Worst-case test network optimization for community detection methods

Iveta Dirgová Luptáková*, Marek Šimon, Jiří Pospíchal

***Abstract*—** In control and automation, applications of network community detection range from black-start recovery of power systems, truss structure manufacturing, neural motor control, water distribution, up to image segmentation for finding cracks in bridges. For such important technical applications, it is necessary to know the limits of various methods, how much they can deviate from optimum. Network community detection finds communities/clusters of densely connected nodes with few edges outside the cluster. The robustness of the community detection methods were already compared on a number of test networks, both real world and artificial. However, these test networks were implicitly average cases. Here we evolve networks, which produce a maximum error of modularity measure for selected methods of community detection. This worst-case test networks were evolved for Edge betweenness, Fast greedy, Infomap, Louvain, and Walktrap modularity detection. Such a comparison provides a tougher test of robustness than previous approaches. A bit surprisingly, after the Blondel's Louvain method, the next best result was provided by fast greedy, while otherwise favored Infomap fared rather poorly.

***Keywords*—**community detection; networks; stochastic optimization; clustering; worst-case test

I. INTRODUCTION

The amount of network or graph structured data increases exponentially with network structures found in all areas of human interest. Nodes represent computers, cellular phones, or web pages in the internet or telecommunications networks, people or blogs in social networks, electrical devices in power distribution, molecules in metabolic networks, neurons in neural networks, or persons in biological networks describing the spread of disease [1, 2]. Network analysis becomes crucial also in security [3-6].

Even when nodes are not characterized by other properties than those coming from the network topology, it is often useful to divide the nodes of networks into (in our case non-overlapping) parts (modules, groups, clusters, communities). Then we can deal with these parts more effectively separately, whether they are customers in a social network or a virus infected set of computers. These parts or communities often have different characteristics like node degrees, clustering coefficients, betweenness, etc. [2], which influence effectivity of the algorithm applied to them separately instead of applying them to the whole network. For this purpose, partitioning, clustering and/or community detection methods are used.

This research was supported by a grant SK-SRB-2016-0003 of Slovak Research and Development Agency.

Iveta Dirgová Luptáková, Marek Šimon, Jiří Pospíchal are with the Department of Applied Informatics and Mathematics, Fac. Nat. Sci.University of SS. Cyril and Methodius, Trnava, Slovakia, (e-mail: iveta.dirgova@ucm.sk)

There is mainly a formal difference between partitioning, clustering and community detection. In partitioning, the number and/or sizes of clusters usually should be known in advance. In clustering, one typically strives to get a cluster with a high clustering coefficient, which computation is based on the number of triangles. This is impossible to use e.g. for a network of heterosexual relationships, where triangles do not appear. Unlike graph partitioning or some of the clustering methods, community detection does not require advance knowledge of the number of communities or their sizes, and it can find partitions with communities of different sizes. In community detection, the decision about the number and size of the communities is made internally by the methods and their intrinsic parameters [1, 2].

Even though community detection is typically associated with social networks, it would be wrong to narrow our understanding of their applications to friendships in Facebook. Community detection has a wide variety of applications in control and automation, like black-start recovery of power systems [7], truss structure manufacturing [8], neural motor control [9], water distribution [10], up to image segmentation [11], used e.g. for finding cracks in bridges [12].

A number of criteria can measure the quality of the split of the network. These criteria can be borrowed from cluster analysis or partitioning, but none of these measures are considered dominant [13]. In community detection, the measures should reflect a satisfaction of the requirement, that within the clusters or communities the number of edges should be high, while edges connecting nodes from two communities are ideally scarce. Newman [14] developed modularity function Q, which is defined as the fraction of the edges inside communities minus the expected fraction if the edges were distributed at random. Q range is $[-1/2,1)$. This measure is widely accepted [15, 16], even though it can ignore very small communities and exhibits other limitations [17]. Therefore, we selected this Q measure for our worst-case test network optimization.

The quality of various community detection methods was already tested many times, using various quality measures both for real world networks [18] as well as for artificial benchmark networks or their mix [19-24].

