



A spanning tree approach to social network sampling with degree constraints

Alireza Rezvanian¹ · S. Mehdi Vahidipour² · Zeinab S. Jalali³

Received: 28 November 2023 / Revised: 10 March 2024 / Accepted: 19 March 2024 / Published online: 18 May 2024
© The Author(s), under exclusive licence to Springer-Verlag GmbH Austria, part of Springer Nature 2024, corrected publication 2024

Abstract

Online social networks (OSNs) have become increasingly popular on the web in recent years. There are millions of users on these networks, and they generate a great deal of interaction among them. Network sampling is often used to study and measure OSNs, which produce a small, accessible network with a limited number of edges and nodes. In this paper, two algorithms are proposed to sample OSNs. The first algorithm finds several spanning trees with the highest number of nodes that satisfy a given degree constraint from randomly chosen root nodes; edges and nodes of computed spanning trees are ranked based on how often they appear in them. Finally, a fraction of the highly ranked nodes and edges are considered in the sampled network. Second, a partial spanning tree algorithm is proposed in place of a full spanning tree algorithm to improve the performance of the first algorithm. We conduct several experiments on well-known real networks to determine the performance of the proposed sampling algorithms. As a result of the experiments, the proposed algorithms outperformed some of the baseline and recently presented algorithms in terms of the Kolmogorov-Smirnov (KS) test, Skew Divergence (SD) distance, and Normalized Distance (ND).

Keywords Sampling · Spanning tree · Degree constraint spanning tree · Social networks

1 Introduction

The growth of online social networks (OSN) like *Facebook* and *Twitter* has enabled users to interact with their friends and share information in real-time. Several research projects rely on the study and characterization of OSNs. However, due to the large size and access limitations of these networks, many large OSNs are currently studied via sampling techniques (Kurant et al. 2012; Rezvanian et al. 2019). The sampling techniques also could be applied for simulation,

data mining, measuring performance of protocols, information retrieval, viral marketing experimentations, and fraud detection in social network analysis (Ahmed et al. 2010). Sampling techniques should consider every aspect of the initial network including the presence or absence of influential edges and nodes and the connection topology of these nodes and edges to present a sampled network that can preserve characteristics of the initial network. In sampling literature, most sampling techniques focus on either preserving influential nodes or influential edges, meaning they could not consider both of them. Besides, most existing sampling methods including random nodes and random edges work based on random selection, thus the topology of the original network is not considered. In this paper, we propose two novel sampling algorithms using several spanning trees with the highest number of nodes that satisfy a given degree constraint. The proposed algorithm not only takes into account important edges but also makes use of important nodes.

The proposed algorithm first finds several **Spanning Trees** with the **Highest Number** of nodes that satisfy a certain degree of **Constraint** (STHNC) from the input network. Based on the number of times each edge and each node in the set of computed spanning trees has appeared in the set of spanning trees, the edges and nodes with degrees greater

✉ Alireza Rezvanian
rezvanian@gmail.com; rezvanian@usc.ac.ir

S. Mehdi Vahidipour
vahidipour@kashanu.ac.ir

Zeinab S. Jalali
zsaghati@syr.edu

¹ Department of Computer Engineering, University of Science and Culture, Tehran, Iran

² Computer Engineering Department, Faculty of Electrical and Computer Engineering, University of Kashan, Kashan, Iran

³ Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA

than the given constraint are ranked. A sub-graph is constructed as the sampled network containing a fraction of nodes and edges that are highly ranked. According to the algorithm proposed, the edges with the highest number of presences in spanning trees of an arbitrary number of STHNCs with different starting points can be considered important edges that reflect several characteristics of the original network, since they reflect several characteristics of the original network. Besides STHNC, visits edges in a uniform manner which prevents the sampled network from being biased to high-degree nodes, but sometimes it is inevitable to have some nodes with degrees higher than the given constraint. These nodes preserve many connections, and in the absence of these nodes, many connections will be lost. So, considering these nodes as influential nodes helps better preserve the properties of the network.

We evaluate the efficiency of the proposed sampling algorithms via simulation. In terms of Kolmogorov-Smirnov (KS) test, Skew Divergence (SD) distance, and Normalized Distance (ND), we compare the results of the proposed algorithms with those reported in the literature recently. Our results show the superiority of our proposed algorithm over well-known algorithms in terms of several criteria.

The rest of the paper is organized as follows: In the second section, we discuss the essential information needed to understand the rest of the paper. A review of related works about sampling methods in complex networks is presented in the third section. The proposed methods are discussed in Sect. 4. In Sect. 5, simulation experiments on standard graphs of complex networks are used to assess the performance of the proposed algorithm. As a conclusion, Sect. 6 concludes the paper.

2 Preliminaries

Social networks can be represented as graph $G = \langle V, E \rangle$ where $V = \{v_1, v_2, \dots, v_n\}$ is the vertex set that represents the users of an OSN and $E = \{e_1, e_2, \dots, e_m\}$ is the set of edges that represent a kind of relationship between users in an OSN.

A sampling technique on input graph G can be defined as 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 $|V_s| = \varphi \times |V|$ (Jalali et al. 2016).

2.1 Evaluation metrics on sampling

A sampling algorithm's quality or representativeness is evaluated using network statistics. In network statistics, edges, nodes, and sub-graphs are taken into account.

Two well-known and widely used statistical properties provide insight into sampling algorithms: clustering coefficient distribution (CCD) as a local property and degree distribution (DD) as a global property. Graph connectivity is understood by looking at degree distributions, which show the fraction of nodes with a degree k (for all $k > 0$). A function on the number of triangles centered on a node is called the clustering coefficient. Clustering coefficient distributions show the proportion of nodes with clustering coefficients C . In a network, it determines the strength of the connectivity between a node and its nearest neighbors. The distance between both distributions of the statistical property for initial graph G and sampled network G_s is computed according to some distance functions such as *Kolmogorov–Smirnov Test (KSD)*, *Skew-Divergence distance (SDD)*, and *Normalized-Distance (ND)* (Ahmed et al. 2014b). In the remainder of this section, we describe these distance measures.

When two cumulative distribution functions (CDFs) are compared, *Kolmogorov's D statistic (KSD)* is used to calculate the distance between them. The *KSD* measures the agreement between the actual distribution F_1 and the estimated distribution F_2 . A value between 0 and 1 is returned by this test. Both distributions will be more similar as they get closer to zero; and as they get closer to one, they will be more different.

This measure has been defined by Eq. (1), where F_1 and F_2 denote the cumulative distribution function (CDF) of the original distribution and estimated distribution respectively. Also, x represents the random variable's range. The distance between two distributions is therefore calculated as the maximum vertical distance (James 2006).

$$KS(F_1, F_2) = \max_x |F_1(x) - F_2(x)| \quad (1)$$

A **Normalized Distance (ND)** is a measure of how far there is a difference between two positive m -dimensional vectors that are F_1 and F_2 where F_1 represents the original dimensional vector and F_2 represents the estimated dimensional vector. *ND* is defined by Eq. (2) (Rezvanian et al. 2019).

$$ND(F_1, F_2) = \frac{\|F_1 - F_2\|}{\|F_1\|} \quad (2)$$

Using the Skewed Divergence Distance (*SDD*) we can compare two probability density functions (PDF). The *SDD* is a *Kullback–Leibler (KL)* divergence between two PDFs F_1 and F_2 with no continuous support (e.g., skewed degree). *KL* measures the average number of bits needed to represent samples from the original distribution when using sampled distributions. Since *KL* divergence cannot be defined for distributions with different support areas, skew divergence smooths the two PDFs before computing the *KL* divergence

(Rezvanian et al. 2019). Equation (3) describes SD distance, in which the constant α lies between 0 and 1. The divergence of P from Q is calculated by KL.

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

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

Cost measures are also defined to assess the efficiency of sampling algorithms during the construction of a sampled network (Jalali et al. 2016). It is calculated using two numbers, C_t and C_p , and two cost units, c_1 and c_2 . A sampling network is constructed by traversing edges C_t times and each edge travel has a cost c_1 . The construction of a sampling network requires C_p number of pre- or post-operations. Let c_2 be the cost of each such process. As shown below (Eq. (5)), the cost per sample is calculated by dividing the weighted sum of these two costs by the total number of edges.

$$Cost = \frac{c_1 \cdot C_t + c_2 \cdot C_p}{|E|} \quad (5)$$

In Eq. (5), $|E|$ represents the number of edges of the original graph. In any pre- or post-processing step, an edge traversal cost is constant c_1 and a processing operation cost is constant c_2 .

3 Related work

In addition to simple measures, advanced sampling techniques can be used to uncover useful information in social network graphs. However, to study OSNs directly, graph sampling is needed to reduce the size of the graphs while keeping their important features. Network sampling techniques can be classified in different ways in the literature. For instance, they can be divided into vertex sampling, edge sampling, and topology-based sampling (Ahmed et al. 2014a). Vertex sampling randomly and independently selects graph nodes. Breadth First Search (BFS) and Metropolis Hasting Random Walk (MHRW) are two examples of vertex sampling algorithms. Edge sampling randomly and independently chooses some edges of a given network. Frontier sampling (FS) is a type of edge sampling algorithm. Topology-based sampling selects nodes and edges based on the graph's structure. Snowball Sampling (SS), Forest Fire Sampling (FFS), and Random Walk Sampling (RWS) (Lovász 1993) are some topology-based sampling algorithms (Kurant et al. 2011a).

Random sampling methods are introduced as a category of sampling algorithms in (Kurant et al. 2011b), where each vertex is visited only once (Leskovec and Faloutsos 2006).

Graph traversal sampling is also introduced in (Kurant et al. 2011b) as the other category of sampling algorithms. A graph traversal sampling technique, such as Random Walk Sampling (RWS), iteratively samples nodes adjacent to the last sampled node (Kurant et al. 2011b).

There are two other categories of sampling algorithms presented in (Kurant et al. 2011b): random vertex sampling and graph traversal sampling. In random vertex sampling, each vertex is visited only once and is not replaced during sampling (Leskovec and Faloutsos 2006). A graph traversal sampling technique, such as Random Walk Sampling (RWS), picks the next node iteratively according to some criteria among those adjacent to the last node sampled.

There are two types of sampling algorithms based on their processing methods: classical and modern. Classical sampling algorithms build a sampled network by choosing or visiting edges or nodes only once. Examples of these algorithms are BFS, RWS (Lovász 1993), FFS (Leskovec and Faloutsos 2006), and RDS (Gile and Handcock 2010). Modern sampling algorithms add an extra step of pre/post-processing that may involve selecting or visiting edges or nodes more than once. Some of these algorithms are DLAS (Rezvanian et al. 2014), SSP (Rezvanian and Meybodi 2015), RPN (Yoon et al. 2015), and DPL (Yoon et al. 2015). Classical sampling algorithms are fast and simple, but they have different results for each run and may not suit different network settings. Modern sampling algorithms are more accurate and adaptable but require more processing. They use extra information about the network structure, such as some measures (Peng et al. 2014), network properties (Blagus et al. 2015), or community structures (Yoon et al. 2015). This trade-off between processing and accuracy is a key challenge for modern sampling methods. It is possible to use the sampled network for a variety of applications and analyses once it is obtained. The main goal of designing a sampling algorithm is to create a sampled network that preserves the characteristics of the original network, rather than minimizing time complexity.

