IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Scalability of Betweenness Approximation Algorithms: An Experimental Review

SEBASTIAN WANDELT[1], XING SHI[1], and XIAOQIAN SUN[1]
[1]School of Electronic and Information Engineering, Beihang University, Beijing, China

Corresponding author: X. Sun (e-mail: sunxq@buaa.edu.cn).

This study is supported by the Research Fund from National Natural Science Foundation of China (Grants No. 61861136005, No. 61601013, No. 61851110763, No. 71731001).

ABSTRACT Betweenness centrality, which measures the contribution of an individual node to network's connectivity by counting the number of shortest paths a node appears in, is widely used for the analysis of complex networks. The computation of exact betweenness centrality is prohibitively expensive for large networks, given a worst-case complexity of $O(N * E)$, where N is the number of nodes and E is the number of edges in the network. Accordingly, a multitude of approximation algorithms have been proposed in the literature. Obtaining an overview on the state-of-the-art is difficult, given a combination of numerous algorithms, parameters and network topologies.

In this paper, we report on the results of the probably largest benchmark performed in this field. Specifically, we select 100 networks with distinct topologies and scales, covering various domains. We devise and compare eight selected measures to evaluate the accuracy of approximation, compared to exact betweenness computation. All experiments, including those to obtain the exact betweenness values, have been performed on one computer using a single thread, in order to provide a fair comparison. We implemented typical approximation methods and report sensitivity analysis results with a variety of parameters. We find that a uniformly random sampling method, one of the earliest proposed methods in this field, still delivers the best performance; nicely addressing a sweet spot between quality and runtime complexity. In addition, we carried out robustness experiments based on the ranking order of approximated betweenness, in order to show the effect of different approximations on a real-world task. Our study aims at being a reference for choosing a betweenness approximation method, with consideration of network type, required level of accuracy and available computational resources.

INDEX TERMS Complex networks, Network topology, Approximation algorithms, Experimental review, Betweenness Centrality

I. INTRODUCTION

In an increasingly connected world, network science is widely used way to analyze and improve real-world systems; covering a wide range of fields, including biology [1], [2], transportation [3], [4] and other critical infrastructure [5]. Systems are modeled as graphs, where nodes represent system components and links model interactions/dependencies between components. One fundamental question in network science is to assess the centrality of nodes, with the goal to quantify the importance of each node. Throughout the last decades, many local and global node centrality measures have been developed, including degree centrality [6], closeness centrality [7], eigenvector centrality [8], Katz centrality [9] and betweenness centrality [10]. The betweenness centrality [10] of a node v refers to the fraction of shortest paths which pass through v compared to all shortest paths between each source node s to each target node t. [11] proposed another betweenness measure that counts all paths between nodes based on random walks. Intuitively, the betweenness centrality measures the relevance of a node for transportation of information, goods, or other flows through the network. Empirical studies have used betweenness centrality to analyze protein-protein interaction, to evaluate traffic in communication networks, to evaluate the importance of a node [12], and to identify

hubs in transportation networks [13]. Moreover, recent research suggests that betweenness centrality is of great importance for studying the robustness of networks [14], and spreading phenomena. Compared to other simple centrality measures (e.g., degree), betweenness centrality does not appear to admit a simple expression in terms of the adjacency matrix [15].

The computation of betweenness centrality is time consuming for very large networks, given that all pairs of shortest paths need to be evaluated for exact betweenness. [16] proposed a classic algorithm to obtain exact values of all nodes based on executing Breadth-First Search (BFS) from each node. The algorithm takes $O(N * E)$ runtime on unweighted graphs, where N is the number of nodes and E is the number of edges in the network. The time complexity is prohibitive for analyzing large networks with millions of nodes on nowadays' hardware. Accordingly, several methods for computing approximations of node betweenness have been proposed in the literature, in order to reduce the number of computation steps compared to exact betweenness. Most of these methods use sampling techniques, considering shortest paths of a subset of node pairs or limit the maximum length of shortest paths, only taking into account pairs of nodes within a certain distance.

The plethora of methods proposed for estimating betweenness centrality in networks, together with a wide range of parameters trading approximation quality for execution speed, has led to a rather incomprehensible research landscape on algorithms, implementations, and their effectiveness on real-world problems. This paper aims to bring order into betweenness approximation by comparing a large set of competing instances on a number of real-world networks, with a focus on the algorithms' scalability. In order to ensure coverage of different network structures, we choose a large experimental network data set that contains 100 real-world networks and covers a wide variety of fields. The size of the selected networks ranges from 20 nodes to 2,216,688 nodes, covering heterogeneous topologies. For a thorough comparison of methods, we choose eight different measures to evaluate the accuracy of approximation, given the that estimation of approximation quality largely depends on the application and the induced constraints. For reference purposes, we obtained the exact betweenness of each node for each network; a process which took 46 days for the largest network in our study. All experiments were performed on the same hardware (a high-end workstation with 40 cores and 450 GB RAM, running Fedora 28), in order to allow for a fair comparison.

Compared to existing studies, the unique, distinctive properties of our work is to 1) perform all experiments on the same hardware, 2) execute all experiments in a single-threaded fashion, 3) report results on the largest number of networks (=100) from different domains with more than one million nodes, and 4) consider a rich set of eight evaluation measures. Altogether, the unique combination of features, makes our study contribute to the literature as a reference benchmark, guiding the choice of betweenness centrality methods, given requirements on accuracy and limited computational resources.

The remainder of this study is structured as follows. Section II describes the fundamental problem, solution methods and experimental setup used in our study, including the devised measures for approximation quality. Section III reports on our experiments for a collection of 100 real-world networks across various different domains. Section IV summarizes the insights obtained in our study and recommends some directions for future work.

II. METHODS

A. THE FORMAL PROBLEM OF NODE BETWEENNESS CENTRALITY

We revisit the formal definition of betweenness centrality and a widely-used algorithm for computing the exact betweenness values of all nodes in a network first. The betweenness centrality [17] of node i in network G is defined as :

$$B(i) = \sum_{s,t \in G, s \neq t \neq i} \frac{\sigma_{st}(i)}{\sigma_{st}}$$

where σ_{st} is the number of all shortest paths from node s to node t, and $\sigma_{st}(i)$ is the number of shortest paths from node s to node t that pass through node i. Computing betweenness centrality values of all nodes takes quadratic time regarding the number of nodes in the network. [16] developed an approach based on Breadth-First Search (BFS) to obtain betweenness centrality values. For a pair of nodes $s, t \in G$, Brandes defined the pair-dependency $\delta_{st}(i) = \frac{\sigma_{st}(i)}{\sigma_{st}}$ and dependency of node $s \in G$ on $i \in G$, denoted by $\delta_{s\bullet}(i) = \sum_{t \in G} \delta_{st}(i)$. Then the betweenness centrality of node i: $B(i)$ can be written as:

$$B(i) = \sum_{s \neq i \neq t \in G} \delta_{st}(i) = \sum_{s \in G, s \neq i} \delta_{s\bullet}(i)$$

Brandes proved that the dependency of s on any $i \in G$ obeys:

$$\delta_{s\bullet}(i) = \sum_{w : i \in P_s(w)} \frac{\sigma_{si}}{\sigma_{sw}} (1 + \delta_{s\bullet}(w))$$

where $P_s(w)$ is a set of parents of node w in the BFS Directed Acyclic Graph (DAG). Based on the above observations, Brandes designed a two-step algorithm:

1) Run BFS from source node s to compute σ_{st} and $P_s(t)$ for all target nodes t;

2) Execute reverse BFS to compute $\delta_{s\bullet}(i)$.

Brandes algorithm, abbreviated as Betw in the following sections, requires a BFS from every node in the network. Thus, the time complexity is (at least) quadratic in the number of nodes and prohibitive for large networks with millions of nodes. In fact, for dense networks, the worst

case time complexity of the algorithm is cubic in the number of nodes. In order to avoid calculations of BFS from every node, several betweenness approximation methods have been proposed in the literature.

B. TECHNIQUES FOR BETWEENNESS CENTRALITY APPROXIMATION

The majority of betweenness approximation methods use sampling to reduce the execution cost. Here, sampling is the process of computing shortest paths of a subset of node pairs only. The strategies for sampling important nodes or pairs of nodes are diverse, and constitute the essential difference between approximation methods. In addition, there are several methods that use bounded traversal to limit the maximum length of shortest paths and only consider pairs of nodes within a certain distance. In general, algorithms for betweenness approximation can be categorized into the following three classifications:

1) **Pivots sampling algorithms:** Conduct BFS from selected nodes, referred to as pivots and compute dependencies of selected nodes.
2) **Node pairs sampling algorithms:** Sample pairs of nodes and compute pair dependencies.
3) **Bounded traversal algorithms:** Conduct BFS with a stop condition and compute pair dependencies of two nodes with a certain distance.

Related work regarding these three classes of algorithms are described below.

Pivots sampling Based on Brandes' algorithm, [18] modified the algorithm to run BFS only on a subset of nodes; considering the betweenness contribution of selected source nodes and finally re-scale the approximated betweenness values based on the sample size. Besides, they evaluated the performance of different strategies for choosing the subset of nodes: randomly sampling, selecting nodes with high degree, nodes with maximum distance and nodes with minimum distance. Their results showed that sampling nodes at random is the best strategy for approximation. [19] used progressive sampling to approximate betweenness of a given node v. Their method utilizes an adaptive sampling technique [20], which keeps sampling pivots until the sum of the dependencies $\delta_{s\bullet}(v))$ of each sampled pivot s on the given node v is larger than $5 * N$. They estimated the values of betweenness of Top-1% nodes and computed the mean percentage approximation error. Their method performs well on some real-world networks, including road networks and biological networks. [21] sampled a subset of nodes at random and they balanced the betweenness contribution of nodes at different distances from the source node. They suggested that the nodes close to the source node are overestimated when scaling up by the same factor. Moreover, they used a linear function to adjust the imbalance caused by different distance of nodes. They performed experiments with different constant numbers of samples. [21] compared their results with [18] on Euclidean distance, inversion numbers and runtime. They found that their algorithm showed better performance than [18] for all their tested data sets on the measures they chose.

