Noise Corrected Sampling of Online Social Networks

MICHELE COSCIA, IT University of Copenhagen, Denmark

In this paper, we propose a new method to perform topological network sampling. Topological network sampling is the process through which an analyst can extract a subset of nodes and edges from a larger network, such that analyses on the sample provide results and conclusions comparable to the ones they would return if ran on whole structure. We need network sampling because the largest online network datasets are accessed through low-throughput API systems, rendering the collection of the whole network infeasible. Our method is inspired by the literature on network backboning, specifically the noise-corrected backbone. We select the next node to explore by following the edge we identify as the one providing the largest information gain, given the topology of the sample explored so far. We evaluate our method against the most commonly used sampling methods. We do so in a realistic framework, considering a wide array of network topologies, network analysis, and features of API systems. There is no method that can provide the best samples in all possible scenarios, thus in our result section we show the cases in which our method performs best and the cases in which it performs worst. Overall, the performance of noise corrected network sampling is high, as it has the second best rank average among the tested methods.

CCS Concepts: • **Information systems → Social networks**.

Additional Key Words and Phrases: network sampling, network backboning, social media, social networks

ACM Reference Format:
Michele Coscia. 2018. Noise Corrected Sampling of Online Social Networks. *J. ACM* 37, 4, Article 111 (August 2018), 16 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Nowadays, humanity produces data – particularly network data – at an unprecedented pace. For instance, in January 2020 Facebook alone reached 2.5 billion active users[1], producing new content and connections on its platform. These large online social media are the most valuable resources to study global-scale social phenomena. However, accessing their data needs to happen via their API platforms, which severely reduce the accessible throughput. For this reason, a network scientist needs to select wisely the subset of nodes and edges they access. In other word, they need to perform topological network sampling.

Topological network sampling is the process through which one extracts a subset of nodes and edges from a larger network. Typically, the starting point is one or more "seed nodes". The algorithm collects all connections from the seed nodes and then has to decide which next node to explore, selecting it among the unexplored nodes discovered via the edge collection. The aim of

[1]https://www.theverge.com/2020/1/29/21114130/facebook-q4-2019-earnings-instagram-whatsapp-messenger

Author's address: Michele Coscia, mcos@itu.dk, IT University of Copenhagen, Rued Langgaards Vej 7, Copenhagen, 2300, Denmark.

0004-5411/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

network sampling is to reconstruct a sample able to support results and conclusions comparable to the larger whole network. For this reason, many different methods have been defined.

The difference between two methods lies in the criterion they use to select the next node to explore. The exploration strategy has a massive effect on the potential results of the analyses on the sample. That is because the size of the samples are usually negligible compared to the original structure. For instance, by abiding to Twitter's API limitations, a one-year sampling process would only collect 0.14% of its monthly active users – assuming no connectivity issues.

For this reason we propose a new approach to topological network sampling. We are inspired by the problem of network backboning. In network backboning, the aim is to identify the most significant connections in a dense network. We specifically focus on our previous work on the noise-corrected backboning strategy [5]. In noise-corrected backboning, we implement a Bayesian framework, which associates to each connection the amount of surprise – i.e. information gain – that the connection gives us about the whole structure. If we assume that we are observing an incomplete network, which a network sample is, then the information gain guides us towards the edges we should follow, because they are the ones which would allow us to discover more about the whole network.

We evaluate our approach against the most commonly used sampling methods. The space of application scenarios and possible analyses in network science is vast. To evaluate network sampling, we cover a wide array of dimensions: (i) different network topologies (from random graphs to clustered networks with communities); (ii) different network analyses (from estimating the exponent of the degree distribution to community detection); and (iii) different API systems. The latter is an important dimension because, as we see in experiments and previous works [6], it influences the amount of information obtainable from a social media in non trivial ways.

Given the vastness of the search space, it is impossible to define a single network sampling strategy able to constantly provide the best samples. The aim of a paper presenting a new sampling strategy such as this one is to highlight in which scenarios the new algorithm is a valuable – if not the best – alternative to what is already present; and in which other scenarios it should be avoided. For this reason, in our experiment section, we show the cases in which the noise corrected sampling performed relatively best among the alternatives, and the ones in which it performed worst.

Overall, we believe that our approach is a valuable addition to the network sampling literature. We base our conviction on the fact that, overall, its average rank among the tested alternatives is good. It is the second best overall method, surpassed only by the Metropolis-Hastings Random Walk sampler.

2 RELATED WORK

This paper deals with the problem of sampling social network data via API systems. There are many ways in which one might want to sample a network. The main two splits in the literature are whether we perform a topological or an induced sampling; and whether we are performing an online sampling on an evolving structure [23] or a static one. For this paper, we focus on static network sampling. The reason is that online social media are vast, and thus their topological characteristics change slowly, on much larger time scales than the ones typically involving most crawl strategies.

