International Journal of Modern Physics C
Vol. 27, No. 5 (2016) 1650052 (23 pages)

DOI: 10.1142/S0129183116500522

World Scientific
www.worldscientific.com

Social network sampling using spanning trees

Zeinab S. Jalali, Alireza Rezvanian* and Mohammad Reza Meybodi

Soft Computing Laboratory
Computer Engineering and Information Technology Department
Amirkabir University of Technology (Tehran Polytechnic)
Hafez Ave., 424, Tehran, Iran
**a.rezvanian@aut.ac.ir*

Received 27 January 2015
Accepted 5 October 2015
Published 23 December 2015

Due to the large scales and limitations in accessing most online social networks, it is hard or infeasible to directly access them in a reasonable amount of time for studying and analysis. Hence, network sampling has emerged as a suitable technique to study and analyze real networks. The main goal of sampling online social networks is constructing a small scale sampled network which preserves the most important properties of the original network. In this paper, we propose two sampling algorithms for sampling online social networks using spanning trees. The first proposed sampling algorithm finds several spanning trees from randomly chosen starting nodes; then the edges in these spanning trees are ranked according to the number of times that each edge has appeared in the set of found spanning trees in the given network. The sampled network is then constructed as a sub-graph of the original network which contains a fraction of nodes that are incident on highly ranked edges. In order to avoid traversing the entire network, the second sampling algorithm is proposed using partial spanning trees. The second sampling algorithm is similar to the first algorithm except that it uses partial spanning trees. Several experiments are conducted to examine the performance of the proposed sampling algorithms on well-known real networks. The obtained results in comparison with other popular sampling methods demonstrate the efficiency of the proposed sampling algorithms in terms of Kolmogorov–Smirnov distance (KSD), skew divergence distance (SDD) and normalized distance (ND).

Keywords: Online social networks; network sampling; spanning trees.

PACS Nos.: 02.10.Ox, 02.60.Cb.

1. Introduction

In recent years, various researchers have focused on the analysis of dynamic behaviors of individuals in social networks, such as cascading failures and synchronization.[1–6] However, there are several issues, such as the large scale, complexity, dynamicity and limitations on accessing these networks which make them hard or even impossible to study[7]. So network sampling methods have emerged to generate

*Corresponding author.

representative networks with smaller sizes that are reasonable to study and perform several kinds of analyses in an offline manner with reasonable time and lower cost.[8–11] Accuracy of most studies and analyses on real networks critically depends on the accuracy of the sampled networks obtained from the original networks. The main purpose of network sampling algorithms is to construct an appropriate sampled network that directly resembles the original network and preserves most properties of the original network. In order to reach a desirable accuracy for sampled network, sometimes more processing is required for the sampling algorithm. Although recently several simple network sampling algorithms have been presented, most of these algorithms independently sample just the vertices or edges with low computational cost, which results in sampled networks that are poor in recovering the important natural characteristics of original networks. A proper sampling algorithm should be able to preserve different natural characteristics of social networks, such as the presence or absence of important, central and influential individuals.[12]

In this paper, two new algorithms for sampling online social network using spanning trees are proposed. The proposed sampling algorithms first find several spanning trees of the input graph. Then, the edges in these spanning trees are ranked according to the number of times that each edge has appeared in the set of found spanning trees. Subsequently, the sampled network is constructed as a sub-graph of the given network which contains a fraction of highly ranked edges. The proposed algorithms are based on the hypothesis that by considering a set of arbitrary number of spanning trees with different starting points, edges with higher number of presence in these spanning trees can be considered as important edges that reflect several characteristics of the original network. In the second proposed sampling algorithm, the cost of algorithm is reduced by replacing spanning tress with *partial spanning tress* in order to avoid visiting entire network and visiting each node several times. To investigate the performance of the proposed sampling algorithms, several numerical simulations are conducted on the well-known real networks. The experimental results are compared with recently reported sampling methods in terms of Kolmogorov–Simonov distance (KSD), skew divergence distance (SDD) and normalized distance (ND). Obtained results show the superiority of the proposed algorithms over popular sampling algorithms.

The rest of this paper is organized as follows. Section 2 provides preliminaries about network sampling and an overview of sampling algorithms. In Sec. 3, the proposed sampling algorithm using spanning trees is described. The performances of the proposed algorithms are investigated through several experiments on well-known social network graphs in Sec. 4. Section 5 concludes the paper.

2. Preliminaries

2.1. *Network sampling*

A social network can be represented as a graph $G = \langle V, E \rangle$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex-set that represent the users of an online social network (OSN) and

$E = \{e_1, e_2, \ldots, e_m\}$ is the set of edges that represent a kind of relationship between users in an OSN. Let $G = \langle V, E \rangle$ denote the input graph, a sampling technique is a function $f: G \rightarrow G_s$ with sampling rate $0 < \varphi < 1$, where $G_s = \langle V_s, E_s \rangle$ is the sampled network in which $V_s \subseteq V$ and $|V_s| = \varphi \times n$.[13]

2.2. *Related work*

From simple measures to advanced sampling techniques, social network graphs can reveal valuable information for analysis of social networks. However, the huge size of OSNs makes them difficult to study directly, thus requiring graph sampling as a solution to turn them into graphs of more manageable sizes, whilst preserving their important characteristics. In literature, there are several classifications for network sampling techniques. According to Ref. 14, network sampling algorithms are categorized into vertex sampling, edge sampling and topology based sampling. In vertex sampling, vertices of a graph are sampled independently at random. Breadth first search (BFS) and Metropolis–Hasting random walk (MHRW) are examples of vertex sampling-based algorithms. In edge sampling, edges of the given network are sampled independently at random; frontier sampling (FS) is an example of this algorithm. In topology-based sampling, selection of a vertex or an edge depends on the topology of the original graph. Snowball sampling (SBS),[15] forest fire sampling (FFS) and random walk (RW)-based sampling[16–18] are considered topology-based sampling algorithms.

In Ref. 19, the authors classified sampling algorithms as random vertex sampling (RVS) and graph traversal sampling. In RVS, each vertex is visited at most once and there is no replacement during the process of sampling. BFS, FFS,[20] SS and respondent driven sampling (RDS)[21] are examples of RVS. Graph traversal sampling algorithms (also called crawling-based sampling), such as FFS and RWs (RWS),[16] iteratively select the next node according to some criteria among all adjacent nodes of that node for collecting samples from networks.

One of the first studies on network sampling was presented by Lescovec and Faloutros.[20] They introduced several sampling algorithms in order to either find a sampled graph with the most similarity to the initial graph or to find past versions of the initial graph. They also compared a large variety of graph sampling algorithms and showed that FFS is better than other existing sampling algorithms. In Ref. 22, efficiency and feasibility of RDS as a web-based sampling algorithm has been studied by Wejnert and Heckathorn In Ref. 23, the authors implemented and compared several crawling algorithms to obtain representative samples of *Facebook* users and it has been shown that MHRW and reweighted RW (RWRW) perform remarkably well, while the most traditional algorithms such as BFS and RWS lead to substantial biases. In Ref. 19, a procedure for correcting the bias of sampling methods has been proposed by Kurant *et al.* Their study indicated that the degree of graph is overestimated by BFS, while it is underestimated by RWS. Ribeiro *et al.* in Ref. 24 studied the steady state behavior of continuous-time random walks (CTRW) on

Markov dynamic networks and compared two types of CTRWs including walks at a constant rate (CTRW-C) and walks with a rate proportional to the vertex degree (CTRW-D).

