Graph Sampling for Visual Analytics

Fangyan Zhang▲ **and Song Zhang**
Mississippi State University, USA
E-mail: fz56@msstate.edu

Pak Chung Wong
Pacific Northwest National Laboratory, USA

Abstract. *Effectively visualizing large graphs and capturing the statistical properties are two challenging tasks. To aid in these two tasks, many sampling approaches for graph simplification have been proposed, falling into three categories: node sampling, edge sampling, and traversal-based sampling. It is still unknown which approach is the best. The authors evaluate commonly used graph sampling methods through a combined visual and statistical comparison of graphs sampled at various rates. They conduct their evaluation on three graph models: random graphs, small-world graphs, and scale-free graphs. Initial results indicate that the effectiveness of a sampling method is dependent on the graph model, the size of the graph, and the desired statistical property. This benchmark study can be used as a guideline in choosing the appropriate method for a particular graph sampling task, and the results presented can be incorporated into graph visualization and analysis tools.* © *2017 Society for Imaging Science and Technology.*
[DOI: 10.2352/J.ImagingSci.Technol.2017.61.4.000000]

INTRODUCTION

Graph analysis and visualization[1] has evolved into a very active area of research over the last several decades with applications in social network, security, high-performance computing, etc. However, as the size of a graph grows, effectively analyzing and displaying all of the vertices and edges becomes extremely difficult.[2]

Graph sampling is needed in graph analysis for several reasons. The first reason is visualization. Displaying even a relatively small graph of several thousand vertices on a screen is challenging because of the limit in screen size. Further, even if we could display all of the vertices and edges, it is often difficult to discern the internal structure. Sampling provides an abstract version of the original graph. Thus, visualizing sampling results is easier than visualizing the original.[2] The second reason is that analysis of a large graph is costly. Proper sampling approaches help us estimate the graph properties on a smaller sample, thereby greatly reducing the computational cost.[3] The third reason is incomplete graph data.[4] In some cases, obtaining all data for a graph is not permitted or is very time-consuming. Thus, we must obtain the properties of the graph by sampling.

For the above reasons, sampling algorithms aim to reduce the complexity of graph drawing while preserving properties of the original graph, allowing analysis of the small sample to yield the characteristics similar to those of the original graph.

While numerous graph sampling techniques have been proposed,[5,6] there has not been a systematic empirical comparison of existing methods. Practical questions often arise regarding which sampling method one should use for a particular application. To answer these questions, we need to identify graph properties and metrics that facilitate a fair and conclusive comparison of sampling approaches. In turn, we need to use these metrics to ascertain which sampling methods are most suitable for estimating specific graph properties. To reflect the diversity of real-world graphs in this study, we choose three commonly seen graph types: random graphs, small-world graphs, and scale-free graphs.

Although these three graph models are widely discussed in graph research, many real-world graphs are too complex to be sufficiently modeled by any current research approaches. We designed a benchmark for comparing sampling methods for artificial random graphs, artificial small-world graphs, artificial scale-free graphs, and real-world graphs. Our comparison considers two complementary aspects: (1) how effectively the method preserves the graph's visual properties and (2) how well it preserves the graph's statistical properties. We conducted our study on directed and undirected graphs separately and used a number of statistical properties for comparison. To properly compare graph sampling methods for visualization, we fixed the graph layout in both the original and sampled graphs. The visual and statistical comparison provided criteria for selecting sampling methods in application.

The main contributions of our work are as follows:

- We implemented twelve graph sampling techniques in the benchmark.
- We built a benchmark for evaluating graph sampling methods with both visual and statistical properties.
- We studied a number of graph data sets with the benchmark and analyzed the results.

▲IS&T Member.

Received July 18, 2016; accepted for publication Jan. 3, 2017; published online xx xx, xxxx. Associate Editor: David kao.
1062-3701/2017/61(4)/000000/12/$25.00

RELATED WORK

Existing graph sampling algorithms can be classified into three types: node sampling, edge sampling, and traversal-based sampling.[5,7–9] Node sampling constructs subgraphs based on sampling vertices, often uniformly. In some cases, node sampling methods integrate traversal-based sampling in order to use graph topology information, such as random walk sampling. The metropolis algorithm[10] is a modified version of node sampling. It replaces some sampled vertices with other vertices, which often leads to sampled graph properties that are consistent with the original. Similarly, edge sampling builds a subgraph by randomly sampling edges. Traversal-based sampling creates subgraphs based on the topological information from the original graph. These methods do not sample vertices or edges directly but instead select vertices using traversal-based algorithms. Breadth-first,[11] random walk,[12,13] and snowball sampling[12] are commonly used traversal-based sampling algorithms that select vertices based on the topological information of the graph.

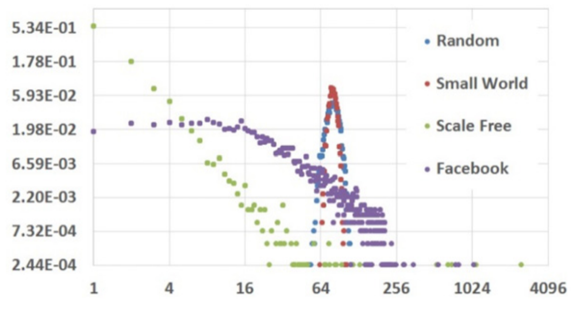
One purpose of sampling is to simplify the graph for better visualization. With millions or billions of vertices or edges, it is challenging to clearly visualize all of them. Even when the entire graph can be displayed, graph visibility and usability are issues.[14] Numerous techniques have been proposed to approach graph visualization, such as clustering,[14] sampling,[2] and special layout.[15] These techniques aim to reduce the overlap between vertices and edges. Sampling approaches improve visualization by sampling the original graph, resulting in fewer vertices and edges. Layout techniques explore vertex and edge arrangements when displaying graphs. Many layout techniques have been proposed, such as Tree layout,[16,17] 3D layout,[18] hyperbolic layout,[19] and force-directed layout.[20] Clustering reduces vertex and edge overlap by replacing clusters with vertices. Two graph clustering techniques are vertex clustering[21] and edge clustering.[22]

VISUAL AND STATISTICAL BENCHMARK

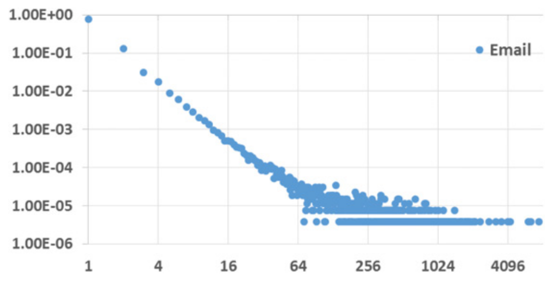
To build visual and statistical benchmark for sampling methods, several graph models and their typical degree distribution will be introduced, and additionally eight undirected graph properties and nine directed graph properties will be presented. These properties are used for comparing twelve widely used sampling methods. Those sampling methods are also discussed in this section. Finally, we will talk about eight datasets used in our experiment.