In induced sampling, one determines a set of nodes (node induced) or a set of edges (edge induced) and then generates a sample by collecting all nodes/edges adjacent to the sample. Popular edge sample methods are Partially and Totally Induced Edge Samples (TIES and PIES [1, 2]), while node sample methods are either fully uniform, or degree (RDN) or PageRank weighted samples (RPN)

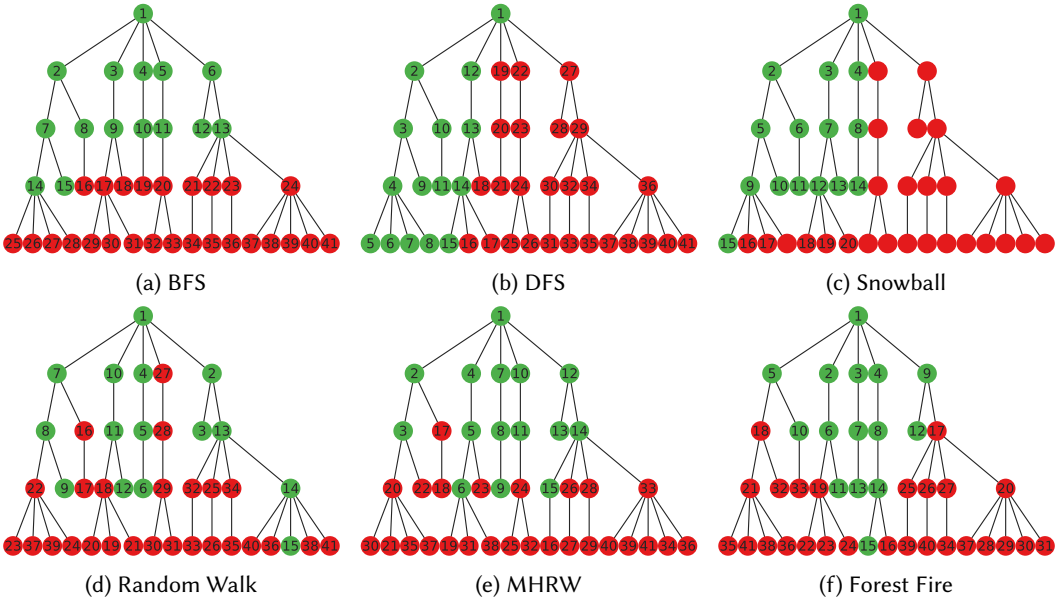


Fig. 1. Examples of how different network sampling strategies explore a given network. Each node is labeled with the order in which it is explored. The node color shows whether the node was sampled (green) or not (red), assuming a budget of 15 units and a constant cost of 1 unit per node. Snowball assumes $n = 3$ (unlabeled nodes are not explored due to this parametric restriction), while Forest Fire has a burn probability of .5.

– see [14] for a discussion of these methods. We choose not to focus on these methods, because usually social media APIs do not allow for an easy application of such strategies.

In topological sampling, one chooses a random starting point and then explores the network with a walk along its edges. The walk can be deterministic – as in classical Depth or Breadth First Search or Snowball sampling – or probabilistic – random walks [12, 15, 22, 24, 27] or probabilistic modification of the deterministic approaches such as forest fire [14] or Neighbor Reservoir Sampling [3, 16].

In Breadth-First Search (BFS) [20, 21], we start from a seed v and we put its neighbors in a First-In, First-Out (FIFO) queue. In this case we call v a parent, and its neighbors are its children. Children of the same parent are siblings. We then explore the neighbors one by one, inserting the neighbor's neighbors in the FIFO queue. All the siblings of a node will be explored before switching to their children. Figure 1(a) shows an example of this approach. Depth-First Search (DFS) [28] is similar to BFS, but uses a Last-In, First-Out (LIFO) queue instead of a FIFO (see Figure 1(b)).

Snowball sampling [9] is almost identical to BFS, but only explores at most n neighbors of a node (Figure 1(c)). In social media data gathering, there is often little cost in exploring all neighbors of a node: in that case Snowball is equivalent to BFS, and that is the reason why we do not include it in our experiment section.

When sampling a network using random walks (RW), the procedure starts from v and chooses one of its neighbors at random [14, 15]. The same process is repeated for every node visited. In many practical applications, random walk sampling takes a parameter p which establishes a probability of restarting the walk from the seed v, or teleporting to an arbitrary node. Figure 1(d) shows an example of this approach.

Exploring a network through random walks has known biases. The probability of visiting a node is proportional to its degree. As a consequence, high degree nodes are oversampled, leading to representativeness issues. Network scientists have developed a Metropolis-Hastings correction for random walk sampling (MHRW) [12, 27]. The network is explored via random walks, but moving from the currently explored node v to its neighbor u is done with probability k_v/k_u, where k_v is v's number of connections (degree). In practice, if u has a higher degree than v, there is a chance that we will attempt to select a different u' to continue the random walk. Figure 1(e) shows an example of this approach.