In Ref. 25 an analytical comparison between BFS and RS has been presented by Son *et al.* In order to systematically study the effects of sampling biases, Siciliano *et al.*[26] proposed an adaptive threshold algorithm to be used for network research. In Ref. 27, a modified RWS algorithm for estimating characteristics of directed graphs has been proposed by Ribeiro *et al.* that allows random jumps. An algorithm for quickly collecting information from the neighborhood of a user in a dynamic manner by avoiding visiting all vertices in the vicinity of a user is proposed by Papgelis *et al.* in Ref. 8. The advantages of their algorithm were demonstrated only by experimentations. Pina-Garcia and Gu in Ref. 28 altered the behavior of MHRW using spirals as a probability distribution instead of a classic normal distribution. Their results showed that their algorithm outperforms normal MHRW in case of illusion spiral. Their results also verified previous estimations provided in literature by Gjoka *et al*[23]. In Ref. 29, a sampling algorithm, called star sampling that takes all the neighbors as valid samples was proposed by Wang and Lu. This sampling algorithm is an enhancement of RWS by a factor of the average degrees.

In Ref. 30, Rezvanian *et al.* proposed a sampling algorithm which uses distributed learning automata sampling (DLAS).[30] In DLAS, a set of distributed learning automata cooperate with each other in order to take appropriate samples from the given network. The results of their algorithm performed much better than the results of RDS and RWS in terms of KS test. Their algorithm outperforms some other sampling algorithms, especially in sampling rates lower than 0.2, but still needs to correct its biases as it is based on high degree vertices. In Ref. 31, Rezvanian and Meybodi proposed a sampling method using the concept of shortest path and showed its effectiveness by exhaustive simulations on well-known real and synthetic networks. In Ref. 12, two sampling methods are presented by Peng *et al.*, the former is an improved version of the stratified random sampling method and selects the high degree vertices with higher probability by classifying the nodes according to their degree distribution and the latter is an improved version of SS method to sample the targeted vertices selectively. On the basis of SS, a sampling method that uses multiple random snowballs and the *Cohen* process is developed by Gao *et al*[32]. Simulations on computer generated networks showed that their sampling method can preserve local and global structures of the network.

One may also classify the sampling algorithms into two groups: (i) one-phase sampling methods which construct the sampled network by random selection of edges/nodes or using some kind of graph traversal procedure (e.g. BFS,[19] RWS,[16] FFS [20] and RDS,[21] to name a few); (ii) two-phase sampling methods which construct sampled networks using a graph traversal procedure and some pre- or post-processing (e.g. DLAS,[30] SSP,[31] RPN[33] and DPL,[33] to mention a few). The first group of sampling algorithms is relatively simple and has low cost, low accuracy and fails to perform well on all kinds of networks. The second group uses additional

pre- or post- processing in order to get information about networks such as classifying important nodes into groups based on Katz centrality measures,[12] scoring important nodes by PageRank,[33] extracting groups of nodes in network[34] and extracting communities[33] to achieve higher accuracy. Such pre- or post-processing of course increases the cost of the sampling algorithms which must be paid if achieving higher accuracy is the goal. It should also be noted that the sampled network, once constructed can be used many times for various applications and analyses. The algorithms proposed in this paper fall into the second category.

2.3. *Basic idea*

The proposed sampling algorithms try to highlight some important edges by using a set of spanning trees. The algorithm is based on the hypothesis that by considering a set of arbitrary number of spanning trees with different starting points, edges with higher number of presence in these spanning trees can be considered as important edges which reflect several characteristics of the original network. Some of the evidence that supports the idea that using spanning trees may be a promising approach for designing effective sampling methods for social networks is briefly given below.

- Ansari *et al.* in Ref. 35 has shown that even with a random spanning tree, modular structure and topological characteristics of a network can be captured and the topological organization of the network can be visualized easily by properties of its spanning trees.
- Experimentations have shown the importance of the spanning trees to overcome the problems of propagating information in a network. Spanning trees are used in building efficient and reliable edge state propagation. This means that a spanning tree is a good structure that is able to propagate information in a network completely.[36,37]
- A spanning tree is able to embed connections between vertices by one pathway[38] and is also able to provide the underlying structure for any algorithm of chaining.[39] Bearman *et al.* in Ref. 40 investigated romantic relationships between 800 adolescents over an 18 month period in a social network, and found out that in a large spanning tree, all romantic relationships can be embedded. This work shows the ability of spanning trees in revealing important information of a graph so that a long and large chain of communications among a large population can be embedded in a spanning tree.
- In Refs. 41 and 42, the authors represented brain networks by a minimum spanning tree (MST) as a promising solution for unbiased characterization of complex brain networks. Using MST, a unique acyclic sub-graph is generated that connects all nodes and maximizes a property of interest, such as synchronization between brain areas that preserves global and local properties of brain networks.

3. Proposed Sampling Algorithms

3.1. *Social network sampling using spanning trees (SST)*

The first proposed algorithm starts by constructing k different spanning trees. The starting vertex of each spanning tree is chosen randomly from the set of vertices of the input graph. After constructing the spanning trees, a rank is assigned to each edge appearing in the set of spanning trees based on the number of spanning trees along which that edge has appeared. The sampled network will be constructed by successively adding the nodes which are incident on edges selected from the beginning of the list of edges ordered according their rank as long as the number of nodes in the sample graph being constructed does not exceed φ fraction of the nodes of the network. Every time a node is added, the edges of that node to the nodes previously added are established. Figure 1 gives the pseudo-code of the proposed sampling algorithm called SST.

The time needed by the proposed sampling algorithm consists of two parts: (i) the time needed for computation of spanning trees. (ii) The time needed to sort the list of edges (L_s) appearing in the set of spanning trees. The time needed for constructing all the spanning trees is $O(kn^2)$, where k ($k \ll n$) is the total number of spanning trees constructed by the algorithm and $n = |V|$. The time for sorting list L_s in the worst case is $O(n^2 \log n)$ and hence the time complexity for the algorithm is $O(kn^2) + O(n^2 \log n) = O(n^2 \log n)$.

Algorithm 1. Sampling with spanning trees (SST)

Input: Graph $G = \langle V,E \rangle$, sampling rate φ
Assumptions
k: the maximum number of constructed spanning trees;
L_s: list of selected vertices with size of n;
L: sorted version of L_s;
t: the iteration number;
Output: Sampled network $G_s = \langle V_s, E_s \rangle$
Begin
$t \leftarrow 1$;
Let τ_t denotes the t^{th} spanning tree found at iteration t;
While ($t < k$)
Select a vertex randomly from V as the starting vertex of spanning tree τ_t;
Construct the spanning tree τ_t;
$t \leftarrow t+1$;
End While
$L_s \leftarrow$ Compute the rank for every edge appearing in the set of k constructed spanning trees;
$L \leftarrow$ Sort L_s in descending order;
$G_s \leftarrow$ construct a sample by successively adding the nodes which are incident on edges which are selected from the beginning of the list L as long as the number of nodes in the sampled network being constructed does not exceed a particular percent of the nodes of the network. Every time a node is added, the edges of that vertex to the vertices previously added are established.
End algorithm

Fig. 1. The pseudo-code for the first proposed sampling algorithm (SST).

