Methods and Applications of Network Sampling

Chapter · November 2016

CITATIONS
0

READS
255

1 author:

Mohammad Hasan
Indiana University-Purdue University Indianapolis
80 PUBLICATIONS **1,338** CITATIONS

SEE PROFILE

TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2016

 | ISBN 978-0-9843378-8-0

Methods and Applications of Network Sampling

Mohammad Al Hasan
Department of Computer Science, Indiana University Purdue University Indianapolis, IN 46202,
alhasan@iupui.edu

Abstract

Abstract: Network data appears in various domains, including social, communication, and information sciences. Analysis of such data is crucial for making inferences and predictions about these networks, and moreover, for understanding the different processes that drive their evolution. However, a major bottleneck to perform such an analysis is the massive size of real-life networks, which makes modeling and analyzing these networks simply infeasible. Further, many networks, specifically, those that belong to social and communication domains are not visible to the public due to privacy concerns, and other networks, such as the Web, are only accessible via crawling. Therefore, to overcome the above challenges, researchers use network sampling overwhelmingly as a key statistical approach to select a sub-population of interest that can be studied thoroughly.

In this tutorial, we aim to cover a diverse collection of methodologies and applications of network sampling. We will base the discussion of network sampling in terms of population of interest (vertices, edges, motifs), and sampling methodologies (such as Metropolis-Hastings, random walk, and importance sampling). We will also present a number of applications of these methods.

Keywords Network Sampling; Breadth-first Sampling; Markov Chain Monte Carlo Sampling; Metropolis-Hastings Sampling, Importance Sampling

1. Introduction

In recent years, graphs are increasingly being used as the basic representation for data appearing in various domains; examples include road networks, communication networks, file sharing networks, online social and professional networks, citation networks, collaboration or association networks, and biological networks. In these examples, the vertices of a graph typically represent an object, a person, or an entity, and edges represent a communication link, a reference link, a *similarity* link or an actual relationship borne in the real world. Through vertices and edges, a graphical structure provides an effective, compact, flexible, and comprehensible representation for large-scale data. Graphical representation of data also provides an ideal instrumentation for performing machine learning and data mining tasks using hundreds of algorithms that are built over the years for graph data analysis. Another key strength of graph data is its real-life utility for exploratory data analysis, through visualization, and local pattern analysis.

As the scope of graph data representation expands over many disciplines, the diversity of graph analysis tasks increases. A complete listing of these tasks is nearly impossible, but below we list a small set of most common network analysis tasks. The most studied tasks in network analysis is probably the study of the evolution of a network over time for finding an analytical model for network evolution. Complementary to this modeling task is an inference task known as *link prediction*, which predicts the likelihood of a future link

formation between two nodes in a given network. The second most studied task is probably *centrality analysis*, which measures the importance of a node in a network in terms of its topological positioning in the network. Several centrality metrics have been studied, including degree centrality, closeness centrality, betweenness centrality, and eigenvector centrality. Beyond node-centric metrics, such as centralities, some global metrics have also been of interest for analyzing networks; examples include clustering co-efficient, diameter, degree distribution, and hop-count distribution. Between the two extremes, node-centric analysis and global analysis, some studies exist which consider local topological group of nodes and explore topological patterns comprising these nodes; the most studies patterns are triangles, graphlets, network motifs, frequent subgraphs, and network communities. There exist various network metrics that are computed by analyzing frequency distribution of the above patterns. Finally, for graphs that have vertex and/or edge attributes, an important analysis task is the discovery of correlation between attribute values and network structure, which leads to the understanding of the effect of local topological structure on the attribute values of nodes or edges.

Many of the above graph analysis tasks are computationally costly, so for large networks performing such analysis is challenging. Let's assume, n is the number of vertices and m is the number of edges in a graph. Computation of various centrality metrics (such as, closeness centrality, and betweenness centrality) has the same cost as finding all pair shortest path distances in the graph, which has $O(nm \lg n)$ running time. Computing eigenvector centrality using a typical linear algebraic method takes roughly $O(n^3)$ time. Finding local structures in the graph, such as, triangles, graphlets, and community is also costly. For instance, detecting community with Girvan-Newman algorithm, one of the most efficient among the existing methods, takes $O(m^2 n)$ time. Graphlet counting for graphlet up to size k has a worst-case time complexity of $O(n^k)$. Even, counting triangles, which is the most simplest graph analysis task, has the best run time complexity of $O(m^{1.41})$. Some of the pattern discovery tasks, such as, finding frequent subgraph patterns, or finding subgraph motif is $\mathcal{NP}$-complete for arbitrary sized pattern due to the fact that these tasks require to solve numerous subgraph isomorphism problems. For graphs with millions of nodes, none of these tasks can be solved in a reasonable amount of time!

Besides high run-time computational complexity, there also exist other challenges for large-scale network analysis. For instance, many networks are too large to process offline because of their massive size. In October 2012, Facebook reported to have 1 billions users. Using 8 bytes for userID, 100 friends per user, storing the raw edges takes 1 billion $\times$ 100 $\times$ 8 bytes = 800 GB. In-memory graph algorithms are simply infeasible for such a large dataset. Some networks, specifically online social networks, are hidden due to privacy concern. Such networks can only be crawled, where the crawler has no access beyond one-hop neighbors of currently visiting node. This is a severe restriction, because most of the graph analysis algorithms use random access of the vertices or edges in the network and they cannot run on such a restricted setup. Finally, some networks are dynamic, so understanding and analyzing them require multiple snapshots of the network, which worsens the computational complexity of graph analysis methodologies due to the addition of data layers along the temporal dimension. There are two ongoing research efforts to overcome the high computational complexity of typical graph analysis tasks. The first is to use parallel and distributed algorithms and the second is to use sampling. The focus of this tutorial is to discuss sampling based methodologies. Specifically, we will show how to use sampling as a tool for certain network analysis tasks, both for the full and the restricted data access assumption.

To organize our discussion we categorize the overall network sampling methodologies from two orthogonal aspects. The first is the population of interest, i.e. the network objects that we are sampling. We consider two different kinds of population of interest: first, the vertices or edges, and second, connected induced sub-networks, or in short, *patterns*. The second aspect for categorizing network sampling methodologies is the data access assumption. We

consider two different data access assumptions: full access, and one-hop restricted access. In case of full access, the entire network is visible and a random node or a random edge in the network can be selected by a constant time. User has full knowledge of the number of vertices and the number of edges in the network. Typically for any real-life graph analysis task where full access assumption holds, we also assume that entire graph fits in the main memory of the computer. In one-hop restricted access we assume that the network is hidden, however it supports crawling, i.e. for currently visited node, the exploration of its neighbors is permitted. To start the crawling, access to an arbitrary node in the network is available. One-hop restricted access supports very large network, with an assumption that only a part of the network is in the memory, but the remaining parts of the network are on disk or in a graph database.

For evaluating the quality of a sampling method we can first analytically prove that the sampling method in fact obtains samples from the desired distribution. However, when our objective is to estimate the expectation of a network statistics, we need to show that the sampler provides unbiased estimate of the statistics of interest. We can also empirically compute that mean and variance from the samples and then claim that the estimation is probably unbiased by showing that the sample mean is almost equal to the expectation of the statistics. Sometimes we want to generate distribution of a network metric from the samples. For such a sampling task, we can use well-known metrics that compare two distribution for their divergence, for example Kolmogorov-Smirnov (KS) D-Statistics, and KL divergence. For many sampling tasks it is important to know the number of samples that are needed to obtain a desired approximation for an estimation. For this we can use concentration inequalities, such as, Hoeffding bounds and Chernoff bounds.

The remaining parts of this document are organized as follows. In Section 2 we provide a brief discussion of some basic sampling concepts that will be core of many sampling algorithms that we discuss. In Section 3 we discuss methodologies for sampling vertices or edges of a given network. In Section 4 we discuss methodologies for sampling patterns from a given network. In Section 5, we conclude the tutorial.

2. Preliminaries

$G(V, E)$ is a graph where V is the set of vertices and E is the set of edges. We use n and m for representing the number of vertices ($|V|$) and the number of edges ($|E|$). Each vertex in the graph can be uniquely identified by a number between 1 and n. The assignment of identifier can be arbitrary, but it is fixed. We also consider that G is simple, connected, and undirected. Since G is simple, between a pair of vertices u, and v, there exists at most one edge. We define that edge by (u, v) where $u < v$. For a vertex u, we use $d(u)$ to denote the degree value of u, $adj(u)$ to denote the set of u's neighboring vertices, and $inc(u)$ to denote the edges that are incident to u. Likewise, for an edge e, we use $inc(e)$ to denote the incidence vertices of the edge e.

2.1. Triples and Triangles

A triple (u, v, w) at a vertex v is a path of length two for which v is the center vertex. If the other two vertices (u and w) are also connected by an edge, the triple is called a closed triple (triangle), otherwise it is called an open triple. A triangle actually contains three closed triples, one centered on each of its vertices.

We use the symbol Π_v to represent the set of triples that are centered at the vertex v. The set of triples in a graph $G = (V, E)$ is Π, which is the union of the set of triples at each of its node, i.e. $\Pi = \bigcup_{v \in V} \Pi_v$. If the degree of each of the vertices in known, the total number of triples can be computed efficiently as below:

$$|\Pi| = \sum_{v \in V} |\Pi_v| = \sum_{v \in V} \binom{d(v)}{2}. \tag{1}$$

Based on whether the triple is open or closed (in terms of its induced embedding in the graph G), we partition the set Π into $\Pi^{\angle}$ (open triples) and $\Pi^{\triangle}$ (closed triples). Note that, each of the nodes of a triangle in a graph G contributes one distinct triple in the set $\Pi^{\triangle}$. To represent the set of open and closed triples centered at a vertex v, we use $\Pi_v^{\angle}$ and $\Pi_v^{\triangle}$, respectively. If $\mathbf{t}(G)$ is the number of triangles in the graph G, then

$$\mathbf{t}(G) = \frac{1}{3}|\Pi^{\triangle}| = \frac{1}{3}\sum_{v \in V}|\Pi_v^{\triangle}|. \tag{2}$$

In the above equation, the term $\frac{1}{3}$ is needed because each triangle is counted thrice in the total triple count as three distinct triples centered at each of its vertices.

