DOI: 10.15439/2024F7644 ISSN 2300-5963 ACSIS, Vol. 39

A network clustering method based on intersection of random spanning trees

László Hajdu[*][†], András London[‡], and András Pluhár[‡]
[*]Innorenew CoE, Livade 6a, SI-6310 Izola, Slovenia
[†]University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technologies
Glagoljaška 8, SI-6000 Koper, Slovenia, https://orcid.org/0000-0002-1832-6944
[‡]University of Szeged, Institute of Informatics, H-6726 Árpád tér 2, Hungary

***Abstract*—We use a special edge centrality measure for node clustering in complex networks. The measure is based on the ‘spanning tree intersection’ value motivated by previous work on the intersection and minimum expected overlap of random spanning trees in complex networks. First, we show that this new metric differs from some well-known edge centralities on random network models and real-world networks. Then, we show the applicability of the metric for clustering the nodes and point out some advantages over some other edge centrality based hierarchical clustering methods.**

I. Introduction

IN NETWORK science, paths and shortest paths between network nodes are of extraordinary interest due to the enormous number of straightforward applications [1], [2]. Besides the applications and algorithm development [3], [4], several metrics have been derived from paths and shortest paths, such as betweenness and closeness centrality, k-path centrality, etc., and used for different purposes. For example, some community detection algorithms have efficiently used path-based edge centrality measures, e.g., [5], [6], [7].

A widely used centrality based on shortest paths is *edge betweenness*, which measures the importance of an edge in a network by the number of shortest paths that pass through the edge. A shortest path between two nodes in an unweighted network is a path that has the minimum number of edges. The edge betweenness of an edge is then calculated as the fraction of the shortest paths between all pairs of nodes in the network that pass through that edge. The importance of edges with high edge betweenness is obvious. They are vital in connecting different network parts and facilitating communication and information flow.

On the other hand, as many real-world situations show, there is “something artificial about shortest paths. It seems that shortest paths are sometimes too short and do not correspond to the underlying natural logic of the network", see [1] for more details. Information or traffic does not always prefer the shortest paths, but it sometimes also takes much longer paths. Furthermore, betweenness centrality, a widely used metric that is based on shortest paths, is unstable: adding just one ‘shortcut’ link to the network can dramatically change the scores of edges in the network [8].

Such considerations have led to an alternative model of edge importance based on randomly selected spanning trees [9], [10]. *A spanning tree* of a connected network of n nodes is a tree composed of $n-1$ edges such that they connect all n nodes in the network. Many results support the intuition of using spanning trees to compute edge centrality. In social networks, the spread of information follows tree-like cascades [11]; in technological (e.g., roads, electricity) and transaction networks (e.g., financial transactions, trade), the routes used often show tree-like structures [12].

Spanning Centrality is a measure of the importance of an edge in a network based on the number of spanning trees that contain the edge. The spanning centrality of an edge is calculated as the proportion of all spanning trees that contain the edge. Edges with high spanning centrality are essential because they are involved in many different paths through the network. This means they play a vital role in connecting other parts of the network and facilitating communication and information flow. Spanning centrality can be used to analyze the structure of networks and identify bottlenecks or critical points that could disrupt the flow of information or communication [13].

Recently it has been discussed that the average intersection (i.e., the number of common edges) of random spanning trees encodes some structural information about the network regarding homogeneity [14] and resilience [15], and also closely related to several concepts like *fairest edge usage*, *spanning tree modulus* and *secure broadcasting* over networks, see [14], [16]. The idea of checking that an edge of a network will likely be in the intersection of randomly chosen spanning trees then comes naturally.

In this short paper, we introduce the concept of *spanning tree intersection edge centrality* with the goal of using it in top-down hierarchical clustering of network nodes. This is motivated by at least two things. One is to use a metric, instead of betwenness, which is faster to calculate, but possibly provide a similar clustering. The other is to see how much it gives different clusters than the usual clustering or community detection procedures.

Throughout this paper, $G = (V, E)$ will be a finite, connected, undirected, and unweighted graph representing a network with $|V| = n$ nodes and $|E| = m$ edges.