II. COMMUNITY DETECTION METHODS

The community detection methods are based on a wide variety of principles. For our comparison, we planned to apply the most popular and widely used methods. Their time complexity is further expressed for the sparse networks with N nodes and E edges. They are already implemented in a

number of standard packages. A bit surprisingly, the packages often substantially differ in their speed, a good comparison can be found in [25]. Based on this comparison, we selected the igraph package [26].

The methods considered for comparison were following:

A. Edge betweenness

The algorithm was designed by Girvan and Newman [27], who adapted Freeman's betweenness centrality [28] to edge betweenness defined as the number of the shortest paths that go through an edge. The edges connecting communities are presumed to have a high edge betweenness, and these edges are removed to separate the communities. The algorithm's complexity is $O(E^2N)$.

B. Fast greedy

The algorithm was invented by Clauset *et al*. [29]. It starts with nodes as separate communities and repeatedly merges the pairs of communities, which give maximum improvement of modularity, until no improvement is possible. The algorithm's complexity is $O(N \log^2(N))$.

C. Infomap

The algorithm was proposed by Rosvall et al. [30, 31]. It is based on the map equation and uses the duality between finding community structure in networks and minimizing the description length of a random walker's movements on a network. The algorithm's complexity is O(E) [32].

D. Label propagation

The algorithm was developed by Raghavan et al. [33]. It starts with nodes as separate communities (labels), in a random order goes through the nodes and assigns them the label of the majority of its neighbors, until each node has the same label as the majority of its neighbors. The algorithm's complexity is $O(E)$.

E. Leading eigenvector

The algorithm was designed by Newman [34], using leading eigenvector of the modularity matrix to split the graph in two parts, so that modularity is maximized. The algorithm's complexity is $O(N^2)$.

F. Louvain

Otherwise called Multilevel was introduced by Blondel et al. [35]. It starts with nodes as separate communities and nodes (in further repetition communities) are moved to the community of its neighbor, which provides a maximum increase of modularity, until no improvement is possible. The algorithm's complexity is $O(N \log(N))$.

G. Optimal

Brandes et al. [36] maximize modularity measure using Integer Linear Programming. The algorithm has exponential time complexity, in practice good for networks with up to 50 nodes.

H. Spinglass

Reichardt and Bornholdt [37] based their search for communities on simulated annealing, which aims to keep nodes of the same community connected and nodes of different communities disconnected. The algorithm's complexity is $O(N^{3.2})$.

I. Walktrap

Pons and Latapy [38] started their algorithm from the nodes as separate communities, and connects the communities based on the fact, that random walker tends to be trapped in dense part of a network. The algorithm's complexity is $O(N^2 \log(N))$ [39].

III. Network optimization

In order to find out a worst-case test network for each tested community detection method, we have used a stochastic hillclimbing, starting with 5x5-lattice network. Surely, we could find even worse test network cases using initial networks with different number of nodes and/or edges, but our main goal was to compare the robustness of community methods and for these purposes, such a simple initial network was satisfactory. Results of Brandes Optimal method [36] was considered as the true Q measure modularity value (result of function `OptimalModularity(currentNet)`) and all tested algorithms were pitted against it. In the following pseudocode the results of modularity tested community detection method and currentNet network is depicted by `TestedComDetMod(currentNet)`. The current network is stochastically mutated or perturbed by sequentially going through all edges and with a probability $2/|E|$ deleting the edge and creating it between other two unconnected vertices in the function `NEIGHBOR(currentNet)`.

```
Stochastic Hill Climbing Worst Network Optimization
   currentNet = 5x5 lattice network;
   currentEval = OptimalModularity(currentNet)
                 -TestedComDetMod(currentNet);
   for iter = 1 to 1000
      nextNet = NEIGHBOR(currentNet);
      nextEval = OptimalModularity(nextNet)
                 -TestedComDetMod(nextNet);
      if (nextEval >= currentEval)
              currentNet = nextNet;
              currentEval = nextEval;
   return currentNode, currentEval;
```

Each method was tested ten times with different random seed. Unfortunately, it was not feasible to test thoroughly all the considered community detection methods. Label propagation and Leading eigenvector methods with standard settings achieved a huge error from the start, bundling the whole graph into one big community. This result remained through the whole optimization, so the test network evolved against Optimal method, which failed the purpose of optimization. The Spinglass method run into numerical difficulties within the thousands iteration. Therefore, these methods were excluded from further comparison.