Algorithm 2. Sampling with partial spanning trees (PSST)

Procedure *β-partial spanning tree* (*G, v,* β, τ): // τ is empty in the first iteration
Add *v* to τ
  for all edges from *v* to *w* in neighbors of *v* **do**
    if vertex *w* is not present in τ and number of nodes in τ less than β × |V| **then**
      β-partial spanning tree (*G, w,* β, τ)
    end if
  end for
End algorithm

Fig. 2. The pseudo-code for constructing β-partial spanning tree.

3.2. *Social network sampling using β-partial spanning tree*

The first proposed sampling algorithm in the previous section visits each node of the network several times. It seems that visiting each node several times is not practical; therefore, we propose the second sampling algorithm using *partial spanning trees* as a remedy to improve the first proposed sampling algorithm in practice by reducing the number of times that each node in the network is visited. A *β-partial spanning* tree is defined as follows.

Definition. A *β-partial spanning tree* τ with root vertex v of graph G is a sub-graph of graph G which is a tree with vertex root v whose number of nodes is $\beta \times |V|$ for $0 < \beta < 1$. The size of β is proportional to the given sampling rate.

The algorithm given in Fig. 2 computes a β-partial spanning tree for a given β. The revised version of SST uses β-partial spanning trees instead of spanning trees. This improvement results in speeding up the sampling algorithm without sacrificing the performance as indicated in the experiments.

4. Simulation Results

To show the performance of the proposed sampling algorithms several numerical simulations are conducted on several real networks: *Robots.net*,[43] *Squeak Foundation.org*,[43] Epinions1,[43] *Cit-HepPh*,[44] *Email-EuAll*[45] and *Slashdot0902*.[45] The results are compared with other popular sampling algorithms such as RVS,[46] random edge sampling (RES),[11] RWS, MHRW sampling[47] and spiral sampling (SS).[28] The graph instances that are used in this set of experiments are shown in Table 1.

4.1. *Evaluation*

For evaluation of sampling algorithms, network statistics are used to measure the quality of parameters or representativeness of the sampled network. These network statistics are focused on a distribution of network characteristics like vertices, edges and sub-graphs.[48] Two well-known and mostly used network statistical properties: *clustering coefficient distribution* (*ccd*) as a local statistical property and *degree distribution* (*dd*) as a global statistical property are used for testing and comparing sampling algorithms. DD represents the fraction of vertices with degree k, for all

$k > 0$. DD is used for understanding the connectivity of graphs. Clustering coefficient of a vertex is the number of triangles centered on that vertex. CCD represents the fraction of vertices with clustering coefficient c.[16] This parameter enables measuring the strength of paths between a vertex and its nearest neighbors in a network.[12] The distance between both distributions of the statistical property for initial network G and sampled network G_s are computed according to a number of distance functions, including KSD, SDD and ND. These distance measures are described in the rest of this subsection.[49] Also, for assessing the efficiency of sampling algorithms in terms of cost, a new measure is defined which reflects the cost of each algorithm during the process of constructing a sampled network.

4.1.1. *KSD*

K-SD-statistic is one of the statistical test algorithms used to assess the distance between two cumulative distribution functions (CDF). KSD is a measure for acceptability between original distribution F_1 and estimated distribution F_2. The result of this test is a value between 0 and 1. The closer it is to zero, the higher the similarity between two distributions.[50] This measure is defined as

$$KSD(F_1, F_2) = max_x|F_1(x) - F_2(x)|, \tag{1}$$

where F_1 and F_2 denote CDF of the original distribution and the estimated distribution. This measure is used for both DD and clustering coefficient. Also, x represents the range of the random variables. Hence, it is computed as the maximum vertical distance between the two distributions.[51]

4.1.2. *ND*

In some cases, for evaluation, the distance between two positive m-dimensional real vectors F_1 and F_2 are measured, where F_1 is the original vector and F_2 is the estimated vector. ND[52] is defined as follows:

$$\mathrm{ND}(F_1, F_2) = \frac{\|F_1 - F_2\|}{\|F_2\|}. \tag{2}$$

4.1.3. *SDD*

SD is used to assess the difference between two probability density functions (PDFs). SD uses *Kullback–Leibler* (KL) divergence between two PDFs F_1 and F_2 that do not have continuous support over the full range of values (e.g. skewed degree). KL measures the average number of extra bits required to represent samples from the original distribution when using the sampled distribution. However, since KL divergence is not defined for distributions with different areas of support, SD smooths the two PDFs before computing the KL divergence[53]

$$SD(F_1, F_2) = KL[\alpha F_1 + (1 - \alpha)F_2||\alpha F_2 + (1 - \alpha)F_1], \tag{3}$$

where α is a constant in the range [0, 1] and KL divergence of P from Q calculate as follows:

$$\mathrm{KL}(P||Q) = \sum_i \ln\left(\frac{P_i}{Q_i}\right) P_i. \tag{4}$$

4.1.4. *Cost*

To assess the cost of a sampling algorithm, a new measure for the cost of a sampling algorithm has been defined as the total number of times that the edges in the graph are traversed during the traversal of the graph (C_t) plus the total number of operations performed during the pre/post-processing phase (C_p) divided by the total number of edges in the graph as given by Eq. 5.

$$Cost = \frac{c_1 C_t + c_2 C_p}{|E|}, \tag{5}$$

where constant c_1 is the coefficient cost of traversing an edge and constant c_2 is the coefficient cost of performing an operation during the pre/post-processing phase. In the proposed algorithm, the cost of additional processing is the cost of comparisons required for ranking the traversed edges.

4.2. *Experimental results*

To investigate the performance of the proposed algorithms, several experiments are conducted on several datasets described in Table 1 and an average result over 30 runs is reported for each experiment.

4.2.1. *Experiment I*

This experiment is conducted to study the impact of different strategies for choosing the starting vertices for spanning trees in the first proposed sampling algorithm. For this purpose, in addition to random starting vertex strategy (called *Rnd*), three other strategies for choosing starting vertices are also studied. The first strategy called high degree centrality (HDC) uses vertices with high degrees which are considered as hub-nodes in the network. The second strategy called high betweenness centrality (HBC)

Table 1. Description of test networks.

Network	Vertex	Edge	Description
Robots.net	1706	3561	News and discussion site about robots (social networks)
Squeak Foundation.org	1133	5451	Mailing list for answers to even the most basic questions
Epinions1	75879	508837	A site for answers to even the most basic questions
Cit-HepPh	34546	421578	Physics theory citation network
Email-EuAll	265214	420045	A network of research institution
Slashdot0902	82168	948464	A snapshot of Slashdot Zoo social network from February 2009

uses vertices with high betweenness, defined as the vertices through which a high number of shortest paths from all vertices to all others have passed. A vertex with HBC has a large influence on the transfer of information through the network.[54] The third strategy called *Per* uses peripheral vertices (vertices with lowest degrees) as starting vertex. Results of this experiment for *Robots.net*, *Squeak-foundation*, *Epinions1*, *CitHepPh*, *Email-EuAll* and *Slashdot0902* with respect to KSD for *dd*, KSD for *ccd*, SDD for *dd*, SD for *ccd*, ND for *dd* and ND for *ccd* for the sampling rate of 30% are given in Table 2. Based on the results, we can conclude that the proposed sampling algorithm is not sensitive to the strategy of choosing starting nodes.

4.2.2. *Experiment II*

This experiment is carried out to compare the performance of the first proposed sampling algorithm using spanning trees SST (with 20 spanning trees) and the second proposed sampling algorithm using partial spanning trees (PSST) with

Table 2. The results for the proposed algorithm for different starting node selection strategies for sampling rate = 30%.