Node pairs sampling

[22] is based on random sampling of shortest paths instead of samples nodes. The approach guaranteed that the error of each node is less than ϵ with probability at least $1 - \delta$, referred to as $(\epsilon, \delta) - approximation$, assuming that sample size did not depend on the size of the network and using Vapnik-Chervonenkis dimension (VC-dimension) [23] to determine the sample size. They evaluated accuracy of their method by computing average estimation error and the maximum error. Their results showed that their method is fast and scalable with certain approximation guarantees. [24] proposed an algorithm for unweighted and weighted networks which guarantees to bound the absolute error for betweenness approximation in large evolving networks. Moreover, [25] introduced a fully-dynamic algorithm which is suitable for disconnected networks. These two methods sample pairs of nodes and determine the sample size by VC-dimension [23]. [26] presented ABRA based on progressive sampling. Rademacher Averages [27] and pseudodimension [28] from statistical learning theory were used. ABRA keeps sampling pairs of nodes (s, t) until reaching a stop condition which is determined by statistical learning theory. Given an (ϵ, δ), they estimated betweenness for all nodes and they analyzed runtime, sample size and accuracy. [29] introduced an adaptive algorithm KADABRA for betweenness approximation based on a balanced bidirectional Breadth-First Search (bb-BFS). Besides estimating betweenness of all nodes, KADABRA can also approximate the top-k betweenness set.

Bounded traversal

[30] examined the relationship between the exact betweenness of node v in a network and the betweenness of node v in its ego network. Experiments showed that ego betweenness could be a good measure for approximating betweenness values on real-world networks. [31] elaborated on the idea of bounded traversal and proposed one algorithm for approximate betweenness. They used Pearson Correlation, Top-10-Hits and Top-1-Hits to verify the approximation quality and they also evaluated runtime. Their results showed that k-centrality measures could quickly approximate betweenness values in $O(N)$ time for networks with constant average degree.

Other algorithms

Some methods use online sampling to identify nodes with high betweenness. [32] estimated the k most central nodes based on online sampling. [33] made use of online sampling to find high centrality nodes in social networks. Except for online sampling, [34] suggested sampling nodes with different probability instead of the same probability $\frac{1}{N}$ (uniform).

C. METHODS AND PARAMETERS ASSESSED IN THIS STUDY

Based on a comprehensive survey of approximation algorithms, we selected five representative approximation methods which can make nice trade-offs between runtime and accuracy on large networks with millions of nodes for this study: Two methods based on sampling pivots, one method based on sampling pairs of nodes, and two methods based on bounded traversal. We describe the selected methods and the chosen parameter ranges next.

We selected RAND1 [18] which was proposed by Brandes. Instead of running BFS from every node in the network, RAND1 only considers dependencies from a subset of source nodes. Besides, these source nodes in the selected subset, referred to as pivots, are sampled uniformly at random. After obtaining dependencies $\delta_{s\bullet}(i)$ for each pivot s in the selected subset P, RAND1 approximates the betweenness centrality values by multiplying dependencies of pivots by $\frac{N}{|P|}$, where N is the number of nodes in the network and $|P|$ is the number of selected pivots. Moreover, Brandes evaluated other strategies for sampling (e.g., selecting nodes with highest degree) and compared these strategies with RAND1. The results of [18] revealed that RAND1 performs better than other strategies in all tested networks. However, the number of pivots can affect the accuracy of approximation. On the one hand, selecting few pivots will reduce accuracy, on the other hand, selecting a multitude of pivots will result in redundant calculations. To implement RAND1, we chose different number of pivots including 1, 2, 4, 8, 16, 32, 64, 128 and 256. However, such algorithm overestimates betweenness of unimportant nodes that happen to be near a selected pivot, as RAND1 scales up $\delta_{s\bullet}(i)$ by the same factor $\frac{N}{|P|}$. Therefore, we chose the number of sample pivots up to 256 and do not sample more pivots.

Another method for sampling pivots is RAND2 [21]. In order to address the overestimation problem of RAND1, RAND2 reduces the contributions for nodes near to the pivot: The betweenness contribution of a pivot s depends linearly on the distance to the pivot s, that is, these nodes closer to the pivot s take lower contributions. Results of [21] indicated that using the linear scaling function $f(x) = x$ can achieve better approximation, where x is the distance from the pivot. While, because RAND2 samples pivots uniformly at random, the number of pivots can affect the results. As [21] suggests, we have evaluated the performance of RAND2 method by setting different number of pivots to be {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096}.

The classic algorithm RAND1 [18] offers good quality compared to other methods (i.e. KPATH [31] and EGO [30]) on some tested networks. Though [21] found that RAND2 showed better performance than RAND1, from both theoretical and experimental perspective, we still consider RAND1 [18] for evaluation on our larger tested data sets, making this experimental study complete and comprehensive.

Besides sampling pivots, sampling of shortest paths can also avoid prohibitively expensive computation. We chose DIAM algorithm presented in [22]. This algorithm estimates betweenness based on vertex-diameter of the network. DIAM computes pair dependencies of node pairs instead of dependencies of selected pivots to approximate betweennes centrality values. [22] suggested that the sample size should be determined by the vertex-diameter $VD(G)$ instead of the number of nodes. More specifically, given a network G with its $VD(G)$ and the allowed additive error ϵ with probability $1-\delta$, DIAM determines the resulting sample size by the following formula:

$$r = \frac{0.5}{\epsilon^2}([\log_2(VD(G)-2)]+1+\ln\frac{1}{\delta}) \quad (1)$$

However, the computation of $VD(G)$ is as expensive as the computation of the exact betweenness centrality values. To address this problem, [22] presented a linear-time constant-factor approximation algorithm. Such algorithm considers SSSP (single-source shortest path) problem from a node v that is selected uniformly at random, and set $VD(G)$ to be the sum of the lengths of the two longest shortest paths from node v. After approximating $VD(G)$, DIAM uses Formula 1 to obtain the required sample size. Then, DIAM keeps sampling node pairs until sample size is satisfied. In our tests, we set ϵ to be {0.02, 0.03, 0.07, 0.1, 0.2, 0.3} and set δ to be 0.1 and 0.2.

These methods we selected above are using sampling and considering a subset of nodes or shortest paths. Moreover, we considered two algorithms which are based on bounded traversal. Such methods sample pairs of nodes within a specific distance and compute pair dependencies to approximate the betweenness centrality values. [31] elaborated the idea of using k-centrality measures as approximations for betweenness centrality values. They stated that nodes distant from v have small effects on the betweenness centrality value of v. Their algorithm KPATH [31] works similarly to Betw [16] but all traversals are bounded by k steps, that is, only pair dependencies of node pairs with shortest-k-paths contribute to the betweenness centrality values. Above all, the choice of k could greatly affect the results: if we set k extremely large, it can take long runtime; if we set k intensely small, these local features can lead to poor quality. For the implementation of KPATH, we used the available implementation by the authors in NetworKit library [35]. Furthermore, there are two parameters of KPATH need to be adjusted:

1) $\alpha_2 \in [-0.5, 0.5]$: it determines the trade-off between runtime and precision by setting the number of iterations to be proportional to $N^{1-2\alpha_2}$.
2) k: it determines the steps of bounded traversals.

We set k to be {4, 8, 16, 32}. [36] set k to be $0.2VD(G)$. Based on the algorithm for approximation of $VD(G)$ [22] above, we also choose $k = 0.2VD(G)$ as a parameter setting. Besides, considering the α_2 settings,

TABLE 1: Overview on the selected methods of betweenness approximation evaluated in this study. [a] KPATH_0.5_0.2: use the algorithm for approximation of $VD(G)$ in [22].

Method	Parameters	Description
Betw [16]		Computing exact betweenness centrality values.
RAND1 [18]	$\|P\| \in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$	Sampling pivots uniformly at random.
RAND2 [21]	$\|P\| \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096\}$	Uniform random sampling and linearly upscaling.
DIAM [22]	$\epsilon \in \{0.02, 0.03, 0.07, 0.1, 0.2, 0.3\}$, $\delta \in \{0.1, 0.2\}$	Sampling node pairs based on $VD(G)$ and dependencies.
KPATH [31]	$\alpha_2 \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, $k \in \{4, 8, 16, 32, 0.2VD(G)^a\}$	Considering shortest-k-paths
EGO [30]	$\alpha_1 \in \{-0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$	Computing betweenness values in the ego network.

we set it to be in $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ to make comprehensive comparison.

Finally, another method of bounded traversal we selected is EGO [30]. The ego network of a node is the network consisting of this node (ego) together with the neighbors and all the edges among those nodes. [30] explored the correlation between the betweenness of node i in its ego network and the exact betweenness of i in the whole network. Their results showed that there is a strong relationship between the local betweenness of i in its ego network and the exact betweenness of i. Though computing the betweenness of nodes in the ego network with direct neighbors is simple, it could not achieve favorable accuracy for the betweenness values of all nodes in the ego network. Everett and Borgatti suggested that using all neighbors with distance two can get better results. Thus, we implemented EGO based on KPATH but set k = 2. As for the settings of α_1 which sets the sample size to be proportional to $N^{1-2\alpha_1}$, we set it to be {-0.2, -0.1, 0.0, 0.1, 0.2, 0.3, 0.4, 0.5}.

All the selected methods, which were evaluated in this study, together with their parameter settings and descriptions are shown in Table 1. We have five types of approximation methods and different parameter settings, yielding 58 competitors in total. For each competitor, the naming scheme is chosen as: method_parameter (e.g., DIAM_0.03_0.1 is DIAM method with $\epsilon = 0.03$ and $\delta = 0.1$).

