# Network Sampling: From Static to Streaming Graphs

NESREEN K. AHMED, JENNIFER NEVILLE, and RAMANA KOMPELLA, Purdue University

Network sampling is integral to the analysis of social, information, and biological networks. Since many real-world networks are massive in size, continuously evolving, and/or distributed in nature, the network structure is often sampled in order to facilitate study. For these reasons, a more thorough and complete understanding of network sampling is critical to support the field of network science. In this paper, we outline a framework for the general problem of network sampling by highlighting the different objectives, population and units of interest, and classes of network sampling methods. In addition, we propose a spectrum of computational models for network sampling methods, ranging from the traditionally studied model based on the assumption of a static domain to a more challenging model that is appropriate for streaming domains. We design a family of sampling methods based on the concept of graph induction that generalize across the full spectrum of computational models (from static to streaming) while efficiently preserving many of the topological properties of the input graphs. Furthermore, we demonstrate how traditional static sampling algorithms can be modified for graph streams for each of the three main classes of sampling methods: node, edge, and topology-based sampling. Experimental results indicate that our proposed family of sampling methods more accurately preserve the underlying properties of the graph in both static and streaming domains. Finally, we study the impact of network sampling algorithms on the parameter estimation and performance evaluation of relational classification algorithms.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms, Design, Experimentation, Performance

Additional Key Words and Phrases: Network sampling, social network analysis, graph streams, relational classification

**ACM Reference Format:**
Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. 2014. Network sampling: From static to streaming graphs. ACM Trans. Knowl. Discov. Data 8, 2, Article 7 (May 2014), 56 pages.
DOI: http://dx.doi.org/10.1145/2601438

## 1. INTRODUCTION

Networks arise as a natural representation for data in a variety of settings, ranging from social to biological to information domains. However, many of these real-world networks are massive and continuously evolving over time (i.e., streaming). For example, consider online activity and interaction networks formed from electronic communication (e.g., email, SMS, IMs), social media (e.g., Twitter, blogs, web pages), and content sharing (e.g., Facebook, Flicker, Youtube). These social processes produce a prolific

This research is supported by ARO and NSF under contract numbers W911NF-08-1-0238, IIS-1017898, IIS-1149789, and IIS-1219015. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of ARO, NSF, or the U.S. Government.
Authors' addresses: N. K. Ahmed, J. Neville, and R. Kompella, Department of Computer Science, Purdue University, West Lafayette, IN 47906 USA; email: {nkahmed, neville, kompella}@cs.purdue.edu.

DOI: http://dx.doi.org/10.1145/2601438

amount of continuous, interaction data (e.g., Facebook users post 3.2 billion likes and comments every day [Lafferty 2012]) that is naturally represented as a dynamic network—where the nodes are people or objects and the edges are the interactions among them. Modeling and analyzing these large dynamic networks have become increasingly important for many applications, such as identifying the behavior and interests of individuals (e.g., viral marketing, online advertising) and investigating how the structure and dynamics of human-formed groups evolve over time.

Unfortunately, many factors make it difficult, if not impossible, to study these networks in their entirety. First and foremost, the sheer size of many networks makes it computationally infeasible to study the entire network. Moreover, some networks are not completely visible to the public (e.g., Facebook) or can only be accessed through crawling (e.g.,World Wide Web). In other cases, the size of the full network may not be as large but the measurements required to observe the underlying network are costly (e.g., experiments in biological networks). Thus, network sampling provides the foundation for studies to understand network structure—since researchers typically need to select a (tractable) subset of the nodes and edges from which to make inferences about the full network.

From peer-to-peer to social communication networks, sampling arises across many different settings. Sampled networks may be used in simulations and experimentation—to measure performance before deploying new protocols and systems in the field, such as new Internet protocols, social/viral marketing schemes, and/or fraud detection algorithms. Moreover, many of the network datasets currently being analyzed as complete networks are themselves samples due to the limitations in data collection. Thus, it is critical that researchers understand the impact of sampling methods on the structure of the constructed networks. All of these factors motivate the need for a more refined and complete understanding of *network sampling*. In this article, we outline a spectrum of computational models for network sampling and investigate methods of sampling that generalize across this spectrum, going from the simplest and least constrained model focused on sampling from static graphs to the more difficult and most constrained model of sampling from graph streams.

Traditionally, network sampling has been studied in the case of simple static graphs [Leskovec and Faloutsos 2006]. These works make the simplifying assumption that the graphs are of moderate size and have static structure. Specifically, it is assumed that the graphs fit in the main memory (i.e., algorithms assume the full neighborhood of each node can be explored in constant time), and many of the intrinsic complexities of realistic networks, such as the time-evolving nature of these systems, are ignored. For domains where these assumptions are appropriate, we propose a new family of sampling methods based on the concept of graph induction and evaluate our methods against state-of-the-art approaches from the three classes of network sampling algorithms (node, edge, and topology-based sampling). We then show that our proposed methods preserve the properties of different graphs more accurately than competing sampling methods.

Although studying static graphs is indeed important, the assumption that the graph fits in memory is not always realistic for real-world domains (e.g., online social networks). When the network is too large to fit in memory, sampling requires random disk accesses that incur large I/O costs. Naturally, this raises the question: how can we sample from these large networks *sequentially*, one edge at a time, while *minimizing* the number of *passes* over the edges? In this context, most of the topology-based sampling procedures such as breadth-first search (BFS), random walk, or forest-fire sampling (FFS) are not appropriate as they require random exploration of a node's neighbors (which would require many passes over the edges). To address this, we demonstrate that our proposed sampling methods naturally apply to large graphs, requiring only

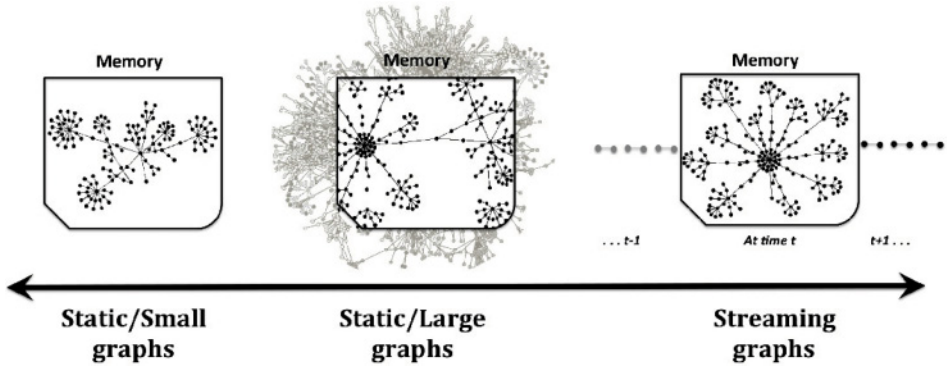



Fig. 1. Spectrum of Computational Models for network sampling: from static to streaming.

*two passes* over the edges. Moreover, we show that our proposed methods are still able to accurately preserve the properties of large graphs while minimizing the number of passes over the edges (more accurately than alternative algorithms).

Finally, in addition to their massive size, many real-world networks are also likely to be dynamically formed over time. A *streaming graph* is a rapid, continuous, and possibly unbounded time-varying stream of *edges* that is clearly too large to fit in memory except for short windows of time (e.g., a single day). Streaming graphs occur frequently in the real world and can be found in many modern online and communication applications such as Twitter posts, Facebook likes/comments, email communications, network monitoring, and sensor networks, among many other applications. Although these domains are quite prevalent, there has been little focus on developing network sampling algorithms that address the complexities of streaming domains. Graph streams differ from static graphs in three main aspects: (i) the massive volume of edges is far too large to fit in the main memory; (ii) the graph structure is not fully observable at any point in time (i.e., only sequential access is feasible, not random access); and (iii) efficient, real-time processing is of critical importance.

This discussion shows a natural progression of computational models for sampling—from static to streaming. The majority of previous work has focused on sampling from static graphs, which is the simplest and least restrictive problem setup. In this article, we also investigate the more challenging issues of sampling from disk-resident graphs and graph streams. This leads us to propose a spectrum of computational models for network sampling as shown in Figure 1, where we outline the three computational models for sampling from: (1) static graphs, (2) large graphs, and (3) streaming graphs. This spectrum not only provides insights into the complexity of the computational models (i.e., static vs. streaming), but also the complexity of the algorithms that are designed for each scenario. More complex algorithms are more suitable for the simplest computational model of sampling from static graphs. In contrast, as the complexity of the computational model increases to a streaming scenario, efficient algorithms become necessary. Thus, there is a tradeoff between the complexity of the sampling algorithm and the complexity of the computational model (static $\rightarrow$ streaming). A subtle but important consequence is that any algorithm designed to work over graph streams is also applicable in the simpler computational models (i.e., static graphs). However, the converse is not true: algorithms designed for sampling a static graph that can fit in memory may not be generally applicable for graph streams (as their implementation may require an intractable number of passes over the data).

Within the spectrum of computational models for network sampling, in Section 2 we formally discuss the various objectives of network sampling (e.g., sampling to

estimate network parameters). We provide insights on how conventional objectives in static domains can be naturally adapted to the more challenging scenario of streaming graphs. Then, in Section 3, we discuss the different models of computation that can be used to implement network sampling methods. Specifically, we formalize the problem of sampling from graph streams and outline how stream sampling algorithms need to decide whether to include an edge $e \in E$ in the sample or not as the edge is observed in the stream. The sampling algorithm may maintain a state $|\Psi|$ and consult the state to determine whether to sample the edge or not, but the total storage associated with the algorithm is preferred to be on the order of the size of the output sampled subgraph $G_s$, that is, $|\Psi| = O(|G_s|)$.

In Sections 5 and 6, we primarily investigate the task of sampling *representative* subgraphs from static and streaming graphs. Formally, the input is assumed to be a graph $G = (V, E)$, presented as a stream of edges $E$ in arbitrary order. Then, the goal is to sample a subgraph $G_s = (V_s, E_s)$ with a subset of the nodes ($V_s \subset V$) and/or edges ($E_s \subset E$) from the population graph stream $G$. The objective is to ensure that $G_s$ is representative, in that it matches many of the topological properties of $G$. In addition to the sample representativeness requirement, a graph stream sampling algorithm is also required to be *computationally efficient*.

In Section 6, we outline extensions of traditional sampling algorithms from each of the three classes of methods (node, edge, and topology-based sampling) for use on graph streams. For edge sampling from graph streams, we use the approach in Aggarwal et al. [2011]. For node and topology-based sampling, we develop streaming implementations based on reservoir sampling [Vitter 1985]. In addition, we propose a novel graph stream sampling method (from our family of methods based on the concept of graph induction), which is both space-efficient and runs in a *single pass* over the edges.

Overall, our family of proposed sampling methods generalize across the full spectrum of computational models (from static to streaming) while efficiently preserving many of the graph properties for streaming and static graphs. The proposed methods offer a good balance between algorithm complexity and sample representativeness while remaining general enough for any computational model. Notably, our family of sampling algorithms, although less complex than previous methods, also preserve the properties of *static* graphs more accurately than more complex algorithms (that do not generalize to streaming scenarios) over a broad set of real-world datasets.

In conclusion, we summarize the contributions of this article as follows:

—Detailed framework outlining the general problem of network sampling, highlighting the different goals, population and units, and classes of network sampling methods (see Section 2)
—Further elaboration of this framework to include a spectrum of computational models within which to design network sampling methods (i.e., going from static to streaming graphs) (see Section 3)
—Proposed family of sampling methods based on the concept of *graph induction* that has the following properties (see Sections 5 and 6):
  (1) Preserves many key graph characteristics more accurately than alternative state-of-the-art algorithms
  (2) Generalizes to a streaming computational model with a minimum storage space (i.e., space complexity is on the order of the sample size)
  (3) Runs efficiently (i.e., runtime complexity is on the order of the number of edges)
—Systematic investigation of three classes of static sampling methods on a variety of datasets and extension of these algorithms for application to graph streams (see Sections 5 and 6)

—Empirical evaluation showing our sampling methods are accurate and efficient for large graphs that do not fit into main memory (see Sections 5 and 6)
—Further task-based evaluation of sampling algorithm performance in the context of relational classification, which illustrates the impact of network sampling on parameter estimation and evaluation of classification algorithms overlaid on sampled networks (see Section 7)

## 2. FOUNDATIONS OF NETWORK SAMPLING

In the context of statistical data analysis, a number of issues need to be considered carefully before collecting data and making inferences based on them. First, we need to identify the relevant *population* to be studied. Then, if sampling is necessary, we need to decide how to sample from that population. Generally, the term *population* is defined as the full set of representative units that one wishes to study (e.g., individuals in a particular city). In some instances, the population may be relatively small and therefore easy to study in its entirety (i.e., without sampling). For instance, it is fairly easy to study the *full* set of graduate students in a particular academic department. However, in many situations, the population is large, unbounded, or difficult and/or costly to access in its entirety (e.g., the complete set of Facebook users). In this case, for efficiency reasons, a sample of units can be collected, and characteristics of the population can then be estimated from the sampled units.

Network sampling is of interest to a variety of researchers in a range of distinct fields (e.g. statistics, social science, databases, data mining, machine learning) due to the numerous complex datasets that can be represented as graphs. Although each area may investigate different types of networks, they have all considered *how* to sample. For example, in social science, snowball sampling is used extensively to run survey sampling in populations that are difficult to access (e.g., the set of drug users in a city) [Watters and Biernacki 1989]. Similarly, in Internet topology measurements, breadth-first search (BFS) is used to *crawl* distributed, large-scale online social networks [Mislove et al. 2007]. In structured data mining and machine learning, the focus has been on developing algorithms to sample small(er) subgraphs from a single large network [Leskovec and Faloutsos 2006]. These sampled subgraphs are further used to learn models (e.g., relational classification models [Friedman et al. 1999]), evaluate and compare the performance of algorithms (e.g., different classification methods [Rossi and Neville 2012; Rossi et al. 2012]), and study complex network processes (e.g., information diffusion [Bakshy et al. 2012]). Section 4 provides a more detailed discussion of related work. Although this large body of research has developed methods to sample from networks, much of the work is problem-specific, and there has been less work focused on developing a broader foundation for network sampling. More specifically, it is often not clear *when* and *why* particular sampling methods are appropriate. This is because the goals and population are often not explicitly defined or stated up front, which makes it difficult to evaluate the quality of the recovered samples for other applications. One of the primary aims of this article is to define and discuss the foundations of network sampling more explicitly, such as objectives/goals, population of interest, units, classes of sampling algorithms (i.e., node, edge, and topology-based), and techniques to evaluate a sample (e.g., network statistics and distance metrics). In this section, we outline a solid methodological framework for network sampling. The framework will facilitate the comparison of various network sampling algorithms, and help us to understand their relative strengths and weaknesses with respect to particular sampling goals.

### 2.1. Notation

Formally, we consider an input network represented as a graph $G = (V, E)$ with the node set $V = \{v_1, v_2, \ldots, v_N\}$ and edge set $E = \{e_1, e_2, \ldots, e_M\}$, such that $N = |V|$ is the

number of nodes, and $M = |E|$ is the number of edges. We denote $\eta(.)$ as any topological graph property. Therefore, $\eta(G)$ could be *a point statistic* (e.g., average degree of nodes in $V$) or *a distribution* (e.g., degree distribution of $V$ in $G$).

Further, we define $\Lambda = \{a_1, a_2, \ldots, a_k\}$ as the set of $k$ attributes associated with the nodes describing their properties. Each node $v_i \in V$ is associated with an attribute vector $[a_1(v_i), a_2(v_i), \ldots, a_k(v_i)]$, where $a_j(v_i)$ is the $j^{th}$ attribute value of node $v_i$. For instance, in a Facebook network where nodes represent users and edges represent friendships, the node attributes may include age, political view, and relationship status of the user.

Similarly, we denote $\beta = \{b_1, b_2, \ldots, b_l\}$ as the set of $l$ attributes associated with the edges describing their properties. Each edge $e_{ij} = (v_i, v_j) \in E$ is associated with an attribute vector $[b_1(e_{ij}), b_2(e_{ij}), \ldots, b_l(e_{ij})]$. In the Facebook example, edge attributes may include relationship type (e.g., friends, married), relationship strength, and type of communication (e.g., wall post, photo tag).

Now, we define the network sampling process. Let $\sigma$ be any sampling algorithm that selects a random sample $S$ from $G$ (i.e., $S = \sigma(G)$). The sampled set $S$ could be a subset of the nodes ($S = V_s \subset V$), or edges ($S = E_s \subset E$), or a subgraph ($S = (V_s, E_s)$, where $V_s \subset V$ and $E_s \subset E$). The size of the sample $S$ is defined relative to the graph size with respect to a sampling fraction $\phi$ ($0 \leq \phi \leq 1$). In most cases, the sample size is defined as a fraction of the nodes in the input graph, for example, $|S| = \phi \cdot |V|$. But, in some cases, the sample size is defined relative to the number of edges ($|S| = \phi \cdot |E|$).

### 2.2. Goals, Units, and Population

Although the explicit aim of many network sampling algorithms is to select a smaller subgraph from $G$, there are often other more implicit goals of the process that are left unstated. Here, we formally outline a range of possible goals for network sampling:

(1) Estimate network parameters
Let $S \subset V$ or $S \subset E$. Then, for a property $\eta$ of $G$, if $\eta(S) \approx \eta(G)$, $S$ is considered a *good* sample of $G$.

For example, let $S = V_s \subset V$ be the subset of sampled nodes; we can then estimate the average degree of nodes $G$ using $S$:

$$\hat{deg}_{avg} = \frac{1}{|S|} \sum_{v_i \in S} deg(v_i \in G),$$

where $deg(v_i \in G)$ is the degree of node $v_i$ as it appears in $G$, and a direct application of statistical estimators helps to correct sampling bias in $\hat{deg}_{avg}$ [Hansen and Hurwitz 1943].

(2) Sample a representative subgraph
Let $S = G_s$ refer to a subgraph $G_s = (V_s, E_s)$ sampled from $G$. Then, for a set of topological properties $\eta_A$ of $G$, if $\eta_A(S) \approx \eta_A(G)$, $S$ is considered a *good* sample of $G$.

Generally, subgraph representativeness is evaluated by selecting a set of graph topological properties that are important for a wide range of applications. This ensures that the sample subgraph $S$ can be used in place of $G$ for testing algorithms, systems, and/or models in an application. For example, Leskovec and Faloutsos [2006] evaluated sample quality using topological properties like degree, clustering, and eigenvalues.

(3) Estimate node attributes
Let $S \subset V$. Then, for a function $f_a$ of node attribute $a$, if $f_a(S) \approx f_a(V)$, $S$ is considered a *good* sample of $V$ (where $V$ is the set of nodes in $G$).

For example, if $a$ represents the age of users, we can estimate the average in $G$ using $S$:

$$\hat{a}_{avg} = \frac{1}{|S|} \sum_{v_i \in S} a(v_i)$$

Similar to goal 1, statistical estimators can be used to correct for bias.

(4) Estimate edge attributes

Let $S \subset E$. Then, for a function $f_b$ of edge attribute $b$, if $f_b(S) \approx f_b(E)$, $S$ is considered a *good* sample of $E$ (where $E$ is the set of edges in $G$).

For example, if $b$ represents the relationship type (e.g., married, coworkers), we can estimate the proportion of married relationships in $G$ using $S$:

$$\hat{p}_{married} = \frac{1}{|S|} \sum_{e_{ij} \in S} 1_{(b(e_{ij})=married)}$$

Clearly, the first two goals (1 and 2) focus on characteristics of entire networks, whereas the last two goals (3 and 4) focus on characteristics of nodes or edges in isolation. Therefore, these goals may be difficult to satisfy simultaneously—that is, if the sampled data enable accurate study for one, they may not allow accurate study for others. For instance, a representative subgraph sample could produce a biased estimate of node attributes.

Once the goal is outlined, the population of interest can be defined relative to the goal. In many cases, the definition of the population may be obvious (e.g., nodes in the network). The main challenge is then to select a representative subset of units in the population in order to make the study cost efficient and feasible. Other times, the population may be less tangible and difficult to define. For example, if one wishes to study the characteristics of a system or *process*, there is not a clearly defined set of items to study. Instead, one is often interested in the overall behavior of the system. In this case, the population can be defined as the set of possible outcomes from the system (e.g., measurements over all settings), and these *units* should be sampled according to their underlying probability distribution.

In the first two goals outlined, the objective of study is an entire network (either for structure or parameter estimation). In goal 1, if the objective is to estimate local properties from the nodes (e.g., degree distribution of $G$), then the elementary units are the nodes, and then the population would be the set of all nodes $V$ in $G$. However, if the objective is to estimate global properties (e.g., diameter of $G$), then the elementary units correspond to subgraphs (any $G_s \subset G$) rather than nodes, and the population should be defined as the set of subgraphs of a particular size that could be drawn from $G$. In goal 2, the objective is to select a subgraph $G_s$; thus, the elementary units correspond to subgraphs rather than nodes or edges (goal 3 and 4). As such, the population should also be defined as the set of subgraphs of a particular size that could be drawn from $G$.

### 2.3. Classes of Sampling Methods

Once the population has been defined, a sampling algorithm $\sigma$ must be chosen to sample from $G$. Sampling algorithms can be categorized as node, edge, and topology-based sampling, based on whether nodes or edges are locally selected from $G$ (node and edge-based sampling) or if the selection of nodes and edges depends more on the existing topology of $G$ (topology-based sampling).

Graph sampling algorithms have two basic steps:

(1) *Node selection:* used to sample a subset of nodes $S = V_s$ from $G$, (i.e., $V_s \subset V$)
(2) *Edge selection:* used to sample a subset of edges $S = E_s$ from $G$, (i.e., $E_s \subset E$)

When the objective is to sample only nodes or edges (e.g., goals 3, 4, or 1), then either step 1 or step 2 is used to form the sample $S$. When the objective is to sample a subgraph $G_s$ from $G$ (e.g., goals 2 or 1), then both step 1 and 2 are used to form $S$ (i.e., $S = (V_s, E_s)$). In this case, the edge selection is often conditioned on the selected node set in order to form an *induced subgraph* by sampling a subset of the edges incident to $V_s$ (i.e., $E_s = \{e_{ij} = (v_i, v_j) | e_{ij} \in E \wedge v_i, v_j \in V_s\}$). This process is called graph induction. We distinguish between two approaches of graph induction—*total* and *partial* graph induction—which differ by whether *all* or *some* of the edges incident on $V_s$ are selected. The resulting sampled graphs are referred to as the *induced subgraph* and *partially induced subgraph* respectively.

Although the discussion of the algorithms in the next sections focuses more on sampling a subgraph $G_s$ from $G$, they can easily generalize to sampling only nodes or edges.

*2.3.1. Node Sampling (NS).* In classic node sampling, nodes are chosen independently and uniformly at random from $G$ for inclusion in the sampled graph $G_s$. For a target fraction $\phi$ of nodes required, each node is simply sampled with a probability of $\phi$. Once the nodes are selected for $V_s$, the sampled subgraph is constructed to be the *induced subgraph* over the nodes $V_s$; that is, all edges among the $V_s \in G$ are added to $E_s$. Although node sampling is intuitive and relatively straightforward, the work in Stumpf et al. [2005] show that it does not accurately capture properties of graphs with power-law degree distributions. Similarly, Lee et al. [2006] show that although node sampling appears to capture nodes of different degrees well, due to its inclusion of all edges for a chosen node set only, the original level of connectivity is not likely to be preserved.

*2.3.2. Edge Sampling (ES).* In classic edge sampling, edges are chosen independently and uniformly at random from $G$ for inclusion in the sampled graph $G_s$. Since edge sampling focuses on the selection of edges rather than nodes to populate the sample, the node set is constructed by including both incident nodes in $V_s$ when a particular edge is sampled (and added to $E_s$). The resulting subgraph is partially induced, which means no extra edges are added over and above those that were chosen during the random edge selection process. Unfortunately, ES fails to preserve many desired graph properties. Due to the independent sampling of edges, it does not preserve clustering and connectivity. It is however more likely to capture path lengths due to its bias toward high-degree nodes and the inclusion of both end points of selected edges.