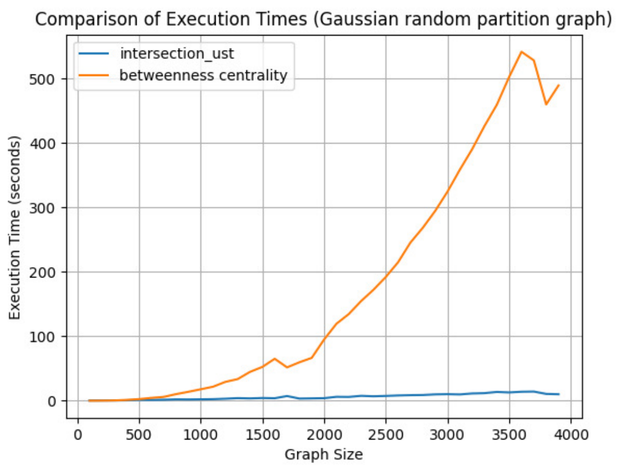

Fig. 1. Running time comparison on Gaussian random partition network.

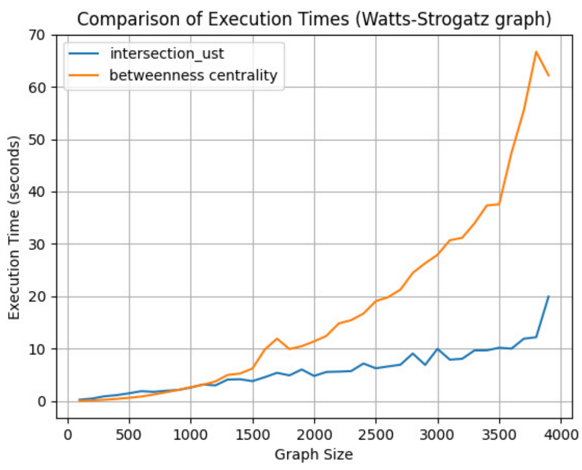

Fig. 2. Running time comparison on Watts-Strogatz network.

II. Definitions and methods

A. Edge centrality measures

Given a network G, edge centrality is a function that assigns a number to each edge of the network. Next, we define the three edge centrality measures we want to compare and briefly discuss the algorithms' complexity for calculating them.

Edge betweenness: Let σ_{ij} be the number of shortest paths between nodes i and j of an undirected and connected graph $G = (V, E)$, and $\sigma_{ij}(e)$ be the number of shortest paths between them that pass through edge $e \in E$. Note that there can be more than one shortest path between any two nodes in the case of unweighted graphs. The betweenness centrality $BC(e) \in (0, 1]$ of edge e is defined as $BC(e) = \sum_{i \neq j} \sigma_{ij}(e)/\sigma_{ij}$. The greater the number of shortest paths that pass through a particular edge, the greater the importance of that edge. Intuitively, a high $BC(e)$ identifies edges that act as bridges or intermediaries between nodes in the network and are essential for maintaining efficient communication and information flow between different parts of the network.

Spanning tree centrality: Given an undirected and connected graph $G = (V, E)$, the spanning centrality $SC(e) \in (0, 1]$ of an edge $e \in E$ is defined as the fraction of spanning trees of G that contain e. Intuitively, a high $SC(e)$ quantifies how important the edge e is for G to ensure its connectivity. An edge with a high SC value is present in most spanning trees, all of which will fall apart if the edge is removed from G. In the case where $SC(e) = 1$, G is broken when e is removed, i.e., e is a cutting edge if and only if $SC(e) = 1$. This means that such an edge participates in all possible spanning trees. Important edges participate in many spanning trees, assuming that spanning trees encode candidate paths through which information flows.

Spanning tree intersection centrality: The *spanning tree intersection* centrality value $STI(e) \in (0, 1]$ of an edge $e \in E$ is the fraction of spanning tree pairs among all such pairs that both contain e, i.e., e is in the intersection of the pair. In other words, $STI(e)$ shows how likely an edge will be in the intersection of randomly chosen spanning tree pairs. Intuitively, STI is very similar to SC but more restrictive in giving a high value for an edge. It also holds here that $STI(e) = 1$ if and only if e is a cutting edge. The relevancy of edges with high STI values can be seen from the perspective of spanning tree overlap. These are the edges that are somehow crucial in order to get a not empty overlap between randomly chosen spanning trees.