		Distance functions					
		KS		SD		ND	
Networks	Strategies	*dd*	*ccd*	*dd*	*ccd*	*dd*	*ccd*
Robots.net	Rnd	0.01	0.11	0.97	0.8	0.03	0.19
	HDC	0.01	0.11	0.98	0.79	0.03	0.19
	HBC	0.01	0.11	0.98	0.79	0.03	0.19
	Per	0.01	0.11	0.98	0.79	0.03	0.19
Squeak-foundation	Rnd	0.16	0.02	1.23	0.97	0.34	0.04
	HDC	0.16	0.02	1.23	0.97	0.34	0.04
	HBC	0.16	0.02	1.23	0.97	0.34	0.04
	Per	0.16	0.02	1.23	0.97	0.34	0.04
Epinions1	Rnd	0.27	0.01	0.99	0.88	0.68	0.03
	HDC	0.27	0.01	0.99	0.88	0.68	0.03
	HBC	0.27	0.01	0.99	0.88	0.68	0.03
	Per	0.27	0.01	0.99	0.88	0.68	0.03
CitHepPh	Rnd	0.39	0.08	0.72	1.68	0.58	0.14
	HDC	0.39	0.08	0.72	1.68	0.58	0.14
	HBC	0.39	0.08	0.72	1.68	0.58	0.14
	Per	0.39	0.08	0.72	1.68	0.58	0.14
Slashdot0902	Rnd	0.34	0.22	1.42	1.38	0.96	0.44
	HDC	0.34	0.22	1.42	1.38	0.96	0.44
	HBC	0.34	0.22	1.42	1.38	0.96	0.44
	Per	0.34	0.22	1.42	1.38	0.96	0.44
Email-EuAll	Rnd	0.19	0.07	0.52	0.98	0.24	0.14
	HDC	0.19	0.07	0.52	0.98	0.24	0.14
	HBC	0.19	0.07	0.52	0.98	0.24	0.14
	Per	0.19	0.07	0.52	0.98	0.24	0.14

random vertex sampling as RVS,[46] random edge sampling as RES,[11] random walk sampling as RWS, Metropolis–Hastings random walk sampling as MHRW[47] and spiral sampling as SS[28] in terms of KSD for *dd*, KSD for *ccd*, SDD for *dd*, SDD for *ccd*, ND for *dd* and ND for *ccd* for sampling rate 15%. Each reported result is an average over 30 runs. We note that the size of beta for the second proposed sampling algorithm using PSST is proportional to the sampling rate. In order to provide statistical confidence, we performed a parametric test (t-Test with 58 degrees of freedom at the 95% significance level). The t-Test result is shown as "✓, "×" or "∼" when SST is better than, worse than or similar to that of the corresponding sampling method, respectively. Results of this experiment are shown in Tables 3–8. Best results are highlighted in boldface. The last columns of the following tables show the percentage of other algorithms that SST works better than with respect to a particular performance measure. From these results, we may conclude that the proposed sampling algorithms in most cases perform better than other sampling algorithms in terms of KSD, SDD and ND. Also, the first proposed sampling algorithm outperforms the second proposed sampling algorithm.

4.2.3. *Experiment III*

To answer the question that whether or not the ranking of the traversed edges computed by the first proposed sampling method is an approximation of edge betweenness, we conducted an experiment to investigate the conjecture that the top $K\%$ of edges ranked by the edge betweenness centrality and top $k\%$ of edges ranked by the proposed algorithm (SST) are the same. For this purpose, the above two ordered lists are computed for different values of $K \in (1, 5, 10)$. Similarity of these two ordered lists is computed using Eq. 6 as given below

$$PrK = \frac{|\mathrm{Top}K(L) \cap \mathrm{Top}K(L')|}{K}, \tag{6}$$

where L and L' are the ordered edge list obtained from the proposed algorithm (SST) and edge betweenness centrality, respectively. The results of this experiment are summarized in Table 9. In Table 9, we see that the similarity result between the ordered list based on edge betweenness centrality and the ordered list computed by the proposed algorithm increases as K increases. As indicated in Table 9, the highest similarity obtained is 0.34 and for this reason, we may say that edge ranking computed by the proposed algorithm cannot be a good approximation of edge betweenness.

4.2.4. *Experiment IV*

In this experiment, efficiency of the proposed sampling algorithms (SST and PSST) are compared with some of the existing two-phase sampling algorithms including shortest path sampling (SPS)[31] and DLAS.[30] For this experiment, sampling rate is set to 30%. The cost of algorithms increase as the number of computed shortest paths

Table 3. Comparison of different sampling algorithms for *Robots.net* for sampling rate = 15%.

Distance function	Statistical property	Sampling algorithms							
		RVS	RWS	RES	MHRW	SS	PSST	SST	
KSD	*dd*	0.60 ± 0.01 ✓	0.06 ± 0.02 ✓	0.33 ± 0.02 ✓	0.19 ± 0.07 ∼	$\mathbf{0.15} \pm 0.08$ ×	0.20 ± 0.01 ∼	0.19 ± 0.01	80%
	ccd	0.27 ± 0.06 ✓	0.13 ± 0.05 ✓	0.20 ± 0.01 ✓	0.24 ± 0.08 ✓	0.29 ± 0.1 ✓	0.16 ± 0.03 ✓	$\mathbf{0.10} \pm 0.04$	100%
SDD	*dd*	0.85 ± 0.16 ✓	0.91 ± 0.03 ×	0.70 ± 0.00 ✓	0.84 ± 0.16 ✓	$\mathbf{0.95} \pm 0.21$ ×	1.61 ± 0.03 ✓	1.14 ± 0.07	60%
	ccd	0.02 ± 0.00 ✓	0.79 ± 0.03 ✓	0.76 ± 0.00 ✓	0.79 ± 0.05 ✓	$\mathbf{0.85} \pm 0.17$ ×	0.77 ± 0.01 ✓	0.82 ± 0.05	80%
ND	*dd*	1.09 ± 0.07 ✓	$\mathbf{0.11} \pm 0.04$ ×	0.35 ± 0.00 ✓	0.32 ± 0.12 ✓	0.27 ± 0.14 ✓	0.37 ± 0.02 ✓	0.26 ± 0.02	80%
	ccd	0.38 ± 0.00 ✓	0.23 ± 0.00 ✓	$\mathbf{0.14} \pm 0.00$ ×	0.43 ± 0.00 ✓	0.51 ± 0.00 ✓	0.29 ± 0.00 ✓	0.17 ± 0.00	80%

Table 4. Comparison of different sampling algorithms for *Squeak foundation* for sampling rate = 15%.