Leskovec et al. were among the first to study network sampling (Leskovec and Faloutsos 2006). They proposed several sampling algorithms to find a sampled graph that is similar to the original graph or a previous version of it. A comparison was also made between several algorithms for graph sampling. This comparison shows that FFS performed better than the others. As a web-based sampling algorithm, RDS was evaluated by Wejnert et al. for its effectiveness and feasibility (Wejnert and Heckathorn 2008). The analysis of Facebook users was conducted by Gjoka et al. through the implementation and comparison of several crawling algorithms. They showed that BFS and RWS are biased, while MHRW and RWRW perform very well (Gjoka et al. 2010). Kurant et al. suggested a method to correct sampling bias (Kurant et al. 2011b). Based on their study, the

BFS overestimated, and the RWS underestimated the graph degree in the absence of the graph degree. Ribeiro et al. analyzed the steady state behavior of CTRWs on Markov dynamic networks and compared two types of rates in walking: a constant rate (CTRW-C) and a vertex degree-related rate (CTRW-D) (Ribeiro et al. 2011).

Son et al. (2012) compared BFS and RS analytically and studied their sampling biases. Siciliano et al. (2012) suggested an adaptive threshold algorithm for network research that can deal with sampling biases. Ribeiro et al. (2012) introduced a modified random walk sampling algorithm for directed graphs that can make random jumps. Pappelis et al. (2013) proposed an algorithm that can quickly collect information from the user's neighborhood in a dynamic way without visiting all the nearby nodes. They only showed the benefits of their algorithm through experiments. Pina-Garcia et al. changed the way MHRW performed by using spirals rather than normal distributions as a probabilistic distribution (Piña-García and Gu 2013). They demonstrated that their algorithm outperformed normal MHRW in illusion spiral cases. In addition, they confirmed Gjoka et al.'s (2010) estimations. In Wang et al.'s sampling algorithm, all neighbors are considered valid samples (Wang and Lu 2013). The average degree of this algorithm improves random walk sampling.

Rezvanian et al. (2014) suggested an algorithm based on distributed learning automata (DLAS). This algorithm cooperates to sample a network. Based on KS tests, the algorithm outperformed RDS and RWS. They also achieved better results with low sampling rates. Although it relied on nodes of high degree, it still had some biases. Rezvanian and Meybodi (2015) also proposed a sampling method using the shortest path concept and demonstrated its effectiveness by simulations on well-known data sets. Peng et al. (2014) presented two sampling methods: one that improved the stratified random sampling method by selecting high-degree nodes with higher probability based on their degree distribution, and another that improved the snowball sampling method by selectively sampling the targeted nodes. Gao et al. developed a sampling method that used multiple random snowballs and Cohen's process based on snowball sampling (Gao et al. 2014). They simulated their method on model networks and verified that it preserved the network's local and global structures.

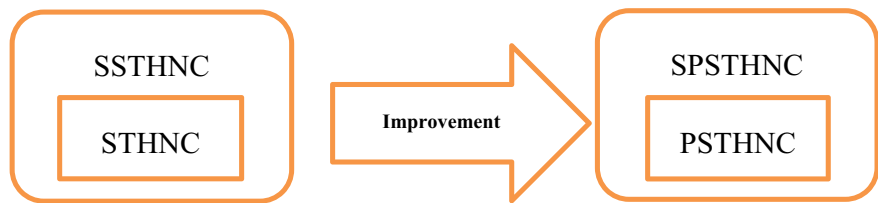
Jaouadi and Romdhane (2021) presented a parameter-less model called Distributed Graph Sampling (DGS) for sampling large-scale social networks to reduce the graph's size. A key feature of their proposal is its distributed strategy based on the MapReduce paradigm to sample a large graph in a parallel way to reduce the size of the network and conserve its key properties. In Ebadi Jokandan et al. (2021), proposed an optimization approach to efficiently sample large-scale social networks using a combination of

cuckoo search (CS) and genetic algorithms (GA). In Liu et al. (2022), two concepts of node neighborhood clustering coefficient and random walk are combined to introduce a hierarchical sampling algorithm to efficiently sample nodes at different levels of the network hierarchy. Roohollahi et al. (2022), introduced an improved version of a learning automata-based approach to sampling in weighted social networks using a Levy flight-based learning automata. The authors combined Levy flights as stochastic search processes with learning automata to adapt and learn from their network environment. In Luo et al. (2024), the authors introduced a novel approach for sampling hypergraphs using joint unbiased random walks. They proposed a new random walk algorithm that efficiently explores the hypergraph structure and generates statistically unbiased samples. Their method is based on a combination of node sampling and edge sampling, which allows for more efficient and accurate sampling of hypergraphs compared to existing methods.

To name some evidence for the importance of the concept of spanning trees in network analysis, we briefly review some of these methods in the rest of this section. Spanning trees are a way of connecting vertices by a single path (White and Newman 2001) and provide the basis for any chaining algorithm (Hill 1999). Spanning trees can also reveal important information from graphs, such as the structure of romantic relationships in a social network of 800 adolescents over 18 months, as studied by Bearman et al. (2004). They showed that every romantic relationship exists in a large spanning tree. This can capture the long and extensive chains of communication between large groups of people.

A study by Ansari et al. (2004) shows that a random spanning tree can preserve the topological and modular features of a network and that the network's structure can be easily visualized from the properties of its spanning tree. Spanning trees are crucial for solving propagation problems in networks, as shown by previous studies (Bellur and Ogier 1999). Spanning trees can efficiently and reliably propagate edge states. Therefore, spanning trees can propagate complete information across a network, making them effective structures for information transmission. Tewarie et al. (2015) used a minimum-spanning tree (MST) to unbiasedly characterize a network of complex brain connections. MST is a subgraph that connects all its nodes without cycles. It can maximize a property of interest, such as brain area synchronization, that preserves both global and local properties. Jalali et al. (2016) discuss a novel method for sampling social network data using spanning trees. The authors propose an algorithm that efficiently constructs a spanning tree from a social network graph, which can be used to sample the network while maintaining important structural features. In a recent study by Blomsma et al. (2022) with the aid of the concept of a spanning tree, the authors examine the effects of network size on the analysis, as well as the

Fig. 1 Relations among proposed methods



STHNC: Spanning Tree with Highest Number of nodes that satisfy a desired degree Constraint
 SSTHNC: Social Network Sampling using a set of STHNC.
 PSTHNC: a Partial version of STHNC.
 SPSTHNC: Social Network sampling using a set of PSTHNC.

sensitivity of this approach for detecting neuropsychiatric pathology. They also explore the specificity of disorder-related network alterations. The review highlights the potential of minimum spanning tree analysis as a valuable tool for understanding the organization of brain networks in health and disease.

4 Proposed sampling algorithm

As a sampling algorithm, the proposed method uses spanning trees with degree constraints to highlight some important edges. In this work, we used a set of degree constraint-spanning trees with a variety of starting points as a basis for the algorithm. It was concluded from the results that edges with a large presence in spanning trees are important edges that capture several characteristics of the original network.

Figure 1 shows the relation among proposed methods. The remainder of this section describes in detail what STHNC and SSTHNC are. We then briefly describe SPSTHNC and SPSTHNC.

4.1 STHNC: a spanning tree with a degree constraint

An example of the degree-constrained spanning tree problem is determining whether a given network has a particular spanning tree for a given c (Lee 2001). It is an NP-complete problem (Garey and Johnson 1979), which is shown by reducing it to the Hamiltonian path problem. This section proposes an algorithm that generates a degree-constrained spanning tree if it can find one, and if not, an approximation by minimizing the number of nodes that violate the given degree constraint. The basic idea of this algorithm is borrowed from (Sundar et al. 2012).

The proposed algorithm, STHNC, starts with a randomly initial vertex and iteratively tries to find a spanning tree with the highest number of nodes satisfying a given degree constraint. Consider the initial vertex v randomly selected in the first step of STHNC. The vertex v is added to the set of spanning tree nodes T . In addition, v is marked as visited. The algorithm forms an edge set E' containing edges of the resulting spanning tree. The algorithm runs through several iterations until T contains all the vertices of the input graph (i.e. V). Each iteration consists of two parts, described below:

Part I: A vertex v is chosen from T as the current node. The selection of the current node from set T is performed as follows:

- If set T has only the initial vertex, then this node is considered the current node v .
- If the set T has more than one node, then the algorithm considers the degree of nodes in the spanning tree (i.e., $deg(v, S)$):
 - o Initially, such vertices with degrees below c (c is the degree constraint) are considered in T ($deg(v, S) < c$). From them, node v with the least number of unvisited neighbors is chosen as the current node.
 - o Else, if there is a node that violates the constraint and also has unvisited neighbors then it is selected as the current node v .
 - o Otherwise, a vertex with a maximum number of unvisited neighbors is selected as the current node v .

Part II. Among unvisited neighbors of node v , vertex u with the least unvisited neighbors is chosen if such a vertex exists. Whenever such a vertex does not exist, another iteration occurs. Then edge (v, u) is added to set E' and vertex v is

Algorithm 1: Create a spanning tree STHNC

Input: Graph $G=(V, E)$ Degree constraint c
Output: degree constraint spanning tree $S=(T, E')$

Begin
 $T = \{\}$
 Select a node $v \in V$ randomly as the initial vertex of the spanning tree
 Add node v to T ;
While ($|T| < |V|$) **do**
 If (there exists a node $v \in T$ such that $deg(v, S) < c$ and v has unvisited neighbors in G)
 Select node v with the minimum number of unvisited neighboring nodes;
 Else If (there exists a node v with a degree in S higher than c and has unvisited neighbors)
 Select node v ;
 Else
 Select node $v \in T$ with the maximum number of unvisited neighboring node
 End if
 While (the set of unvisited neighbors of $v \in G$ is not empty) **do**
 Find node u from unvisited neighbors of v in G with the minimum number of unvisited neighbors;
 Mark node u as visited;
 Add edge (v, u) to E' ;
 Add node u to T ;
 $v \leftarrow u$
 End While
End while
End algorithm

Fig. 2 The pseudo-code of the spanning tree with the highest number of nodes that satisfy the given degree constraint

marked as visited and added to set T . The Current node is set to vertex u ($v \leftarrow u$) and the algorithm goes to part II.

The pseudo-code of the algorithm is shown in Fig. 2.

In the proposed algorithm **STHNC**, selecting the current node plays a significant role in obtaining a spanning tree with the highest number of nodes satisfying the degree constraint C . The proposed algorithm selects a current node with the minimum number of unselected neighbors, because the greater the number of unselected neighbors, the higher the probability of the vertex being selected in the future. Using this selection strategy, the spanning tree can be satisfied with the given constraint.

However, in some cases, the algorithm encounters a situation that has to allow a vertex to break satisfying the given constraint. To reduce the number of such vertices, a vertex with a maximum number of unselected neighboring vertices is selected as the current vertex. Until this vertex has any unselected neighbors, the algorithm will not allow other nodes to violate the given constraint. As a result of this selection, the number of vertices with degrees higher than the constraint is reduced.

4.2 SSTHNC: social network sampling using a set of STHNCs

SSTHNC starts by constructing k different spanning trees with the highest number of vertices that satisfy the given degree constraint (i.e., k different STHNCs). Each STHNC starts with a random vertex. Based on the constructed STHNCs, each edge present in the set of STHNCs is ranked. Edges with more appearances in STHNCs have a higher rank.

