Shortest path algorithms for large complex networks based on community detection

Huixiong Wang and Xing Pan

***Abstract*—**The shortest path query is a common task in different domains, while large network scale and large amount of processing are challenging the traditional shortest path algorithms in performance and efficiency. This article proposes a shortest path algorithm which combines the traditional algorithms with community detection. In the proposed algorithm, community detection is utilized to integrate the fuzzy structure information in the network. As a result, the scale of the network is significantly reduced, which accelerates the calculation. If the algorithm is used for multiple SP query processing, the community information can be reused, leading to considerable improvement of performance. Based on the results in evaluation, it turns out that in middle (500 nodes) and large scale (1500 or 5000 nodes) BA networks, our algorithm is more efficient than traditional Dijkstra method in both single query (5%-138%) and multiple query (104%-3905%).

***Keywords*—**Algorithm design and analysis, computational efficiency, network theory, shortest path problem.

I. Introduction

THE shortest path (SP) problem in large-scale networks is a common task in different domains. Due to its various applications in areas, this problem has been considered by researchers from various communities: route planning in computer networks [1]-[3], public transportation routes design [4]-[6], reachability queries [7], distance oracles[8], and social science [9], etc. Traditional SP algorithms, such as Bellman–Ford and Dijkstra's algorithms, are widely used in both industry and in research because of their simplicity in algorithmic design [10]-[13].

However, for those SP problems in large-scale networks, traditional algorithms have low performance and will cause considerable time and computing resources consumption [14]. Therefore, to proposed a feasible and efficient SP algorithm is significant in improving the performance of the algorithm and saving resource consumption. Previous literature [15]-[18] studied different factors that can influence the efficiency of SP solving, and affirmed that the topology, density and scale of the network have different levels of influence on the efficiency of SP algorithm.

The community structure is an important characteristic in networks, especially in large-scale complex networks, e.g., the World Wide Web [19][20], transportation networks [21], biological networks [22], electrical power grids [23], and other self-organizing systems, and has attracted huge attention in scientific research. In electrical power grids, regional power distribution system consists of numerous open or closed loop circuits, while the cross-regional power transmission is carried by a small number of high-voltage electricity transmission lines [23][24]. In a local area network (LAN), the routers are arranged on the basis of spatial locations of the terminal devices, which means that devices in the same room (floor, block, etc.) have closer and denser connections, while devices that are relatively far apart are connected by backbone cables and routers. The community structure in complex network is significant for solving the problems in different domains of research and industry, e.g., information spreading [25], logistics distribution [26], and recommendation systems [27].

In this paper, we propose a novel SP algorithm that combines community detection and traditional algorithms. The network is divided into several groups by community detection, with the boundary nodes defined. In this way, the original procedure of the algorithm is divided into 3 steps: community detection, network reconstruction, inter-community and terminal path calculation. With the preprocessing of community detection, the network is reduced in scale, which made the calculating scale much smaller. We design a series of experiments to compare the traditional algorithms and the proposed algorithm, and find that the utilization of community detection can improve the time efficiency by 5%-3905%, depending on network type, scale and the amount of calculation.

II. Proposed algorithm

In this Section, we introduce the proposed algorithm, which consists of 3 stages: community structure detection, reduction and reconstruction of network, and shortest path calculation. For ease of expression, the community structure detection and the reduction of the network are termed as the preprocessing of the algorithm.

A. Community Structure Detection

To detect the community structure of a network is to decide which nodes should be or should not be in a community, and divide the whole network into several communities [28]-[30].

This work is supported by the National Key Research and Development Program of China under Grants No. 2017YFF0206504, the National Natural Science Foundation of China under Grants No. 71171008/No. 71571004, and the Special Scientific Research Project for Civil Aircraft of the Ministry of Industry and Information Technology.

Huixiong Wang is with the School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China. Phone: +8619800333183; e-mail: wanghuixiong@buaa.edu.cn.