The final approach we consider in the random walk family is the re-weighted correction [22, 24] (RWRW, also known as Respondent Driven Sampling, or RDS). In RWRW, the network exploration is conducted with the vanilla RW approach described above. Once the exploration is done, each statistical property of $G - \pi -$ is reconstructed with the formula:

$$\pi_i = \frac{\sum_{v \in A_i} k_v^{-1}}{\sum_{v \in V} k_v^{-1}},$$

where i is the value of the property, e.g. the degree, A_i is the set of nodes with exactly i value for the property, V is the full set of nodes, $k_v = i$ in the numerator and equal to the value of the property of node v in the denominator, and p_i is the estimated probability of observing the value i in the full network. There is no need to show the G' sample extracted by RWRW, because it is equivalent to the one extracted by RW. Therefore, the downside of RWRW is that it can only return a correct estimation of G's properties, but not a G' itself that can be then used as input for any arbitrary algorithm. This is the reason why we do not include RWRW in our experiments, because we are interested in the actual extracted samples, for which it is equivalent to RW.

In Forest Fire [14] (FF), one starts by performing a BFS exploration of the graph. However, FF introduces a parameter: the "burning probability". If a node passes the burning probability test, the node is "burned" and the BFS exploration will move on to its neighbors. Figure 1(f) shows an example of this approach.

Finally, we have Neighbor Reservoir Sampling [3, 16] (NRS). NRS starts by building a small core of explored nodes using RW. However, the majority of NRS's budget is spent on a second phase. Suppose that in the first phase the algorithm has sampled $|V'|$ nodes. We select at random a neighbor u directly connected to any of the V' sampled nodes. We also select at random a sampled node v from V'. We add u to V' if and only if: (i) by adding u and removing v, G' still has only a single connected component, and (ii) we extract a uniform random number $\alpha < |V'|/i$, where i is initially set to $|V'|$ and increased by one for every attempt to add a new neighbor u. In practice, $|V'|$ stays constant, but each subsequent attempt to add new us becomes less likely to succeed – because of increasing i. NRS is by far the most computationally expensive method because at each potential node addition it has to verify that G' still has a single connected component. Also, given the diminishing acceptance probability, the last units of budget take a long time before they are spent. For this reason, we cannot include this method in our experiment section, because we cannot run it enough times to guarantee the statistical robustness of our results.

Sampling a network via an API system is not a neutral operation. API systems usually paginate results, meaning that they return only a subset of a node's edges. Moreover, they impose restrictions on how frequently one can query their results. In previous work, we build a framework to test the effectiveness of network sampling techniques [6]. We show that API systems with lower throughput in number of edges returned per second can still return more representative samples, due to the typical broad degree distributions of real world networks.

This paper presents a sampling technique that is inspired by our work on network backboning [5]. Network backboning is the problem in which one has to identify which connections are significant in a dense weighted network. We present the details of our noise corrected sampling strategy in Section 3. However, here we point out that is is reasonable to believe other network backboning techniques could be useful for sampling networks.

The examples we know in the literature are the following. In High Salience Skeleton [10] one re-weights edges according to the number of shortest path trees passing through them and then filters out the least used edges. In the Doubly-Stochastic Transformation [26] one re weights the edges so that both the rows and the columns of the adjacency matrix sum to one, and then filters the network according to these new weights. In Disparity Filter [25], edge weights are tested against a null model of the node emitting them: if an edge weight is significant compared to the total outgoing weights of the node, the edge is kept in the backbone. Finally, an approach similar to ours sees edges as the result of an extraction process from a Polya urn [17].

Any of these methods could provide another potential strategy to select the next edge to explore in a network sampling process. For now, we focus on adapting the noise corrected backboning strategy, leaving exploring the performance of each alternative for future works.

3 METHODS

The algorithm at the basis of noise corrected sampling is simple. The first step is starting from a random seed and collecting all its neighbors. Then, we select an edge at random and we collect the neighbors of the second node we visit. At this point, and for each subsequent step, we use the noise corrected backboning technique to estimate the z-score of each edge. We then select the edges with the maximum z-score and we collect all its neighbors. We stop when we spend the entire crawl budget.

We explain in Section 3.1 how we calculate the z-score and in Section 3.2 the experimental setup, including the budget spending criterion via a simulated API system, and how we evaluate the performance of a sampling strategy.

3.1 Noise-Corrected Sampling

3.1.1 Estimating the z-score. To estimate the z-score of each edge, we need to explain how the noise corrected backboning works. This is explained more in details in our previous work [5]. Here we summarize the description. Note that hereafter we assume that the network is directed and weighted, i.e. that we work with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, N)$, where $\mathcal{V}$ is the set of vertices; $N \subseteq \mathbb{R}^+$ is the set of non-negative real edge weights; and $\mathcal{E}$ is a set of triples (i, j, n) with $i, j \in \mathcal{V}$ and $n \in N$. However, we can assume that all edges are reciprocal and have weight equal to one and the sampling strategy will still work.