2.2. Transitivity

Newman, Watts and Strogatz [20] defined the transitivity of a graph G (say, $\gamma(G)$) as the fraction that represents the number of closed triples divided by the number of all the triples over the entire network.

$$\gamma(G) = \frac{|\Pi^{\triangle}|}{|\Pi|} = \frac{|\Pi^{\triangle}|}{|\Pi^{\angle}| + |\Pi^{\triangle}|}. \tag{3}$$

Using Equation 2 and Equation 3, the triangle count ($\mathbf{t}(G)$) of a network can be obtained from the transitivity of the network as below:

$$\mathbf{t}(G) = \frac{1}{3} \cdot \gamma(G) \cdot |\Pi|. \tag{4}$$

2.3. Graphlets

Graphlets are collection of all possible connected graphical topologies for a given number of vertices. The graphlets which have k vertices are called k-graphlets. The number of distinct k-graphlets increases exponentially with the value of k. In Figure 3, we show all of the 3-graphlets, 4-graphlets, and 5-graphlets. A graphlet count in a graph is the number of induced embedding of that graphlet in that graph.

2.4. Subgraph Concentration

The frequency of a particular p-subgraph (an induced subgraph with p vertices) topology g in an input graph G is the number of times it appears in G. We denote it by $f_G(g)$. The concentration of g in G is $C_G(g)$, which is defined as the normalized frequency over the cumulative frequency of all the subgraph topologies in the set Λ_p. Mathematically,

$$C_G(g) = \frac{f_G(g)}{\sum_{h \in \Lambda_p} f_G(h)}. \tag{5}$$

2.5. Motifs

A Motif is a subgraph topology which occurs in an input network at a significantly higher frequency than it occurs in a set of random networks with identical characteristics. For this purpose, the random networks are generated from the input network by imposing the constraint that the vertices of a random network have the identical degree distribution as those of the input network. The significance of frequency deviation between the input network and the set of random networks is typically measured using z-score and p-value. If $\overline{f_{G_r}(g)}$ is the mean frequency of g in a set of randomized graphs G_r (constructed from G), and $\sigma_{G_r}(g)$ is the corresponding standard deviation, then z-score of g for the input network G is defined as:

$$z_G(g) = \frac{f_G(g) - \overline{f_{G_r}(g)}}{\sigma_{G_r}(g)}. \tag{6}$$

If the z-score of g is greater than some pre-specified threshold then we call g a motif. Since, setting this threshold requires domain expertise, all the existing motif finding methods consider it as a user-defined parameter. For sampling based solution, we use concentration of subgraph instead of their frequency. Hence, z-score is defined as below:

$$\widehat{z}_G(g) = \frac{\widehat{C}_G(g) - \overline{\widehat{C}_{G_r}(g)}}{\widehat{\sigma}_{G_r}(g)}. \tag{7}$$

In Equation 7, we use $\widehat{C}_G$, and $\widehat{\sigma}_G$ to denote that they are statistics obtain from random sample of appropriately-sized embeddings.

2.6. Inverse Transform Method for Sampling

Let x be a discrete random variable; the probability mass function (PMF) of x is f and the cumulative mass function (CMF) of x is F. Since F is a nondecreasing function, the inverse function F^{-1} can be defined as $F^{-1}(y) = \inf\{x : F(x) \geq y\}, 0 \leq y \leq 1$. If $u \sim \mathbb{U}(0,1)$, i.e. if u is a number generated uniformly between 0 and 1, then $x = F^{-1}(u)$ has CMF F. In other words, x is a sample that is drawn from the distribution f. So, the algorithm for generating a sample from a random variable x with an arbitrary PMF f is the following: First, for the given f, we generate the corresponding CMF function F. Then, we generate a uniform random variable u between 0 and 1 and apply the inverse CMF function to map u to a value of x. The value of x is a sample from the distribution f. This method is called inverse transform method and it is used for sampling from an arbitrary distribution for which PMF function is available.

2.7. Metropolis-Hastings (MH) Algorithm

MH algorithm is a variant of Markov chain Monte Carlo algorithm; its goal is to draw samples from some distribution $p(x)$, called the *target distribution*, where, $p(x) = f(x)/K$; here K is a normalizing constant which may not be known and difficult to compute. MH algorithm can be used together with a random walk to perform Markov Chain Monte Carlo (MCMC) sampling. For this, the MH algorithm draws a sequence of samples from the target distribution as follows:

i It picks an initial state (say, x) satisfying $f(x) > 0$.
ii From current state x, it samples a state y using a distribution $q(x, y)$, referred as *proposal distribution*.
iii Then, it calculates the *acceptance probability* $\alpha(x, y)$ (Equation 8)

The process continues until the Markov chain reaches to a stationary distribution.

$$\alpha(x,y) = \min\left(\frac{p(y)q(y,x)}{p(x)q(x,y)}, 1\right) = \min\left(\frac{f(y)q(y,x)}{f(x)q(x,y)}, 1\right). \tag{8}$$

Note that, sampling using random walk is called an indirect sampling method, as oppose to the sampling directly from CMF, the latter known as direct sampling method.

2.8. Importance Sampling

Importance sampling is a general approach for estimating expectation of a function $f(x)$ relative to some distribution $p(x) = \tilde{p}(x)/K$, called the *target distribution*. This expectation is represented as $\mathbb{E}_p[f(x)]$. Importance sampling is useful in scenarios where it is difficult for obtaining samples from the distribution p. One possible reason for this difficulty is that the normalizing factor K is unknown, another can be p is complex and it does not have a good factorization. Though sampling from p is difficult, it is easy to evaluate $\tilde{p}(x)$ for a given x. Hence, to evaluate $\mathbb{E}_p[f(x)]$ we may prefer to obtain samples from a different distribution $q(x)$, which is called proposal distribution. Theoretically, q can be arbitrary distribution, except that $q(x) > 0$ whenever $p(x) > 0$. However for practical reason, q should be as similar as p, because the sampling quality of importance sampling approach depends strongly on the extent of similarity between the target distribution (p) and the proposal distribution (q). Below, we show the mathematical expression of the expectation $\mathbb{E}_p[f(x)]$ from samples which are obtained from the proposal distribution q.

$$\begin{aligned}\mathbb{E}_p[f(x)] &= \sum_x p(x)f(x) = \frac{1}{K}\sum_x q(x)f(x)\frac{\tilde{p}(x)}{q(x)} \\ &= \frac{1}{K}\sum_x q(x)f(x)w(x) = \frac{1}{K}\mathbb{E}_q[f(x)w(x)] \end{aligned} \tag{9}$$

where $w(x)$ is a function which is equal to $w(x) = \tilde{p}(x)/q(x)$. Now,

$$\begin{aligned}\mathbb{E}_q[w(x)] &= \sum_x q(x)w(x) = \sum_x q(x)\frac{\tilde{p}(x)}{q(x)} \\ &= K\sum_x q(x)\frac{p(x)}{q(x)} = K\sum_x p(x) = K \end{aligned} \tag{10}$$

By using the result from Equation 10 in Equation 9, we obtain

$$\mathbb{E}_p[f(x)] = \frac{\mathbb{E}_q[f(x)w(x)]}{\mathbb{E}_q[w(x)]}$$

If we have M samples from distribution q: $x_1, x_2, \cdots, x_M$, the sample estimate of the expectation of $f(x)$ under distribution p is:

$$\tilde{\mathbb{E}}_p[f(x)] = \sum_{i=1}^{M} f(x_i)w(x_i) \tag{11}$$

where $w(x_i) = \frac{\tilde{p}(x_i)/q(x_i)}{\sum_{j=1}^{M}(\tilde{p}(x_j)/q(x_j))}$.

Sometimes, we may not have a normalized representation of distribution q as well, i.e. $q(x) = \tilde{q}(x)/L$ for some constant L. It can be shown that the sample estimate of the expectation of $f(x)$ under distribution p for this case is:

$$\tilde{\mathbb{E}}_p[f(x)] = \sum_{i=1}^{M} f(x_i)w(x_i) \tag{12}$$

where $w(x_i) = \frac{\tilde{p}(x_i)/\tilde{q}(x_i)}{\sum_{j=1}^{M}(\tilde{p}(x_j)/\tilde{q}(x_j))}$. If the desired distribution is uniform then $\forall x_i,\ \tilde{p}(x_i) = 1$, and $w(x_i) = \frac{1/\tilde{q}(x_i)}{\sum_{j=1}^{M}(1/\tilde{q}(x_j))}$

Table 1. Various Sampling methods for Full Access Assumption

Method Name	Description	Sampling probability, $\pi(u), u \in V$
Random Node Selection (RN)	Select a node uniformly	$\frac{1}{n}$
Random Degree Node Sampling (RDN)	Select a node in proportion to its degree. Another way is to select an edge uniformly and then return either of its endpoint with equal probability.	$\frac{d(u)}{2m}$, $d(u)$ is the degree of vertex u
Random Pagerank Node Sampling (RPN)	Select a node in proportion to its pagerank value. Applies for both directed and undirected networks.	$\pi(u)$ is u'th entry in the stationary distribution vector $\mathbf{p}$ satisfying $\mathbf{p} = \left(cA^T D^{-1} + (1-c)U\right)\mathbf{p}$, where A is the adjacency matrix, D is the degree matrix, $U = \frac{1}{n}\mathbf{1}$ and $1-c$ is teleportation probability.
Random Edge Selection (RE)	Select an edge uniformly	$\sim \frac{d(u)}{2m}$
Random Node-Edge Selection (RNE)	Select a vertex uniformly, and then pick an edge incident to the selected vertex uniformly.	$\frac{1}{n}\left(1 + \sum_{x \in adj(u)} \frac{1}{d(x)}\right)$

3. Sampling Vertices or Edges

For many network studies, we are interested to measure the expectation of a vertex attribute or an edge attribute by sampling. The most common example of such a task is the approximation of average degree or degree distribution of a very large network by sampling vertices from the network. Another example task can be approximating the assortativity by sampling edges in a network. Sometimes networks are attributed, and we want to approximate the expectation of that attribute. An example of such task is to compute the fraction of Facebook users who are *single*. In all the above tasks, we are interested to obtain a uniform sample of vertices (or edges) of the network so that the expectation estimation is unbiased.

