https://doi.org/10.2352/ISSN.2470-1173.2018.01.VDA-379

Implementation and Evaluation of Distributed Graph Sampling Methods with Spark

Fangyan Zhang, Song Zhang, Christopher Lightsey
Mississippi State University, Mississippi State, MS. USA

Abstract

The growth of graph size has created new problems in graph visualization and graph analysis. To solve the problem, several graph sampling techniques have been proposed dedicated to obtaining a representative subgraph from a complex network. While prior research indicates that sampling on a large-scale graph is not an easy task, especially for topology-based sampling methods (e.g. breadth first sampling). Topology-based sampling methods can produce a more accurate subgraph than node sampling and edge sampling in preserving statistical graph properties. In this paper, we propose three types of distributed sampling algorithms and develop a sampling package on Spark. To evaluate the effectiveness of these distributed sampling techniques, we apply them to three graph datasets and compare them with traditional/non-distributed sampling approaches. The results show that (1) our distributed sampling approaches are as reliable as the non-distributed sampling techniques, and (2) they are a great improvement in sampling efficiency, especially for topology-based sampling. In addition, (3) the distributed architecture of these algorithms causes them to have horizontal scalability.

Keywords: Graph sampling, Graph properties, Sampling Method, Visualization, Visual analytics

Introduction

Graph, as a standard data model, is widely used in various application domains to represent entities and their relationships, such as social network [1], Internet network [2], citation network [3], and biological network [4]. In the past few years, graphs have been growing explosively in numerous research fields. It was reported that Facebook had 1.6 billion active users during a month in late 2015 [5]. There are 3 billion emails created every day worldwide [6].

Effectively understanding huge graphs becomes more and more difficult with the increase of graph size. It is often imperative to sample a small subgraph from a large-scale graph. This is necessary for several reasons. First, sometimes visualizing a large-scale graph seems to be impossible because of limits of screen size. A good small representative subgraph can help us understand the topological structure of the original graph. The findings from the subgraph allow us to make more accurate estimations about graph features on the original graph. Second, graph property calculation is an essential approach to analyze graph features, but the calculation of graph properties on a large-scale graph can be costly or unaffordable. If an accurate estimation graph properties can be made from a representative of original graph, then substantial computational cost may be avoided. Third, for some large-scale graphs, it is not practical to obtain complete datasets because of graph size or other various reasons. For example, total Facebook graph has more than 250TB of HTML data [7]. It is impractical to obtain complete data considering the time-cost. Graph sampling techniques allow us to get a representative sample while preserving features in original graph.

Given a graph $G = (V, E)$, V and E represent vertices and edges respectively. Sampling techniques try to get a subgraph $G_s = (V_s, E_s)$, where $V_s \subset V$, $E_s \subset E$. Prior sampling approaches explore networks from two aspects [6]: making accurate estimations of the original graph [8] [9]; selecting a representative sample whose topology is similar to the original [10].

Many sampling techniques have been proposed aiming to sample a representative subgraph from the original graph. Those sampling methods can be classified in three categories: node sampling [11] [12], edge sampling [12] [13] and topology-based sampling [14] [15]. Unfortunately, many of these non-distributed sampling methods, especially topology-based sampling, are not practical for large-scale graphs (more than 1 million vertices) because of efficiency issue or memory issue. For example, in our previous research [16], we found that sampling on an Email graph (vertices: 265, 214; edges: 420, 045) took more than 24 hours while using topology-based sampling. Most prior sampling research only worked on graphs with less than 1 million vertices [11] [6] [17]. With the increase of graph size, scalable sampling methods are in urgent demand in graph research.

In this paper, we aim to design and develop a number of scalable topology-based sampling methods based on a MapReduce framework-Spark [18], and make comprehensive evaluations between non-distributed sampling approaches and our new methods. For completeness, scalable node sampling and scalable edge sampling techniques are also included in our work. These scalable sampling techniques can greatly improve sampling efficiency without losing sampling accuracy. To improve the reliability of these distributed sampling algorithms, we apply both non-distributed sampling methods and new sampling methods to three datasets, and make statistical and visual comparison between them.

The main contributions of our work are as follows:

- We designed and developed nine scalable graph sampling methods based on Spark.
- We made visual and statistical comparison between non-distributed sampling methods and distributed graph sampling methods.
- We implemented a large-scale graph sampling package which can be incorporated into GraphX for graph research.
- We analyzed the nine distributed graph sampling methods from visual and statistical perspectives, and summarized their merits compared to non-distributed sampling techniques.

Related Work

Graph sampling has been an interesting field within graph research for many years. Many sampling techniques have been proposed in last few decades. Existing graph sampling algorithms can be classified into three categories [11] [6]: node sampling, edge sampling, and topology-based sampling.

Node sampling is a simple sampling method which creates representative graph by sampling nodes independently and uniformly. For example, in random node sampling [11], vertices are sampled randomly and uniformly, creating a subgraph from the original graph. The edges between the sampled vertices in the original graph are included the sample graph as well. Sometimes, node sampling also considers neighbors of the sampled vertices, for instance, random node-edge sampling [19] and random node-neighbor sampling [11]. In random node-edge sampling, when vertices are uniformly sampled, edges that are incident to these vertices are also uniformly sampled in the sample graph. In random node-neighbor sampling, all the edges that are connected to these vertices in the original graph are sampled into subgraphs. In some cases, node sampling methods integrate topology-based sampling in order to use graph topology information, such as random walk sampling [20]. 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. Random edge sampling [13] is one typical edge sampling in which edges are sampled randomly and uniformly, and then a subgraph is created from those edges. Induced edge sampling [12] is another edge sampling method, which 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 paper, we implement totally induced edge sampling.