If we want to estimate the z-score of an (i, j) edge weight N_{ij} we cannot do so directly, because most real world networks have broad edge weight distributions [25], and thus violate the normality assumption. Thus we transform N_{ij} in two steps. First, we calculate how much it exceeds our expectation:

$$L_{ij} = \frac{\hat{N}_{ij}}{E[N_{ij}]},$$

with

$$E[N_{ij}] = N_{i.} \frac{N_{.j}}{N_{..}},$$

and $N_{i.} = \sum_j N_{ij}$ is the total outgoing weights of i; $N_{.j} = \sum_j N_{ij}$ is the total incoming weights of j; and $N_{..} = \sum_{i,j} N_{ij}$ the total weights in the network.

Then we ensure that L_{ij} becomes a symmetric measure centered on zero:

$$\tilde{L}_{ij} = \frac{L_{ij} - 1}{L_{ij} + 1} = \frac{\kappa N_{ij} - 1}{\kappa N_{ij} + 1} \tag{1}$$

where $\kappa = \frac{1}{E[N_{ij}]}$. We can now estimate the z-score of $\tilde{L}_{ij}$. To do so, we need to compute its variance, which is given by:

$$V\left[\tilde{L}_{ij}\right] = V\left[\frac{\kappa N_{ij} - 1}{\kappa N_{ij} + 1}\right].$$

Applying the delta method, we get:

$$V\left[\tilde{L}_{ij}\right] = V[N_{ij}]\left(\frac{2\left(\kappa + N_{ij}\frac{d\kappa}{dN_{ij}}\right)}{\left(\kappa N_{ij} + 1\right)^2}\right)^2$$

with

$$\frac{d\kappa}{dN_{ij}} = \frac{1}{N_{i.}N_{.j}} - N_{..}\frac{N_{i.} + N_{.j}}{\left(N_{i.}N_{.j}\right)^2}.$$

The only thing left to estimate in this equation is $V[N_{ij}]$. Given that we have interpreted edge weights as the sum of independent unitary interactions, N_{ij} follows a Binomial distribution with variance:

$$V\left[N_{ij}\right] = N_{..}P_{ij}\left(1 - P_{ij}\right) \tag{2}$$

P_{ij} is the unitary interaction probability of nodes i and j, i.e. the probability that, at each new interaction, the weight of the (i, j) edge will increase by one. This is unknown, but can be estimated as the observed frequency with which interactions occur:

$$P_{ij} = \frac{N_{ij}}{N_{..}}.$$

A problem arises when $N_{ij} = 0$. That is, when edge weights are zero for certain node pairs. Given that many real-world networks are sparse, this situation is quite common. For these node pairs, $V\left[N_{ij}\right] = 0$, which would suggest that measurement error is absent in these edges. However, in reality, there is simply too little information to estimate P_{ij} with sufficient precision. This affects not only cases where edge weights are zero, but also where information is sparse, i.e., when focusing on the interactions among nodes of low degree. To improve on this, we estimate P_{ij} in a Bayesian framework. That is:

$$Pr\left[N_{ij} = n_{ij} \middle| N_{..} = n_{..}, P_{ij} = p_{ij}\right] = \binom{n_{..}}{n_{ij}} p_{ij}^{n_{ij}}\left(1 - p_{ij}\right)^{n_{..} - n_{ij}}$$

Using Bayes' law, we get:

$$Pr\left[P_{ij} = p_{ij} \middle| N_{..} = n_{..}, N_{ij} = n_{ij}\right] =$$

$$\frac{Pr\left[N_{ij}=n_{ij}\left|N_{..}=n_{..},P_{ij}=p_{ij}\right.\right]Pr\left[P_{ij}=p_{ij}\left|N_{..}=n_{..}\right.\right]}{\int_0^1 Pr\left[N_{ij}=n_{ij}\left|N_{..}=n_{..},P_{ij}=q_{ij}\right.\right]Pr\left[P_{ij}=q_{ij}\left|N_{..}=n_{..}\right.\right]dq_{ij}} \tag{3}$$

Choosing a $BETA\left[\alpha,\beta\right]$ distribution, the conjugate prior of the Binomial distribution, as a prior for P_{ij}, the posterior distribution is also a $BETA$ distribution. In particular, the posterior distribution of P_{ij} becomes:

$$P_{ij}\sim BETA\left[n_{ij}+\alpha,n_{..}-n_{ij}+\beta\right]. \tag{4}$$

We still have to choose values for α and β that would give plausible prior expectations for the mean and variance of P_{ij}. To do so, assume that the total weight of i and j is given. In other words, think of edge weights as arising from a process in which, each time node i increases its total weight by one, it draws a node j at random from the pool of possible nodes. That is, edge weight generation follows a hypergeometric distribution. This gives the following prior means and variances for P_{ij}:

$$E\left[P_{ij}\right]=E\left[\frac{N_{ij}}{N_{..}}\right]=\frac{1}{N_{..}}E\left[N_{ij}\right]:=\frac{1}{N_{..}}\frac{N_{i.}N_{.j}}{N_{..}}$$

$$V\left[P_{ij}\right]=\frac{1}{N_{..}^2}V\left[N_{ij}\right]:=\frac{1}{N_{..}^2}\frac{N_{i.}N_{.j}\left(N_{..}-N_{i.}\right)\left(N_{..}-N_{.j}\right)}{N_{..}^2\left(N_{..}-1\right)}.$$