Xing Pan is with the School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China. (e-mail: panxing@buaa.edu.cn).

A good partition of community should satisfy the requirement that the connection in a community is dense, and the connection between communities is sparse. The quality of the community detection is often measured by an index: modularity (Q) [20][21]. The modularity of a network partition is a larger-the-better value between -1 and 1 that measures the density of links inside communities as compared to links between communities [31][32]. In weighted networks, modularity of the partition can be simply understood as the sum of edge weights within each community minus that between the communities. Mathematically, the modularity of the partition is defined and can be simplified as [21]:

$$Q = \sum_{c}[\frac{\Sigma in}{2m} - (\frac{\Sigma tot}{2m})^2] \quad (1)$$

where Σin represents the sum of weights of the edges in community c, Σtot represents the sum of weights of the edges which connects nodes in community c to other communities, and m represents the sum of weights of all edges in the network.

The goal of modularity-based community detection algorithm is to maximize the modularity of the partition [31]. In the proposed algorithm, we use Louvain Method [20]:

a) *Initialization*: Take each node as a community. For a network consisting of n nodes, we now have n communities.

b) *Assignment*: Successively assign node i to merge with each of its neighbor community, and calculate the increment of modularity ΔQ after the assignment. Mathematically, a simplified ΔQ can be represented as follows:

$$\Delta Q = \frac{k_{i,in}}{2m} - \frac{\Sigma_{tot} k_i}{2m^2} \quad (2)$$

where $k_{i,in}$ represents the sum of weights of the edges that connect node i with its neighbors in the same community, and k_i represents the sum of the weights of all edges attached to node i.

When all nodes are traversed, we choose the neighbor community k with the greatest nonnegative ΔQ to merge with node i. If all assignment of merging result in negative ΔQ, we let the node not to merge with any community.

c) *Circulation*: Successively merge community c with each of its neighbor community, and choose the community with the greatest nonnegative ΔQ.

d) *Finish*: when the merging of any pair of community will produce a negative ΔQ, the community detection is finished and we now have a community partition with the highest modularity Q.

B. Reduction and Reconstruction of Network

The purpose of community detection in the proposed SP algorithm is to reduce the scale of the network, and minimize the amount of information traversed in the calculation. In the reconstruction of networks, previous literature [16][20] adopted the aggregation of the community to reconstruct the network, in which nodes in the same network are compressed into one node, and self-loops are added to represent the edges that connect the original nodes within the same community. From the perspective of visualization, we can adopt this method and use different sizes and colors of nodes to represent the scale of each community and the relationship between communities, therefore, this aggregation method can help us demonstrate the community structure in a large-scale network explicitly with a reduced and concise network [28][33]. However, this aggregation omits some necessary information of edge and node connection inside the community for solving the shortest path. In our proposed algorithm, the procedure of the reducing and reconstructing the network is as follows:

a) *Extracting the boundary of community*: If there exists nodes i and j which satisfy: 1. i and j are neighbors; 2. i belongs to community c_1 and j belongs to community c_2, we say that node i is a boundary node of community c_1 and j is a boundary node of community c_2 [29]. All the boundary nodes of the same community constitute the boundary of the community. Based on this definition, we can extract all boundary nodes from the network, and the boundary node can be represented as node i_c, where c represents the community to which i_c belongs.

b) *Internal distance calculation*: When all boundary nodes are extracted, we calculate the distance between each pair of boundary nodes that belong to the same community. Then, in each community, we add edges to connect each pair of boundary nodes, take the distance calculated in this step as the weight of the new edges, and remove the other edges. As a result of adding these edges, all boundary nodes are globally connected in each community.

c) *External distance calculation:* On the basis of the first two steps, we calculate the distance between each community, i.e., the minimum sum of weight associated to the edges necessary to travel between each connected pair of community. Now we have reconstructed a reduced network, which contains the boundary nodes, the distance between boundary nodes within the same community and the distance between connected communities.