Additionally, each vertex appearing in STHNCs is ranked based on how many times it violated the degree constraint. Afterward, the top- m vertices of the list of nodes sorted in decreasing order by rank are selected, where m represents the minimum number of degree-constraint violations in any STHNC constructed. As a result, these m vertices are referred to as hubs.

SSTHNC constructs an induced subgraph¹ as the sampled network. A new sample vertex is inserted into the sampled graph, along with all edges between it and older samples, if they do not already exist.

¹ A subgraph H is an induced subgraph of G if for any two vertices uv in H , u and v are adjacent in H if and only if they are adjacent in G .

Algorithm 2: Sampling with STHNCs (SSTHNC)**Input:** Graph $G=(V, E)$, Sampling rate ϕ **Output:** Sampled graph $G_s=(V_s, E_s)$ **Begin**Call Algorithm 1, k times to construct a set of STHNCsFind the edges in STHNCs. List edges in descending order of appearances in STHNCs. The sorted list is L .Check hub nodes in STHNCs (i.e. nodes that violate the degree constraint). Based on the number of violations, rank the hubs descending. The ranking list is N .Find the minimum number of hub nodes in the STHNC. This number is m . $V_s = \{\}, E_s = \{\}$ Select the m -top hub node from N and insert it into V_s .**while** (L is not empty and $|V_s| < \mu * |V|$) Let e be the edge between two nodes in V_s that do not belong to the edge set E_s . Insert e into E_s and remove it from list L . Find the first edge in L , edge e , that connects node u in V_s to node u in $V - V_s$. Add e into E_s and y into V_s . Remove it from list L .**End while****End algorithm****Fig. 3** The pseudocode of the proposed sampling graph with a set of STHNC (SSTHNC)

SSTHNC first inserts hub nodes into the sampled network. All edges between hubs are inserted into the sampled network due to the induced property of the subgraph. It is inserted sequentially the (distinct) vertices incident on the edges selected from the beginning of the list of ordered edges as long as the number of nodes in the sampled network is not greater than x fraction of the original nodes. When a vertex is added to the sampled network, its edges to previous vertices are established every time a vertex is added. Figure 3 shows the pseudocode for the proposed sampling algorithm SSTHNC.

4.3 SPSTHNC: social network sampling using partial STHNC

SSTHNC can be enhanced by an α -partial version of STHNC (i.e. PSTHNC) with the highest number of nodes that meet a desired degree constraint. A PSTHNC with starting vertex v is a subgraph of graph G and is a tree with starting vertex v with the number of nodes $\alpha * |V|$ for $0 < \alpha < 1$. α -PSTHNC works like STHNC, as shown in Fig. 4.

Algorithm PSTHNC computes an α -partial spanning tree for a given α . The revised SSTHNC version, called SPSTHNC, uses PSTHNC instead of STHNC. This includes all the vertices of the input graph.

4.4 Some remarks on the proposed methods

- By using degree constraint spanning trees, edges could be visited uniformly. This prevents the sampled graph from being biased to high-degree nodes and prevents variance from having large values. There is no doubt that when the algorithm uniformly samples the input graph, some nodes with high degrees will inevitably be sampled. Several connections would be lost if these nodes were unavailable. By considering these influential nodes, we can better preserve the network statistics.
- By considering a spanning tree with a minimum number of hub nodes (influential nodes), the sampled network is prevented from being biased towards influential nodes in the network.
- In a variety of fields, scale-free networks, such as online social networks, are dominated by a few nodes with many neighbors (Santos and Pacheco 2005). Also, according to the 20/80 rule, a small number of nodes typically contribute a large amount to a network's activities (Woolhouse et al. 1997). It seems that it is more convenient to consider both nodes and significant edges when looking at network samples. This is to obtain better samples.

Algorithm 3: Create a α -partial spanning tree PSTHNC**Input:** Graph $G=(V, E)$, Degree constraint c **Output:** degree constraint spanning tree $S=(T, E')$ **Begin** $T = \{\}$ Select a node $v \in V$ randomly as the initial vertex of the spanning treeAdd node v to T ;**While** ($|T| < \alpha * |V|$) **do****If** (there exists a node $v \in T$ such that $deg(v, S) < c$ and v has unvisited neighbors in G) Select node v with the minimum number of unvisited neighboring nodes;**Else If** (there exists a node v with a degree in S higher than c and has unvisited neighbors) Select node v ;**Else** Select node $v \in T$ with the maximum number of unvisited neighboring node**End if****While** (the set of unvisited neighbors of $v \in G$ is not empty) **do** Find node u from unvisited neighbors of v in G with the minimum number of unvisited neighbors; Mark node u as visited; Add edge (v, u) to E' ; Add node u to T ; $v \leftarrow u$ **End While****End while****End algorithm****Fig. 4** The pseudo-code of constructing an α -partial degree constraint spanning tree with the highest number of nodes that satisfy the given degree constraint

4.5 The time complexity of the proposed algorithms

In this section, we try to analyze the time complexity of the proposed algorithms. To find the time complexity of the proposed algorithms, we need to count the number of operations that the algorithm performs in the worst case and express it as a function of the input graph size.

4.5.1 The time complexity of STHNC

The STHNC consists of two nested while loops. The outer loop iterates until all the nodes are added to the spanning tree T , which is at most $|V|$ times. The inner loop iterates until all the unvisited neighbors of the current node v are visited, which is at most $|E|$ times in total for all the nodes. Therefore, the total number of iterations of the inner loop is $O(|E|)$. Inside the inner loop, the algorithm performs some constant time operations, such as finding the node u with the minimum number of unvisited neighbors, marking u as visited, adding the edge (v, u) to E' , adding u to

T , and updating v . These operations take $O(1)$ time each. Therefore, the total time complexity of the algorithm is $O(|V| \times |E| \times 1) = O(|V| \times |E|)$.

4.5.2 The time complexity of SSTHNC

SSTHNC consists of several steps. The first step is to call STHNC, k times. The time complexity of STHNC is $O(k \times |V| \times |E|)$. The second step is to find the edges in STHNCs and list them in descending order of appearance. This can be done by using a hash map and a sorting algorithm. The hash map can store the edges and their frequencies in $O(|E|)$ time and space. The sorting algorithm can sort the edges in $O(|E| \log |E|)$ time. Therefore, the second step takes $O(|E| + |E| \log |E|) = O(|E| \log |E|)$ time. The third step is to check the hub nodes in STHNCs and rank them based on the number of violations. This can be done by using a hash map and a sorting algorithm. Same the second step, this step takes $O(|V| \log |V|)$ time. The fourth step is to find the minimum number of hub nodes in the STHNC. This can be done by using a linear search on the sorted list of nodes. This

Table 1 Description of test networks

Name	Network	Node	Edge	Description
D1	Squeak Foundation.org (Leskovec et al. 2005)	1133	5451	A sample network of mailing lists for answers in Squeak Foundation.org
D2	Robots.net (Leskovec et al. 2005)	1706	3561	A sample network from robots.net social network
D3	Cit-HepPh (Leskovec et al. 2007)	34,546	421,578	A sample network from the citation network of Physics Theory
D4	Epinions1 (Leskovec et al. 2005)	75,879	508,837	A sample network of consumer reviews from Epinions.com
D5	Slashdot0902 (Leskovec et al. 2009)	82,168	948,464	A sample network of Slashdot Zoo social network
D6	Email-EuAll (Leskovec et al. 2009)	265,214	420,045	A sample network from emails of a large European research institution

takes $O(|V|)$ time. The fifth step is to select the $|E|$ -top hub nodes from the list and insert them into V_s . This takes $O(|E|)$ time. The sixth step is the main loop of the algorithm, which iterates until L is empty or V_s reaches the desired size. The loop body consists of some constant time operations, such as inserting and removing edges, finding the first edge that connects two nodes, and adding nodes to V_s . These operations take $O(1)$ time each. Therefore, the loop body takes $O(1)$ time. The loop condition depends on the size of L and V_s . In the worst case, L has m edges and V_s has $\varphi \times |V|$ nodes. Therefore, the loop condition takes $O(|E| + \varphi \times |V|)$ time to check. Therefore, the total time complexity of the algorithm is $O(k \times |V| \times |E| + |E| \log |E| + |V| \log |V| + |V| + |E| + (|E| + \varphi \times |V|) \times 1) = O(k \times |V| \times |E| + |E| \log |E| + |V| \log |V| + \varphi \times |V|) = O(|V| \times |E|)$.

5 Simulation results

This section compares the proposed algorithms with other popular sampling algorithms on six graphs drawn and described in Table 1 including *Squeak Foundation.org* (Leskovec et al. 2005) as D1, *Robots.net* (Leskovec et al. 2005) as D2, *Cit-HepPh* (Leskovec et al. 2007), as D3, *Epinions1* (Leskovec et al. 2005) as D4, *Slashdot0902* (Leskovec et al. 2009) as D5, *Email-EuAll* (Leskovec et al. 2009) as D6. In addition, artificial graphs are also used in some experiments. Other sampling methods include Random Node Sampling (RVS) (Maiya and Berger-Wolf 2011), Random Edge Sampling (RES) (Ahmed et al. 2010), Random Walk Sampling (RWS) (Lovász 1993), Metropolis–Hastings Random Walk Sampling (MHRW) (Murai et al. 2013) and Spiral Sampling (SS) (Piña-García and Gu 2013), distributed learning automation sampling (DLAS) (Rezvanian et al. 2014) the shortest path sampling (SPS) (Rezvanian and Meybodi 2015), sampling by spanning trees (SST) (Jalali et al. 2016), Hybrid cuckoo search and genetic algorithm sampling (GACS) (Ebadi Jokandan et al. 2021), and

Levy flight–Based Distributed Learning Automata sampling (LBDLA) (Roohollahi et al. 2022).

5.1 Experiment I

This experiment is designed to study the performance of the first proposed algorithm (SSTHNC) for finding degree constraint spanning trees with the highest number of nodes that satisfy the given degree constraint on some synthetic random connected small graphs based on the Erdos–Rényi model (Erdos and Rényi 1960). The size of these networks varies from 20 to 100 nodes. The proposed algorithms are compared with brute force search, which constructs all spanning trees on a synthesis network. Among all these spanning trees, such a tree with a minimum degree would be considered the most suitable solution. In this spanning tree, the number of nodes that violate the degree constraint c is denoted as the real value (reported in Table 2). In contrast, a spanning tree can be constructed by running the proposed algorithm considering a predefined degree constraint c . This will reveal the number of nodes that do not satisfy the degree constraint. Using the proposed algorithm for multiple runs, an average number can be calculated, which is also reported in Table 2.