*2.3.3. Topology-Based Sampling.* Due to the known limitations of NS [Stumpf et al. 2005; Lee et al. 2006] and ES (bias toward high-degree nodes), researchers have also considered many other topology-based sampling methods (also referred to as exploration sampling) that use BFS (i.e., sampling without replacement) or random walks (i.e., sampling with replacement) over the graph to construct a sample.

One example is snowball sampling, which adds nodes and edges using BFS from a randomly selected seed node but stops early once it reaches a particular size. Snowball sampling accurately maintains the network connectivity within the snowball, but it suffers from *boundary bias* in that many peripheral nodes (i.e., those sampled on the last round) will be missing a large number of neighbors [Lee et al. 2006].

Another example is the Forest Fire Sampling (FFS) method [Leskovec and Faloutsos 2006], which uses *partial* BFS where only a fraction of neighbors are followed for each node. The algorithm starts by picking a node uniformly at random and adding it to the sample. It then "burns" a random proportion of its outgoing links and adds those edges, along with the incident nodes, to the sample. The fraction is determined by sampling from a geometric distribution with mean $(p_f/(1-p_f))$. The authors recommend setting

Table I. Description of Network Statistics

| Network Statistic | Description |
|---|---|
| DEGREE DIST. | Distribution of degrees for all nodes in the network |
| PATH LENGTH DIST. | Distribution of (finite) shortest path lengths between all pairs of nodes in the network |
| CLUSTERING COEFFICIENT DIST. | Distribution of local clustering for all nodes in the network |
| K-CORE DIST. | Distribution of *k-core* decomposition of the network |
| EIGENVALUES | Distribution of the eigenvalues of the network adjacency matrix vs. their rank |
| NETWORK VALUES | Distribution of eigenvector components vs. their rank, for the largest eigenvalue of the network adjacency matrix |

$p_f = 0.7$, which results in an average burn of 2.33 edges per node. The process is repeated recursively for each burned neighbor until no new node is selected, then a new random node is chosen to continue the process until the desired sample size is obtained. There are other algorithms, such as respondent-driven sampling [Heckathorn 1997] and expansion sampling [Maiya and Berger-Wolf 2010], that we give more details on in Section 4.

In general, such topology-based sampling approaches form the sampled graph out of the explored nodes and edges and usually perform better than simple algorithms such as NS and ES.

### 2.4. Evaluation of Sampling Methods

When the goal is to approximate the entire input network—either for estimating parameters (goal 1) or to select a representative subgraph structure (goal 2)—the accuracy of network sampling methods is often measured by comparing structural network statistics (e.g., degree). We first define a suite of common network statistics and then discuss how they can be used to quantitatively compare sampling methods.

*2.4.1. Network Statistics.* The commonly considered network statistics can be compared along two dimensions: *local* versus *global* statistics, and *point* statistic versus *distribution*. A local statistic is used to describe a characteristic of a local graph element (e.g., node, edge, subgraph); for example, node degree and node clustering coefficient. On the other hand, a global statistic is used to describe a characteristic of the entire graph; for example, global clustering coefficient and graph diameter. Similarly, there is also the distinction between point statistics and distributions. A point-statistic is a single value statistic (e.g., diameter), whereas a distribution is a multivalued statistic (e.g., distribution of path length for all pairs of nodes). Clearly, a range of network statistics is important to investigate the full graph structure.

In this work, we focus on the goal of sampling a representative subgraph $G_s$ from $G$ by using distributions of network characteristics calculated on the level of nodes, edges, sets of nodes or edges, and subgraphs. Table I provides a summary for the six network statistics we use, and we formally define the statistics in the following:

(1) *Degree distribution*: The fraction of nodes with degree $k$, for all $k > 0$

$$p_k = \frac{|\{v \in V | deg(v) = k\}|}{N}$$

Degree distribution has been widely studied by many researchers to understand the connectivity in graphs. Many real-world networks were shown to have a power-law degree distribution, for example in the Web [Kleinberg et al. 1999], citation graphs [Redner 1998], and online social networks [Chakrabarti et al. 2004].

(2) *Path length distribution*: Also known as the *hop plot* distribution, it denotes the fraction of pairs $(u, v) \in V$ with a shortest-path distance $(dist(u, v))$ of $h$, for all

$h > 0$ and $h \neq \infty$

$$p_h = \frac{|\{(u, v) \in V | dist(u, v) = h\}|}{N^2}$$

The path length distribution is essential to know how the number of paths between nodes expand as a function of distance (i.e., number of hops).

(3) *Clustering coefficient distribution*: The fraction of nodes with clustering coefficient ($cc(v)$) $c$, for all $0 \leq c \leq 1$

$$p_c = \frac{|\{v \in V' | cc(v) = c\}|}{|V'|}, \quad \text{where} \quad V' = \{v \in V | deg(v) > 1\}$$

Here, the clustering coefficient of a node $v$ is calculated as the number of triangles centered on $v$ divided by the number of pairs of neighbors of $v$ (e.g., the proportion of $v$'s neighbor that are linked). In social networks and many other real networks, nodes tend to cluster. Thus, the clustering coefficient is an important measure to capture the transitivity of the graph [Watts and Strogatz 1998].

(4) *K-core distribution*: The fraction of nodes in graph $G$ participating in a *k-core* of order $k$. The *k-core* of $G$ is the largest induced subgraph with minimum degree $k$. Formally, let $U \subseteq V$, and $G_{[U]} = (U, E')$, where $E' = \{e_{u,v} \in E | u, v \in U\}$. Then, $G_{[U]}$ is a *k-core* of order $k$ if $\forall v \in U$ $deg_{G_{[U]}}(v) \geq k$.

Studying *k-cores* is an essential part of social network analysis because they demonstrate the connectivity and community structure of the graph [Carmi et al. 2007; Alvarez-Hamelin et al. 2005; Kumar et al. 2006]. We denote the *maximum core number* as the maximum value of $k$ in the *k-core* distribution. The *maximum core number* can be used as a lower bound on the degree of the nodes that participate in the largest induced subgraph of $G$. Also, the core sizes can be used to demonstrate the localized density of subgraphs in $G$ [Seshadhri et al. 2013].

(5) *Eigenvalues*: The set of real eigenvalues $\lambda_1 \geq \cdots \geq \lambda_N$ of the corresponding adjacency matrix $A$ of $G$. Since *eigenvalues* are the basis of spectral graph analysis, we compare the largest 25 eigenvalues of the sampled graphs to their counterparts in $G$.

(6) *Network values*: The distribution of the eigenvector values in the component associated with the largest eigenvalue $\lambda_{max}$. We compare the largest 100 network values of the sampled graphs to their counterparts in $G$.

Next, we describe the use of these statistics for comparing sampling methods quantitatively.

*2.4.2. Distance Measures for Quantitatively Comparing Sampling Methods.* A *good* sample has properties that approximate the full graph $G$ (i.e., $\eta(S) \approx \eta(G)$). Thus, distance between the property in $G$ and the property in $G_s$ ($dist[\eta(G), \eta(G_S)]$) is often used to evaluate sample *representativeness* quantitatively, and sampling algorithms that minimize the distance are considered superior. When the goal is to provide estimates of global network parameters (e.g., average degree), then $\eta(.)$ may return *point statistics*. However, when the goal is to provide a representative subgraph sample, then $\eta(.)$ may return *distributions* of network properties (e.g., degree distribution). These distributions reflect how the graph structure is distributed across nodes and edges. The *dist* function used for evaluation could be typically any distance measure (e.g., absolute difference). In this article, since we focus on using distributions to characterize graph structure, we use four different distributional distance measures for evaluation.

(1) *Kolmogorov-Smirnov (KS) statistic:* Used to assess the distance between two Cumulative Distribution Functions (CDF). The KS statistic is a widely used measure

of the agreement between two distributions, including in Leskovec and Faloutsos [2006], where it is used to illustrate the accuracy of FFS. It is computed as the maximum vertical distance between the two distributions, where $x$ represents the range of the random variable and $F_1$ and $F_2$ represent two CDFs:

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

(2) *Skew Divergence (SD):* Used to assess the difference between two Probability Density Functions (PDF) [Lee 2001]. Skew divergence is used to measure the Kullback-Leibler (KL) divergence between two PDFs $P_1$ and $P_2$ that do not have continuous support over the full range of values (e.g., discrete degrees). KL measures the average number of extra bits required to represent samples from the original distribution when using the sampled distribution. However, since KL divergence is not defined for distributions with different areas of support, skew divergence *smooths* the two PDFs before computing the KL divergence:

$$SD(P_1, P_2, \alpha) = KL[\alpha P_1 + (1-\alpha)P_2 \,||\, \alpha P_2 + (1-\alpha)P_1]$$

The results shown in Lee [2001] indicate that using SD yields better results than other methods to approximate KL divergence on nonsmoothed distributions. In this article, as in Lee [2001], we use $\alpha = 0.99$.

(3) *Normalized $L_1$ distance:* In some cases, for evaluation we will need to measure the distance between two positive m-dimensional real vectors $p$ and $q$ such that $p$ is the true vector and $q$ is the estimated vector; for example, to compute the distance between two vectors of eigenvalues. In this case, we use the normalized $L_1$ distance:

$$L_1(p, q) = \frac{1}{m}\sum_{i=1}^{m}\frac{|p_i - q_i|}{p_i}$$

(4) *Normalized $L_2$ distance:* In other cases, when the vector components are fractions (less than one), we use the normalized euclidean distance $L_2$ distance (e.g., to compute the distance between two vectors of network values):

$$L_2(p, q) = \frac{||p - q||}{||p||}$$

## 3. MODELS OF COMPUTATION

In this section, we discuss the different models of computation that can be used to implement network sampling methods. At first, let us assume the network $G = (V, E)$ is given (e.g., stored on a large storage device). Then, the goal is to select a sample $S$ from $G$.

Traditionally, network sampling has been explored in the context of a *static model of computation*. This simple model makes the fundamental assumption that it is easy and fast (i.e., constant time) to randomly access any location of the graph $G$. For example, random access may be used to query the entire set of nodes $V$ or to query the neighbors $\mathcal{N}(v_i)$ of a particular node $v_i$ (where $\mathcal{N}(v_i) = \{v_j \in V | e_{ij} = (v_i, v_j) \in E\}$). However, random accesses on disks are much slower than random accesses in main memory. A key disadvantage of the static model of computation is that it does not differentiate between a graph that can fit entirely in the main memory and a graph that cannot. Conversely, the primary advantage of the static model is that it is a natural extension of how we *understand* and *view* the graph—thus, it is a simple framework within which to design algorithms.

Although designing sampling algorithms with a static model of computation in mind is indeed appropriate for some applications, this approach assumes the input graphs

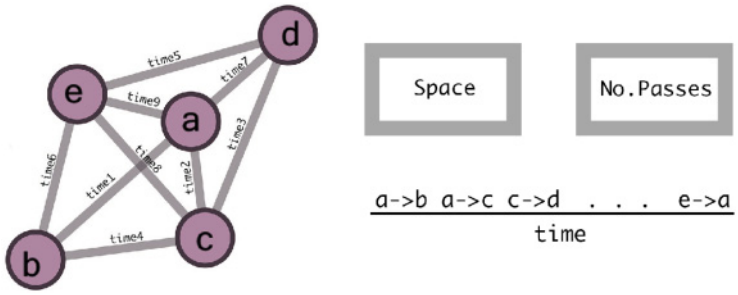



Fig. 2. Illustration of a *graph stream*—a sequence of edges ordered over time and the complexity constraints of streaming algorithms (space and number of passes).

are relatively small, can fit entirely into main memory, and have static structure (i.e., not changing over the time). This is unrealistic for many domains. For instance, many social, communication, and information networks naturally change over time and are massive in size (e.g., Facebook, Twitter, Flickr). The sheer size and dynamic nature of these networks makes it difficult to load the full graph entirely in the main memory. Therefore, the static model of computation cannot realistically capture all the intricacies of many real-world graphs that we study today.

In addition, many real-world networks that are currently of interest are *too large* to fit into memory. In this case, sampling methods that require random disk access can incur large I/O costs for loading and reading the data. Naturally, this raises a question as to how we can sample from large networks more efficiently (i.e., in a sequential fashion rather than assuming random access). In this context, most of the topology-based sampling procedures such as BFS and random-walk sampling are no longer appropriate because they require the ability to randomly access a node's neighbors $\mathcal{N}(v_i)$. If access is restricted to sequential passes over the edges, a large number of passes over the edges would be needed to repeatedly query $\mathcal{N}(\cdot)$. In a similar way, node sampling would no longer be appropriate because it not only requires random access for querying a node's neighbors but it also requires random access to the entire node set $V$ in order to obtain a uniform random sample.

A *streaming model of computation,* in which the graph can only be accessed sequentially as a stream of edges, is therefore more preferable for these situations [Zhang 2010]. The streaming model completely discards the possibility of random access to $G$, and the graph can only be accessed through an ordered scan of the edge stream. A sampling algorithm in this context may use the main memory for holding a portion of the edges temporarily and perform random accesses on that subset. In addition, the sampling algorithm may access the edges repeatedly by making multiple passes over the graph stream. Formally, for any input network $G$, we assume $G$ is represented as a graph stream (e.g., as in Figure 2).

*Definition* 3.1 (*Graph Stream*). A *graph stream* is an ordered sequence of edges $e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(M)}$, where $\pi$ is any arbitrary permutation on the edge indices $[M] = \{1, 2, \ldots, M\}$, $\pi : [M] \rightarrow [M]$.

Definition 3.1 is usually called the "adjacency stream" model in which the graph is presented as a stream of edges in an arbitrary order. In contrast, the "incidence stream" model assumes all edges incident to a vertex are presented in order successively [Buriol et al. 2006]. In this article, we use the adjacency stream model because it is more reflective of the temporal ordering we observe in real world datasets.

Although most real-world networks are too large to fit into main memory, many are also likely to occur naturally as *streaming* data. A streaming graph is a rapid, continuous, and possibly unbounded, time-varying stream of edges that is both too large and too dynamic to fit into memory. These types of streaming graphs occur frequently in real-world communication and information domains; for example, in real-time tweets between users in Twitter, email logs, IP traffic, sensor networks, web search traffic, and many other applications. Although sampling from these streaming networks is clearly challenging, their characteristics preclude the use of static models of computation, and thus a more in-depth investigation of streaming models of computation is warranted. This naturally raises a follow-up question: how can we sample from large graph streams in a *single pass* over the edges? Generally streaming graphs differ from static graphs in three main aspects:

(1) The massive volume of edges streaming over the time is far too large to fit in main memory.
(2) The graph can only be accessed sequentially in a single pass (i.e., random access to neighboring nodes or to the entire graph is not possible).
(3) Efficient, real-time processing is of critical importance.

In a streaming model, as each edge $e \in E$ arrives, the sampling algorithm $\sigma$ needs to decide whether to include the edge or not as the edge *streams* by. The sampling algorithm $\sigma$ may also maintain state $\Psi$ and consult the state to determine whether to sample $e$ or not.

The complexity of a streaming sampling algorithm is measured by:

(1) Number of passes over the stream $\omega$
(2) Space required to store the state $\Psi$ and the output
(3) Representativeness of the output sample $S$

Multiple passes over the stream (i.e., $\omega > 1$) may be allowed for massive disk-resident graphs, but multiple passes are not realistic for datasets where the graph is continuously streaming over time. In this case, a requirement of a single pass is more suitable (i.e., $\omega = 1$). The total storage space (i.e., $\Psi$) is usually of the order of the size of the output: $|\Psi| = O(|G_s|)$. Note that this requirement is potentially larger than the $o(N, t)$ (and preferably $polylog(N, t)$) that streaming algorithms typically require [Muthukrishnan 2005]. But, since any algorithm cannot require less space than its output, we relax this requirement in our definition as follows:

*Definition* 3.2 (*Streaming Graph Sampling*). A *streaming graph sampling algorithm* is any sampling algorithm $\sigma$ that produces a sampled graph $G_s$ by sampling edges of the input graph $G$ in a sequential order, preferably in *one pass* (i.e., $\omega = 1$), while maintaining state $\Psi$ such that $|\Psi| \leq O(|G_s|)$.

Clearly, it is more difficult to design sampling algorithms for the *graph stream* model, but it is critical to address the fundamental intricacies of, and implementation requirements for, real-world graphs that we see today.

We now have what can be viewed as a complete spectrum of computational models for network sampling, which ranges from the simple, yet less realistic, static graph model to the more complex, but more realistic, streaming model (see Figure 1). In Sections 5 and 6, we evaluate algorithms for representative subgraph sampling in each computation model from the spectrum.

We note that our assumption in this work is that the population graph $G$ is visible in its entirety (*collected and stored on disk*). In many domains, this assumption is valid, but, in some cases, the full structure of the population graph may be unknown prior to the sampling process (e.g., the deep Web or distributed information in peer-to-peer

networks). Web/network crawling is used extensively to sample from graphs that are not fully visible to the public but naturally allow methods to explore the neighbors of a given node (e.g., hyperlinks in a web page). Topology-based sampling methods (e.g., BFS, random walk) have been widely used in this context. However, these methods typically assume the graph $G$ is *well connected* and remains *static* during crawling, as discussed in Gjoka et al. [2011].

## 4. RELATED WORK

Generally speaking, there are two bodies of work related to this article: (i) network sampling methods, which investigate and evaluate sampling methods with different goals for collecting a sample; and (ii) graph stream methods, including work on mining and querying streaming data. In this section, we describe related work and put it in the perspective of the framework we discussed in Section 2.

The problem of sampling graphs has been of interest in many different fields of research. Most of this body of research has focused on *how* to sample and each project evaluates the "goodness" of the resulting samples relative to its specific research goal(s).

### 4.1. Network Sampling in Social Science

In social science, the classic work done by Frank [1977] (also see the review papers [Frank 1980, 1981]) provides basic solutions to the initial problems that arise when only a sample of the actors in a social network is available. Goodman [1961] introduced the first snowball sampling method and originated the concept of "chain-referral" sampling. Furthermore, Granovetter introduced the network community to the problem of making inferences about the entire population from a sample (e.g., estimation of network density) [Granovetter 1976]. Later, respondent-driven sampling was proposed in Heckathorn [1997] and analyzed in Gile and Handcock [2010] to reduce the biases associated with chain referral sampling of hidden populations. For an excellent survey about estimation of network properties from samples, we refer the reader to Kolaczyk [2009]. Generally, work in this area focuses on either the estimation of global network parameters (e.g., density) or the estimation of actors (node) attributes, that is, goals 1 and 3.

### 4.2. Statistical Properties of Network Sampling

Another important trend of research focused on analyzing the statistical properties of sampled subgraphs. For example, the work in Lee et al. [2006] and Yoon et al. [2007] studied the statistical properties of sampled subgraphs produced by classical node, edge, and random walk sampling methods and discussed the bias in estimates of topological properties. Similarly, the work of Stumpf et al. [2005] showed that the sampled subgraph of a scale- free network is far from being scale free. Conversely, the work in Lakhina et al. [2003] shows that under traceroute sampling, the resulting degree distribution follows a power law even when the original distribution is Poisson. Clearly, work in this area has focused on representative subgraph sampling (i.e., goal 2), considering how the topological properties of samples differ from those of the original network.

### 4.3. Network Sampling in Network Systems Research

A large body of research in network systems has focused on Internet *measurement*, which targets the problem of topology measurements in large-scale networks, such as Peer-to-Peer Networks (P2P), the World Wide Web (WWW), and Online Social Networks (OSN). The sheer size and distributed structure of these networks make it hard to measure the properties of the entire network. Network sampling, via *crawling*, has been used extensively in this context. In OSNs, sampling methods that do not revisit

nodes are widely used (e.g., BFS [Ahn et al. 2007; Mislove et al. 2007; Wilson et al. 2009]). Breadth-first search has been shown to be biased toward high-degree nodes [Ye et al. 2010], but the work in Kurant et al. [2011] suggested analytical solutions to correct the bias. Random walk sampling has also been used to sample a uniform sample from users in Facebook and Last.fm (see e.g., Gjoka et al. [2010]). For a recent survey covering assumptions and comparing different methods of crawling, we refer the reader to Gjoka et al. [2011]. Similar to OSNs, random walk sampling and its variants were used extensively to sample the WWW [Baykan et al. 2009; Henzinger et al. 2000], and P2P networks [Gkantsidis et al. 2004]. Since the classical random walk is biased toward high-degree nodes, some improvements were applied to correct the bias. For example, the work in Stutzbach et al. [2006] applied Metropolis-Hastings Random Walks (MHRW) to sample peers in Gnutella network, and the work in Rasti et al. [2009] applied Re-Weighted Random Walk (RWRW) to sample P2P networks. Other work used multiple dependent random walks and random walks with jumps [Ribeiro and Towsley 2010; Avrachenkov et al. 2010].

Overall, the work done in this area has focused extensively on sampling a uniform subset of nodes from the graph, to estimate topological properties of the entire network from the set of sampled nodes (i.e., goal 1).

### 4.4. Network Sampling in Structured Data Mining

Network sampling is a core part of data mining research. Representative subgraph sampling was first defined in Leskovec and Faloutsos [2006]. Hubler et al. [2008] then proposed a generic Metropolis algorithm to optimize the representativeness of a sampled subgraph—by minimizing the distance of several graph properties from the sample to the original graph. Unfortunately, the number of steps until convergence is not known in advance and is usually quite large in practice. In addition, each step requires the computation of a complex distance function, which may be costly. In contrast to sampling, Krishnamurthy et al. [2007] explored reductive methods to shrink the existing topology of the graph. At the same time, other work discussed the difficulty of constructing a "universally representative" subgraph that preserves *all* properties of the original network. For example, our past work discussed the correlations between network properties and showed that accurately preserving some properties leads to under- or overestimation of other properties (e.g., preserving the average degree of the original network leads to a sampled subgraph with overestimated density) [Ahmed et al. 2010b]. Also, Maiya and Berger-Wolf [2011] investigated the connection between the biases of topology-based sampling methods (e.g., BFS) and some topological properties of the network (e.g., degree). These findings led to work focused on obtaining samples for specific applications, where the goal is to reproduce specific properties of the target network—for example, to preserve the community structure [Maiya and Berger-Wolf 2010], to preserve the pagerank between all pairs of sampled nodes [Vattani et al. 2011], or to visualize the graph [Jia et al. 2008].

Other network sampling goals have been considered as well; for example, sampling nodes to perform A/B testing of social features [Backstrom and Kleinberg 2011], sampling connected subgraphs [Lu and Bressan 2012], sampling nodes to analyze the fraction of users with a certain property [Dasgupta et al. 2012], (i.e., goal 3), and sampling tweets (edges) to analyze the language used in Twitter (i.e., goal 4). In addition, Al Hasan and Zaki [2009] and Bhuiyan et al. [2012] sample the output space of graph mining algorithms (e.g., graphlets), Papagelis et al. [2013] collected information from social peers to enhance the information needs of users, and De Choudhury et al. [2010] studied the impact of sampling on the discovery of information diffusion.

Much of this work has focused on sampling in the context of a static model of computation—where algorithms assume that the graph can be loaded entirely into

main memory, or the graph is distributed in a manner that allows exploration of a node's neighborhood in a crawling fashion.