C. Shortest Path Calculation

In the proposed algorithm, the process above is termed the preprocessing of the network. After the preprocessing of the network, most of the nodes and edges in the network are omitted, which can accelerate the solving of SP problem.

After reducing and reconstruct the network, the shortest path between the source and destination nodes is divided into 3 parts: the inter-community route and the terminal routes on each end of the path. The terminal route in a shortest path problem means: 1. the path from the source node to the nearest community boundary; or, 2. the path from the nearest boundary to the destination node during the travel. The inter-community route is, therefore, the remaining part of route that travels across the communities.

We use a modified method to undertake the final calculation of shortest path:

a) We enumerate the distance from the source node i_{c1} to every boundary node in community c_1 and from the destination node j_{c2} to every boundary node in community c_2. Note that if the source node or the destination node happens to be a boundary node, this step can be skipped.

b) In order to navigate to the source and destination node, each possible terminal route is embedded into the new network.

c) We calculate the shortest path in the reduced network with the traditional Bellman–Ford or Dijkstra's algorithm. When the shortest path in the reduced network is ascertained, we return the whole path with detailed information about the nodes and edge weight, which was integrated to a few edges during the reduction and reconstruction of the network, and thus we now have solved the entire shortest path in the SP problem.

III. PERFORMANCE EVALUATION

To assess the performance of the proposed algorithm, we generate different types (Erdös–Rényi random graph and Barabási–Albert network) and scales (200, 500, 1500, 5000 nodes) of networks to inspect the efficiency of the proposed algorithm. In the assessment, we use unmodified Dijkstra's algorithm, Bellman–Ford algorithm and the proposed algorithm to calculate the distance between 50 randomly chosen pairs of source and destination nodes. The running time in each trial is recorded for performance assessment.

System environment declaration: The experiments were conducted on a Windows-based workstation with an Intel i7-7700k processor. To ensure that the trials are run with the same computing resource, the algorithms are edited, compiled, and run with single-thread Python scripts.

A. Performance in Single Shortest Path Query

The first phase of the experiment is designed to compare the efficiency of different algorithms when there is only one shortest path query. Therefore, when testing the modified algorithms, the time used in preprocessing steps should also be included in the running time. To compare the efficiency of different algorithms, we use the average ratio of the time spent in unmodified algorithm to the time spent in proposed algorithm to exhibit the improvement of the computational efficiency:

$$eff_D = \sum_{i=1}^{50} \frac{t_{Di}}{t_{mDi}} / 50 \tag{3}$$

$$eff_{BF} = \sum_{i=1}^{50} \frac{t_{BFi}}{t_{mBFi}} / 50 \tag{4}$$

where t_{Di}, t_{BFi}, t_{mDi}, and t_{mBFi} respectively represents the time spent in unmodified Dijkstra's algorithm, unmodified Bellman–Ford algorithm, the proposed modified Dijkstra's algorithm, and the proposed modified Bellman–Ford algorithm.

The results of the experiment are shown in Table I. We can see that the efficiency of the proposed algorithm is different in terms of the type and scale of the network. In general, the proposed algorithm is more efficient in solving SP problems in BA network, and even more efficient in networks with larger scale. However, when solving the SP problems in random graphs, the proposed algorithm is less efficient than the traditional Bellman–Ford and Dijkstra's algorithm. This is mainly because the community structure in ER graph is insignificant [34], and thus will result in considerable time usage and poor improvement in the calculating process. Therefore, we can find that when there is only one SP problem in one network, the community detection based-SP algorithm is efficient in large-scale BA network, while in ER graph the proposed algorithm will end up with performance degradation.