Graph Models

We hypothesize that graph type is one of the main factors affecting sampling methods' performance. Thus, we simulate three types of graphs in this study. The key motivation is to develop graph models that fit many real-world graphs. Towards this end, we use the random graph model, the small-world graph model, and the scale-free graph model as well as real-world graphs.



(a)



(b)

Figure 1. (a) Degree distribution of random graph model (blue), small-world graph model (red), scale-free graph model (green), and real social graph (magenta) with Log 2 axis. (b) Degree distribution of email graph (email graph is a type of graph data we used in the article. See details in Graph Datasets).

The degree distributions of the three graph models and a real social network graph are illustrated in Figure 1(a). The degree distribution of an email graph (an email graph is a type of graph data we used in the article; see details in Graph Datasets) is shown in Fig. 1(b). They are drawn separately because of their different scales in degree distribution. We find that the real social network graph is a complex graph that is different from any theoretical models, although it shows some similarities to a small-world graph. We apply sampling methods to the three graph models and the real-world graph. The details of the data are further discussed later in the Graph Datasets section.

Graph Properties

We use eight graph properties for comparing undirected graphs and nine graph properties for comparing directed graphs. For undirected graphs, we use the degree distribution (DD), average neighbor degree distribution (ANDD), degree centrality distribution (DCD), node betweenness centrality distribution (NBCD), edge betweenness centrality distribution (EBCD), local clustering coefficient distribution (LCCD), closeness centrality (CCD), and eigenvector centrality distribution (EVCD). For directed graphs, we use in-degree distribution (InDD), out-degree distribution (OutDD), in-degree centrality distribution (InCD), out-degree centrality distribution (OutCD), ANDD, NBCD, EBCD, CCD, and EVCD.

- **Degree distribution.** A vertex's degree is the number of edges connected to that vertex. The degree distribution is the probability distribution of a vertex's degree. For directed graphs, because a vertex has incoming and outgoing edges, the graph has in-degree distribution and out-degree distribution.

- **Degree centrality.** Degree centrality describes the importance of vertices by using the degree metric of the graph. The degree centrality for a vertex v is the fraction of vertices it is connected to. Given a graph $G = (V, E)$, n vertices, for a vertex v, degree centrality can be represented as in Ref. 23

$$C_{(d)}(v) = \frac{\deg(v)}{n-1}.$$

- **Average neighbor degree.** This property returns the average degree of the neighborhood of each vertex. It is represented as follows:[24]

$$A_{(d)}(v) = \sum_{i=1}^{n(v)} \frac{d_{(i)}}{n(v)}$$

where $n(v)$ are the numbers of neighbors of vertex v and $d_{(i)}$ is the degree of vertex i which is connected to vertex v.

- **Betweenness centrality.**[25,26] Betweenness centrality indicates the probability of the vertex acting as a bridge along the shortest path between two other vertices. Betweenness has two categories: vertex betweenness centrality and edge betweenness centrality. Vertex betweenness centrality is an indicator that shows a vertex's centrality in a graph, which refers to how many shortest paths from all vertices to all others pass through that vertex.[25] If a vertex has a high probability to be chosen as a bridge in shortest path between two other vertices, then it would have a high betweenness. Betweenness centrality of a vertex $C_B(v)$ is defined as

$$C_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} represents the number of shortest geodesic paths from s to t, and σ_{st} (v) is the number of shortest paths from s to t via v.

Likewise, edge betweenness centrality is the probability of an edge acting as a bridge between two other edges in their shortest path. Edge betweenness centrality is defined in the formula:

$$C_B(e) = \sum_{s \neq t \neq v \in V} \frac{\sigma_{(s,t)}(e)}{\sigma_{(s,t)}}$$

where σ_{st} represents the number of shortest geodesic paths from s to t, and $\sigma_{st}(e)$ is the number of shortest paths from s to t through e.

- **Clustering coefficient.** Clustering coefficient is a measure of how vertices in a graph cluster together. It includes the global clustering coefficient and the local clustering coefficient. The former, which is a graph property, shows the overall indication of the clustering in the graph. The latter is a vertex property indicating the embeddedness of the vertex.

Global clustering coefficient is defined based on triangles. This coefficient measures the clustering in the whole graph. The global clustering coefficient can be represented as in Ref. 27,

$$C_g = \frac{3 \times N_{triangles}}{N_{triplets}}.$$

$N_{triangles}$ represents the number of triangles in the graph. $N_{triplets}$ stands for the number of connected triplets of vertices. A triplet consists of three graph vertices. If two of them are connected, then the triplet is called an "open triplet." If three of them are connected, then it is called a "closed triplet." Local clustering coefficient is a metric to evaluate how close a vertex's neighbors are to each other. If k_v denotes the number of neighbors of v, and e_v is the number of connected pairs between all neighbors of v, then the local clustering coefficient can be defined in undirected graphs as in Ref. 28 $C_v = (2e_v/k_v(k_v - 1))$ and in directed graphs as

$$C_v = \frac{e_v}{k_v(k_v - 1)}.$$

- **Closeness centrality.**[29] Closeness centrality describes how central the vertex is in the graph. It is defined as the reciprocal of the sum of the distances from all other vertices that the vertex is connected to.

$$C(v) = \frac{1}{\sum_{t \in V} dist(v, t)}$$

- **Eigenvector centrality.**[30] Eigenvector centrality is a measure of the weight of a vertex in a graph. Each vertex is assigned a value based on the concept that connections to high-scoring vertices contribute more to the score of the vertex than equal connections to low-scoring vertices.

If $\boldsymbol{A}$ is the adjacency matrix of the graph and $\mathbf{x}$ is an initial eigenvector, then there will be a positive solution λ to the following equation:

$$\mathbf{Ax} = \lambda \mathbf{x}.$$

There will be a positive solution λ with the final eigenvector from the formula after using a power method based on Perron–Frobenius theorem.[31] λ is also the largest eigenvalue associated with the eigenvector of the adjacency matrix.

In this article, we use Graph-tool[32] to analyze the above graph properties because of its fast speed.

Graph Sampling Methods

Within the benchmark, we implement twelve widely used sampling methods. These sampling methods include random node (RN), random node-edge (RNE), random node-neighbor (RNN), random edge (RE), induced edge (IE), breadth-first (BF), depth-first (DF), random-first (RF), snowball (SB), random walk (RW), random walk with escape (RWE), and forest fire (FF) sampling. For all these sampling methods, sampling rates are defined as the ratio between the edges after the sampling and before the sampling.