### 4.5. Graph Streams

Data stream querying and mining has garnered a lot of interest over the past few years [Babcock et al. 2002a; Golab and Özsu 2003; Muthukrishnan 2005; Aggarwal 2007], as for example, sampling sequences (e.g., reservoir sampling) [Vitter 1985; Babcock et al. 2002b; Aggarwal 2006], computing frequency counts [Manku and Motwani 2002; Charikar et al. 2002] and load shedding [Tatbul et al. 2003], mining concept drifting data streams [Wang et al. 2003; Gao et al. 2007; Fan 2004a, 2004b], clustering evolving data streams [Guha et al. 2003; Aggarwal et al. 2003], active mining and learning in data streams [Fan et al. 2004; Li et al. 2009], and other related mining tasks [Domingos and Hulten 2000; Hulten et al. 2001; Gaber et al. 2005; Wang et al. 2010].

Recently, as a result of the proliferation of graph data (e.g., social networks, emails, IP traffic, Twitter hashtags), there has been an increased interest in mining and querying *graph streams*. Following the earliest work on graph streams [Henzinger et al. 1999], various problems were explored in the field of mining graph streams; for example, counting triangles [Bar-Yossef et al. 2002; Buriol et al. 2006], finding common neighborhoods [Buchsbaum et al. 2003], estimating pagerank values [Sarma et al. 2008], and characterizing degree sequences in multigraph streams [Cormode and Muthukrishnan 2005]. More recently, there is work on clustering graph streams [Aggarwal et al. 2010b], outlier detection [Aggarwal et al. 2011], searching for subgraph patterns [Chen and Wang 2010], and mining dense structural patterns [Aggarwal et al. 2010a].

Graph stream sampling was utilized in some of the work just mentioned. For example, Sarma et al. [2008] performed short random walks from uniformly sampled nodes to estimate pagerank scores. Also, Buriol et al. [2006] used sampling to estimate number of triangles in the graph stream. Moreover, Cormode and Muthukrishnan [2005] used a min-wise hash function to sample nearly uniformly from the set of all edges that have been at any time in the stream. The sampled edges were later used to maintain cascaded summaries of the graph stream. More recently, Aggarwal et al. [2011] designed a structural reservoir sampling approach (based on min-wise hash sampling of edges) for structural summarization. For an excellent survey on mining graph streams, we refer the reader to McGregor [2009] and Zhang [2010].

The majority of this work has focused on sampling a subset of nodes uniformly from the stream to estimate parameters such as the number of triangles or pagerank scores of the graph stream (i.e., goal 1). Also, as discussed earlier, other work has focused on sampling a subset of edges uniformly from the graph stream to maintain summaries (i.e., goal 2). These summaries can be further pruned (by lowering a threshold on the hash value [Aggarwal et al. 2011]) to satisfy a specific stopping constraint (e.g., specific number of nodes in the summary). In this article, since we focus primarily on sampling a representative subgraph $G_s \subset G$ from the graph stream, we compare to some of these methods in Section 6.

## 5. SAMPLING FROM STATIC GRAPHS

In this section, we focus on how to sample a representative subgraph $G_s = (V_s, E_s)$ from $G = (V, E)$ (i.e., goal 2 from Section 2.2). A representative sample $G_s$ is essential for many applications in machine learning, data mining, and network simulations. As an example, it can be used to drive realistic simulations and experimentation before deploying new protocols and systems in the field [Krishnamurthy et al. 2007]. We evaluate the representativeness of $G_s$ relative to $G$ by comparing distributions of six topological properties calculated over nodes, edges, and subgraphs (as summarized in Table I).

First, we distinguish between the degree of the sampled nodes before and after sampling. For any node $v_i \in V_s$, we denote $k_i$ to be the node degree of $v_i$ in the input graph $G$. Similarly, we denote $k_i^s$ to be the node degree of $v_i$ in the sampled subgraph $G_s$. Note that $k_i = |\mathcal{N}(v_i)|$, where $\mathcal{N}(v_i) = \{v_j \in V \mid e_{ij} = (v_i, v_j) \in E\}$) is the set of neighbors of node $v_i$. Clearly, when a node is sampled, it is not necessarily the case that all its neighbors are sampled as well, and therefore $0 \leq k_i^s \leq k_i$.

In this section, we propose a simple and efficient sampling algorithm based on the concept of graph-induction: *induced edge sampling* (for brevity ES-i). ES-i has several advantages over current sampling methods as we show later in this section:

(1) ES-i preserves the topological properties of $G$ better than many of current sampling algorithms.
(2) ES-i can be easily implemented as a *streaming* sampling algorithm using only two passes over the edges of $G$ (i.e., $\omega = 2$).
(3) ES-i is suitable for sampling large graphs that cannot fit into main memory.

### 5.1. Algorithm

We formally specify ES-i in Algorithm 1. Initially, ES-i selects the nodes in pairs by sampling edges uniformly (i.e., $p(e_{ij}$ is selected$) = 1/|E|$) and adds them to the sample ($V_s$). Then, ES-i augments the sample with all edges that exist between any of the sampled nodes ($E_s = \{e_{ij} = (v_i, v_j) \in E \mid v_i, v_j \in V_S\}$). These two steps together form the sample subgraph $G_s = (V_s, E_s)$. For example, suppose edges $e_{12} = (v_1, v_2)$ and $e_{34} = (v_3, v_4)$ are sampled in the first step, which leads to the addition of the vertices $v_1, \ldots, v_4$ into the sampled graph. In the second step, ES-i adds all the edges that exist between the sampled nodes—for example, edges $e_{12} = (v_1, v_2)$, $e_{34} = (v_3, v_4)$, $e_{13} = (v_1, v_3)$, $e_{24} = (v_2, v_4)$, and any other possible combinations involving $v_1, \ldots, v_4$ that appear in $G$.

**ALGORITHM 1:** Induced Edge Sampling ES-i($\phi$, $E$)

**Input**: Sample fraction $\phi$, Edge set $E$
**Output**: Sampled Subgraph $G_s = (V_s, E_s)$

```
1  V_s = ∅, E_s = ∅
2  // Node selection step
3  while |V_s| < φ × |V| do
4  |   r = random (1, |E|)
5  |   // uniformly random
6  |   e_r = (u, v)
7  |   V_s = V_s ∪ {u, v}
8  // Edge selection step
9  for k = 1 : |E| do
10 |   e_k = (u, v)
11 |   if u ∈ V_s AND v ∈ V_s then
12 |   |   E_s = E_s ∪ {e_k}
```

Since any sampling algorithm (by definition) selects only a subset of the nodes/edges in the graph $G$, it naturally produces subgraphs with underestimated degrees in the degree distribution of $G_s$. We refer to this as a *downward bias* and note that it is a property of all network sampling methods, since only a fraction of a node's neighbors may be selected for inclusion in the sample (i.e., $k_i^s \leq k_i$ for any sampled node $v_i$).

Our proposed sampling methods exploits two key observations. First, by selecting nodes via edge sampling, the method is inherently biased toward the selection of nodes

with high degrees, resulting in an *upward bias* in the (original) degree distribution if it is only measured from the sampled nodes (i.e., using the degree of the sampled nodes as observed in $G$). The upward bias resulting from edge sampling can help offset the downward bias of the sampled degree distribution of $G_s$. Furthermore, in addition to improving estimates of the sampled degree distribution, selecting high-degree nodes also helps to produce a more connected sample subgraph that preserves the topological properties of the graph $G$. This is due to the fact that high-degree nodes often represent *hubs* in the graph, which serve as good navigators through the graph (e.g., many shortest paths usually pass through these hub nodes).

However, although the upward bias of edge sampling can help offset some issues of sample selection bias, it is not sufficient to use it in isolation to construct a good sampled subgraph. Specifically, since the edges are each sampled independently, edge sampling is unlikely to preserve much structure *surrounding* each of the selected nodes. This leads us to our second observation, that a simple *graph induction* step over the sampled nodes (where we sample all the edges between any sampled nodes from $G$) is crucial to recover much of the connectivity in the sampled subgraph—offsetting the downward degree bias as well as increasing local clustering in the sampled graph. More specifically, graph induction increases the likelihood that triangles will be sampled among the set of selected nodes, producing higher clustering coefficients and shorter path lengths in $G_s$.

These observations, although simple, make the sample subgraph $G_s$ approximate the characteristics of the original graph $G$ more accurately, even better than topology-based sampling methods.

Since many real networks are now too large to fit into main memory, this raises the question of how to sample from $G$ sequentially, one edge at a time, while minimizing the number of passes over the edges? Section 3 discussed how most of the topology-based sampling methods are no longer applicable in this scenario, since they require many passes over the edges. In contrast, ES-i can be implemented to run sequentially and requires only two passes over the edges of $G$ (i.e., $\omega = 2$). Next, we analyze the characteristics of ES-i and, after that, in the evaluation, we show how it accurately preserves many of the properties of the graph $G$.

### 5.2. Analysis of Sampling Bias

In this section, we analyze the bias of ES-i's node selection analytically by comparing it to the unbiased case of uniform sampling in which all nodes are sampled with a uniform probability (i.e., $p = \frac{1}{N}$). First, we denote $f_D$ to be the degree sequence of $G$, where $f_D(k)$ is the number of nodes with degree $k$ in graph $G$. Let $n = |V_s|$ be the target number of nodes in the sample subgraph $G_s$ (i.e., $\phi = \frac{n}{N}$).

*5.2.1. Selection Bias Toward High-Degree Nodes.* We start by analyzing the upward bias to select high-degree nodes by calculating the expected value of the number of nodes with original degree $k$ that are added to the sample set of nodes $V_s$. Let $\mathrm{E}_{UN}[f_D(k)]$ be the expected value of $f_D(k)$ for the sampled set $V_s$ when $n$ nodes are sampled uniformly with probability $p = \frac{1}{N}$:

$$\begin{aligned} \mathrm{E}_{UN}[f_D(k)] &= f_D(k) \cdot n \cdot p \\ &= f_D(k) \cdot \frac{n}{N} \end{aligned}$$

Because ES-i selects nodes proportional to their degree, the probability of sampling a node $v_i$ with degree $k_i = k$ is $p' = \frac{k}{\sum_{j=1}^{N} k_j}$. Note that we can also express the probability

as $p' = \frac{k}{2 \cdot |E|}$. Then, we let $\mathrm{E}_{ES_i}[f_D(k)]$ denote the expected value of $f_D(k)$ for the sampled set $V_s$ when nodes are sampled with ES-i:

$$\begin{aligned} \mathrm{E}_{ES_i}[f_D(k)] &= f_D(k) \cdot n \cdot p' \\ &= f_D(k) \cdot n \cdot \frac{k}{2 \cdot |E|} \end{aligned}$$

This leads us to define Lemma 5.1, which shows ES-i's bias toward high-degree nodes.

LEMMA 5.1. *ES-i is biased to select high-degree nodes.*
*Let $k_{avg} = \frac{2 \cdot |E|}{N}$ be the average degree in G. Then, for $k \geq k_{avg}$, $\mathrm{E}_{ES_i}[f_D(k)] \geq \mathrm{E}_{UN}[f_D(k)]$.*

PROOF. Consider the threshold $k$ at which the expected value of $f_D(k)$ using ES-i sampling is greater than the expected value of $f_D(k)$ using uniform random sampling:

$$\begin{aligned} \mathrm{E}_{ES_i}[f_D(k)] - \mathrm{E}_{UN}[f_D(k)] &= f_D(k) \cdot n \cdot \frac{k}{2 \cdot |E|} - f_D(k) \cdot \frac{n}{N} \\ &= f_D(k) \cdot n \left[ \frac{k}{2 \cdot |E|} - \frac{1}{N} \right] \\ &= f_D(k) \cdot \frac{n}{2 \cdot |E|} [k - k_{avg}] \\ &\geq 0 \qquad \text{when } k \geq k_{avg} \quad \square \end{aligned}$$

*5.2.2. Downward Bias Due to Sampling Subgraphs.* Next, instead of focusing on the *original* degree $k$ as observed in the graph $G$, we focus on the *sampled* degree $k^s$ as observed in the sample subgraph $G_s$, where $0 \leq k^s \leq k$. Let $k_i^s$ be a random variable that represents the sampled degree of node $v_i$ in $G_s$, given that the original degree of node $v_i$ in $G$ was $k_i$. We compare the expected value of $k_i^s$ when using uniform sampling to the expected value of $k_i^s$ when using ES-i. Generally, the degree of the node $v_i$ in $G_s$ depends on how many of its neighbors in $G$ are sampled. When using uniform sampling, the probability of sampling one of the node's neighbors is $p = \frac{1}{N}$:

$$\mathrm{E}_{UN}\left[k_i^s\right] = \sum_{j=1}^{k_i} p \cdot n = k_i \cdot \frac{n}{N}$$

Now, let us consider the variable $k_{\mathcal{N}} = \sum_{k'} k' \cdot P(k'|k)$, where $k_{\mathcal{N}}$ represents the average degree of the neighbors of a node with degree $k$ as observed in $G$. The function $k_{\mathcal{N}}$ has been widely used as a global measure of the *assortativity* in a network [Newman 2002]. If $k_{\mathcal{N}}$ is increasing with $k$, then the network is assortative—indicating that nodes with high degree connect to, on average, other nodes with high degree. Alternatively, if $k_{\mathcal{N}}$ is decreasing with $k$, then the network is disassortative—indicating that nodes of high degree tend to connect to nodes of low degree.

When using ES-i, the probability of sampling any of the node's neighbors is proportional to the degree of the neighbor. Let $v_j$ be a neighbor of $v_i$ (i.e., $e_{ij} = (v_i, v_j) \in E$), then the probability of sampling $v_j$ is $p' = \frac{k_j}{2 \cdot |E|}$. Then, we define $k_{\mathcal{N}i} = \frac{\sum_{j=1}^{k_i} k_j}{k_i}$ to be the average degree of the neighbors of node $v_i$. Note that $k_{\mathcal{N}i} \geq 1$. In this context, $k_{\mathcal{N}i}$

represents the local assortativity of node $v_i$, so then:

$$\begin{aligned}
\mathrm{E}_{ES_i}\left[k_i^s\right] &= \sum_{j=1}^{k_i} p' \cdot n \\
&= \frac{n}{N} \cdot \frac{\sum_{j=1}^{k_i} k_j}{k_{avg}} \\
&= \frac{n}{N} \cdot \frac{k_i}{k_{avg}} \cdot \frac{\sum_{j=1}^{k_i} k_j}{k_i} \\
&= k_i \cdot \frac{n}{N} \cdot \frac{k_{\mathcal{N}i}}{k_{avg}}
\end{aligned}$$

This leads us to define Lemma 5.2, which shows when the sampled degrees using ES-i are higher than uniform random sampling.

LEMMA 5.2. *The expected sampled degree in $V_s$ using ES-i is greater than the expected sampled degree based on uniform node sampling when the local assortativity of the node is high.*

*Let $k_{avg} = \frac{2 \cdot |E|}{N}$ be the average degree in $G$ and let $k_{\mathcal{N}i} = \frac{\sum_{j=1}^{k_i} k_j}{k_i}$ be the average degree of $v_i$'s neighbors in $G$. For any node $v_i \in V_s$, if $k_{\mathcal{N}i} \geq k_{avg}$, then $\mathrm{E}_{ES_i}[k_i^s] \geq \mathrm{E}_{UN}[k_i^s]$.*

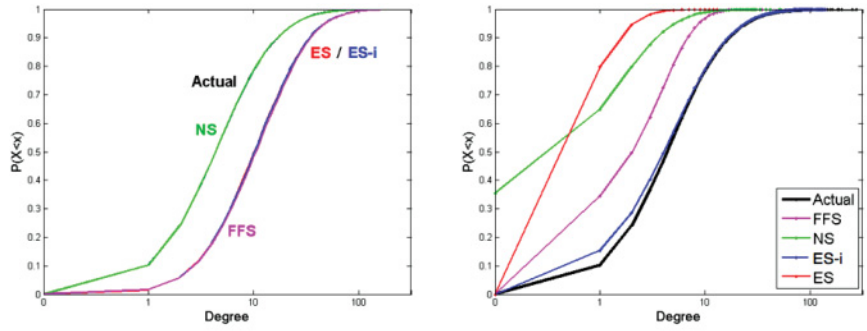
PROOF. Consider the threshold $k$ at which the expected value of $k^s$ using ES-i sampling is greater than the expected value of $k^s$ using uniform random sampling:

$$\begin{aligned}
\mathrm{E}_{ES_i}\left[k_i^s\right] - \mathrm{E}_{UN}\left[k_i^s\right] &= k_i \cdot \frac{n}{N} \cdot \frac{k_{\mathcal{N}i}}{k_{avg}} - k_i \cdot \frac{n}{N} \\
&= k_i \cdot \frac{n}{N}\left[\frac{k_{\mathcal{N}i}}{k_{avg}} - 1\right] \\
&\geq 0 \qquad \text{when } k_{\mathcal{N}i} \geq k_{avg} \quad \square
\end{aligned}$$

Generally, for any sampled node $v_i$, if the average degree of $v_i$'s neighbors is greater than the average degree of $G$, then its expected sampled degree under ES-i is higher than it would be if uniform sampling was used. This is often the case in many real networks where high-degree nodes are connected with other high-degree nodes.

In Figure 3, we empirically investigate the difference between the sampled degrees $k^s$ and original degrees $k$ for an example network—the CONDMAT graph. Specifically, in Figure 3(a), we compare the cumulative degree distribution (the CDF) of $G$ when measured from the full set of nodes $V$ to the CDF of $G$ when measured only from the set of sampled nodes $V_s$. To contrast this with the *sampled* degrees, in Figure 3(b), we compare the CDF of $G_s$ to the CDF of $G$ (measured from the full set of nodes $V$). Note that, in Figure 3, the x-axis denotes the degree ($k$ in Figure 3(a) and $k^s$ in Figure 3(b)), and the y-axis denotes the cumulative probability ($P(X < x)$).

In Figure 3(a), the NS curve is directly on top of the *Actual* CDF, indicating that NS accurately estimates the true degree distribution as observed in $G$. However, in Figure 3(b), the NS curve is skewed upward, indicating that NS underestimates the sampled degree distribution in $G_s$. On the other hand, in Figure 3(a), ES, FFS, and ES-i all overestimate the true degree distribution in $G$, which is expected because they are biased to selecting higher-degree nodes. At the same time, in Figure 3(b), ES and FFS both underestimate the sampled degree distribution $G_s$. In contrast to other sampling




(a) Original degree in $G$, $k$ (b) Sampled degree in $G_s$, $k^s$

Fig. 3. Illustration of original degrees (in $G$) vs. sampled degrees (in $G_s$) for subgraphs selected by NS, ES, ES-i, and FFS on the CondMAT network.

Table II. Characteristics of Network Datasets

| Graph | Nodes | Edges | Weak Comps. | Avg. Path | Density | Global Clust. |
|---|---|---|---|---|---|---|
| HepPH | 34,546 | 420,877 | 61 | 4.33 | $7 \times 10^{-4}$ | 0.146 |
| CondMAT | 23,133 | 93,439 | 567 | 5.35 | $4 \times 10^{-4}$ | 0.264 |
| Twitter | 8,581 | 27,889 | 162 | 4.17 | $7 \times 10^{-4}$ | 0.061 |
| Facebook | 46,952 | 183,412 | 842 | 5.6 | $2 \times 10^{-4}$ | 0.085 |
| Flickr | 820,878 | 6,625,280 | 1 | 6.5 | $1.9 \times 10^{-5}$ | 0.116 |
| LiveJournal | 4,847,571 | 68,993,773 | 1876 | 6.5 | $5.8 \times 10^{-6}$ | 0.288 |
| Youtube | 1,134,890 | 2,987,624 | 1 | 5.29 | $2.3 \times 10^{-6}$ | 0.006 |
| Pokec | 1,632,803 | 30,622,564 | 1 | 4.63 | $1.2 \times 10^{-5}$ | 0.047 |
| Web-Stanford | 281,903 | 2,312,497 | 1 | 6.93 | $2.9 \times 10^{-5}$ | 0.008 |
| Email-Univ | 214,893 | 1,270,285 | 24 | 3.91 | $5.5 \times 10^{-5}$ | 0.002 |

methods, ES-i comes closer to replicating the original degree distribution of $G$ in $G_s$. This is from the combination of node selection bias (toward high-degree nodes) with further augmentation through graph induction, which help ES-i to compensate for the underestimation caused by sampling subgraphs.

### 5.3. Experimental Evaluation

In this section, we present results evaluating the various sampling methods on static graphs. We compare the performance of our proposed algorithm ES-i to other algorithms from each class (as discussed in Section 2.3): NS, ES, and FFS. We compare the algorithms on seven different real-world networks. We use online social networks from Facebook in the city of New Orleans [Viswanath et al. 2009] and Flickr [Gleich 2012]. We use a social media network drawn from Twitter, corresponding to users tweets surrounding the United Nations Climate Change Conference in Copenhagen, December 2009 (#*cop*15) [Ahmed et al. 2010a]. Also, we use a citation graph from ArXiv HepPh and a collaboration graph from CondMAT [SNAP 2013]. We consider an email network Email-Univ that corresponds to a month of email communication collected from Purdue University mail-servers [Ahmed et al. 2012a]. Finally, we compare the methods on a large social network from LiveJournal [Leskovec] with 4 million nodes (included only at the 20% sample size). Table II provides a summary of the global statistics of the network datasets we use.

Here, we discuss the experimental results. For each experiment, we applied the sampling methods to the full network and sampled subgraphs over a range of sampling

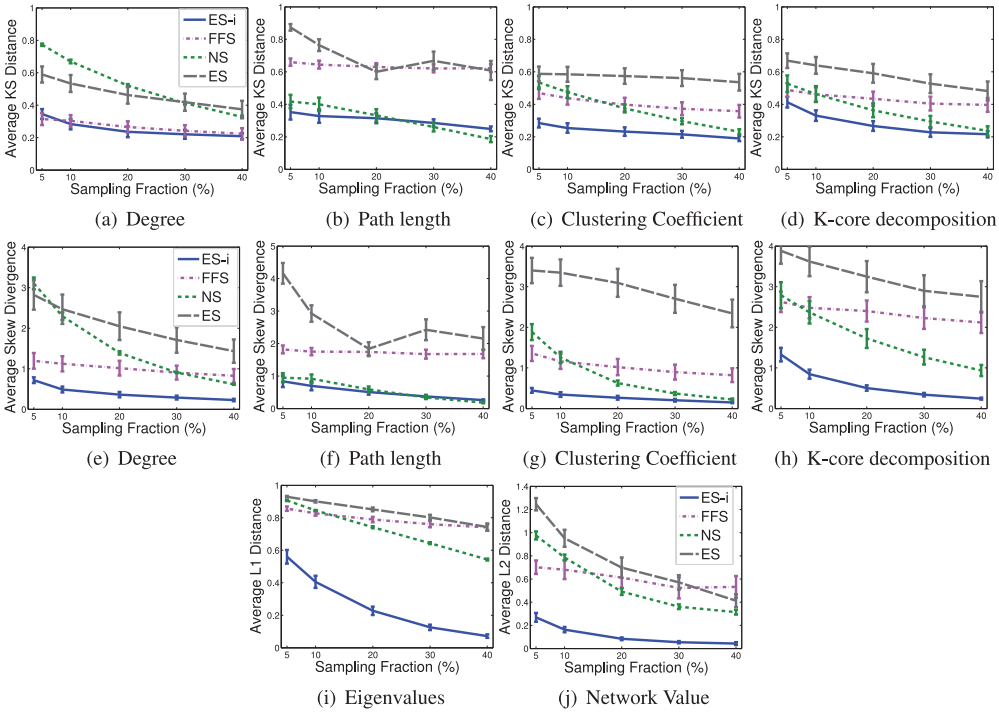



Fig. 4. (a–d) Average KS distance, (e–h) average skew divergence, and (i–j) average L1 and L2 distance respectively, across six datasets.

fractions $\phi = [5\%, 40\%]$. For each sampling fraction, we report the average results over ten different trials.

