Estimating Graph Degree Distributions: A Survey

Jerry Lai, SID 3031843855, thejerrylai@berkeley.edu

December 11, 2020

Abstract
Estimating the degree distribution of a graph with minimal queries is a task with many practical applications. Currently, there are no proven optimal upper bounds with established lower bounds. This survey covers basic algorithms that sample vertices, edges, and walks, as well as a few papers that show advanced algorithms with promising performance.

1 Introduction

1.1 Background

The degree distribution of a graph is a fundamental and important property that is frequently used in the analysis of large graphs. However, often, due to the size of these graphs or the ways in which they are presented, it would be impractical to attempt to find the exact degree distribution.

Instead, sublinear algorithms are used to estimate the degree distributions with high probability. These algorithms sample or store data from a small subset of the nodes or edges in the large graphs, then use different methods to approximate the degree distribution of the entire graph, usually to an acceptable level of accuracy and precision.

In general, these algorithms fall into three main categories: Vertex sampling, Edge sampling, and Walk or Crawl sampling. [3]

- Vertex sampling These algorithms randomly or pseudo-randomly sample a small subset of the vertices of a large graph then estimate the degree distribution using the properties of the vertices in its subset. Generally, these algorithms are very accurate when estimating the head of the distribution (usually the low-degree side of the distribution), as that is where most of the sampled vertices lie, but they are woefully inadequate at estimating the tails of the distribution. [2] Examples of this are:

- Edge sampling These algorithms randomly sample edges, then store the degree distributions of the vertices that are adjacent to these edges. Unlike vertex sampling, these tend to sample high-degree vertices, making them prone to overestimating the tails (high-degree ends of the distributions). In cases where a graph is presented as a stream of edges, edge sampling is the best way to sketch the degree distribution of the entire graph using memory that is sublinear to the size of the graph. Examples of this are:

- Walk or Crawl sampling These algorithms sample vertices by sampling the neighbors of previously sampled vertices. Like edge sampling, they are often biased to overestimate the tail of the distribution, as they are more likely to sample vertices with high degrees. In many real-life scenarios, randomly selecting edges or vertices is impossible. For example, if one tries to model the degree distribution of a graph representation of websites on the internet, the only easy way to visit other websites is to visit adjacent sites through hyperlinks. Examples of this are:

This survey includes previous work and algorithms in each of the three categories, as well as hybrid algorithms that sample vertices in two or more of the aforementioned ways.

1.2 Problem Statement

- Suggested by: C. Seshadhri
- Source: WOLA 2019
- Open Problems link: https://sublinear.info/98

Specifically, we are trying to estimate the degree distribution of a graph using the minimum number of queries to access the graph.

In the original formulation of the problem, we are allowed to use three types of queries:

- sampling a vertex uniformly at random
- querying the degree of a given vertex
- sample a neighbor of a given vertex uniformly at random.

The degree distribution that we are trying to approximate takes the form of a complementary cumulative distribution function of the degrees of all vertices.

With $n(d)$ defined as the number of vertices with degree d, the CCDF $N(d)$ is defined as:

$$N(d) \overset{\text{def}}{=} \sum_{d' \geq d} n(d'), \qquad d \geq 0\,.$$

Problem Goal

Our goal is to obtain a $(1 \pm \epsilon)$ approximation $\hat{N}$ of N such that for all d:

$$(1-\varepsilon)N((1-\varepsilon)d) \leq \hat{N}(d) \leq (1+\varepsilon)N((1+\varepsilon)d)\,.$$

As an example, here is the output of one of the algorithms (Eden et.al, 2018) [2] covered by this survey. The graph in question is a representation of the connections in the Gowalla social network.