Besides, the SSTHNC algorithm is conducted on the chosen real networks as described in Table 1. The results of the number of vertices that do not satisfy constraints and run-time are reported in Table 3. In this experiment degree constraint spanning tree is executed with different degree constraints varying from 2 to 5. For each experiment, the algorithm is executed 30 times, and average results are reported in Table 3. The results are shown in terms of the number of nodes with degrees higher than the given constraint and times in seconds needed to construct a degree constraint spanning tree. From the results, we may conclude that by increasing the constraint, the number of vertices that violate the constraints (also called branch vertices) will decrease and also using the algorithm in a reasonable time, it can have a degree constraint spanning

Table 2 Comparison of results for number of vertices that do not satisfy constraints

Degree constraint c	Size of networks									
	20		40		60		80		100	
	Real	Average	Real	Average	Real	Average	Real	Average	Real	Average
2	2	2.00	3	3.8	3	3.11	4	4.89	7	9.70
3	1	1.32	2	2.35	3	3.12	4	5.12	7	9.54
4	1	1.43	2	2.23	2	2.72	3	3.78	6	8.30
5	1	1.45	2	2.27	2	2.26	3	3.62	5	6.70

Table 3 Comparison of results for the number of vertices that do not satisfy constraints and run-time for varying degree constraints

Degree constraint c	Networks											
	D1		D2		D3		D4		D5		D6	
	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time	Value	Time
2	7.30	0.01	14.03	0.02	21.23	0.08	91.46	6.86	8.40	5.57	8.30	16.39
3	7.73	0.01	6.70	0.02	13.56	0.19	58.80	15.67	7.86	41.00	7.50	49.88
4	4.90	0.01	5.93	0.01	12.00	0.18	57.70	13.22	7.83	42.42	7.63	41.88
5	5.56	0.01	4.36	0.01	12.46	0.23	65.80	31.41	7.90	42.00	7.90	45.85

trees with the highest number of nodes that satisfy the given degree constraint.

5.2 Experiment II

This experiment studies the minimum degree of constraint on spanning trees required to meet a given sampling rate. In the first step, the first algorithm (SSTHNC) makes some BKDSTs. SKBDST is executed with different degree constraint values from 2 to 10 to examine minimum degree constraint c for spanning trees. Results of this experiment are demonstrated for chosen networks as described in Table 1. with respect to KSD for DD, KSD for CCD, SDD for DD, SDD for CCD, ND for DD, and ND for CCD in Fig. 5. From Fig. 5, one can conclude that for a sampling rate of 30%, depending on the type of the network at least a degree constraint $k = 5$ is required to meet a specific sampling rate. The results of this experiment will be used in other experiments to choose the right number of spanning trees.

5.3 Experiment III

In this experiment, the minimal number of spanning trees that must be used to reach a given sampling rate is determined. In this case, we ran the first proposed algorithm, SSTHNC, for a sampling rate of 30% on different test networks as described in Table 1 and then plotted the number of spanning trees versus KSD for DD, KSD for CCD, SDD for DD, SDD for CCD, ND for DD and ND for CCD as given in Fig. 6. As a result of these findings, we can state that

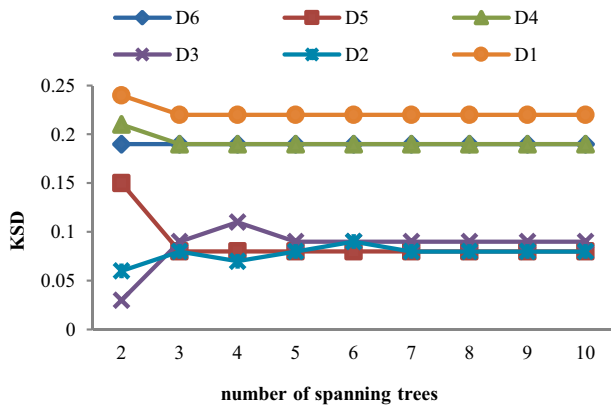
increasing the number of computed spanning trees beyond 50 will not impact any of the statistical properties discussed above. Therefore, for the following experiments, we will set the number of computed spanning trees used to construct a sample to 50 to obtain a representative sample.

5.4 Experiment IV

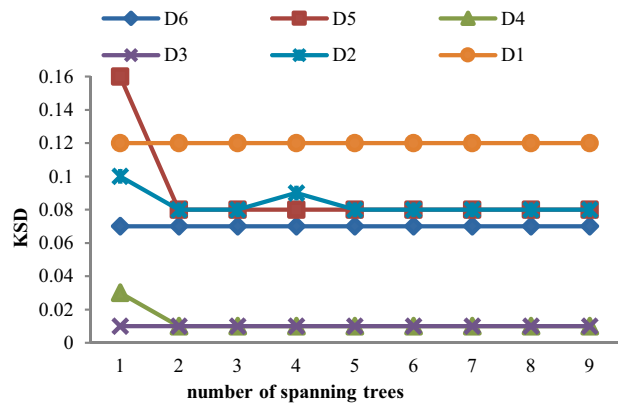
This experiment examines the impact of selecting starting vertices on spanning trees based on different strategies. To achieve this, three strategies for selecting vertex have been examined: Rd , Hd , and Hb . Rd is a random starting vertex strategy. In Hd strategy, vertices with high degrees are used as hub vertices. For Hb strategy, high betweenness vertices are used. A node's betweenness is measured as the percentage of times it is located on the shortest path among all other nodes. The transfer of information items through a network is greatly affected by nodes with high betweenness. Results of this experiment for chosen test networks as described in Table 1 with respect to KSD for DD, KSD for CCD, SDD for DD, SDD for CCD, ND for DD, and ND for CCD are given in Table 4. We can conclude from the results that it does not matter how the starting vertices are chosen for degree constraint spanning trees.

5.5 Experiment V

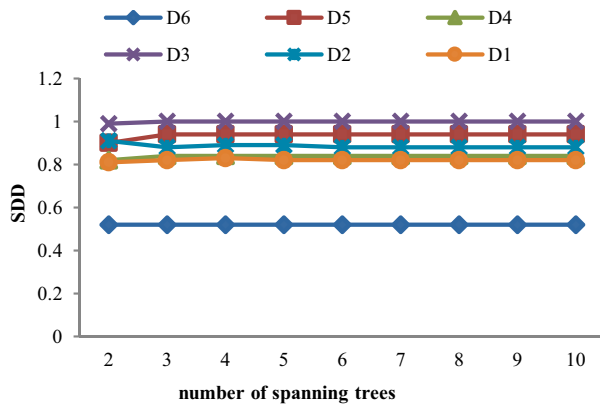
This experiment compares two proposed algorithms SSTHNC (the first) and SPTHNC (the second) with Random Node Sampling (RVS) (Maiya and Berger-Wolf



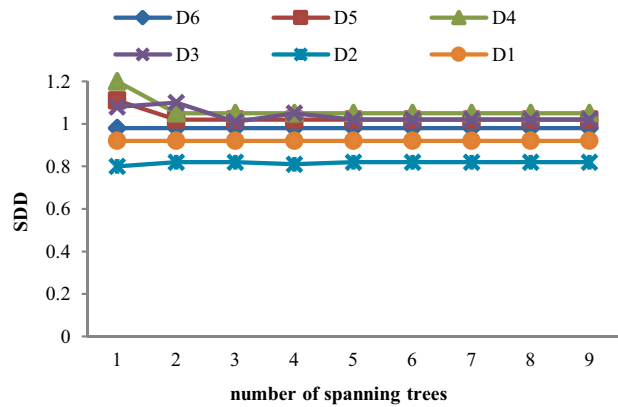
a) KSD for *DD* by SSTHNC with different numbers of spanning trees



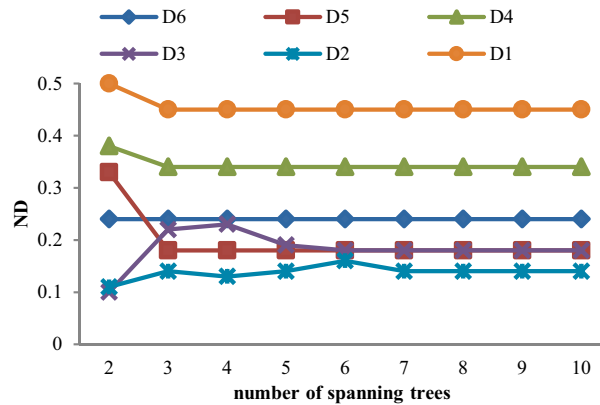
b) KSD for *CCD* by SSTHNC with different numbers of spanning trees



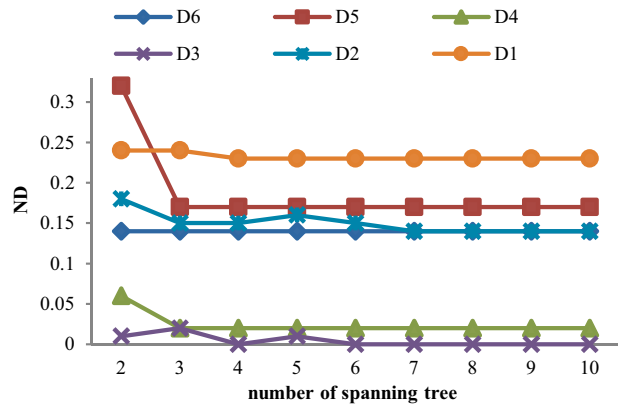
d) SDD for *DD* by SSTHNC with different numbers of spanning trees



e) SDD for *CCD* by SSTHNC with different numbers of spanning trees

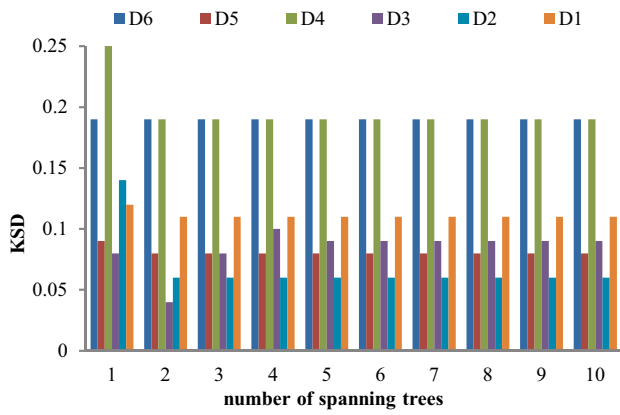


f) ND for *DD* by SSTHNC with different numbers of spanning trees

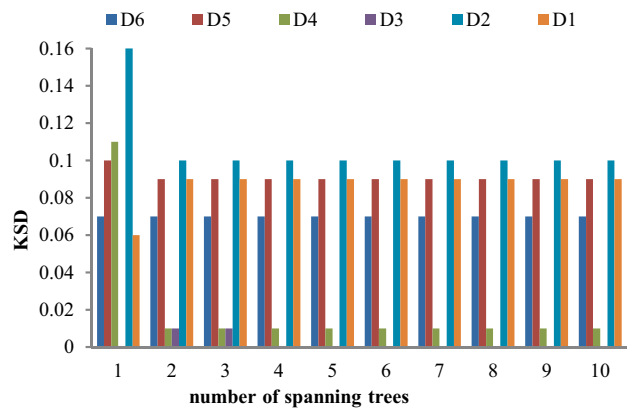


g) ND for *CCD* by SSTHNC with different numbers of spanning trees

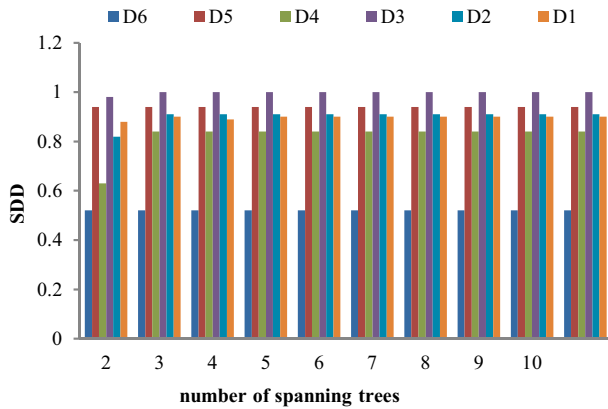
Fig. 5 The minimum degree of constraint on spanning trees required to meet a given sampling rate