*5.3.1. Distance Metrics.* Figures 4(a)–4(d) show the average KS statistic for degree, path length, clustering coefficient, and k-core distributions on the six datasets. Generally, ES-i outperforms the other methods for each of the four distributions. FFS performs similar to ES-i in the degree distribution; however, it does not perform well for path length, clustering coefficient, and k-core distributions. This implies that FFS can capture the degree distribution but not connectivity between the sampled nodes. NS performs better than FFS and ES for path length, clustering coefficient, and k-core statistics but not for the degree statistics. This is due to the uniform sampling of the nodes that makes NS more likely to sample fewer high-degree nodes (as discussed in 5.2). Clearly, as the sample size increases, NS is able to select more nodes, and thus the KS statistic decreases. ES-i and NS perform similarly for path length distribution. This is because they both form a fully induced subgraph out of the sampled nodes. Since induced subgraphs are more connected, the distance between pairs of nodes is generally smaller.

In addition, we also used skew divergence as a second evaluation measure. Figures 4(e)–4(h) show the average skew divergence statistic for degree, path length, clustering coefficient, and k-core distributions on the six datasets. Note that skew divergence computes the divergence between the sampled and the real distributions on the entire support of the distributions. While the skew divergence is similar to KS statistic in that it still shows ES-i outperforming the other methods, it also shows significant gains in some cases, indicating that ES-i produces samples that capture the entire distribution more accurately.

Table III. Comparison of *max-core-number* at the 20% Sample Size for ES-i, NS, ES, FFS versus the Original Value in *G*

| Graph | Real max core no. | ES-i | NS | ES | FFS |
|---|---|---|---|---|---|
| HepPH | 30 | 23* | 8 | 2 | 4 |
| CondMAT | 25 | 20* | 7 | 2 | 6 |
| Twitter | 18 | 18* | 5 | 2 | 3 |
| Facebook | 16 | 14* | 4 | 2 | 3 |
| Flickr | 406 | 406* | 83 | 21 | 7 |
| LiveJournal | 372 | 372* | 84 | 6 | 7 |
| Email-UNIV | 47 | 46* | 15 | 3 | 7 |

Finally, Figures 4(i) and 4(j) show the L1 and L2 distances for eigenvalues and network values, respectively. Clearly, ES-i outperforms all the other methods that fail to improve their performance even when the sample size is increased up to 40% of the full graph.

*5.3.2. Distributions.* Although the distance measures are important to quantify the divergence between the sampled and the real distributions, by analyzing only the distance measures it is unclear whether the sampled statistics are an overestimate or underestimate of the original statistics. Therefore, we plot the distributions for all networks at the 20% sample size. We choose the 20% as a representative sample size; however, we note that the same conclusions hold for the other sample sizes. Note that we plot the *Complementary Cumulative Distribution* (*CCDF*) for degree and k-core distributions, *CDF* for path length and clustering coefficient distribution, and we plot eigenvalues and network values versus their rank. Appendix A includes Figures 13–19, which show the distributions for all networks. Here, we discuss the main findings we can observe from inspecting the distributions.

*5.3.2.1. Degree Distribution.* Across all networks, ES-i captures the tail of the degree distribution (high-degree nodes) better than NS, ES, and FFS. However, ES-i underestimates the low- degree nodes for Twitter, Email-Univ, and Flickr. FFS and NS capture a large fraction of low-degree nodes but they fail to capture the high-degree nodes.

*5.3.2.2. Path Length Distribution.* ES-i preserves the path length distribution of HepPH, CondMAT, and LiveJournal; however, it underestimates the distributions of Twitter, Email-Univ, and Flickr. Conversely, NS overestimates the distributions of HepPH, CondMAT, and LiveJournal but successfully preserves the distributions of the other datasets.

*5.3.2.3. Clustering Coefficient Distribution.* ES-i generally captures the clustering coefficient more accurately than other methods. While ES-i underestimates the low clustering coefficients, particularly in Email-Univ, and Flickr, the other methods fail to capture the clustered structures in almost all the datasets.

*5.3.2.4. K-Core Distribution.* Similar to the previous statistics, ES-i nicely preserves the distribution of core sizes for HepPH, CondMAT, and Facebook, but it overestimates the core structures of the other datasets. On the other hand, NS, ES, and FFS generally fail to capture the core structures for the majority of the datasets (with the exception of Flickr). In addition to the distribution of the core sizes, we compared the *max-core* number in the sampled graphs to their real counterparts for the 20% sample size (Table III). Note that the *max-core number* is the maximum value of $k$ in the k-core distribution. In contrast to ES-i, the *max-core number* in samples for NS, ES, and FFS is consistently an order of magnitude smaller than the real *max-core number*. This indicates that NS, ES, and FFS do not preserve the local density in the sampled subgraph structures.

*5.3.2.5. Eigenvalues.* The NS, ES, and FFS methods generally fail to approximate the eigenvalues of the original graph in the sample. In contrast, ES-i accurately approximates the eigenvalues of TWITTER, EMAIL-UNIV, FLICKR, and LIVEJOURNAL and closely approximates the eigenvalues of HEPPH, CONDMAT, and FACEBOOK (at 20% sample size). By the *interlacing* theorem of the eigenvalues of induced subgraphs [Haemers 1995], the eigenvalues of ES-i in $G_s$ can be used to estimate bounds on their counterparts in the input graph $G$: $\lambda_i \leq \mu_i \leq \lambda_{i+(N-n)}$ such that $\mu_i$ is the $i^{th}$ eigenvalue of $G_s$, and $\lambda_i$ is the $i^{th}$ eigenvalue of $G$.

*5.3.2.6. Network Values.* Similar to the other graph measures, ES-i more accurately approximates the network values of the graph compared to other methods.

*5.3.3. Comparison to Metropolis Graph Sampling.* Metropolis sampling has been used frequently to improve the quality of collected samples. The main idea of Metropolis graph sampling is to draw a sample from the sample space $X$ with a particular density, such that *good* samples will be sampled more frequently than bad ones. In Hubler et al. [2008], the authors used Markov chain sampling to improve the quality of uniform NS. Their algorithm starts with an initial sample collected by NS, then the sampled nodes are replaced randomly such that that the new sampled subgraph $G_s$ will better match the properties of the full graph $G$ (e.g., degree distribution). Clearly, this method requires computing the properties of the full graph $G$ in advance, and this requirement might be hard to satisfy when graphs are too large. In Ahmed et al. [2011], we found that "bad" nodes (i.e., ones that do not improve the match) are sampled more frequently than "good" ones as the sample size increases (typically $\geq$ 1,000 nodes), which makes it difficult for the method to converge. Consequently, this method produces graphs with properties that are comparable to node sampling.

Recently, Neighbor Reservoir Sampling (NRS) has been proposed in Lu and Bressan [2012] to sample *connected* subgraphs. The key idea is to start by an initial sample $G_s$ (chosen by random-walk approaches), then nodes in $G_s$ are randomly replaced by other potential nodes (chosen from the neighbors of $G_s$), such that the new sample subgraph is connected (i.e., $G_s$ has a single component). Random-walk methods are generally biased to high-degree nodes, and thus NRS helps to reduce their bias by randomly replacing nodes that were initially selected by random-walk methods.

In Table IV, we compare ES-i to NRS for TWITTER, FACEBOOK, HEPPH, and CONDMAT. Note that the results are the average of the 20% and 30% sample sizes. We show the average KS distance computed over the distributions of degree, path length, clustering, and k-core. We observe that NRS outperforms ES-i for sparse graphs (TWITTER and FACEBOOK). This shows that NRS is indeed effective to reduce the bias of random-walk sampling approaches. However, ES-i outperforms NRS for dense graphs (HEPPH and CONDMAT). We also compare the L1/L2 distances computed for the distribution of eigenvalues and network values, respectively, and we observe that ES-i outperforms NRS for the four datasets. We tried to use NRS for sampling very large graphs, but the method was slow and did not converge for any of our larger datasets (within a limit of 2 hours of running time). Further, we test the effect of graph induction on FFS-i in Table IV. ES-i outperforms FFS-i on the average KS distance computed over the distributions of degree, path length, clustering, and k-core. For the L1/L2 distances, ES-i and FFS-i performs comparably.

***Summary**—We summarize the main empirical conclusions in this section:*

(*1*) *Sampled subgraphs collected and constructed by ES-i accurately preserve a range of network statistics that capture both local and global distributions of the graph structure.*

Table IV. Comparison of ES-i to Neighbor Reservoir (NRS) and Forest Fire Sampling (FFS-i) with Graph Induction

| Data | ES-i | NRS | FFS-i |
|---|---|---|---|
| | Average KS dist. (DEG, PL, CLUST, KCORE) | | |
| TWITTER | 0.2801 | 0.0631 | 0.3009 |
| FACEBOOK | 0.1918 | 0.1054 | 0.2650 |
| HEPPH | 0.0895 | 0.3321 | 0.1229 |
| CONDMAT | 0.1125 | 0.1918 | 0.1892 |
| | **0.1685** | 0.1731 | 0.2195 |
| | Average L1 dist. (EIGENVAL) | | |
| TWITTER | 0.1923 | 0.3323 | 0.2080 |
| FACEBOOK | 0.1647 | 0.4676 | 0.1671 |
| HEPPH | 0.3459 | 0.6512 | 0.3051 |
| CONDMAT | 0.2479 | 0.4812 | 0.1689 |
| | 0.2377 | 0.4831 | **0.2123** |
| | Average L2 dist. (NETVAL) | | |
| TWITTER | 0.0481 | 0.1029 | 0.0549 |
| FACEBOOK | 0.0787 | 0.2801 | 0.0759 |
| HEPPH | 0.1804 | 0.3637 | 0.2482 |
| CONDMAT | 0.0944 | 0.1750 | 0.0852 |
| | **0.1004** | 0.2304 | 0.1161 |

(*2*) *Due to its bias for selecting high-degree nodes, ES-i generally favors dense and clustered areas of the graph, which results in connected sample subgraphs—in contrast with other methods.*
(*3*) *NS, ES, and FFS generally construct more sparsely connected sample subgraphs.*

## 6. SAMPLING FROM STREAMING GRAPHS

In this section, we focus on how to sample a representative subgraph $G_s$ from $G$ when $G$ is presented as a stream of edges in no particular order. Note that in this work we focus on space-efficient sampling methods. Using the definition of a streaming graph sampling algorithm as discussed in Section 3, we now present streaming variants of different sampling algorithms from Section 2.3.

### 6.1. Algorithms

*6.1.1. Streaming Node Sampling.* One key problem with traditional node sampling (discussed in 2.3) is that the algorithm assumes that nodes can be accessed at random. In our stream setting, new nodes arrive into the system only when an edge that contains the new node is observed in the system. It is therefore difficult to identify which $n$ nodes to select *a priori*. To address this, we utilize the idea of reservoir sampling [Vitter 1985] to implement a streaming variant of node sampling (see Algorithm 2).

The main idea is to select nodes uniformly at random with the help of a uniform random hash function – $h(v_i) \sim \text{Uniform}(0, 1)$. A uniform random hash function defines a true random permutation on the nodes in the graph, meaning that any node is equally likely to be the node with the minimum value. Specifically, we keep track of the nodes with the $n$ smallest hash values in the graph. Nodes are only added to the sample if their hash values are among the top-$n$ minimum hashes seen thus far in the stream. Any edge that has both vertices already in the reservoir is automatically added to the original graph. Since the reservoir is finite, a node with smaller hash value may arrive late in the stream and replace a node that was sampled earlier. In this case, all edges incident to the replaced/dropped node will be removed from the sampled subgraph. Once the reservoir is filled up to $n$ nodes, it will remain at $n$ nodes, but since selection is based on the hash values, nodes will be dropped and added as the algorithm samples

**ALGORITHM 2:** Streaming Node Sampling NS($n$, $S$)

**Input**: Sample Size $n$, Graph Stream $S$
**Output**: Sampled Subgraph $G_s = (V_s, E_s)$

$V_s = \emptyset, E_s = \emptyset$
$h$ is fixed uniform random hash function
$t = 1$
**for** $e_t$ *in the graph stream* $S$ **do**
  $(u, v) = e_t$
  **if** $u \notin V_s$& $h(u)$ *is top-n min hash* **then**
    $V_s = V_s \cup u$
    Let $w \in V_s$ be the node s.t. $h(w)$ is no longer a top-$n$ min hash
    $V_s = V_s - \{w\}$; Remove all edges incident on $w$ from $E_s$
  **if** $v \notin V_s$& $h(v)$ *is top-n min hash* **then**
    $V_s = V_s \cup v$
    Let $w \in V_s$ be the node s.t. $h(w)$ is no longer a top-$n$ min hash
    $V_s = V_s - \{w\}$; Remove all edges incident on $w$ from $E_s$
  **if** $u, v \in V_s$ **then**
    $E_s = E_s \cup e_t$
  $t = t + 1$

from all portions of the stream (not just the front). Therefore, it guarantees a uniformly sampled set of nodes from the graph stream.

*6.1.2. Streaming Edge Sampling.* Streaming edge sampling can be implemented similar to streaming node sampling. Instead of hashing individual nodes, we focus on using hash-based selection of edges (as shown in Algorithm 3). We use the approach that was first proposed in Aggarwal et al. [2011]. More precisely, if we are interested in sampling $m$ edges at random from the stream, we can simply keep a reservoir of the $m$ edges with minimum hash values. Thus, if a new edge streams into the system, we check if its hash value is less than the top-$m$ minimum hash value. If it is not, the edge is not selected; otherwise, it is added to the reservoir, replacing the edge with the previous highest hash value. One problem with this approach is that our goal is often in terms of sampling a certain number of nodes $n$. Since we use a reservoir of edges, determining

**ALGORITHM 3:** Streaming Edge Sampling ES($n$, $S$)

**Input**: Sample Size $n$, Edge Selection Size $m$, Graph Stream $S$
**Output**: Sampled Subgraph $G_s = (V_s, E_s)$

$V_s = \emptyset, E_s = \emptyset$
$h$ is fixed uniform random hash function
$t = 1$
**for** $e_t$ *in the graph stream* $S$ **do**
  $(u, v) = e_t$
  **if** $h(e_t)$ *is in top-m min hash* **then**
    $E_s = E_s \cup e_t$
    $V_s = V_s \cup \{u, v\}$
    Let $e_k \in E_s$ be the edge s.t. $h(e_k)$ is no longer a top-$m$ min hash
    $E_s = E_s - \{e_k\}$; Remove any nodes from $V_s$ that have no incident edges
  Iteratively remove edges in $E_s$ in decreasing order until $|V_s| = n$ nodes
  $t = t + 1$

the value of $m$ that provides $n$ nodes is difficult. The value may also vary throughout the stream, depending on which edges the algorithm ends up selecting. Note that sampling fraction could also be specified in terms of fraction of edges; the choice of defining it in terms of nodes is somewhat arbitrary in that sense. For our comparison purposes, we ensured that we chose a large enough $m$ such that the number of nodes was much higher than $n$, but later iteratively pruned out sampled edges with the maximum hash values until the target number of nodes $n$ was reached.

**ALGORITHM 4:** Streaming Breadth-First Search BFS($n$, $S$, $wsize$)

**Input**: Sample Size $n$, Graph Stream $S$, Window Size=$wsize$
**Output**: Sampled Subgraph $G_s = (V_s, E_s)$

$V_s = \emptyset$; $E_s = \emptyset$; $W = \emptyset$
Add the first $wsize$ edges to $W$
$t' = wsize$
Create a queue $Q$
// uniformly sample a seed node from $W$
$u = Uniform(V_W)$; $V_s = V_s \cup \{u\}$
**for** *$e_t$ in the graph stream $S$ starting at $t'$* **do**
  **if** $W.incident\_edges(u) = \emptyset$ **then**
    **if** $Q \neq \emptyset$ **then** $u = Q.dequeue()$
    **else** $u = Uniform(V_W)$
    **if** $u \notin V_s$ **then** $V_s = V_s \cup \{u\}$
  **else**
    Sample $e_s = (u, v)$ from $W.incident\_edges(u)$
    $E_s = E_s \cup e_s$
    $V_s = V_s \cup \{v\}$
    $W = W - \{e_s\}$
    Enqueue $v$ onto $Q$
  // Move the window $W$
  $W = W - \{e_{t-wsize}\}$; $W = W \cup \{e_t\}$
  **if** $|V_s| > n$ **then**
    Retain $[e] \subset E_s$ such that $[e]$ has $n$ nodes
    Output $G_s = (V_s, E_s)$
  $t = t + 1$

*6.1.3. Streaming Topology-Based Sampling.* We also consider a streaming variant of a topology-based sampling algorithm. Specifically, we consider a simple BFS-based algorithm (shown in Algorithm 4) that works as follows. This algorithm essentially implements a simple BFS on a sliding window of $w$ edges in the stream. In many respects, this algorithm is similar to the FFS algorithm. Just as in FFS, it starts at a random node in the graph and selects an edge, among all edges incident on that node within the sliding window, to burn (as in FFS parlance). For every edge burned, let $v$ be the incident node at the other end of the burned edge. We enqueue $v$ onto a queue $Q$ in order to get a chance to burn its incident edges within the window. For every new streaming edge observed, the sliding window moves one step, which means the oldest edge in the window is dropped and a new edge is added. (If that oldest edge was sampled, it will still be part of the sampled graph.) If, as a result of the sliding window moving one step, the node has no more edges left to burn, then the burning process will dequeue a new node from $Q$. If the queue is empty, the process jumps to a random node within the sliding window (just as in FFS). In this way, it does BFS as much as possible within a sliding window, with random jumps if there are no more edges left to explore.

Note that there may be other ways to implement a one-pass streaming approach to topology-based sampling, but since, to our knowledge, there are no streaming methods in the literature, we include this as a reasonable approximation for comparison. This algorithm has a similar problem as the edge sampling variant in that it is difficult to control the exact number of sampled nodes, and hence some additional pruning needs to be done at the end (see Algorithm 4).

*6.1.4. Partially Induced Edge Sampling (PIES).* We finally present our main algorithm called PIES that outperforms the earlier implementations of stream sampling algorithms. Our ES-i approach discussed in Section 5 outlines a sampling algorithm based on edge sampling concepts. A key advantage of using edge sampling is its bias toward high-degree nodes. This upward bias helps offset the downward bias (caused by subgraph sampling) to some extent. Afterward, forming the induced graph will help capture the connectivity among the sampled nodes.

Unfortunately, full graph induction in a streaming fashion is difficult because node selection and graph induction require at least two passes (when implemented in the obvious, straightforward way). Thus, instead of full induction of the edges between the sampled nodes, in a streaming environment, we can utilize *partial* induction and combine edge-based node sampling with the graph induction (as shown in Algorithm 5) in a single pass. The partial induction step induces the sample in the forward direction only. In other words, it adds an edge among a pair of sampled nodes if it occurs *after* both two nodes were added to the sample.

**ALGORITHM 5:** Partially Induced Edge Sampling PIES (Sample Size $n$, Stream $S$)

**Input**: Sample Size $n$, Graph Stream $S$
**Output**: Sampled Subgraph $G_s = (V_s, E_s)$

$V_s = \emptyset, E_s = \emptyset$
$t = 1$
**while** *graph is streaming* **do**
  $(u, v) = e_t$
  **if** $|V_s| < n$ **then**
    **if** $u \notin V_s$ **then** $V_s = V_s \cup \{u\}$
    **if** $v \notin V_s$ **then** $V_s = V_s \cup \{v\}$
    $E_s = E_s \cup \{e_t\}$
    $m = |E_s|$
  **else**
    $p_e = \frac{m}{t}$
    draw $r$ from continuous Uniform(0,1)
    **if** $r \leq p_e$ **then**
      draw $i$ and $j$ from discrete Uniform[1,$|V_s|$]
      **if** $u \notin V_s$ **then** $V_s = V_s \cup \{u\}$, drop node $V_s[i]$ with all its incident edges
      **if** $v \notin V_s$ **then** $V_s = V_s \cup \{v\}$, drop node $V_s[j]$ with all its incident edges
    **if** $u \in V_s$ *AND* $v \in V_s$ **then** $E_s = E_s \cup \{e_t\}$
  $t = t + 1$

PIES aims to maintain a dynamic sample while the graph is streaming by utilizing the same reservoir sampling idea we have used before. In brief, we add the first set of edges in the stream to a *reservoir* and then the rest of the stream is processed by randomly replacing existing records in the reservoir. PIES runs over the stream in a single pass and adds deterministically the first $m$ edges incident to $n$ nodes to the sample. Once it achieves the target sample size of $n$, then for any streaming edge, it

(probabilistically) adds the incident nodes to the sample by replacing other sampled nodes from the node set (selected uniformly at random). At each step, the algorithm will also add the edge to the sample if its two incident nodes are already in the sampled node set—producing a partial induction effect.

Next, we discuss the properties of PIES to illustrate its characteristics.

(1) *PIES is a two-phase sampling method.* A two-phase sampling method is a method in which an initial sample of units is selected from the population (e.g., the graph stream), and then a second sample is selected as a subsample of the first. PIES can be analyzed as a two-phase sampling method. The first phase samples edges (i.e., edge sampling) from the graph stream with probability $p_e = \frac{m}{t}$ if the edge is incident to at least one node that does not belong to the reservoir, where $t$ is the variable representing the time of the stream and $m$ is the number of initial edges in the reservoir. Also, an edge is sampled with probability $p_e = 1$ if the edge is incident to two nodes that belong to the reservoir. After that, the second phase samples a subgraph uniformly (i.e., node sampling) to maintain only $n$ nodes in the reservoir (i.e., all nodes in the reservoir are equally likely to be sampled).
(2) *PIES has a selection bias to high-degree nodes.* PIES is biased to high-degree nodes due to its first phase that relies on edge sampling. Naturally, edge sampling is biased toward high-degree nodes since they tend to have more edges compared to lower degree nodes.
(3) *PIES samples an induced subgraph uniformly from the subsampled edge stream $E'(t)$ at any time $t$ in the stream.* At any time $t$ in the graph stream $E$, PIES subsamples $E(t)$ to $E'(t)$ (such that $|E'(t)| \leq |E(t)|$). Then, PIES samples a uniform induced subgraph from $E'(t)$, such that all nodes in $E'(t)$ have an equal chance to be selected. Now that we have discussed the main properties of PIES, it can be easily adapted depending on a specific choice of the network sampling goal.

### 6.2. Experimental Evaluation

In this section, we present results of sampling from streaming graphs observed as an arbitrarily ordered sequence of edges. The experimental setup is similar to what we used in Section 5.3. We compare the performance of our proposed algorithm PIES to the streaming implementation of NS, ES, and BFS sampling methods. Note that we implement BFS using a sliding window of 100 edges.

Similar to the experiments in Section 5.3, we report average performance over 10 different runs. To assess algorithm variation based on the edge sequence ordering, we randomly permute the edges in each run (while ensuring that all sampling methods use the same sequential order). Note that all streaming algorithms run in $O(|E|)$, and therefore, in addition to the datasets we used in Section 5.3, we also compare the methods on large networks with millions of nodes/edges from YOUTUBE, POKEC, and WEB-STANFORD [SNAP 2013]. Note that large datasets are included only for 20% sample size as they take longer running time.