The := equality indicates where we make our assumptions. From the $BETA\left[\alpha,\beta\right]$ distribution, we get:

$$E\left[p_{ij}\right]=\mu=\frac{\alpha}{\alpha+\beta} \tag{5}$$

$$V\left[p_{ij}\right]=\sigma^2=\frac{\alpha\beta}{\left(\alpha+\beta\right)^2\left(\alpha+\beta+1\right)} \tag{6}$$

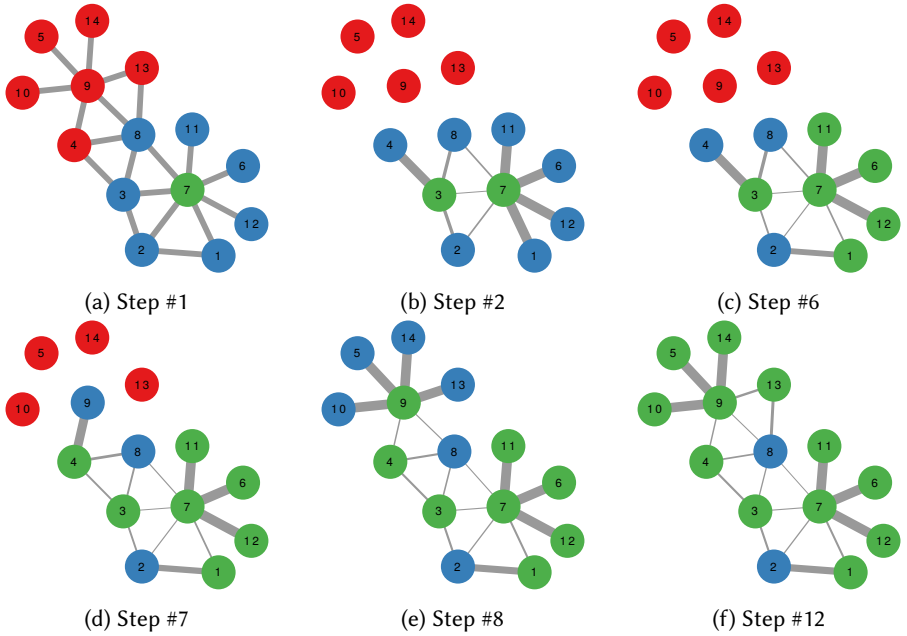
Solving for α and β, we get:

$$\alpha=\frac{\mu^2}{\sigma^2}\left(1-\mu\right)-\mu \tag{7}$$

$$\beta=\mu\left(\frac{\left(1-\mu\right)^2}{\sigma^2}+1\right)-1 \tag{8}$$

Eqs. 4, 5, 6, 7 and 8 now define a posterior expectation for P_{ij} for each node pair. We can use this posterior expectation of P_{ij} to recalculate variances of edge weights in Eq. 2. Because the posterior expectation of P_{ij} is always strictly larger than zero, variance estimates do not degenerate.

Since now we have an estimation of expected edge weight and the variance of such expectation, we can combine them with the observed edge weight to derive the z score.



(a) Step #1 (b) Step #2 (c) Step #6

(d) Step #7 (e) Step #8 (f) Step #12

Fig. 2. Exploring an unweighted undirected graph using the noise corrected strategy. Node color determines the status of the node: green = explored, blue = discovered, red = unknown. The edge thickness represents its z-score (with the exception of Figure 2(a), where we show the input unweighted network).

3.1.2 Adaptation to Sampling. In the previous section we described how to calculate the z-scores for each edge in the currently sampled network. Here we show how to apply this backboning technique to perform a topological sample of a network. The basic algorithm is simple. If G' is the current sample, G' contains V' nodes. Some of those nodes were explored, while others were discovered, i.e. they were not explored, but at least one of their neighbors was. We perform the noise corrected backbone on G' and we explore the discovered node that is attached to the edge with the highest z-score.

We show an example of how this strategy works. We start from the toy network in Figure 2(a). We specifically choose to use an unweighted network as our example, to prove that edge weights are not necessary to identify the next node to explore. Our example would work also with weighted and directed edges.