According to [15], the minimum of the expected number of edges in the intersection of two randomly chosen spanning trees is $(n-1)^2/m$, while the expected value can be calculated precisely as $\sum_{e \in E} p^2(e)$, where $p(e)$ is the probability that the edge e is in a uniform random spanning tree. It suggests (not dealing with the variance) that the probability of an edge e being in the intersection of randomly chosen spanning trees is $\approx p^2(e)$. In this sense, STI can be derived from SC, but we argue that it can provide some added value to it, and more restrictive in assigning high values to the edges.

B. Algorithms

Betweenness centrality can be computed efficiently, e.g., using the Brandes algorithm, in $O(nm)$ time.

The number of spanning trees of G is explicitly known by Kirchhoff's matrix tree theorem [17] and can be computed as the product of the positive eigenvalues of the graph Laplacian L divided by n. It can be calculated in polynomial, $(O(n^3))$, time (let $\tilde{L}$ be the Laplacian with be the Laplacian with row and column i^{th} removed (for any i), then $\det\tilde{L}$ be the number of spanning trees). This allows us to precisely define *uniform random spanning trees*, i.e., the uniform probability distribution over all spanning trees.

For an edge $e \in E(G)$, we can compute the probability $p(e)$, that the edge e is in a uniform random spanning tree of G, in polynomial time. This is $p(e) = \det(\tilde{L}_{G\setminus\{e\}})/\det\tilde{L}$, where $\tilde{L}_{G\setminus\{e\}}$ is defined similarly as above for the graph $G \setminus e$, the graph obtained by contracting the original one after the edge e is deleted. So, the nominator is the number of spanning trees containing e.

Although we can determine the probability of picking a uniform spanning tree, it does not follow that we can generate one. Moreover, calculating determinants is still expensive. Several algorithms have been developed to generate random spanning trees of an undirected graph; see, for instance, Broder's [18] and Wilson's [19] algorithms, worst case $O(nm)$, but on average $O(n \log n)$ time. Moreover, almost linear time algorithms exist [20]. That allows us to perform Monte-Carlo-style experiments, described in detail next.

Figure 1 shows the running times of the betweenness centrality and our spanning tree intersection centrality in a Gaussian random partition network. The network was generated with the size parameter n varying from 100 to 4000 in increments of 100. Each network was constructed with sub-community sizes $s = n/10$ and $v = n/20$, and probabilities of intra-community and inter-community connections set at 0.7 and 0.001, respectively. We also generated Watts-Strogatz networks where the size parameted varied from 100 to 4000 in increments of 100. For each network, we set the number of nearest neighbours in the ring topology (k) to 4 and the rewiring probability (p) to 0.1. The corresponding running times can be seen on Figure 2.

The running times for many other random network models have a similar shape. Our choice of the two models we present was based on the motivation of using the metric for community detection purposes, and these two models provide well-structured networks with community structure and other small-world properties that appear in real-world networks.

III. Experimental results

A. Experiment design

The main objective of the simulation was to explore the behavior of the two spanning tree-based centrality measures and compare them with the well-known and widely used edge betweenness centrality. As we noted the direct connection between SC and STI before, we restrict the presentation of the comparison results to those between BC and STI. To perform a detailed comparison, we collected real-world networks with varying properties and a curated set of random model networks. In the following section, we present the selection of networks we used in the simulation.