		Sampling algorithms							
Distance function	Statistical property	RVS	RWS	RES	MHRW	SS	PSST	SST	
KSD	*dd*	0.18 ± 0.01	0.19 ± 0.03	0.42 ± 0.01	0.35 ± 0.09	0.36 ± 0.11	$\mathbf{0.09} \pm 0.01$	$\mathbf{0.16} \pm 0.00$	100%
		✓	✓	✓	✓	✓	×		
	ccd	0.19 ± 0.05	0.13 ± 0.03	0.15 ± 0.02	0.19 ± 0.07	0.23 ± 0.06	$\mathbf{0.02} \pm 0.01$	$\mathbf{0.02} \pm 0.01$	100%
		✓	✓	✓	✓	✓	∼		
SDD	*dd*	0.86 ± 0.03	0.84 ± 0.01	0.71 ± 0.00	0.74 ± 0.05	0.74 ± 0.06	$\mathbf{1.05} \pm 0.05$	1.23 ± 0.01	60%
		×	×	✓	✓	✓	×		
	ccd	0.94 ± 0.02	0.93 ± 0.01	0.93 ± 0.00	$\mathbf{0.99} \pm 0.08$	1.02 ± 0.08	$\mathbf{0.99} \pm 0.02$	0.97 ± 0.02	60%
		✓	✓	✓	×	×	×		
ND	*dd*	0.36 ± 0.06	0.38 ± 0.07	0.73 ± 0.02	0.27 ± 0.14	0.64 ± 0.18	$\mathbf{0.14} \pm 0.01$	0.34 ± 0.01	80%
		✓	✓	✓	×	✓	×		
	ccd	0.38 ± 0.00	0.24 ± 0.00	0.29 ± 0.00	0.51 ± 0.00	0.45 ± 0.00	0.43 ± 0.00	$\mathbf{0.04} \pm 0.00$	100%
		✓	✓	✓	✓	✓	✓		

Table 5. Comparison of different sampling algorithms for *CitHepPh* for sampling rate = 15%.

Distance function	Statistical property	Sampling algorithms							
		RVS	RWS	RES	MHRW	SS	PSST	SST	
KSD	*dd*	0.81 ± 0.01 ✓	0.17 ± 0.00 ×	$\mathbf{0.07} \pm 0.00$ ×	0.07 ± 0.02 ×	0.62 ± 0.01 ✓	0.46 ± 0.00 ×	0.27 ± 0.00	40%
	ccd	0.05 ± 0.03 ✓	0.02 ± 0.00 ✓	$\mathbf{0.00} \pm 0.00$ ∼	0.05 ± 0.01 ✓	0.01 ± 0.01 ∼	0.04 ± 0.00 ✓	0.01 ± 0.00	100%
SDD	*dd*	0.88 ± 0.01 ✓	0.85 ± 0.01 ✓	$\mathbf{0.99} \pm 0.00$ ∼	$\mathbf{0.99} \pm 0.00$ ∼	0.59 ± 0.00 ✓	0.71 ± 0.00 ✓	$\mathbf{0.99} \pm 0.00$	60%
	ccd	0.76 ± 0.10 ✓	1.11 ± 0.02 ×	$\mathbf{1.04} \pm 0.03$ ×	0.75 ± 0.03 ✓	1.06 ± 0.08 ×	1.27 ± 0.00 ✓	0.88 ± 0.00	40%
ND	*dd*	0.00 ± 0.17 ×	0.32 ± 0.01 ✓	0.24 ± 0.01 ✓	$\mathbf{0.19} \pm 0.02$ ×	0.95 ± 0.00 ✓	0.82 ± 0.00 ✓	0.68 ± 0.00	80%
	ccd	0.09 ± 0.00 ✓	0.04 ± 0.00 ✓	$\mathbf{0.01} \pm 0.00$ ×	0.08 ± 0.00 ✓	0.02 ± 0.00 ∼	0.08 ± 0.00 ✓	0.03 ± 0.00	80%

Table 6. Comparison of different sampling algorithms for *Epinions1* for sampling rate = 15%.

		Sampling algorithms							
Distance function	Statistical property	RVS	RWS	RES	MHRW	SS	PSST	SST	
KSD	*dd*	0.18 ± 0.02	0.17 ± 0.00	0.45 ± 0.00	0.39 ± 0.02	0.44 ± 0.01	$\mathbf{0.10} \pm 0.00$	0.39 ± 0.00	60%
		×	×	✓	∼	✓	×		
	ccd	0.10 ± 0.03	0.10 ± 0.00	0.02 ± 0.00	0.08 ± 0.01	0.08 ± 0.01	$\mathbf{0.02} \pm 0.00$	0.08 ± 0.00	80%
		✓	✓	×	∼	∼	×		
SDD	*dd*	1.26 ± 0.07	0.85 ± 0.01	0.69 ± 0.00	0.72 ± 0.00	0.69 ± 0.00	1.12 ± 0.00	$\mathbf{0.92} \pm 0.00$	100%
		✓	✓	✓	✓	✓	✓		
	ccd	2.38 ± 1.13	$\mathbf{1.11} \pm 0.02$	1.98 ± 0.03	1.68 ± 0.09	1.69 ± 0.08	1.12 ± 0.00	1.68 ± 0.00	80%
		✓	×	✓	∼	✓	×		
ND	*dd*	0.34 ± 0.06	0.32 ± 0.01	0.71 ± 0.01	0.58 ± 0.02	0.68 ± 0.03	$\mathbf{0.22} \pm 0.00$	0.28 ± 0.00	100%
		✓	✓	✓	✓	✓	×		
	ccd	0.18 ± 0.00	0.15 ± 0.00	0.19 ± 0.02	$\mathbf{0.04} \pm 0.00$	0.14 ± 0.00	$\mathbf{0.04} \pm 0.00$	$\mathbf{0.04} \pm 0.00$	80%
		✓	✓	✓	∼	✓	∼		

Table 7. Comparison of different sampling algorithms for *Slashdot0902* for sampling rate = 15%.

Distance function	Statistical property	Different methods							
		RVS	RWS	RES	MHRW	SS	PSST	SST	
KSD	*dd*	0.79 ± 0.01 ✓	$\mathbf{0.04} \pm 0.00$ ×	0.43 ± 0.00 ✓	0.39 ± 0.02 ✓	0.40 ± 0.01 ✓	$\mathbf{0.08} \pm 0.00$ ✓	0.05 ± 0.00	80%
	ccd	0.28 ± 0.02 ✓	0.05 ± 0.00 ✓	0.35 ± 0.00 ✓	0.30 ± 0.01 ✓	0.32 ± 0.01 ✓	$\mathbf{0.08} \pm 0.00$ ✓	$\mathbf{0.03} \pm 0.00$	100%
SDD	*dd*	1.03 ± 0.02 ✓	$\mathbf{0.97} \pm 0.01$ ∼	0.77 ± 0.00 ✓	0.80 ± 0.00 ✓	0.79 ± 0.00 ✓	0.94 ± 0.00 ✓	$\mathbf{0.97} \pm 0.00$	100%
	ccd	1.77 ± 0.18 ✓	$\mathbf{1.00} \pm 0.02$ ∼	2.11 ± 0.03 ✓	1.68 ± 0.14 ✓	1.85 ± 0.25 ✓	$\mathbf{1.02} \pm 0.00$ ✓	$\mathbf{1.00} \pm 0.00$	100%
ND	*dd*	2.16 ± 0.03 ✓	$\mathbf{0.10} \pm 0.00$ ∼	0.76 ± 0.01 ✓	0.66 ± 0.04 ✓	0.71 ± 0.03 ✓	$\mathbf{0.18} \pm 0.00$ ✓	$\mathbf{0.10} \pm 0.00$	100%
	ccd	0.56 ± 0.00 ✓	$\mathbf{0.04} \pm 0.00$ ×	0.71 ± 0.00 ✓	0.61 ± 0.00 ✓	0.65 ± 0.00 ✓	$\mathbf{0.17} \pm 0.00$ ✓	$\mathbf{0.06} \pm 0.00$	80%

Table 8. Comparison of different sampling algorithms for *Email-EuAll* for sampling rate = 15%.