TABLE I. COMPARISON OF TRADITIONAL AND THE PROPOSED ALGORITHM IN SINGLE SHORTEST PATH QUERY

Type of networks	Scale of networks (number of nodes)	eff_D	eff_{BF}
Erdös–Rényi random graph (ER graph)	200	1.033	0.956
	500	0.862	0.809
	1500	0.507	0.483
Barabási–Albert network (BA network)	200	0.904	0.982
	500	1.254	1.051
	1500	1.902	1.559
	5000	2.380	2.047

B. Performance in Multiple Shortest Path Query

As is introduced above, the proposed algorithm consists of two process: preprocessing and calculating, and the preprocessing of the network does not change with the source and destination nodes. Therefore, a practical way to enhance the performance of the algorithm is to reuse the community information in each SP query [35][36]. To assess the performance in multiple SP query, we conduct the second phase of experiment, in which the time in preprocessing time and calculating are recorded separately.

In Fig. 1 we demonstrate the time consumption when the proposed algorithm is run on different scale of BA networks. The results in this figure are obtained with modified Dijkstra's algorithm and are normalized for comparison. From the results shown in Fig. 1, we can find that as the network scale increases, the preprocessing time accounts for larger proportion in the total time consumption. Therefore, in large-scale networks, the community detection will reduce the amount of calculation to a larger degree, which indicates that if the information obtained in the community detection can be reused, more potential of the proposed modified algorithm can be unleashed, particularly in large-scale networks.

In Fig. 2 we demonstrate the efficiency of the proposed modified Dijkstra's algorithm when it is used in multiple SP query in a 1500-node BA network. For ease of comparison, we modify the data, and take the processing time in one SP query with traditional Dijkstra's algorithm as a time unit.

From the results shown in Fig. 2, we find that the proposed algorithm can provide more improvement in calculation efficiency when more SP problems with different source and destination nodes are to be solved. As the number of SP problems increases, the average time to calculate each SP will decrease to a certain value, and theoretically the computational efficiency in multiple query eff_{nmD} satisfies:

$$\lim_{n_{sp}\to\infty} eff_{nmD} = eff_{mD} \times \frac{t_{cd}+t_{rr}+t_{cal}}{t_{cal}} \quad (5)$$

where n_{sp} represents the number of SPs to be calculated, t_{cd} represents the time in community detection of the network, t_{rr} represents the time in reduction and reconstruction of the network, and t_{cal} represents the time in calculating.

Another observation made in Fig 2 is that the performance of the proposed algorithm converges when there are more than 6 SP problems to solve. The threshold of the number of SPs,

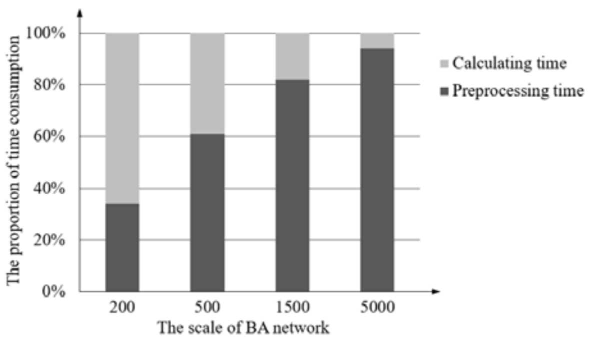

Fig. 1. Time consumption in proposed modified Dijkstra's algorithm.

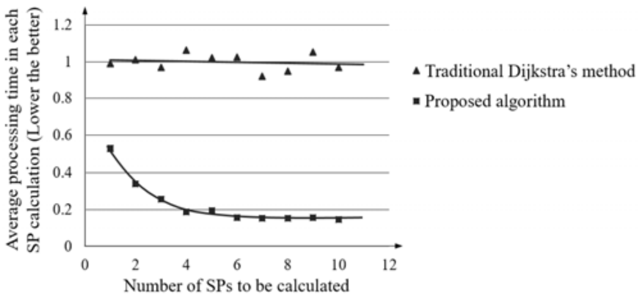

Fig. 2. Average processing time to solve each SP problem.

which makes eff_{nmD} approach its limit, might change, depending on the scale and topology of the network. However, this threshold is relatively small (i.e., easy to reach) comparing to the large quantities of calculation in real application scenarios of SP query, which indicates that the proposed algorithm is efficient in most of the practical application.