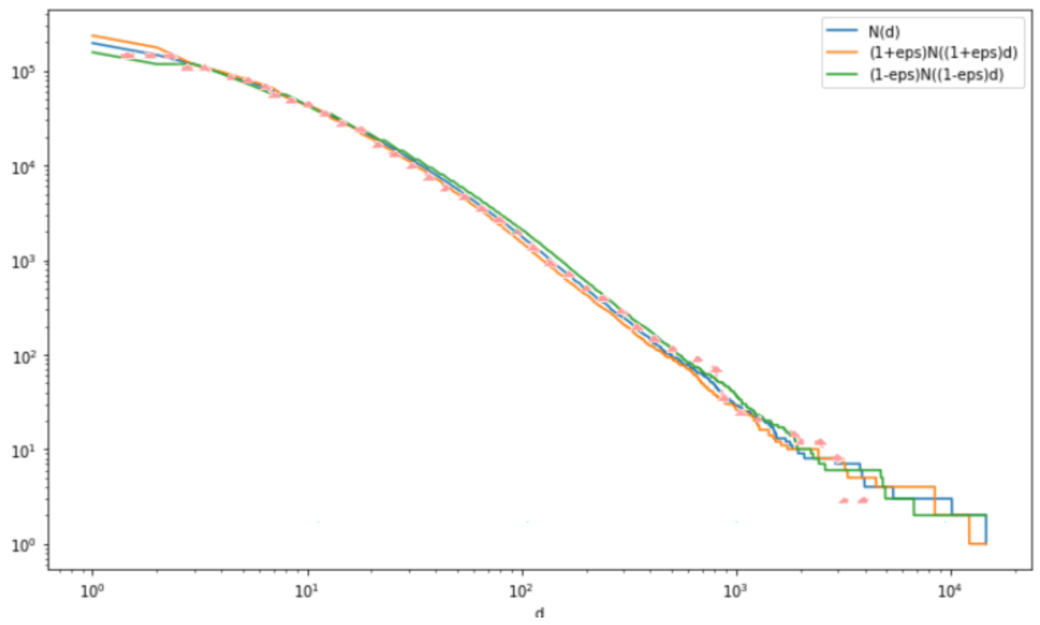

Figure 1: The pink dots are the output of the algorithm. They fall within the acceptable interval ($\epsilon = 0.2$) for most of the graph but become inaccurate towards the tail.

1.3 Evaluation Criteria

These algorithms are measured in two ways:

- Efficiency. The original criterion of the problem: How many query an algorithm needs to make in order to provide an accurate estimate.
- Accuracy. In some situations, especially with larger graphs, it is impractical to run the full algorithm; instead, the number of samples that could be taken is limited and the algorithm is evaluated on how well it can estimate the degree distribution.

2 Vertex Sampling

2.1 Naive Vertex Sampling

2.1.1 Description

Random Vertex Sampling is a simple algorithm that is usually used as a baseline that researchers compare against when coming up with their own algorithms.

The algorithm itself is very simple: it selects vertices at random, calculates the degree distribution of the selection, then scales it up.[7]

2.1.2 Performance

According to several papers on more advanced sampling algorithms, such as [2] (which introduced SADDLES), [5] (which introduced the inverse method), and [4] (which introduced Frontier Sampling), Vertex Sampling provides accurate distribution estimates of the head of the distribution, but fails at its tail as there are not enough samples to provide a reliable estimate. This is shown in Figure 2, where naive VS is represented by dark green.

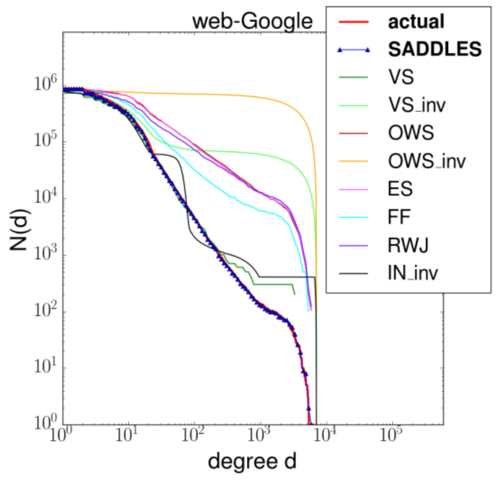

Figure 2: This image, taken from [2], shows the performance of some algorithms after sampling 1 percent of the graph. This graph is from google and represents the internet as seen from the POV of its web crawlers.