The result of the Optimum community detection method on the initial 5x5-lattice network is shown in Fig. 1, with Q value 0.4740625. All the further tested methods, i.e. Edge

betweenness, Fast greedy, Infomap, Louvain, and Walktrap achieved similarly good results on this network.

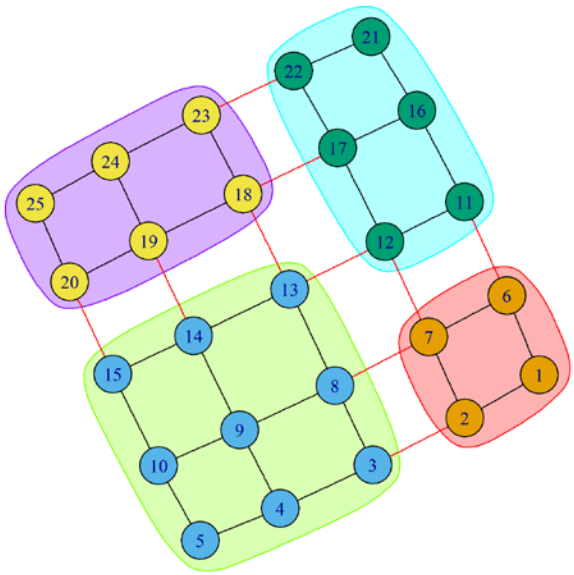

Fig. 1. Result of Optimum method for the initial 5x5 lattice network with Q value 0.4740625.

In Figs. 2-6 you can see the most extreme worst case test network optimization results from 10 runs of the optimization method, each with 1000 iterations. The methods are Louvain, Fast greedy, Edge betweenness, Walktrap, and Infomap, in sequence of the quality of their average error from 10 runs, with result of optimum modularity detection algorithm on the left and the tested modularity detection algorithm on the right hand side. Tab. 1 shows the statistical results of the maximization of error from 10 runs, providing means, standard deviation, minimum and maximum of errors of Q measure from the results after 1000 iterations.

TABLE I. ERROR MAXIMIZATION STATISTICS

Community detection method	**Mean of modularity error**	**Standard deviation of error**	**Min. of error**	**Max. of error**
Louvain	0.1124	0.0102	0.1000	0.1266
Fast greedy	0.1206	0.0155	0.1016	0.1547
Edge betweenness	0.1780	0.0396	0.0978	0.2503
Walktrap	0.1887	0.0192	0.1619	0.2225
Infomap	0.4088	0.0060	0.3997	0.4216

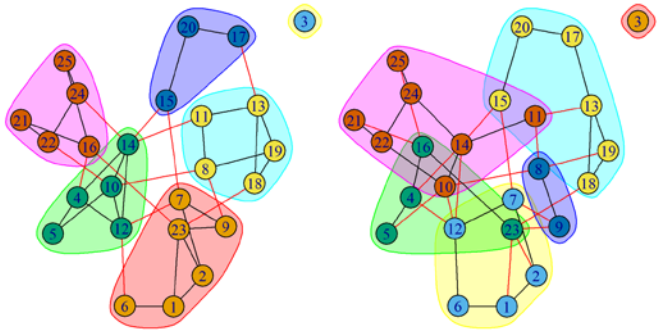

Fig. 2. Result of optimization for maximum Q error for Louvain method. Optimum method's modularity result is on the left with Q value 0.4534375. Louvain result is on the right hand side with Q value 0.326875.