Selection of random and real-world networks: During the initialization of the experiment, we generated four different types of networks with the following parameters:

1) Random regular graph with parameter $d = 3, 4, \dots, 9$ (degree of each node) and number of nodes $n = 100, 200, 500$ [21], [22].
2) Erdős-Rényi random graph with parameter $p = 0.1, 0.2, \dots, 1$ (probability of edge creation) and number of nodes $n = 100, 200, 500$ [23], [24].
3) Preferential attachment random network using the barabasi_albert_graph function from NetworkX with parameter $m = 1, 2, \dots, 10$ (number of edges to attach from a newly created node to existing nodes) and number of nodes $n = 100, 200, 500$ [25].
4) Watts-Strogatz network using the watts_strogatz_graph function from NetworkX with parameters $k = 4$ (connected nearest neighbors in the ring topology) and $p = 0.1, 0.2, \dots, 1$ (probability of rewiring the edges) and number of nodes $n = 100, 200, 500$ [26].

Algorithm 1 Random Spanning Tree Simulation (RSTS)

```
1:  Input: Graph $G(V, E)$, sample size $s$
2:  $j \leftarrow 0$
3:  $\forall e \in E : C_e^{\text{intersection}} = 0$
4:  While $j < s$
5:      $H1 = \text{WilsonRST}(G)$
6:      $H2 = \text{WilsonRST}(G)$
7:      $I = \text{intersection}(H1, H2)$
8:      For $e$ in $E$
9:          If $e \in I$: $C_e^{\text{intersection}} \leftarrow C_e^{\text{intersection}} + 1$
10:     End For
11: End While
12: $\forall e \in E : C_e^{\text{intersection}} \leftarrow C_e^{\text{intersection}} / s$
```

For each type and parameter, we generated 100 networks and then calculated sample means and standard deviations.

In the case of real-world networks, we selected 15 well-known networks from the field of network science with various graph properties. The list of the selected graphs and their important basic properties can be seen in Table 1.

Tools and Libraries: The simulation environment was implemented in Python 3.9 using the following libraries. NetworkX (2.6.3) library was used for the betweenness centrality calculation, random network generation, and other network manipulation. Our own modified version of the open-source library DPPy (0.3.2) [27] was used to extract the spanning trees. However, the Wilson algorithm for the random spanning tree generation (sample method) remained unchanged. (The original library could not return the network graph containing the random spanning tree itself.). We visualized the results using the Matplotlib (3.5.2) and Seaborn (0.12.1) libraries. Pandas (1.4.2) was used for basic data manipulation, while Numpy (1.22.3) was used for basic calculations regarding the results.

Simulation Environment: The pseudocode shows the steps of our Random Spanning Tree Simulation environment. In the case of a network where we had multiple connected components, we always chose the largest connected component.

We generated a $2s$ number of random spanning trees during the simulation. The algorithm counts how often it is in the intersection of two random spanning trees. To calculate the expected value of being the intersection, we divide the result by the sample size for each edge. Throughout our experiment, we used $s = 1000$ as the sample size parameter in the case of both random and real networks (except for the real Email-Enron network, due to its large size, where we used $s = 100$).

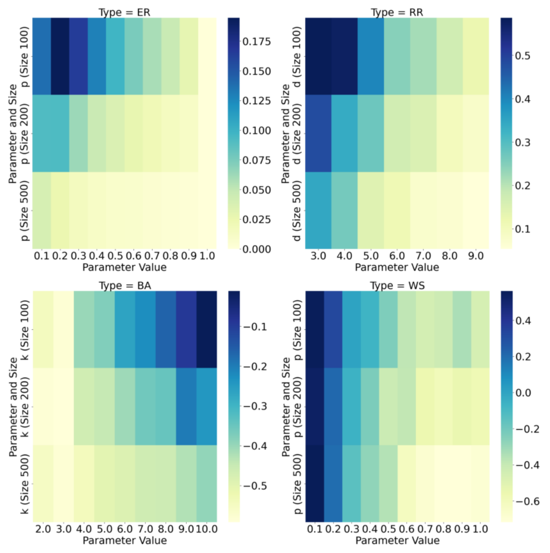

Fig. 3. Correlations between Intersection Probability and Edge Betweenness measures.

B. Comparison on random network models

As mentioned above, we generated 100 networks for each type and parameter. To gain valuable insights from the simulation, we took the mean $STI(e)$ and $BC(e)$ values (i.e., “intersection probabilities” and edge betweenness) for each edge, computed their correlations for each graph, and then aggregated them across the different parameters, sizes, and graph types to compare the patterns and behavior of the different measures.