At the first step in Figure 2(a), we pick a random seed node, in this case node 7, and we explore it. This means that our sample after the first step is a star centered on node 7. In this situation, all edges have the same z-score, and thus we follow one at random, in this example to node 3 (Figure 2(b)). Now that we discovered that nodes 7 and 3 have a common neighbor (node 2), the (7, 2) edge becomes the least significant edge in the structure and it is thus ignored, in favor of any neighbor of 7 (or 3) that is not in common with already explored nodes (Figure 2(c)). In Figure 2(c), we perform multiple steps at once, because for each step we always select one of the neighbors of 7 without common neighbors with the already explored nodes.

According to the same logic, once we explored all neighbors of 7 without common neighbors with already explored nodes, the only choice is to explore a neighbor of any explored node which has no common explored nodes. This is node 4 (Figure 2(d)). We continue with the same logic exploring node 9 (Figure 2(e)) which leads to a full graph exploration: all nodes are either explored

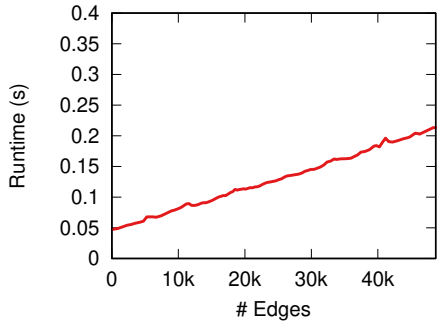


Fig. 3. The amount of time it takes to calculate the noise corrected backbone (y axis) for a network with a given number of edges (x axis).

or discovered. If we were to keep going, we would explore the neighborhood of node 9 (Figure 2(f)) before turning to the remaining nodes 2 and 8, in that order.

3.1.3 Scalability. Sampling networks via noise corrected backboning cannot be as time efficient as alternative methods. Sampling via backboning means to check, for every sampling step, all edges in the sample at least once. This cannot beat fully local methods like random walks or DFS, because at each sampling step their complexity is $O(1)$.

However, we argue that this is not a major issue. There are two reasons. First, noise corrected backboning has a linear complexity in terms of number of edges, $O(|E_s|)$. This means that even large samples can be efficiently analyzed. We prove this point in Figure 3. The figure shows that the relation between the number of edges in the sample and the runtime is linear.

The figure also provides support for the second reason why scalability is not an issue: even for the largest samples, the runtime is negligible, in the neighborhood of one or two tenths of a second. When sampling online social media, the computation happening locally to determine the next sampling step is not usually the bottleneck. The sum of all delays – network latency, the time required for the server to retrieve the information, and the time it takes to transfer the information – is likely higher than the backboning computation time. For instance, the average ping time we measured for Instagram's APIs was $\sim$ 27ms, and this is considering that no computation nor data transfer was requested to the server.

Moreover, as we see in the next section, social media platform usually impose temporal rate limits between queries. This means that a sampler needs to wait one or more seconds between requests. The sampler can perform its local computations during the waiting period, rendering irrelevant the time cost of running the noise corrected backboning.

3.2 Evaluation Criteria

When evaluating a sampling method one has to determine how good a sample is given the budget needed to obtain it. In this paper we assume that the budget is expressed in seconds. In other words, we assume that the sampler runs for a given amount of time, after which it terminates. This simulates the data gathering phase of a research project.

This paper specifically focuses on the problem of sampling networks via the API system of an online social media. Thus we need to use a realistic evaluation framework, targeted to this specific scenario. We use the benchmark system we developed as previous work [6], which allows us to simulate calls to an API system. This, in turn, allows us to estimate the time cost of sampling a

API	p_s	t_s	p_s/t_s
SPHL	10	10	1
SPLL	5	2	2.5
LPHL	100	20	5
LPLL	40	4	10

Table 1. The characteristic of each API system: p_s = number of edges per request; t_s = time lag between requests; p_s/t_s = API throughput, in edges per second.

node, which we can use to spend our time budget. We have three main dimensions over which we evaluate methods: by social media API policy, by target analysis, and by network topology.

3.2.1 API Policies. Sampling via an API system introduces restrictions and peculiarities that are rarely considered in the network sampling literature. The first peculiarity of online social media is their pagination policy. Pagination means that, even if the connections of a single node are potentially available through a single call, the actual number of calls needed to gather them is defined by the number of connections of the node and by the size of the page allowed by the social media platform. In the case of a node v member of a social network s, the number of calls required to collect all its connections will be:

$$calls(v, s) = \left\lceil \frac{k_v}{p_s} \right\rceil,$$

where k_v is v's degree, and p_s is the number of edges returned with each call as defined in social media's s pagination policy, which we call "page size".