Topology-based sampling is regarded as the state-of-art sampling methods because they have good ability to preserve graph properties [6]. Graph traversal algorithms are often used in these sampling methods. For example, breadth-first sampling [17] creates a subgraph by using breadth first search algorithm. 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 visited vertices and those edges that are connected between those sampled vertices in the original graph. Forest fire sampling [6] [12] 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 $p_f/(1-p_f)$. Random walk sampling 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 [13] and multiple independent random walkers [19] are proposed based on the classic random walk sampling method. Random walk with escape or jump sampling is more random than random walk sampling since the current walker vertex jumps to another random vertex with probability *p*.

Spark [18] is an in-memory distributed computing framework that manipulates datasets in memory across distributed machines, which is different from Hadoop MapReduce [21] in that it keeps intermediate data in memory instead of storing it on disk. This strategy makes Spark run up to 100 times faster than Hadoop MapReduce [18]. It allows us create iterative algorithms efficiently because it offers us a MapReduce environment. GraphX [22], a Spark library, takes the advantage of data-parallel and graph-parallel systems in Spark framework. Within GraphX, several graph property calculation algorithms are implemented, such as PageRank, connected component, shortest paths, etc. In this paper, we design distributed sampling methods and implement them on Spark.

Distributed Graph Sampling Methods

In this section, we present three types of novel graph sampling techniques: 1) distributed node sampling, 2) distributed edge sampling, and 3) distributed topology-based sampling. We make analytical comparisons between non-distributed sampling methods vs. our distributed methods. The graph is partitioned by GraphX using the strategy of *Random Vertex Cut* and distributed into multiple machines. The node sampling includes random node sampling (RN), random node edge sampling (RNE), and random node neighbor sampling (RNN). The edge sampling includes random edge sampling (RE), induced edge sampling (IE), and random hybrid sampling (RH). The topology-based sampling includes breadth first sampling (BF), snowball sampling (SB) and forest fire sampling (FF).

Distributed Node Sampling

Node sampling constructs subgraphs based on sampling vertices from the original graph. In random node sampling, vertices are sampled randomly and uniformly.

Given a graph $G = (V, E)$, where *V* and *E* are vertices and edges of *G*. Let *G*'s degree sequence be $\{1, 2\ldots i\ldots k\}$, the number of degree *i* is $N(i)$. If node sampling rate is *r*, then the expected nodes in non-distributed random node sampling can be represented as:

$$E_{non\text{-}distributed}(NS(r)) = N(1)*r + N(2)*r + \ldots + N(k)*r = r*\sum_{i=1}^{k} N(i) \quad (1)$$

For distributed node sampling algorithms, since graphs are distributed into multiple machines, each machine contains one or more partitions. In distributed node sampling, nodes are sampled from each partition independently. If a graph is distributed into *n* partitions, then the expected nodes in one partition can be represented as:

$E_p(NS(r)) = N_p(1)*r + N_p(2)*r + \ldots + N_p(t)*r$, where *t* is maximum degree in partition *p*, and $t <= k$. If the graph has *n* partitions, then for overall partitions:

$$E_{distributed}(NS(r)) = E_1(NS(r)) + E_2(NS(r)) + \ldots + E_n(NS(r)) = \sum_{t=1}^{n} E_t(NS(r)),$$

Since $N_1(i) + N_2(i) + \ldots + N_n(i) = N(i)$, then

$$E_{distributed}(NS(r)) = \sum_{t=1}^{n} E_t(NS(r)) = N(1)*r + N(2)*r + \ldots + N(k)*r = \sum_{i=1}^{k}\sum_{t=1}^{n} N_t(i)*r = r*\sum_{i=1}^{k} N(i) \quad (2)$$

From equation (1) and (2), theoretically, both distributed node sampling and non-distributed node sampling should have the similar results.

Distributed Edge Sampling

Edge sampling builds a subgraph by randomly sampling edges. If $|E|$ is the number of edges, sampling rate is *r*, then there should be $r*|E|$ edges sampled in sampling results. Suppose the degree original graph is $\{1, 2, \ldots i\ldots k\}$, the number of degrees *i* is $N(i)$, the probability of one node with certain degree falling into sample is

$[1-(1-\frac{i}{|E|})]^{r|E|}$. Thus, the expected nodes in random edge sampling can be described as:

$$E\ (ES(r)) = N\ (1) * [1-(1-\frac{1}{|E|})]^{r|E|} + N\ (2) * [1-(1-\frac{2}{|E|})]^{r|E|} + \ldots + N\ (k) * [1-(1-\frac{k}{|E|})]^{r|E|}$$

$$= \sum_{t=1}^{k} N(t) * [1-(1-\frac{t}{|E|})]^{r|E|} \quad (3)$$

Similarly, in distributed edge sampling algorithms, each machine also contains one or more partitions. In each partition, edges are sampled independently. If one graph is distributed into *n* partitions, then the expected nodes in one partition can be represented as:

$E_p\ (ES(r)) = N_p\ (1) * [1-(1-\frac{1}{|E|})]^{r|E|} + N_p\ (2) * [1-(1-\frac{2}{|E|})]^{r|E|} + \ldots + N_p\ (t) * [1-(1-\frac{t}{|E|})]^{r|E|}$, where *t* is maximum degree in partition *p*, and $t <= k$. If graph has *n* partitions, then for overall partitions:

$$E_{distributed}\ (ES(r)) = E_1\ (ES(r)) + E_2\ (ES(r)) + \ldots + E_n\ (ES(r)) = \sum_{t=1}^{n} E_t\ (ES(r)),$$

$$E_{distributed}(ES(r)) = \sum_{t=1}^{n} E_t\ (ES(r)) = N\ (1) * [1-(1-\frac{1}{|E|})]^{r|E|} + N\ (2) * [1-(1-\frac{2}{|E|})]^{r|E|} + \ldots + N\ (k) * [1-(1-\frac{k}{|E|})]^{r|E|} = \sum_{i=1}^{k}\sum_{t=1}^{n} N_t(i) * [1-(1-\frac{i}{|E|})]^{r|E|}$$

$$= \sum_{i=1}^{k} N(i) * [1-(1-\frac{i}{|E|})]^{r|E|} \quad (4)$$

From equation (3) and (4), theoretically, both distributed edge sampling and non-distributed edge sampling should have the similar results as well.