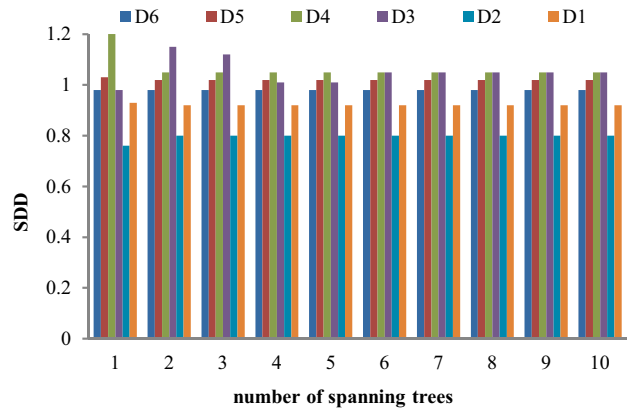
a) KSD for *DD* by SSTHNC with different numbers of spanning trees



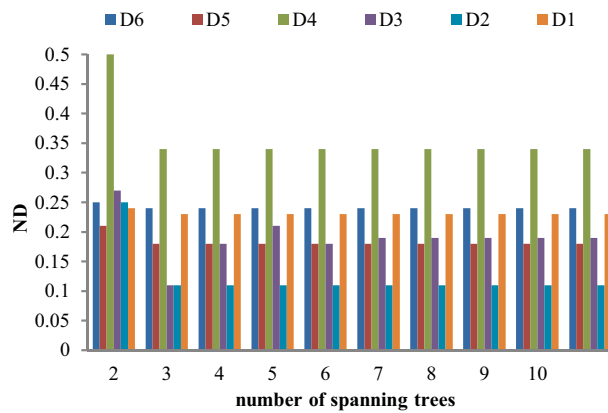
b) KSD for *CCD* by SSTHNC with different numbers of spanning trees



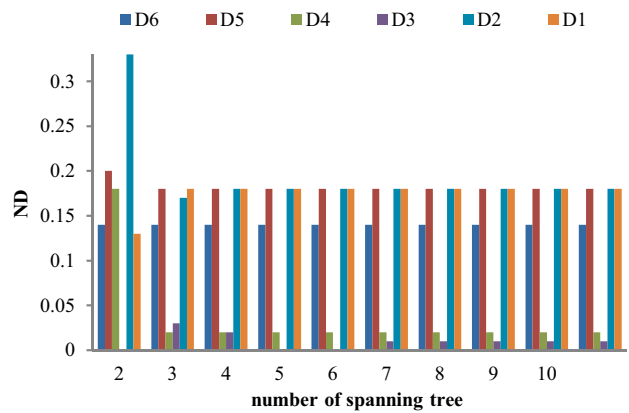
c) SDD for *DD* by SSTHNC with different numbers of spanning trees



d) SDD for *CCD* by SSTHNC with different numbers of spanning trees



e) ND for *DD* by SSTHNC with different numbers of spanning trees



f) ND for *CCD* by SSTHNC with different numbers of spanning trees

Fig. 6 A given sampling rate requires a certain number of spanning trees

Table 4 Results of the proposed algorithm for different starting vertex selection strategies: Rd, Hd, and Hb

Distance function	Statistical property	Networks																	
		D1			D2			D3			D4			D5			D6		
		Rd	Hd	Hb	Rd	Hd	Hb	Rd	Hd	Hb	Rd	Hd	Hb	Rd	Hd	Hb	Rd	Hd	Hb
KSS	DD	0.13	0.13	0.13	0.03	0.03	0.03	0.09	0.09	0.09	0.19	0.19	0.19	0.05	0.05	0.05	0.19	0.19	0.19
	CCD	0.17	0.17	0.17	0.18	0.18	0.18	0.00	0.00	0.00	0.01	0.01	0.01	0.03	0.03	0.03	0.07	0.07	0.07
SDD	DD	0.89	0.89	0.89	1.03	1.03	1.03	1.00	1.00	1.00	0.84	0.84	0.84	0.97	0.97	0.97	0.52	0.52	0.52
	CCD	0.94	0.94	0.94	0.77	0.77	0.77	1.03	1.03	1.03	1.06	1.06	1.06	1.00	1.00	1.00	0.98	0.98	0.98
ND	DD	0.30	0.30	0.30	0.06	0.06	0.06	0.19	0.19	0.19	0.34	0.34	0.34	0.10	0.10	0.10	0.24	0.24	0.24
	CCD	0.23	0.23	0.23	0.32	0.32	0.32	0.01	0.01	0.01	0.02	0.02	0.02	0.06	0.06	0.06	0.14	0.14	0.14

2011), Random Edge sampling (RES) (Ahmed et al. 2010), Random Walk sampling (RWS) (Lovász 1993), Metropolis–Hastings Random Walk sampling (MHRW) (Murai et al. 2013), Spiral Sampling (SS) (Piña-García and Gu 2013), distributed learning automation sampling (DLAS) (Rezvanian et al. 2014) the shortest path sampling (SPS) (Rezvanian and Meybodi 2015), sampling by spanning trees (SST) (Jalali et al. 2016), Hybrid cuckoo search and genetic algorithm sampling (GACS) (Ebadi Jokandan et al. 2021), and Levy flight-Based Distributed Learning Automata sampling (LBDLA) (Roohollahi et al. 2022) based on KSD for DD, KSD for CCD, SDD for DD, SDD for CCD, ND for DD, and ND for CCD. The sampling rate is 15%. The results reported are averaged over 30 runs. For statistical significance, a parametric t-test with 28 degrees of freedom was conducted at a 95% level of significance. The t-test result on SSTHNC is reported as better than (shown by “√”), worse than (shown by “×”), similar to (shown by “~”) that of the corresponding sampling method. This experiment's results are shown in Tables 5, 6, 7, 8, 9 and Table 10. As the results of the t-test, the boldfaced results indicate the best results in Tables 5, 6, 7, 8, 9 and 10 and also we show the percentages of the cases that the SSTHNC outperforms significantly other algorithms in the last column. Based on these results, our proposed sampling algorithm outperforms sampling algorithms based on KSD, SDD, and ND. As well as outperforming the second sampling algorithm, the first is more efficient.

5.6 Experiment VI

This experiment studies the impact of sampling rate on the performance of the first proposed sampling algorithm. This goal is achieved by running the algorithm for changes in sampling rate from 5 to 30% with an increment of 5% on different test networks. The results are given in Fig. 7. According to the results, we may say that the impact of the sampling rate on the algorithm's performance is very

much dependent on the type of network. Generally, however, increasing the sampling rate leads to better results in most networks.

5.7 Experiment VII

A comparison of SSTHNC and PSTHNC is presented along with a few existing two-phase sampling algorithms, including the shortest path sampling (SPS) (Rezvanian and Meybodi 2015), distributed learning automation sampling (DLAS) (Rezvanian et al. 2014), sampling by spanning trees (SST) (Jalali et al. 2016), and Levy flight-Based Distributed Learning Automata sampling (LBDLA) (Roohollahi et al. 2022). For this experiment, the sampling rate is 30%. As shortest paths or spanning trees increase, algorithms cost more. In this experiment, the accuracy of sampling algorithms is compared based on KSD for DD and KSD for CCD versus sampling cost (as defined by Eq. (5)). In Figs. 8 and 9, we present the cost results of this experiment using KSDs for DD and CCD for different two-phase sampling algorithms. As shown, increasing sampling costs result in decreasing KSDs for DD and CCD. Results say that at the same cost, the PSTHNC algorithm generally outperforms other two-phase sampling algorithms at least in five of the six networks based on KSD for DD and four of the six networks based on KSD for CCD. For some networks such as D3, SPS sometimes outperforms SSTHNC based on KSD for CCD. For the D1 network, SSTHNC and PSTHNC produce similar results for the same KSD.

6 Conclusion

This paper proposed two sampling algorithms for social networks using degree constraint spanning trees. The idea behind these algorithms was that by building STHNCs, they could identify important nodes (those that exceed the degree limit) and edges (those that appear more often in

Table 5 Comparison of different algorithms for sampling in *D1*

Distance function	Algorithms for Sampling													
	RVS	RWS	RES	MHRW	SS	DLAS	LBDLA	GACS	SPS	SST	SPTHNC	SSTHNC		
KSD	DD	0.18 ± 0.01	0.19 ± 0.03	0.42 ± 0.01	0.35 ± 0.09	0.36 ± 0.11	0.16 ± 0.01	0.15 ± 0.00	0.18 ± 0.00	0.14 ± 0.01	0.16 ± 0.00	0.14 ± 0.01	0.13 ± 0.00	100%
		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCD	DD	0.19 ± 0.05	0.13 ± 0.03	0.15 ± 0.02	0.19 ± 0.07	0.23 ± 0.06	0.17 ± 0.01	0.16 ± 0.00	0.15 ± 0.00	0.14 ± 0.01	0.02 ± 0.01	0.08 ± 0.03	0.17 ± 0.00	30%
		✓	✗	✗	✓	~	~	✗	✗	✗	✗	✓	✓	✓
SDD	DD	0.86 ± 0.03	0.84 ± 0.01	0.71 ± 0.00	0.74 ± 0.05	0.74 ± 0.06	1.43 ± 0.09	1.33 ± 0.00	1.29 ± 0.00	1.31 ± 0.04	1.23 ± 0.01	0.87 ± 0.03	0.89 ± 0.00	50%
		✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCD	DD	0.94 ± 0.02	0.93 ± 0.01	0.93 ± 0.00	0.99 ± 0.08	1.02 ± 0.08	0.92 ± 0.01	0.91 ± 0.00	0.94 ± 0.00	0.91 ± 0.01	0.97 ± 0.02	0.93 ± 0.01	0.94 ± 0.00	50%
		~	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✓
ND	DD	0.36 ± 0.06	0.38 ± 0.07	0.73 ± 0.02	0.27 ± 0.14	0.64 ± 0.18	0.38 ± 0.01	0.34 ± 0.00	0.37 ± 0.00	0.37 ± 0.01	0.34 ± 0.01	0.30 ± 0.02	0.30 ± 0.01	90%
		✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCD	DD	0.38 ± 0.00	0.24 ± 0.00	0.29 ± 0.00	0.51 ± 0.00	0.45 ± 0.00	0.27 ± 0.01	0.31 ± 0.00	0.24 ± 0.00	0.25 ± 0.01	0.04 ± 0.00	0.14 ± 0.01	0.23 ± 0.00	90%
		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗

Bold indicates the best results

Table 6 Comparison of different algorithms for sampling in *D2*

Distance function	Algorithms for Sampling													
	RVS	RWS	RES	MHRW	SS	DLAS	LBDLA	GACS	SPS	SST	SPTHNC	SSTHNC		
KSD	DD	0.60 ± 0.01	0.06 ± 0.02	0.33 ± 0.02	0.19 ± 0.07	0.15 ± 0.08	0.32 ± 0.08	0.22 ± 0.07	0.09 ± 0.11	0.29 ± 0.01	0.19 ± 0.01	0.13 ± 0.01	0.03 ± 0.01	100%
		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCD	DD	0.27 ± 0.06	0.13 ± 0.05	0.20 ± 0.01	0.24 ± 0.08	0.29 ± 0.10	0.15 ± 0.02	0.16 ± 0.01	0.12 ± 0.01	0.14 ± 0.01	0.10 ± 0.04	0.10 ± 0.03	0.18 ± 0.03	40%
		✓	✗	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓
SDD	DD	0.85 ± 0.16	0.91 ± 0.03	0.70 ± 0.00	0.84 ± 0.16	0.95 ± 0.21	1.49 ± 0.15	1.23 ± 0.01	1.06 ± 0.01	1.32 ± 0.11	1.14 ± 0.07	1.27 ± 0.03	1.03 ± 0.03	50%
		✗	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCD	DD	0.02 ± 0.00	0.79 ± 0.03	0.76 ± 0.00	0.79 ± 0.05	0.85 ± 0.17	0.81 ± 0.06	0.82 ± 0.01	0.81 ± 0.01	0.80 ± 0.05	0.82 ± 0.05	0.80 ± 0.01	0.80 ± 0.01	30%
		✗	~	✗	~	✓	~	✓	~	~	✓	✓	✓	✓
ND	DD	1.09 ± 0.07	0.11 ± 0.04	0.35 ± 0.00	0.32 ± 0.12	0.27 ± 0.14	0.42 ± 0.09	0.39 ± 0.01	0.41 ± 0.01	0.40 ± 0.07	0.26 ± 0.02	0.24 ± 0.02	0.06 ± 0.02	100%
		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CCD	DD	0.38 ± 0.00	0.23 ± 0.00	0.14 ± 0.00	0.43 ± 0.00	0.51 ± 0.00	0.31 ± 0.04	0.29 ± 0.01	0.28 ± 0.01	0.29 ± 0.02	0.17 ± 0.00	0.18 ± 0.00	0.32 ± 0.00	70%
		✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗

Bold indicates the best results

Table 7 Comparison of different algorithms for sampling in *D3*

Distance function	Statistical features	Algorithms for Sampling											SSTHNC	
		RVS	RWS	RES	MHRW	SS	DLAS	LBDLA	GACS	SPS	SST	SPTHNC		
KSD	DD	0.81 ± 0.01	0.17 ± 0.00	0.07 ± 0.00	0.07 ± 0.02	0.62 ± 0.01	0.49 ± 0.01	0.18 ± 0.00	0.13 ± 0.00	0.51 ± 0.01	0.27 ± 0.00	0.16 ± 0.00	0.09 ± 0.00	80%
	CCD	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓
SDD	DD	0.05 ± 0.03	0.02 ± 0.00	0.00 ± 0.00	0.05 ± 0.01	0.01 ± 0.01	0.07 ± 0.01	0.04 ± 0.00	0.05 ± 0.00	0.06 ± 0.01	0.01 ± 0.00	0.01 ± 0.00	0.00 ± 0.00	90%
	CCD	✓	✓	~	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ND	DD	0.88 ± 0.01	0.85 ± 0.01	0.99 ± 0.00	0.99 ± 0.00	0.59 ± 0.00	1.31 ± 0.00	0.99 ± 0.00	0.89 ± 0.00	0.85 ± 0.00	0.99 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	20%
	CCD	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓
ND	DD	0.76 ± 0.10	1.11 ± 0.02	1.04 ± 0.03	0.75 ± 0.03	1.06 ± 0.08	0.88 ± 0.01	0.87 ± 0.00	0.89 ± 0.00	0.89 ± 0.01	0.88 ± 0.00	1.07 ± 0.00	1.03 ± 0.00	20%
	CCD	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
ND	DD	0.00 ± 0.17	0.32 ± 0.01	0.24 ± 0.01	0.19 ± 0.02	0.95 ± 0.00	0.96 ± 0.10	0.92 ± 0.00	0.75 ± 0.00	0.76 ± 0.04	0.68 ± 0.00	0.36 ± 0.00	0.19 ± 0.00	80%
	CCD	✗	✓	✓	~	✓	✓	✓	✓	✓	✓	✓	✓	✓
ND	DD	0.09 ± 0.00	0.04 ± 0.00	0.01 ± 0.00	0.08 ± 0.00	0.02 ± 0.00	0.10 ± 0.02	0.08 ± 0.00	0.07 ± 0.00	0.09 ± 0.02	0.03 ± 0.00	0.01 ± 0.00	0.01 ± 0.00	90%
	CCD	✓	✓	~	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Bold indicates the best results

Table 8 Comparison of different algorithms for sampling in *D4*

Distance function	Statistical features	Algorithms for sampling											SSTHNC	
		RVS	RWS	RES	MHRW	SS	DLAS	LBDLA	GACS	SPS	SST	SPTHNC		
KSD	DD	0.18 ± 0.02	0.17 ± 0.00	0.45 ± 0.00	0.39 ± 0.02	0.44 ± 0.01	0.27 ± 0.01	0.26 ± 0.00	0.25 ± 0.00	0.25 ± 0.01	0.39 ± 0.00	0.17 ± 0.00	0.19 ± 0.00	90%
	CCD	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SDD	DD	0.10 ± 0.03	0.10 ± 0.00	0.02 ± 0.00	0.08 ± 0.01	0.08 ± 0.01	0.13 ± 0.01	0.16 ± 0.00	0.13 ± 0.00	0.11 ± 0.01	0.08 ± 0.00	0.02 ± 0.00	0.01 ± 0.00	100%
	CCD	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ND	DD	1.26 ± 0.07	0.85 ± 0.01	0.69 ± 0.00	0.72 ± 0.00	0.69 ± 0.00	1.19 ± 0.00	0.95 ± 0.00	0.91 ± 0.00	1.14 ± 0.00	0.92 ± 0.00	0.30 ± 0.00	0.84 ± 0.00	70%
	CCD	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓
ND	DD	2.38 ± 1.13	1.11 ± 0.02	1.98 ± 0.03	1.68 ± 0.09	1.69 ± 0.08	1.42 ± 0.05	1.24 ± 0.00	1.31 ± 0.00	1.37 ± 0.04	1.68 ± 0.00	0.85 ± 0.00	1.06 ± 0.00	100%
	CCD	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ND	DD	0.34 ± 0.06	0.32 ± 0.01	0.71 ± 0.01	0.58 ± 0.02	0.68 ± 0.03	0.26 ± 0.01	0.31 ± 0.00	0.35 ± 0.00	0.26 ± 0.01	0.28 ± 0.00	1.11 ± 0.00	0.34 ± 0.00	50%
	CCD	~	✗	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓
ND	DD	0.18 ± 0.00	0.15 ± 0.00	0.19 ± 0.02	0.04 ± 0.00	0.14 ± 0.00	0.05 ± 0.00	0.05 ± 0.00	0.04 ± 0.00	0.05 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.02 ± 0.00	100%
	CCD	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Bold indicates the best results

Table 9 Comparison of different algorithms for sampling in *D5*

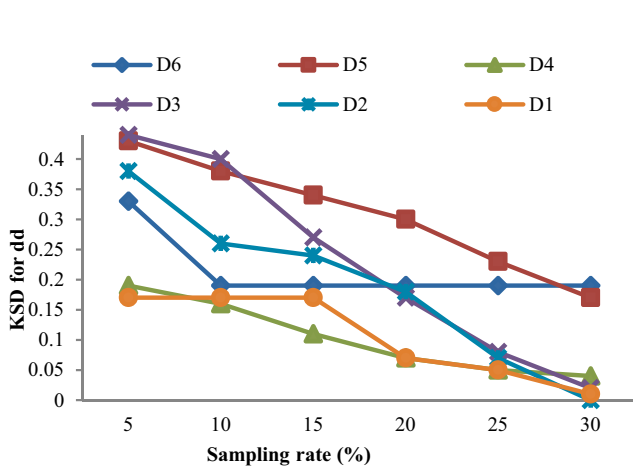
Distance function	Statistical features	Different methods												
		RVS	RWS	RES	MHRW	SS	DLAS	LBDLA	GACS	SPS	SST	SPTHNC	SSSTHNC	
KSD	DD	0.79±0.01	0.04±0.00	0.43±0.00	0.39±0.02	0.40±0.01	0.13±0.01	0.17±0.00	0.12±0.00	0.17±0.01	0.05±0.00	0.05±0.00	0.05±0.00	80%
		✓	✗	✓	✓	✓	✓	✓	✓	✓	~	~	~	
CCD	DD	0.28±0.02	0.05±0.00	0.35±0.00	0.30±0.01	0.32±0.01	0.11±0.01	0.09±0.00	0.15±0.00	0.10±0.01	0.03±0.00	0.03±0.00	0.03±0.00	90%
		✓	✓	✓	✓	✓	✓	✓	✓	✓	~	~	~	
SDD	DD	1.03±0.02	0.97±0.01	0.77±0.00	0.80±0.00	0.79±0.00	1.20±0.00	1.24±0.00	1.06±0.00	1.14±0.00	0.97±0.00	0.97±0.00	0.97±0.00	50%
		✓	~	✗	✗	✗	✓	✓	✓	✓	~	~	~	
ND	CCD	1.77±0.18	1.00±0.02	2.11±0.03	1.68±0.14	1.85±0.25	1.06±0.01	1.07±0.00	1.08±0.00	1.05±0.01	1.00±0.00	1.00±0.00	1.00±0.00	80%
		✓	~	✓	✓	✓	✓	✓	✓	✓	~	~	~	
CCD	DD	2.16±0.03	0.10±0.00	0.76±0.01	0.66±0.04	0.71±0.03	0.21±0.02	0.18±0.00	0.23±0.00	0.17±0.01	0.10±0.00	0.10±0.00	0.10±0.00	80%
		✓	~	✓	✓	✓	✓	✓	✓	✓	~	~	~	
CCD	DD	0.56±0.00	0.04±0.00	0.71±0.00	0.61±0.00	0.65±0.00	0.20±0.00	0.08±0.00	0.17±0.00	0.19±0.00	0.06±0.00	0.06±0.00	0.06±0.00	80%
		✓	✗	✓	✓	✓	✓	✓	✓	✓	~	~	~	

Bold indicates the best results

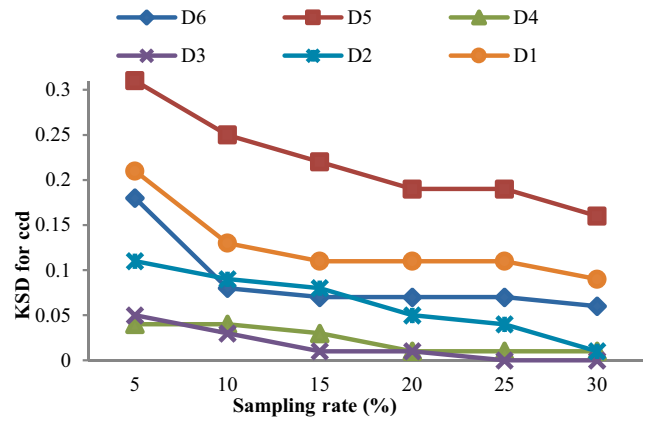
Table 10 Comparison of different algorithms for sampling in *D6*