D. MEASURES FOR ESTIMATING THE QUALITY OF BETWEENNESS APPROXIMATION

The selection of measures for estimating approximation quality depends on the application requirements. BeBeCA [36] used four measures: Top-1%-Hits, Kendall correlation coefficient, max error and average error. However, in most practical applications, people are more concerned with the identification of nodes with high betweenness and the sorting of nodes. As there are many nodes with betweenness = 0 and most approximation algorithms can effectively identify these nodes, the average error could be underestimated. Therefore, we kept Top-1%-Hits and Kendall correlation coefficient and devised additional six measures. Our eight measures are summarized as follows:

1) **Top-1%-Hits**: it is defined as the number of nodes identified by the method in the top 1% nodes with the highest betweenness.
2) **Top-10%-Hits**: it represents how many nodes the method can identify in the top 10% nodes.
3) **Inversion**: it measures the sortedness of an estimated sorting with the exact sorting as a standard. That is, for node i and node j with $B(i) > B(j)$, if the approximation betweenness value $APB(i) < APB(j)$, it contributes 1 to inversion number. We normalize the exact inversion number inv as Inversion = $(1 - 2 * inv/(N * (N - 1)))$, where N is the number of nodes. The normalized value is referred to as Inversion below. It can be computed in $O(Nlog(N))$ runtime.
4) **Kendall's tau coefficient** [37]: Kendall's tau coefficient, referred to as Kendall, measures the correspondence between approximate betweenness ranking with exact betweenness ranking.
5) R^2 **(R-square)**: it is a classic correlation coefficient in statistics and it shows the effect of fitting estimated ranking with exact ranking.
6) **MaxRankingError**: we define MaxRankingError as: $1 - max(|r(i) - apr(i)|)/(N-1)$, where $r(i)$ is the index of node i in the exact betweenness ranking and $apr(i)$ is the index of such node in approximation betweenness ranking. Thus, the closer the value is to 1, the better the method performs.
7) **Weightedtau**: it is a weighted version of Kendall's tau coefficient. An element with rank r is mapped to weight 1/(r+1) and an exchange between two elements with rank r and s has weight 1/(r+1)+1/(s+1). Thus, top ranked elements have higher weight and the exchanges between top ranked elements are more influential.
8) **Wilcoxon**: it is the value of the Wilcoxon signed-rank test. The higher it is, the less both ranking orders match. We normalized Wilcoxon values among all competitors.

To sum up, we have chosen eight measures for estimating the quality of betweenness approximation. Besides, all of these eight measures are normalized. That is, higher values (closer to one) represent more accurate methods.

We use a small network example to illustrate the

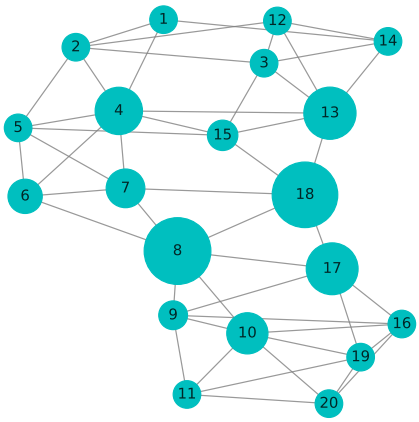

FIGURE 1: Small network example with 20 nodes and 49 edges (ENZYMES_g364).

computation and significance of these eight measures. Figure 1 shows a simple cheminformatics network ENZYMES_g364 with 20 nodes and 49 edges. We set the size of each node to be proportional to its exact betweenness in this visualization. The exact ranking of nodes from higher exact betweenness to lower exact betweennness is R1=[8, 18, 13, 17, 4, 10, 7, 6, 15, 9, 5, 2, 19, 16, 3, 14, 1, 12, 11, 20]. If we use RAND1_4 (sampling four pivots), we can get an estimated ranking R2=[8, 4, 17, 18, 10, 7, 6, 9, 13, 5, 15, 14, 19, 3, 2, 12, 1, 11, 16, 20]. If we only consider Top-1%-Hits (only one node in this small network), this method can get 100% accuracy as it correctly identify Node 8. However, the value of Top-10%-Hits is 0.5. Compared with the exact ranking R1, the exact value of inversion number inv is 22 and the normalized Inversion is $(1 - 2 * inv/(N * (N - 1)))$ = 0.884. The Kendall value is 0.347 and the Weightedtau value is 0.583, representing the correspondence of R1 and R2. Besides, we can see that the ranking of node 13 in R2 is 9th but it should be 3rd. Thus, the MaxRankingError is (1 - 6/19) = 0.684. Obviously, if R2 is the same as R1, the MaxRankingError will be 1 (the best and no error). R-square of R2 and R1 is 0.202, which represents correlation coefficient value. The exact value of Wilcoxon is 66.5 and the larger this exact value is, the worse such approximation method performs. Normalized Wilcoxon can be obtained by comparing such exact values among all methods.

E. EXPERIMENTAL SETUP

In our study we report the performance of all approximation methods regarding one hundred real-world networks of different sizes and diverse structures. All of our data sets can be downloaded from Network Repository [38]. Our data sets cover a total of 15 categories, which are described below:

1) **Social** (12 networks): networks representing the social interactions between individuals. Nodes are are individuals and links represent their interactions, e.g. friendships are follower-property. The largest social network in our data sets is soc-youtube-snap with 1,134,890 nodes and 2,987,624 edges, which contains all links of user-to-user.
2) **Biological** (10 networks): networks appearing in biological systems. Nodes are elements in these systems and links encoded their interactions. We selected networks from different biological systems, include one human network with 9,186 nodes and 31,038 edges. The largest biological network in our data sets is a worm network, bio-CE-CX with 15,063 nodes and 245,862 edges.
3) **Brain** (10 networks): such networks show functional connectivity in brains. We selected brain networks of mouse, cat, macaque and fly. The brain network of fly: bn-fly-drosophila_medulla_1 is the largest among these brain networks with 1,770 nodes and 9,010 edges.
4) **Citation** (3 networks): such networks contain paper sources that linked by co-citation relationships. However, there are very few data sets on Citation networks and we used three of them. The largest one cit-HepPh has 28,045 nodes and 3,148,414 edges, which describes paper citation on Arxiv High Energy Physics.
5) **Ecology** (4 networks): such networks include species and the interactions that occur between species within a community. These networks are usually small. The largest ecology network, eco-florida with 128 nodes and 2,075 edges, represents south Florida ecosystems.
6) **Economic** (5 networks): such networks are a diverse set of interconnected economic agents. The largest one in our date set is econ-poli-large with 15,575 and 33,043 edges.
7) **Facebook** (5 networks): such networks contain users of Facebook. The largest one we use is socfb-BU10 with 19,666 nodes and 637,509 edges and it consists of people (nodes) with edges representing friendship ties.
8) **Infrastructure** (4 networks): such networks show interlinks between fundamental facilities and systems. We choose one power network, two air networks and one road network. The road network inf-roadNet-PA is the largest with 1,087,562 nodes and 1,541,514 edges.
9) **Power** (5 networks): such networks contain components deployed to supply, transfer, and use electric power. The largest one is power-bcspwr10 with 5,300 nodes and 13,571 edges.
10) **Road** (8 networks): nodes of such networks represent intersections and edges represent roads connecting the intersections. The largest road network (also the largest among all 100 networks) is road-netherlands-osm with 2,216,688 nodes and 2,441,238 edges.

TABLE 2: Overview on the data sets covering 15 categories in this study.

Category	Range of Nodes	Range of Edges	Range of Density	Range of Maximum degree
social networks	[34, 1134890]	[78, 2987624]	[5e-06, 0.139037]	[17, 28754]
biological networks	[453, 15063]	[1948, 245862]	[0.000736, 0.019780]	[37, 523]
brain networks	[29, 1770]	[44, 16089]	[0.003157, 0.712596]	[9, 927]
citation networks	[12495, 28045]	[49563, 3148414]	[0.000635, 0.009471]	[709, 8718]
ecology networks	[54, 128]	[350, 2075]	[0.244584, 0.375107]	[48, 110]
economic networks	[257, 15575]	[2375, 86768]	[0.000144, 0.357055]	[106, 1544]
facebook networks	[1446, 19666]	[32984, 637509]	[0.003297, 0.057037]	[300, 1819]
infrastructure networks	[332, 1087562]	[2126, 1541514]	[3e-06, 0.038693]	[9, 242]
power networks	[494, 5300]	[586, 8271]	[0.000589, 0.004812]	[9, 17]
road networks	[39, 2216688]	[170, 2760388]	[1e-06, 0.229420]	[5, 33]
techological networks	[2113, 190914]	[6632, 607610]	[3e-05, 0.002972]	[109, 2628]
web graphs	[643, 1864433]	[2280, 4507315]	[3e-06, 0.011046]	[59, 10991]
email networks	[143, 33696]	[623, 180811]	[0.000103, 0.061361]	[42, 1383]
retweet networks	[2280, 1112702]	[2464, 2278852]	[4e-06, 0.000948]	[267, 31556]
cheminformatics networks	[20, 125]	[49, 141]	[0.018, 0.258]	[5, 7]

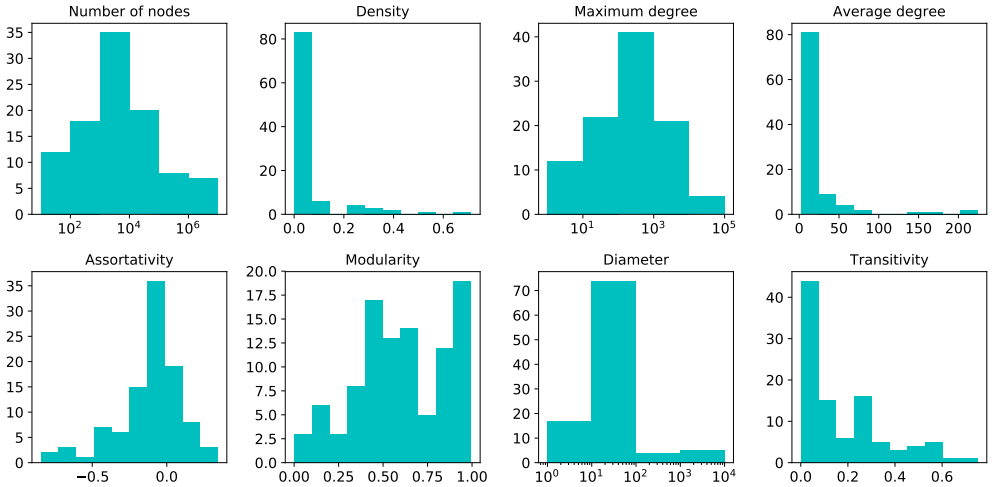

FIGURE 2: Statistics of selected topological metric values of 100 selected networks. The x-axis shows the range of values for each metric and the y-axis represents frequency of networks having a metric value in such range.