In the performance evaluation of the proposed algorithm, the beneficial outcomes brought by community detection are clearly observed. In single SP query, the proposed algorithm can only accelerate the calculation in middle or large-scale BA networks, while in multiple SP query, the proposed algorithm will produce considerable performance increase.

IV. Conclusion

The proposed shortest path algorithm, in this manuscript, combines community detection with the traditional Bellman–Ford and Dijkstra's algorithm, and has been proved to be efficient in processing SP query in large-scale BA networks, especially in scenarios where multiple SP queries are to be processed. Averagely, the proposed algorithm can increase the processing speed by 5%-3905%, depending on the scale and processing amount. In the proposed SP algorithm, the main purpose of utilizing community detection method is to integrate the tedious information about the topology and structure of the network, so that the network can be reduced, and the calculating process can be simplified. Furthermore, when processing multiple SP queries, the information in community structure can be reused, which can further enhance the efficiency of the proposed algorithm.

This study will open practical and straightforward extensions. Firstly, the community detection in SP algorithm is a so-called 'time-space trade-off' (i.e., exchange apace for time or vice versa) approach to improve the performance of the algorithm [37][38]. Therefore, to what extent can we preprocess the network, i.e., how much space can be sacrificed for efficiency, is an issue worth studying. Secondly, the effectiveness and performance of the proposed algorithm in special application scenarios that possesses different features, e.g., network of networks [39][40], dynamic network [41], are to be examined. Thirdly, this article proposes using community detection in SP algorithm, and developed an elementary implementation with classic method of community detection and SP calculation, thus further research can develop and examine other combinations of methods and algorithms to enhance the validity and efficiency of the algorithm further.

References

[1] A. Frechette, F. B. Shepherd, M. K. Thottan and P. J. Winzer, "Shortest Path Versus Multihub Routing in Networks With Uncertain Demand," in IEEE/ACM Transactions on Networking, vol. 23, no. 6, pp. 1931-1943, 2016.

[2] M. M. Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," (in English), IEEE transactions on neural networks, vol. 4, no. 6, pp. 941-54, 1993 1993.

[3] Y. Yang, A. N. Zincir-Heywood, M. I. Heywood, and S. Srinivas, "Agent-based routing algorithms on a LAN," Canadian Conference on Electrcial and Computer Engineering, vols 1-3, New York: IEEE, 2002, pp. 1442-1447.

[4] S. Yang and F. A. Kuipers, "Traffic Uncertainty Models in Network Planning," IEEE Communications Magazine, vol. 52, no. 2, pp. 172-177, Feb 2014.

[5] N. Isa, A. Mohamed, and M. Yusoff, Implementation of Dynamic Traffic Routing for Traffic Congestion: A Review. Singapore, 2015, pp. 174-186: Springer Singapore.

[6] H. W. Ferng, "Modeling of Split Traffic Under Probabilistic Routing," IEEE Communications Letters, vol. 8, no. 7, pp. 470-472, 2004.

[7] T. Zeume and T. Schwentick, "On the quantifier-free dynamic complexity of Reachability," (in English), Information and Computation, vol. 240, pp. 108-129, Feb 2015.

[8] J. Sankaranarayanan and H. Samet, "Query Processing Using Distance Oracles for Spatial Networks," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 8, pp. 1158-1175, 2010.

[9] M. E. Newman, "Scientific collaboration networks. I. Network construction and fundamental results," Phys Rev E Stat Nonlin Soft Matter Phys, vol. 64, no. 1 Pt 2, p. 016131, Jul 2001.