From the distributed random node sampling and distributed random edge sampling, we can easily demonstrate that distributed random node edge sampling, distributed random node neighbor sampling, distributed induced edge sampling, and distributed random hybrid sampling can produce the similar sampling results as its corresponding non-distributed sampling method.

Distributed Topology-based Sampling

Since topology-based sampling creates subgraphs based on the topological information of the original graph, instead of sampling vertices or edges directly, which selects nodes or edges in sequential order. The topological visited indices represents the order that one node or edge enter sampling results. In distributed topology-based sampling, there are two challenges in the sampling process. First, the graph is distributed into multiple machines, it is not easy to create visited index in such a graph because it involves frequent communication between partitions, particularly while sampling on multiple partitions at the same time. Second, it is important to take into consideration that that one graph may have multiple unconnected components. Those components might be partitioned into multiple machines. Which component starts the sampling process has a great influence in sampling results.

To solve the aforementioned challenges, we develop new strategies to implement topology-based sampling on distributed graphs. The topology-based sampling process is divided into two stages: vertex labeling and sampling. Initially, a screening for the quantity of unconnected components the graph contains will be completed. All components have equal probability to act as the seed node, and they are visited in sequential order. We keep a record of the number of vertices for each component during labeling stage. Each vertex has an index number that indicates how many vertices were visited before reaching the current vertex. Thus, the sampling process can be tracked with the label index in the sampling stage. Because our distributed topology-based sampling uses the same principle as non-distributed topology-based sampling, they should produce similar sampling results. Forest fire sampling is a randomized version of breadth first sampling [6], and snowball sampling is similar to breadth first sampling [13]. Here we only provide the pseudocode of distributed breadth first sampling methods.

```
Distributed Breadth First Sampling

Input: Graph G = (V, E), sampling rate $\theta$
Output: Sampled graph $G_s = (V_s, E_s)$
Begin
    // cache sampling rate on each machine
    Broadcast ($\theta$)
    $G_c \leftarrow$ add one vertex attribute (index $\leftarrow$ MaxValue)
    step $\leftarrow 1$
    componentsList $\leftarrow$ components in $G_c$
    Repeat: each component c in $G_c$
        startNode $\leftarrow$ choose a random node in c
        iteratively modify vertex attribute in c (index $\leftarrow$ step)
        step $\leftarrow$ step + 1
    Until termination;
    Repeat: s in 1 to step
        check the number of vertices when index $< s$
    Until the number of vertices is satisfied
    $G_{subgraph} \leftarrow G_c$ ((V, index $< s$), E)
End algorithm
```

Distributed Sampling Algorithms Implementation

The nine distributed sampling methods provided above are implemented using GraphX in Spark [18]. GraphX provides two special versions of resilient distributed datasets (RDD): a VertexRDD and an EdgeRDD, which are used to represent vertex information and edge information in memory or hard disk. The distributed sampling algorithms are written in Scala language and compiled into a JAR file for distribution. This package and source code will be uploaded to GitHub for public access upon the publication of this work.

In our experiments, the distributed graph sampling on large scale graph is done on Shadow (a 322-teraflop Cray CS300-LC cluster supercomputer with Intel® Xeon® E5-2600 v2 processors and Intel® Xeon Phi™ coprocessors), each node has 500GB and 20 processors.

Experimental Evaluation

Here we present our experiments to evaluate distributed sampling methods from visual and statistical perspectives by making comparisons with non-distributed sampling methods on three graph datasets ranging from 88234 edges to 1,806,067,135 edges (described in Table 1). Specifically, we will introduce some graph properties, and statistical and visual comparison techniques used in the evaluation.

Graph Datasets

Table 1: Three test datasets and their properties

Graph Dataset	Graph Type	Model	# Vertices	# Edges
Facebook	Undirected	Real	4,039	88,234
Amazon	Undirected	Real	334,863	925,872
Friendster	Undirected	Real	65,608,366	1,806,067,135

We apply our sampling algorithms on the three datasets: Facebook graph, Amazon graph, and Friendster graph that are collected from Stanford Network Analysis Platform (SNAP) [23]. Facebook, as an anonymized graph using an integer number as user id, was collected from the Facebook app [24]. Amazon was created from the Amazon website based on the relation between items that are co-purchased [25]. Friendster is an on-line gaming network created from a social networking site [25]. These graph datasets are summarized in Table 1.

Graph Metrics

For evaluation of distributed sampling methods, several graph metrics are used to assess the quality of sampling results in statistical comparisons. The comparison between sampling results and original graphs is realized by comparing their graph property distributions in statistics. Here we focus on degree distribution (DD), average neighbor degree distribution (ANDD), triangle distribution (TD), PageRank distribution (PR), and local clustering coefficient distribution (LCCD).

Statistical Comparison

To evaluate distributed sampling methods and non-distributed sampling methods, we appraise their performance by how well the sampled results preserve each of graph properties. A good sampling method should produce a subgraph that approximates 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 [26]. 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 Ahmed [6] and Lee [26] 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 Ahmed [6] and Lee [26]. Previous work [26] has proven that SD has better performance approximating KL divergence on non-smoothed distributions.