The heatmap in Figure 3 shows the correlation between intersection probability and edge betweenness. In this case, in the case of Erdős-Rényi networks, no significant pattern can be observed. In the case of Barabási-Albert networks, there is a negative correlation with increasing size and k parameter values. The values are between -0.01 and -0.58. It can be seen that in the case of size 500, the values are between -0.39 and -0.55, so there is a negative correlation even at higher k-parameters. Random regular and Watts-Strogats networks again behave similarly for small d and p parameters. However, at higher parameter values, the correlation becomes negative in the case of WS. On the other hand, the random regular shows a slight decrease in correlations with decreasing size and parameter, with values between 0.05 and 0.59.

C. Comparison on real-world networks

This section presents, analyzes, and interprets our results for the real-world networks shown in Table 1 and introduces the behavior of the different measures on an arbitrary real network (as Figure 4 shows). The table briefly overviews various network characteristics and metrics for the given set of real-world networks. These metrics include the number of nodes, edges, network density, clustering coefficient, average path length, expected intersection (equals to $(n-1)^2/m$),

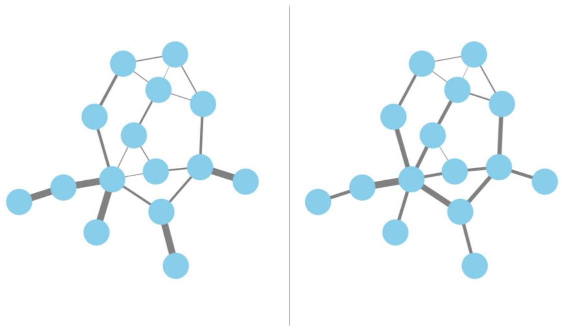

Fig. 4. Comparison intersection probability measure with the edge betweenness on the Florentine graph. (left: Intersection Probability, right: Edge Betweenness) Thicker edge refers to higher edge centrality.

observed intersection (observed intersection value based on the simulations), std (standard deviation), the adjusted index

$$RTI = \frac{\text{observed mean} - \text{min.expected}}{n - \text{min.expected}},$$

modularity, and the same correlation pairs as in the case of the random network in the previous section. More precise definitions of the calculated properties can be found in [28].

As an example shown in Figure 4, the thickness of the edge shows the actual value of the given measure; more precisely, the thicker the edge between two nodes, the higher the corresponding measure of the edge.

Table 1. shows the results on each of the real networks we used during the evaluation. The results show that our intersection probability and spanning tree probability measures have a relatively higher correlation with edge betweenness in the case of “Florentine", “Adjnoun", “Jazz" and “C-elegans" networks. As the previous section shows, our two measures are also highly correlated with each other in real networks, which means they express similar properties of the edges in the network.

The relationship between the correlation between spanning tree intersection centrality and betweenness centrality with Newman modularity (left) and RTI (right), respectively, on the investigated real-world dataset is shown in Figure 5. It suggests that the modularity and RTI are higher if the correlation between the two different centralities is high and lower in the case of a lower correlation. It would be worth investigating this effect in the future, as it may help us better understand the mesoscale structure of networks (more details will be given below).

IV. Community detection based on edge importance

A well-known community detection algorithm proposed by Newman and Girvan [6] uses centrality indices to find community boundaries. It is assumed that communities or groups are only loosely connected by a few edges between groups. Therefore, all shortest paths between communities must be along one of these few edges. Then, the edges connecting communities should have a high edge betweenness.