[10] R. E. Bellman, “On a routing problem,” Quarterly of Applied Mathematics, vol. 16, pp. 87–90.
[11] L. R. Ford Jr., and D. R. Fulkerson, Flows in Networks. Princeton, NJ, USA: Princeton Univ. Press, 1962, pp. 17-45
[12] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple Object Tracking Using K-Shortest Paths Optimization," IEEE Trans Pattern Anal Mach Intell, vol. 33, no. 9, pp. 1806-19, Sep 2011.
[13] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," Social Networks, vol. 30, no. 2, pp. 136-145, 2008.
[14] C. Sommer, "Shortest-path queries in static networks," ACM Computing Surveys, vol. 46, no. 4, pp. 1-31, 2014.
[15] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: theory and experimental evaluation," Mathematical Programming, vol. 73, no. 2, pp. 129-174, 1996.
[16] M. Gong, G. Li, Z. Wang, L. Ma, and D. Tian, "An efficient shortest path approach for social networks based on community structure," CAAI Transactions on Intelligence Technology, vol. 1, no. 1, pp. 114-123, 2016.
[17] Y. Gao, "Shortest path problem with uncertain arc lengths," Computers & Mathematics with Applications, vol. 62, no. 6, pp. 2591-2600, 2011.
[18] H. J. Caulfield and J. M. Kinser, "Finding the shortest path in the shortest time using PCNN's," IEEE Transactions on Neural Networks, vol. 10, no. 3, pp. 604-606, 1999.
[19] R. Albert, H. Jeong, and A.-L. Barabási, "Diameter of the World-Wide Web," Nature, vol. 401, p. 130, 09/09/online 1999.
[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, p. P10008, 2008.
[21] M. E. Newman, "Analysis of weighted networks," Phys Rev E Stat Nonlin Soft Matter Phys, vol. 70, no. 5 Pt 2, p. 056131, Nov 2004.
[22] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," Proceedings of the National Academy of Sciences, vol. 99, no. 12, pp. 7821-7826, 2002.
[23] C.-M. Liu, L.-G. Liu, and R. Pirjola, "Geomagnetically Induced Currents in the High-Voltage Power Grid in China," IEEE Transactions on Power Delivery, vol. 24, no. 4, pp. 2368-2374, 2009.
[24] F. Li et al., "Smart Transmission Grid: Vision and Framework," IEEE Transactions on Smart Grid, vol. 1, no. 2, pp. 168-177, 2010.
[25] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, "Community detection in Social Media Performance and application considerations," (in English), Data Mining and Knowledge Discovery, vol. 24, no. 3, pp. 515-554, May 2012.
[26] L. C. Leung, W. M. Cheung, and Y. Van Hui, "A framework for a logistics e-commerce community network: The Hong Kong air cargo industry," (in English), IEEE Transactions on Systems Man and Cybernetics Part a-Systems and Humans, Article vol. 30, no. 4, pp. 446-455, Jul 2000.
[27] D. Lalwani, D. V. L. N. Somayajulu and P. R. Krishna, "A community driven social recommendation system," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, 2015, pp. 821-826.
[28] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," New Journal of Physics, vol. 11, no. 3, p. 033015, 2009.
[29] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," Knowledge and Information Systems, vol. 42, no. 1, pp. 181-213, 2013.
[30] E. A. Leicht and M. E. Newman, "Community structure in directed networks," Phys Rev Lett, vol. 100, no. 11, p. 118703, Mar 21 2008.
[31] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," Physical Review E, vol. 69, no. 2, p. 026113, 02/26/ 2004.
[32] M. E. J. Newman, "Modularity and community structure in networks," Proceedings of the National Academy of Sciences, vol. 103, no. 23, pp. 8577-8582, 2006.
[33] X. Duan, C. Wang, Z. Li, and S. Shi, "Two Community Network Generation Algorithms," in 2009 WASE International Conference on Information Engineering, 2009, vol. 2, pp. 121-124.
[34] S. Fortunato and D. Hric, "Community detection in networks: A user guide," Physics Reports, vol. 659, pp. 1-44, 2016.
[35] Y. J. Na, S. Abdullaev, and F. I. S. Ko, "An Optimization of CDN Using Efficient Load Distribution and RADS Caching Algorithm," (in English), Journal of Universal Computer Science, Article; Proceedings Paper vol. 14, no. 14, pp. 2329-2342, 2008.
[36] Q. K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," (in English), Applied Mathematics and Computation, Article vol. 216, no. 3, pp. 830-848, Apr 2010.
[37] P. Beame, M. Saks, X. D. Sun, and E. Vee, "Time-space trade-off lower bounds for randomized computation of decision problems," (in English), Journal of the ACM, Article vol. 50, no. 2, pp. 154-195, Mar 2003.
[38] A. Aldroubi, J. Davis, and I. Krishtal, "Dynamical sampling: Time-space trade-off," (in English), Applied and Computational Harmonic Analysis, Article vol. 34, no. 3, pp. 495-503, May 2013.
[39] J. X. Gao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, "Robustness of a Network of Networks," (in English), Physical Review Letters, Article vol. 107, no. 19, p. 5, Nov 2011, Art. no. 195701.
[40] R. Q. Lu, W. W. Yu, J. H. Lu, and A. K. Xue, "Synchronization on Complex Networks of Networks," (in English), IEEE Transactions on Neural Networks and Learning Systems, Article vol. 25, no. 11, pp. 2110-2118, Nov 2014.
[41] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," (in English), Nature, vol. 464, no. 7291, pp. 1025-1028, Apr 2010.