Visual Comparison

In addition to making comparisons between sampled subgraphs and the original graph statistically, we also compare them visually by using Gephi [27]. We first draw the original graph and export the decorated graph into a file with vertex decorations preserved (e.g. vertex color, label size, location). When sampling on the decorated graph, vertex attributes are also sampled along with vertices. The layout in the sampling results should have the same layout as in the decorated graph—i.e., the same vertex in all sampled graphs will occupy the same location as in the decorated 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. The vertex attributes, for example, vertex position, color, are preserved in subgraph because we consider that such vertex attributes are significant in visual comparison. In this way, the similarity and difference within or between sampling results can be easily identified.

Because of space limits, only the visual comparison of Facebook graph is provided in this paper.

Results

In our experiment, both non-distributed and distributed sampling methods are applied to Facebook and Amazon graph datasets. For Friendster, since the Friendster graph is too large for non-distributed sampling methods because of unaffordable sampling time cost, we do not apply non-distributed graph sampling to this graph data. We investigate the performance of distributed sampling methods by comparing their abilities to preserve the features of original graph. When sampling these graphs, we set the sampling rate at 15% and 25% based on the number of vertices for each graph. For each sampling rate, there are 5 different runs carried out in this experiment. We take the average SD value as final value for analyzing each graph-metric.

We first compare sampling results visually on Facebook data in Figure 2 (For space limits, we only provide visual comparison at 15% sampling rate). The non-distributed sampling results and distributed sampling results are visualized in two columns. Each row in Figure 2 represents one sampling method. To allow for better comparison, the original graph is also visualized in Figure 1.

We then compare sampling results in statistics on Facebook (Figure 3) and Amazon (Figure 4) graph, and each comparison includes two sampling rates: 15% and 25%. In each statistical comparison, the line charts (in Figure 3 and 4) indicate the SD between sampling results and original graphs for each sampling method on each statistical property. 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 denotes more consistency between the sampling result 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 statistical comparison, we can identify how those sampling methods perform, and similarities or differences between them.

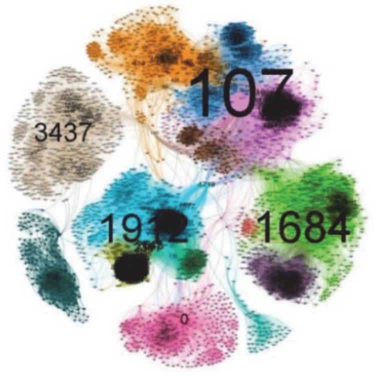
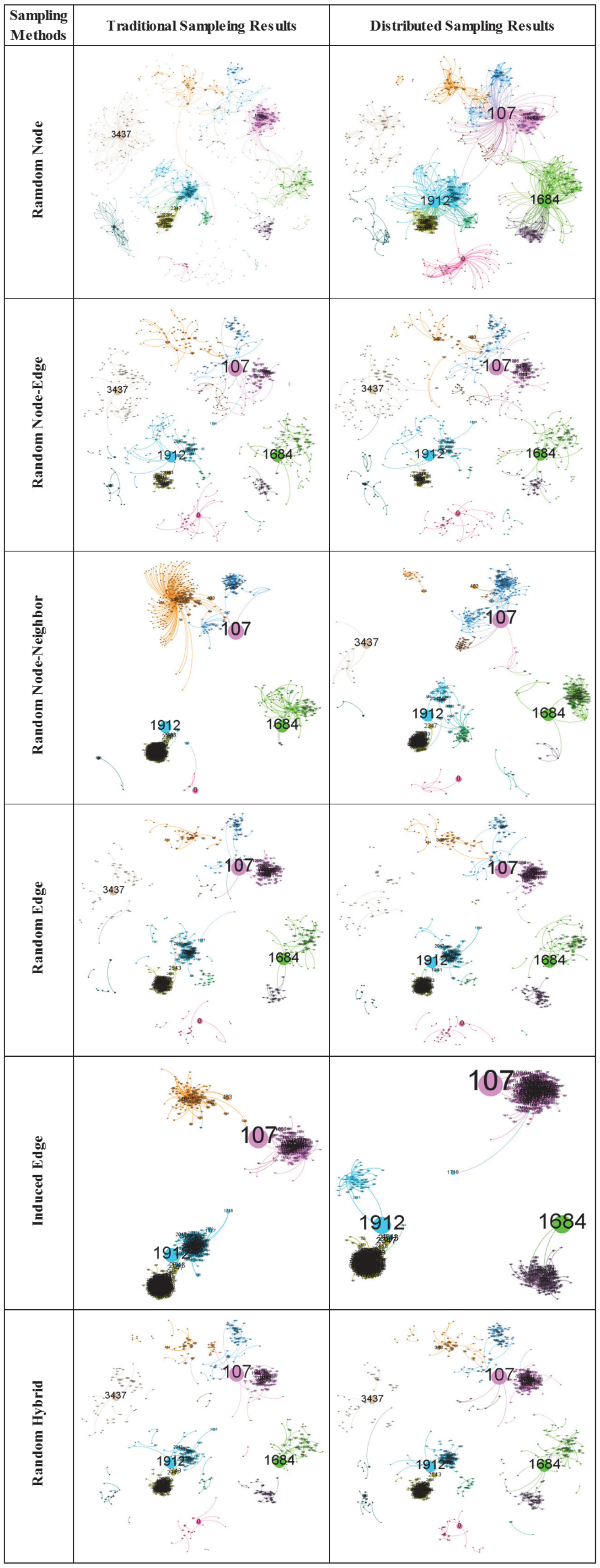


Figure 1: Visualization of original Facebook graph. Each group of vertices and edges with unique color represents one clusters. Label size is proportional to vertices' degree.