11) **Techological** (5 networks): such networks are include interlinks between technology systems. The largest one is tech-RL-caida with 190,914 nodes and 607,610 edges, which describes router-level network topology via Rocketfuel.
12) **Web** (10 networks): such networks describe the links between pages of the World Wide Web. The network web-wikipedia2009 is the largest and contains 1,864,433 nodes and 4,507,315 edges.
13) **Email** (5 networks): Nodes of such networks are email addresses and edges represent mail contacts between two addresses. The largest email network in our data sets is email-enron-large. It has 33,696 nodes and 180,811 edges.
14) **Retweet** (10 networks): such networks show retweeting relationships on the Twitter. The largest retweet network in our data sets is rt-retweet-crawl. It has 1,112,702 nodes and 2,278,852 edges.
15) **Cheminformatics** (4 networks): such networks represent chemical interactions of materials. These networks are small in general and the largest one has 123 nodes and 278 edges.

It should be noted that some networks are annotated with class labels from multiple categories in the original data set. During our selection we ensured to select 100 distinct networks, despite partially overlapping categories. An overview on our data sets is shown in Table 2. Considering that the giant connected component (GCC) is the most important part of maintaining functionality of the network, we extracted the GCC of each network in our data set. All the experimental results we obtained are based on the GCC. To show the diversity of our networks, we summarize statistical properties of all the 100 networks in Figure 2. The diameter values were obtained by a $VD(G)$ approximation algorithm [22], since the computation of exact diameter values is extremely expensive.

We report results as performed on a workstation with 40 cores and 450 GB RAM, running Fedora 28. It should be noted that parallelization can reduce the runtime as the majority of these sampling-based algorithms can be nicely parallelized. Figure 3 reports the speed-ups for a variable number of threads. We find that speed-ups scale-up sub-linearly for a small number of threads (10-20), but quickly reach a limit, as induced by the available number of processing units (40 in our case) and overhead for

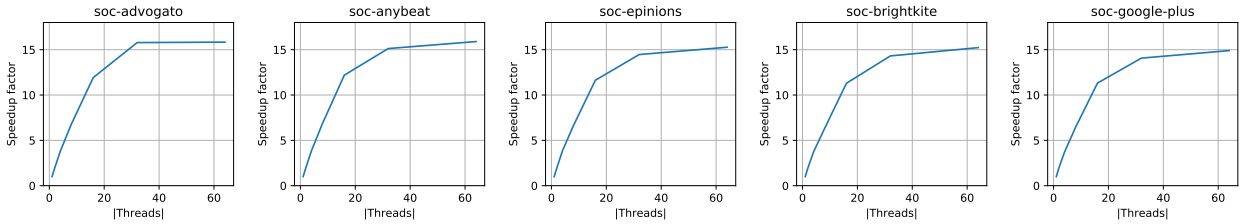

FIGURE 3: Speed-ups of using multi-threads compared to using single thread on five networks, with the number of nodes from 5,054 to 201,949.

merging the parallel sub-results. Overall, parallelization offers constant speed-up at best, i.e. if an algorithm requires T runtime, execution with U processing units cannot reduce the runtime below $\frac{T}{U}$. Since the underlying computational complexity of betweenness computation is cubic in the number of nodes (worst-case), constant-time improvements by parallelization have rather limited use for speeding up computations over very large networks. Therefore, in order to compare all methods in a fair manner, we executed each method independently with a single thread. We computed the exact betweenness values of 100 networks using Brandes' algorithm. It took around six weeks for computing the exact betweenness on the largest network on commodity hardware.

III. EXPERIMENTAL EVALUATION

Section III-A reports the results of sensitivity analysis regarding parameter choices for DIAM, EGO, KPATH, RAND1, and RAND2, respectively. Later sections will evaluate combinations of selected interesting parameters for each method only. Section III-B compares the approximation quality for chosen competitors. Section III-C takes into account the runtime of methods, when assessing the quality of an estimation; with the ultimate goal to have a method which runs reasonably fast and provides close-to-exact results. Section III-D shows the speed-ups of approximation methods compared to exact Betweenness. Section III-E breaks down experimental results to the level of network types. Section III-F reports additional experiments on a real-world problem that requires the computation of betweennesss scores; showing the effect of approximations on the quality of a network dismantling.

A. SENSITIVITY ANALYSIS

DIAM: We evaluated DIAM with 11 initial parameter settings for ϵ and δ. The results are visualized in Figure 4. The performance of $\epsilon = 0.02$ and $\delta = 0.1$ can be considered best. However, it should be noted that DIAM_0.02_0.1 takes already around five hours on the largest data set(road-netherlands-osm with 2,216,688 nodes). While, δ has negligible effects on the results, ϵ greatly affects the results of DIAM. With ϵ decreases, the runtime will increase exponentially as shown in Figure 5. It takes 10

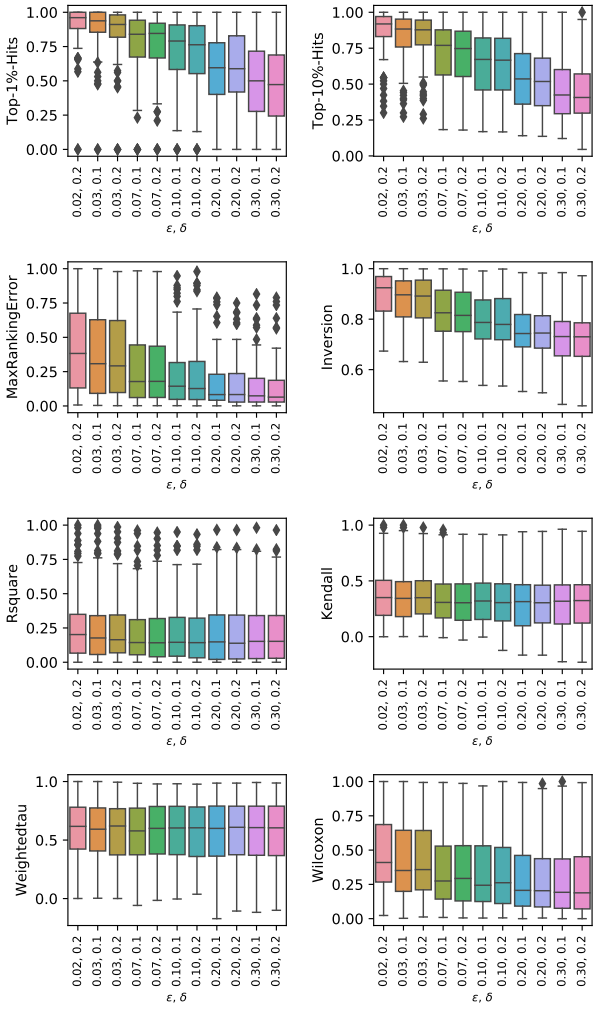

FIGURE 4: Effect of ϵ and δ of DIAM on the values of eight measures.

minutes for $\epsilon = 0.1$ and 20 minutes for $\epsilon = 0.07$ on the largest network and the median values of Top-1%-Hits and Top-10%-Hits on 100 networks can reach 80%. Thus, it can make a good trade-off when we set ϵ to be 0.1 or 0.07.

EGO: We present the results of our sensitivity analysis on EGO in Figure 6. As α_1 increases, the changes of values of different measures are dissimi-

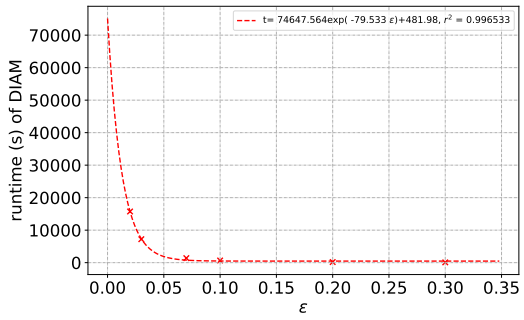

FIGURE 5: Effect of ϵ on the runtime of DIAM.

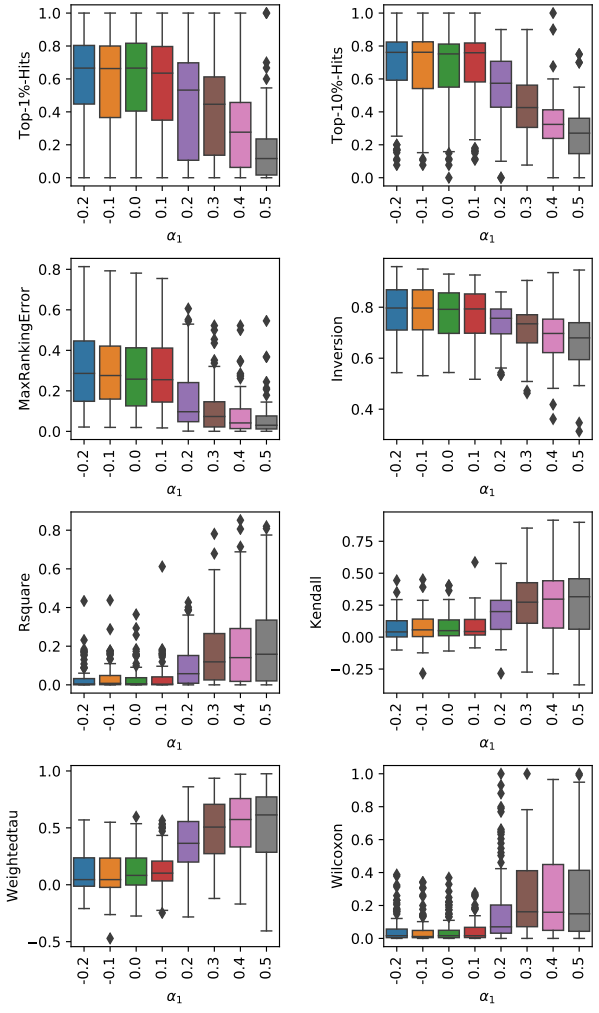

FIGURE 6: Effect of α_1 for EGO on the values of eight measures.

lar. The values of Top-1%-Hits, MaxRankingError and Inversion increase as α_1 decreases but the values of Kendall, Rsquare, Weightedtau and Wilcoxon decrease as α_1 decreases. Moreover, Figure 7 shows that runtime will rapidly increase when α_1 decreases as $t = 186.431e^{-29.344\alpha_1} + 43.135$ on the largest data set. Thus, setting α_1 = -0.2 will consume a lot of runtime. Therefore, α_1 = 0.0 can get a good trade-off on Top-1%-Hits, MaxRankingError and Inversion, as the values are similar to the values of α_1 = -0.2.

