bagler, G. Identification of critical regulatory genes in cancer signaling network using controllability analysis. *Physica A: Statistical Mechanics and its Applications* **474**, 134–143 (2017).
15. Zhang, X., Lv, T., Yang, X. & Zhang, B. Structural controllability of complex networks based on preferential matching. *PloS one* **9**, e112039 (2014).
16. Menichetti, G., Dall'Asta, L. & Bianconi, G. Network controllability is determined by the density of low in-degree and out-degree nodes. *Physical review letters* **113**, 078701 (2014).
17. Nacher, J. C. & Akutsu, T. Dominating scale-free networks with variable scaling exponent: heterogeneous networks are not difficult to control. *New Journal of Physics* **14**, 073005 (2012).
18. Nacher, J. C. & Akutsu, T. Structural controllability of unidirectional bipartite networks. *Scientific reports* **3**, 1647 (2013).
19. Delpini, D. *et al*. Evolution of controllability in interbank networks. *Scientific reports* **3**, 1626 (2013).
20. Kumar, A., Vlachos, I., Aertsen, A. & Boucsein, C. Challenges of understanding brain function by selective modulation of neuronal subpopulations. *Trends in neurosciences* **36**, 579–586 (2013).
21. Zdeborová, L. & Mézard, M. The number of matchings in random graphs. *Journal of Statistical Mechanics: Theory and Experiment* **2006**, P05003 (2006).
22. Jia, T. *et al*. Emergence of bimodality in controlling complex networks. *Nature communications* **4** (2013).
23. Uno, T. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. *Algorithms and Computation*, 92–101 (1997).
24. Uno, T. In *International Symposium on Algorithms and Computation*. 367–379 (Springer).
25. Tassa, T. Finding all maximally-matchable edges in a bipartite graph. *Theoretical Computer Science* **423**, 50–58 (2012).
26. Berge, C. Two theorems in graph theory. *Proceedings of the National Academy of Sciences* **43**, 842–844 (1957).
27. Hopcroft, J. E. & Karp, R. M. In *Switching and Automata Theory, 1971., 12th Annual Symposium on*. 122–125 (IEEE).
28. Bastian, M., Heymann, S. & Jacomy, M. Gephi: an open source software for exploring and manipulating networks. *ICWSM* **8**, 361–362 (2009).
29. Maslov, S. & Sneppen, K. Specificity and stability in topology of protein networks. *Science* **296**, 910–913 (2002).
30. Shen-Orr, S. S., Milo, R., Mangan, S. & Alon, U. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature genetics* **31**, 64–68 (2002).
31. Balaji, S., Babu, M. M., Iyer, L. M., Luscombe, N. M. & Aravind, L. Comprehensive analysis of combinatorial regulation using the transcriptional regulatory network of yeast. *Journal of molecular biology* **360**, 213–227 (2006).
32. Milo, R. *et al*. Network motifs: simple building blocks of complex networks. *Science* **298**, 824–827 (2002).
33. Vinayagam, A. *et al*. A directed protein interaction network for investigating intracellular signal transduction. *Sci. Signal.* **4**, rs8–rs8 (2011).
34. Leskovec, J., Lang, K. J., Dasgupta, A. & Mahoney, M. W. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* **6**, 29–123 (2009).
35. Leskovec, J., Huttenlocher, D. & Kleinberg, J. In *Proceedings of the 19th international conference on World wide web*. 641–650 (ACM).
36. De Nooy, W., Mrvar, A. & Batagelj, V. *Exploratory social network analysis with Pajek*. Vol. 27 (Cambridge University Press, 2011).
37. Handcock, M. S., Hunter, D., Butts, C. T., Goodreau, S. M. & Morris, M. statnet: An R package for the Statistical Modeling of Social Networks. Web page http://www.csde.washington.edu/statnet (2003).
38. Ripeanu, M., Foster, I. & Iamnitchi, A. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028* (2002).
39. Leskovec, J., Adamic, L. A. & Huberman, B. A. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)* **1**, 5 (2007).
40. Leskovec, J. & Mcauley, J. J. In *Advances in neural information processing systems*. 539–547.
41. De Domenico, M., Lima, A., Mougel, P. & Musolesi, M. The anatomy of a scientific rumor. *arXiv preprint arXiv* **1301**, 2952 (2013).
42. Opsahl, T. & Panzarasa, P. Clustering in weighted networks. *Social networks* **31**, 155–163 (2009).
43. Baird, D., Luczkovich, J. & Christian, R. R. Assessment of spatial and temporal variability in ecosystem attributes of the St Marks National Wildlife Refuge, Apalachee Bay, Florida. *Estuarine, Coastal and Shelf Science* **47**, 329–349 (1998).
44. Dunne, J. A., Williams, R. J. & Martinez, N. D. Food-web structure and network theory: the role of connectance and size. *Proceedings of the National Academy of Sciences* **99**, 12917–12922 (2002).

Acknowledgements

This research was supported by the Fundamental Research Funds for the Central Universities of China under grand number N140404011, and the Natural Science Foundation of China under grant number 91546110, and China Scholarship Council under grant number 201606085011, and the Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase).

Author Contributions

X.-Z.Z. proved the theorem and designed algorithm. J.-F.H. wrote the code and performed the experiments. X.-Z.Z. and W.-X.Z. wrote the paper. All authors reviewed the manuscript.

Additional Information

Supplementary information accompanies this paper at doi:10.1038/s41598-017-10744-w

Competing Interests: The authors declare that they have no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.