*6.2.1. Distance Metrics.* Figures 5(a)–5(d) show the average KS statistic for degree, path length, clustering coefficient, and k-core distributions as an average over the six datasets (similar to Section 5.3). PIES outperforms all other methods for capturing the degree distribution. NS performs almost as well as PIES for path length, clustering coefficient, and k-core distributions. As we explained in Section 6, PIES is biased to high-degree nodes (due to its first phase) compared to NS. Both BFS and ES perform the worst among the four methods. This shows that the limited observability of the graph structure using a window of 100 edges does not facilitate effective BFS. While increasing the window size may help improve the performance of BFS, we did not

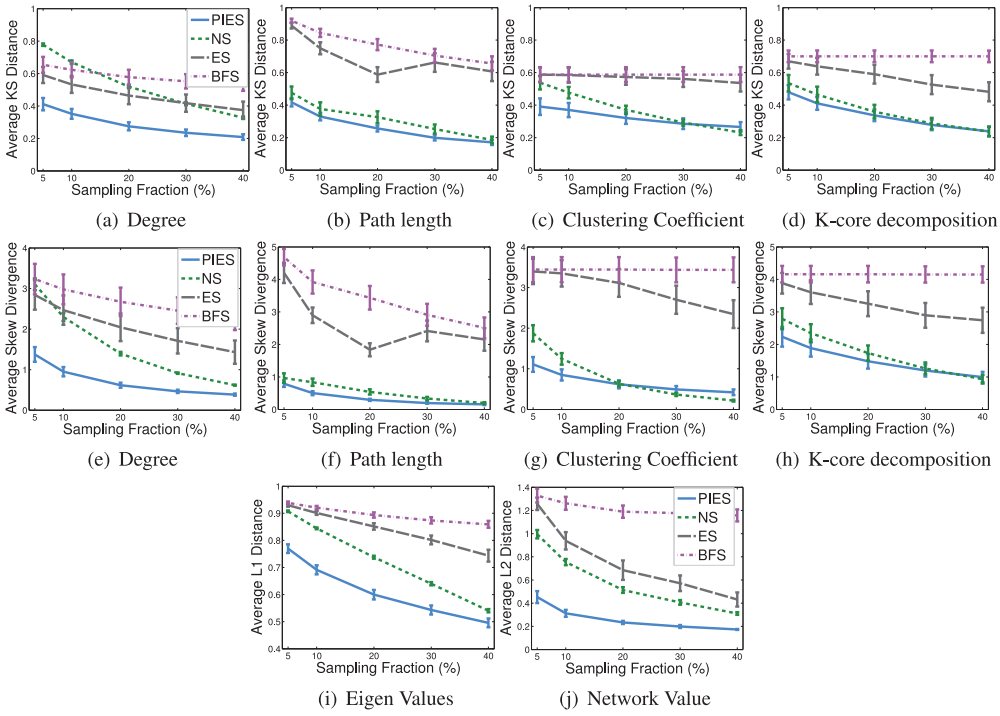



Fig. 5. (a–d) KS distance, (e–h) average skew divergence, and (i–j) average L1 and L2 distance respectively, averaged across six datasets.

explore this as our focus was primarily on space-efficient sampling. Figures 5(e)–5(h) show the skew divergence results are similar to that of the KS statistic.

Finally, Figures 5(i)–5(j) show the L1 and L2 distance for eigenvalues and network values, respectively. PIES outperforms all other methods. However, even though PIES performs the best, the distance is almost 50% for the eigenvalues. This implies PIES is not suitable for capturing the eigenvalues of the graph.

*6.2.2. Distributions.* We plot the distributions of the network statistics at the 20% sample size. Appendix A includes Figures 20–29, the plots for all the distributions across all datasets. Here, we discuss the main findings we can observe from inspecting the distributions.

*6.2.2.1. Degree Distribution.* We observe across the all datasets, PIES outperforms the other methods for FACEBOOK, TWITTER, EMAIL-UNIV, FLICKR, LIVEJOURNAL, YOUTUBE, POKEC, and WEB-STANFORD. However, PIES only performs slightly better than NS for HEPPH and CONDMAT. This behavior appears to be related to the specific properties of the network datasets themselves. HEPPH and CONDMAT are more clustered and dense compared to other graphs used in the evaluation. We will discuss the behavior of the sampling methods for dense versus sparse graphs later in this section.

*6.2.2.2. Path Length Distribution.* PIES preserves the path length distribution of FACEBOOK, TWITTER, EMAIL-UNIV, FLICKR, LIVEJOURNAL, YOUTUBE, POKEC, and WEB-STANFORD, however, it overestimates the shortest path for HEPPH and CONDMAT.

*6.2.2.3. Clustering Distribution.* PIES generally underestimates the clustering coefficient in the graph by missing some of the clustering surrounding the sampled nodes.

Table V. Comparison of *max-core-number* at the 20% Sample Size for PIES, NS, ES, BFS versus the Original Value in *G*

| Graph | Real max core no. | PIES | NS | ES | BFS |
|---|---|---|---|---|---|
| HepPH | 30 | 8* | 8* | 2 | 1 |
| CondMAT | 25 | 7* | 7* | 2 | 1 |
| Twitter | 18 | 7* | 4 | 3 | 1 |
| Facebook | 16 | 6* | 4 | 2 | 1 |
| Flickr | 406 | 166* | 81 | 19 | 1 |
| LiveJournal | 372 | 117* | 82 | 5 | 1 |
| Youtube | 51 | 22* | 12 | 5 | 2 |
| Pokec | 47 | 19* | 13 | 2 | 1 |
| Web-Stanford | 71 | 33* | 19 | 3 | 3 |
| Email-Univ | 47 | 22* | 15 | 3 | 1 |

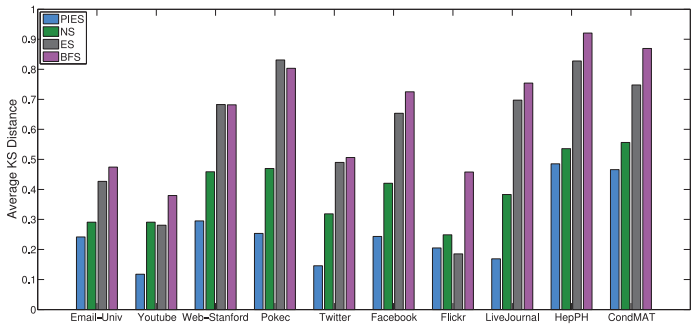



Fig. 6. Average KS Statistics for different networks (sorted in increasing order of clustering/density from left to right).

This behavior is more clear in HepPH, CondMAT, and Web-Stanford because they are more clustered initially.

*6.2.2.4. K-Core Distribution.* Similar to the previous distributions considered above, PIES outperforms the other methods for Facebook, Twitter, LiveJournal, Youtube, Pokec, and Web-Stanford. For HepPH and CondMAT, PIES performs almost as good as NS. In addition to the distribution of the core sizes, we compared the *max-core number* in the sampled subgraphs to their real counterparts for the 20% sample size (Table V).

*6.2.2.5. Eigenvalues.* Although PIES captures the eigenvalues better than ES and BFS, its eigenvalues are orders of magnitude smaller than the real graph's eigenvalues. This shows that none of the streaming algorithms accurately captures the eigenvalues of the full graph (compared to ES-i in Section 5).

*6.2.2.6. Network Values.* PIES accurately estimates the network values of the majority of the datasets, in contrast to other methods.

*6.2.3. Analysis of Dense versus Sparse Graphs.* Further to the discussion of the distributional results, we note that PIES is more accurate for sparse, less clustered graphs. To illustrate this, we report the performance of the stream sampling methods for each network in Figure 6, sorted from left to right in ascending order by clustering coefficient and density. Note that the bars represent the KS statistic (averaged over degree, path length, clustering, and k-core) for the 20% sample size. Clearly, the KS statistic for all methods increases as the graph becomes more dense and clustered. PIES maintains a KS distance of approximately $\leq 26\%$ for eight out of 10 networks. These results

Table VI. Average Probability of Isolated Nodes for NS and PIES Methods, 20% Sample Size

| Graph | PIES | NS |
|---|---|---|
| HepPH | 0.05 | 0.15 |
| CondMAT | 0.14 | 0.36 |
| Twitter | 0.15 | 0.51 |
| Facebook | 0.13 | 0.43 |
| Flickr | 0.14 | 0.56 |
| LiveJournal | 0.06 | 0.36 |
| Youtube | 0.22 | 0.63 |
| Pokec | 0.01 | 0.21 |
| Web-Stanford | 0.08 | 0.32 |
| Email-UNIV | 0.07 | 0.51 |

indicate that PIES performs better in networks that are generally sparse and less clustered. This interesting result shows that PIES will be more suitable to sample rapidly changing graph streams that have lower density and less clustering—which is likely to be the case for many large-scale dynamic communication and activity networks.

Moreover, we analyzed the number of isolated nodes for both NS and PIES in Table VI. Since both NS and PIES sample nodes independently, it is expected that their sampled subgraph contains some nodes with zero degree (i.e., isolated nodes). This implies that PIES carries isolated nodes in the reservoir as the graph streams by. From the process of PIES, we know that each time a new edge is sampled from the stream, its incident nodes replace randomly selected nodes from the reservoir. This random replacement policy could replace high-degree nodes while other isolated nodes remain in the reservoir. With this observation in mind, we propose a modification for PIES such that a newly added node replaces the node with minimum degree that has stayed in the reservoir the longest amount of time without acquiring more edges. This strategy favors retaining high-degree nodes over isolated and/or low-degree nodes in the sample. We show the results of this modification in Appendix A, Tables IX and VIII, which compare the KS distance, and L1/L2 distances, respectively, for each dataset (averaged over the two reasonable sample sizes 20% and 30%). Note that we refer to the modification of PIES as "PIES (MIN)." The results show that modifying PIES in this manner achieves better results for dense graphs such as HepPH and CondMAT. Note that there is a tradeoff between preventing nodes from being replaced (based on recency of adding nodes in the sample) and degree. We handle the tradeoff between degree and recency by keeping two arrays, one for the recent (not to be replaced) nodes, and the other for the oldest (can be replaced) nodes. When the number of the oldest nodes becomes less than 50% of the total sample size, we merge the two arrays, consider all nodes amenable to replacement, and replace the node with minimum degree.

### 6.3. Sampling from Multigraph Streams

In the previous experiments, we focused on sampling a representative subgraph from *simple* graph streams; that is, graph streams in which edges appear only once, similar to the problem definition studied in Buriol et al. [2006]. In time-evolving graph streams, there are often two characteristics to study: (1) graph structure, and (2) graph dynamics.

Studying simple graph streams is particularly focused on the *structure* of the streaming graph $G$ (i.e., topological properties such as degree, clustering) while ignoring the *dynamics* that take place on top of the graph structure. These dynamics describe the strength (frequency) of communications between pairs of nodes, as well as the strength of individual nodes over time. Therefore, we also study the problem of sampling from *multigraph* streams; that is, graph streams in which edges may appear more than once, similar to the problem definition studied in Cormode and Muthukrishnan [2005].

Table VII. Characteristics of Multigraph Datasets

| Graph | Nodes | Edges | Avg. Node Strength | Avg. Edge Weight | Density | Global Clust. |
|---|---|---|---|---|---|---|
| FACEBOOK-CITY | 46,952 | 855,542 | 37.4 | 4.7 | $4 \times 10^{-4}$ | 0.085 |
| FACEBOOK-UNIV | 49,825 | 1,518,155 | 60.9 | 4.2 | $6 \times 10^{-4}$ | 0.060 |
| TWITTER-COP15 | 8,578 | 45,771 | 10.7 | 1.6 | $6 \times 10^{-4}$ | 0.061 |
| RETWEET-POL | 18,470 | 61,157 | 6.6 | 1.3 | $2 \times 10^{-4}$ | 0.027 |
| INFECTIOUS-SOCIO | 10,972 | 415,912 | 75.8 | 9.3 | $3 \times 10^{-3}$ | 0.44 |

Formally, let $G = (V, E)$ be a snapshot of streaming undirected graph of time length $T$, such that $E = \{e_t, \forall t \in [1, T]\}$. We denote by $w$ the weight of an edge $e = (u, v)$; that is, the number of times an edge $e$ appeared in the stream, and we denote by $s$ the strength of a node $v$, equal to the sum of the weights of the incident edges to node $v$, that is, $s = \sum_{(u,v)\in E} w$. We also denote by $k$ the degree of node $v$; that is, the number of unique neighbors of $v$. These functions were previously used by Gautreau et al. [2009].

Table VII describes the characteristics of five real multigraph datasets. We use graph streams (with real timestamps), from Facebook wall communications in the city network of New Orleans—FACEBOOK-CITY graph [Mislove et al. 2007], from Facebook wall communications in the university network of Purdue University—FACEBOOK-UNIV graph [Ahmed et al. 2010a], from the Twitter hashtag #*cop*15—TWITTER-COP15 graph [Ahmed et al. 2010a], from the retweets collected from Twitter hashtags with political content—RETWEET-POL graph [Conover et al. 2011], and from face-to-face interactions at the INFECTIOUS exhibition in Dublin, Ireland—INFECTIOUS-SOCIO graph [Isella et al. 2011].

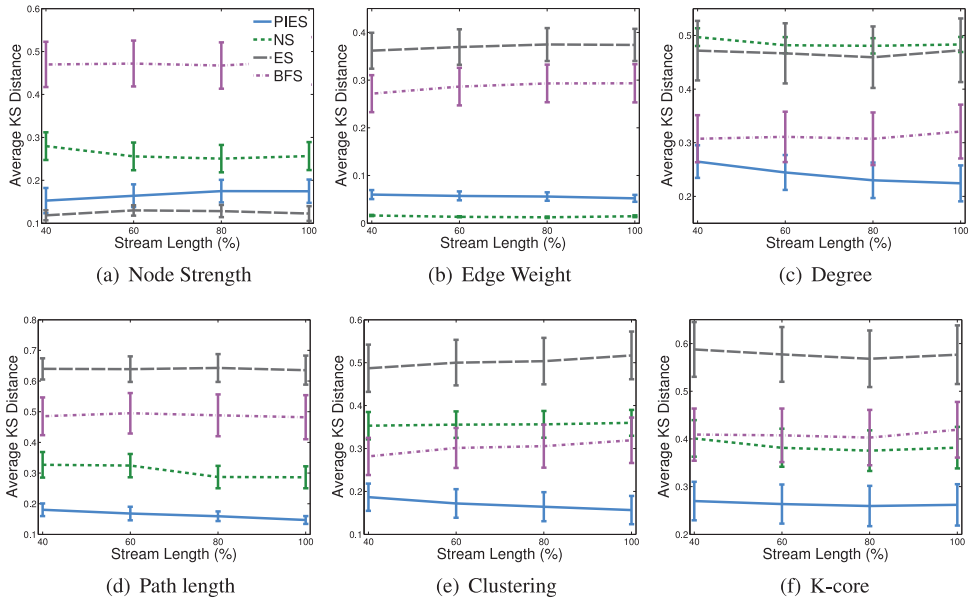
In the experimental setup, we used the *real* timestamps of the datasets, and we ran 10 experiments for each dataset. In each experiment, we sample 20% of the total number of nodes that appear in the stream as it is progressing. We evaluate the quality of the sample at different time points in the stream, from 40% to 100% of the stream length. Note that the stream length is the number of edges in the streaming graph ordered by timestamps. For example, if the total number of edges in the stream is 1,000 edges, we evaluate using the graph sampled from the first 400, 600, 800, and 1,000 edges, respectively.

*6.3.1. KS Statistics.* Figure 7 shows the average KS statistics over the five datasets, while the stream is progressing. Figures 7(a) and 7(b) show the KS statistics of the distributions of node strength and edge weight, respectively. The results show that PIES performs consistently well for both node strength and edge strength. In contrast, NS and ES each only perform well in one of the two properties.

In addition to the dynamics properties, we also evaluate the quality of the sample using the structural properties. Figures 7(c)–7(f) show the KS statistics for the distributions of degree, path length, clustering, and k-core. These results show that PIES captures both the structure and dynamic properties as the stream is progressing.

*6.3.2. Distributions.* Figure 8 shows the CCDF distributions of node strength and edge weight for a 20% sample of the FACEBOOK-CITY and INFECTIOUS-SOCIO graphs, constructed using the entire stream (i.e., 100% of the stream length). The results show that ES captures the distribution of node strength but overestimates the distribution of edge weight. Also, the results show that NS captures the distribution of edge weight but underestimates the distribution of node strength. However, PIES again balances the estimation of both node strength and edge weight, producing a more accurate sample overall. Notably, BFS performs significantly worse than the other methods.

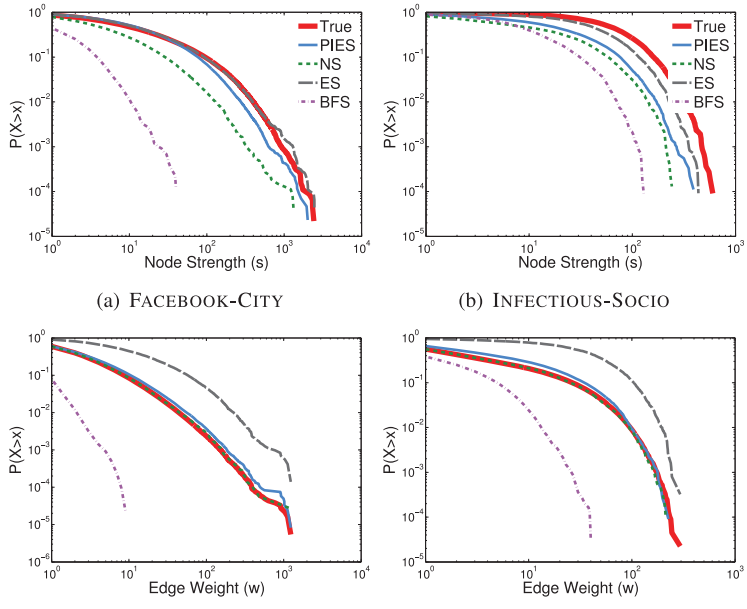
These results are not surprising since ES (unlike NS) is naturally biased toward edges with high weights because they appear more frequently in the stream. We found




(a) Node Strength (b) Edge Weight (c) Degree

(d) Path length (e) Clustering (f) K-core

Fig. 7. Average KS distance of across five datasets vs. percentage of stream length.




(a) FACEBOOK-CITY (b) INFECTIOUS-SOCIO

(c) FACEBOOK-CITY (d) INFECTIOUS-SOCIO

Fig. 8. (a–b) Distributions of node strength (s) and (c–d) Distributions of edge weight (w), at 20% sample for FACEBOOK-CITY and INFECTIOUS-SOCIO graphs.

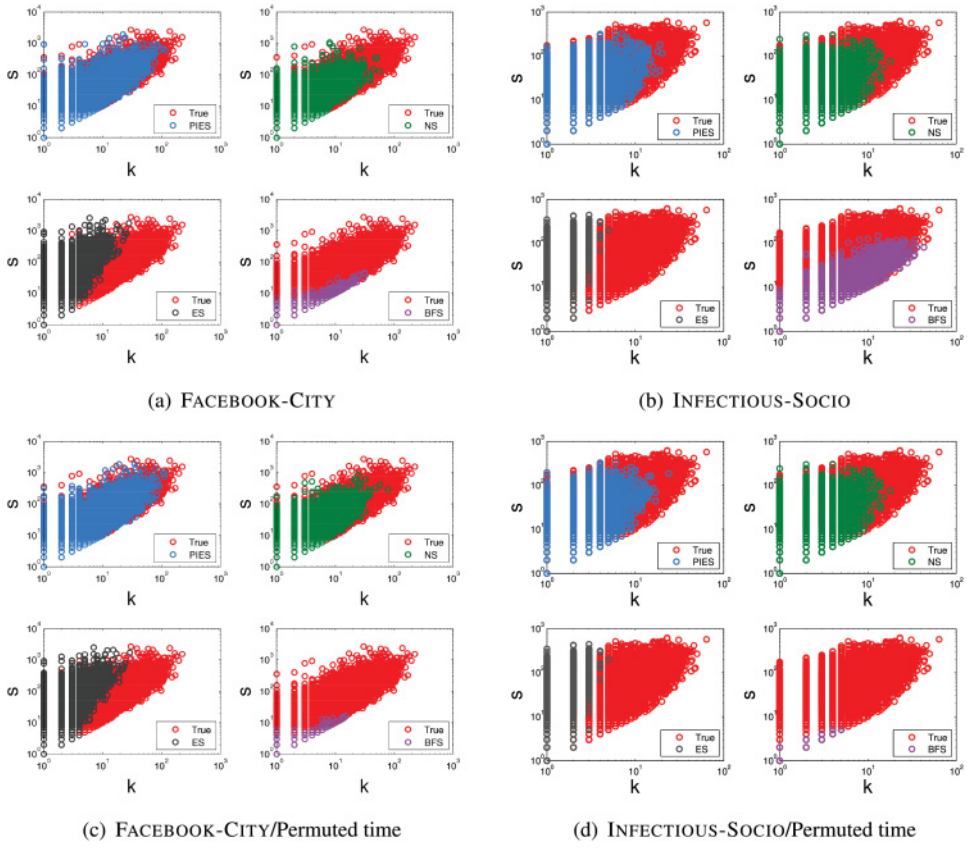



Fig. 9. Node strength ($s$) versus degree ($k$) for PIES, NS, ES, and BFS at 20% sample size. (a–b) Streams with original timestamp ordering (c–d) Streams with permuted ordering.

that in all the datasets, there is a strong positive correlation between the node strength and its degree (e.g., $corr(k, s) = 0.76$ in FACEBOOK-CITY). Due to this correlation, sampling methods that are biased to high-degree nodes also have a greater chance of capturing node strength compared to methods that are biased to low-degree nodes. On the other hand, the bias toward high-degree nodes also may result in over-estimation of the edge weights. This is clearly the case for ES. Similar results can be seen in the other datasets, plotted in Appendix A, Figure 30. We also note that these results are consistent at different points in the stream.

*6.3.3. Node Strength versus Degree.* We further investigate the interplay between graph dynamics and structure by plotting node degree versus node strength, comparing the samples to the original data. Figures 9(a) and 9(b) show scatter plots for FACEBOOK-CITY and INFECTIOUS-SOCIO graphs, respectively (for a 20% sample from the total stream). The plots for other datasets are included in Appendix A, Figure 31.

The results show that PIES outperforms all other methods when it comes to capturing the correlation between node degree and strength. NS performs similar to PIES but generally fails to capture high-degree nodes. Clearly, ES is biased to sampling frequents edges with incident nodes that may or may not have a high degree. Therefore, we observe that ES captures the strength of the nodes but fails to capture their degree. In contrast, BFS captures the degree but fails to capture the strength of the nodes.

From this experiment, we conclude that the choice of stream sampling method can have a significant impact on the accuracy of sampling multigraph streams. This is because the *probability* of sampling nodes depends not only on the graph structure (i.e., node degree), but also on graph dynamics (i.e., node strength and edge weight). Therefore, methods that are biased to sampling frequent edges (e.g., ES) will capture more of the dynamics but less of the structure. At the same time, methods that sample nodes (nearly) uniformly (e.g., NS, PIES) are able to capture both the graph structure and its dynamics. We also found that graphs that are more clustered (e.g., INFECTIOUS-SOCIO) are generally more difficult to sample accurately compared to less clustered graphs (e.g., TWITTER-COP15).

Randomization tests provide a way to test the effects of time on the graph structure and degree/strength correlation. To investigate the effects of time-clustering on the various sampling methods, we permuted the timestamps in the original stream before sampling. If node interactions are grouped together in small intervals of time, then the permutation will destroy this aspect of the stream. Figures 9(c) and 9(d) show plots of node degree versus node strength for FACEBOOK-CITY and INFECTIOUS-SOCIO when sampled from time-permuted graphs. The results show no impact on the accuracy of PIES, NS, and ES. However, BFS is significantly impacted by the time permutation, which shows that BFS performance is highly dependent on the temporal clustering of edges in the stream. We also found that the effect of time clustering is more significant in INFECTIOUS-SOCIO and less significant in the Twitter communication graphs we studied.