Node Sampling

- **Random node sampling.**[7,8] Vertices are sampled randomly and uniformly, creating a subgraph of the original graph. Those edges that are connected between those sampled vertices in the original graph enter the sample graph.
- **Random node-edge sampling.**[5,8] This method is based on random node sampling. After vertices are uniformly sampled, edges that are incident to these vertices are uniformly included in the sample graph.
- **Random node-neighbor sampling.**[6] This sampling method is similar to random node-edge sampling. Vertices are uniformly random sampled first, but all the edges connected to these vertices in the original graph are sampled into subgraph.

Edge Sampling

- **Random edge sampling.**[8,12] Edges are sampled randomly and uniformly, and then a subgraph is created from those edges.
- **Induced edge sampling.**[7] Induced edge sampling includes totally induced edge sampling and partially induced edge sampling. Totally induced edge sampling has two steps. First, it conducts random edge sampling and obtains adjacent vertices from these edges. Second, all edges attached to those vertices are sampled in a subgraph. Partially induced edge sampling performs edge sampling in a single pass in which edges are selected with a probability. Incident vertices are also added to the sampled graph if one edge is selected. In this article, we implement totally induced edge sampling.

Traversal-based Sampling

- **Breadth-first sampling.**[5,11] This sampling method is induced from the graph traversal algorithm breadth-first search. It begins with a random vertex and visits its neighbors iteratively. For each iteration, the first visited vertex will enter the sample first. A subgraph is created from sampled vertices and those edges that are connected between those sampled vertices in the original graph.
- **Depth-first sampling.**[33] This approach derives from the depth-first search algorithm. For each iteration, the first visited vertex will enter the sample last.
- **Random-first sampling.**[5] This algorithm is similar to breadth-first sampling and depth-first sampling except that vertices are selected randomly in each iteration.
- **Snowball sampling.**[12] This sampling method first picks up a starting vertex at random and puts it in the current vertex set, and then all vertices that are connected to any vertex in the current vertex set are chosen and put into the current vertex set recursively until the required number of vertices is selected.
- **Random walk sampling.**[33] This method starts at a seed vertex, and then chooses a vertex uniformly at random from the neighbors of the current vertex. A subgraph is created from the walking paths. Random walk with escape or jump and multiple independent random walkers are proposed based on the classic random walk sampling method.
- **Random walk with escape or jump sampling.**[8,34] This sampling method is the same as random walk except that the current walker vertex jumps to another random vertex with probability p.
- **Forest fire sampling.**[7] This sampling approach can be regarded as a probabilistic version of breadth-first sampling. Neighbors are chosen to be added to the subgraph with probability p. The number of vertices to be chosen is a random number taken from a geometric distribution with mean $p_f/(1-p_f)$. (p_f is set to 0.5 in this article.) **Q.1**

We apply these sampling methods to four types of graphs: scale-free graph, random graph, small-world graph, and real-world graphs. These graphs can be undirected or directed.

Graph Datasets

The datasets we used in the article are collected from several data sources. The social graphs, citation graphs, email communication graphs, and internet graphs are downloaded from Stanford Network Analysis Platform (SNAP). The small-world and random graphs are created from NetworkX[35] via corresponding graph models.

In NetworkX, a random graph is created using the Erdős–Rényi graph model, introduced by Paul Erdős and Alfréd Rényi in 1959.[36] Vertices in the graph are connected randomly, and each edge exists in the graph with probability p. If a random graph has n vertices, and the edge's probability is p, then the probability of a vertex with degree k is

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-k-1}$$

where n is the number of vertices in graph. The degree distribution of the graph follows a Poisson distribution. Random graph creation[36] is straightforward and simple. Given a random graph with n vertices, an edge existing between any pair of vertices has the probability p, independent of the existing edges in the graph.

Small-world graphs are generated using small-world graph models presented by Watts and Strogatz.[28] Small average shortest path length and high clustering coefficient characterize the graph model. The typical distance L between

Table I. Eight test datasets and their properties.

Graph Dataset	Graph Type	Model	# Vertices	# Edges
Random	Directed	Model	10,000	100,246
Small-World	Undirected	Model	10,000	21,895
Scale-Free	Directed	Model	10,000	18,838
Email	Directed	Real	265,214	420,045
Citation	Directed	Real	34,546	421,578
Internet	Directed	Real	10,876	39,994
Facebook	Undirected	Real	4,039	88,234
U.S. Flight	Undirected	Real	235	1,297

any two vertices is proportional to logarithm of number of graph vertices N

$$L \propto \mathrm{Log} N.$$

The generation process of a small-world graph can be described as follows.[28] The initial graph is a ring with n vertices, and each vertex has k edges connected to its nearest neighbors. This is a regular graph in which each vertex has the same number of connection to other vertices. After the regular graph is generated, each edge is rewired to another uniformly vertex with probability p. If p equals 0, then the graph is a one-dimensional lattice graph or a regular graph. If p is 1, then the graph will become a random graph. From the generation of a small-world graph, we also find that the rewiring process reduces average path length, but the clustering coefficient is generally as high as in regular graphs.

Scale-free graphs are constructed using the scale-free graph model proposed by Baraasi and Albert.[37] In a scale-free graph, vertices asymptotically follow a power law on degree distribution:

$$P(k) \approx \frac{1}{k^r} \quad (2 < r < 3)$$

where $p(k)$ is the fraction of degree k in graph. r is a constant value usually ranging from 2 to 3.

The generation process of scale-free graph starts from n vertices and no edge connecting them. New vertices and edges are added to the graph in each step. The probability of the new vertex connecting to existing vertices is proportional to their degrees in the graph:

$$P_i = \frac{k_i}{\sum_{j=1}^{n} k_j}$$

where P_i is the probability of vertex i with degree k_i to connect to a new vertex. n is the existing number of vertices. The preferential bias to high degree of vertices is termed "preferential attachment." The growth of a graph based on preferential attachment results in a power-law degree distribution. Table I summarizes the properties of the eight datasets used for the comparison study.

Statistical Comparison

From those graph properties mentioned above, we can obtain graph property distributions of vertices or edges. We evaluate the sampling techniques based on the comparison of the graph property distributions between sampling methods. A good sampling method should produce a sampled graph with sampling results that approximate the original graph. That is, the probability distributions of the properties of the two graphs should have a short distance between them. Here we use skew divergence (SD) to evaluate the difference between two distributions.[38] Generally, skew divergence is used to measure Kullback–Leibler (KL) divergence between two probability density distributions that do not have continuous support over the range of values. Because graph properties distributions are not continuous, e.g., clustering coefficient, the two probability density distribution should be smoothed before computing KL divergence. We use the same strategy as the article[38,39] to smooth the distributions:

$$SD(P, Q, \alpha) = KL[\alpha P + (1-\alpha)Q || \alpha Q + (1-\alpha)P].$$

To better compare the sampling results, we use the average SD defined in the article, and α is set to 0.99 as in the article.[38,39] Previous work[38] has proven that SD has better performance to approximate KL divergence on non-smoothed distributions.