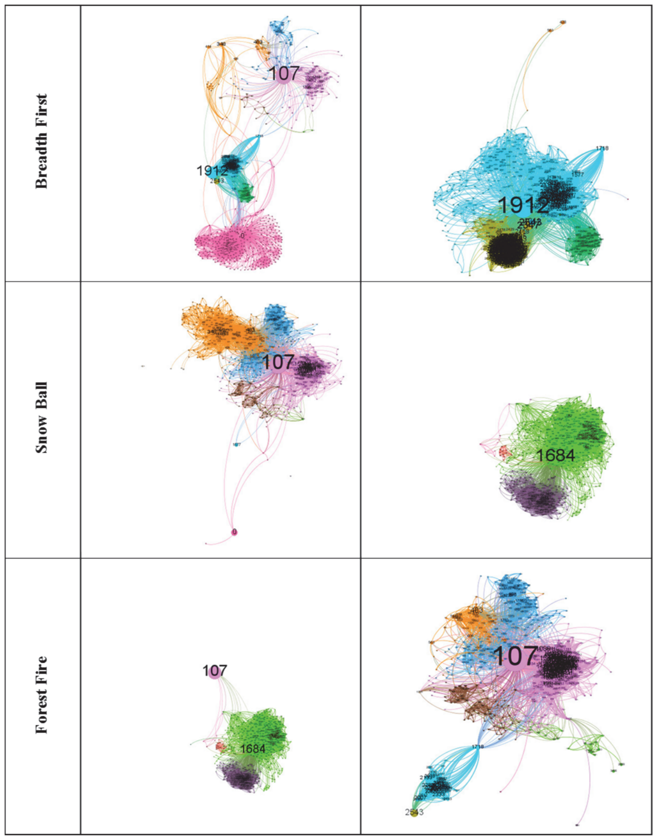


Figure 2: Visual comparison between non-distributed sampling methods and distributed sampling methods (in columns) at 15% sampling rate on Facebook graph. Each image stands for one sampling result created by corresponding sampling method in row.

The last but the most important comparison between non-distributed sampling methods and distributed sampling methods is on efficiency shown in Figure 5. Charts are presented in two columns representing efficiency comparison at two sampling rates (15% and 25%). For the Friendster graph, the execution time on non-distributed sampling methods is not provided in Figure 5 for the reason mentioned above.

Analysis

From the experiment results, we can compare sampling methods quantitatively or qualitatively in several aspects. First, from statistical comparison, we can identify how sampling methods preserve graph properties and the similarities and differences between two types of sampling methods. Second, from visual comparison, we discuss the similarities and differences between distributed sampling methods and non-distributed sampling approaches. Third, we compare those sampling methods in efficiency for each dataset. This study demonstrates that different distributed sampling methods have different advantages from traditional sampling methods. This study also provides starting point for other researchers based on the goals that they hope to accomplish and it provides more clear indications of with sampling methods should be used for different purposes. Finally, we summarize characteristics of distributed sampling methods.

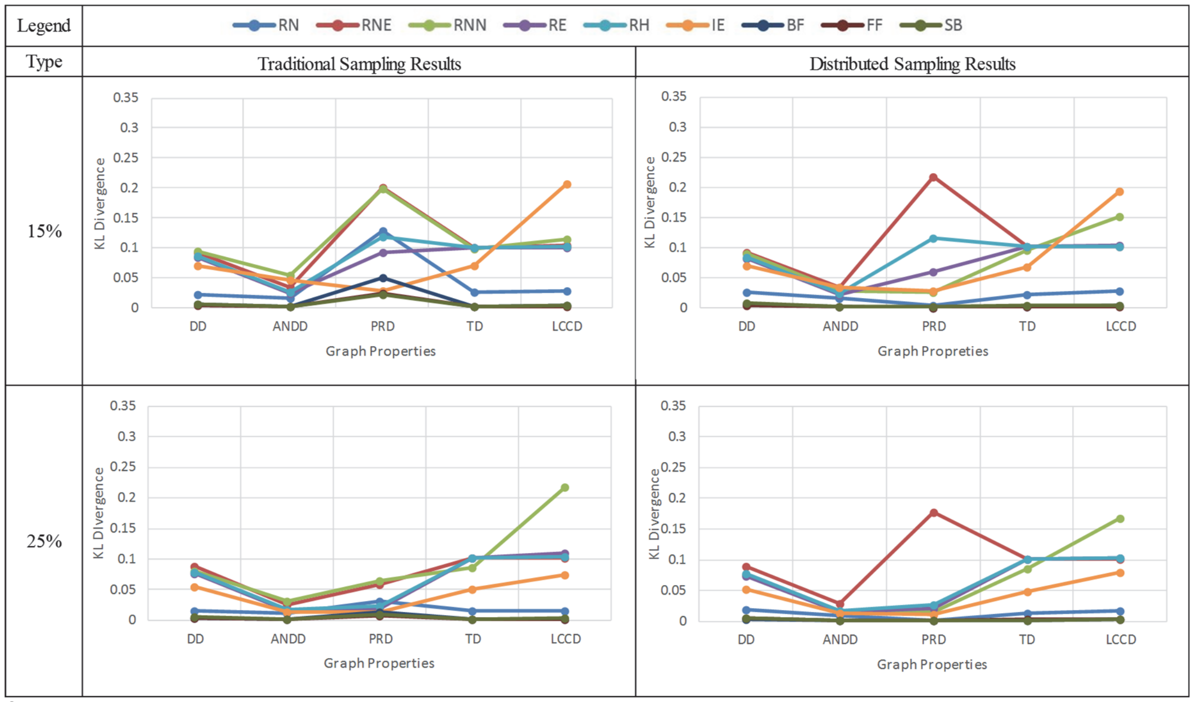


Figure 3: Statistical comparisons between non-distributed sampling methods and distributed sampling methods (in two column) on Facebook graph. The vertical axis is SD values, horizontal axis represents graph properties, and lines represent one sampling methods.