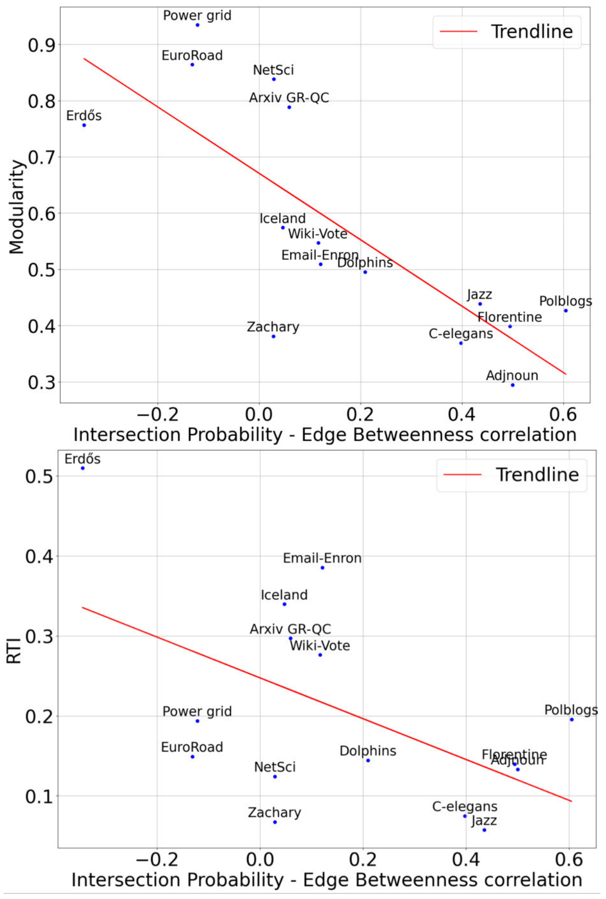

Fig. 5. Scatter plot that shows the connection of Intersection Probability - Edge betweenness correlation with Modularity measure and RTI.

By iteratively removing these edges, the communities can be separated at different hierarchical levels.

A natural idea is to change edge betweenness to spanning centrality, spanning tree intersection centrality, or a metric derived from these. This should be done carefully, as dangling edges (connected to 1 degree nodes) always have STI (and SC) values equal to one (see Figure 4). To handle such nodes, one can perform a preprocessing step that removes these nodes after automatically assigning the label of their only neighbor (which label will be assigned later in the process), or a unique label, depending on the specified requirement for the clustering algorithm. After this step, the edges are removed one by one according to the decreasing order of STI values.At the end of the process, all edges are removed. If we think of this process as a top-down hierarchical procedure, then the cutters are defined as the connected components of the network at a particular hierarchical level. The pseudocode of the algorithm is given below. Fig. 4 shows a benchmark example with the clusters found by the algorithm. Since the cluster structure is determined at a particular hierarchical level (i.e. after removing a certain number of edges to get separate components of the network), the level should be specified. This can be done, for instance maximizing the Newman modularity function [29], that can be calculated for any given clustering.

Algorithm 2 Iterative edge removal based on RTI values

```
1: Input: Graph $G(V,E)$,
2: Repeat
3:     For $e$ in $E$
4:         If $RTI(e)=1$, $e=(u,v)$ is a dangling edge with
           $d(v)=1$ then $C(v)=C(u)$, remove $v$
5:         End If
6:         If $RTI(e) > \max(RTI)$ then
7:             $\max(RTI) = RTI(e)$
8:             $\operatorname{argmax}(STI) = e$
9:         End If Remove $e$ from $G$
10:    End For
11: Until there is no more edge in $G$
12: Output: cluster dendogram
```

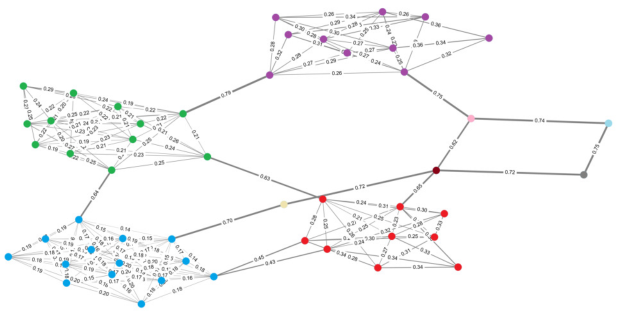

Fig. 6. Clustering given by the RTI value based top-down hierarchical clustering method. The network was generated by the gaussian_random_partition_graph() method from the NetworkX package.