Distance function	Statistical property	Different methods							
		RVS	RWS	RES	MHRW	SS	PSST	SST	
KSD	*dd*	0.18 ± 0.02	$\mathbf{0.18} \pm 0.00$	0.14 ± 0.02	0.21 ± 0.02	0.22 ± 0.01	0.19 ± 0.00	0.19 ± 0.00	80%
		✓	×	✓	✓	✓	∼		
	ccd	0.10 ± 0.03	0.08 ± 0.06	$\mathbf{0.03} \pm 0.00$	0.08 ± 0.01	0.09 ± 0.01	0.07 ± 0.00	0.07 ± 0.00	100%
		✓	✓	×	✓	✓	∼		
SDD	*dd*	$\mathbf{1.26} \pm 0.07$	0.58 ± 0.33	0.57 ± 0.03	0.50 ± 0.00	0.49 ± 0.00	0.52 ± 0.00	0.52 ± 0.00	40%
		×	×	×	✓	✓	∼		
	ccd	2.38 ± 1.13	3.44 ± 3.18	$\mathbf{0.98} \pm 0.03$	$\mathbf{0.98} \pm 0.14$	$\mathbf{0.98} \pm 0.00$	$\mathbf{0.98} \pm 0.00$	$\mathbf{0.98} \pm 0.00$	100%
		✓	✓	∼	∼	∼	∼		
ND	*dd*	0.34 ± 0.06	$\mathbf{0.24} \pm 0.01$	0.18 ± 0.02	0.26 ± 0.04	0.29 ± 0.00	$\mathbf{0.24} \pm 0.00$	$\mathbf{0.24} \pm 0.00$	80%
		✓	∼	×	✓	✓	∼		
	ccd	0.18 ± 0.00	0.16 ± 0.00	$\mathbf{0.06} \pm 0.00$	0.16 ± 0.00	0.18 ± 0.00	0.14 ± 0.00	0.14 ± 0.00	80%
		✓	✓	×	✓	✓	∼		

Table 9. Comparison of correlation between proposed algorithm and edge betweenness.

	Data sets					
K(%)	Robots.net	Squeak foundation	CitHepPh	Epinions1	Slashdot0902	Email-Eu All
1	0.05	0.09	0.02	0.06	0.05	0.01
5	0.21	0.19	0.03	0.08	0.21	0.26
10	0.34	0.23	0.06	0.13	0.25	0.31

or the number of computed spanning trees increase. The comparison is performed with respect to accuracy of sampling algorithms in terms of KSD for *dd* and KSD for *ccd* vs. sampling cost (as defined by Eq. (5)). The results of this experiment for different two-phase sampling algorithms are presented in Figs. 3 and 4 in terms of KSD for *dd* and KSD for *ccd*, respectively. As it is shown, increasing the sampling cost results in decreasing values of KSD for *dd* and KSD for *ccd*. The results indicate that with the same cost, the second proposed sampling algorithm (PSST) performs better than other two-phase sampling algorithms at least for five out of six networks in terms of KDS for *dd* and four out of six networks in terms of KSD for *cdd*. For some networks, such as CitHepPh, SPS sometimes outperforms the first proposed

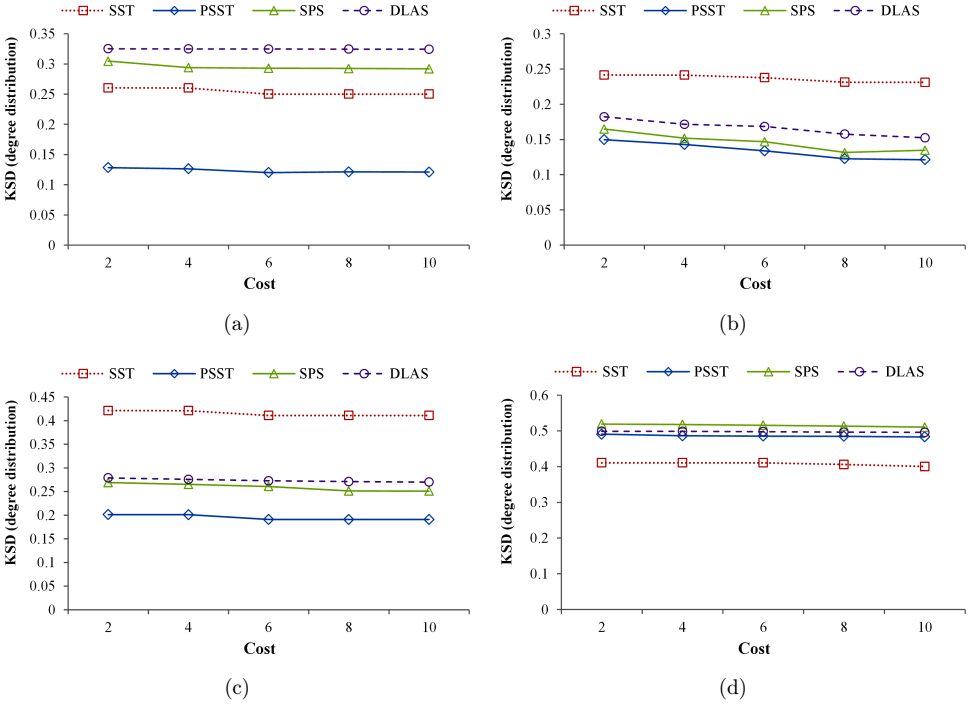

Fig. 3. (Color online) Comparing accuracy of sampling algorithms in terms of KSD for *dd* versus the sampling cost. (a) Robots.net, (b) Squeak foundation, (c) Epinions1, (d) CitHepPh, (e) Email-EuAll and (f) Slashdot0902.

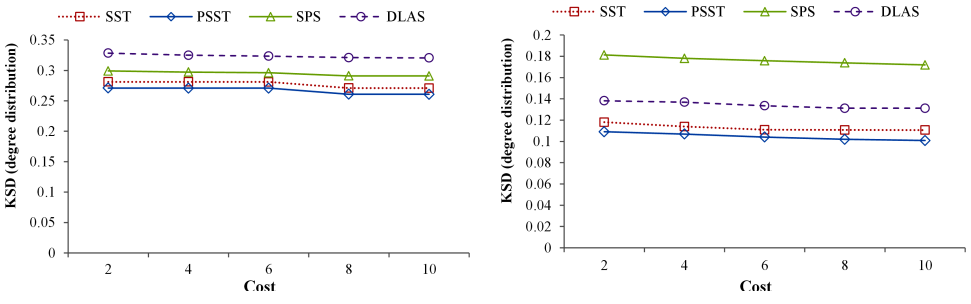

Fig. 3. (*Continued*)

sampling algorithm (SST) in terms of KSD for *ccd*. For Robots.net network, SST and PSST yield similar results for same costs in terms of KDS for *ccd*. Also note the result given in Figs. 5 and 6. In these figures, accuracies that can be obtained for one-phase sampling algorithms, such as RVS, RES, RWS and MHRWS when sampling rate is 30% are given. As shown, one-phase algorithms cannot compete with two-phase