Sometimes, we may want to sample from a biased distribution. For instance, say we want to sample a collection of vertices from a very large network and build a network induced by those vertices. If we wish that the network built from the sampled vertices preserves some of the network properties (say, clustering coefficient), uniform sampling is not a very good choice though. The reason for that is the fact that real-life networks exhibit power-law degree distribution, where very few vertices have high degree and uniform sampling of vertices generally misses those vertices. A carefully chosen biased sampling of vertices is a better option for sampling a sub-network preserving real-life network properties. In general, we need biased sampling, if we want to sample vertices (or edges) in proportional to a score defined over the vertices (or edges).

3.1. Sampling by Full Access

Sampling vertices or edges with full access assumption is a simple task. For such a sampling scenario, we assume that the entire network is available in memory in an adjacency vector data structure. We can obtain the adjacency vector of any vertex in $O(1)$ time. We also assume that, in the adjacency vector of a vertex u, the neighbors of u, $\mathcal{N}(u)$ are sorted. So, the existence of an edge (u, v) can be answered in $O(\lg n)$ time using binary search on that vector. We also know the size of the network—the number of vertices, n and the number of edges, m.

In existing works [14, 1], a collection of sampling variants are proposed for sampling vertices and edges considering full access assumption. These variants differ in the distribution from which they sample. Obviously, the most common distribution is uniform distribution as it provides a simple method for approximating the expectation of a function defined over the vertices (or edges). On the other hand, non-uniform sampling preferentially selects a vertex (or an edge) based on some criterion. Table 1 lists some of the most common node and edge sampling variants that are used in existing literature [14]. In this table, we also list the probability by which a node is sampled when each of the above methods is used. Among the methods that are listed in Table 1, RN (Random Node Selection), RDN (Random Degree Node Sampling), and RPN (Random Pagerank Node Sampling) are node sampler, and the remaining two, RE (Random Edge Selection), and RNE (Random Node-Edge Selection) are edge sampler. Note that edge sampling methods also sample nodes that are incident to the sampled edges. Below we provide a brief discussion of each of the above sampling variants.

3.1.1. Vertex Sampling

RN This method selects a node uniformly, i.e. the probability of sampling a vertex, $\pi(u)$ is $1/n$, where n is the number of vertices in the network. The sampling task is trivial since we have full access to the entire network data structure from which we can compute n and then select a vertex with id i, where $i \sim \mathbb{U}(1,n)$, a random number generated uniformly between 1 and n. This is a perfect sampling method for providing unbiased estimation of average degree or degree distribution of a large network. In fact, using this sampling method, we can obtain an estimation of expectation of any function $\alpha(u)$ where α is defined over the node attributes.

RDN This method selects a node in proportion to its degree. If $\pi(u)$ is the probability of selecting the node u, $\pi(u) = d(u)/2m$, where m is the number of edges in the network. Note that, $\pi(u)$ is a probability distribution because $\sum_{u \in V} \pi(u) = 1$. For RDN sampling, we can use inverse-transform method (discussed in Section 2.6)—we simply construct a CMF function, say Π, and sample a node u by $u = \Pi^{-1}(p)$ where $p \sim \mathbb{U}(0,1)$. The complexity of this method is $O(n)$ because the size of the CMF vector (Π) is n, which we must construct first. There exists another efficient method for performing RDN sampling, if the listing of all the edges are available—choose an edge uniformly, then return one of its incident vertices (say, u) with equal probability. This algorithm is degree proportional sampling because the probability of sampling a vertex u is equal to $\pi(u) = \sum_{e \in inc(u)} \frac{1}{2m} = \frac{d(u)}{2m}$. It is obvious that for RDN, high degree nodes have higher change to be selected. Estimation of average degree from the sampled nodes is higher than actual, and degree distribution is biased towards high-degree nodes. Any nodal estimate is also biased towards high-degree nodes.

RPN This method selects a node in proportion to its pagerank. Pagerank is the stationary distribution vector of a specially constructed Markov process on a graph. According to this Markov process, an agent visits a neighbor of currently visiting node by following an adjacent link with probability c (typically kept at 0.85) and jumps to a random node (uniformly) with probability $1-c$ (typically kept at 0.15); $1-c$ is known as teleportation probability. Pagerank vector satisfies the following eigenvector equation: $\mathbf{p} = (cA^T D^{-1} + (1-c)U)\mathbf{p}$, where A is the adjacency matrix, D is the degree matrix, U is a matrix where all entries are $1/n$. From the eigenvector equation it is obvious that $\mathbf{p}$ is the stationary distribution of the Markov process, and the probability, $\pi(u)$ by which the node u is sampled is equal to u'th entry in vector $\mathbf{p}$. Pagerank can be computed efficiently by power iteration method. For an undirected graph, when $c = 1$, the sampling is identical to RDN, i.e. $\pi(u) = \frac{d(u)}{2m}$. On the other hand, when $c = 0$ the sampling is similar to RN, having $\pi(u) = \frac{1}{n}$. For other c values, the sampling probability value is in between the values for the above two cases. Similar to RDN, for RPN also, nodes with high degree have higher chance to be selected. However, due to the presence of uniform random jump the high degree bias is

lessened for the case of RPN. Leskovec et al. [14] have provided empirical evidences that RPN provides better estimation accuracy than RDN for several graph properties, including clustering co-efficient, average degree and degree distribution.

3.1.2. Edge Sampling

RE This is an edge sampling method, where each edge is chosen independently and uniformly at random. Generally, edge sampling is used for obtaining a sample graph constructed with the sampled edges. If E_s is a set of sampled edges, and $\rho = \frac{|E_s|}{|E|}$ is the probability by which an edge is selected, then the probability that a vertex u is selected is $1-(1-\rho)^{d(u)}$; this is due to the fact that a vertex is not selected if none of its incident edges are selected. As $\rho \to 0$, $\pi(u) = \rho \cdot d(u)$. So, when sample size is small, RE samples vertices in proportional to their degree. The selection of vertices are not independent as both endpoints of an edge are selected. Due to the degree bias, nodal statistics is biased towards high-degree nodes. Edge statistics is unbiased due to uniform edge selection. When edge sampling is used for making a sampled network, the sampled network fails to preserve many desired graph properties, specifically it does not preserve clustering and connectivity. It is more likely to capture path length due to its bias towards high degree nodes, and the inclusion of both end points of selected edges [1].

RNE This is also an edge sampling method. In this method first a vertex is selected uniformly, then an edge incident to the selected vertex is sampled, also uniformly. A vertex u can be selected either in the first stage or it can be selected in the second stage as an incident vertex of the selected edge. The probability for u to be selected in the first stage is $\frac{1}{n}$ and in the second stage is $\frac{1}{n}\sum_{x \in adj(u)} \frac{1}{d(x)}$. Thus the total probability for u to be selected is proportional to $\frac{1}{n}\left(1+\sum_{x \in adj(u)} \frac{1}{d(x)}\right)$. If the graph is assortative (social networks exhibit this property), the probability is almost uniform for all the vertices, so nodal estimates are better that the case of RE. Using RNE, edge sampling is non-uniform, edges that are incident to high degree nodes are under-sampled, and those that are incident to low degree nodes are over-sampled.

3.1.3. Computational Complexity of Direct Sampling For sampling vertices from a network uniformly we can simply generate a random number (say, x) between 1 and n and return the vertex with id x. On the other hand, for sampling an edge uniformly we first generate a random number (say, y) between 1 and m, and then return the yth edge by considering a complete order of the edges. For fixing a complete order we can assume that the adjacency vectors of all the vertices are appended into a single vector in the order of their vertex id and an edge's order is simply its position in that single vector. By keeping an additional vector of size n, which lists the cumulative degree of the vertices, we can obtain an edge sample in $O(\lg n)$ time, first by performing a binary search on the cumulative degree vector to select a vertex and then another binary search on the adjacency vector of the selected vertex. For sampling from a non-uniform distribution we can use inverse-transform method, which requires to construct the CMF function; the complexity of generating this function is at least linear—$O(n)$ for vertices, and $O(m)$ for edges.

3.2. Sampling by One-hop Restricted Access

Many networks are not fully accessible. They can only be crawled i.e. an analyst can only explore the neighbors of the currently visiting node. We assume that access to one seed vertex or a collection of seed vertices of the network is available so that the crawling can be initiated. We further assume that the network is connected; if not connected we assume that the largest (giant) connected component of the network covers the majority of the vertices and the part of the network excluding the giant component can be ignored. Since, the network is connected, one can attempt to crawl the entire network by using graph traversal

methodologies and save the network (in memory or disk) for future processing using direct sampling methodologies. However we assume that the network is very large (say, Internet network) and it does not fit in the main memory. So, direct sampling methodologies do not work on such networks, or to the least, such methodologies are highly inefficient.

Sampling vertices or edges from a network in a restricted setting is much more challenging than the full access setting. In restricted access, we cannot apply the inverse transform method for sampling vertices (or edges) of a network, because the network is only partially visible—even, the number of vertices (n) or the number of edges (m) in the network is not known. Hence, a full instantiation of probability mass function and cumulative distribution function cannot be obtained because they need the information of the size of sampling space.

Sampling of vertices or edges from a network in restricted access scenario is performed by using random walk over the vertices of the network. These methodologies can be further divided into two groups: (1) graph traversal techniques, and (2) exploration techniques. We will discuss the following four traversal techniques: breadth-first search (BFS), depth-first search (DFS), forest fire sampling (FFS), and Snowball Sampling (SBS). Then we will discuss the following four exploration techniques: Simple Random Walk (RWS), Random walk with restart (RWR), Random walk with random jump (RWJ), and Markov Chain Monte Carlo (MCMC) using Metropolis-Hastings algorithm (RWMH). The main difference between these two groups is that the methods belonging to graph traversal sample vertices (or edges) without replacement (an object can be sampled at most once), and those belonging to exploration technique sample vertices (or edges) with replacement (same object can be sampled multiple times). Thus, a traversal based sampling method terminates when the entire population is sampled i.e. all vertices are sampled. During the traversal or walk, the visited nodes are considered to be part of the sample set. Since a traversal or an exploration follows the edges of the network, we can also use each of these methods for sampling edges, by selecting the edges that the random walk traces. Now, we discuss each of the above eight methodologies for sampling vertices by one-hop restricted access.