The second peculiarity of online social media is rate limits. APIs regulate the number of calls per unit of time that a sampler is allowed to make. By restricting how many calls to the API are possible per second, rate limits define the actual budget, expressed in time, needed to collect the list of edges incident to a node. Using t_s as the number of seconds social media s forces the user to wait before submitting another query to the API system, the final cost – in number of seconds – of crawling node v is:

$$cost(v, s) = t_s \times calls(v, s).$$

This cost function seems relatively innocuous, but our previous work [6] shows that it has profound repercussions on the performance of network samplers. One could be tempted to characterize social media APIs by their edge throughput: how many edges they return per unit of time. Yet, in some cases APIs with lower throughput can allow a more efficient sampling of a network. This is because $cost(v, s)$ is linked to k_v, the degree of node v: in real world networks, most nodes have low k_v. Thus, a policy with low p_s and t_s can explore more nodes per unit of time than a policy with high p_s and t_s, even if the latter returns more edges per second. This is provided that most nodes have $k_v \leq p_s$, which is true for most realistic scenarios.

For this paper, we test the network sampling models on four archetypal API systems, defined across the page size and query latency dimensions. The page size can be either large (LP = Large Page) or small (SP = Small Page); the wait time between queries can either by high (HL = High Latency) or low (LL = Low Latency). Thus, the four archetypal API systems are: SPHL, SPLL, LPHL, and LPLL. Table 1 shows the characteristics of each API system.

3.2.2 Analyses. Some network sampling methods are developed with a specific analysis in mind, meaning that they attempt to preserve a key characteristics of the network in the sample. As a

consequence, the type of analysis one wants to perform on the network is important in determining which sampling method they choose. Here, we target a set of six network properties as a possible analyses one might want to perform.

Degree distribution. If we think that the network at large has a power law degree distribution, we might want the sample to have the same distribution exponent. Thus, for this analysis, we calculate the Kolmogorov-Smirnov distance [18] between the degree distributions of the original network and of the sample.

Node centrality. We want to make sure that the nodes that are central/peripheral in the original network are also central/peripheral in the sample. The way we estimate the sample quality is by calculating the Spearman rank correlation between the degrees of the nodes in the original network and in the sample.

Assortativity / Disassortativity. Nodes have attributes, which they usually correlate across the network [19]. An assortative node attribute means that nodes with the same value tend to connect to each other, a disassortative attribute means that nodes tend to connect with neighbors with unlike value. Here, we calculate the absolute difference between the assortativity of the attribute in the original network and in the sample. This is a double test, because we generate two attributes: one assortative and one disassortative.

Community strength. The original network might or might not have communities. A way to determine this is by trying to partition it into communities and then calculate the coverage of the partition: the ratio of the number of intra-community edges to the total number of edges in the graph [8]. In this test, a better sample will have a lower absolute coverage difference with the original network.

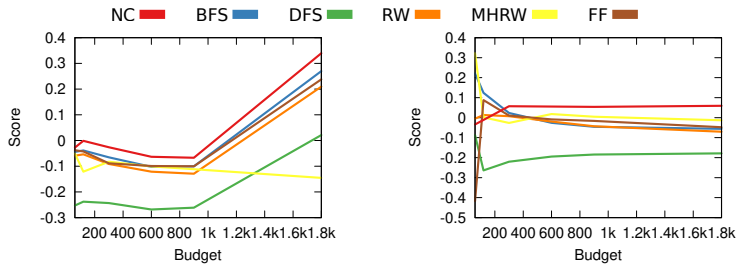
Community similarity. Besides community strength, we want also to group nodes in the sample in the same communities as in the original network. Thus groups should contain the same nodes. We can estimate the similarity of two partitions by calculating their mutual information [29]. In this scenario, a good sample is one that has a high normalized mutual information with the original graph.

3.2.3 Topologies. Not all social networks have the same topology, and not all topologies can be efficiently explored with the same strategy. Thus we test network sampling methods on a set of topologies. We range from least to most realistic by examining: Erdős-Rényi random graphs, preferential attachment, powercluster networks, and LFR benchmarks.

An Erdős-Rényi random graph is a graph in which any pair of nodes is connected with a fixed probability [7]. This is the least realistic type of network, which satisfies some key real-world graph properties such as low average degree and small world – i.e. the longest shortest path grows logarithmically with the network size in number of nodes. It has unrealistic degree distribution, clustering coefficient, and it lacks communities, a commonly observed network property.

The preferential attachment model [4] adds a realistic power law degree distribution to the Erdős-Rényi random graph. It does so by growing a network one node at a time from a seed of nodes. Each new node in the network connects to k already existing nodes, with a probability proportional to their current degree. The powercluster model [11] works exactly as the preferntial attachment, but at each new node it also closes a possible triangle in the network at random. Thus, it not only has a realistic power law degree distribution, but it also has a realistic clustering coefficient.

Finally, the LFR benchmark [13] is a network with all the desirable characteristic of a real world network. It plants communities in the structure and adds edges preferentially among nodes in the same communities.