Distance function	Statistical features	Different methods												
		RVS	RWS	RES	MHRW	SS	DLAS	LBDLA	GACS	SPS	SST	SSPTHNC	SSSTHNC	
KSD	DD	0.18±0.02	0.18±0.00	0.14±0.02	0.21±0.02	0.22±0.01	0.32±0.01	0.36±0.00	0.31±0.00	0.29±0.01	0.19±0.00	0.22±0.00	0.19±0.00	60%
		✗	✗	✗	✓	✓	✓	✓	✓	✓	~	~	~	
CCD	DD	0.10±0.03	0.08±0.06	0.03±0.00	0.08±0.01	0.09±0.01	0.13±0.01	0.17±0.00	0.15±0.00	0.11±0.02	0.07±0.00	0.07±0.00	0.07±0.00	80%
		✓	✓	✗	✓	✓	✓	✓	✓	✓	~	~	~	
SDD	DD	1.26±0.07	0.58±0.33	0.57±0.03	0.50±0.00	0.49±0.00	0.61±0.00	0.58±0.00	0.53±0.00	0.64±0.00	0.52±0.00	0.49±0.00	0.52±0.00	70%
		✓	✓	✓	✗	✗	✓	✓	✓	✓	~	~	~	
ND	CCD	2.38±1.13	3.44±3.18	0.98±0.03	0.98±0.14	0.98±0.00	1.06±0.01	1.04±0.00	1.02±0.00	1.05±0.01	0.98±0.00	0.98±0.00	0.98±0.00	60%
		✓	✓	~	~	~	✓	✓	✓	✓	~	~	~	
CCD	DD	0.34±0.06	0.24±0.01	0.18±0.02	0.26±0.04	0.29±0.00	0.27±0.00	0.35±0.00	0.27±0.00	0.27±0.00	0.24±0.00	0.29±0.00	0.24±0.00	70%
		✓	~	✗	✓	✓	✓	✓	✓	✓	~	~	~	
CCD	DD	0.18±0.00	0.16±0.00	0.06±0.00	0.16±0.00	0.18±0.00	0.17±0.00	0.18±0.00	0.13±0.00	0.16±0.00	0.14±0.00	0.15±0.00	0.14±0.00	70%
		✓	✓	✗	✓	✓	✓	✓	✗	✓	~	~	~	

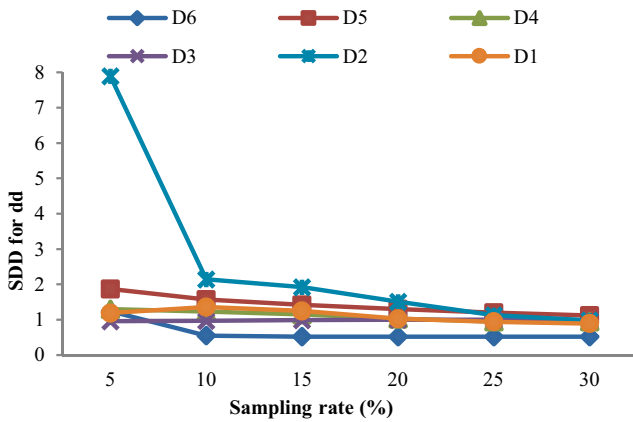
Bold indicates the best results



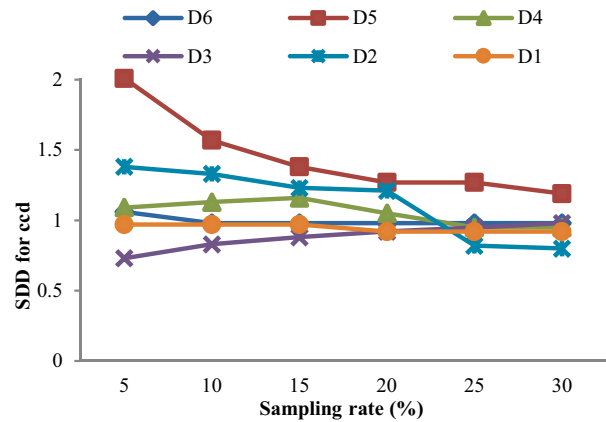
a) KSD for *DD* by SSTHNC with different sampling rates



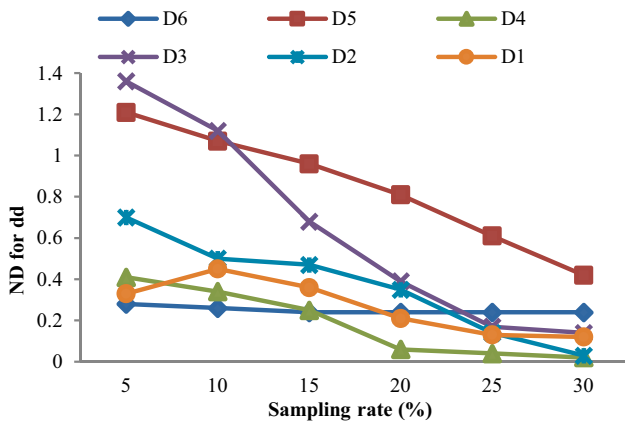
b) KSD for *CCD* by SSTHNC with different sampling rates



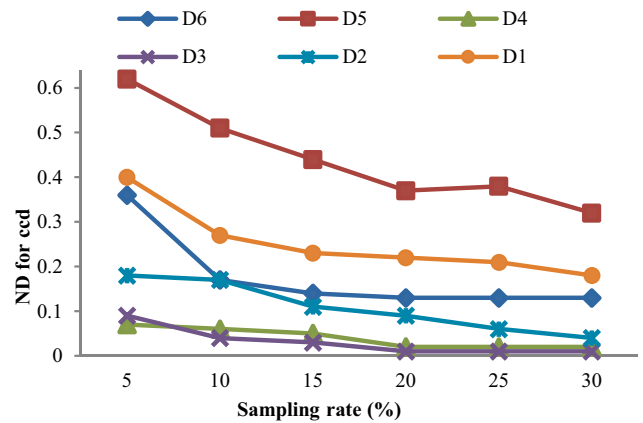
c) SDD for *DD* by SSTHNC with different sampling rates



d) SDD for *CCD* by SSTHNC with different sampling rates



e) ND for *DD* by SSTHNC with different sampling rates



f) ND for *CCD* by SSTHNC with different sampling rates

Fig. 7 Comparison of KSD, SDD, and ND with respect to *DD* and *CCD* for different sampling rates

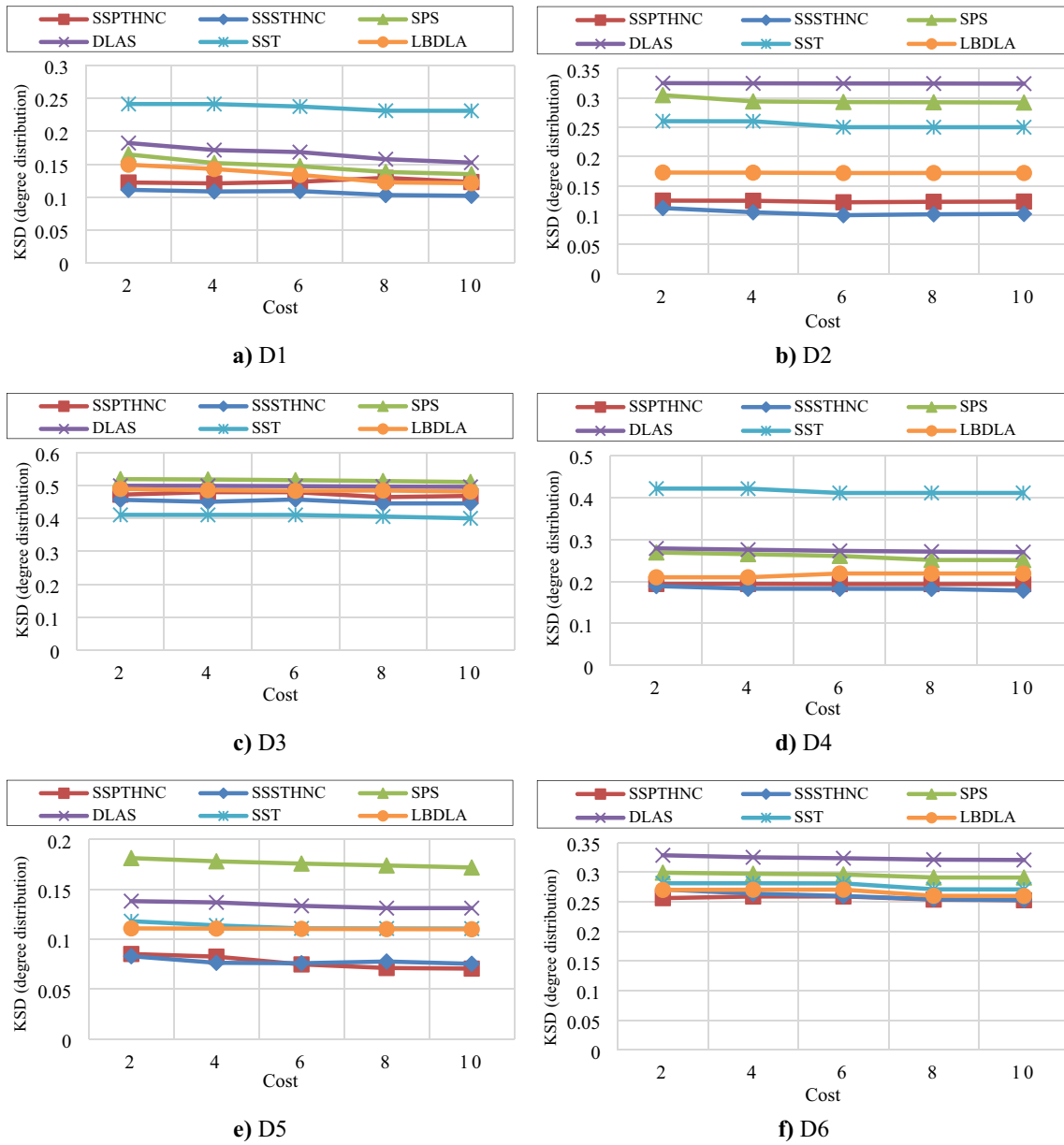


Fig. 8 Using KSD for DD versus sampling costs to compare two-phase sampling algorithms