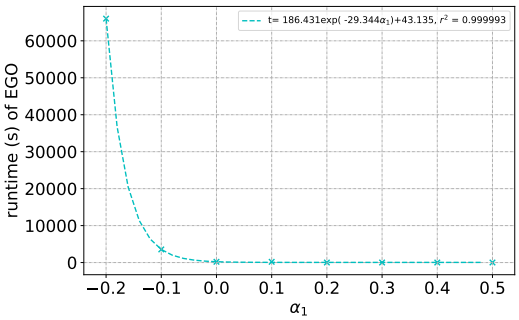

FIGURE 7: Effect of α_1 on the runtime of EGO.

KPATH: Figure 8 shows the results of KPATH. The quality is unstable when we set $\alpha_2 > 0.2$ and it increases when we add the maximum length of the shortest paths (the value of k) or reduce the value of α_2. Based on the results in Figure 7, decreasing α_2 will exponentially increase runtime. KPATH_0.0_8 takes around two hours on the largest data set. Figure 9 shows that, given a fixed α_2 value, runtime and maximum length of shortest paths have a quadratic power relationship. With combining two charts, we can see that KPATH_0.2_8 offers a nice trade-off.

RAND1: We performed sensitivity analysis of the approximation quality against the number of sample pivots for RAND1. We evaluated nine parameter settings: 1, 2, 4, 8, 16, 32, 64, 128 and 256. For each measure, we plot a box chart which shows the distribution of the values of such measure with 9 different number of pivots in Figure 10. As Figure 10 shows, the values of Top-%-Hits, MaxRankingError, Wilcoxon and Inversion increase clearly as we sample more number of pivots. However, for other measures, the effect of the number of pivots on quality is not clearly. Since RAND1_1 only samples 1 pivot, it is the fastest. RAND1_256 can offer the best quality but it takes more runtime. Figure 11 presents the runtime of RAND1 on the largest data set. The runtime increases linearly with increasing number of sample pivots, which is consistent with the $O(|P|E)$ complexity of the RAND1, where $|P|$ is the number of sample pivots and E is the number of edges. Besides, we can see that starting from 64 pivots, the values of measures are close to the values obtained by sampling 256 pivots.

RAND2: We analyzed the sensitivity of the eight measures for a given number of sample pivots for RAND2. As the paper of RAND2 suggests, we set the number of pivots to be 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096. Figure 12 presents the results. RAND2_4096 is outstanding regarding all measures. However, similar to RAND1, increasing the number of sample pivots means linearly increasing the amount of computation with the $O(|P|E)$ complexity. Figure 13 shows how the number

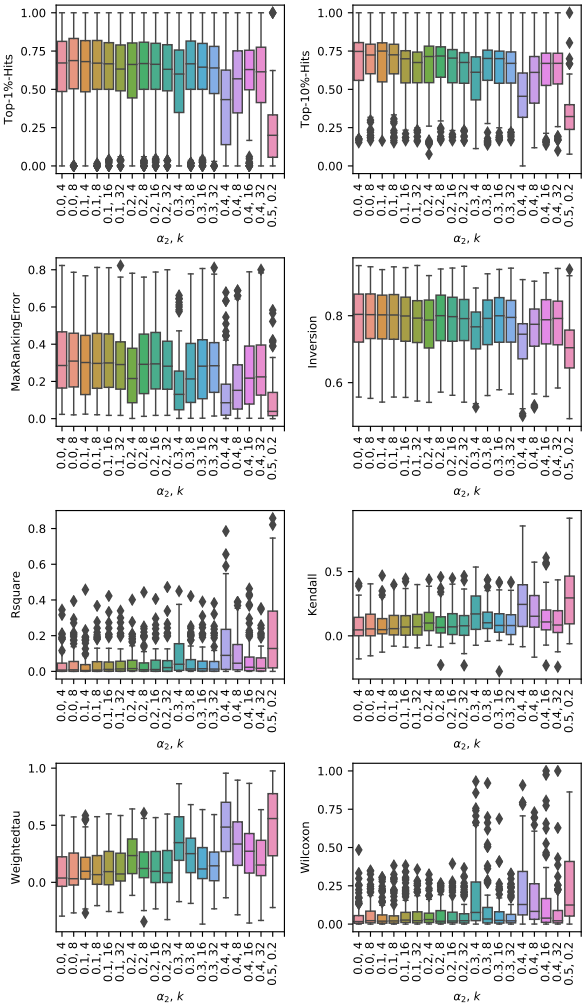

FIGURE 8: Effect of α_2 and k of KPATH on the values of eight measures.

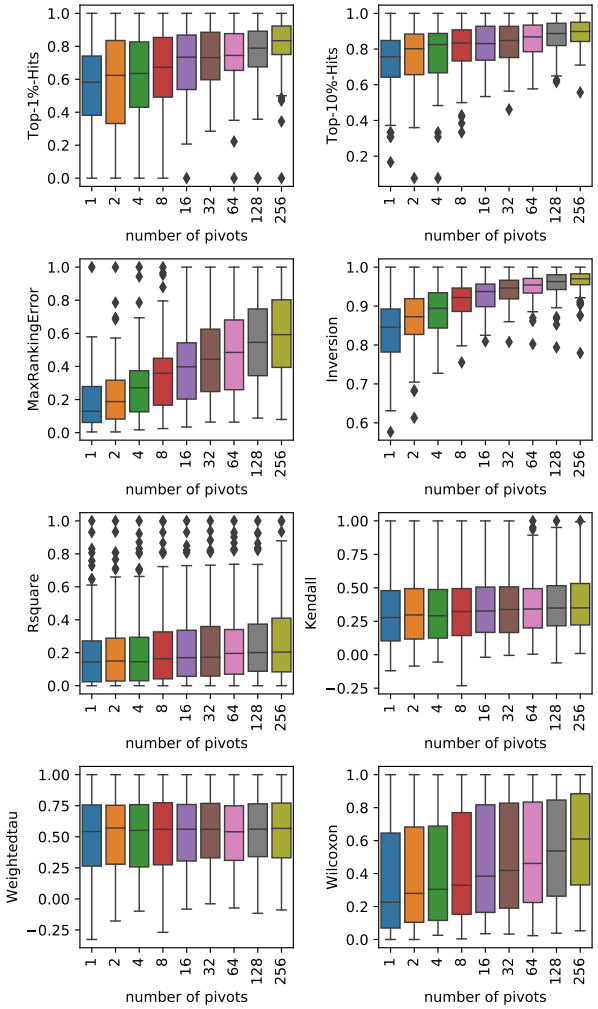

FIGURE 10: Effect of the number of sample pivots for RAND1 on the values of eight measures.

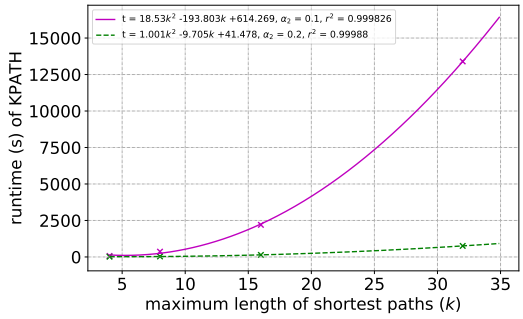

FIGURE 9: Effect of α_2 and k on the runtime of KPATH.

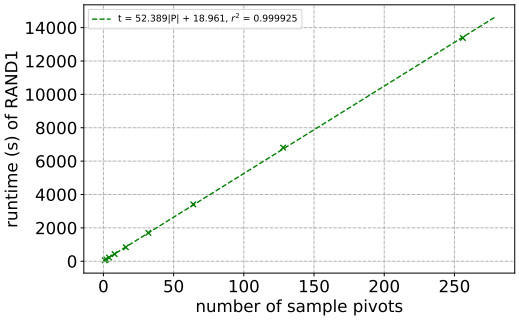

FIGURE 11: Effect of the number of sample pivots on the runtime of RAND1.

of pivots affects the runtime of RAND2 on the largest data set. As Figure 12 shows, the median value of Top-1%-Hits reaches 80% and the median value of Top-10%-Hits reaches 90% when we sample 64 pivots. The median value of Top-1%-Hits reaches 90% when we run RAND2_512. Moreover, RAND_64 only takes few minutes and RAND2_512 can finish in around 10 minutes on the largest network. Thus, these two methods can offer good trade-offs between computational efficiency and approximation quality.