3.2.1. Traversal: Sampling without Replacement

BFS BFS sampling follows the breadth-first graph traversal method. During traversal, every node has one of the following three status: not-discovered, discovered, and visited. BFS sampling is initiate by visiting the seed vertex. At the beginning, the seed vertex is in discovered state and all the remaining vertices are in not-discovered state. Anytime a node is visited, all its adjacent vertices are in discovered state. In each sampling iteration of BFS, the earliest discovered but not yet visited node is visited. The sampling set comprises of the vertices that are visited. BFS sampling is very efficient as it runs in linear time of the number of samples that are needed. The problem of BFS is that it samples nodes from a specific region of the network. In fact, it discovers all nodes within some distance from the seed node. This sampling is biased as high-degree nodes have higher change to be sampled in BFS. Subsequently, nodal estimations are biased towards nodes with higher degree.

DFS Similar to BFS, DFS is another graph traversal method, where the vertices have three status: not-discovered, discovered, and visited. When a node is visited, all its adjacent vertices are discovered. Each iteration of DFS selects the latest discovered, but not yet visited node. DFS generally explores nodes that are far away from the seed. The sampling is biased as high degree nodes have higher chance to be selected. Same as BFS walk, estimation is biased towards nodes with higher degree.

FFS Forest fire sampling is a randomized version of BFS. Using FFS, neighbors of a visiting node are visited with a probability p. For $p = 1$ FFS becomes BFS. FFS has a chance to die before the desired number of samples are obtained. This sampling method is inspired by a graph evolution model [15]. Leskovec et al. [14] have shown that FFS can capture many graph properties, including diameter, and clustering coefficient and hence, it is a good

sampling strategy for obtaining a smaller network from a large network by sampling. When estimating a nodal attribute by sampling, FFS's performance is similar as BFS sampling, i.e. the sampling is biased towards the high-degree nodes.

SBS Snowball sampling is one of the earliest sampling strategy proposed for sampling from hidden networks [9]. It has a parameter n, so it is sometimes called n-name snowball sampling. It is also similar to BFS. While visiting a node v, not all of its neighbors but exactly n of v's neighbors are chosen randomly and put on the visit schedule. A neighbor is chosen only if it has not been visited before. While estimating nodal attributes, the performance of snowball sampling is similar to BFS sampling.

3.2.2. Exploration: Sampling with Replacement

In this section we discuss four exploration based methodologies for sampling vertices (or edges) from a network. Such a method samples with replacement i.e. for vertex sampling, the same node can be sampled multiple times.

RWS Simple random walk is the well-known random walk over the vertices of a network. Starting from a seed vertex, each iteration of RWS chooses one of the neighbors of the currently visiting node, similar to a DFS sampling. However, unlike DFS the sampling process chooses a neighbor node irrespective of its previous visit status; also, each node of the neighbors has the same probability to be selected as the visiting node of the next iteration. If u is the currently visiting node, and v is the visiting node in the next iteration, we have,

$$p(u,v) = \begin{cases} \frac{1}{d(u)} & v \in adj(u), \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

This sampling is biased as high degree nodes have higher chance to be sampled. In fact, the probability that a node u is sampled, $\pi(u) = \frac{d(u)}{2m}$. This can be shown by proving that the stationary distribution of RWS is the vector $\left(\frac{d(u_1)}{2m}, \frac{d(u_2)}{2m}, \cdots, \frac{d(u_n)}{2m}\right)$, where u_i's are all the vertices of the network. Note that this method samples each edge uniformly, as the probability of selecting the directed edge (u,v) is $\pi(u) \cdot p(u,v) = \frac{d(u)}{2m} \cdot \frac{1}{d(u)} = \frac{1}{2m}$. Also, the probability of selecting the directed edge (v,u) is $\pi(v) \cdot p(v,u) = \frac{d(v)}{2m} \cdot \frac{1}{d(v)} = \frac{1}{2m}$. So, the overall selection probability of the edge (u,v) is $\frac{1}{m}$ which is uniform. Nevertheless, RWS's uniform edge sampling is different from that of RE method that we have discussed under full access assumption. For RE the sampled edges can be disconnected, but for RWS the sampled edges make a connected network. Due to the high degree bias, the nodal statistics estimated by RWS is biased towards high-degree nodes.

RWR Random walk with restart behaves like RWS, but at each iteration, RWS restarts the walk from a fixed node, w with a probability $(1-c)$. The stationary distribution of RWR over the nodes models a non-trivial similarity function from the fixed node w. This stationary distribution vector is also known as rooted pagerank of the nodes with respect to the node w. RWS preferentially samples nodes around the neighborhood of w, so the nodal statistics is biased towards the values of the nodes in this neighborhood.

RWJ Random walk with random jump behaves like RWR, but instead of a fixed node the walk jumps to an arbitrary node with uniform probability. This sampling is identical to the pagerank proportional sampling (RPN) that we discussed earlier; in fact, this walk simply simulates the process for which the stationary distribution is the pagerank of the nodes. So, identical to RPN sampling, the probability of selecting a node comes from the stationary distribution vector of this walk which satisfies the following eigenvector equation: $\mathbf{p} = (cA^T D^{(} - 1) + (1-c)U)\mathbf{p}$.

RWMH All of the above exploration based sampling is biased towards high degree node, specifically, a node's sampling probability using simple random walk (RWS) is proportional to the degree of that node. When we want to estimate a nodal statistics we need uniform node sampling; to achieve this we need to find a way to counter-bias the high degree preference. This is not an easy task, because in the restricted access scenario complete knowledge about the sample space is not available. Fortunately, there is a elegant solution known as Monte Carlo Markov chain (MCMC) sampling, which uses random walk for sampling without the complete knowledge of the sample space. The idea of MCMC sampling is to perform a random walk on the sampling space such that the stationary distribution of this random walk aligns with a desired target distribution. Once the walk converges towards the target distribution, any vertex visited by the random walk is a sample from the desired target distribution.

Metropolis-Hastings (MH) is a variant of MCMC algorithm (discussed in Section 2.7), which we can use for accomplishing our goal of uniform sampling of nodes. To apply this algorithm we need to first choose a proposal distribution q. For MH, q can be arbitrary, however we want q which can be computed efficiently. One choice of q can be such that for a currently visited node, u, q chooses one of u's neighbor (say, v) uniformly. Thus $q(u,v) = \frac{1}{d(u)}$. Once the node v is chosen using the proposal distribution, we can compute the accept probability using Equation 8, which is:

$$\begin{aligned}\alpha(u,v) &= \min\left(\frac{\pi(v)q(v,u)}{\pi(u)q(u,v)}, 1\right)\\ &= \min\left(\frac{1/n \cdot q(v,u)}{1/n \cdot q(u,v)}, 1\right), \text{using } \pi(u) = \pi(v) = 1/n\\ &= \min\left(\frac{1/d(v)}{1/d(u)}, 1\right) = \min\left(\frac{d(u)}{d(v)}, 1\right)\end{aligned} \tag{14}$$

In the second line above we have used $\pi(u) = \pi(v) = 1/n$, because we want the target distribution to be uniform, where all the vertices are sampled with identical probability. An important observation is that for the target distribution $\pi(\cdot) = 1/n$, the normalized factor n is canceled out from the denominator and numerator in the final expression of Equation 14, this is extremely important for restricted access case because in such scenario the normalizing factor of the target distribution n is typically not known, yet we can use RWMH for obtaining samples from the desired uniform distribution. The final expression of Equation 14, $\min(d(u)/d(v), 1)$, is intuitive; it shows that a proposal move from a high-degree node to a low-degree node is always accepted with probability one. On the other hand, the proposal move from a low-degree node to a high-degree node is selected with a probability only equal to $d(u)/d(v)$. This counter-bias effect helps us obtaining a uniform sampling of the nodes using this sampling algorithm. In fact, as soon as the chain converges to its stationary distribution, MH algorithm samples the nodes uniformly.

We want to emphasize that although RWMH samples from an identical distribution, the nodes that it samples are not independent. This is due to the fact that for any pair of nodes that are visited consecutively, the latter node is dependent on the earlier node as the latter node is taken only from the neighbors of the earlier node. So, we do not really have iid (independent and identically distributed) samples. Dependency can be decreased by skipping a few nodes between two samples. The number of nodes to skip depends on how good the Markov chain mixes to the desired distribution. Detailed discussion on mixing time bound of Markov chain is outside the scope of this tutorial, interested reader can look into this excellent reference [17].

3.3. Application of Vertex/Edge Sampling

Vertex and edge sampling has numerous applications in social sciences, data mining, and network analysis. In social sciences, vertex sampling, specifically snowball sampling and its variant called respondent driven sampling has been used for surveying hidden population [28] such as drug users, female sex workers, etc. Recently it is also used for estimating the size of a hidden network [10]. In data mining and network analysis, graph sparsification is a key reason for vertex and edge sampling. The main objective of sampling is to obtain a smaller network that preserves many of the topological properties, such as clustering coefficient, diameter, and degree distribution [14]. Sometimes, sampling is used for obtaining a small network that maintains the community structure in a graph [18]. However, the main reason for vertex sampling is to obtain an estimate of any nodal attribute; we discuss this more details in the following paragraphs. Specifically, we will discuss methodologies for estimating average degree and degree distribution of a large network by sampling [6]; nevertheless, the analysis is identical for the estimation of any other nodal attributes.

3.3.1. Average Degree Estimation by Sampling To obtain the expectation of any nodal attribute, we need unbiased samples of nodes. In full access scenario it is easy as we simply need to sample a node uniformly by generating a random number between 1 and n. In restricted access scenarios, uniform sampling of vertices can be performed by using Metropolis-Hastings algorithm as we have seen earlier. Say, $S = \{u_1, u_2, \cdots, u_{|S|}\} \subset V$ is a small set of vertices which are sampled uniformly, then the estimate of average degree $\widehat{d}_{\text{avg}} = \sum_{i=1}^{|S|} d(u_i)$.