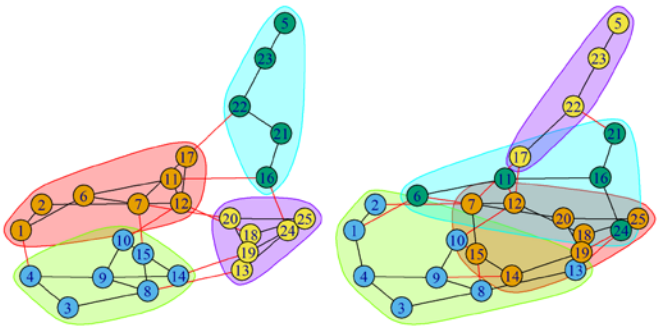

Fig. 3. Result of optimization for maximum Q error for Fast greedy method. Optimum method's modularity result is on the left with Q value 0.481875. Louvain result is on the right hand side with Q value 0.327188.

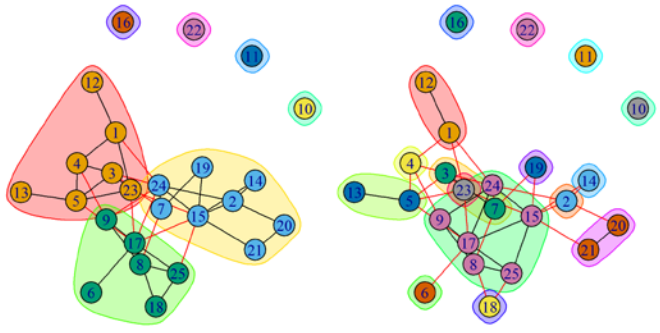

Fig. 4. Result of optimization for maximum Q error for Edge betweenness method. Optimum method's modularity result is on the left with Q value 0.3621188. Edge betweenness result is on the right hand side with Q value 0.111875.

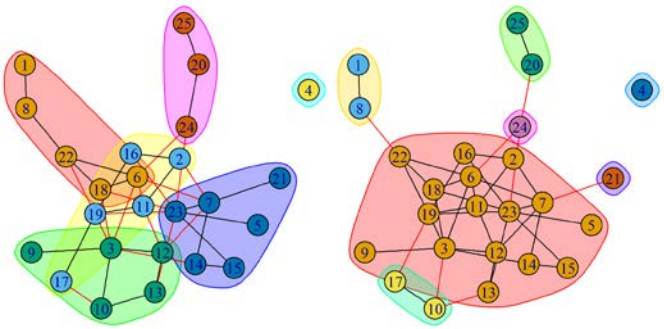

Fig. 5. Result of optimization for maximum Q error for Walktrap method. Optimum method's modularity result is on the left with Q value 0.354063. Walktrap result is on the right hand side with Q value 0.131563.

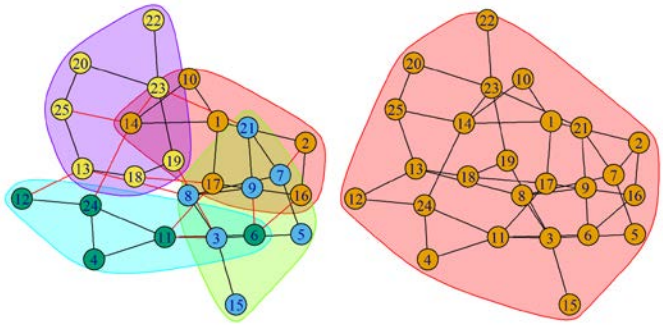

Fig. 6. Result of optimization for maximum Q error for Infomap method. Optimum method's modularity result is on the left with Q value 0.41563. Walktrap result is on the right hand side with Q value 0.

Further Figs. 7-11 show evolution of the maximum achieved error for Louvain, Fast greedy, Edge betweenness, Walktrap and Infomap methods during 1000 iterations. Mean, max and min differences from 10 runs show the evolved values of Q modularity differences between currently evaluated method and the Optimum methods. The upper green lines show mean values of Q from Optimum method, magenta lines show mean values of Q from currently evaluated method. While the Q values of Optimum method vacillates or goes slightly lower, the Q values of the compared method decreases more substantially. This corresponds to the red lines expressing these differences, which go up substantially.