In our experiment, the above eight datasets are used in statistical comparison. To compare computing time between sampling methods, we record the execution time for each method.

Visual Comparison

We use Gephi[40] to visually compare sampling methods. We first draw the original graph and use this layout for all sampled graphs—i.e., the same vertex in all sampled graphs will occupy the same location as in the original graph. Also, the same vertex in all sampled graphs has the same color and label size as the original graph. We do not preserve the attributes of edges, such as edge color, edge weight, etc.

Because of space limitations, only two datasets of visual comparison are provided in the article: U.S. flight graph data and social graph data. For the flight graph, we use the geospatial layout, and for the social graph data, we use the force-directed layout.

RESULTS AND ANALYSIS

Results

We apply the sampling approaches to the U.S. flight graph, social graph, citation graph, internet graph data, email communication graph, random graph, scale-free graph, and small-world graph data. In this article, we conduct experiments using a sampling rate ranging from 10 to 50% with a 10% interval on all eight graph datasets. For each sampling rate, we perform graph sampling 10 times, and take the average SD value for this sampling rate. All sampling rates are based on the number of edges. We use the average SD value as final results for analyzing each graph metric. Because

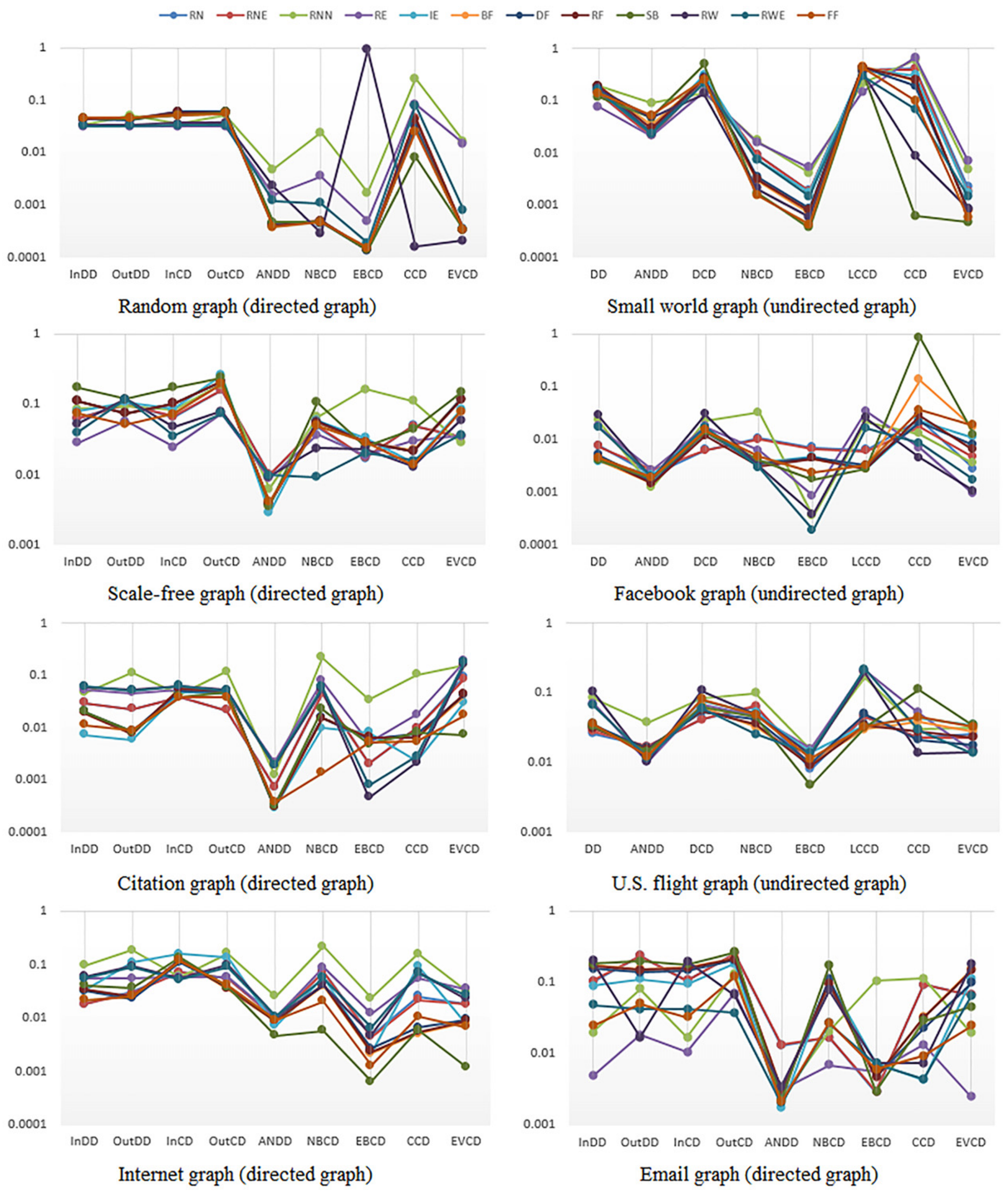


Figure 2. Average result of the statistical comparisons between sampling methods with 10–50% sampling rates. The vertical axis is SD values, horizontal axis represents graph properties, and lines are sampling methods.

of space limitations, we do not list all individual results in this article.

The line charts in Figure 2 show the SD divergence between the sampling result and the original graph for each sampling method on statistical properties.

The vertical axis in the line chart is the SD value between the sampling result and the original graph ranging from 0 to 1. A smaller value indicates more consistency between the sampling results and the original graph. The horizontal axis lists the graph properties. Each line in the chart represents one sampling method. Its value indicates the sampling method's performance. From the benchmark, a user who is working on a particular type of graph can identify which sampling method performs the best for each graph property for that particular type of graph.

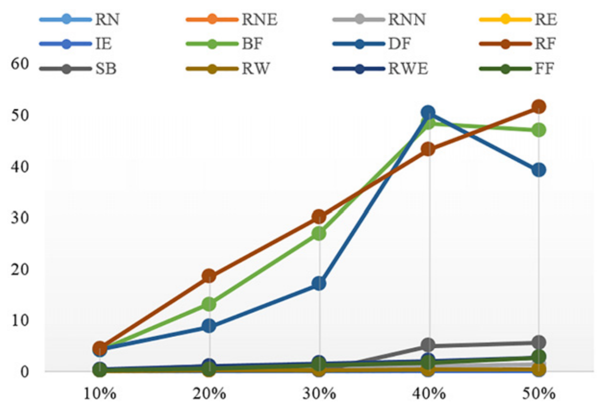


Figure 3. Execution time between sampled methods for Facebook graph data. Lines represent sampling methods. X axis represents sampling rate for each sampling method. Y axis represents execution time in seconds.

In addition, when conducting sampling on the graph dataset, we record the execution time for each sampling method. For example, the computing time for Facebook data is shown in Figure 3.