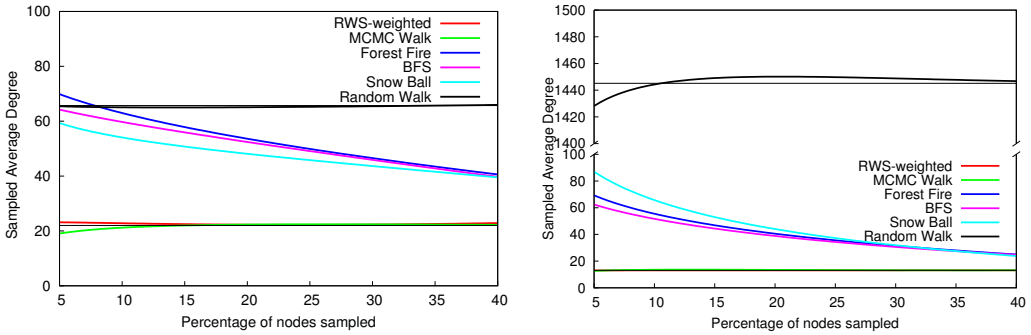
However, for the case of restricted access, if the purpose of sampling is to obtain an unbiased estimate of a nodal attribute (say average degree value), we can use another elegant solution which does not require us to use MH like approach for ensuring uniform sampling. That mean, we can sample vertices from a biased distribution q, and then use a post-processing mechanism to obtain unbiased estimate of a nodal attribute. This solution uses the concept of importance sampling (discussed in Section 2.8). Note that, importance sampling is used for obtaining the expectation of a variable under a distribution p, even if the sampling is performed under a different distribution q.

Say, we are performing a simple random walk (RWS) over a network for estimating the average degree of the vertices in the network through sampling. If $S \subset V = \{u_1, u_2, \cdots, u_{|S|}\}$ is a small set of vertices which is sampled by using RWS; computing expected value of the average degree using $\frac{1}{|S|}\sum_{u_i \in S} d(u_i)$ will be incorrect. It will overestimate the degree value due to sampling bias towards high-degree nodes. In fact, using RWS the probability of sampling a node u_i is $\pi(u_i) = \frac{d(u_i)}{2m}$. The unbiased estimate of average degree is $d_{\text{avg}} = 2m/n$. However, the expected average degree value using RWS biased sampling, $\mathbb{E}_{\text{RWS}}[d_{\text{avg}}]$ is equal to $\frac{(d^2)_{\text{avg}}}{d_{\text{avg}}}$; a proof of this fact is given below:

$$\begin{aligned}\mathbb{E}_{\text{RWS}}\left[d_{\text{avg}}\right] &= \sum_{i=1}^{|S|} \pi(u_i) d(u_i) = \sum_{i=1}^{|S|} \frac{d(u_i)}{2m} \cdot d(u_i) = \sum_{i=1}^{|S|} \frac{d(u_i)^2}{2m} \\ &= \frac{\sum_{i=1}^{|S|} d(u_i)^2}{\sum_{i=1}^{|S|} 2m} = \frac{\sum_{i=1}^{|S|} d(u_i)^2/n}{\sum_{i=1}^{|S|} 2m/n} = \frac{(d^2)_{\text{avg}}}{d_{\text{avg}}}. \end{aligned} \tag{15}$$

Note that,$(d^2)_{\text{avg}}$ is the average of the square of the degree value of the nodes in the graph. Clearly, the average degree is overestimated due to the sampling bias towards high-degree nodes. However, we can use the concept of importance sampling to correct the bias and obtain an unbiased estimate of average degree as the following. We observe, using RWS the probability of sampling a node u_i is $\frac{d(u_i)}{2m}$. Considering the sampling distribution of vertices using RWS as q, we can write $q(u_i) = \frac{d(u_i)}{2m}$ and say p is a uniform distribution vector, $p(u_i) = \frac{\tilde{p}(u_i)}{n} = \frac{1}{n}$. For a vertex u_i, the importance weight $w_i = \frac{\tilde{p}(u_i)}{q(u_i)} = \frac{1}{d(u_i)/2m}$. By

FIGURE 1. The estimate of average degree for different traversal and random walk based sampling: (Left) AstroPh Graph (left), As-Skitter Graph (right)



substituting these values in Equation 11, we obtain unbiased estimate of average degree. Clearly,

$$\mathbb{E}_p\left[d(u)w(u)\right] = \frac{1}{n}\mathbb{E}_p\left[d(u)\frac{1}{d(u)/2m}\right] = \frac{1}{n}\mathbb{E}_p\left[2m\right] = \frac{2m}{n} = d_{\text{avg}}, \tag{16}$$

the estimation of average degree becomes unbiased, as desired.

A crucial problem of using the above formulation in restricted setting is that we do not know n and m, hence we cannot compute $w(u)$. However, note that we have $p(u) = \tilde{p}/n$ where $\tilde{p}$ is a vector of all 1. Similarly, we can take $q(u) = \tilde{q}(u)/2m$, where $\tilde{q}(u)$ is a vector of $d(u)$ for all $u \in V(G)$. Then we can use the Equation 12 for finding the unbiased expectation,

$$\widehat{d}_{\text{avg}} = \sum_{i=1}^{|S|} d(u_i) \cdot w(i), \tag{17}$$

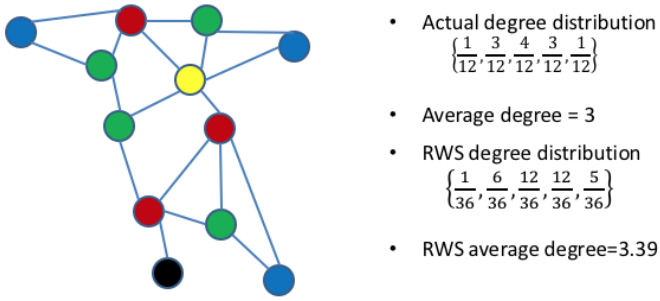
where, $w(u_i) = \frac{\tilde{p}(u_i)/\tilde{q}(u_i)}{\sum_{j=1}^{|S|}(\tilde{p}(u_j)/\tilde{q}(u_j))} = \frac{1/d(u_i)}{\sum_{j=1}^{|S|}(1/d(u_j))}$, Which yields:

$$\begin{aligned}\widehat{d}_{\text{avg}} &= \sum_{i=1}^{|S|} d(u_i) \cdot \frac{1/d(u_i)}{\sum_{j=1}^{|S|} 1/d(u_j)} \\ &= \frac{\sum_{i=1}^{|S|} 1}{\sum_{j=1}^{|S|} \frac{1}{d(u_j)}}, \\ &= \frac{|S|}{\sum_{j=1}^{|S|} \frac{1}{d(u_j)}}.\end{aligned} \tag{18}$$

To use this method for estimating average degree we simply need to keep the cumulative sum of $1/d(u_i)$ over the number of samples; then using Equation 18, we can obtain unbiased estimate of average degree. We will refer to this method as RWS-W (Simple random walk Weighted) in future discussion.

To compare various graph traversal and random walk based methods for their ability to estimate the average degree of large networks, we perform experiments on several large networks from SNAP dataset [16]. The results for two of the networks are shown in Figure 1: AstroPh on the left and As-Skitter on the right. In each of these charts, X-axis represents the percentage of node sampled, and Y-axis represents the estimated average degree, and each curve represents one of the sampling methods. Besides the curves representing the sampling methods, we also draw two horizontal lines (thin black), one at d_{avg} and the

FIGURE 2. Degree distribution: original vs. RWS sampling based estimation



other at $(d^2)_{\text{avg}}/d_{\text{avg}}$ value. The d_{avg} value for these networks are approximately 23 and 13, respectively.

Let's first observe the behavior of the random walk based methods. As we can see RWS-weighted (red curve) and RWMH walk (green curve) estimate the average degree value correctly as those lines almost perfectly align with the black horizontal line representing d_{avg}. The estimation is good even with small number of samples—as low as 5%. As we take more and more samples, the estimation remains stable, which can be seen by the alignment of horizontal red and green lines with the d_{avg} line. The accuracy of both RWS-weighted and RWMH is almost identical. On the other hand, simple random walk (RWS) without any correction (thick black curve) overestimates the average degree and aligns with the $(d^2)_{\text{avg}}/d_{\text{avg}}$ line. In the left plot the alignment is perfect, and in the right plot there is a small fluctuation, which withers away as we take more and more samples. These results agree with the analytical derivation that we have made earlier.

Now, the behavior of traversal based methods is very different. Their estimation start near $(d^2)_{\text{avg}}/d_{\text{avg}}$, and as more and more samples are included, the estimation slowly converges towards its true value, d_{avg}. If the sample size is 100%, it will have the exact estimation. The reason for traversal based methods to have different estimation for different sample size is that these methods sample without replacement; so when new samples are added, its contribution to the mean degree is correctly accounted for. On the other hand, exploration based method samples with replacement, so a new sample may be one which has already been sampled, so its contribution to average degree is biased by its sampling probability. Among the replacement based methods, we do not see much difference in terms of their estimation accuracy. The curves of forest fire (navy blue), snow ball (cyan), BFS (magenta) are packed together for both of the graphs and with more samples the curves slowly get indistinguishable. For smaller sample size (say 5%), in the AstroPh graph snow ball sampling has the best estimation, but it has the worst performance in the other graph. Our finding in this work on the behavior of exploration based sampling is similar to what was reported in Gjoka el al. [7]. Kurant et al. [13] have analyzed the behavior of traversal based sampling and have provided a method for obtaining unbiased estimator using BFS sampling.

3.3.2. Estimating degree distribution For a positive integer value k, let's assume δ_k is the fraction of vertices for which degree is k. If the network is connected each vertices has a degree value at least 1, so we have $\sum_{k>0} \delta_k = 1$ and average degree value is $\sum_{k>0} k \cdot \delta_k$. If the sampling is uniform, node sampling probability, $\pi(u) = 1/n$. Then the expected value of δ_k:

$$\begin{aligned} \mathbb{E}_{\text{uniform}}[\delta_k] &= \sum_{u \in V} \pi(u) \cdot \mathbb{I}_{d(u)=k} \\ &= \frac{1}{n} \delta_k \cdot n = \delta_k \end{aligned} \tag{19}$$

which is unbiased. Expected average node degree is $\sum_{k>0} k \cdot \mathbb{E}_{\text{uniform}}[\delta_k] = \sum_{k>0} k \cdot \delta_k$, which is unbiased.