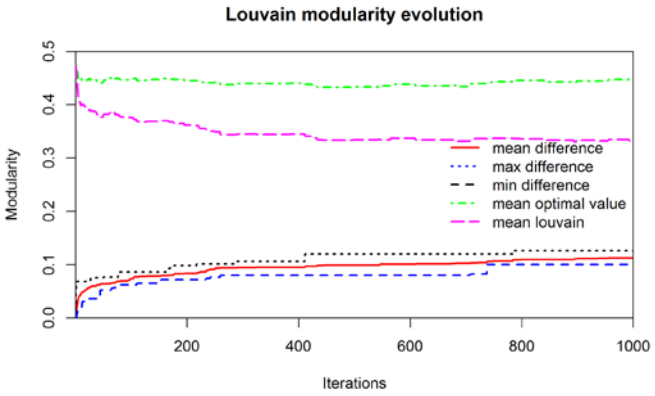

Fig. 7. Evolution of maximization of error for Louvain modularity method.

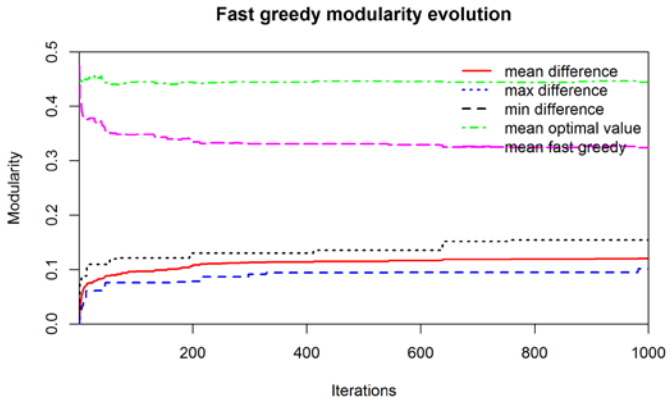

Fig. 8. Evolution of maximization of error for Fast greedy modularity method.

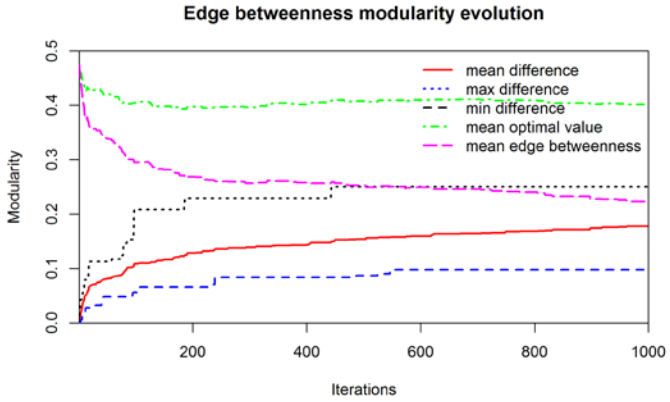

Fig. 9. Evolution of maximization of error for Edge betweenness modularity method.

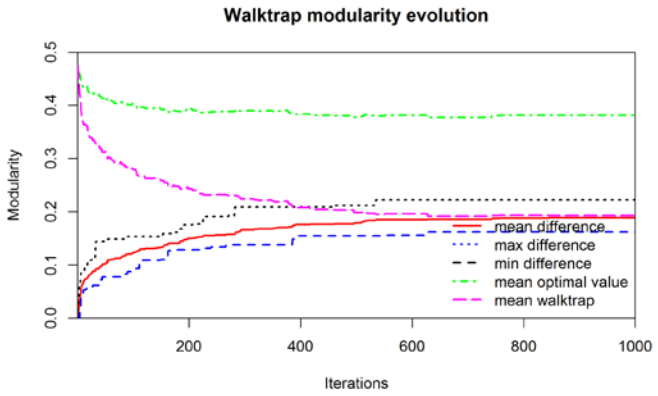

Fig. 10. Evolution of maximization of error for Walktrap modularity method.