***Summary***—*We summarize the main empirical conclusions in this section:*

(*1*) *Sampled subgraphs constructed by PIES accurately preserve many network statistics and distributions (*e.g.*, degree, path length, and k-core).*
(*2*) *PIES produces better samples when the graph is more sparse and less clustered (*e.g.,TWITTER *and* LIVEJOURNAL *datasets).*
(*3*) *PIES can be easily adapted to reduce the number of isolated nodes in the sample by modifying the reservoir replacement mechanism (*i.e.*, PIES(MIN)).*
(*4*) *PIES(MIN) more accurately preserves the properties of dense graphs as well as certain statistics (*e.g.*, eigenvalues) that are difficult to capture with PIES.*
(*5*) *PIES is better able to capture both the dynamics and structure of multigraph streams compared to other methods.*
(*6*) *The results show that the structure of the sampled subgraph $G_s$ depends on the manner in which the topology of the graph G, the nature of target property $\eta$ (*e.g.*, degree distribution), and the characteristics of the sampling method interact. In future work, we aim to study how to adapt the sample given prior knowledge of the graph properties.*

## 7. PRACTICAL APPLICATIONS OF NETWORK SAMPLING

In Section 2, we discussed how network sampling arises in many different applications (e.g., social science). Most data mining research on network sampling has focused on how to collect a sample that closely match *topological* properties of the network [Leskovec and Faloutsos 2006; Maiya and Berger-Wolf 2011]. However, since the topological properties are never entirely preserved, it is also important to study how the sampling process impacts applications overlaid on the sampled networks. One such study recently investigated the impact of sampling methods on the investigation of information diffusion [De Choudhury et al. 2010]. The study shows that sampling methods that consider both topology and user context improve discovery compared to other naive methods. In this section, we consider the impact of sampling on relational learning. Network sampling is a core part of relational learning since relational data are often represented as attributed graphs. Sampled relational data is used in many

different contexts—including parameter estimation, active learning, collective inference, and algorithm evaluation.

Network sampling can produce samples with imbalance in class membership and/or bias in topological features (e.g., path length, clustering) due to missing nodes/edges—thus the sampling process can significantly impact the accuracy of relational classification methods. Biases may result from the size of the sample, the applied sampling method, or both. Although most previous work in relational learning has focused on analyzing a single *input network,* and research has considered how to further split the input network into training and testing networks for evaluation [Körner and Wrobel 2005; Macskassy and Provost 2007; Neville et al. 2009], the fact that the input network is often itself sampled from an unknown target network has largely been ignored. There has been little focus on how the construction of the input networks may impact the performance of relational algorithms and evaluation methods [Ahmed et al. 2012b].

In this section, we study the question of how the choice of the sampling method can impact the *parameter estimation* and *performance evaluation* of relational classification algorithms. We aim to evaluate the impact of network sampling on relational classification using two different goals:

(1) *Parameter estimation:* we study the impact of network sampling on the estimation of class priors, that is, goal 3.
(2) *Performance evaluation:* we study the impact of network sampling on the estimation of classification accuracy of relational learners, that is, goal 2.

### 7.1. Case Study: Relational Classification

Conventional classification algorithms focus on the problem of identifying the unknown class (e.g., group) to which an entity (e.g., person) belongs. Classification models are learned from a training set of entities, which are assumed to be independent and identically distributed (i.i.d.) and drawn from the underlying population of instances. However, relational classification problems differs from this conventional view in that entities violate the i.i.d. assumption. In relational data, entities (e.g., social network users) can exhibit complex dependencies. For example, friends often share similar interests (e.g., political views).

Recently, there has been a great deal of research in relational learning and classification. For example, Friedman et al. [1999] and Taskar et al. [2001] outline probabilistic relational learning algorithms that search the space of relational attributes and neighbor correlations to improve classification accuracy. Macskassy and Provost [2007] proposed a simple relational neighbor classifier (weighted-vote relational neighbor wvRN) that requires no learning and iteratively classifies the entities in a relational network based on the relational structure. Macskassy showed that wvRN often performs competitively when compared to other more complex relational learning algorithms.

*7.1.1. Impact of Sampling on Parameter Estimation.* Let $a$ be the node attribute representing the class label of any node $v_i \in V$ in graph $G$. We denote $\mathcal{C} = \{c_1, c_2, \ldots\}$ as the set of possible class labels, where $c_l$ is the class label of node $v_i$ (i.e., $a(v_i) = c_l$).

We study the impact of network sampling on the estimation of class priors in $G$ (i.e., the distribution of class labels) using the following procedure:

(1) Choose a set of nodes $S$ from $V$ using a sampling algorithm $\sigma$.
(2) For each node $v_i \in S$, observe $v_i$'s class label.
(3) Estimate the class label distribution $\hat{p}_{c_l}$ from $S$, using the following equation,

$$\hat{p}_{c_l} = \frac{1}{|S|} \sum_{v_i \in S} 1_{(a(v_i)=c_l)}$$

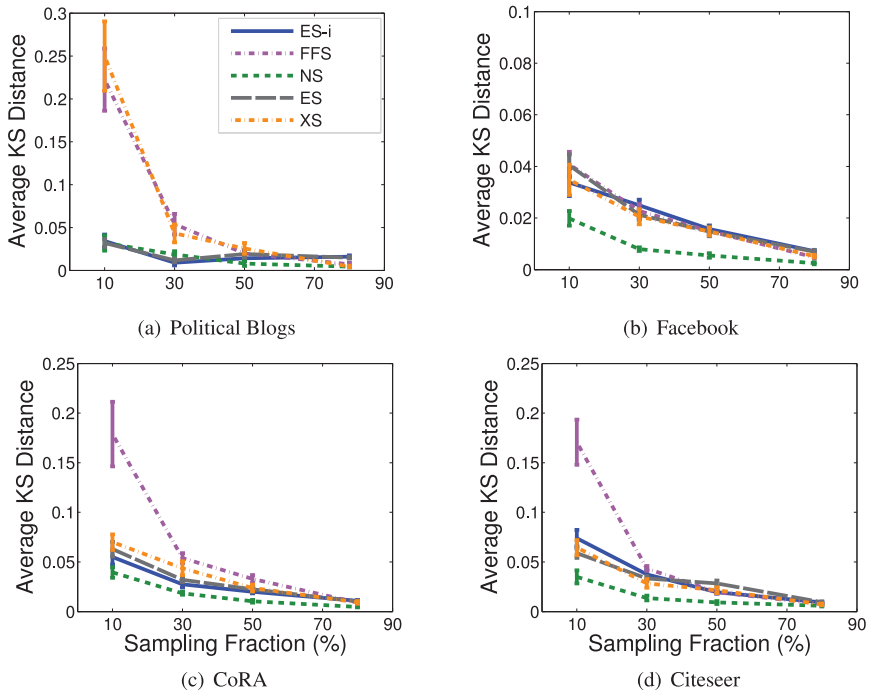



Fig. 10. Average KS distance of class priors for NS, ES, FFS, XS, and ES-i.

In our experiments, we consider four real networks: two citation networks CoRA with 2,708 nodes and Citeseer with 3,312 nodes [Sen et al. 2008], Facebook collected from Facebook Purdue network with 7,315 users with their political views [Xiang et al. 2010], and a single day's snapshot of 1,490 political blogs that shows the interactions between liberal and conservative blogs [Adamic and Glance 2005]. We sample a subset of the nodes using NS, ES, ES-i, FFS, and Expansion Sampling (XS). XS is a snowball sampling method that samples nodes deterministically to maximize the sample expansion [Maiya and Berger-Wolf 2010]. We varied the sample size from 10% to 80% of the size of the graph $G$. For each sample size, we take the average of 10 different runs. Then, we compare the estimated class prior to the actual class prior in the full graph $G$ using the average KS distance measure. As shown in Figures 10(a)–10(d), NS estimates the class priors more accurately than other methods. In contrast, we note that FFS produces a large bias in most of the graphs at small sample sizes (i.e., 10%).

Although XS performs similar to ES-i in CoRA, Citeseer, and Facebook, it performs significantly worse when estimating the class priors for the political blogs network (at 10% sample size). In Adamic and Glance [2005], the authors show there is a clear division between the liberal and conservative political blogs, and that 91% of the links originating within either the conservative or liberal communities stay within that community. According to these observations, the political blogs network is structurally divided into two dense communities, and the nodes are homogeneously labeled within each community. We find in networks with such a dense community structure that XS (and FFS) are likely to be largely biased toward exploring only a small fraction of the communities. For instance, in the political blogs network, we find that at the

10% sample size, XS (and FFS) only sample from one of the two communities (either conservatives or liberals), effectively ignoring the other. This indicates that topology-based sampling methods (such as XS, FFS) may produce biased estimates of class priors from networks that have dense, homogeneously labeled communities.

To solve this problem, we can modify the topology-based methods (like XS and FFS) to randomly jump with a small probability $\alpha$ to a new node in the graph (which can be chosen uniformly at random). We note that $\alpha$ should be set relative to the density within the communities and the sparsity of edges between communities.

*7.1.2. Impact of Sampling on Classification Accuracy.* Let $\mathcal{R}$ be a relational classifier that takes a graph $G$ as input. The goal is to predict the class labels of nodes in $G$. Therefore, $\mathcal{R}$ uses a proportion of nodes in graph $G$ with known class labels as a *training set* to learn a model. Afterward, $\mathcal{R}$ is used to predict the label of the remaining (unlabeled) nodes in $G$ (i.e., *test set*). Generally, the performance of $\mathcal{R}$ can be evaluated based on the accuracy of the predicted class labels. In this work, we calculate the accuracy using area under the ROC curve (AUC) measure.

We study the impact of network sampling on the accuracy of relational classification using the following procedure:

(1) Sample a subgraph $G_s$ from $G$ using a sampling algorithm $\sigma$.
(2) Estimate the classification accuracy of a classifier $\mathcal{R}$ on $G_s$: $\hat{auc} = \mathcal{R}(G_s)$.

We compare the actual classification accuracy on $G$ to the estimated classification accuracy on $G_s$. Formally, we compare $auc = \mathcal{R}(G)$ to $\hat{auc} = \mathcal{R}(G_s)$, and $G_s$ is said to be representative to $G$, if $\hat{auc} \approx auc$.

In our experiments, we use the wvRN as our base classifier $\mathcal{R}$ [Macskassy and Provost 2007]. In wvRN, the class membership probability of a node $v_i$ belonging to class $c_l$ is defined as:

$$P(c_l|v_i) = \frac{1}{Z} \sum_{v_j \in \mathcal{N}(v_i)} w(v_i, v_j) * P(c_l|v_j),$$

where $\mathcal{N}(v_i)$ is the set of neighbors of node $v_i$, $w(v_i, v_j)$ is the weight of the edge $e_{ij} = (v_i, v_j)$, and $Z = \sum_{v_j \in \mathcal{N}(v_i)} w(v_i, v_j)$ is the normalization term.

We follow the common methodology used in Macskassy and Provost [2007] to compute the classification accuracy. First, we vary the proportion of randomly selected labeled nodes from 10% to 80% and use 5-fold cross validation to compute the average AUC. Then, we repeat this procedure for both the graph $G$ and the sample subgraph $G_s$ (at different sample sizes). Note that AUC is calculated for the most prevalent class. Figures 11(a)–11(d) show the plots of AUC versus the sample size ($\phi = [10\%, 80\%]$) with 10% labeled nodes. Figures 12(a)–12(d) show the plots of AUC versus the proportion of labeled nodes, where the AUC is averaged over all sample sizes (10%–80%). We observe that AUC of $G$ is generally underestimated for sample sizes <30% in the case of NS, ES, and FFS. However, generally, ES-i and XS perform better than other sampling methods and converge to the "True" AUC in $G$. In Figures 11(b) and 12(b), ES-i and XS slightly overestimate the true AUC. These results show that in some cases when the graph is noisy (with low autocorrelation between labels of neighboring nodes), sampling of nodes and edges can enhance the performance of relational classification by reducing noise and overfitting.

**Summary.** We observe that many sampling methods fail to simultaneously satisfy the two different goals even though both are needed for relational learning applications (i.e., parameter estimation and accuracy evaluation). For example, while NS estimates

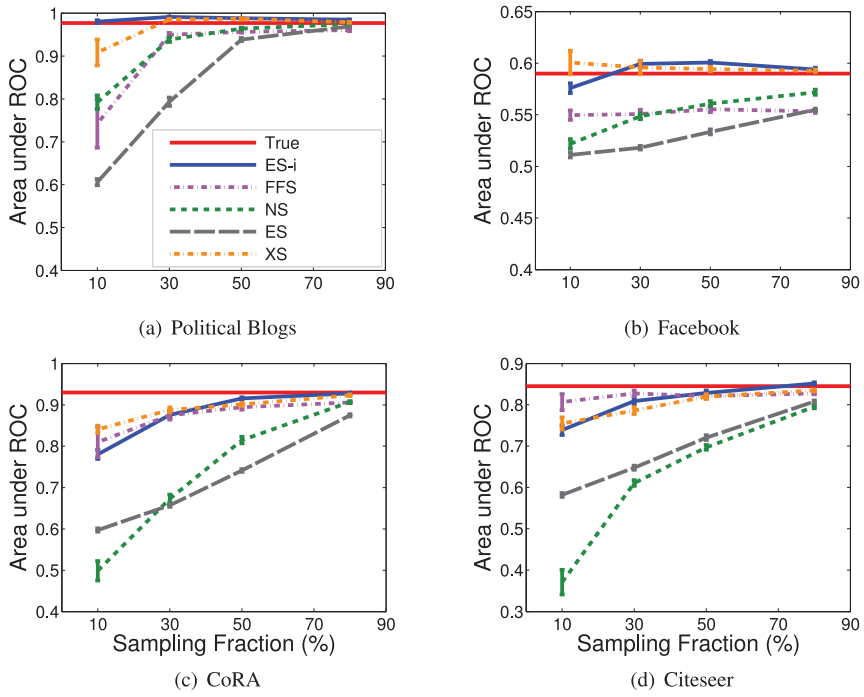



Fig. 11. Classification accuracy versus sampling fraction, where 10% of nodes in the graph are initially labeled for prediction.

the class priors better than other methods, it cannot accurately estimate classification accuracy due to lack of connectivity in the samples. Edge sampling performs similarly to node sampling. On the other hand, while FFS and XS methods may be more accurate for estimating classification accuracy, they are generally not robust for estimating unbiased class priors (particularly for networks with homogeneously labeled dense communities). ES-i provides a good balance for satisfying the two goals with a small bias at the smaller sample sizes.

## 8. CONCLUSION AND FUTURE WORK

In this article, we outlined a framework for the general problem of network sampling, by highlighting the different goals of study, population and units of interest, and classes of network sampling methods. This framework should facilitate the comparison of the strengths and weaknesses of different sampling algorithms—relative to the particular goal of study. In addition, we proposed and discussed a spectrum of computational models for network sampling methods, ranging from the simplest and least restrictive model of sampling static graphs to the more realistic, but more restrictive, model of sampling streaming graphs. Within this context, we designed a family of sampling methods based on the concept of graph induction, which generalizes across the full spectrum of computational models (from static to streaming) while efficiently preserving many of the topological properties of static and streaming graphs. Our experimental results indicate that our family of sampling methods more accurately preserves the underlying properties of the graph for both static and streaming graphs. Finally, we studied the impact of network sampling algorithms on parameter estimation and performance evaluation of relational classification algorithms. The results indicate our sampling

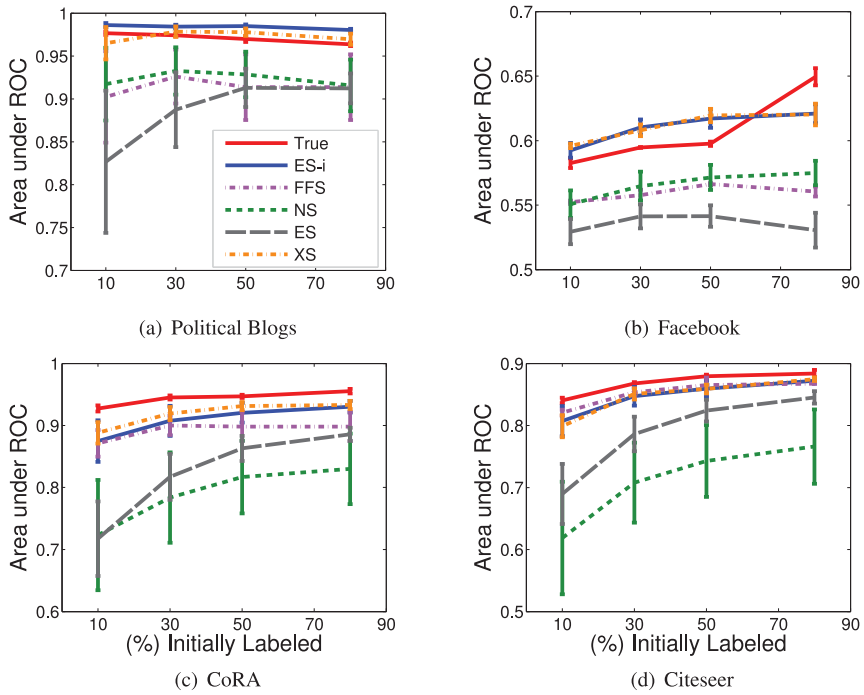



Fig. 12. Classification accuracy versus proportion of nodes in the graph that are initially labeled for prediction, averaged over sample sizes 10% to 80%.

method produces a good balance across the two goals, producing accurate estimates of class priors and classification accuracy. In future work, we plan to investigate the performance of sublinear time stream sampling methods for sampling a representative subgraph in addition to extending our analysis to other task-based evaluations.

## A. APPENDIX A

### A.1. Tables of Results of PIES (MIN)

Table VIII. Average L1/L2 Distance for PIES, PIES (MIN), NS, ES, and BFS Stream Sampling

| Data | | PIES | PIES (MIN) | NS | ES | BFS |
|---|---|---|---|---|---|---|
| EMAIL-UNIV | *EigenVal* | 0.4487 | **0.074*** | 0.7018 | 0.7588 | 0.838 |
| | *NetVal* | 0.199 | **0.0201*** | 0.5785 | 0.3799 | 1.007 |
| TWITTER | *EigenVal* | 0.4981 | **0.1851*** | 0.6411 | 0.7217 | 0.7964 |
| | *NetVal* | 0.1431 | **0.043*** | 0.3108 | 0.4271 | 0.8385 |
| FACEBOOK | *EigenVal* | 0.591 | **0.1143*** | 0.6771 | 0.8417 | 0.9018 |
| | *NetVal* | 0.306 | **0.0617*** | 0.5383 | 1.0984 | 1.5027 |
| FLICKR | *EigenVal* | 0.5503 | **0.0049*** | 0.7227 | 0.8491 | 0.9298 |
| | *NetVal* | 0.1657 | **0.0005*** | 0.5626 | 0.0574 | 1.5193 |
| HEPPH | *EigenVal* | 0.7083 | **0.2825*** | 0.7232 | 0.9373 | 0.95 |
| | *NetVal* | 0.3 | **0.1817*** | 0.3198 | 1.0821 | 1.2477 |
| CONDMAT | *EigenVal* | 0.6278 | **0.1475*** | 0.6843 | 0.8507 | 0.8875 |
| | *NetVal* | 0.2254 | **0.06*** | 0.3235 | 0.7514 | 0.9853 |

Table IX. Average KS Distance for PIES, PIES (MIN), NS, ES, and BFS Stream Sampling Methods

| Data | | PIES | PIES (MIN) | NS | ES | BFS |
|---|---|---|---|---|---|---|
| EMAIL-UNIV | *Deg* | 0.2348 | 0.5803 | 0.4547 | 0.2186 | 0.3724 |
| | *PL* | 0.204 | 0.5621 | 0.19 | 0.6989 | 0.5114 |
| | *Clust* | 0.1108 | 0.4728 | 0.188 | 0.3302 | 0.3473 |
| | *KCore* | 0.2819 | 0.5821 | 0.1985 | 0.3219 | 0.5759 |
| | | **0.2079*** | 0.5493 | 0.2578 | 0.3924 | 0.4518 |
| TWITTER | *Deg* | 0.1521 | 0.2598 | 0.4667 | 0.3052 | 0.4194 |
| | *PL* | 0.0528 | 0.3941 | 0.1243 | 0.617 | 0.4811 |
| | *Clust* | 0.2462 | 0.2269 | 0.346 | 0.4673 | 0.482 |
| | *KCore* | 0.1001 | 0.2886 | 0.2271 | 0.4393 | 0.5929 |
| | | **0.1378*** | 0.2923 | 0.291 | 0.4572 | 0.4938 |
| FACEBOOK | *Deg* | 0.1848 | 0.2357 | 0.3804 | 0.4912 | 0.6917 |
| | *PL* | 0.2121 | 0.3171 | 0.4337 | 0.8762 | 0.9557 |
| | *Clust* | 0.2594 | 0.2314 | 0.3496 | 0.4975 | 0.5017 |
| | *KCore* | 0.2375 | 0.2447 | 0.3569 | 0.661 | 0.7275 |
| | | **0.2234*** | 0.2572 | 0.3802 | 0.6315 | 0.7192 |
| FLICKR | *Deg* | 0.1503 | 0.399 | 0.514 | 0.0924 | 0.2706 |
| | *PL* | 0.2845 | 0.4936 | 0.0789 | 0.1487 | 0.6763 |
| | *Clust* | 0.1426 | 0.3754 | 0.2404 | 0.3156 | 0.3931 |
| | *KCore* | 0.1654 | 0.4289 | 0.0595 | 0.1295 | 0.4541 |
| | | 0.1857 | 0.4242 | 0.2232 | **0.1716*** | 0.4485 |
| HEPPH | *Deg* | 0.4103 | 0.1304 | 0.483 | 0.8585 | 0.8923 |
| | *PL* | 0.306 | 0.1959 | 0.431 | 0.749 | 0.8676 |
| | *Clust* | 0.4636 | 0.0393 | 0.3441 | 0.9156 | 0.9171 |
| | *KCore* | 0.592 | 0.1674 | 0.6233 | 0.9402 | 0.9592 |
| | | 0.443 | **0.1332*** | 0.4704 | 0.8658 | 0.909 |
| CONDMAT | *Deg* | 0.4042 | 0.1259 | 0.5006 | 0.6787 | 0.7471 |
| | *PL* | 0.2944 | 0.2758 | 0.5211 | 0.6981 | 0.9205 |
| | *Clust* | 0.5927 | 0.3285 | 0.5341 | 0.878 | 0.8853 |
| | *KCore* | 0.4692 | 0.1512 | 0.4955 | 0.858 | 0.8909 |
| | | 0.4401 | **0.2203*** | 0.5128 | 0.7782 | 0.8609 |
| Average for all Datasets | | **0.2730*** | 0.3128 | 0.3559 | 0.5494 | 0.6472 |

## A.2. Distributions for Static Graphs (at 20% sample size)

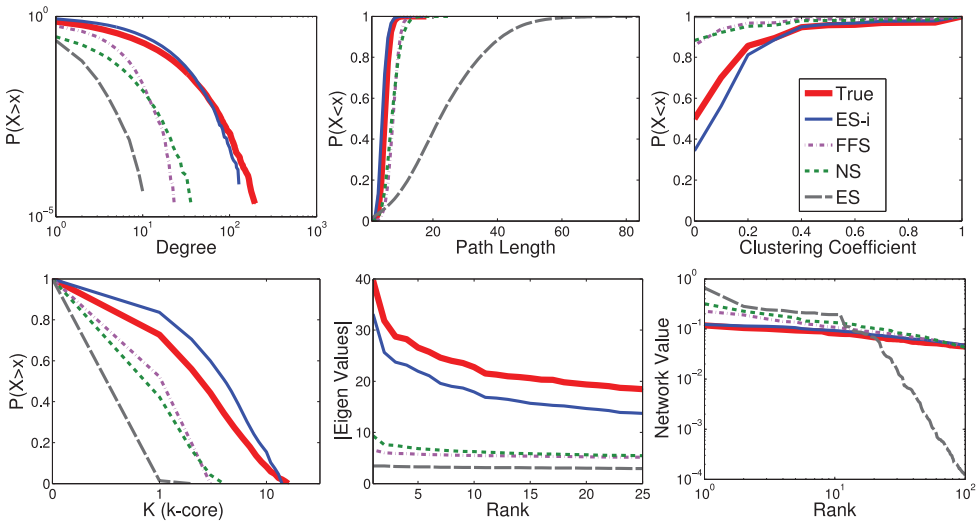



Fig. 13. FACEBOOK graph.