On the other hand, if we perform simple random walk (RWS), the node sampling probability, $\pi(u) = d(u)/2m$. Then the expected value of δ_k is:

$$\begin{aligned}\mathbb{E}_{\text{RWS}}[\delta_k] &= \sum_{u \in V} \pi(u) \cdot \mathbb{I}_{d(u)=k} \\ &= \frac{k}{2m} \cdot (\delta_k \cdot n) = \frac{k \cdot \delta_k}{2m/n} \\ &= \frac{k \cdot \delta_k}{d_{\text{avg}}},\end{aligned} \tag{20}$$

which is biased. The expected average node degree is:

$$\sum_{k>0} k \cdot \mathbb{E}_{\text{RWS}}[\delta_k] = \sum_{k>0} k \cdot \frac{k \cdot \delta_k}{d_{\text{avg}}} = \frac{1}{d_{\text{avg}}} \cdot \sum_{k>0} k^2 \cdot \delta_k = \frac{(d^2)_{\text{avg}}}{d_{\text{avg}}}, \tag{21}$$

which is also biased. We remind the reader that we have derived this identical expression earlier in Equation 15.

In Figure 2, we show a small network; the color of the vertices represents the degree of the nodes, e.g. blue for nodes with degree 2, red for nodes with degree 3, and so on. There are one degree-1 node, three degree-2 nodes, four degree-3 nodes, three degree-4 nodes, and one degree-5 node. The average degree value is 3. The actual degree distribution and RWS estimated degree distribution are shown. By comparing these two degree distributions we can see that RWS overestimates high-degree nodes, and under-estimates low degree nodes. For example, the actual proportion of degree 2 nodes, $\delta_2 = 3/12$, but RWS under-estimates it as $\frac{k \cdot \delta_k}{d_{\text{avg}}} = \frac{2 \cdot (3/12)}{3} = \frac{6}{36}$.

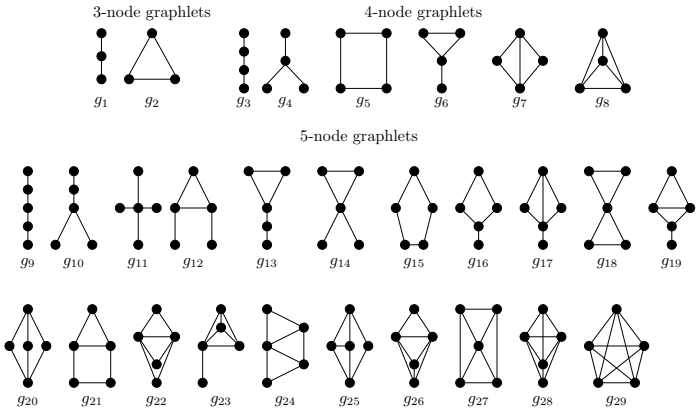
3.3.3. Estimating Edge Attributes Estimating edge attributes in a restricted access sampling is much easier. In fact a simple random walk (RWS), where we choose each outgoing link with equal probability, samples edges uniformly. To show this, consider an edge (u, v); this edge can be sampled by first sampling u and subsequently in the next iteration sampling v. Also, it can be sampled by first sampling v and subsequently in the next iteration sampling u. The probability of sampling in the u, v order using RWS is equal to $\pi(u)p(u \to v) = \frac{d(u)}{2m} \times \frac{1}{d(u)} = 1/2m$. Identically, the probability of sampling in the v, u order using RWS is equal to $\frac{1}{2m}$. Together, the probability of sampling the edge $(u, v) = 1/m$, which is uniform over all the edges.

4. Sampling Network Patterns

In Section 3, we have discussed how to sample vertices or edges from a network. In this section we will discuss how to sample topological patterns from a network. Given a network, this sampling task is to obtain a collection of induced occurrences of connected patterns of a given size from that network. The most common of such a sampling task is triple sampling [12, 25], where the pattern to sample is a 3-node connected triple. A variant of triple sampling is triangle sampling, where the object of interest is triangle—a 3-node clique [25, 30]. Methods also exist for sampling cliques of higher order [8]. Sometimes, researchers want to sample graphlets (defined in Section 2.3)—all induced connected patterns of a given size [24, 23]. There also exist works for sampling subgraphs that are frequent over a database of graphs [27] or for sampling induced subgraphs that are motif (defined in 2.5) [26].

While sampling network patterns, we mainly focus on small, and connected topologies. For instance, the most popular patterns that are sampled are triples and triangles, both have only three vertices. For motif sampling also, topologies that have up to three vertices

FIGURE 3. All 3-nodes, 4-nodes, and 5-nodes undirected Graphlets



are considered [11]. For graphlet sampling, existing works [22, 24, 23] consider graphlets that have up to five vertices, which are shown in Figure 3. The main reason for choosing small patterns for sampling is that for most of the applications of pattern sampling, these patterns are used for capturing local topologies around a vertex, and given the small diameter of real-life networks, topologies that have between three and five vertices are sufficient for capturing the local neighborhood.

The task of sampling topological structures arises in many real-life network analysis tasks. For example, a large number of works exist which sample triangles for obtaining a census of the triangles in a large network [4, 5, 25]. Such census is found to be useful for approximating transitivity of a network [29] and also for query plan optimization in databases [2]. Rahman et al. [24] have used sampling of graphlets for building a frequency histogram of graphlets, which they have used as a fingerprint for clustering graphs arising from different domains. Graphlet frequency histogram is also used in biological network analysis [22]. Milo et al. have shown that small graphical patterns are building blocks of a large network and they play key roles in network's functionalities [19].

A key challenge of sampling graphical structures from large networks is that the sampling space is very large. A real-life network having a few hundred thousands vertices may have billions of triangles. Triangles have only three nodes, if we consider patterns with higher number of nodes, the count can be astronomically large. Also, the number of distinct graphical patterns for a given size increases exponentially, so distinguishing their embeddings on the host network becomes increasingly difficult. Finally, direct sampling using inverse transform method, which instantiates the cumulative distribution function (CDF), is simply impossible for sampling large patterns, so sampling topological patterns is dominated by indirect sampling methodologies using random walk. In the following two sections we will discuss sampling of triples, and graphlets. Like before we will first consider the full access scenario in Section 4.1, then we will discuss the restricted access scenario in Section 4.2.

4.1. Full Access Scenario

In full access scenario we have random access to every vertices and edges of the network. Still direct sampling of network patterns of a given size is difficult due to the reason that the population from which we are sampling is the embeddings of graphical patterns in the given network and we generally do not have an enumeration of these patterns. We also do not have a count of these embeddings so the population size is also not known. When the population size is not known CMF cannot be instantiated, and inverse transform method cannot be applied. Fortunately, for size-3 patterns (also known as triples) the population

size is available (see Equation 1), so triple sampling is easier and several works exist that propose direct sampling methodologies for this task [29]. Below we discuss methodologies for sampling different patterns considering full access assumption.

4.1.1. Triple Sampling The simplest among the topological patterns are triples, which are graphlets of size 3. As we can see in Figure 3, there are two kinds of triples: open triple and closed triple (or triangle). The task of triple sampling is to sample a subset of triples from all the induced embeddings of all kinds of triples in the given network.

One simple triple sampling approach is to take a vertex v uniformly, and then choose two of its neighbors u and w, again, uniformly. Return the triple $\langle u, v, w\rangle$ which is centered around the vertex v. This does not yield uniform sampling as the number of triples centered at the vertices of a network is not uniform. In fact, for a vertex v the number of triples that are centered at v, is exactly $\binom{d(v)}{2}$. Then,

$$p(\text{triple } \langle u, v, w\rangle \text{ is sampled}) = \frac{1}{n \cdot \binom{d(v)}{2}}, \tag{22}$$

by using the above algorithm. So, the triples that are centered around high degree vertices will be under sampled and those that are centered around low degree vertices will be over sampled.

If we want to obtain uniform triple sampling we need to sample the first vertex v in proportional to the number of triples that are centered at it, i.e. proportional to $\binom{d(v)}{2}$ and return two of v's adjacent vertices, u and w, uniformly as before. If Π is the set of triples in a network i.e. $|\Pi| = \sum_{i=1}^{n} \binom{d(u_i)}{2}$. For uniform triple sampling, the probability of sampling v is $\frac{\binom{d(v)}{2}}{|\Pi|}$. The other two vertices u and w are chosen uniformly among all pairs of adjacent vertices of v, so they are selected with probability $\frac{1}{\binom{d(v)}{2}}$. Then,

$$p(\text{triple } \langle u, v, w\rangle \text{ is sampled}) = \frac{\binom{d(v)}{2}}{|\Pi|} \cdot \frac{1}{\binom{d(v)}{2}} = \frac{1}{|\Pi|}, \tag{23}$$

which is uniform. For a full access scenario, we can assume that the degree of all the vertices is known (or can be computed). So, for sampling v, we simply use inverse transform by first building a normalizing triple count vector of size n, and then constructing CDF function of this probability vector. Sampling u and w is trivial as the adjacency list of v is available.

4.1.2. Triangle Sampling Triple sampling samples both open and closed triples. Sometimes, we want to sample closed triples, aka triangles only. Tsourakakis et al. [30] proposed an approximate triangle sampling method, namely DOULION, which works by edge sampling. Given a network, DOULION samples every edge of the network with probability p and with probability $(1-p)$ it discards an edge. The sampled edges make a reduced network. A triangle in the original network is retained in the reduced network with probability $1/p^3$. In this way, this method can be seen as a uniform triangle sampling method, where the triangles in the reduced network constitute the sample set which are obtained through uniform sampling with probability $1/p^3$ from the triangles in the original network. Tsourakakis et al. [30] use DOULION for obtaining approximate count of triangles in the network. If t_s is the number of triangles in the reduced network, then an unbiased estimate of the total number of triangles in the original network is t_s/p^3, because the reduction step keeps every triangle in the original network with probability $1/p^3$.

In another work, Pagh et al. [21] proposed an improved version of DOULION. Say, $G(V, E)$ is a graph. An integer number $N = 1/p$ is chosen, and each vertex in V is given a color value between 1 and N, chosen uniformly. Then all the monochromatic edges (both vertices have the same color) are considered in the set E'. Now, if t_s is the number of triangles in (V, E'), the triangles in this reduced graph are sampled from the original graph with probability p^2 and an approximate estimate of triangles can be obtained by t_s/p^2.