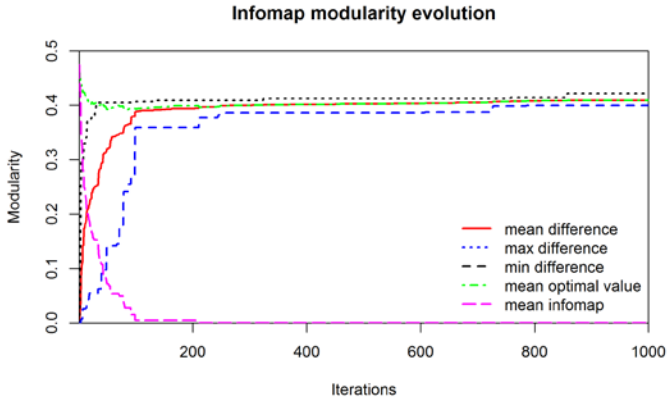

Fig. 11. Evolution of maximization of error for Infomap modularity method.

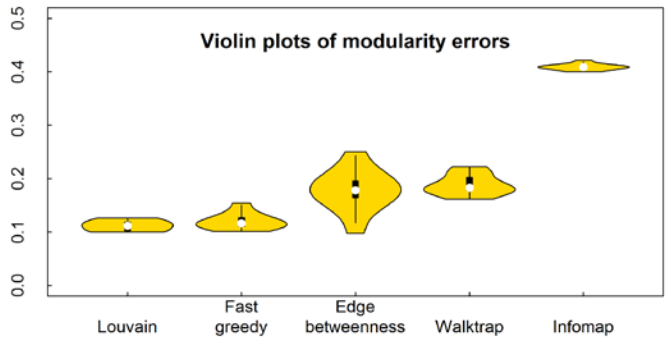

Fig. 12. Violin plots for results of 10 runs of maximization of error for the studied community detection methods. Even though Edge betweenness had a lower average of the modularity error than Walktrap, it had a higher maximum value.

We were also interested, whether the methods correlate in the evolved worst-case test networks. Therefore, we have measured *Q* values for the evolved worst-case test graphs shown in Figs. 3-6 also for modules found by all the other methods. The results are shown in Tab. 2. Except for the Walktrap worst-case test network, which also produced high *Q* error for the Infomap method, on average the worst test cases for single methods proved to be easy for the remaining methods, producing very low errors. Comparison of community detection methods for the worst cases

Worst case test graphs for method in the column (with maximum modularity error, shown in Figs. 3-6)					
method	Louvain	Fast greedy	Edge betw.	Walktrap	Infomap
Louvain	**0.1266**	0.0391	0.0000	0.0013	0.0147
Fast greedy	0.0291	**0.1547**	0.0000	0.0013	0.0147
Edge betw.	0.0100	0.0122	**0.2503**	0.1050	0.0269
Walktrap	0.0000	0.0300	0.0244	**0.2225**	0.0428
Infomap	0.0000	0.0000	0.0000	0.2238	**0.4216**

It would seem, that evolution of worst-case test network for Infomap provided regularly a network, for which Infomap produced a high value of error. However, during the tests we realized that unlike the other tested methods, Infomap quite heavily depends on a random seed and its results differ widely for the same network. Therefore, we took the network evolved as worst- case test for Infomap and run it through Infomap modularity detection one hundred times and calculated the *Q* values. The result can be seen in the violin plot in Fig. 13. The maximum *Q* value = 0.421563, mean *Q* value = 0.4024028, standard deviation of *Q* value =0.0477751 and the minimum = 0. This result shows that Infomap mostly results in a low *Q* error, but occasionally it quite fails. Our evolution of modularity error in Fig. 11 actually shows only these failures, when the Infomap produced a low error, it was considered as a test case not "bad enough" for saving, so it does not show on the graph. The evolved worst-case test network for Infomap shown in Fig. 6 is not actually bad; Infomap fails on it only occasionally.