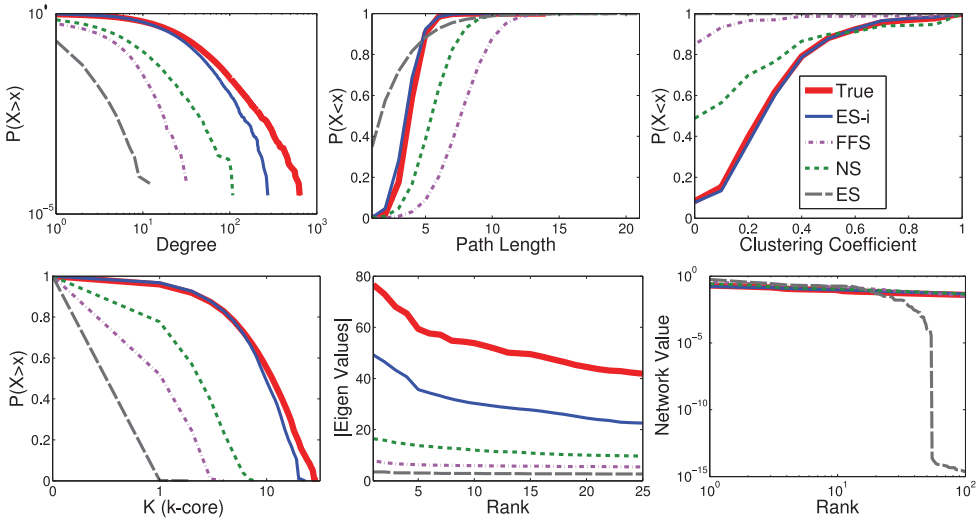



Fig. 14. HEPPH graph.

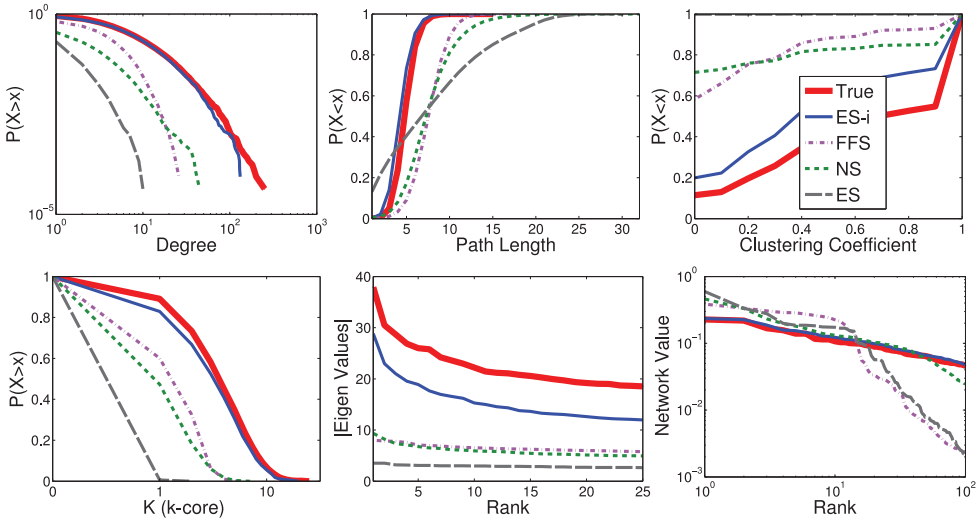



Fig. 15. CONDMAT graph.

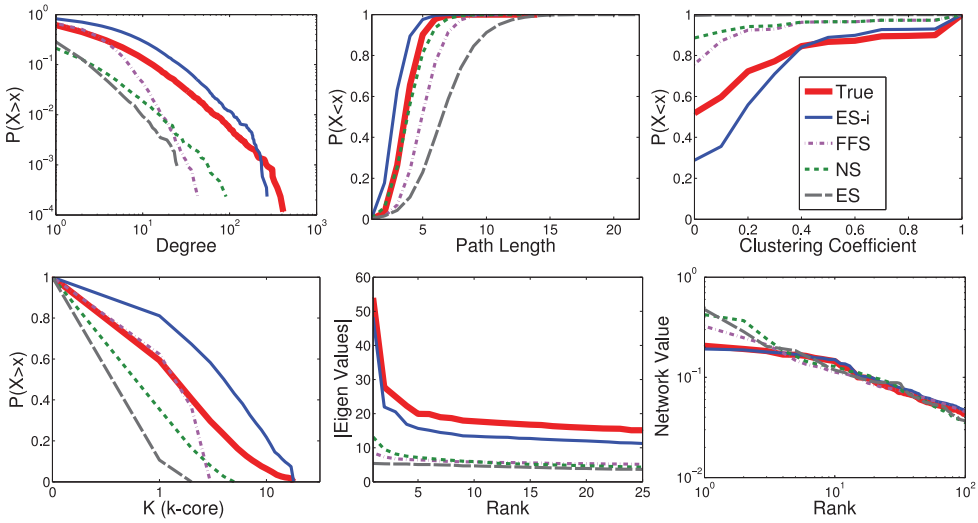



Fig. 16. TWITTER graph.

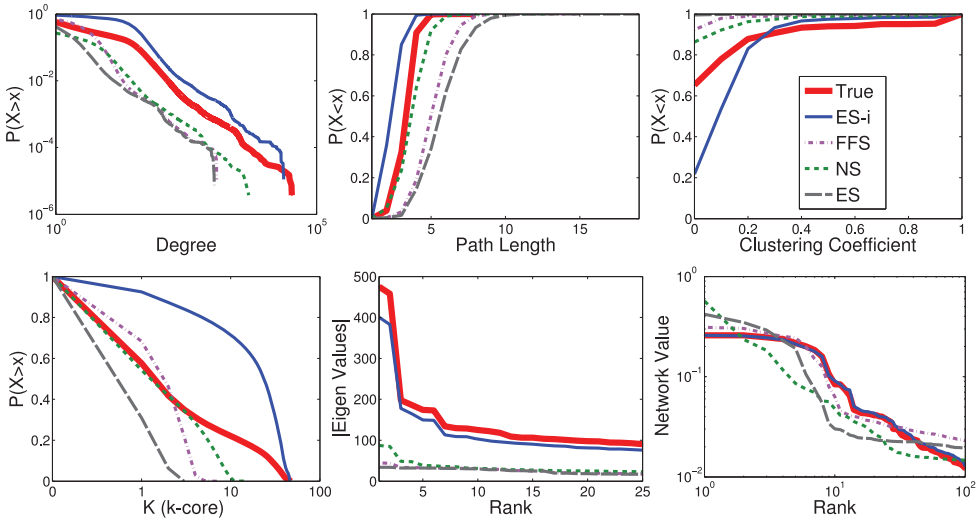



Fig. 17. EMAIL-UNIV graph.

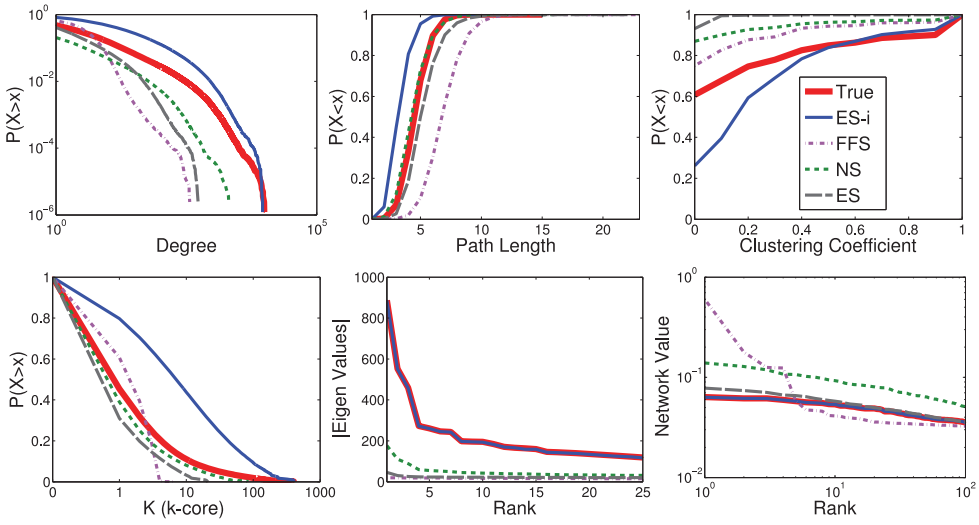



Fig. 18. FLICKR graph.

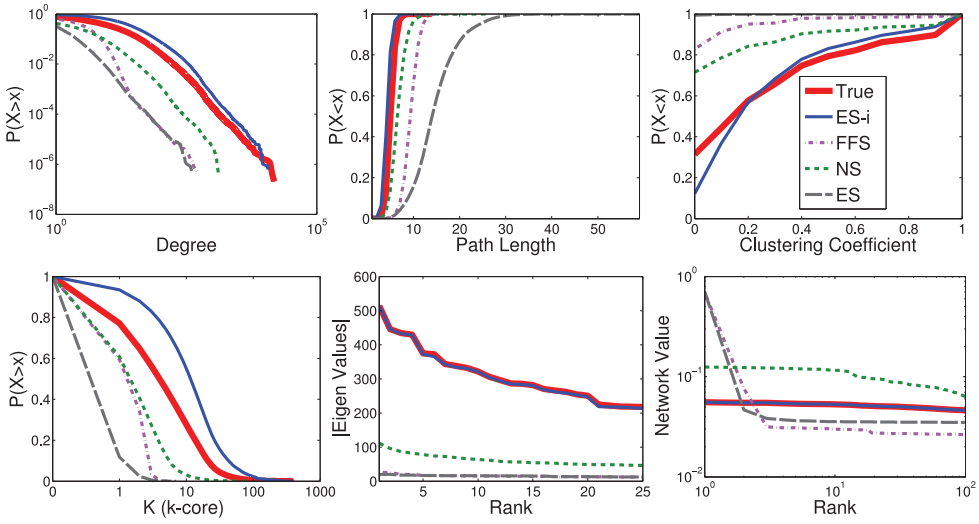



Fig. 19. LIVEJOURNAL graph.

### A.3. Distributions for Streaming Graphs (at 20% sample size)

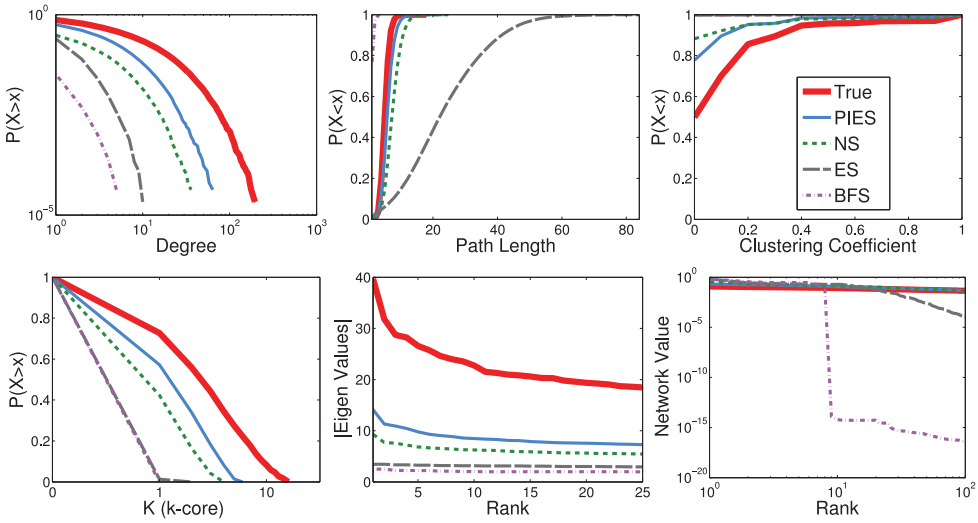



Fig. 20. FACEBOOK graph.

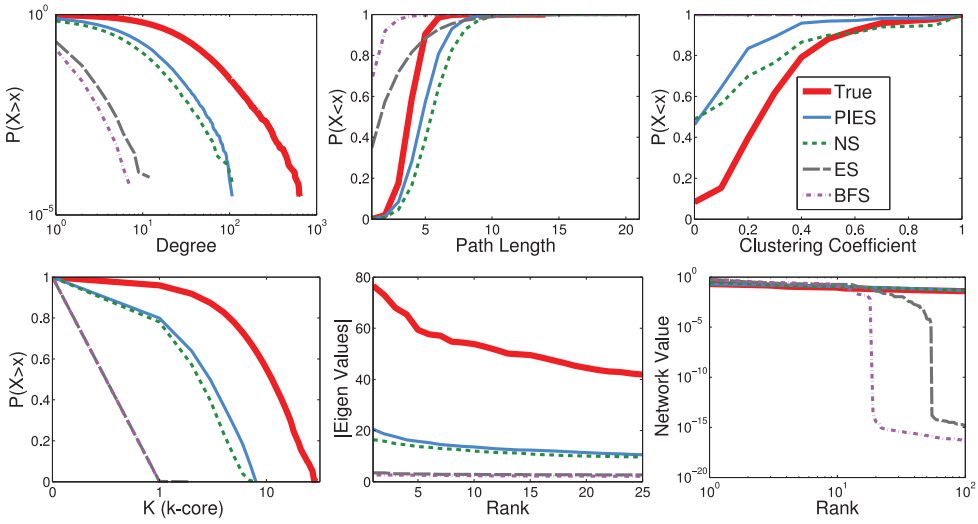



Fig. 21. HEPPH graph.

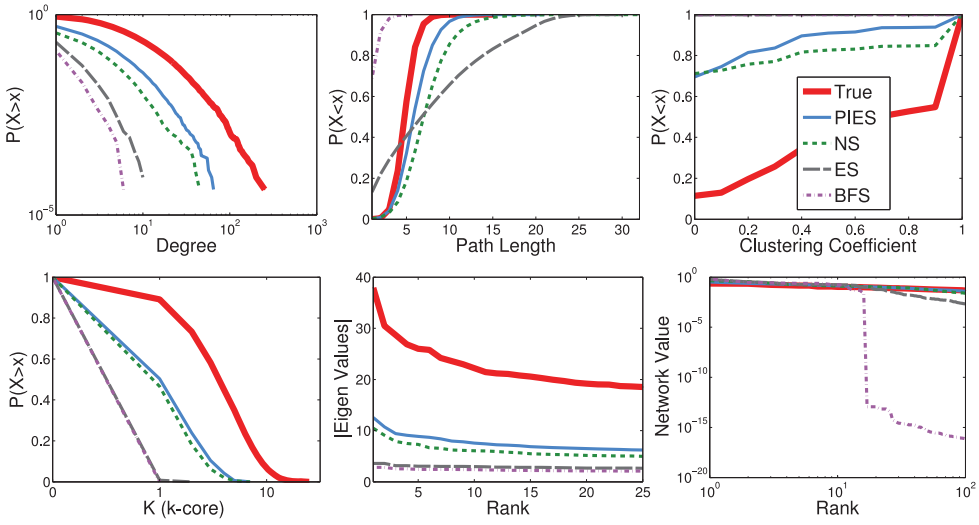



Fig. 22. CondMAT graph.

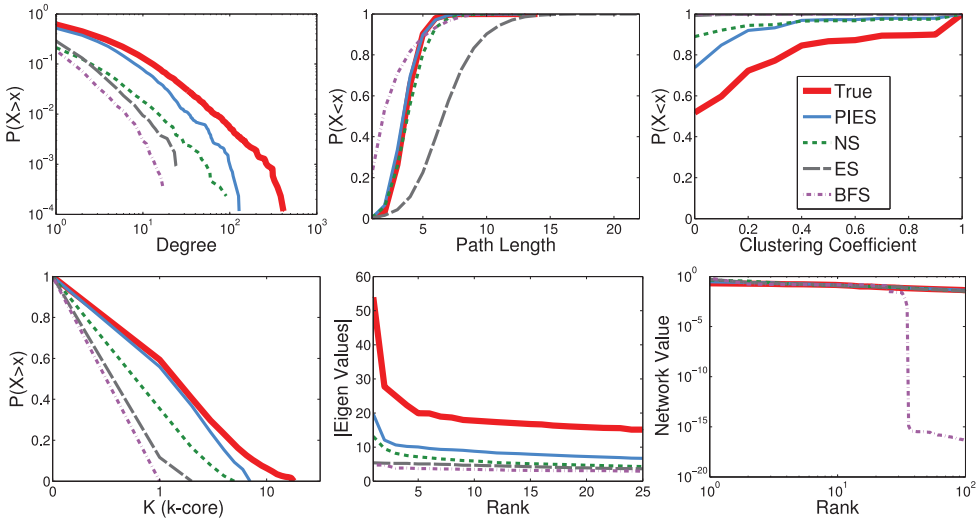



Fig. 23. Twitter graph.

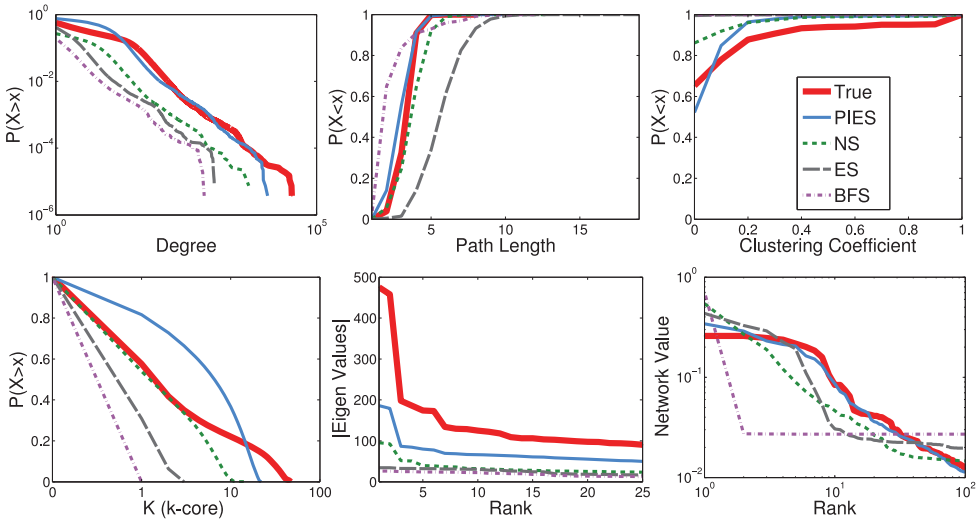



Fig. 24. EMAIL-UNIV graph.

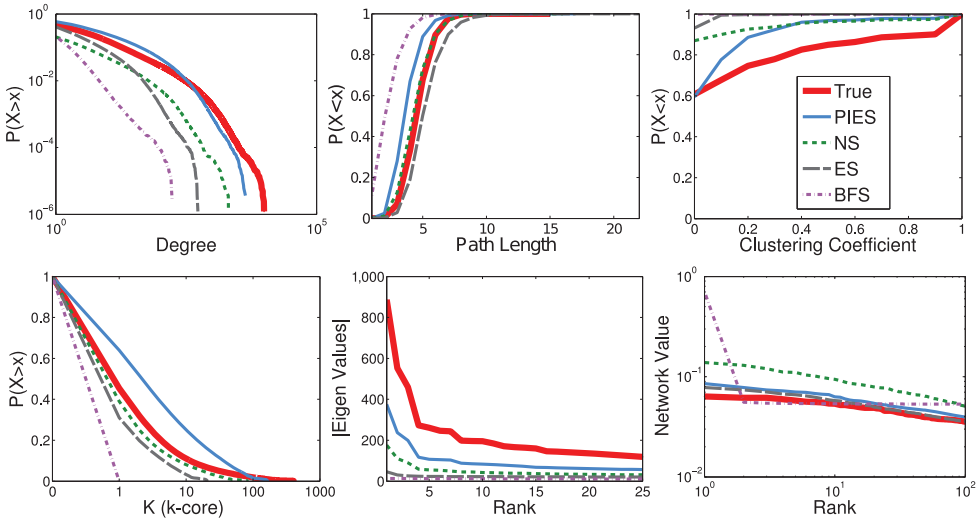



Fig. 25. FLICKR graph.

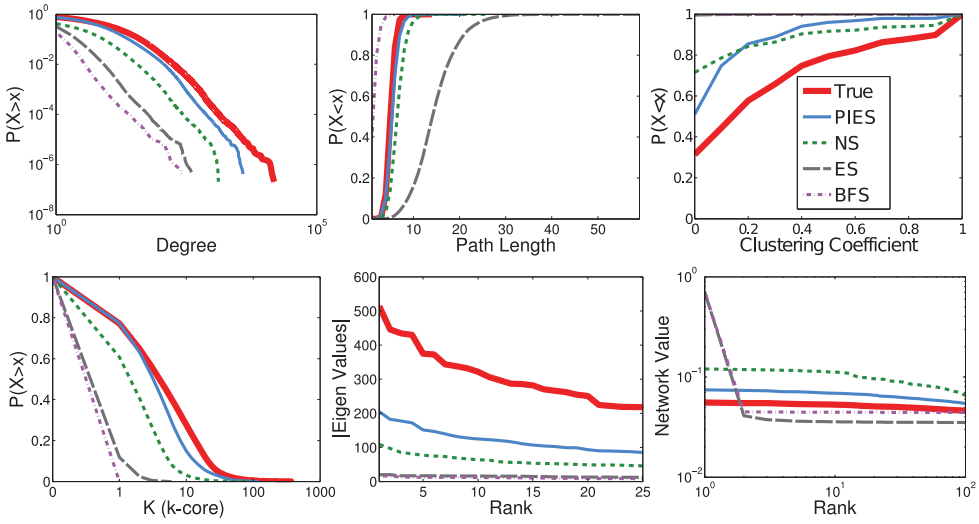



Fig. 26. LIVEJOURNAL graph.

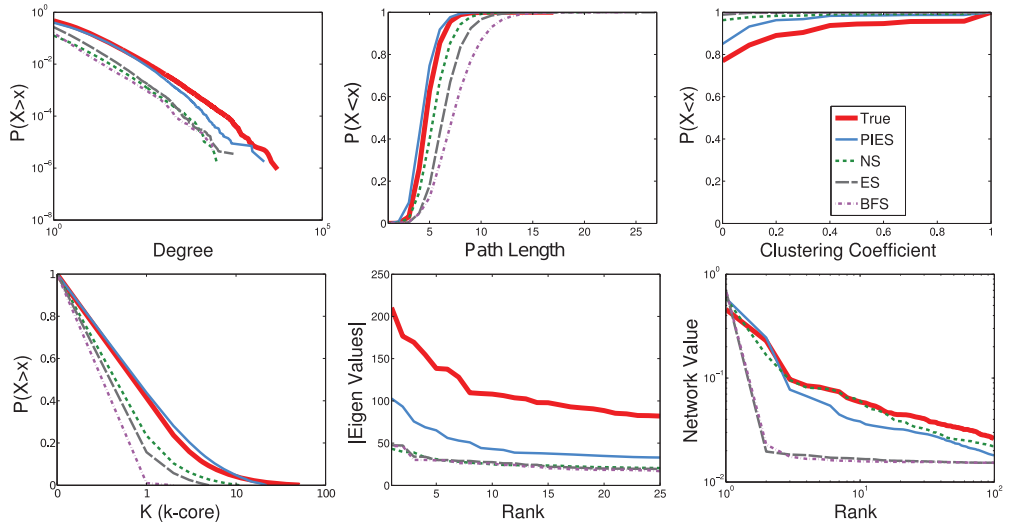



Fig. 27. YOUTUBE graph.