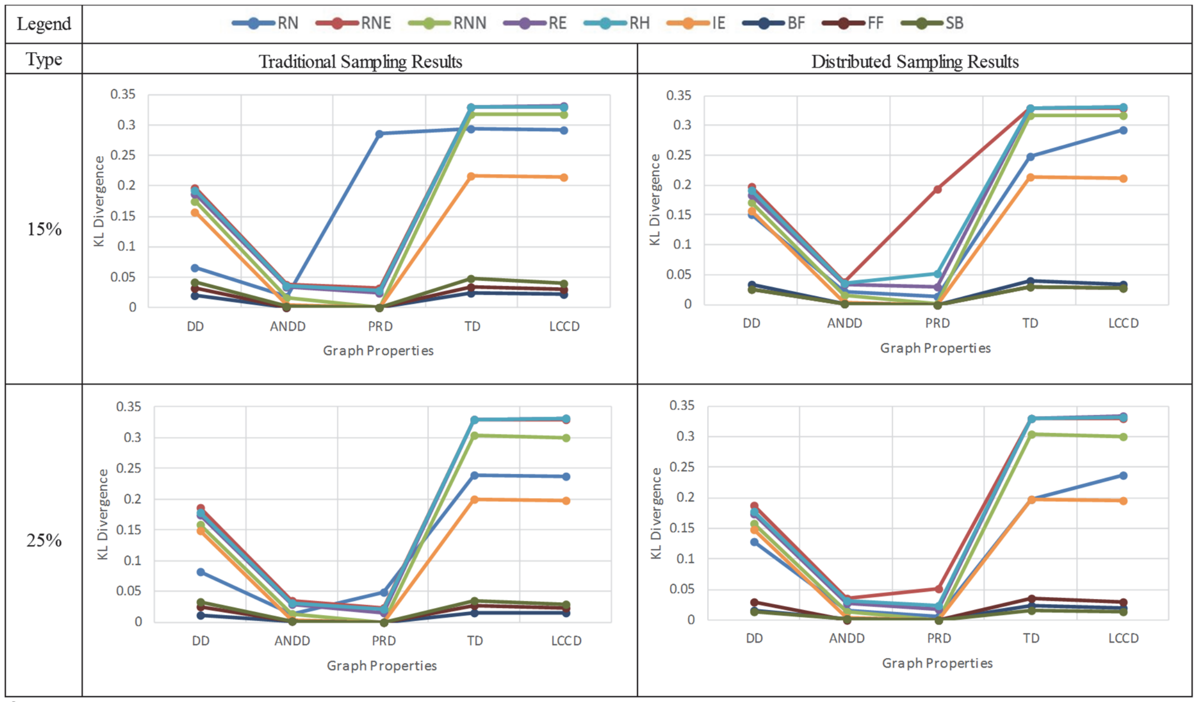


Figure 4: Statistical comparisons between non-distributed sampling methods and distributed sampling methods (in two columns) on Amazon graph. The vertical axis is SD values, horizontal axis represents graph properties, and lines represent one sampling methods.