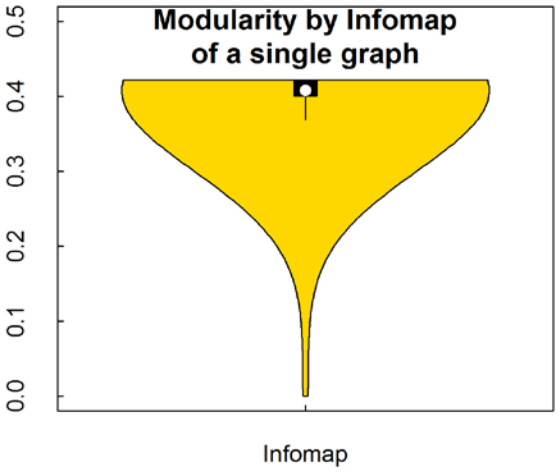

Fig. 13. Violin plot for *Q* values results of 100 runs of Infomap community detection method on the same evolved worst case test network. Infomap mostly finds a good *Q* value, it fails only seldom.

Nevertheless, the above analysis does not really excuses the Infomap method. If Infomap is used in some application just once, the result shows that it can occasionally fail and cannot be considered reliable for applications with high-consequence risks.

IV. Conclusions

We have evolved worst-case test case networks with a high *Q* error for a selection of most popular community detection methods. Rather surprisingly, otherwise favored Infomap method proved to be unreliable and should not therefore be used in applications with high-consequence risks. The best results were achieved by Louvain method, followed by Fast greedy. Optimization failed to find a network, for which these methods would fail; the methods produced *Q* value close to optimum. If the worst-case test networks optimized for one method were evaluated by the remaining methods, on average, near optimal *Q* values were found. Therefore, if reliable results are required, best result of a set of the methods should be used.

References

[1] M. Newman, Networks: An Introduction. Oxford University Press, 2010.

[2] A.-L. Barabási, M. Pósfai, Network Science. Cambridge University Press, 2016, see also http://barabasi.com/networksciencebook, last visit 7.4.2017

[3] P.A. Duijn, P.P. Klerks, "Social network analysis applied to criminal networks: Recent developments in Dutch law enforcement", in Networks and network analysis for defence and security, A.J. Masys, ed. Springer International Publishing, 2014, pp. 121-159.

[4] M. Korytar and D. Gabriska, "Integrated security levels and analysis of their implications to the maintenance," J. of Applied Mathematics, Statistics and Informatics, vol. 10(2), pp. 33-42, 2014.

[5] M. Šimon, L. Huraj, M. Čerňanský, "Performance Evaluations of IPTables Firewall Solutions under DDoS attacks," J. of Applied Mathematics, Statistics and Informatics, vol. 11(2), pp. 35-45, 2015.

[6] L. Huraj and V. Siládi, "Authorization through trust chains in ad hoc grids.," in Proceedings of the 2009 Euro American Conference on Telematics and Information Systems: New Opportunities to increase Digital Citizenship, p. 13. ACM, 2009.

[7] Y. Liu, T. Liu , Q. Li , and X. Hu, "Chapter 33. Power system black-start recovery subsystems partition based on improved CNM community detection algorithm," in Proceedings of the 2015 International Conference on Electric, Electronic and Control Engineering (ICEECE 2015), F. Shao, W. Shu, and T. Tian, Eds. CRC Press, 2015, pp. 183-189.

[8] H. Cao, R. Mo, N. Wan, F. Shang, C. Li, and D. Zhang, "A subassembly identification method for truss structures manufacturing based on community detection," Assembly Automation, vol. 35(3), pp. 249-258, 2015.

[9] A. Büschges and A. Borgmann, "Network modularity: back to the future in motor control," Current Biology, vol. 23(20), pp. R936-R938, 2013.

[10] O. Giustolisi and L. Ridolfi, "A novel infrastructure modularity index for the segmentation of water distribution networks," Water Resources Research, vol. 50(10), pp. 7648-7661, 2014.

[11] H. Hu, Y. van Gennip, B. Hunter, A.L. Bertozzi, and M.A. Porter, "Multislice modularity optimization in community detection and image segmentation," in Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on Data Mining, pp. 934-936, IEEE, December 2012.

[12] C.M. Yeum and S.J. Dyke, "Vision-Based Automated Crack Detection for Bridge Inspection," Computer-Aided Civil and Infrastructure Engineering, vol. 30(10), pp. 759-770, 2015.