In synthesis, we evaluated the effects of different parameter settings of five methods on both runtime and the values of eight measures. For further analysis, we

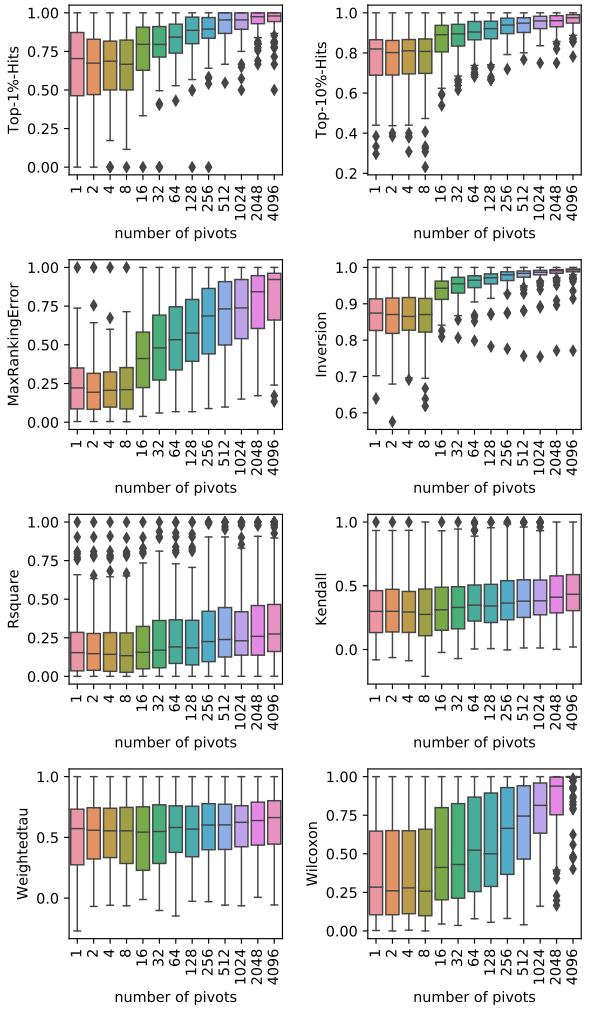

FIGURE 12: Effect of the number of sample pivots of RAND2 on the values of eight measures.

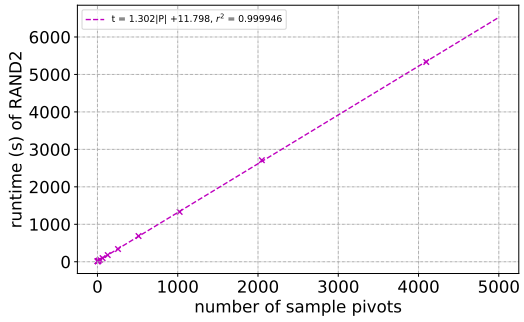

FIGURE 13: Effect of the number of sample pivots on the runtime of RAND2.

selected three reasonable parameter settings for each type of method: a fast one, an accurate one and one that makes a good trade-off between runtime and quality. In total we have 15 competitors: Five methods with three parameter settings each. Table 3 shows an overview on these selected competitors.

TABLE 3: Overview of selected parameter settings of each method.

Method	Parameter	Competitor
RAND1	$\|P\| = 1$	RAND1_1
	$\|P\| = 64$	RAND1_64
	$\|P\| = 256$	RAND1_256
RAND2	$\|P\| = 64$	RAND2_64
	$\|P\| = 512$	RAND2_512
	$\|P\| = 4096$	RAND2_4096
DIAM	$\epsilon = 0.02, \delta = 0.2$	DIAM_0.02_0.2
	$\epsilon = 0.07, \delta = 0.1$	DIAM_0.07_0.1
	$\epsilon = 0.10, \delta = 0.1$	DIAM_0.10_0.1
KPATH	$\alpha_2 = 0.0, k = 8$	KPATH_0.0_8
	$\alpha_2 = 0.2, k = 8$	KPATH_0.2_8
	$\alpha_2 = 0.4, k = 4$	KPATH_0.4_4
EGO	$\alpha_1 = -0.2$	EGO_-0.2
	$\alpha_1 = 0.0$	EGO_0.0
	$\alpha_1 = 0.5$	EGO_0.5

B. APPROXIMATION QUALITY

We normalized the values of eight measures among all 15 competitors on each network. The average values of each competitor are presented in Figure 14. From this heatmap, we can see that RAND2_4096 is outstanding on all measures. Overall, RAND1 and RAND2 methods perform well. Among eight measures, EGO_0.5 and KPATH_0.4_4 only work well on Kendall and Weightedtau. EGO_-0.2, EGO_0.0, KPATH_0.0_8 and KPATH_0.2_8 can get promising results of identifying Top-k% nodes. Regarding Inversion and MaxRankingError, RAND2_4096 can obtain superior accuracy.

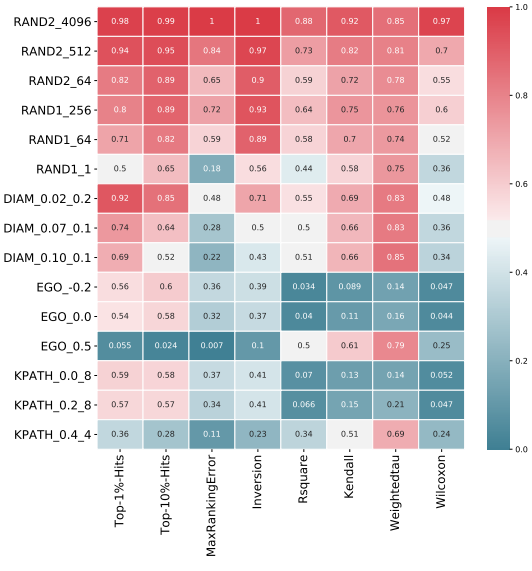

FIGURE 14: Quality of 15 competitors on eight measures.

C. TRADEOFF BETWEEN SCALABILITY AND APPROXIMATION QUALITY

We normalized the runtime of above 15 competitors (fastest = 1, slowest = 0) on each network and computed

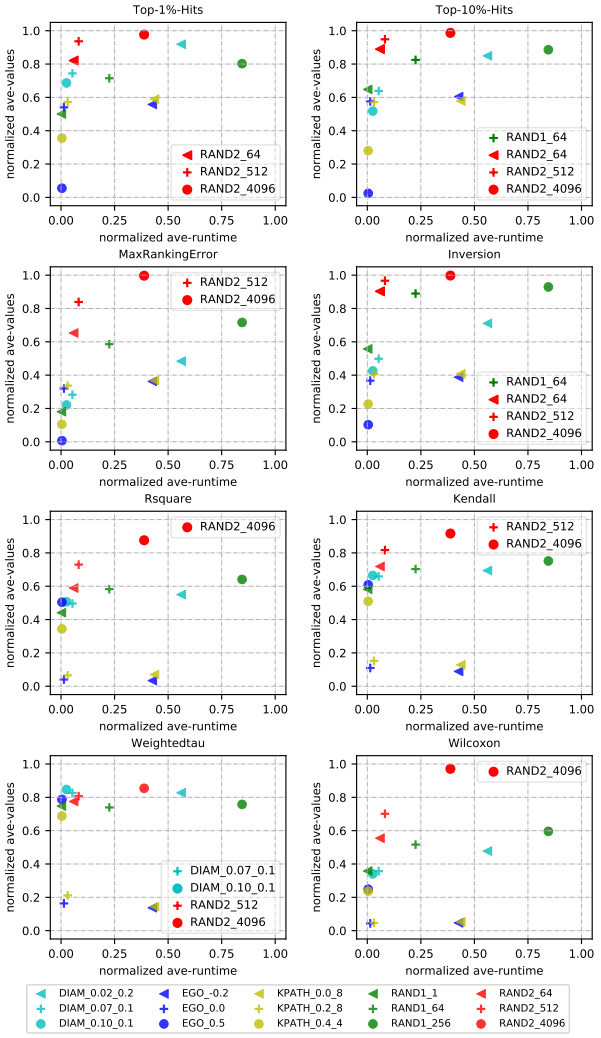

FIGURE 15: Trade-offs between scalability and approximation quality.

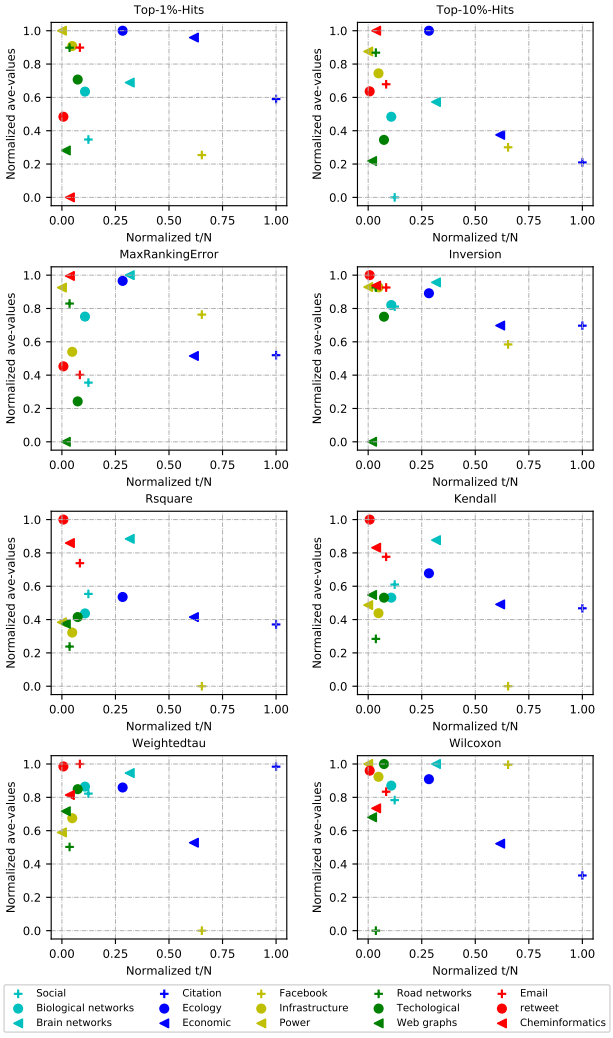

FIGURE 16: Trade-offs of RAND2_4096 on different network categories.

average normalized runtime of each competitor. For each measure, we also normalized the values of measures for 15 competitors. In Figure 15, we report the results of analyzing trade-offs between scalability and approximation quality, considering the measure separately. The x-axis represents average normalized runtime and y-axis represents average normalized values of measures. Figure 15 marks the competitor with normalized ave-runtime smaller than 0.5 and the normalized ave-value of the measure larger than 0.8. Thus, we suggest that these competitors can obtain a nice trade-off. As Figure 15 shows, RAND2_4096 performs outstanding and always makes a superior trade-off on all eight measures. RAND2_512 offers a good trade-off on seven measures and it is very fast. RAND2_64 can also make a trade-off on three measures and RAND1_64 is good on three measures. However, two DIAM methods: DIAM_0.07_0.1 and DIAM_0.10_0.1 only make good trade-offs on Weightedtau.