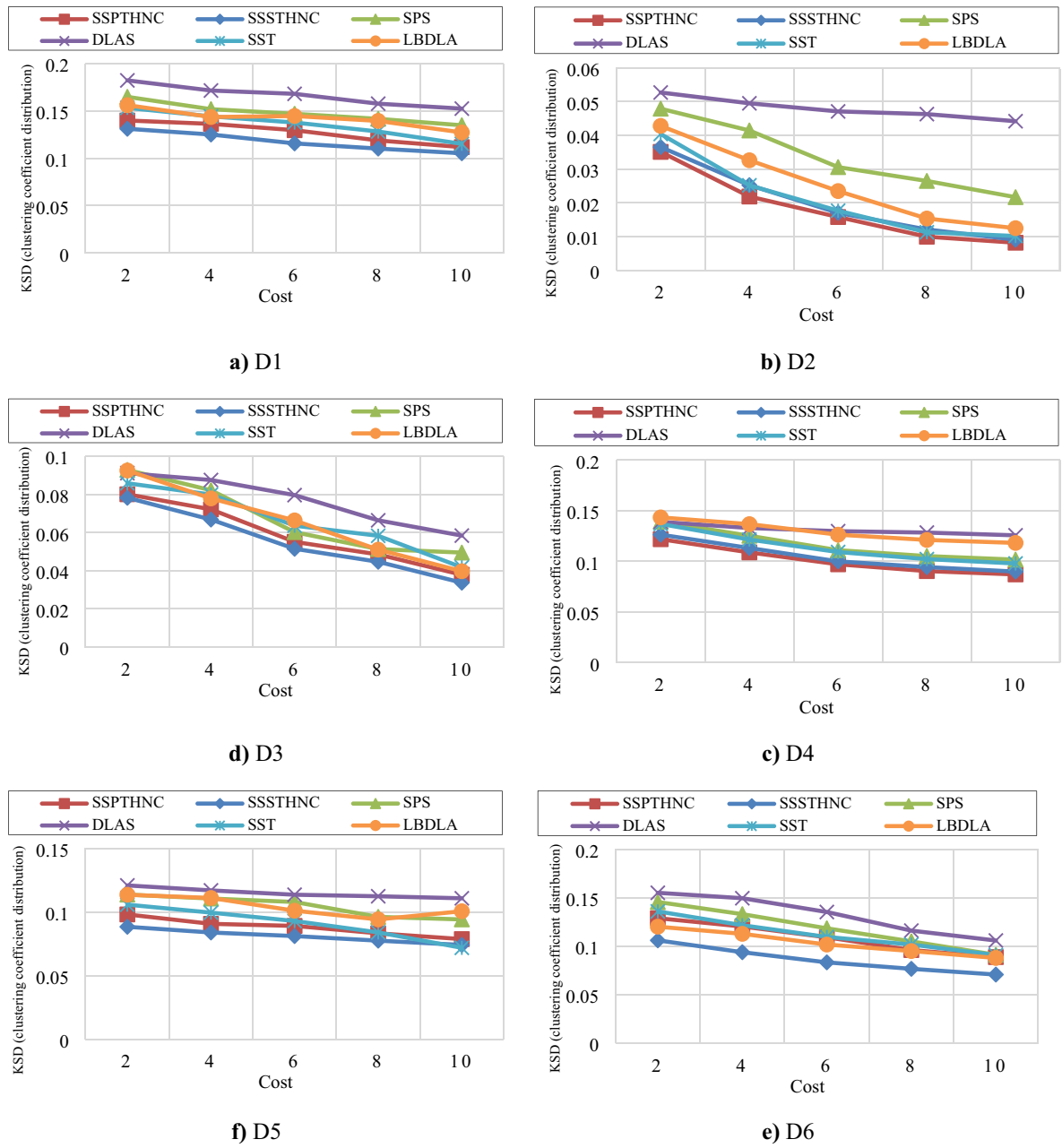


Fig. 9 Using KSD for CCD versus the sampling cost to compare two-phase sampling algorithms

STHNCs) that capture some features of the original network. Moreover, STHNC visited edges uniformly, which reduced the bias towards high-degree nodes in the sampled network. The proposed algorithms were evaluated by running several experiments on data sets based on social networks. Based on the results, the proposed algorithms outperformed other algorithms such as random vertex sampling, random edge sampling, random walk sampling, Metropolis–Hastings random walk sampling, spiral sampling, shortest path sampling, spanning tree sampling, and distributed learning automata sampling with respect to Kolmogorov–Simonov, skew divergence, and normalized distance tests.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s13278-024-01247-4>.

Authors' contributions AR and ZSJ proposed the original idea, developed the code, designed, performed the simulation experiments, wrote the main manuscript text, and prepared Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9 and S.M.V. participated in the coordination of the study. All authors planned the work, analyzed the results, and reviewed the manuscript.

Funding No funding.

Data availability Data is provided within the supplementary information files.

Declarations

Conflict of interest The authors declare no competing interests.

Ethical approval Not applicable.

References

- Ahmed NK, Neville J, Kompella R (2014b) Network sampling: from static to streaming graphs. *ACM Trans Knowl Discov Data (TKDD)* 8:7
- Ahmed NK, Berchmans F, Neville J, Kompella R (2010) Time-based sampling of social network activity graphs. In: *Proceedings of the Eighth workshop on mining and learning with graphs*. ACM, pp 1–9
- Ahmed NK, Duffield N, Neville J, Kompella R (2014a) Graph sample and hold: A framework for big-graph analytics. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp 1446–1455
- Ansari N, Cheng G, Krishnan RN (2004) Efficient and reliable link state information dissemination. *IEEE Commun Lett* 8:317–319
- Bearman PS, Moody J, Stovel K (2004) Chains of affection: the structure of adolescent romantic and sexual networks I. *Am J Sociol* 110:44–91
- Bellur B, Ogier RG (1999) A reliable, efficient topology broadcast protocol for dynamic networks. In: *INFOCOM'99. Eighteenth annual joint conference of the IEEE computer and communications societies*. Proceedings. IEEE. IEEE, pp 178–186
- Blagus N, Šubelj L, Weiss G, Bajec M (2015) Sampling promotes community structure in social and information networks. *Physica A* 432:206–215
- Blomsma N, de Rooy B, Gerritse F et al (2022) Minimum spanning tree analysis of brain networks: a systematic review of network size effects, sensitivity for neuropsychiatric pathology, and disorder specificity. *Netw Neurosci* 6:301–319
- Ebadi Jokandan SM, Bayat P, Farrokhbakht Foumani M (2021) CS- and GA-based hybrid evolutionary sampling algorithm for large-scale social networks. *Soc Netw Anal Min* 11:120. <https://doi.org/10.1007/s13278-021-00836-x>
- Erdos P, Rényi A (1960) On the evolution of random graphs. *Publ Math Inst Hung Acad Sci* 5:17–61
- Gao Q, Ding X, Pan F, Li W (2014) An improved sampling method of complex network. *Int J Mod Phys C* 25:1440007
- Garey MR, Johnson DS (1979) *Computers and intractability: a guide to the theory of NP-completeness*, 1st edn. W. H Freeman, San Francisco
- Gile KJ, Handcock MS (2010) Respondent-driven sampling: an assessment of current methodology. *Sociol Methodol* 40:285–327
- Gjoka M, Kurant M, Butts CT, Markopoulou A (2010) Walking in Facebook: a case study of unbiased sampling of OSNs. In: *Proceedings IEEE INFOCOM 2010*. San Diego, CA, pp 1–9
- Hill RJ (1999) *International comparisons using spanning trees*. In: *International and interarea comparisons of income, Output, and Prices*. University of Chicago Press, pp 109–120
- Jalali ZS, Rezvanian A, Meybodi MR (2016) Social network sampling using spanning trees. *Int J Mod Phys C* 27:1650052
- James F (2006) *Statistical methods in experimental physics*. World Scientific
- Jaouadi M, Romdhane LB (2021) A distributed model for sampling large scale social networks. *Expert Syst Appl* 186:115773
- Kurant M, Markopoulou A, Thiran P (2011b) Towards unbiased BFS sampling. *IEEE J Sel Areas Commun* 29:1799–1809
- Kurant M, Gjoka M, Butts CT, Markopoulou A (2011a) Walking on a graph with a magnifying glass: stratified sampling via weighted random walks. In: *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. ACM, pp 281–292
- Kurant M, Gjoka M, Wang Y, et al (2012) Coarse-grained topology estimation via graph sampling. In: *Proceedings of the 2012 ACM workshop on Workshop on online social networks*. ACM, pp 25–30
- Lee L (2001) On the effectiveness of the skew divergence for statistical language analysis. In: *AISTATS*. Citeseer
- Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. *ACM Trans Knowl Discov Data (TKDD)* 1:1–41
- Leskovec J, Lang KJ, Dasgupta A, Mahoney MW (2009) Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. *Internet Math* 6:29–123
- Leskovec J, Faloutsos C (2006) Sampling from large graphs. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, Philadelphia, pp 631–636
- Leskovec J, Kleinberg J, Faloutsos C (2005) Graphs over time: densification laws, shrinking diameters and possible explanations. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp 177–187
- Liu X, Zhang M, Fiumara G, De Meo P (2022) Complex network hierarchical sampling method combining node neighborhood clustering coefficient with random walk. *New Gener Comput* 40:765–807. <https://doi.org/10.1007/s00354-022-00179-x>
- Lovász L (1993) Random walks on graphs: a survey. *Comb Paul Erdos Eighty* 2:1–46
- Luo Q, Xie Z, Liu Y et al (2024) Sampling hypergraphs via joint unbiased random walk. *World Wide Web* 27:15. <https://doi.org/10.1007/s11280-024-01253-8>

- Maiya AS, Berger-Wolf TY (2011) Benefits of bias: towards better characterization of network sampling. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 105–113
- Murai F, Ribeiro B, Towsley D, Wang P (2013) On set size distribution estimation and the characterization of large networks via sampling. *IEEE J Sel Areas Commun* 31:1017–1025
- Papagelis M, Das G, Koudas N (2013) Sampling online social networks. *IEEE Trans Knowl Data Eng* 25:662–676
- Peng L, Yongli L, Chong W (2014) Towards cost-efficient sampling methods. <http://arxiv.org/abs/arXiv:14055756>
- Piña-García CA, Gu D (2013) Spiraling facebook: an alternative Metropolis–Hastings random walk using a spiral proposal distribution. *Soc Netw Anal Min* 3:1403–1415
- Rezvani A, Meybodi MR (2015) Sampling social networks using shortest paths. *Physica A* 424:254–268
- Rezvani A, Rahmati M, Meybodi MR (2014) Sampling from complex networks using distributed learning automata. *Physica A* 396:224–234
- Rezvani A, Moradabadi B, Ghavipour M, et al (2019) Social network sampling. In: Learning automata approach for social networks. Springer, pp 91–149
- Ribeiro B, Figueiredo D, de Souza e Silva E, Towsley D (2011) Characterizing continuous-time random walks on dynamic networks. In: Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems. ACM, pp 151–152
- Ribeiro B, Wang P, Murai F, Towsley D (2012) Sampling directed graphs with random walks. In: Proceedings IEEE INFOCOM. Orlando, FL, pp 1692–1700
- Roohollahi S, Khatibi Bardsiri A, Keynia F (2022) Sampling in weighted social networks using a levy flight-based learning automata. *J Supercomput* 78:1458–1478. <https://doi.org/10.1007/s11227-021-03905-2>
- Santos FC, Pacheco JM (2005) Scale-free networks provide a unifying framework for the emergence of cooperation. *Phys Rev Lett* 95:098104. <https://doi.org/10.1103/PhysRevLett.95.098104>
- Siciliano MD, Yenigun D, Ertan G (2012) Estimating network structure via random sampling: cognitive social structures and the adaptive threshold method. *Soc Netw* 34:585–600
- Son S-W, Christensen C, Bizhani G et al (2012) Sampling properties of directed networks. *Phys Rev E* 86:046104
- Sundar S, Singh A, Rossi A (2012) New heuristics for two bounded-degree spanning tree problems. *Inf Sci* 195:226–240. <https://doi.org/10.1016/j.ins.2012.01.037>
- Tewarie P, Van Dellen E, Hillebrand A, Stam CJ (2015) The minimum spanning tree: an unbiased method for brain network analysis. *Neuroimage* 104:177–188
- Wang H, Lu J (2013) Detect inflated follower numbers in OSN using star sampling. In: Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining. ACM, pp 127–133
- Wejnert C, Heckathorn DD (2008) Web-based network sampling: efficiency and efficacy of respondent-driven sampling for online research. *Sociological Methods and Research*
- White DR, Newman M (2001) Fast approximation algorithms for finding node-independent paths in networks
- Woolhouse ME, Dye C, Etard JF et al (1997) Heterogeneities in the transmission of infectious agents: implications for the design of control programs. *Proc Natl Acad Sci USA* 94:338–342. <https://doi.org/10.1073/pnas.94.1.338>
- Yoon S-H, Kim K-N, Hong J et al (2015) A community-based sampling method using DPL for online social networks. *Inf Sci* 306:53–69

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.