

A Visual Evaluation Study of Graph Sampling Techniques

Fangyan Zhang¹, Song Zhang¹, Pak Chung Wong² Hugh Medal¹ Linkan Bian¹ J. Edward Swan II¹ T.J. Jankun-Kelly¹

¹Mississippi State University, USA

²Pacific Northwest National Laboratory, USA

Abstract

We evaluate a dozen prevailing graph-sampling techniques with an ultimate goal to better visualize and understand big and complex graphs that exhibit different properties and structures. The evaluation uses eight benchmark datasets with four different graph types collected from Stanford Network Analysis Platform and NetworkX to give a comprehensive comparison of various types of graphs. The study provides a practical guideline for visualizing big graphs of different sizes and structures. The paper discusses results and important observations from the study.

Keywords: Big graphs, Graph sampling, Graph properties, Graph drawing, Visualization, Visual analytics.

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 aims 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.

Related Work

Existing graph sampling algorithms can be classified into three types: node sampling, edge sampling, and traversal-based sampling [5] [6] [7] [8]. 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 [9] 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 [10], random walk [11] [12], and snowball sampling [11] 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 [13]. Numerous techniques have been proposed to approach graph visualization, such as clustering [13], sampling [2], and special layout [14]. 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 [15] [16], 3D layout [17], hyperbolic layout [18], and force-directed layout [19]. Clustering reduces vertex and edge

overlap by replacing clusters with vertices. Two graph clustering techniques are vertex clustering [20] and edge clustering [21].

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 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.

The degree distributions of the three graph models and a real social network graph are illustrated in Figure 1. 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.

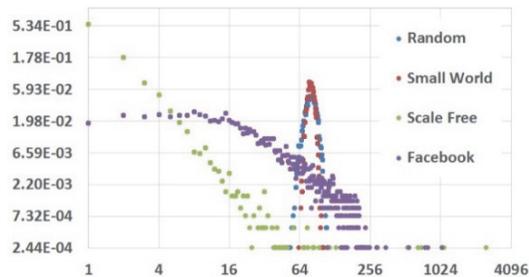


Figure 1. (a) Degree distribution of random graph model (blue), small-world graph model (red), scale-free graph model (green), and real social graph (magenta) with Log 2 axis.

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 is the probability distribution of a vertex's degree. A vertex's degree is the number of edges connected to that vertex. 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. Average neighbor degree returns the average degree of the neighborhood of each vertex [22]. Betweenness centrality [23] [24] 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. Clustering coefficient is a measure of how vertices in a graph cluster together. It includes the global clustering coefficient and the local clustering coefficient. Closeness centrality [25] 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. Eigenvector centrality [26] 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.

In this paper, we use Graph-tool [27] 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

In random node sampling, vertices are sampled randomly and uniformly. A subgraph is created from sampled nodes and existing edges of original graph. If edges that are incident to these vertices are uniformly included in the sample graph, then this approach is random node-edge sampling. If all the edges connected to these vertices in the original graph are sampled into subgraph, then this method is called random node-neighbor sampling.

Edge Sampling

In random edge sampling, edges are sampled randomly and uniformly, and then a subgraph is created from those edges. 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. We implement totally induced edge sampling in the paper.

Traversal-based sampling

Traversal-based sampling uses topology information to sample a subgraph. Many types of traversal-based sampling methods have been proposed. For example, breadth-first sampling [5] [10] is induced from the graph traversal algorithm breadth-first search. It begins with a random vertex and visits its neighbors iteratively. Depth-first sampling [28] is derived from the depth-first search algorithm. Random-first sampling [5] is similar to breadth-first sampling and depth-first sampling except that vertices are selected randomly in each iteration. Snowball sampling [11] is another traversal-based sampling. First, it randomly selects a starting vertex 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 [28] 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 [29] [30] and multiple independent random walkers are proposed based on the

classic random walk sampling method. Forest fire sampling [7] can be regarded as a probabilistic version of breadth-first sampling. Neighbors are chosen to be added to the subgraph with probability p . (p is set to 0.5 in this paper.)

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 paper 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 [31] via corresponding graph models.

Table 1 summarizes the properties of the eight datasets used for the comparison study.