Figure 4 shows the visual comparison between sampling methods using a 10% sampling results for the U.S. flight graph data. Figure 5 shows the same type of comparison on the Facebook graph data. In both visualizations, vertex label size is positively proportional to its degree.

Analysis

The benchmark allows us to compare sampling methods quantitatively and qualitatively in several aspects. First, the sampling experiment covers U.S. flight graph, social graph, citation graph, internet graph, email communication graph, scale-free graph, random graph, and small-world graph. This approach allows us to analyze the sampling results for different graph data types. Second, for each sampling method, we use nine properties for directed graph and eight properties for undirected graph. For each graph property, we observe the average SD divergence between the sampling results and the original graph, and then determine whether the sampling methods have stable performance for that graph property. Third, in order to compare the efficiency between sampling methods, we also record the execution time for each sampling algorithm. Finally, we conduct visual comparison between these sampling results. We list the observations from the results below. These findings could potentially provide guidance for users when choosing sampling methods in their applications.

Comparison between Graph Types

In our experiment, we apply sampling methods to eight graph datasets, including random graph, small-world graph, scale-free graph, and five real-world graphs. We summarize the sampling results for each graph type by averaging the results from different sampling methods and then compare them. Figure 6 is summarized from Fig. 2 by averaging the SD values of all sampling method. Because directed graph and undirected graph have different properties, we summarize directed graphs (Fig. 6 (a)) and undirected graphs (Fig. 6(b)) separately. From Fig. 6 (a), we observe that the sampling result of the small-world graph (red line) deviates significantly from both the Facebook graph (green line) and the U.S. flight graph (dark blue line), but the Email graph (cyan line) has a very similar pattern to the scale-free graph (dark blue line). Further, by observing both Figs. 1 and 6, we find that two graphs with similar degree distributions, e.g., email graph and scale-free graph, share similar sampling results. Degree distribution is an important characteristic of the graph type. From these observations, it is obvious that graph type has significant influence on sampling results and should be considered when sampling graphs.

Comparison between Graph Properties

We analyze how sampling methods perform on each graph property. As shown in Fig. 2, all sampling methods not only fluctuate from property to property but also scatter along each property vertically. Hence, sampling methods are dependent on graph properties.

In addition, after reviewing sampling methods' performance on undirected graph and directed graph respectively, we find that only a few methods act consistently well on certain graph properties across graph types. Random walk sampling works well on closeness centrality distribution in undirected graphs, and induced edge sampling behaves consistently well on average neighbor degree distribution of directed graph. Furthermore, for a certain graph data, some methods preserve certain graph properties very well. For instance, in email graph, scale-free graph, and random graph, random edge sampling performs consistently well in-degree distribution, out-degree distribution, in-degree centrality distribution, and out-degree centrality distribution. In internet graph, snowball sampling performs well in out-degree centrality distribution, average neighbor degree distribution, node betweenness centrality distribution, edge betweenness centrality distribution, local clustering coefficient distribution, closeness centrality, and eigenvector centrality distribution. These observations indicate that graph property should be considered when choosing sampling methods in application.

Comparison of Execution Time

To compare the efficiency of the sampling methods, we record the time for all methods during the sampling process. In Fig. 3, we find that the sampling process generally takes more time as data size increases with the sampling rate. Random sampling methods, such as random node sampling, random edges sampling, random node neighbor, random node edge, and induced edge are not sensitive to sampling rate. However, for traversal-based sampling methods, such as breadth-first, depth-first, random walk sampling, etc., the execution time grows rapidly with the sampling rate. For random sampling, such as, the complexities of random node sampling and random edge sampling are $O(n)$ and $O(m)$, respectively, where n is the number of vertices and m stands for the number of edges. For traversal-based sampling, the

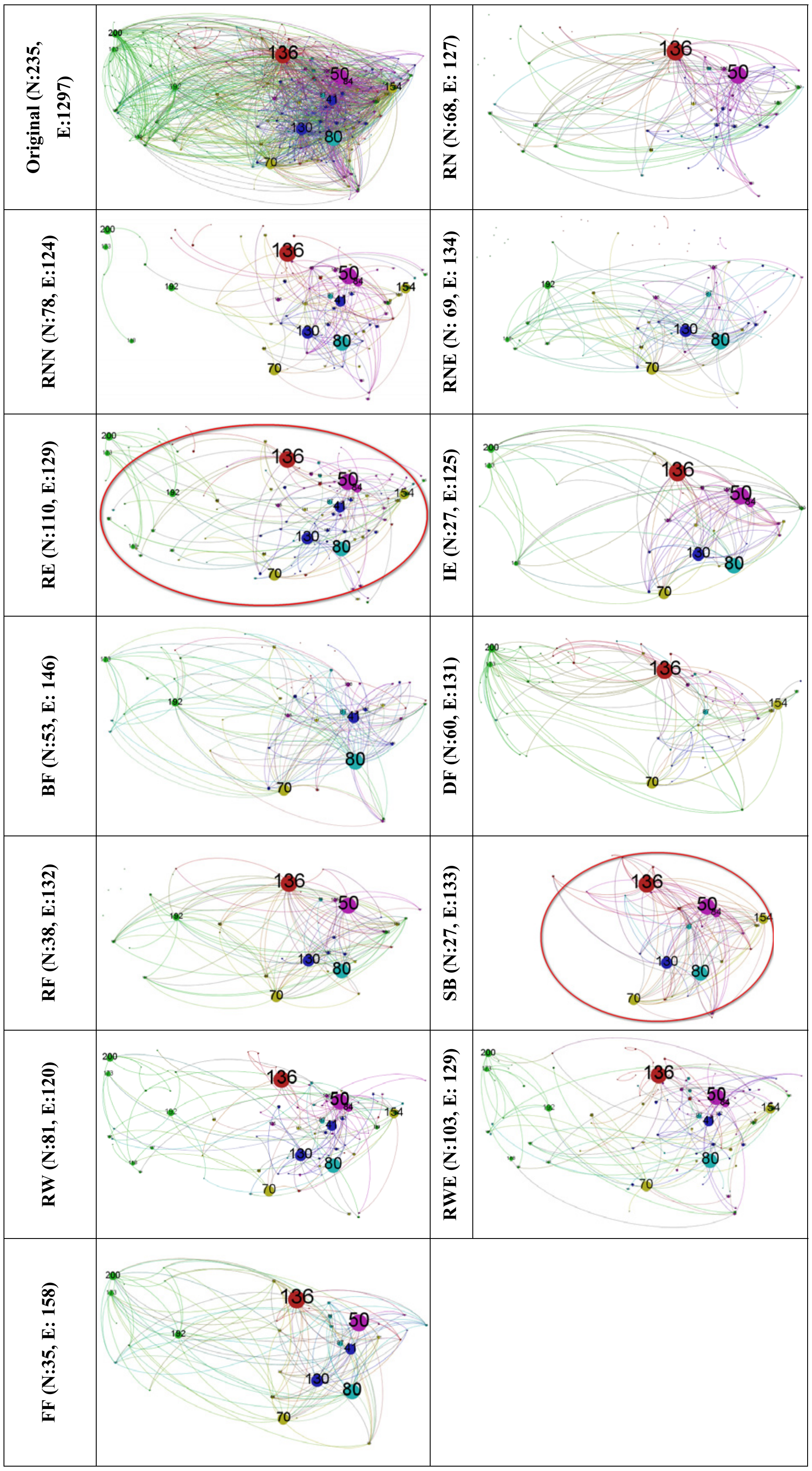


Figure 4. Visual comparison between sampling methods for U.S. flight data (undirected graph) with sampling rate 10% on edges. Red Circles in RE and SB show spatial coverage area of sampling results.