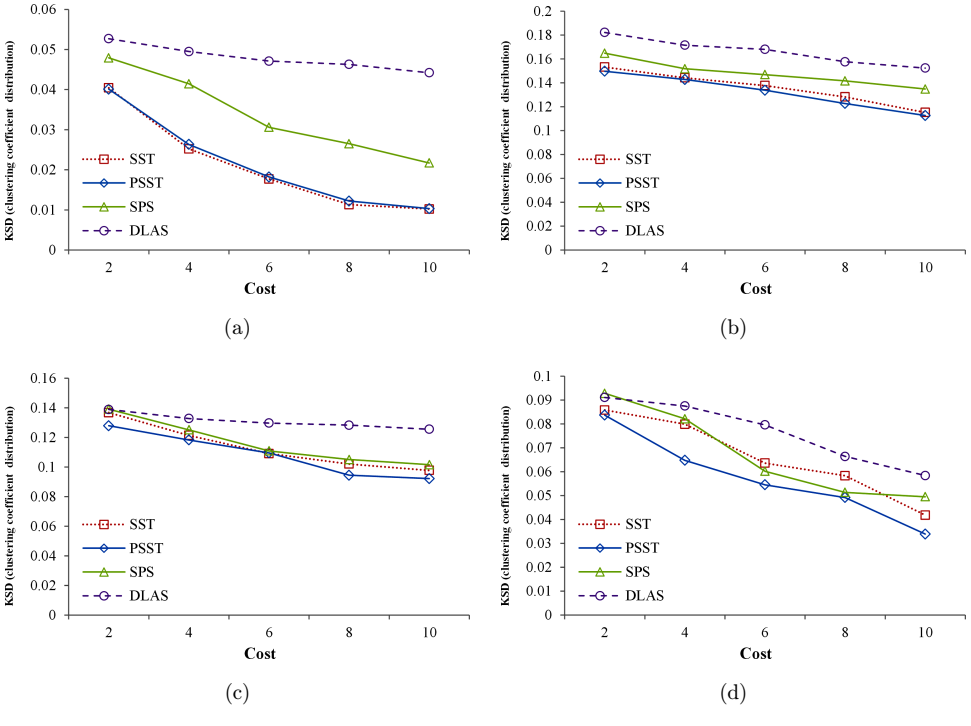

Fig. 4. (Color online) Comparing accuracy of two-phase sampling algorithms (SST, PSST, SPS and DLAS) in terms of KSD for *ccd* versus the sampling cost. (a) Robots.net, (b) Squeak foundation, (c) Epinions1, (d) CitHepPh, (e) Email-EuAll and (f) Slashdot0902.

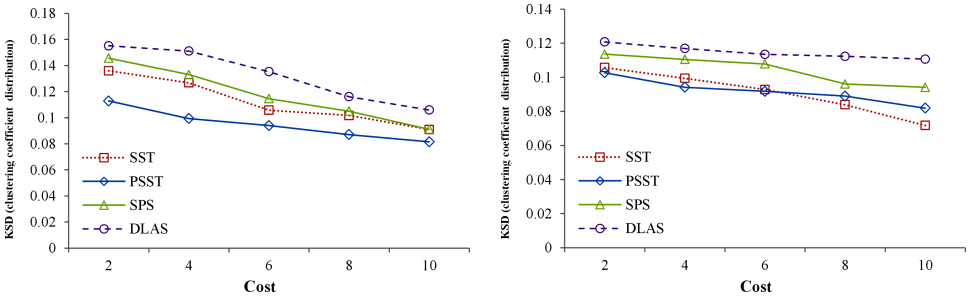

Fig. 4. (*Continued*)

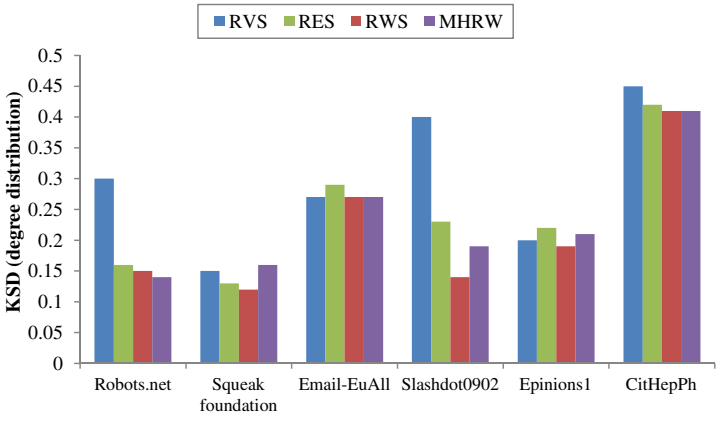

Fig. 5. (Color online) Accuracy of one-phase sampling algorithms (RVS, RES, RWS and MHRWS) in terms of KSD for *dd* for sampling rate = 30%.

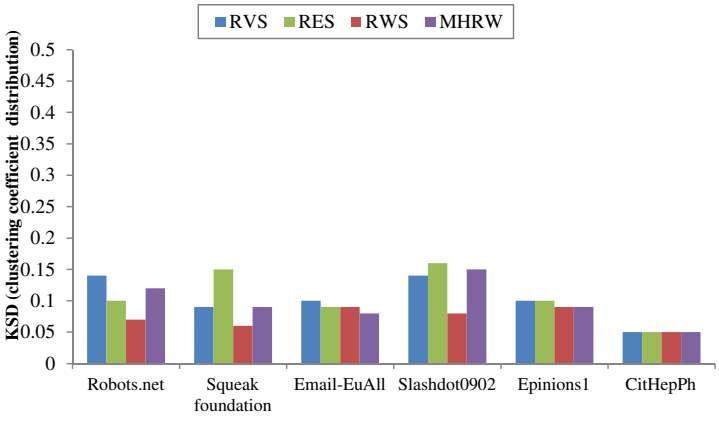

Fig. 6. (Color online) Accuracy of one-phase sampling algorithms (RVS, RES, RWS and MHRWS) in terms of KSD for *ccd* for sampling rate = 30%.

algorithms in terms of performance. It should be noted that a sampled network, once created, can be used over and over in many different analyses. Therefore, when choosing a sampling algorithm, although the cost is an important factor, it is more important to have a sampling algorithm that creates a sampled network with the properties similar to those of the original network. The proposed sampling algorithms require additional time for getting information about the original network similar to other two-phase sampling algorithms. Even if cost is higher, the proposed algorithm demonstrates far superior performance in generating a sampled network with properties quite similar to those of an original network.

5. Conclusion

In this paper, several algorithms based on the concept of spanning trees for sampling social networks were proposed. The proposed sampling algorithms were based on the idea that edges appearing in the spanning trees can represent the most structural information of a network. Since, the first proposed sampling algorithm visits each vertex of the network several times; we proposed the second sampling algorithm using *PSST* as a remedy to improve the first proposed sampling algorithm in practice by reducing the number of times that each vertex in the network is visited. The performance of the proposed sampling algorithms was investigated by conducting several experiments on well-known real social network data sets. The obtained results showed that the proposed sampling algorithms perform better than some of the existing algorithms, such as RVS, RES, RWS, MHRWS, Spiral Sampling, SPS and DLAS in terms of KSSD and ND.

References

1. G. Lin, Z. Di and Y. Fan, *Int. J. Mod. Phys. C* **25**(5) (2014) 1440005.
2. Z. Zavareh and E. Almaas, *Microbial Syst. Biol.* **881** (2012) 551.
3. L. Hongli, Y. Xie, H. Hu and Z. Chen, *Int. J. Mod. Phys. C* **25**(5) (2014) 1440004.
4. A. Rezvanian and M. R. Meybodi, *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **23**(1) (2015) 1–32.
5. M. Soleimani-Pouri, A. Rezvanian and M. R. Meybodi, *State Art Appl. Soc. Netw. Anal.* **14** (2014) 295.
6. M. Soleimani-Pouri, A. Rezvanian and M. R. Meybodi, Finding a maximum clique using ant colony optimization and particle swarm optimization in social networks, in *IEEE/ACM Int. Conf. Advances in Social Networks Analysis*, Istanbul, 2012.
7. M. Huisman, *J. Soc. Struct.* **35**(4) (2009) 1–29.
8. M. Papagelis, G. Das and N. Koudas, *IEEE Trans. Knowl. Data Eng.* **25**(3) (2013) 662.
9. M. Gjoka, C. T. Butts, M. Kurant and A. Markopoulou, *IEEE J. Sel. Areas Commun.* **29**(4) (2011) 1893–1905.
10. E. Volz and D. D. Heckathorn, *J. Off. Stat.-Stockholm* **24**(1) (2008) 79.
11. N. K. Ahmed, F. Berchmans, J. Neville and R. Kompella, Time-based sampling of social network activity graphs, in *8th Workshop on Mining and Learning with Graphs*, Washington, 2010.
12. P. Luo, Y. Li, C. Wu and G. Zhang, *Int. J. Mod. Phys. C* **26**(5) (2015) 1550050.