[13] S.E. Schaeffer, "Graph clustering," Computer science review, vol. 1(1), pp. 27-64, 2007.

[14] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," Phys. Rev. E vol. 69, p. 026113, 2004.

[15] M. Chen, K. Kuzmin, B. K. Szymanski, "Community detection via maximization of modularity and its variants," IEEE Transactions on Computational Social Systems, vol. 1(1), pp. 46-65, 2014.

[16] N.P. Nguyen, T.N. Dinh, Y. Shen, and M.T. Thai, "Dynamic social community detection and its applications," PloS one, vol. 9(4), p. e91431, 2014.

[17] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," J. of Statistical Mechanics: Theory and Experiment, vol. 2005(09), p. P09008, 2005.

[18] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova, "Community detection in large-scale networks: a survey and empirical evaluation," Wiley Interdisciplinary Reviews: Computational Statistics, vol. 6(6), pp. 426-439, 2014.

[19] J. Leskovec, K.J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in Proceedings of the 19th international conference on World wide web, pp. 631-640. ACM, April 2010.

[20] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," Physical review E, vol. 78(4), p. 046110, 2008.

[21] A. Lancichinetti, "Evaluating the performance of clustering algorithms in networks," in Dynamics On and Of Complex Networks, vol. 2. Springer: New York, 2013, pp. 143-158.

[22] G.K. Orman and V. Labatut, "A comparison of community detection algorithms on artificial networks," in International Conference on Discovery Science. Springer: Berlin Heidelberg, 2009, pp. 242-256.

[23] G. Orman, V. Labatut, and H. Cherifi, "Qualitative comparison of community detection algorithms," arXiv preprint arXiv:1207.3603, 2012.

[24] Y. Zhao, R. Algesheimer, and C.J. Tessone, "A Comparative Analysis of Community Detection Algorithms on Artificial Networks," Scientific Reports, vol. 6, 2016.

[25] S. Wandelt and X. Sun, "Complex network analysis: The need for speed.," in Control Conference (CCC), 35th Chinese, pp. 1213-1218. IEEE, July 2016.

[26] G. Csardi and T. Nepusz, "The igraph software package for complex network research," InterJournal, Complex Systems 1695, URL http://igraph.org (2006).

[27] M. Girvan and M.E. Newman, "Community structure in social and biological networks," Proceedings of the National Academy of Sciences, vol. 99, pp. 7821–7826, 2002.

[28] L.C. Freeman, "Centrality in social networks conceptual clarification," Social Networks, vol. 1, pp. 215–239, 1979.

[29] A. Clauset, M.E. Newman, and C. Moore, "Finding community structure in very large networks," Physical Review E, vol. 70, p. 066111, 2004.

[30] M. Rosvall and C.T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks," Proceedings of the National Academy of Sciences, vol. 104, pp. 7327–7331, 2007.

[31] M. Rosvall, D. Axelsson, and C. T. Bergstrom, "The map equation," The European Physical Journal Special Topics, vol. 178, pp. 13–23, 2010.

[32] A. Mukherjee, M. Choudhury, F. Peruani, N. Ganguly, and B. Mitra, "Dynamics On and Of Complex Networks, Volume 2: Applications to Time-Varying Dynamical Systems," Springer Science & Business Media, 2013.

[33] U.N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Physical Review E, vol. 76, p. 036106, 2007.

[34] M.E. Newman, "Finding community structure in networks using the eigenvectors of matrices," Physical Review E, vol. 74, p. 036104, 2006.

[35] 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, p. P10008, 2008.

[36] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," IEEE transactions on knowledge and data engineering, vol. 20(2), pp. 172-188, 2008.

[37] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection," Physical Review E, vol. 74, p. 016110, 2006.

[38] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in Computer and Information Sciences-ISCIS 2005, pp. 284–293 Springer, 2005.

[39] J. Xie and B.K. Szymanski, "Community detection using a neighborhood strength driven label propagation algorithm," in Network Science Workshop, pp. 188–195. IEEE, 2011.