Table II. Top three sampling methods for each graph property for each graph type. The sampling methods in red indicate the corresponding SD values are greater than 0.1.

Undirected Graph	**Small World Graph**								
	DD	**ANDD**	**DCD**	**NBCD**	**EBCD**	**LCCD**	**CCD**	**EVCD**	
	RE	RE	RW	FF	SB	RE	SB	SB	
	SB	RN	RNN	SB	FF	RNN	RW	FF	
	FF	RNE	RE	RW	RW	RWE	RWE	BF	
	US Flight graph								
	DD	**ANDD**	**DCD**	**NBCD**	**EBCD**	**LCCD**	**CCD**	**EVCD**	
	RN	RE	RNE	RWE	SB	BF	RW	RWE	
	RNE	RW	RN	BF	RN	SB	DF	RW	
	RF	RWE	DF	RF	RF	FF	RNE	RE	
	Facebook Graph								
	DD	**ANDD**	**DCD**	**NBCD**	**EBCD**	**LCCD**	**CCD**	**EVCD**	
	IE	RNN	RNE	RWE	RWE	BF	RW	RE	
	SB	RF	RN	RW	RNN	SB	RE	RW	
	BF	BF	RF	RF	RW	IE	RWE	RWE	
Directed Graph	**Scale-Free Graph**								
	InDD	**OutDD**	**InCD**	**OutCD**	**ANDD**	**NBCD**	**EBCD**	**CCD**	**EVCD**
	RE	FF	RE	RE	IE	RWE	RN	RW	RNN
	RWE	RE	RWE	RWE	SB	RW	RNE	FF	RN
	RW	RF	RW	RW	DF	RE	RE	IE	RNE
	Email Graph								
	InDD	**OutDD**	**InCD**	**OutCD**	**ANDD**	**NBCD**	**EBCD**	**CCD**	**EVCD**
	RE	RW	RE	RWE	IE	RE	RN	IE	RE
	RNN	RE	RNN	RW	RWE	RN	RNE	RWE	RNN
	FF	RWE	FF	RE	FF	RNE	SB	RW	FF
	Citation Graph								
	InDD	**OutDD**	**InCD**	**OutCD**	**ANDD**	**NBCD**	**EBCD**	**CCD**	**EVCD**
	IE	IE	FF	RNE	IE	FF	RW	RW	SB
	FF	RF	SB	RN	RF	IE	RWE	IE	FF
	BF	BF	RN	FF	BF	RF	RNE	RWE	IE
	Random Graph								
	InDD	**OutDD**	**InCD**	**OutCD**	**ANDD**	**NBCD**	**EBCD**	**CCD**	**EVCD**
	RE	RE	RE	RE	FF	RW	DF	RW	RW
	RWE	RWE	RW	RWE	IE	DF	SB	SB	DF
	RNN	RW	RNN	RW	RF	SB	BF	FF	BF
	Internet Graph								
	InDD	**OutDD**	**InCD**	**OutCD**	**ANDD**	**NBCD**	**EBCD**	**CCD**	**EVCD**
	RNE	DF	RWE	BF	SB	SB	SB	BF	SB
	RN	BF	RW	RF	IE	FF	FF	RF	FF
	FF	RF	RE	SB	RF	BF	BF	SB	IE

complexity of breadth-first sampling is $O(n+m)$, which explains why random sampling is less sensitive to sampling rate than traversal-based sampling.

Visual Comparison

Using visual comparison, we can find out which sampling method preserves the visual cues of a graph. We provide two criteria for visual comparison and analyze how well each sampling method performs.

First, because the locations of vertices in the graph layout are fixed in our visualization, we define the spatial coverage as one criteria of visual comparison between sampling results. The sampling method that produces similar spatial coverage to that of the original graph is considered good. From the visualization of the sampling results, we find that random sampling methods have better spatial coverage than traversal-based sampling, in particular for a small sampling rate. For example, in Figs. 4 and 5, random edge sampling results (red circle) of the U.S. flight and Facebook graphs cover the major spatial area of the original graph. However, traversal-based sampling, such as snowball sampling, does not produce good spatial coverage. That is because traversal-based sampling cannot sample far-ranging vertices or edges as efficiently as random sampling methods for a small sampling rate. For example, the snowball sampling result (red circle) in Fig. 5 only covers a small local area.

Second, clustering is an important task in graph research. We define another visual comparison criterion in sampling as the ability to preserve the size, shape, and number of clusters. In this regard, we observe that