D. SPEED-UPS

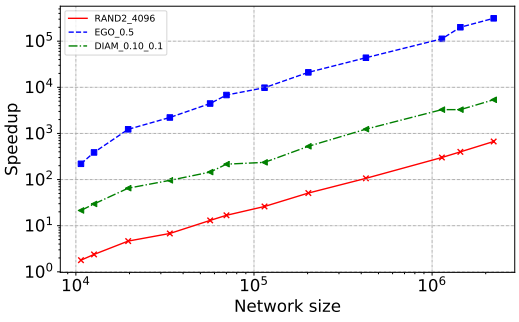

FIGURE 17: Speedups of three competitors regarding different network sizes.

Compared to existing studies, our work is the only one to run all experiments, including the computation of exact betweenness, independently with a single thread

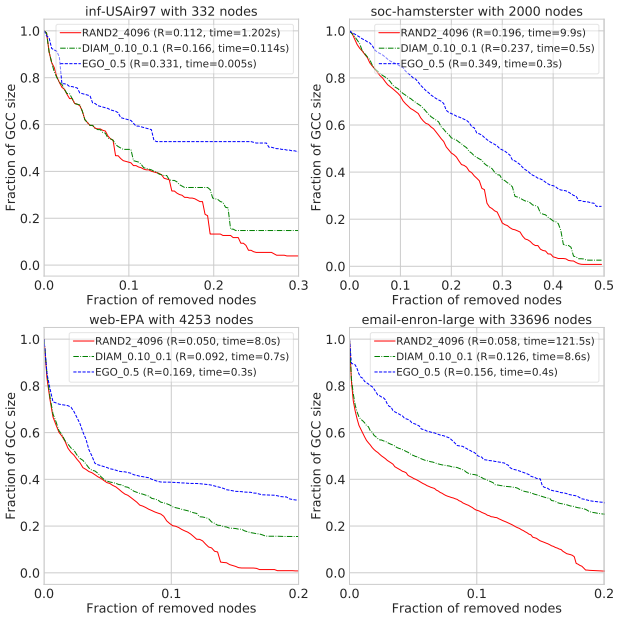

FIGURE 18: Dismantling networks based on estimated ranking.

on the same computer. Based on equivalent hardware conditions, we can further evaluate the speed-ups of approximation methods compared to Betw. As the above results suggest, we chose three competitors to show the results of speed-ups, including a faster one EGO_0.5, a high quality one RAND2_4096 and a trade-off one DIAM_0.10_0.1. Since small networks can not show the speed-ups clearly and RAND2_4096 samples 4096 pivots for networks with N < 4096, we selected 12 networks with N from 10,680 to 2,216,688. Figure 17 presents the speed-ups regarding different network sizes. The speed-ups of three competitors tend to increase as the network size grows. Because EGO_0.5 sacrifices the accuracy, the speed-ups of EGO_0.5 are always the highest. In contrast, RAND2_4096 takes the most runtime for the best accuracy and results in the smallest speed-ups among three competitors.

E. RESULTS OF NETWORK TYPES

Above we have comprehensively discussed the results on 100 networks. Furthermore, We analyzed the results on 15 different network categories. Based on the above analysis, RAND2_4096 can always offer a good trade-off on all networks. Therefore, we evaluated the performance of RAND_4096 on different network categories. As different types of networks have variabel sizes (e.g., cheminformatics networks are very small but retweet networks are very large), we considered the runtime per node on each network and normalized it among all categories. We also normalized the average values of measures for each category. Figure 16 presents the trade-offs between normalized average runtime per node and normalized average values of measures. We can see that performance of RAND2_4096 on Brain, Ecology, Cheminformatics, Retweet and Power networks is acceptable. On Top-1%-Hits, RAND2_4096 can quickly identify key hubs in Ecology, Infrastructure, Power, Road and Email networks.

F. USE CASE: NETWORK ROBUSTNESS

As a final experiment in this study, we assess the efficiency and effectiveness of betweenness approximation methods for one typical use case: Network dismantling. The goal of network dismantling is to break a given network into components, by simulating node failures/attacks. The choice of node order has a significant effect on the robustness of the network [13]. Accordingly, one is usually interested in the best option. Recent research [14] suggests that removing nodes in decreasing order of their betweenness centrality gives an excellent reference for network dismantling. Based on the estimated ranking of nodes on betweenness values, we considered the network dismantling problem and evaluated robustness under attacking by different orders which obtained from three selected approximate competitors: RAND2_4096, DIAM_0.10_0.1 and EGO_0.5. The robustness of a given network with N nodes is defined as $R = \frac{1}{N}\sum_{Q=1}^{N} S(Q)$, where s(Q) is the size of GCC after removing Q nodes [39]. The best attack strategy will offer the minimum R value. However, it takes large amount of computation to obtain the R value. Thanks to [40], we can get a good approximation of the R value quickly by sampling.

Figure 18 presents the results of dismantling on four networks with different number of nodes from 332 to 33,696. We can see that RAND2_4096 can obtain the minimum R value among these three competitors. Besides, though EGO_0.5 is the fastest, it offers the highest R value, which also means the worst attack effect. DIAM_0.10_0.1 makes a trade-off between runtime and attacking effect. The dismantling results are consistent with the previous results. As the nodes with high betweenness can be better identified by RAND2_4096, these high-betweenness nodes can be better attacked by RAND2_4096 in an earlier stage, which leads to a quick destruction of connectivity of the network and, in turn, yields in a fast GCC drop during the node removal process.

IV. CONCLUSIONS

In this paper, we systematically investigate the scalability and accuracy of betweenness approximation algorithms. To ensure coverage of various network structures, we elaborately selected 100 real-world networks of diverse scales and different topological metrics in 15 fields. This is, to the best of our knowledge, the largest study performed in this field. We devise eight measures to evaluate the accuracy in comparison to exact betweenness computation. For each method, we provided three suitable parameter settings based on sensitivity analysis, which

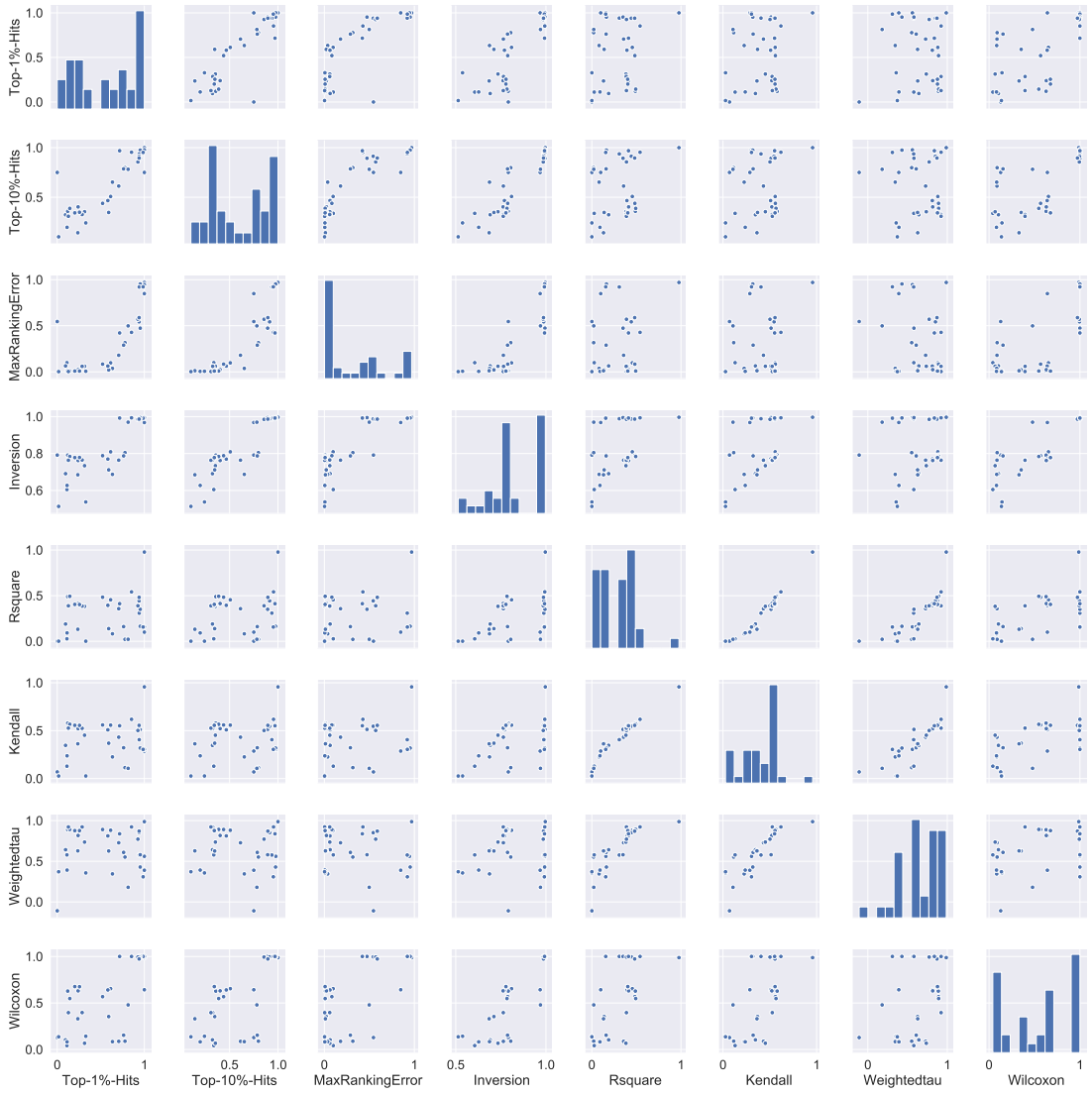

FIGURE 19: Analysis of the correlation between measures for social network on three selected competitors.

can be regarded as a reference for other researchers. Figure 19 visualizes the degree of correlation between pairwise measures, confirming that each measure indeed gives a unique view on the type of desired approximation. From the results of our comprehensive experiments, we found that RAND2 with sampling 4096 pivots (RAND2_4096), a classic method based on uniformly random sampling and linear scaling-up, still provides the most accurate approximation and makes a nice trade-off between scalability and approximation quality. Regarding different network types, the Top-1% nodes can be quickly identified by RAND2_4096 on power networks, road networks and infrastructure networks, which has great significance for the maintenance of critical infrastructure systems. Through our robustness experiments, we showed that the method with the best accuracy can effectively dismantle the network.