First A. Author (M’76–SM’81–F’87) and the other authors may include biographies at the end of regular papers. Biographies are often not included in conference-related papers. This author became a Member (M) of **NAUN** in 1976, a Senior Member (SM) in 1981, and a Fellow (F) in 1987. The first paragraph may contain a place and/or date of birth (list place, then date). Next, the author’s educational background is listed. The degrees should be listed with type of degree in what field, which institution, city, state or country, and year degree was earned. The author’s major field of study should be lower-cased.

The second paragraph uses the pronoun of the person (he or she) and not the author’s last name. It lists military and work experience, including summer and fellowship jobs. Job titles are capitalized. The current job must have a location; previous positions may be listed without one. Information concerning previous publications may be included. Try not to list more than three books or published articles. The format for listing publishers of a book within the biography is: title of book (city, state: publisher name, year) similar to a reference. Current and previous research interests ends the paragraph.

The third paragraph begins with the author’s title and last name (e.g., Dr. Smith, Prof. Jones, Mr. Kajor, Ms. Hunter). List any memberships in professional societies other than the **NAUN**. Finally, list any awards and work for **NAUN** committees and publications. If a photograph is provided, the biography will be indented around it. The photograph is placed at the top left of the biography. Personal hobbies will be deleted from the biography.

Huixiong Wang was born in 1995. He received his B.S. degree in safety science from School of Reliability and Systems Engineering, Beihang University (BUAA), in 2018. He is currently a master student in the same school. His interests of research include systems engineering and complex network.
E-mail: wanghuixiong@buaa.edu.cn

Prof. Xing Pan was born in 1979. He received his B.S. degree in mechanical engineering, and Ph.D. degree in systems engineering from Beihang University (BUAA), Beijing, China, in 2000, and 2005, respectively. From 2005 to 2009, he was an assistant professor with the School of Reliability and Systems Engineering, Beihang University, Beijing, China. Since 2009, he has been an associate professor. From 2012 to 2013, he was a visiting scholar at the Department of Systems and Industrial Engineering, University of Arizona, Tucson, USA. His research interests include reliability engineering, systems engineering, and system risk analysis.
E-mail: panxing@buaa.edu.cn