2.2 Inverse Vertex Sampling

Paper: [5]

2.2.1 Description

In a paper by Zhang et. al, sampling a graph as a degree distribution is posed as an inverse problem. It defines a matrix $\tilde{P}$ to represent the nature of the sampling algorithm (not the graph itself). It then establishes that the estimated distribution, if defined as a vector, is the product of $\tilde{P}$ times the true degree distribution. Thus, the true distribution can be solved by finding the inverse of $\tilde{P}$. This may be made possible by the fact that $\tilde{P}$ is not dependent on the graph.

2.2.2 Performance

Unfortunately, as the paper itself highlights, this inverse problem is potentially ill-posed [5], as $\tilde{P}$ is often ill-conditioned.

Figure 2 shows that, in the example shown, solving the inverse problem for vertex sampling (shown by Lime Green) makes it only slightly better at estimating the trail of the distribution but without as much precision as edge sampling. Solving the inverse problem for One Wave Snowball (not covered by this survey) also does not present a noticeable improvement.

3 Edge Sampling

3.1 Naive Edge Sampling

3.1.1 Description

Random Edge Sampling is a simple algorithm that is usually used as a baseline that researchers compare against when coming up with their own algorithms.

The algorithm itself is very simple: it selects edges at random, queries the degrees of one or both of the vertices of the edge, then scales up the degree distribution of sampled vertices.[3],[5] Random edge sampling does not fall within the standard model of the problem posed: the three allowed queries do not allow for selecting a random edge. However, in streaming models, which are a good representation of dynamic networks, edge sampling is easier than vertex sampling. [6]

3.1.2 Performance

According to [2], [6], and [4], Edge Sampling provides a better estimate of the shape of the tail than any Vertex Sampling algorithm. This is shown in Figure 2, where naive edge sample is Magenta.

3.2 SADDLES

Paper: [2]

3.2.1 Description

This algorithm, published in 2018 by Eden et. al, is currently the most efficient overall algorithm for estimating distributions. It attempts to combine the strengths of both vertex sampling at the head of the distribution and edge sampling at the tail of the distribution.

The algorithm described in the paper is as follows:

1. Sample a set of r vertices and store in set R. Store the degree distribution in variable X.

2. Sample a set of q vertices from R, with the probability of choosing any vertex v being (degree of v)/(total degree of R).

   From these vertices, sample a neighbor from each u.a.r and store the degree distribution of the neighbors in variable Y.

3. Use X to calculate the unscaled version of $N(d)$.

   If it is below a certain threshold, replace this value by using Y instead. Multiply the value calculated from Y by (total degree of R)/(q) so that this value is equally scaled w.r.t the values calculated from X.

4. Multiply the unscaled $N(d)$ by n/r to get $N(d)$.

With this algorithm, at lower degrees (high $N(d)$), this graph is exactly the same as random VS, but at higher degree (low $N(d)$), it simulates random edge selection. This would ideally give it the best of both worlds.

3.2.2 Performance

This algorithm has been proven in its paper to achieve a (ϵ, ϵ) approximation of $N(d)$ with a number of queries bounded by:

$$\mathcal{O}\left(\frac{n}{h} + \frac{m}{\min_d(d \cdot N(d))}\right)$$

Here, m is the number of edges in the graph, n the number of vertices, and h is the value where $h = N(h)$. In Figure 2, for example, h is approximately 420.

4 Random Walk Algorithms (RW)

4.1 Random Walk with Jump (RWJ)

4.1.1 Description

The Random Walk with Jump algorithm starts with a random vertex, then walks along a random path, visiting its neighbors. With a constant probability, it jumps to a different random vertex and continues the walk there.

4.1.2 Performance

As shown in Figure 2, RWJ provides remarkably similar results to random edge sampling, with accurate estimations of the shape of the tail but also biased towards the tail. This makes the algorithm useful if one would like the performance of Edge Sampling, but random selection of edges isn't available. As previously mentioned, Edge Sampling technically isn't allowed in the standard formulation of the problem, making this a good alternative.