From our experimental results on 100 real-world networks with different sizes and structures, together with additional random networks, we derive a recommendation for choosing methods as follows:

a) For sparse, small networks (i.e. density less than 10% and less than 10,000 nodes), all these five methods, including RAND1 [18], RAND2 [21], DIAM [22], KPATH [31] and EGO [30], can all quickly obtain high approximation accuracy. Specifically, RAND2 with 512 pivots can get the best result.

b) If the network is small and dense (e.g., the density is larger than 10% and the number of nodes is less than 10,000), the quality of KPATH and EGO is often better than DIAM. Since DIAM determines the number of sample pairs based on estimated vertex-diameter and the vertex-diameter of a dense network is rather small, DIAM indeed loses accuracy; while KPATH_0.2_8 and EGO_0.1 can be selected. Moreover, for such small and dense networks, RAND2 can still get higher accuracy

compared to other methods when we sample thousands of pivots, if longer runtime is acceptable.
c) For large networks (i.e. the number of nodes is larger than 10,000), KPATH and EGO often perform bad. Accordingly, one should choose DIAM or RAND2. Considering trade-offs between runtime and accuracy, we suggest DIAM_0.07_0.1 and RAND2 with thousands of pivots (i.e. choosing RAND2_4096).
d) For either size of network, if there is a requirement on the approximation error ϵ with a given probability, one needs to choose DIAM as a betweenness approximation method.
In synthesis, we have conducted an experimental review on state-of-the-art methods for estimating betweenness. Our results shed light on approximate betweenness method selection with consideration of accuracy requirements and computational resources. Given the significance of betweenness as a measure of node importance in various fields, we believe that our study will contribute to better analysis and understanding of complex network phenomena.

REFERENCES

[1] R. Albert, "Scale-free networks in cell biology," J. cell science, vol. 118, no. 21, pp. 4947–4957, 2005.
[2] Y. Assenov, F. Ramírez, S.-E. Schelhorn, T. Lengauer, and M. Albrecht, "Computing topological parameters of biological networks," Bioinforma., vol. 24, no. 2, pp. 282–284, 2007.
[3] X. Sun, V. Gollnick, and S. Wandelt, "Robustness analysis metrics for worldwide airport network: A comprehensive study," Chin. J. Aeronaut., vol. 30, no. 2, pp. 500–512, 2017.
[4] S. Wandelt, Z. Wang, and X. Sun, "Worldwide railway skeleton network: Extraction methodology and preliminary analysis," IEEE Transactions on Intell. Transp. Syst., vol. 18, no. 8, pp. 2206–2216, 2017.
[5] G. A. Pagani and M. Aiello, "The power grid as a complex network: a survey," Phys. A: Stat. Mech. its Appl., vol. 392, no. 11, pp. 2688–2700, 2013.
[6] S. V. B. Heidelberg, "Statistical mechanics of complex networks," Rev. Mod. Phys., vol. 74, no. 1, p. xii, 2001.
[7] G. Sabidussi, "The centrality index of a graph," Psychom., vol. 31, no. 4, pp. 581–603, 1966.
[8] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," J. Math. Sociol., vol. 2, no. 1, pp. 113–120, 1972.
[9] L. Katz, "A new status index derived from sociometric analysis," Psychom., vol. 18, no. 1, pp. 39–43, 1953.
[10] L. C. Freeman, "A set of measures of centrality based on betweenness.," Sociom., vol. 40, no. 1, pp. 35–41, 1977.
[11] M. E. J. Newman, "A measure of betweenness centrality based on random walks," Soc. Networks, vol. 27, no. 1, pp. 39–54, 2003.
[12] F. Hu and Y. Liu, "Multi-index algorithm of identifying important nodes in complex networks based on linear discriminant analysis," Mod. Phys. Lett. B, vol. 29, no. 03, pp. 1777–1286, 2015.
[13] X. Sun, S. Wandelt, and X. Cao, "On node criticality in air transportation networks," Networks Spatial Econ., vol. 17, no. 11, pp. 1–25, 2017.
[14] S. Wandelt, X. Sun, D. Feng, M. Zanin, and S. Havlin, "A comparative analysis of approaches to network-dismantling," SCIENTIFIC REPORTS, vol. 8, no. 1, p. 13513, 2018.
[15] M. Benzi and C. Klymko, "On the limiting behavior of parameter-dependent network centrality measures," Siam J. on Matrix Analysis & Appl., vol. 36, no. 2, pp. 686–706, 2013.
[16] U. Brandes, "A faster algorithm for betweenness centrality," The J. Math. Sociol., vol. 25, 03 2001.
[17] L. C. Freeman, "Centrality in social networks conceptual clarification," Soc. Networks, vol. 1, no. 3, pp. 215–239, 1978.
[18] U. Brandes and C. Pich, "Centrality estimation in large networks," Int. J. Bifurc. Chaos, vol. 17, no. 07, p. 16, 2007.
[19] D. A. Bader, S. Kintali, K. Madduri, and M. Mihail, "Approximating betweenness centrality," in Algorithms and Models for the Web-Graph (A. Bonato and F. R. K. Chung, eds.), (Berlin, Heidelberg), pp. 124–137, Springer Berlin Heidelberg, 2007.
[20] R. J. Lipton and J. F. Naughton, "Estimating the size of generalized transitive closures," in Proceedings of the 15th International Conference on Very Large Data Bases, VLDB '89, (San Francisco, CA, USA), pp. 165–171, Morgan Kaufmann Publishers Inc., 1989.
[21] R. Geisberger, P. Sanders, and D. Schultes, "Better approximation of betweenness centrality," in Proceedings of the Meeting on Algorithm Engineering & Expermiments, (Philadelphia, PA, USA), pp. 90–100, Society for Industrial and Applied Mathematics, 2008.
[22] M. Riondato and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling," Data Min. Knowl. Discov., vol. 30, no. 2, pp. 438–475, 2016.
[23] V. N. Vapnik and A. Y. Chervonenkis, On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities, pp. 11–30. Cham: Springer International Publishing, 2015.
[24] E. Bergamini, H. Meyerhenke, and C. Staudt, "Approximating betweenness centrality in large evolving networks," Proc. Work. on Algorithm Eng. Exp., vol. 2015, 09 2014.
[25] E. Bergamini and H. Meyerhenke, "Fully-dynamic approximation of betweenness centrality," in Algorithms - ESA 2015 (N. Bansal and I. Finocchi, eds.), (Berlin, Heidelberg), pp. 155–166, Springer Berlin Heidelberg, 2015.
[26] M. Riondato and E. Upfal, "Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages," ACM Trans. Knowl. Discov. Data, vol. 12, pp. 61:1–61:38, July 2018.
[27] S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms. New York, NY, USA: Cambridge University Press, 2014.
[28] D. Pollard, "Convergence of stochastic process," Econ., vol. 52, 11 1985.
[29] M. Borassi and E. Natale, "KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation," in 24th Annual European Symposium on Algorithms (ESA 2016) (P. Sankowski and C. Zaroliagis, eds.), vol. 57 of Leibniz International Proceedings in Informatics (LIPIcs), (Dagstuhl, Germany), pp. 20:1–20:18, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.
[30] M. Everett and S. P. Borgatti, "Ego network betweenness," Soc. Networks, vol. 27, no. 1, pp. 31–38, 2005.
[31] J. Pfeffer and K. M. Carley, "k-centralities: Local approximations of global measures based on shortest paths," in Proceedings of the 21st International Conference on World Wide Web, WWW '12 Companion, (New York, NY, USA), pp. 1043–1050, ACM, 2012.
[32] Y. sup Lim, D. S. Menasche, B. Ribeiro, D. Towsley, and P. Basu, "Online estimating the k central nodes of a network," in Proceedings of the 2011 IEEE Network Science Workshop, NSW '11, (Washington, DC, USA), pp. 118–122, IEEE Computer Society, 2011.
[33] A. S. Maiya and T. Y. Berger-Wolf, "Online sampling of high centrality individuals in social networks," in Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I, PAKDD'10, (Berlin, Heidelberg), pp. 91–98, Springer-Verlag, 2010.
[34] M. Haghir Chehreghani, "An efficient algorithm for approximate betweenness centrality computation," in Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management, CIKM '13, (New York, NY, USA), pp. 1489–1492, ACM, 2013.
[35] C. L. Staudt, A. Sazonovs, and H. Meyerhenke, "Networkit: An interactive tool suite for high-performance network analysis," Eprint Arxiv, vol. 4, no. 4, pp. 508–530, 2014.
[36] Z. AlGhamdi, F. Jamour, S. Skiadopoulos, and P. Kalnis, "A benchmark for betweenness centrality approximation algorithms on large graphs," in Proceedings of the 29th International Conference on Scientific and Statistical Database Management, SSDBM '17, (New York, NY, USA), pp. 6:1–6:12, ACM, 2017.
[37] P. H. Leslie, "The treatment of ties in ranking problems," Biom., vol. 33, no. 3, pp. 239–251, 1945.
[38] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, pp. 4292–4293, AAAI Press, 2015.

[39] C. M. Schneider, A. A. Moreira, J. S. Andrade, S. Havlin, and H. J. Herrmann, "Mitigation of malicious attacks on networks," Proc. Natl. Acad. Sci., vol. 108, no. 10, pp. 3838–3841, 2011.
[40] S. Wandelt, X. Sun, M. Zanin, and S. Havlin, "Qre: Quick robustness estimation for large complex networks," Futur. Gener. Comput. Syst., vol. 83, 02 2017.