V. Summary

This paper introduced an edge centrality metric based on random spanning tree intersections. We presented an algorithm to compute it and compared it with the widely used edge betweenness and spanning centrality metrics. Our initial results suggest that this metric may be helpful in determining essential links of the network in terms of path usage, connectivity and resilience. We also experimented that this metric is efficiently applicable for clustering the nodes and have some advantages over some other edge centrality based top-down hierarchical clustering methods. We also hypothesize that a metric derived from betweenness and spanning tree intersection centrality could help to optimize the modularity in the Girvan-Newman algorithm. The exploration of this can be the topic of a future work.

Acknowledgement. András London was supported by National Research, Development and Innovation Office—NKFIH Fund No. SNN-135643. László Hajdu acknowledges the European Commission for funding the InnoRenew CoE project (Grant Agreement no. 739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European Regional Development Fund). He is also grateful for the support of the

Slovenian Research Agency (ARRS) through grant N2-0171 and for the support of University of Primorska through postdoc grant No. 2991-10/2022.

References

[1] A. Gulyás, Z. Heszberger, and J. Biró, *Paths: Why is life filled with so many detours?* Springer Nature, 2021, doi:10.1007/978-3-030-47545-1.

[2] A. Gulyás, Z. Heszberger, J. Biró, A. Gulyás, Z. Heszberger, and J. Biró, "The universal nature of paths," *Paths*, pp. 45–65, 2021.

[3] A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. Basalamah, "A survey of shortest-path algorithms," *arXiv preprint arXiv:1705.02044*, 2017.

[4] L. M. Laskov and M. L. Marinov, "List of pareto optimal solutions of a biobjective shortest path problem," in *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2023, pp. 603–613, doi:10.15439/2023F3718.

[5] A. Bóta, M. Krész, and A. Pluhár, "Dynamic communities and their detection," *Acta Cybernetica*, vol. 20, pp. 35–52, 2011.

[6] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002, doi:10.1073/pnas.12265379.

[7] P. G. Sun and Y. Yang, "Methods to find community based on edge centrality," *Physica A: Statistical Mechanics and its Applications*, vol. 392, no. 9, pp. 1977–1988, 2013, doi:10.1016/j.physa.2012.12.024.

[8] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39–54, 2005, doi:10.1016/j.socnet.2004.11.009.

[9] X. Qi, E. Fuller, R. Luo, and C.-q. Zhang, "A novel centrality method for weighted networks based on the kirchhoff polynomial," *Pattern Recognition Letters*, vol. 58, pp. 51–60, 2015, doi:10.1016/j.patrec.2015.02.007.

[10] A. S. Teixeira, P. T. Monteiro, J. A. Carriço, M. Ramirez, and A. P. Francisco, "Spanning edge betweenness," in *Workshop on Mining and Learning with Graphs*, vol. 24, 2013, pp. 27–31.

[11] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 4, pp. 1–37, 2012.

[12] S. Lämmer, B. Gehlsen, and D. Helbing, "Scaling laws in the spatial structure of urban road networks," *Physica A: Statistical Mechanics and its Applications*, vol. 363, no. 1, pp. 89–95, 2006, doi:10.1016/j.physa.2006.01.051.

[13] C. Mavroforakis, R. Garcia-Lebron, I. Koutis, and E. Terzi, "Spanning edge centrality: Large-scale computation and applications," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 732–742, doi:10.1145/2736277.2741125.

[14] N. Albin, J. Clemens, D. Hoare, P. Poggi-Corradini, B. Sit, and S. Tymochko, "Fairest edge usage and minimum expected overlap for random spanning trees," *Discrete Mathematics*, vol. 344, no. 5, p. 112282, 2021.

[15] A. London and A. Pluhár, "Intersection of random spanning trees in small-world networks," in *International Conference on Complex Networks and Their Applications*. Springer, 2022, pp. 337–345, doi:10.1007/978-3-031-21131-7_26.

[16] K. Kottegoda, *Spanning tree modulus and secure broadcast games*. PhD dissertation, Kansas State University, 2020.

[17] E. W. Weisstein, "Matrix tree theorem," *https://mathworld.wolfram.com/*, 2000.