Additionally, in some real-life scenarios, such as web crawling, visiting a neighbor vertex is less costly than visiting a random vertex, making this a potential alternative to VS as well.

4.2 Frontier Sampling (FS)

Source: [4]

4.2.1 Description

This algorithm, similarly to RWJ, uses multiple random walks. Unlike RWJ, it does all the walks simultaneously as follows:

1. Start with a random selection of vertices L.
2. Pick a vertex u from L with with probability $deg(u)/\sum_{v \in L} deg(v)$
3. Replace u with a random neighbor (walking a step) and sample the edge or vertex.

The key difference between FS and simply doing RW or RWJ a bunch of times is that at every time step in FS, it is the higher-degree vertices that get to walk. Ribeiro et. al [4] show that this achieves better performance.

4.2.2 Performance

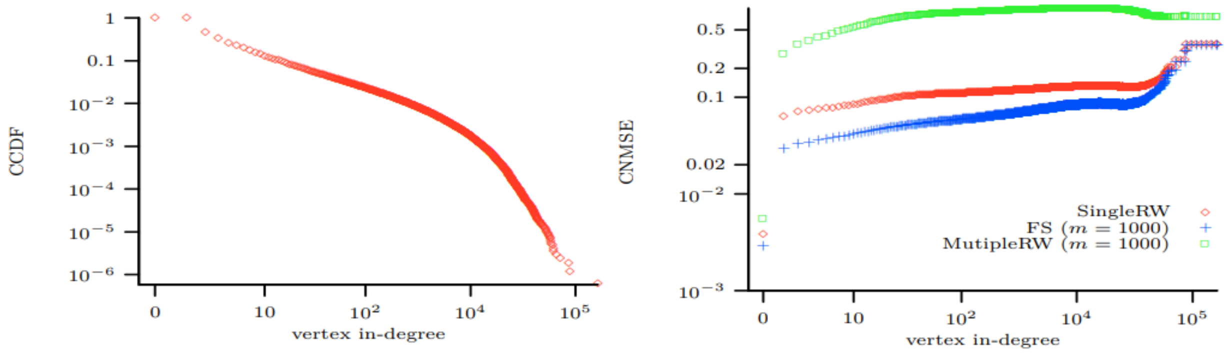

Figure 3: This image, taken from [4], shows the performance of FS. The left graph is the CCDF that the algorithm was trying to estimate, and the right graph is the error of different RW algorithms. The graph is a social network graph from Flickr.

Figure 3 shows that Frontier Sampling achieves better accuracy than both doing a single random walk, and doing random walks. In the figure, the variable m refers to the number of random walks carried out (in the case of MultipleRW) and the number of walkers (in the case of Frontier Sampling).

References

[1] P. Basuchowdhuri, M. K. Shekhawat and S. K. Saha, "Analysis of Product Purchase Patterns in a Co-Purchase Network," 2014 Fourth International Conference of Emerging Applications of Information Technology, Kolkata, 2014, pp. 355-360, doi: 10.1109/EAIT.2014.11.

[2] Eden, Talya, Shweta Jain, Ali Pinar, Dana Ron, and C. Seshadhri. "Provable and practical approximations for the degree distribution using sublinear graph samples." In Proceedings of the 2018 World Wide Web Conference, pp. 449-458. 2018.

[3] B. Ribeiro and D. Towsley. On the estimation accuracy of degree distributions from graph sampling. In Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, pages 5240–5247. IEEE, 2012.

[4] Bruno Ribeiro, and Don Towsley. (2010). Estimating and Sampling Graphs with Multidimensional Random Walks.

[5] Zhang, Y., Kolaczyk, E. D., & Spencer, B. D. (2015). Estimating network degree distributions under sampling: An inverse problem, with applications to monitoring social media networks. The Annals of Applied Statistics, 9(1), 166-199.

[6] Hasan, M, Ahmed, N, & Neville, J. Methods and Applications of Network Sampling

[7] Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. 2013. Network Sampling: From Static to Streaming Graphs. ACM Trans. Knowl. Discov. Data 8, 2, Article 7 (June 2014), 56 pages. DOI:https://doi.org/10.1145/2601438