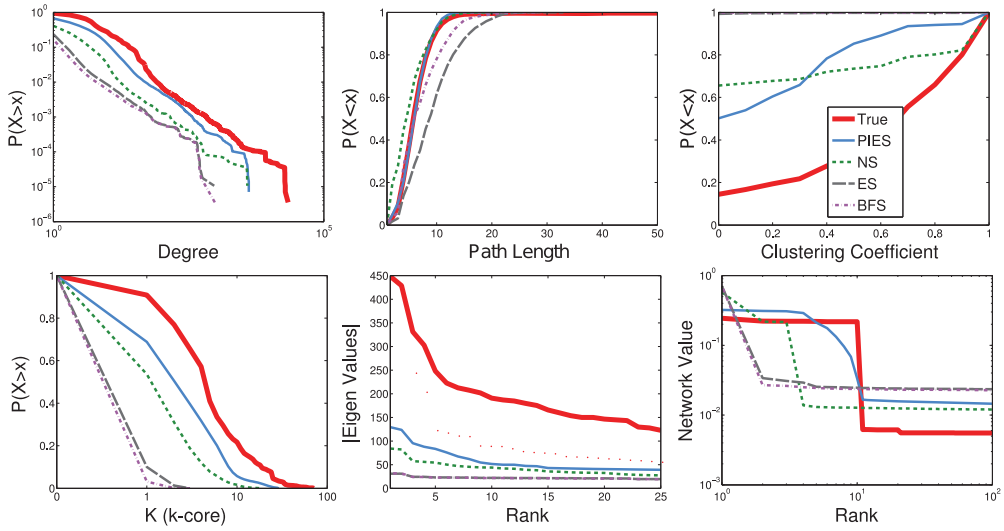



Fig. 28. WEB-STANFORD graph.

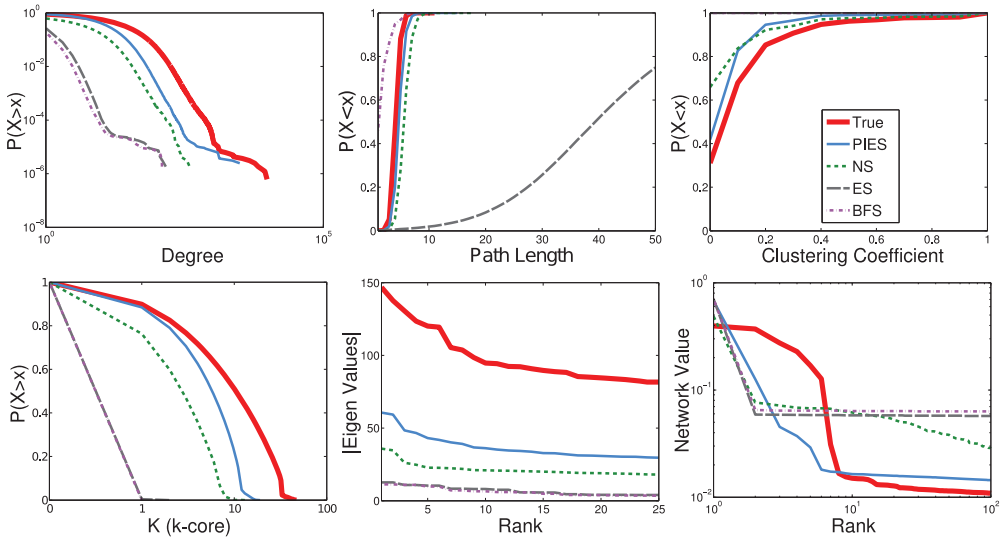
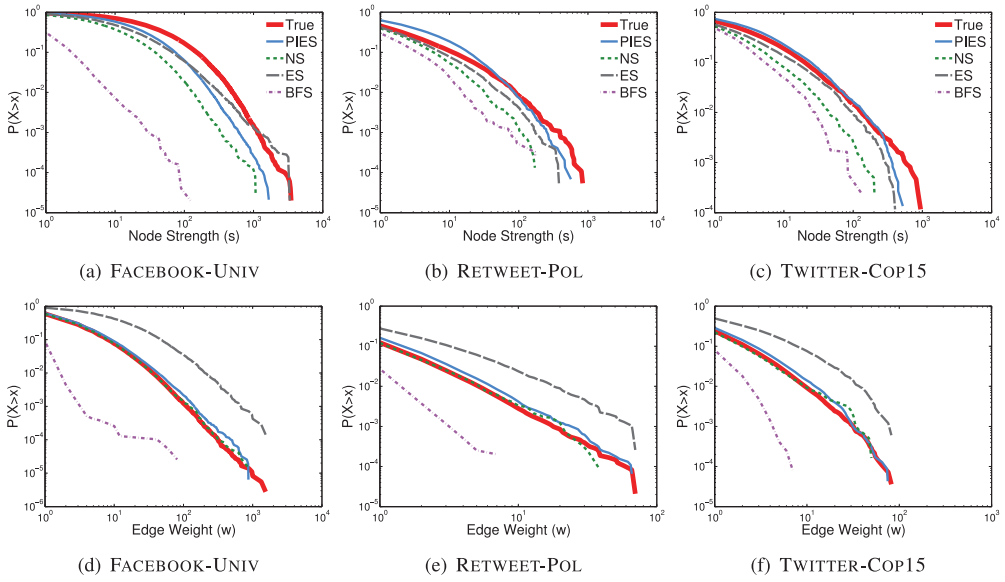


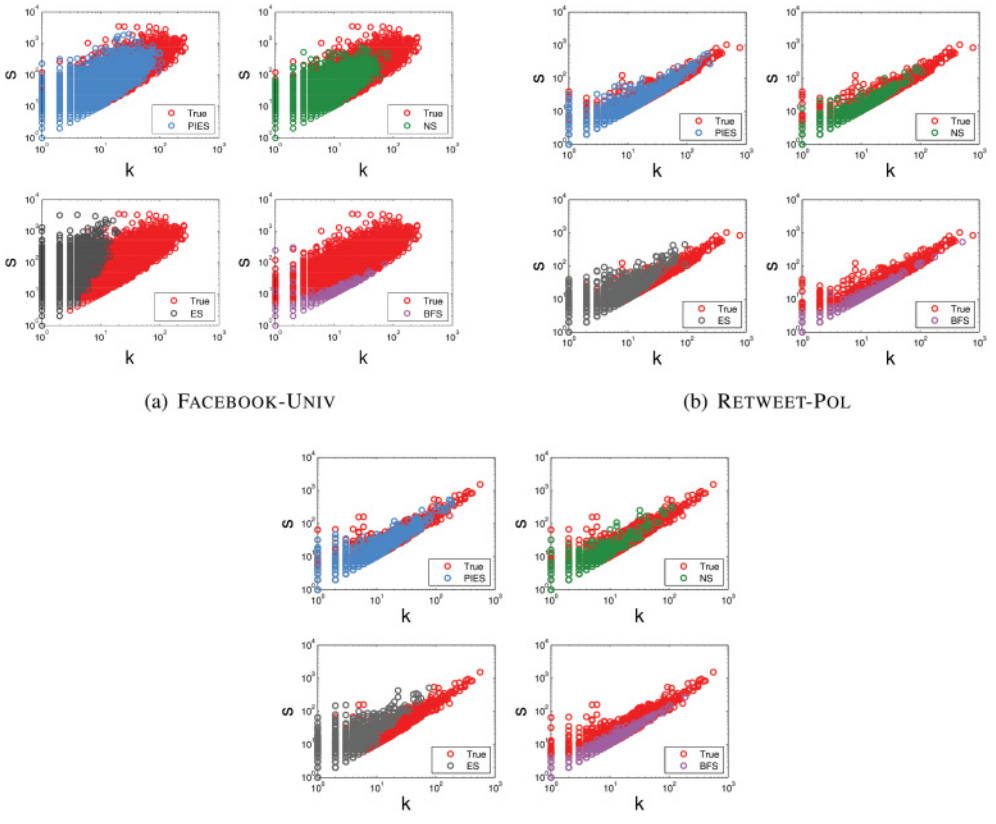

Fig. 29. POKEC graph.




(a) FACEBOOK-UNIV (b) RETWEET-POL (c) TWITTER-COP15

(d) FACEBOOK-UNIV (e) RETWEET-POL (f) TWITTER-COP15

Fig. 30. (a–c) Distributions of node strength (s), (d–f) Distributions of edge weight (w), at 20% sample size.




(a) FACEBOOK-UNIV (b) RETWEET-POL

(c) TWITTER-COP15

Fig. 31. (a–c) Node strength (s) vs. degree (k) at 20% sample size.

## REFERENCES


L. A. Adamic and N. Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*. 36–43.

C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. 2003. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases*. 81–92.

C. C. Aggarwal, Y. Li, P. S. Yu, and R. Jin. 2010a. On dense pattern mining in graph streams. In *Proceedings of the VLDB Endowment* 3, 1–2 (2010), 975–984.

C. Aggarwal, Y. Zhao, and P. Yu. 2010b. On clustering graph streams. In *SIAM International Conference on Data Mining*. 478–489.

C. C. Aggarwal, Y. Zhao, and P. S. Yu. 2011. Outlier detection in graph streams. In *IEEE 27th International Conference on Data Engineering*. 399–409.

Charu C. Aggarwal. 2006. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd International Conference on Very Large Data Bases*. 607–618.

Charu C. Aggarwal (Ed.). 2007. *Data Streams - Models and Algorithms*. Advances in Database Systems, Vol. 31. Springer.

N. K. Ahmed, F. Berchmans, J. Neville, and R. Kompella. 2010a. Time-based sampling of social network activity graphs. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs*. 1–9.

N. K. Ahmed, J. Neville, and R. Kompella. 2012a. Space-efficient sampling from social activity streams. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. 53–60.

Nesreen Ahmed, Jennifer Neville, and Ramana Rao Kompella. 2011. *Network sampling via edge-based node selection with graph induction*. Technical Report 11-016. Purdue Digital Library.

N. K. Ahmed, J. Neville, and R. Kompella. 2010b. Reconsidering the foundations of network sampling. In *Proceedings of the 2nd Workshop on Information in Networks*.

N. K. Ahmed, J. Neville, and R. Kompella. 2012b. Network sampling designs for relational classification. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*. 1–4.

Y. Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. 2007. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th International Conference on World Wide Web*. 835–844.

M. Al Hasan and M. J. Zaki. 2009. Output space sampling for graph patterns. *Proceedings of the VLDB Endowment* 2, 1 (2009), 730–741.

J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. 2005. k-core decomposition of Internet graphs: Hierarchies, self-similarity and measurement biases. *arXiv preprint cs/0511007* (2005).

K. Avrachenkov, B. Ribeiro, and D. Towsley. 2010. Improving random walk estimation accuracy with uniform restarts. In *Algorithms and Models for the Web-Graph*. Lecture Notes in Computer Science, Vol. 6516. Springer, 98–109.

B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. 2002a. Models and issues in data stream systems. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 1–16.

B. Babcock, M. Datar, and R. Motwani. 2002b. Sampling from a moving window over streaming data. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*. 633–634.

L. Backstrom and J. Kleinberg. 2011. Network bucket testing. In *Proceedings of the 20th International Conference on World Wide Web*. 615–624.

E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. 2012. The role of social networks in information diffusion. In *Proceedings of the 21st International Conference on World Wide Web*. 519–528.

Z. Bar-Yossef, R. Kumar, and D. Sivakumar. 2002. Reductions in streaming algorithms with an application to counting triangles in graphs. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*. 623–632.

E. Baykan, M. Henzinger, S. F. Keller, S. De Castelberg, and M. Kinzler. 2009. A comparison of techniques for sampling web pages. *Arxiv preprint arXiv:0902.1604* (2009).

Mansurul A. Bhuiyan, Mahmudur Rahman, Mahmuda Rahman, and Mohammad Al Hasan. 2012. GUISE: Uniform sampling of graphlets for large graph analysis. In *IEEE 12th International Conference on Data Mining*. 91–100.

A. L. Buchsbaum, R. Giancarlo, and J. R. Westbrook. 2003. On finding common neighborhoods in massive graphs. *Theoretical Computer Science* 299, 1 (2003), 707–718.

L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. 2006. Counting triangles in data streams. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 253–262.

S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir. 2007. A model of internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences* 104, 27 (2007), 11150–11154.

D. Chakrabarti, Y. Zhan, and C. Faloutsos. 2004. R-MAT: A recursive model for graph mining. In *SIAM International Conference on Data Mining*. 442–446.

M. Charikar, K. Chen, and M. Farach-Colton. 2002. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*. 693–703.

L. Chen and C. Wang. 2010. Continuous subgraph pattern search over certain and uncertain graph streams. *IEEE Transactions on Knowledge and Data Engineering* 22, 8 (2010), 1093–1109.

M. Conover, J. Ratkiewicz, M. Francisco, B Gonçalves, A. Flammini, and F. Menczer. 2011. Political polarization on twitter. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*. 89–96.

G. Cormode and S. Muthukrishnan. 2005. Space efficient mining of multigraph streams. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 271–282.

A. Dasgupta, R. Kumar, and D. Sivakumar. 2012. Social sampling. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 235–243.

M. De Choudhury, Y. R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher. 2010. How does the data sampling strategy impact the discovery of information diffusion in social media. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*. 34–41.

P. Domingos and G. Hulten. 2000. Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 71–80.

W. Fan. 2004a. StreamMiner: A classifier ensemble-based engine to mine concept-drifting data streams. In *Proceedings of the 30th International Conference on Very Large Data Bases - Volume 30*. 1257–1260.

W. Fan. 2004b. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 128–137.

W. Fan, Y. Huang, H. Wang, and P. S. Yu. 2004. Active mining of data streams. In *SIAM International Conference on Data Mining*. 457–461.

O. Frank. 1977. Survey sampling in graphs. *Journal of Statistical Planning and Inference* 1, 3 (1977), 235–264.

O. Frank. 1980. Sampling and inference in a population graph. *International Statistical Review / Revue Internationale de Statistique* 48, 1 (1980), 33–41.

O. Frank. 1981. A survey of statistical methods for graph analysis. *Sociological Methodology* 12 (1981), 110–155.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. 1999. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*. 1300–1309.

M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. 2005. Mining data streams: A review. *ACM Sigmod Record* 34, 2 (2005), 18–26.

J. Gao, W. Fan, J. Han, and P. S. Yu. 2007. A general framework for mining concept-drifting data streams with skewed distributions. In *SIAM International Conference on Data Mining*. 3–14.

Aurelien Gautreau, Alain Barrat, and Marc Barthélemy. 2009. Microdynamics in stationary complex networks. *Proceedings of the National Academy of Sciences* 106, 22 (2009), 8847–8852.

K. J. Gile and M. S. Handcock. 2010. Respondent-driven sampling: An assessment of current methodology. *Sociological Methodology* 40, 1 (2010), 285–327.

M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. 2010. Walking in Facebook: A case study of unbiased sampling of OSNs. In *IEEE International Conference on Computer Communications*. 1–9.

M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. 2011. Practical recommendations on crawling online social networks. *IEEE Journal on Selected Areas in Communications* 29, 9 (2011), 1872–1892.

C. Gkantsidis, M. Mihail, and A. Saberi. 2004. Random walks in peer-to-peer networks. In *IEEE International Conference on Computer Communications*. 1–12.

David F. Gleich. 2012. Graph of Flickr Photo-Sharing Social Network Crawled in May 2006. (Feb 2012). https://research.hub.purdue.edu/publications/1002.

L. Golab and M. T. Özsu. 2003. Issues in data stream management. *ACM Sigmod Record* 32, 2 (2003), 5–14.

L. A. Goodman. 1961. Snowball sampling. *The Annals of Mathematical Statistics* 32, 1 (1961), 148–170.

M. Granovetter. 1976. Network sampling: Some first steps. *American Journal of Sociology* 81, 6 (1976), 1287–1303.

S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. 2003. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering* 15, 3 (2003), 515–528.

W. H. Haemers. 1995. Interlacing eigenvalues and graphs. *Linear Algebra Applications* 226 (1995), 593–616.

M. H. Hansen and W. N. Hurwitz. 1943. On the theory of sampling from finite populations. *The Annals of Mathematical Statistics* 14, 4 (1943), 333–362.

D. D. Heckathorn. 1997. Respondent-driven sampling: A new approach to the study of hidden populations. *Social Problems* 2 (1997), 174–199.

M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. 2000. On near-uniform URL sampling. *Computer Networks* 33, 1 (2000), 295–308.

M. Henzinger, P. Raghavan, and S. Rajagopalan. 1999. Computing on data streams. In *External Memory Algorithms: Dimacs Workshop External Memory and Visualization*, Vol. 50. 107–118.

Christian Hubler, Hans-Peter Kriegel, Karsten M. Borgwardt, and Zoubin Ghahramani. 2008. Metropolis algorithms for representative subgraph sampling. In *IEEE 8th International Conference on Data Mining*. 283–292.

G. Hulten, L. Spencer, and P. Domingos. 2001. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 97–106.

Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton, and Wouter Van den Broeck. 2011. What's in a crowd? Analysis of face-to-face behavioral networks. *Journal of Theoretical Biology* 271, 1 (2011), 166–180.

Y. Jia, J. Hoberock, M. Garland, and J. Hart. 2008. On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1285–1292.

J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. 1999. The web as a graph: Measurements, models, and methods. In *Computing and Combinatorics*. Lecture Notes in Computer Science, Vol. 1627. Springer, 1–17.

E. D. Kolaczyk. 2009. *Statistical Analysis of Network Data*. Springer, Chapter 5: Sampling and Estimation in Network Graphs, 123–152.

Christine Körner and Stefan Wrobel. 2005. Bias-Free hypothesis evaluation in multirelational domains. In *Proceedings of the 4th International Workshop on Multi-relational Mining*. 33–38.

V. Krishnamurthy, M. Faloutsos, M. Chrobak, J. H. Cui, L. Lao, and A. G. Percus. 2007. Sampling large internet topologies for simulation purposes. *Computer Networks* 51, 15 (2007), 4284–4302.

R. Kumar, J. Novak, and A. Tomkins. 2006. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 337–357.

M. Kurant, A. Markopoulou, and P. Thiran. 2011. Towards unbiased BFS sampling. *IEEE Journal on Selected Areas in Communications* 29, 9 (2011), 1799–1809.

Justin Lafferty. 2012. Facebook Users Worldwide: 3.2 Billion Likes And Comments Per Day. http://allfacebook.com/facebook-marketing-infographic-engagement_b98277. (August 2012).

A. Lakhina, J. W. Byers, M. Crovella, and P. Xie. 2003. Sampling biases in IP topology measurements. In *IEEE International Conference on Computer Communications*. 332–341.

L. Lee. 2001. On the effectiveness of the skew divergence for statistical language analysis. In *Artificial Intelligence and Statistics*. 65–72.

S. Lee, P. Kim, and H. Jeong. 2006. Statistical properties of sampled networks. *Physical Review E* 73 (2006), 016102.

SNAP. 2013. Stanford Network Analysis Project. http://snap.stanford.edu/data/.

J. Leskovec and C. Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 631–636.

X. Li, P. S. Yu, B. Liu, and S. K. Ng. 2009. Positive unlabeled learning for data stream classification. In *SIAM International Conference on Data Mining*. 259–270.

Xuesong Lu and Stéphane Bressan. 2012. Sampling connected induced subgraphs uniformly at random. In *Scientific and Statistical Database Management*. Lecture Notes in Computer Science, Vol. 7338. Springer Berlin Heidelberg, 195–212.

S. Macskassy and F. Provost. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research* 8 (2007), 935–983.

A. S. Maiya and T. Y. Berger-Wolf. 2010. Sampling community structure. In *Proceedings of the 19th International Conference on World Wide Web*. 701–710.

A. S. Maiya and T. Y. Berger-Wolf. 2011. Benefits of bias: Towards better characterization of network sampling. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 105–113.

Gurmeet Singh, Manku and Rajeev Motwani. 2002. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*. 346–357.

A. McGregor. 2009. Graph mining on streams. *Encyclopedia of Database Systems*, Springer, 1271–1275.

Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. 29–42.

S. Muthukrishnan. 2005. *Data streams: Algorithms and applications*. Now Publishers Inc.

J. Neville, B. Gallagher, and T. Eliassi-Rad. 2009. Evaluating statistical tests for within-network classifiers of relational data. In *IEEE 9th International Conference on Data Mining*. 397–406.

M. E. J. Newman. 2002. Assortative mixing in networks. *Physical Review Letters* 89, 20 (2002), 208701.

M. Papagelis, G. Das, and N. Koudas. 2013. Sampling online social networks. *IEEE Transactions on Knowledge and Data Engineering* 25, 3 (2013), 662–676.

A. H. Rasti, M. Torkjazi, R. Rejaie, N. Duffield, W. Willinger, and D. Stutzbach. 2009. Respondent-driven sampling for characterizing unstructured overlays. In *IEEE International Conference on Computer Communications*. 2701–2705.

S. Redner. 1998. How popular is your paper? An empirical study of the citation distribution. *The European Physical Journal B-Condensed Matter and Complex Systems* 4, 2 (1998), 131–134.

Bruno Ribeiro and Don Towsley. 2010. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. 390–403.

Ryan Rossi and Jennifer Neville. 2012. Time-evolving relational classification and ensemble methods. In *Proceedings of the 16th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*-Vol. *Part I*. 1–13.

Ryan A. Rossi, Luke K. McDowell, David W. Aha, and Jennifer Neville. 2012. Transforming graph data for statistical relational learning. *Journal of Artificial Intelligence Research* 45, 1 (2012), 363–441.

A. D. Sarma, S. Gollapudi, and R. Panigrahy. 2008. Estimating pagerank on graph streams. In *Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 69–78.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–106.

C. Seshadhri, Ali Pinar, and Tamara G. Kolda. 2013. An in-depth analysis of stochastic Kronecker graphs. *Journal of the ACM* 60, 2 (2013), 1–32.

M. P. H. Stumpf, C. Wiuf, and R. M. May. 2005. Subnets of scale-free networks are not scale-free: Sampling properties of networks. *Proceedings of the National Academy of Sciences* 102, 12 (2005), 4221–4224.

D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. 2006. On unbiased sampling for unstructured peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*. 27–40.

B. Taskar, E. Segal, and D. Koller. 2001. Probabilistic classification and clustering in relational data. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*. 870–876.

N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. 2003. Load shedding in a data stream manager. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*. 309–320.

A. Vattani, D. Chakrabarti, and M. Gurevich. 2011. Preserving personalized pagerank in subgraphs. In *Proceedings of the 28th International Conference on Machine Learning*. 793–800.

Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. 2009. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks*. 37–42.

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematics Software* 11 (1985), 37–57. Issue 1.

H. Wang, W. Fan, P. S. Yu, and J. Han. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 226–235.

H. Wang, P. S. Yu, and J. Han. 2010. *Data Mining and Knowledge Discovery Handbook*. Springer, Chapter 40: Mining Concept-Drifting Data Streams, 789–802.

J. K. Watters and P. Biernacki. 1989. Targeted sampling: Options for the study of hidden populations. *Social Problems* 36, 4 (1989), 416–430.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of small-world networks. *Nature* 393, 6684 (1998), 440–442.

Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P. N. Puttaswamy, and Ben Y. Zhao. 2009. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European Conference on Computer Systems*. 205–218.

R. Xiang, J. Neville, and M. Rogati. 2010. Modeling relationship strength in online social networks. In *Proceedings of the 19th International Conference on World Wide Web*. 981–990.

S. Ye, J. Lang, and F. Wu. 2010. Crawling online social graphs. In *IEEE 12th International Asia-Pacific Web Conference*. 236–242.

Sooyeon Yoon, Sungmin Lee, Soon-Hyung Yook, and Yup Kim. 2007. Statistical properties of sampled networks by random walks. *Physical Review E* 75 (2007), 046114.

J. Zhang. 2010. *Managing and Mining Graph Data*. Springer, Chapter 13: A survey on streaming algorithms for massive graphs, 393–420.

Received November 2012; revised April 2013; accepted May 2013