[18] A. Z. Broder, "Generating random spanning trees," in *FOCS*, vol. 89, 1989, pp. 442–447, doi:10.1109/SFCS.1989.63516.

[19] D. B. Wilson, "Generating random spanning trees more quickly than the cover time," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 296–303, doi:10.1145/237814.237880.

[20] A. Schild, "An almost-linear time algorithm for uniform random spanning tree generation," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 2018, pp. 214–227.

[21] J. H. Kim and V. H. Vu, "Generating random regular graphs," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '03, New York, NY, USA, 2003, p. 213–222.

[22] A. Steger and N. C. Wormald, "Generating random regular graphs quickly," *Combinatorics, Probability and Computing*, vol. 8, no. 4, p. 377–396, 1999, doi:10.1017/S0963548399003867.

[23] P. Erdős and A. Rényi, "On random graphs i." *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959, doi:10.1515/9781400841356.38.

[24] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141–1144, 1959.

[25] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, 1999, doi:10.1126/science.286.5439.509.

[26] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998, doi:10.1038/30918.

[27] G. Gautier, G. Polito, R. Bardenet, and M. Valko, "DPPy: DPP Sampling with Python," *Journal of Machine Learning Research - Machine Learning Open Source Software (JMLR-MLOSS)*, 2019, http://github.com/guilgautier/DPPy/.

[28] A. London and A. Pluhár, "Intersection of random spanning trees in complex networks," *Applied Network Science*, vol. 8, no. 1, p. 72, 2023.

[29] M. E. Newman, "Modularity and community structure in networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006.

TABLE I
NETWORK STATISTICS AND RESULTS FOR SOME REAL-LIFE NETWORKS

name	nodes	edges	density	*cluster ing*	*avg path length*	expected	observed	std	RTI	*modular ity*	*IntersectP EdgeB*	*SptreeP EdgeB*
Florentine	15	20	0.1905	0.1915	2.4857	9.80	10.527	1.007	0.140	0.3987	0.4945	0.5278
Zachary	34	78	0.1390	0.2557	2.4082	13.96	15.313	2.368	0.067	0.3807	0.0283	-0.0135
Dolphins	62	159	0.0841	0.3088	3.3570	23.40	28.980	2.847	0.145	0.4955	0.2093	0.2414
Iceland	75	114	0.0411	0.1573	3.1996	48.04	57.191	2.351	0.340	0.5742	0.0470	0.0099
Adjnoun	112	425	0.0684	0.1569	2.5356	28.99	40.050	3.970	0.133	0.2947	0.4998	0.4174
Jazz	198	2742	0.1406	0.5203	2.2350	14.15	24.759	3.926	0.058	0.4389	0.4355	0.4458
C-elegans	297	2148	0.0489	0.1807	2.4553	40.79	59.926	5.598	0.075	0.3692	0.3979	0.2964
NetSci	379	914	0.0128	0.4306	6.0419	156.33	184.004	7.553	0.124	0.8383	0.0287	0.0298
Wiki-Vote	889	2914	0.0074	0.1273	4.0962	270.61	441.456	9.888	0.276	0.5470	0.1165	0.0726
EuroRoad	1039	1305	0.0024	0.0353	18.3951	825.63	857.468	6.501	0.149	0.8640	-0.1318	-0.1317
Polblogs	1222	16717	0.0224	0.2260	2.7375	89.18	310.507	10.450	0.195	0.4269	0.6050	0.6123
Arxiv GR-QC	4158	13428	0.0016	0.6289	6.0494	1286.91	2139.144	22.535	0.297	0.7886	0.0592	0.0784
Erdős	4991	7428	0.0006	0.0420	5.5116	3352.19	4187.384	16.113	0.510	0.7568	-0.3451	-0.3903
Power grid	4941	6594	0.0005	0.1032	18.9892	3700.88	3940.939	16.066	0.194	0.9346	-0.1219	-0.1167
Email-Enron	33696	180811	0.0003	0.0851	4.0252	6279.23	16845.420	52.275	0.385	0.5092	0.1210	0.1029