Table 1: Eight test datasets and their properties

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

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 [32]. 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 distributions should be smoothed before computing KL divergence. We use the same strategy as the paper [32] [33] 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 paper, and α is set to 0.99 as in the paper [32] [33]. Previous work [32] has proven that SD performs better on non-smoothed distributions to approximate KL divergence.

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 [34] 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 social graph data of visual comparison are provided in the paper. In visualization, 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 paper, we conduct experiments using a sampling rate ranging from 10 to 50 percent with a 10 percent 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 of space limitations, we do not list all individual results in this paper.

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 shows the visual comparison between sampling methods using a 10% sampling results for the Facebook graph data. In this visualizations, vertex label size is positively proportional to its degree.

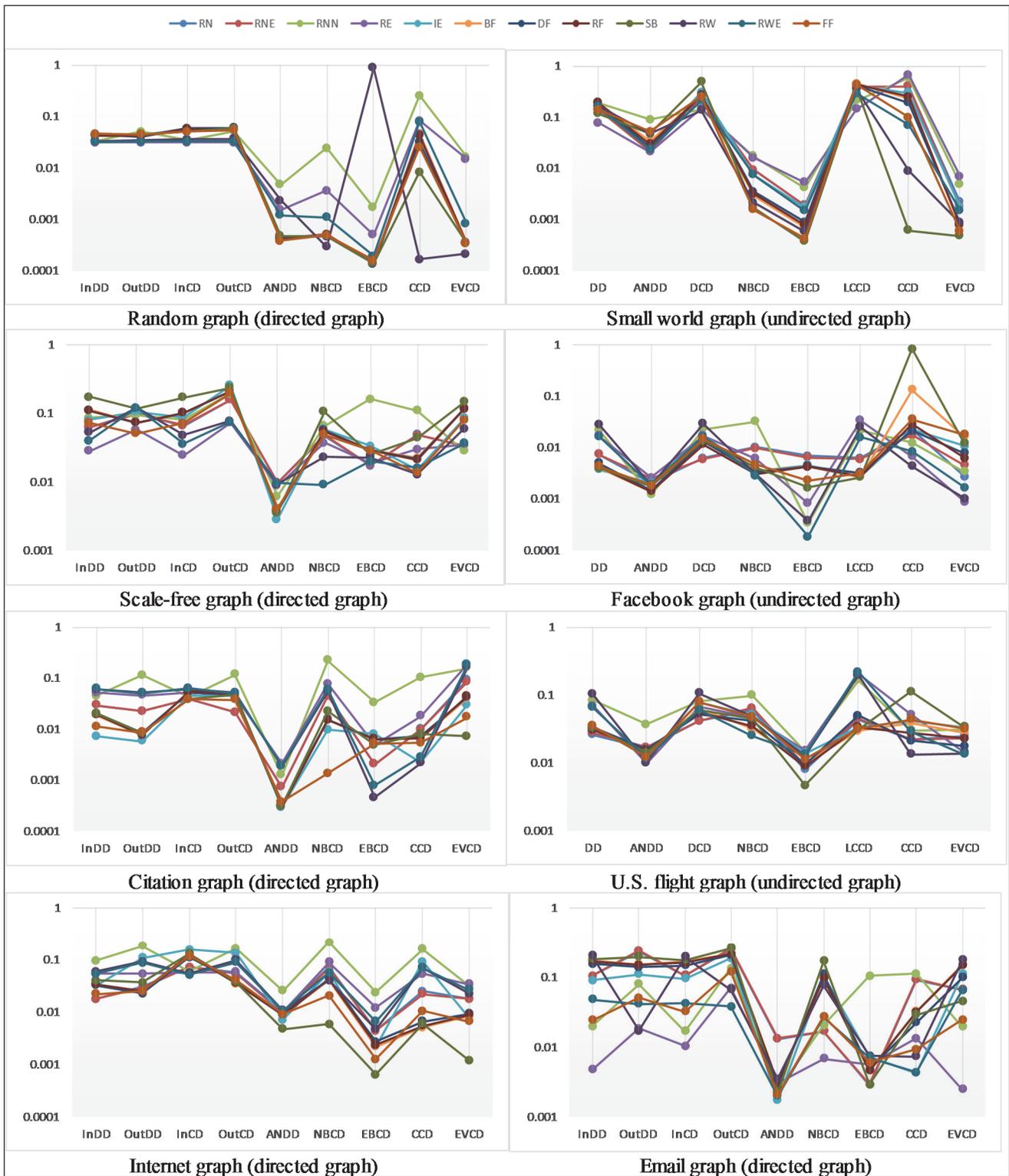
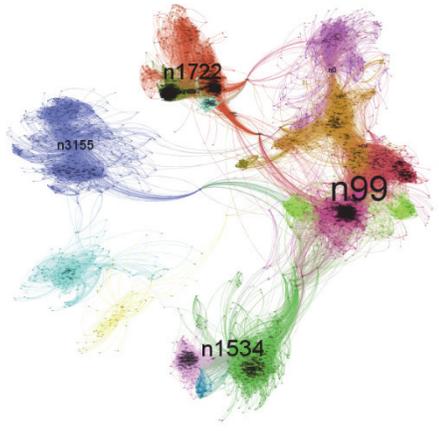
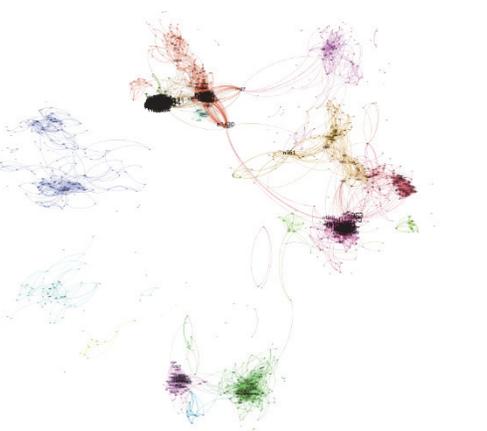
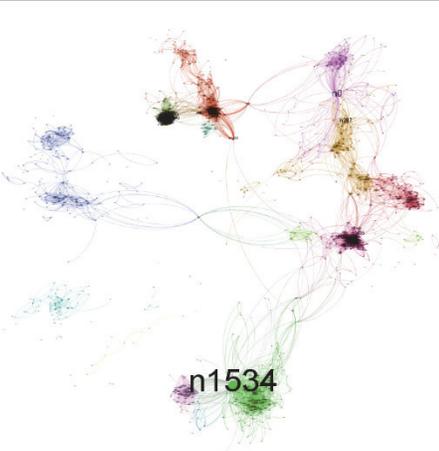
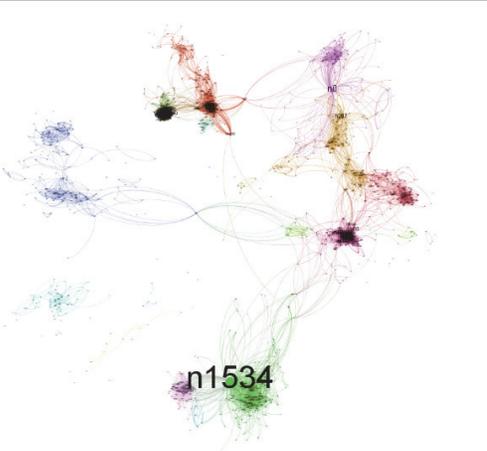
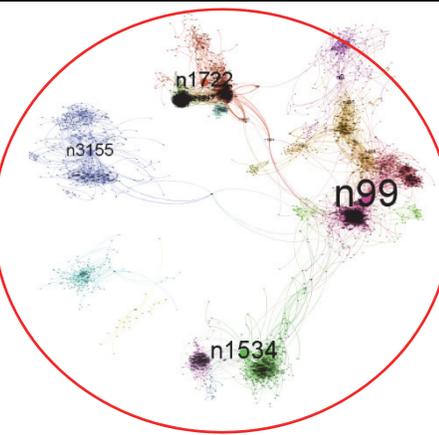
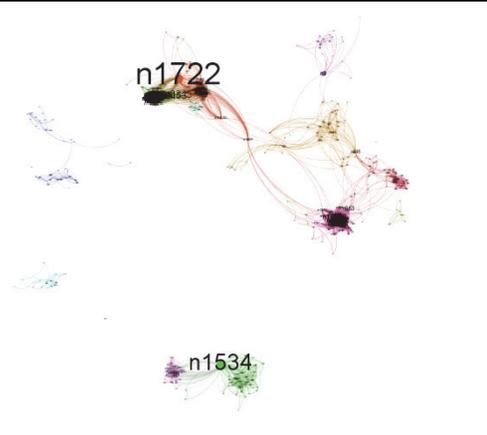