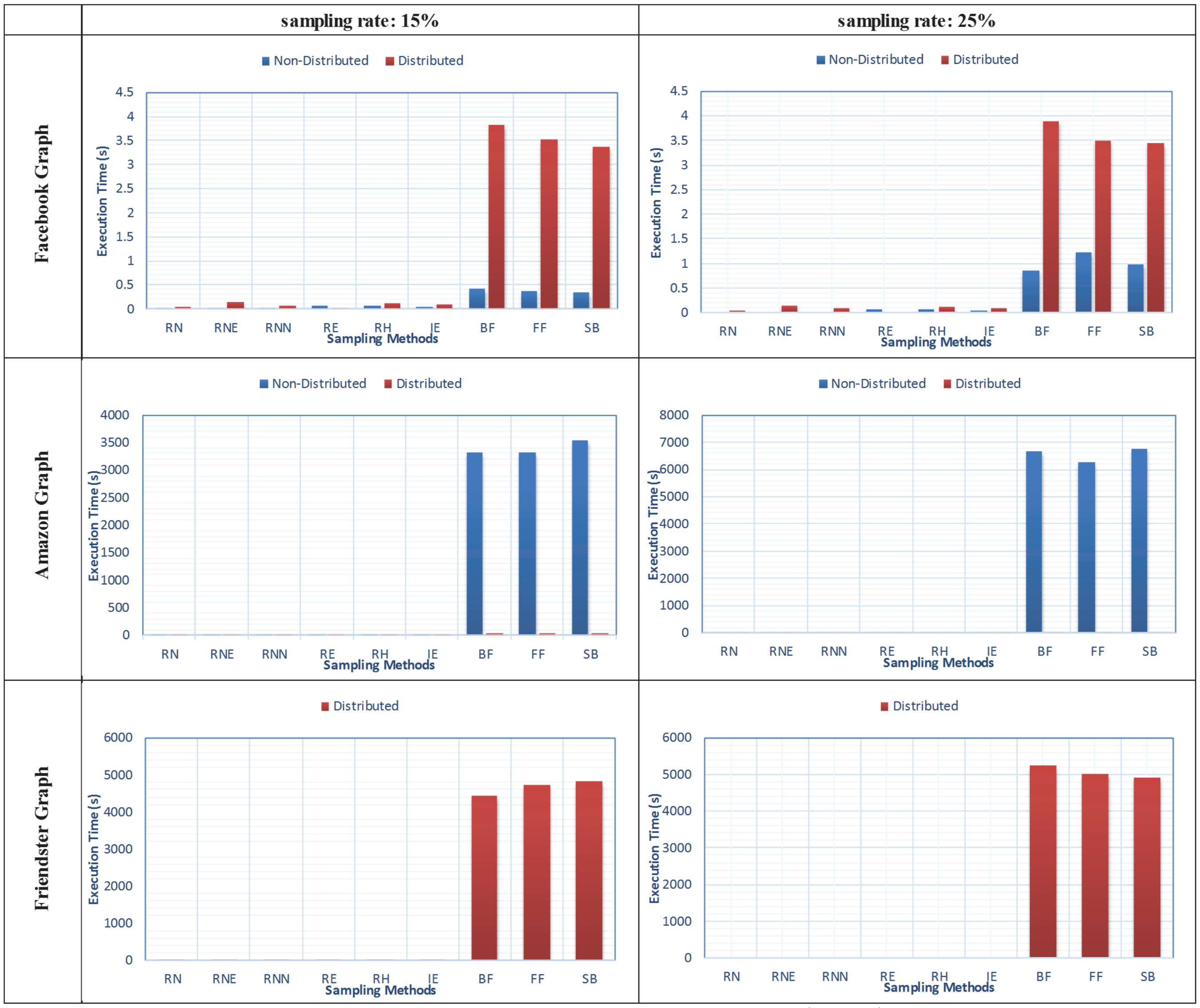


Figure 5: Execution time between non-distributed sampling methods and distributed sampling methods at 15% and 25% (in two columns) sampling rates on Facebook, Amazon, and Friendster datasets (in three rows). In each chart, the horizontal axis represents sampling methods and the vertical axis represents execution time in seconds.

Statistical Comparison

From statistical comparison results in Figure 3 and 4, we can draw some observations. Generally, distributed sampling methods have the similar performance as the non-distributed in preserving graph properties except for a few sampling methods. In Figure 3 and 4, most sampling methods show roughly consistent performance between the two sampling types. There are some abnormal cases. For example, in Facebook data (Figure 3) at sampling rate 15%, distributed random node neighbor sampling shows better than non-distributed random node neighbor sampling in persevering PageRank. In Amazon data (Figure 3) at sampling rate 15%, distributed random node sampling shows better than non-distributed random node sampling in persevering PageRank. Conversely, in Facebook data (Figure 3) at sampling rate 25% and Amazon data (Figure 4) at sample rate 15%, non-distributed random node-edge sampling method preserves PageRank better than distributed node-edge sampling method. In addition, by observing Figure 3 and 4, we find that both non-distributed and distributed random sampling methods (e.g. random node sampling, random edge sampling) cannot behave as well as topology-based sampling in preserving degree, triangle, and local clustering coefficient, particularly for large graphs. For instance, in Figure 4, random sampling methods shows bad performance on degree distribution, triangle distribution, and local clustering coefficient distribution. Topology-based sampling methods perform consistently well in preserving all graph properties. This observation about non-distributed topology-based sampling methods accords with findings in previous studies [6] [16].

Visual Comparison

We visualize sampling results from Facebook graph at sampling rate 15% for both non-distributed and distributed sampling methods, shown in Figure 2. From the visual comparison, we can find out the tendency and preference of each sampling method in preserving visual properties.

First, distributed random sampling methods have very similar performance with non-distributed random sampling methods in preserving spatial coverage. For example, by referring Figure 1, we can find both kinds of random sampling roughly generate same spatial area in both column. This indicates that distributed random sampling methods have the same preference as non-distributed random sampling methods when sampling graphs. Second, sometimes distributed sampling methods have better ability to preserve the number of clusters. From the visual comparison in Figure 2, distributed random node sampling and random node neighbor sampling can sample more clusters than corresponding non-distributed sampling approaches. Third, for topology-based sampling, their ability to preserve spatial coverage and clusters is more random. This is because the seed node is select at random and it has great influence in sampling results. In breadth first sampling, non-distributed sampling seems to be better in preserving spatial coverage and the number clusters. But in forest fire sampling, distributed sampling method shows better than non-distributed sampling in preserving such visual properties. Forth, generally, random sampling methods are more likely to sample broad spatial coverage and more clusters at the same sampling rate with topology-based sampling methods. This conclusion is consistent with previous findings as well [16].

Efficiency Comparison

To compare the efficiency for all sampling methods, we record the execution time for all methods during the sampling process (shown in Figure 5). From this figure, we can draw some interesting observations. First, for small graphs, non-distributed sampling methods have higher efficiency than distributed sampling methods. However, for large-scale graphs, distributed sampling approaches have remarkable improvement in efficiency (around 100 times) than non-distributed sampling methods, in particular for topology-based sampling (e.g. breadth first sampling, forest fire sampling, snowball sampling). For example, in Figure 5, Amazon graph is larger than Facebook graph. Non-distributed sampling methods use less time in Facebook graph at both sampling rates. However, distributed sampling methods are more efficient in the Amazon graph for both sampling rates. For very large-scale graphs, distributed sampling methods have huge advantage over non-distributed sampling methods. For instance, non-distributed topology-based sampling cannot complete the sampling process in an affordable time, but distributed topology-based sampling methods can complete in 2 hours on billion-scale graphs. In addition, for both kinds of sampling approaches, sampling time increases with the increase of sampling rate and graph size, but random sampling (e.g. random node sampling, random edge sampling) is not as sensitive to graph size as topology-based sampling. This observation is also reflected in previous studies [16].

Summary of Distributed Graph Sampling Methods

From the above statistical comparison, visual comparison, and efficiency comparison. The merits of distributed sampling methods can be summarized as follows.