4.1.3. Sampling Graphlet of Size k Sampling graphlets of size $k(>3)$ is more difficult than sampling a triple, for multiple reasons. First, the higher the order of the graphlet, the more complex is it for sampling uniformly. Second, the sampling space, i.e. the number of embeddings of higher order graphlets increases substantially with the size of graphlets, which makes sampling less accurate. Third, for higher order the number of distinct graphlets also increases and identifying the kind of graphlet requires sophisticated graph isomorphism algorithm. For example, for size 3 we have only two graphlets (open triple and close triple), and they can be distinguished by the number of edges; on the other hand we have 21 graphlets for size five, and given a five node connected induced subgraph, recognizing its graphlet type requires more complicated method. Finally, direct sampling requires the knowledge of the size of the sampling space; except size-3 graphlets (triples), we do not have an efficient method for computing the total number of embeddings of patterns for arbitrary k.

Kashtan et al. [11] have proposed the first method for sampling induced k-subgraph through sampling. The method follows a pattern growth approach; it starts from a random edge and extends it by a vertex (along with all induced edges) in each step. Say, the method selects an edge (u_1, u_2) from the graph uniformly in the first step. Then it chooses one vertex, say u_3, also uniformly from the union of u_1 and u_2's adjacency lists. In the next step, it chooses one vertex from the union of adjacency lists of u_1, u_2 and u_3. The process continue until it has a sampled induced subgraph (graphlet) of size k with vertices $u_1, u_2, \cdots, u_k$.

Kashtan's method does not sample a k-graphlet uniformly over all k-sized graphlets in the given graph. Rather, using this method a graphlet in a sparse neighborhood has higher chance to be selected than one in a dense neighborhood; this is so because for the latter case, there are more choices available for selecting a vertex in the extension steps, and hence the generation probability of a specific embedding in a dense neighborhood is small. To make the sampling of Kashtan's method uniform we need to choose the extension vertex (say w) in proportion to the number of graphlets of which w is part-of. But, that information is neither available nor can be computed efficiently. In fact, a prime motivation of graphlet sampling is estimating an approximate count of different graphlets, so counting the number of incident graphlets of each of the vertices obviates the purpose of sampling.

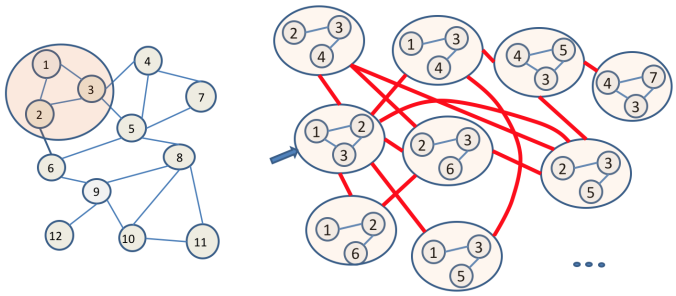
It is also difficult to obtain the probability distribution vector from which each k-graphlets are being sampled using Kashtan's method. However, given a k-graphlet and the order of edges in terms of extension, we can obtain an unnormalized probability expression for the generation probability of that k-graphlet (for the given edge order) by multiplying the probability by which an edge is selected in each of the extension steps. Since a unique embedding of a k-graphlet can be generated by all possible (connected) permutation of its edges, the generation probability of that embedding can be computed by adding the generation probability of each of the feasible edge permutations. We will show later that the generation probability of a pattern is useful for obtaining an unbiased concentration of a k-graphlet.

Wernicke [31] proposes an elegant solution for sampling a set of k-graphlets with uniform probability. This method actually provides an efficient algorithm for enumerating all embeddings of k-graphlets in a given graph. The enumeration process can be shown as a tree where each leaf node is an embedding, and a node at level i is a partial embedding with i vertices. At each level of enumeration tree an enumeration node is expanded with probability p_i. So, all the embeddings at the leaf nodes that are enumerated can be considered as k-graphlet samples. These sampling is uniform because all the leaf nodes are enumerated with an identical probability which is equal to $\prod_{i=1}^{k} p_i$.

4.2. Restricted Access Scenario

Now, we discuss subgraph pattern sampling with restricted access, where at any given node we can access the neighbors of that node. Under such a restricted access we may perform random walk to sample triples or a higher order subgraphs.

FIGURE 4. Left: A network from which triples are sampled by random walk; $\langle 1,2,3\rangle$ is the currently visiting triple. Right: The triple neighborhood graph of the network in the left (incomplete, entire neighborhood graph is not shown).



4.2.1. Sampling Triples One easy way to sample triples uniformly is to perform a random walk on the vertices of the graph so that the stationary distribution of that walk converges to a distribution proportional to the number of triples at a vertex. If π is the desired target distribution of our random walk, for a vertex u, $\pi(u) = \binom{d(u)}{2}$. As we perform this random walk we can simply return a triple uniformly at random such that the sampled triple is centered at the currently visiting vertex. If Π is the set of all the triples, probability of sampling a vertex u is $\frac{\binom{d(u)}{2}}{|\Pi|}$, and the probability to return a triple centered at u is $\frac{\binom{d(u)}{2}}{|\Pi|} \times \frac{1}{\binom{d(u)}{2}} = \frac{1}{|\Pi|}$, which is uniform.

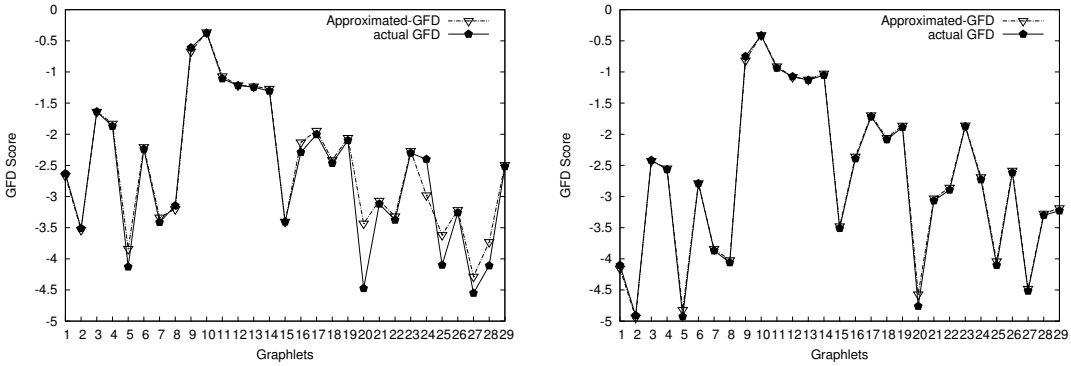
Obviously, for this to work we need to make sure that the stationary distribution of the walk is proportional to $\binom{d(u)}{2}$. One way to ensure this is to use Metropolis-Hastings method. If we choose a proposal distribution q which chooses each neighbor vertex uniformly, for a currently visited node, u, the probability of visiting u's neighbor using q, i.e. $q(u,v) = \frac{1}{d(u)}$. Then we can compute the accept probability using Equation 8 as below:

$$\begin{aligned}
\alpha(u,v) &= \min\left(\frac{\pi(v)q(v,u)}{\pi(u)q(u,v)}, 1\right) \\
&= \min\left(\frac{\binom{d(v)}{2}\cdot(1/d(v))}{\binom{d(u)}{2}\cdot(1/d(u))}, 1\right), \text{using } \pi(u) \propto \binom{d(u)}{2} \\
&= \min\left(\frac{d(v)(d(v)-1)/(2d(v))}{d(u)(d(u)-1)/(2d(u))}, 1\right) = \min\left(\frac{d(v)-1}{d(u)-1}, 1\right) \qquad (24)
\end{aligned}$$

Such a method was used by Rahman et al. [25] for triple sampling in restricted graphs.

In the above works, we perform a random walk on the vertices of the network and sample triples centered at the visited vertex. We can also consider performing random walk over the space of triples. For this, the population is the entire set of triples, and the random walk is performed over the triples by walking from one triple to a neighbor tripe. For this method, we need to define a neighborhood in the triple space; one option can be: two triples t_1 and t_2 are neighbors if they share two vertices. For example, $\langle u,v,w\rangle$ and $\langle v,w,x\rangle$ are neighbors, because vertices v and w are common to both the triples. However note that, in restricted access we cannot enumerate all the triples upfront for making this neighborhood graph on which the random walk is to be performed, rather we construct the neighborhood of a triple lazily, i.e. the possible neighbors of a triple t are enumerated when the random walk is visiting t. In Figure 4 we show a small network (Left) and the corresponding triple neighborhood graph (Right) considering that two triples are neighbors if they share two vertices. The triple $\langle 1,2,3\rangle$ is marked in the given network and also in the triple neighborhood graph.

FIGURE 5. The comparison between exact graphlet frequency distribution (GFD), and approximate GFD constructed using sampling for Hepth Graph (left), As-Skitter Graph (right). Also note that in GFD, the log-scale is used in the y-axis, as the frequency of some graphlets are exponentially larger than some others.



Let's assume that we want uniform triple sampling using the above random walk over the triple space. Say, t is the currently visiting triple. For proposal distribution (say q), we choose one of the triples from t's neighborhood (say, s) uniformly. So, $q(t,s) = \frac{1}{|\mathcal{N}(t)|}$. Here, $\mathcal{N}(t)$ is the set of neighbors of triple t. Using Equation 8, the acceptance probability of the proposal move for achieving uniform triple sampling is obtained as below:

$$\alpha(t,s) = \min\left\{1, \frac{1 \cdot \frac{1}{|\mathcal{N}|(s)|}}{1 \cdot \frac{1}{|\mathcal{N}(t)|}}\right\} = \min\left\{1, \frac{|\mathcal{N}(t)|}{|\mathcal{N}(s)|}\right\} \tag{25}$$

The above method is proposed in [25].