Figure 2: Average result of the statistical comparisons between sampling methods with 10 to 50 percent sampling rates. The vertical axis is SD values, horizontal axis represents graph properties, and lines are sampling methods

<p>Original (N:235, E:1297)</p>		<p>RN (N:1231, E:8841)</p>	
<p>RNN (N:2859, E: 8396)</p>		<p>RNE (N:1252, E:8891)</p>	
<p>RE (N:3289, E:8823)</p>		<p>IE (N:583, E: 8649)</p>	

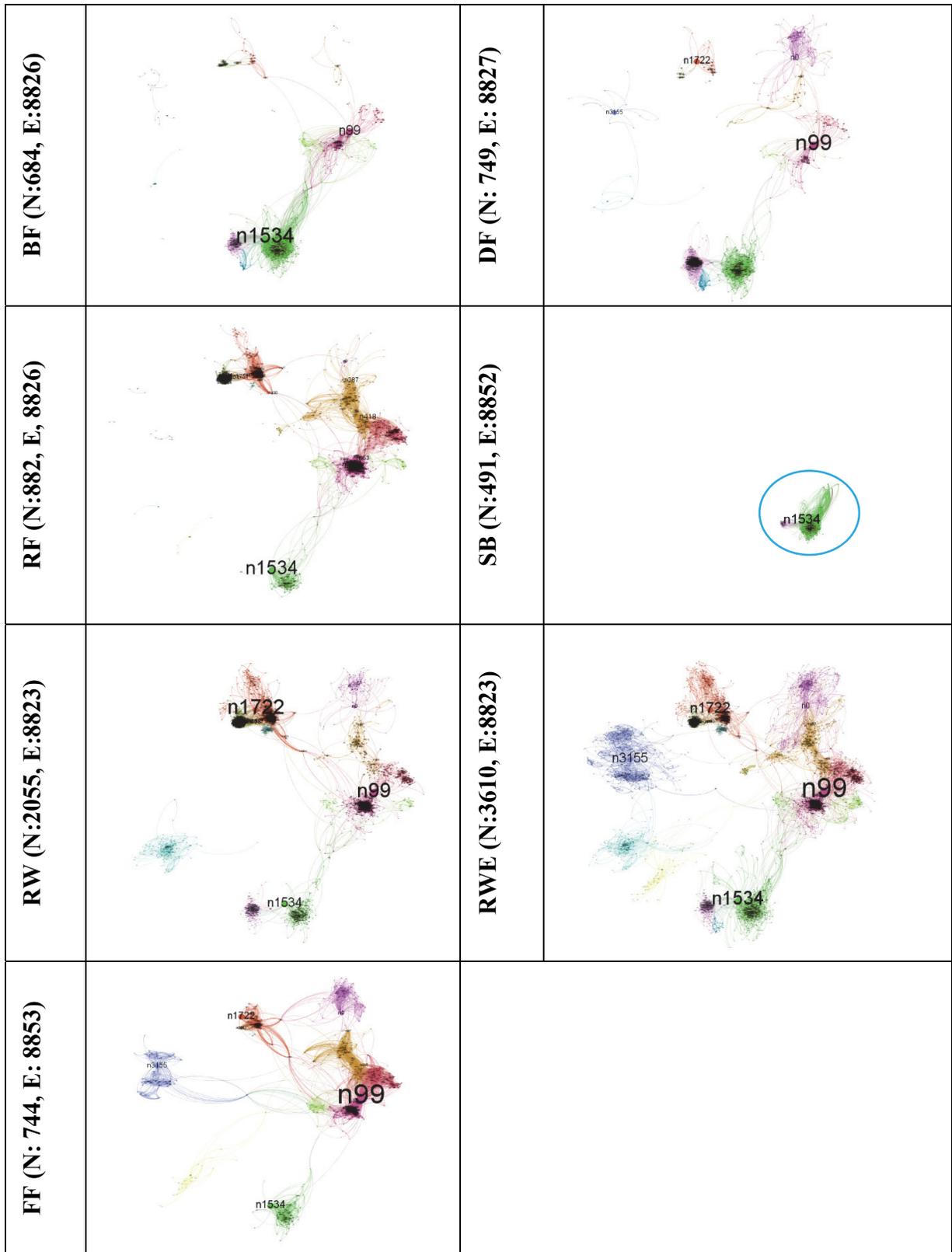


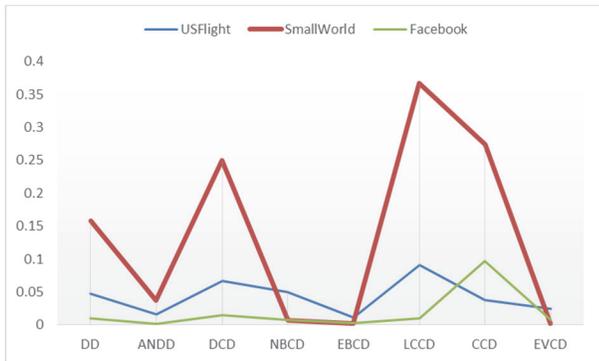
Figure 3: Visual comparison between sampling methods for Facebook graph data (undirected graph) with sampling rate 10% on edges. Red Circles in RE and light blue circle in SB show spatial coverage area of sampling results.

Analysis

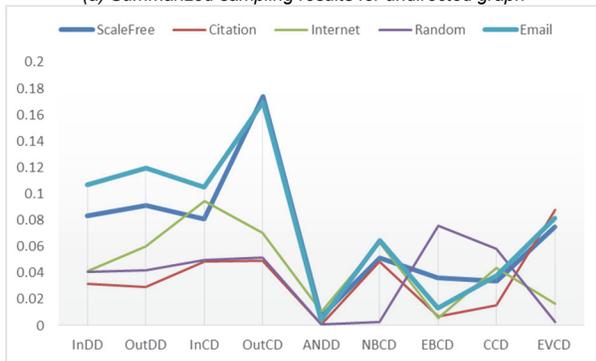
The benchmark allows us to compare sampling methods quantitatively and qualitatively in several aspects. First, the sampling experiment on eight graph datasets help us to analyze the sampling results for different graph data types. Second, for each graph property, we observe the average SD divergence between the sampling results and the original graph, and then determine whether the sampling methods have stable performance for that graph property. Finally, we conduct visual comparison between these sampling results. We list the observations from the results below.

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 4 is summarized from Figure 2 by averaging the SD values of all sampling method. Because directed graph and undirected graph have different properties, we summarize directed graphs (Figure 4 (a)) and undirected graphs (Figure 4 (b)) separately. From Figure 4 (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). From these observations, it is obvious that graph type has significant influence on sampling results and should be considered when sampling graphs.



(a) Summarized sampling results for undirected graph



(b) Summarized sampling results for directed graph

Figure 4: A summary of Figure 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.

Comparison between graph properties

We analyze how sampling methods perform on each graph property. As shown in Figure 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 in degree distribution, out degree distribution, in degree centrality distribution, and out degree centrality distribution. These observations indicate that graph property should be considered when choosing sampling methods in application.

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 criterion of visual comparison between sampling results. A 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 Figure 3, random edge sampling results (red circle) of Facebook 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 (light blue circle) in Figure 3 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 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 Figure 3 is able to preserve most clusters at a 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.

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 several factors, including graph type, desired statistical property, sampling efficiency, and visual requirements. Consideration of all factors will allow users to make more informed choices on sampling methods. If one graph property is particularly important in sampling results, users could choose sampling methods according to their rank in Figure 2. 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.