13. J. A. Torkestani, *J. Supercomput.* **64**(1) (2013) 226.
14. N. K. Ahmed, N. Duffield, J. Neville and R. Kompella, Graph sample and hold: A framework for big-graph analytics, in *20th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, New York, 2014.
15. L. Goodman, *Ann. Math. Stat.* **32** (1961) 148–170.
16. L. Lovász, in *Combinatorics, Paul Erdos is Eighty*, Vol. 2 (Janos Bolyai Mathematical Society, Keszthely, 1993), pp. 1–46.
17. M. R. Henzinger, A. Heydon, M. Mitzenmacher and M. Najork, On near-uniform URL sampling, in *9th Conf. Word Wide Web*, Amsterdam, 2000.
18. M. Kurant, M. Gjoka, C. T. Butts and A. Markopoulou, Walking on a graph with a magnifying glass: Stratified sampling via weighted random walks, in *ACM SIGMETRICS*, San Jose, 2011, pp. 281–292.
19. M. Kurant, A. Markopoulou and P. Thiran, *IEEE J. Sel. Areas Commun.* **29**(9) (2011) 1799–1809.
20. J. Leskovec and C. Faloutsos, Sampling from large graphs in *12th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Philadelphia, 2006.
21. D. Heckathorn, *Soc. Probl.* **44** (1997) 174–199.
22. C. Wejnert and D. D Heckathorn, *Sociol. Meth. Res.* 2008.
23. M. Gjoka, M. Kurant, C. T. Butts and A. Markopoulou, Walking in facebook: A case study of unbiased sampling of OSNs, in *IEEE INFOCOM*, San Diego, 2010.
24. B. Ribeiro, D. Figueiredo, E. de Souza e Silva and D. Towsley, Characterizing continuous-time random walks on dynamic networks, in *ACM SIGMETRICS Joint Int. Conf. Measurement and Modeling of Computer Systems*, San Jose, 2011, pp. 151–152.
25. S. W. Son, C. Christensen, G. Bizhani, D.V. Foster, P. Grassberger and M. Paczuski, *Phys. Rev.* **86**(14) (2012) 046104.
26. M. D. Siciliano, D. Yenigun and G. Ertan, *Soc. Netw.* **34**(9) (2012) 585–600.
27. B. Ribeiro, P. Wang, F. Murai and D. Towsley, Sampling directed graphs with random walks, in *IEEE INFOCOM*, Orlando, 2012.
28. C. A. Pina-Garcia and D. Gu, *Soc. Netw. Anal. Min.* **3**(3) (2013) 1869–5450.
29. H. Wang and J. Lu, Detect inflated follower numbers in OSN using star sampling, in *IEEE/ACM Int. Conf. Advances in Social Networks Analysis and Mining*, Niagara, 2013.
30. A. Rezvanian, M. Rahmati and M. R. Meybodi, *Physica A, Stat. Mech. Appl.* **396** (2014) 224–234.
31. A. Rezvanian and M. R. Meybodi, *Physica A, Stat. Mech. Appl.* **424** (2015) 254–268.
32. Q. Gao, X. Ding, F. Pan and W. Li, *Int. J. Mod. Phys. C* **25**(5) (2014) 1440007.
33. S. H. Yoon, K. N. Kim, J. Hong, S. W. Kim and S. Park, *Inf. Sci.* **306** (2015) 53–69.
34. N. Blagus, L. Subelj, G. Weiss and M. Bajec, *Physica A, Stat. Mech. Appl.* **432** (2015) 206–215.
35. N. Ansari, G. Cheng and R. N. Krishnan, *IEEE Commun. Lett.* **8**(5) (2004) 317–319.
36. P. A. Humblet and S. R. Soloway, *Comput. Netw. ISDN Syst.* **16**(3) (1989) 179–186.
37. B. Bellur and R. G. Ogier, A reliable, efficient topology broadcast protocol for dynamic networks, in *18th Annual Joint Conf. IEEE Computer and Communications Societies*, New York, 1999.
38. D. R. White and M. Newman, *Fast Approximation Algorithms for Finding Node-Independent Paths and k-components*, Santa Fe Institute Working Papers Series, 2001.
39. R. J. Hill, International comparisons using spanning trees, in *International and Interarea Comparisons of Income, Output, and Prices* (University of Chicago Press, Chicago, 1999).
40. P. S. Bearman, J. Moody and K. Stovel, *Am. J. Sociol.* **110**(1) (2004) 44–91.

41. C. J. Stam, P. Tewarie, E. Van Dellen, E. C. W. Van Straaten, A. Hillebrand and P. Van Mieghem, *Int. J. Psychophysiol.* **92**(3) (2014) 129.
42. P. Tewarie, E. Van Dellen, A. Hillebrand and C. J. Stam, *NeuroImage* **104** (2015) 177–188.
43. J. Leskovec, J. Kleinberg and C. Faloutsos, Graphs over time: Densification laws, shrinking diameters and possible explanations, in *The 11th ACM SIGKDD Int. Conf. Knowledge Discovery in Data Mining*, Chicago, 2005.
44. J. Leskovec, J. Kleinberg and C. Faloutsos, Graph evolution: Densification and shrinking diameters, *ACM Trans. Knowl. Discov. Data* **1**(1) (2007) 2.
45. J. Leskovec, K. J. Lang, A. Dasgupta and M. W. Mahoney, *Internet Math.* **6**(1) (2009) 29–123.
46. A. S. Maiya and T. Y. Berger-Wolf, Benefits of bias: Towards better characterization of network sampling, in *17th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Diego, 2011.
47. F. Murai, B. Riberio, D. Towsley and P. Wang, *IEEE J. Sel. Areas Commun.* **31**(6) (2013) 1017–1025.
48. N. K. Ahmed, J. Neville and R. Kompella, *ACM Trans. Knowl. Discov. Data* **8**(2) (2014) 7.
49. C. Seshdhri, A. Pinar and G. Kolda, An in-depth study of stochastic Kronecker graphs, in *11th Int. Conf. Data Mining*, Vancouver, 2011.
50. M. L. Goldstein, S. A. Morris and G. G. Yen, *Eur. Phys. J. B, Condens. Matter Complex Syst.* **41**(2) (2004) 255–258.
51. F. James, *Statistical Methods in Experimental Physics*, Vol. 7 (World Scientific, Singapore, 2006).
52. S. A. Terwijn, L. Torenvliet and P. Vitányi, *J. Comput. Syst. Sci.* **77**(4) (2011) 738–742.
53. L. Lee, *Artif. Intell. Stat.* (2001) 65–72.
54. U. Brandes, *Soc. Netw.* **30** (2008) 136–145.