4.2.2. Sampling Graphlets For sampling graphlets, an identical approach similar to triple sampling can be used. Bhuiyan et al. [24] has proposed such an algorithm which they call GUISE. It uses MCMC algorithm for uniformly sampling graphlets of size 3, 4 and 5 (see Figure 3). GUISE defines a neighborhood in graphlet space and then applies MCMC algorithm with a uniform target distribution. Because the sampling is uniform, the frequency of each kind of graphlets in the sample set when normalized, provides a vector what they name graphlet distribution vector (GFD). The objective of GUISE is to obtain GFD approximately through uniform sampling without finding the exact frequency of each of the graphlets in a large network. In Figure 5 we show line plots of histogram of exact GFD and approximate GFD (obtained through sampling) for two large real-life networks. For both the graphs, GFD obtained through sampling is almost identical to those that are obtained by counting each graphlets exactly. But sampling is much more efficient; for many large real-life networks exact counting takes in the order of days, on the other hand sampling takes in the order of minutes.

4.3. Applications of Topological Pattern Sampling

In this section we discuss some of the applications of topological pattern sampling. The primary application of sampling is approximate counting of some patterns. For instance, uniform triple sampling is used to obtain an approximate count of the triangles in a network. Sometimes, we simply want to obtain frequency distribution of various graphlets of a given size through sampling. Sampling is also used for finding network motifs [26]. In this section we will discuss more details of the methodologies how sampling is used for achieving each of the above objectives. We begin with triangle counting.

For triangle counting, we need to compute an unbiased estimate of transitivity (defined in Section 2.2) by finding the fraction of triples that are closed out of all the sampled triples, as is shown in Equation 3. If T is a set of sampled triples, and $\widehat{\gamma}$ is the estimate of transitivity,

$$\widehat{\gamma} = \frac{\sum_{t \in T} \mathbb{I}_{t \text{ is closed}}}{|T|}. \tag{26}$$

then using Equation 4, an approximate estimate of total number of triangles in the graph is equal to $\frac{1}{3}\widehat{\gamma}|\Pi|$, where $|\Pi| = \sum_{i=1}^{n} \binom{d(u_i)}{2}$, the total number of triples in the given graph. Such a method have been used in several works [29, 12, 25].

If we have a sampling method that does not sample triples uniformly, we can use the idea of importance sampling (discussed in Section 2.8) for obtaining an unbiased estimate of transitivity. For instance, the simplest triple sampling method that we discussed at the beginning of this subsection samples each triple in inverse proportional to the number of triples centered at the first sampled vertex, v (see Equation 22). If we want unbiased estimate of transitivity, we need to have a uniform target distribution. But the triples are sampled from a distribution which is proportional to $1/\binom{d(v)}{2}$, so we use important sampling strategy; specifically, we use Equation 12 of importance sampling to obtain an unbiased estimate of transitivity as below. Consider we have a sample set T in which $t = \langle u, v_t, w \rangle$ is a triple, where v_t is the center node of t. The importance weight,

$$w(t) = \frac{\binom{d(v_t)}{2}}{\sum_{x \in T} \binom{d(v_x)}{2}}. \tag{27}$$

Now, the unbiased estimate of transitivity is simply:

$$\begin{aligned} \widehat{\gamma} &= \sum_{t \in T} \left(w(t) \cdot \mathbb{I}_{t \text{ is closed}} \right) \\ &= \frac{\sum_{t \in T} \binom{d(v_t)}{2} \cdot \mathbb{I}_{t \text{ is closed}}}{\sum_{x \in T} \binom{d(v_x)}{2}} \end{aligned} \tag{28}$$

If we perform a simple random walk and return a triple t uniformly among all the triples incident to the currently visited vertex, v_t, the probability of sampling a triple $\langle u, v_t, w \rangle$ is equal to $\frac{d(v_t)}{2m} \times \frac{1}{\binom{d(v_t)}{2}}$, which is proportional to $1/(d(v_t) - 1)$. So, for a triple t, the unnormalized proposal distribution $\tilde{q}(t) = \frac{1}{d(v_t)-1}$ where v_t is the center vertex of the triple t. We again use Equation 12 for obtaining an unbiased estimate of transitivity γ. If T is a set of sampled triples, then

$$\begin{aligned} \widehat{\gamma} &= \sum_{t \in T} \left(w(t) \cdot \mathbb{I}_{t \text{ is closed}} \right) \\ &= \frac{\sum_{t \in T} \left((d(v_t) - 1) \cdot \mathbb{I}_{t \text{ is closed}} \right)}{\sum_{x \in T} \left(d(v_x) - 1 \right)} \end{aligned} \tag{29}$$

here, we used $w(t) = \frac{1/\tilde{q}(e)}{\sum_{i=1}^{M} (1/\tilde{q}(t))} = \frac{d(v_t)-1}{\sum_{x \in T} (d(v_x)-1)}$

For larger graphlets, total number of graphlets of a given size is not known. So, even if we perform uniform sampling we cannot obtain the count of each of the graphlets. However, uniform sampling provides us a way to obtain a normalized frequency vector of graphlets of a given size. As we have mentioned earlier, GUISE algorithm [3] utilizes this to obtain Graphlet distribution vector (GFD). GFD has been found useful for classifying graphs from different domains [24].

Kashtan et al. [11] have used k-subgraph sampling for biological motif discovery. Motif discovery requires computing the concentration of a specific graphlet in a given graph for

which we need uniform samples of graphlets of a given size. We have shown earlier that Kashtan et al.'s sampling method is biased. But, to obtain unbiased concentration estimate we can use the importance sampling strategy that we have discussed earlier. If S is a set of sampled k-graphlet embeddings from a given graph G, the unbiased concentration of a specific k-graphlet g_k using Equation 12 is:

$$\widehat{C}_G(g_k) = \frac{\sum_{s \in S} w(s) \cdot \mathbb{I}_{s_i.type=g_k}}{W} \tag{30}$$

where $W = \sum_{s \in S} w(s)$ and $w(s) = 1/p(s)$, i.e. the inverse of the generation probability. Kashtan's method has many limitations; the foremost is its poor scalability for larger sized graphlets. In fact, the number of possible edge orders for generating a graphlet increases exponentially, and hence the cost of computing the generation probability, $p(s)$ also grows at the same rate. Wernekie's sampling method [31] is better in terms of runtime performance for motif discovery using sampling.

5. Conclusion

In this tutorial, we discussed various methods and applications of network sampling. We covered sampling of two kinds of network objects using two different data access assumptions. We also showed some interesting network analysis results where network sampling is used for achieving very efficient algorithms.

6. Acknowledgment

This tutorial has borrowed contents from some of the slides of "network sampling tutorial" which I gave at KDD 2013 (Chicago, IL), ICDM 2013 (Dallas, TX), and SDM 2015 (Vancouver, Canada) conferences. Dr. Jennifer Neville, Purdue University and Dr. Nesreen Ahmed, Intel Research Labs were my co-presenters in those earlier tutorials. I sincerely appreciate their helps and supports in preparing many of those slides. Mohammad Hasan's research on sampling based network analysis is supported by NSF CAREER Award (IIS-1149851).

References

[1] Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data*, 8(2):7:1–7:56, June 2013.

[2] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

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

[4] E. Charalampos E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *2008 IEEE 8th International Conference on Data Mining*, pages 608–617, 2008.

[5] E. Charalampos E. Tsourakakis. Counting triangles in real-world networks using projections. *Knowledge and Information Systems*, 26:501–520, 2011.

[6] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

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

[8] Minas Gjoka, Emily Smith, and Carter T. Butts. Estimating clique composition and size distributions from sampled network data. *CoRR*, abs/1308.3297, 2013.

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

[10] Mark S. Handcock, Krista J. Gile, and Corinne M. Mar. Estimating hidden population size using respondent-driven sampling data. *Electronic Journal of Statistics*, 8(1):1491–1521, 2014.

[11] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

[12] Tamara G. Kolda, Ali Pinar, and C. Seshadhri. Triadic measures on graphs: The power of wedge sampling. In *SIAM Data Mining*, pages 10–18. SIAM, 2013.

[13] Maciej Kurant, Athina Markopoulou, and Patrick Thiran. Towards Unbiased BFS Sampling. *IEEE J. Sel. AREAS Commun.*, 29(9):1799–1809, 2011.

[14] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, New York, NY, USA, 2006. ACM.

[15] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.

[16] Jure Leskovec and Andrej Krevl. {SNAP Datasets}: {Stanford} Large Network Dataset Collection. \url{http://snap.stanford.edu/data}, jun 2014.

[17] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[18] Arun S. Maiya and Tanya Y. Berger-Wolf. Sampling community structure. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 701–710, New York, NY, USA, 2010. ACM.

[19] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[20] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *PNAS*, 99(Suppl 1):2566–2572, 2002.

[21] Rasmus Pagh and Charalampos E. Tsourakakis. Colorful triangle counting and a mapreduce implementation. *CoRR*, abs/1103.6073, 2011.

[22] N. Przulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

[23] Mahmudur Rahman, Mansurul A. Bhuiyan, and Mohammad Al Hasan. GRAFT: An efficient graphlet counting method for large graph analysis. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2466–2478, Oct 2014.

[24] Mahmudur Rahman, Mansurul Alam Bhuiyan, Mahmuda Rahman, and Mohammad Al Hasan. GUISE: a uniform sampler for constructing frequency histogram of graphlets. *Knowledge and Information Systems*, 38(3):511–536, 2014.

[25] Mahmudur Rahman and Mohammad Al Hasan. Sampling triples from restricted networks using MCMC strategy. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1519–1528, 2014.

[26] Tanay Kumar Saha and Mohammad Al Hasan. Finding network motifs using MCMC sampling. In *Complex Networks VI - Proceedings of the 6th Workshop on Complex Networks CompleNet 2015, New York City, USA, March 25-27, 2015*, pages 13–24, 2015.

[27] Tanay Kumar Saha and Mohammad Al Hasan. Fs^3: A sampling based method for top-k frequent subgraph mining. *Statistical Analysis and Data Mining*, 8(4):245–261, 2015.

[28] Matthew J. Salganik and Douglas D. Heckathorn. Sampling and estimation in hidden populations using Respondent-Driven sampling. *Sociological Methodology*, 34(1):193–239, 2004.

[29] T. Schank and D. Wagner. Approximating clustering-coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.

[30] Charalampos E. Tsourakakis, U Kang, Gary L. Miller, and Christos Faloutsos. Doulion: Counting triangles in massive graphs with a coin. In *Proceedings of the Fifteen ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2009.

[31] S. Wernicke. Efficient detection of network motifs. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 3(4):347–359, 2006.