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, and visual requirements in sampling results are the three key factors when choose sampling methods. The result could help users 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*, no. 1, pp. 1–11, 2011.
- [2] D. Rafiei and S. Curial, "Effectively visualizing large networks through sampling," in *Proceedings of the IEEE Visualization Conference*, 2005, p. 48.
- [3] H. Zou, T. Hastie, and R. Tibshirani, "Sparse Principal Component Analysis," *J. Comput. Graph. Stat.*, vol. 15, no. 2, pp. 265–286, 2006.
- [4] D. D. Heckathorn, "Respondent-driven sampling: A new approach to the study of hidden populations," *Soc. Probl.*, vol. 44, no. 2, pp. 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. Conf. Knowl. Discov. data Min.*, pp. 631–636, 2006.
- [7] N. K. Ahmed, J. Neville, and R. Kompella, "Network Sampling: From Static to Streaming Graphs," *Tkdd*, vol. V, no. 212, 2013.
- [8] 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*, pp. 1–9, 2010.
- [9] C. Hübler, H. P. Kriegel, K. Borgwardt, and Z. Ghahramani, "Metropolis algorithms for representative subgraph sampling," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, no. 1, pp. 283–292, 2008.
- [10] M. Kurant, A. Markopoulou, and P. Thiran, "Towards unbiased BFS sampling," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1799–1809, 2011.
- [11] P. Ebbes, Z. Huang, and A. Rangaswamy, "Sampling of large-scale social networks: Insights from simulated networks," *8th Annu. Work. Inf. Technol. Syst.*, 2008.
- [12] S. Yoon, S. Lee, S. H. Yook, and Y. Kim, "Statistical properties of sampled networks by random walks," *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 75, no. 4, 2007.
- [13] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas, "Large graph clustering using DCT-based graph clustering," in *Computational Intelligence in Big Data (CIBD), 2014 IEEE Symposium on*, 2014, pp. 1–4.
- [14] J. Diaz, J. Petit, and M. Serna, "A survey of graph layout problems," *ACM Computing Surveys*, vol. 34, pp. 313–356, 2002.
- [15] J. Q. Walker II, "A node-positioning algorithm for general trees," *Softw. -- Pract. Exp.*, vol. 20, pp. 685–705, 1990.
- [16] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Trans. Graph.*, vol. 11, no. 1, pp. 92–99, 1992.
- [17] P. Eades, M. Houle, and R. Webber, "Finding the best viewpoints for three-dimensional graph drawings," *Graph Draw.*, vol. 1353, pp. 87–98, 1997.
- [18] T. Munzner, "H3: Laying out large directed graphs in 3D hyperbolic space," *1997 IEEE Symp. Inf. Vis.*, pp. 2–10, 1997.
- [19] T. Fruchterman and E. Reingold, "Graph drawing by force directed placement," *Softw. Pract. Exp.*, vol. 21, no. NOVEMBER, pp. 1129–1164, 1991.
- [20] A. Y. Wu, M. Garland, and J. Han, "Mining scale-free networks using geodesic clustering," *Proc. 2004 ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, pp. 719–724, 2004.
- [21] C. Muelder and M. Kwan-Liu, "Rapid graph layout using space filling curves," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 6, pp. 1301–1308, 2008.
- [22] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 101, no. 11, pp. 3747–3752, 2004.
- [23] U. Brandes, "A faster algorithm for betweenness centrality*," *J. Math. Sociol.*, vol. 25, no. 2, pp. 163–177, 2001.
- [24] L. A. Adamic and N. Glance, "The Political Blogosphere and the 2004 U.S. Election," *Proc. 3rd Int. Work. Link Discov. - LinkKDD '05*, pp. 36–43, 2005.
- [25] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Soc. Networks*, vol. 32, no. 3, pp. 245–251, 2010.
- [26] A. N. Langville and C. D. Meyer, "A Survey of Eigenvector Methods for Web Information Retrieval," *SIAM Rev.*, vol. 47, no. 1, pp. 135–161, 2005.
- [27] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014.
- [28] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "Sampling Techniques for Large, Dynamic Graphs," *Proc. IEEE INFOCOM 2006. 25TH IEEE Int. Conf. Comput. Commun.*, 2006.
- [29] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, 2006, p. 631.
- [30] B. Ribeiro and D. Towsley, "Estimating and Sampling Graphs with Multidimensional Random Walks," pp. 390–403, 2010.
- [31] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, 2008, vol. 836, pp. 11–15.
- [32] L. Lee, "On the Effectiveness of the Skew Divergence for Statistical Language Analysis," *AISTATS (Artificial Intell. Stat.)*, pp. 65–72, 2001.
- [33] N. Ahmed, J. Neville, and R. R. Kompella, "Network sampling via edge-based node selection with graph induction," 2011.
- [34] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An Open Source Software for Exploring and Manipulating Networks," *Third Int. AAAI Conf. Weblogs Soc. Media*, pp. 361–362, 2009.