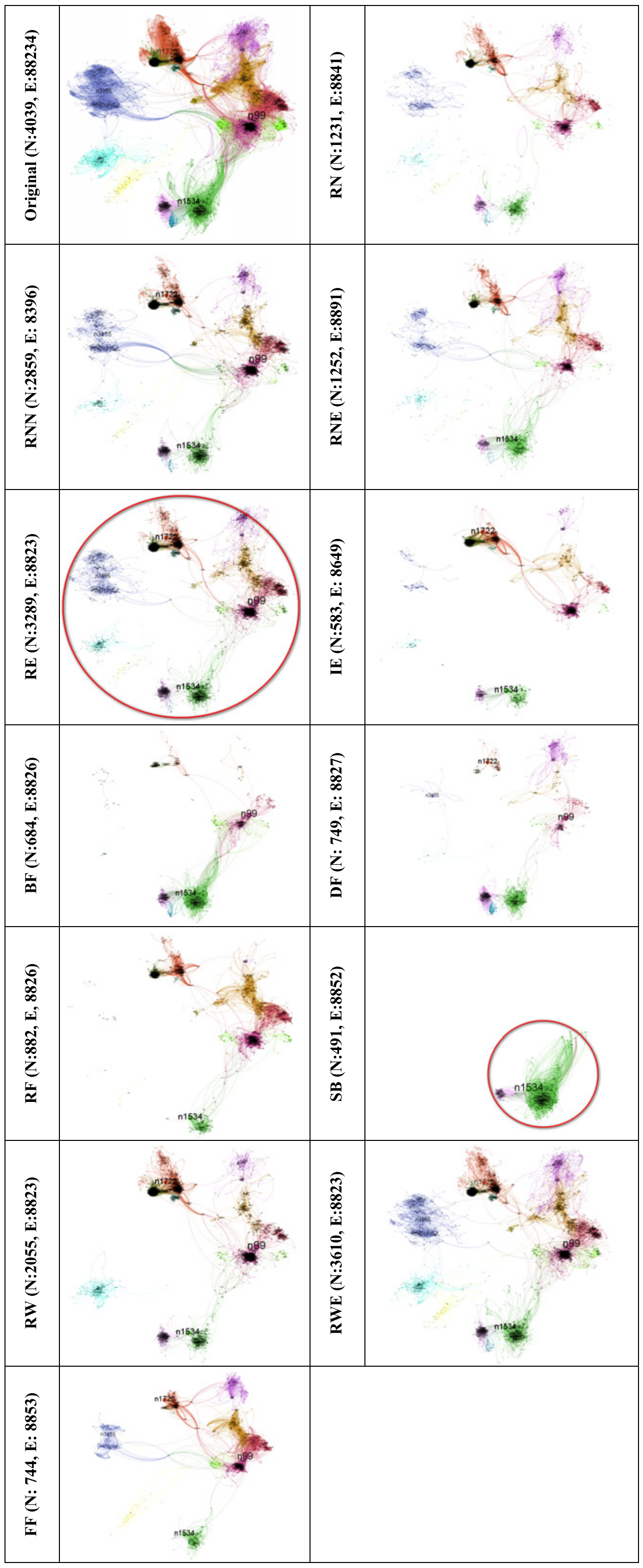


Figure 5. Visual comparison between sampling methods for Facebook graph data (undirected graph) with sampling rate 10% on edges. Red Circles in RE and SB show spatial coverage area of sampling results.

edge-related sampling methods (e.g., random edge) are better than node sampling and traversal-based sampling when the sampling rate is small. For example, random edge sampling in Figs. 4 and 5 is able to preserve most clusters at 10% sampling rate while random node sampling cannot. The reason is that edge-related sampling methods are biased towards high-degree vertices. They are more likely than node sampling to sample large clusters in a graph.

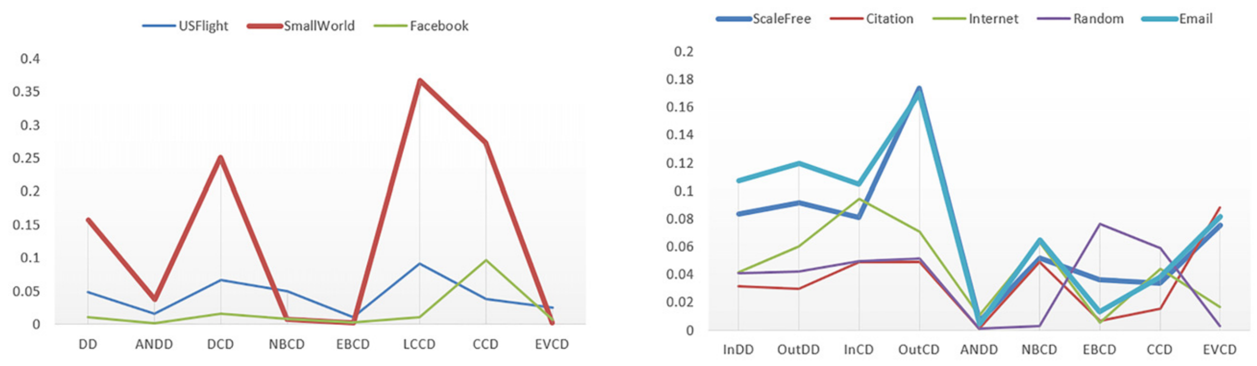
Summary of Good Sampling Methods for each Graph Property
We hope that the sampling results can guide users to choose a suitable sampling method for their applications. Several factors need to be considered, including graph type, graph properties, sampling efficiency, and visual requirements in sampling results. In Fig. 2, we list the top three sampling methods for each graph property and each graph type according to the average SD distance between the sampling results and the original. If the average SD values are greater than 0.1, we mark the sampling methods in red. From Table II, appropriate sampling methods can be chosen based on graph types and graph properties.

In Table II, we have several observations. First, for small-world graph, few sampling methods achieve good results in degree centrality and local clustering coefficient. Second, similar graphs, for example, email graph and scale-free graph, share common choices on sampling methods. Third, induced edge sampling and snowball sampling are good candidates on most graph properties for citation graph and internet graph, respectively.

Based on the above analysis, consideration of all factors will allow users to make more informed choices on sampling methods. For example, if the execution time is important, it is more reasonable to use random sampling methods for very large graphs than traversal-based sampling methods. For a particular graph type and graph property, we can refer to the comparison results to get a good sampling method. For example, to preserve node betweenness centrality distribution and degree distribution on large scale-free graph, random edge sampling is a good choice because the sampling results of random edge show good agreement with the original graph for these two graph properties. If a number of factors need to be considered, we have to sort the priority list of these factors first, and then choose appropriate sampling methods.

DISCUSSION AND OBSERVATIONS

We explored twelve sampling methods and applied those sampling methods to random graph, small-world graph, scale-free graph, and real-world graph. The graph data range from 235 to 265,214 vertices and from 1,297 to 421,578 edges. Eight undirected graph properties and nine directed graph properties are used to evaluate those sampling methods. Our visual and statistical benchmark evaluates sampling methods for their effectiveness in preserving both the quantitative statistical properties and qualitative visual properties of the original graph. The initial analysis indicates that the ranking of these graph sampling methods is dependent on a list of factors, including graph type, desired statistical property, sampling efficiency, and visual requirements. For example, in a time-sensitive task, users will need to consider



(a) Summarized sampling results for undirected graph (b) Summarized sampling results for directed graph

Figure 6. A summary of Fig. 2 by averaging the results from different sampling methods. (a) Summarized sampling results for undirected graph. (b) Summarized sampling results for directed graph. The vertical axis stands for the average SD values, horizontal axis represents average graph properties for all sampling results, and each line shows the results of one dataset.

the computation time for sampling. If one graph property is particularly important in sampling results, users could choose sampling methods according to their rank in Table II.

Furthermore, the visual comparison of the sampling methods gives users an intuitive understanding of the differences among them. The consistent graph layout in the benchmark facilitates the visual comparison and identification of features for each sampling method. In addition, two visual comparison criteria are defined to help users compare sampling methods.

Finally, the results provide insight into the effectiveness of each sampling method in preserving statistical properties. Graph type, graph properties, sampling efficiency, and visual requirements in sampling results are the four key factors when choose sampling methods. The result could help users choose which method to use for a particular application.

ACKNOWLEDGMENTS

This work has been supported by the U.S. Department of Defense. The Pacific Northwest National Laboratory is managed for the U.S. Department of Energy by Battelle under Contract DE-AC05-76RL01830.