- Distributed sampling methods roughly have similar ability in preserving graph properties. They are as reliable as non-distributed sampling methods. In some cases, distributed sampling performs better than non-distributed sampling.
- Generally, distributed sampling methods have similar performance with non-distributed random sampling methods in preserving spatial coverage. On some occasions, distributed sampling methods have better ability to preserve clusters than non-distributed sampling methods.
- Distributed sampling is more efficient than non-distributed sampling on large-scale graphs, which scales with graph size.
- Distributed sampling methods have horizontal scalability. Theoretically, while using distributed sampling methods, more machines can handle larger graphs.

Conclusion

We designed and developed nine distributed sampling methods and made statistical comparisons, visual comparisons, and efficiency comparisons with non-distributed sampling methods by applying those sampling methods to Facebook graph, Amazon graph, and Friendster graph.

In the statistical comparison, there were five graph properties used to evaluate sampling methods in preserving such quantitative statistical properties. We found distributed sampling methods have good reliability compared with non-distributed sampling methods. This observation offers us a reasonable justification to apply such sampling methods to large graphs in application. We can also use such sampling methods to make accurate estimations about graph properties on original graphs.

In visual comparison, we analyzed distributed sampling methods and non-distributed sampling methods on their ability to preserve spatial coverage and size, shape, and number of clusters. Such comparisons provide us an intuitive understanding of the similarities and differences among them. From the comparisons, we learned that distributed random sampling approaches have similar or even better performance in preserving spatial coverage and clusters. This observation gives us another justification to use distributed sampling methods in application.

In our efficiency comparison, we found distributed sampling methods have a great advantage over non-distributed sampling methods when sampling on large-scale graphs. This is the most important reason why we designed and develop distributed sampling methods for large-scale graphs.

From the above three comparison, we summarized the assets of distributed sampling methods. The findings in this paper could help users choose proper sampling methods in application.

Acknowledgments

This work has been supported by the United States Army Corps of Engineers under Contracts W912HZ-17-C-0016 and W912HZ-17-C-0015, by the U.S. Department of Defense, and by the Pacific Northwest National Laboratory which is managed for the U.S. Department of Energy by Battelle under Contract DE-AC05-76RL01830.

Reference

[1] Scott, John, “Social network analysis,” *Sociology* 22, no. 1 (1988): 109-127.

[2] M. Ripeanu, A. Iamnitchi, and I. Foster, “Mapping the Gnutella network,” *IEEE Internet Comput.*, vol. 6, no. 1, pp. 50–57, 2002.

[3] J. Gehrke, P. Ginsparg, and J. Kleinberg, “Overview of the 2003 KDD Cup,” *ACM SIGKDD Explorations Newsletter*, vol. 5. p. 149, 2003.

[4] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks.,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 99, pp. 7821–7826, 2002.

[5] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui, “Evaluation of Graph Sampling: A Visualization Perspective,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1. pp. 401–410, 2017.

[6] N. Ahmed, J. Neville, and R. R. Kompella, “Network sampling via edge-based node selection with graph induction,” *Computer Science Technical Report*. Paper 1747, 2011.

[7] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, “Walking in Facebook: A case study of unbiased sampling of OSNs,” in *Infocom, 2010 Proceedings IEEE*, 2010, pp. 1–9.

[8] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, “Analysis of topological characteristics of huge online social networking services,” in *Proceedings of the 16th international conference on World Wide Web - WWW ’07*, 2007, p. 835.

[9] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, “On unbiased sampling for unstructured peer-to-peer networks,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 377–390, 2009.

[10] 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.

[11] 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.

[12] N. K. Ahmed, J. Neville, and R. Kompella, “Network Sampling: From Static to Streaming Graphs,” *Tkdd*, vol. V, no. 212, 2013.

[13] P. Ebbes, Z. Huang, and A. Rangaswamy, “Sampling of large-scale social networks: Insights from simulated networks,” *8th Annu. Work. Inf. Technol. Syst.*, 2008.

[14] M. Kurant, A. Markopoulou, and P. Thiran, “Towards unbiased BFS sampling,” *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1799–1809, 2011.

[15] B. Ribeiro and D. Towsley, “Estimating and Sampling Graphs with Multidimensional Random Walks,” *In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 390-403. ACM, 2010.

[16] F. Zhang, S. Zhang, P. Chung Wong, H. Medal, L. Bian, J. E. Swan II, and T. J. Jankun-Kelly, “A Visual Evaluation Study of Graph Sampling Techniques,” *Electron. Imaging*, vol. 2017, no. 1, 2017.

[17] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li, “Understanding graph sampling algorithms for social network analysis,” *Proc. - Int. Conf. Distrib. Comput. Syst.*, pp. 123–128, 2011.

[18] Apache Spark, “Apache Spark^{TM} - Lightning-Fast Cluster Computing,” *Spark.Apache.Org*, 2015. .

[19] P. Hu and W. Lau, “A survey and taxonomy of graph sampling,” *arXiv preprint arXiv:1308.5865*, 2013.

[20] 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.

[21] J. Dittrich and J.-A. Quiané-Ruiz, “Efficient big data processing in Hadoop MapReduce,” *Proc. VLDB Endow.*, vol. 5, no. 12, pp. 2014–2015, 2012.

[22] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, “GraphX : Graph Processing in a Distributed Dataflow Framework,” *11th USENIX Symp. Oper. Syst. Des. Implement.*, pp. 599–613, 2014.

[23] J. Leskovec and A. Krevl, “{SNAP Datasets}: {Stanford} Large Network Dataset Collection.” Jun-2014. *URL https://snap.stanford.edu/data*

[24] J. Leskovec and J. Mcauley, “Learning to discover social circles in ego networks,” *Adv. neural Inf. Process.*, pp. 1–9, 2012.

[25] J. Yang and J. Leskovec, “Defining and evaluating network communities based on ground-truth,” *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, 2015.

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

[27] 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.