REFERENCES

[1] E. A. Lopez-Rojas, "Social network analysis in the dataset US air 97 with pajek," LiU 1–11 (2011). Q.2
[2] D. Rafiei and S. Curial, "Effectively visualizing large networks through sampling," *Proc. IEEE Visualization Conf.* (IEEE, Piscataway, NJ, 2005), p. 48.
[3] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," J. Comput. Graph. Stat. **15**, 265–286 (2006).
[4] D. D. Heckathorn, "Respondent-driven sampling: a new approach to the study of hidden populations," Soc. Probl. **44**, 174–199 (1997).
[5] P. Hu and W. Lau, "A survey and taxonomy of graph sampling," *arXiv.org*, pp. 1–34, 2013.
[6] J. Leskovec and C. Faloutsos, "Sampling from large graphs," *Proc. 12th ACM SIGKDD Int'l. Conf. Knowl. Discov. Data Min.* (2006), pp. 631–636. Q.3
[7] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: from static to streaming graphs," Tkdd **V** (2013).
[8] J. Leskovec and C. Faloutsos, "Sampling from large graphs," *Proc. 12th ACM SIGKDD Int'l. Conf. on Knowledge Discovery and Data Mining—KDD '06* (2006), p. 631.
[9] N. K. Ahmed, F. Berchmans, J. Neville, and R. Kompella, "Time-based sampling of social network activity graphs," *Proc. Eighth Work. Min. Learn. with Graphs—MLG '10* (2010), pp. 1–9.
[10] C. Hübler, H. P. Kriegel, K. Borgwardt, and Z. Ghahramani, "Metropolis algorithms for representative subgraph sampling," *Proc.—IEEE Int'l. Conf. Data Mining, ICDM* (IEEE, Piscataway, NJ, 2008), pp. 283–292.
[11] M. Kurant, A. Markopoulou, and P. Thiran, "Towards unbiased BFS sampling," IEEE J. Sel. Areas Commun. **29**, 1799–1809 (2011).
[12] P. Ebbes, Z. Huang, and A. Rangaswamy, "Sampling of large-scale social networks: Insights from simulated networks," *8th Annu. Work. Inf. Technol. Syst.* (2008).
[13] S. Yoon, S. Lee, S. H. Yook, and Y. Kim, "Statistical properties of sampled networks by random walks," Phys. Rev. E **75** (2007).
[14] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas, "Large graph clustering using DCT-based graph clustering," *Computational Intelligence in Big Data (CIBD), 2014 IEEE Symposium* (IEEE, Piscataway, NJ, 2014), pp. 1–4.
[15] J. Díaz, J. Petit, and M. Serna, "A survey of graph layout problems," ACM Computing Surveys **34**, 313–356 (2002).
[16] J. Q. Walker II, "A node-positioning algorithm for general trees," Softw.-Pract. Exp. **20**, 685–705 (1990).
[17] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," ACM Trans. Graph. **11**, 92–99 (1992).
[18] P. Eades, M. Houle, and R. Webber, "Finding the best viewpoints for three-dimensional graph drawings," Graph Draw. **1353**, 87–98 (1997).
[19] T. Munzner, "H3: Laying out large directed graphs in 3D hyperbolic space," *IEEE Symp. Inf. Vis.* (IEEE, Piscataway, NJ, 1997), pp. 2–10. Q.4
[20] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," Soft.-Pract. Exp. **21**, 1129–1164 (1991).
[21] A. Y. Wu, M. Garland, and J. Han, "Mining scale-free networks using geodesic clustering," *Proc. 2004 ACM SIGKDD Int'l. Conf. Knowl. Discov. Data Min.* (2004), pp. 719–724.
[22] C. Muelder and M. Kwan-Liu, "Rapid graph layout using space filling curves," IEEE Trans. Vis. Comput. Graphics **14**, 1301–1308 (2008).
[23] P. Hage and F. Harary, "Eccentricity and centrality in networks," Soc. Networks **17**, 57–63 (1995).
[24] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," Proc. Natl. Acad. Sci. USA **101**, 3747–3752 (2004).
[25] U. Brandes, "A faster algorithm for betweenness centrality*," J. Math. Sociol. **25**, 163–177 (2001).
[26] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. election," *Proc. 3rd Int'l. Work. Link Discov.—LinkKDD '05* (2005), pp. 36–43.
[27] M. E. J. Newman, "The structure and function of complex networks," SIAM Rev. **45**, 167–256 (2003).

[28] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," Nature **393**, 440–442 (1998).

[29] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," Soc. Networks **32**, 245–251 (2010).

[30] A. N. Langville and C. D. Meyer, "A survey of eigenvector methods for web information retrieval," SIAM Rev. **47**, 135–161 (2005).

[31] S. U. Pillai, T. Suel, and S. Cha, "The Perron-Frobenius theorem," IEEE Signal Process. Mag. **22**, 62–75 (2005).

[32] T. P. Peixoto, "The graph-tool python library," Figshare (2014).

Q.5 [33] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "Sampling techniques for large, dynamic graphs," *Proc. IEEE INFOCOM 2006 25th IEEE Int'l. Conf. Comput. Commun.* (IEEE, Piscataway, NJ, 2006).

[34] B. Ribeiro and D. Towsley, *Estimating and Sampling Graphs with Multidimensional Random Walks* (2010), pp. 390–403.

[35] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," *Proc. 7th Python in Science Conf. (SciPy 2008)* (2008), Vol. 836, pp. 11–15.

[36] P. Erdös and A. Rényi, "On the evolution of random graphs," Publ. Math. Inst. Hungar. Acad. Sci **5**, 17–61 (1960).

[37] A.-L. Barabási and E. Bonabeau, "Scale—free networks," Sci. Am. 60–69 (2003).

[38] L. Lee, "On the Effectiveness of the Skew Divergence for Statistical Language Analysis," *AISTATS Artificial Intell. Stat.* (2001), pp. 65–72.

[39] N. Ahmed, J. Neville, and R. R. Kompella, "Network sampling via edge-based node selection with graph induction," 2011.

[40] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," *Third Int'l. AAAI Conf. Weblogs Soc. Media* (2009), pp. 361–362.

Queries for IS&T paper 0271

Journal:	**JIST**
Author:	**F. Zhang, S. Zhang, and P. C. Wong**
Short title:	**Graph sampling for visual analytics**

Page 4

Query 1:
Au: A closing bracket deleted. Please confirm.

Page 11

Query 2:
Au: Ref. [1]: Please provide volume number if possible.

Page 11

Query 3:
Au: Refs. [6, 8, 9, 12, 21, 26, 34, 35, 38, 40]: Please provide place and publisher name if possible.

Page 11

Query 4:
Au: Ref. [19]: Please confirm whether the given reference here is OK.

Page 12

Query 5:
Au: Ref. [39]: Please provide more details if possible.