(a) SPLL (b) LPHL

Fig. 4. The score (y axis) of each sampling method (color) for increasing budget levels (x axis). Result averaged across all analyses and topologies. (Higher is better)

4 EXPERIMENTS

Summing up the Methods section, we are performing tests across: (i) four different API systems (SPHL, SPLL, LPHL, and LPLL); (ii) six network analyses; and (iii) four network topologies. This is a total of $4 \times 6 \times 4 = 96$ tests. Rather than reporting on all of the tests, we aggregate them across one dimension we study. We do so by standardizing the result of each analysis so that its average is zero and its standard deviation is equal to one. Then we multiply by minus one those tests – like mean absolute error – for which "lower is better". Then, we can average score results across different tests into a single score for which "higher is better".

Note that the aim of this section is not to show that the noise corrected sampling is better than the alternatives in all scenarios. No method is. Rather, the objective of this section is to find out *in which* scenarios the noise corrected sampling is better and in which scenarios it is worse. This is the reason why, for each test, we only show two cases: the one in which the noise corrected sampling performed best relatively to the alternatives and the one in which it performed relatively worst.

Here we test the noise corrected sampling method against: DFS, BFS, RW, MHRW, and FF.

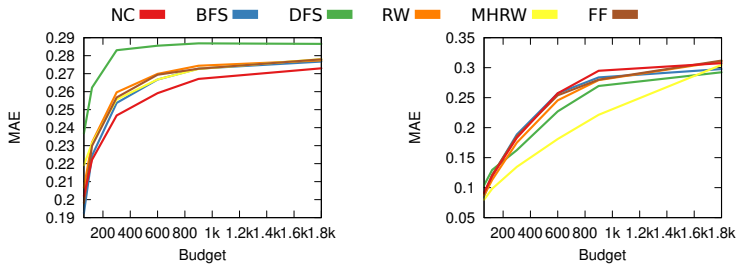
4.1 By API Features

The first dimension we analyze is the features of the API system. The question we ask here is: under which combination of page size and query latency does noise corrected work better than the alternatives? To answer this question, we calculate the average rank of the noise corrected sample relatively to all other tested methods, across budget levels. An average rank of 1 means that the method was always the best across all budget levels.

Figure 4 shows the score evolution for increasing budgets. Since these are aggregate scores, the best performing methods are the ones scoring highest. According to the average rank method, the noise corrected sampling works best for the SPLL API (average rank of 1 – Figure 4(a)) and worst for LPHL (average rank of 2.16 – Figure 4(b)). In general noise corrected prefers low latencies over high ones, and small pages over large ones. We can see that in SPLL, noise corrected is best at all budget levels, while in LPHL it underperforms for low budgets and it is best for high budgets.

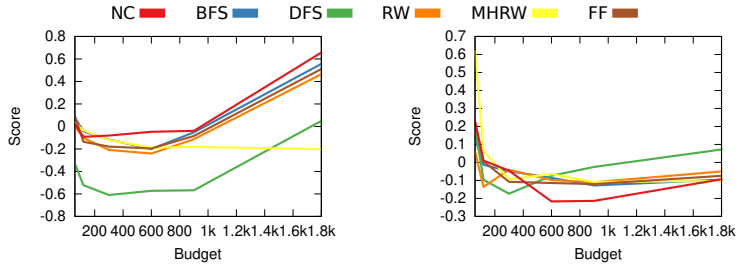
4.2 By Analysis

Here, we repeat the analysis from the previous section, but aggregating results not across API systems, but across all network analyses. Since in this section we show the mean absolute error as a measure of quality, the evaluation criterion is "lower is better".



(a) Disassortativity (b) Community strength

Fig. 5. The score (y axis) of each sampling method (color) for increasing budget levels (x axis). Result averaged across all APIs and topologies. (Lower is better)



(a) PC (b) ER

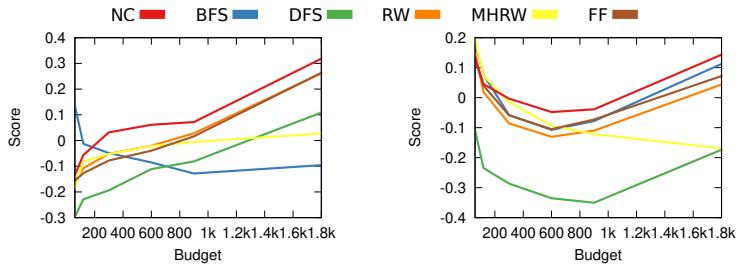
Fig. 6. The score (y axis) of each sampling method (color) for increasing budget levels (x axis). Result averaged across all analyses and APIs. (Higher is better)

Figure 5 shows the score evolution for increasing budgets. According to the average rank method, the noise corrected sampling works best to reconstruct the network's disassortative attribute (average rank of 1.33 – Figure 5(a)) and worst to estimate the strength of the network's communities (average rank of 4.5 – Figure 5(b)). In general noise corrected allows the precise estimation of node-level properties (disassortativity, assortativity, and degree all have average rank < 3) while it performs poorly for meso level analyses such as the ones involving communities (average rank > 3).

4.3 By Topology

In the third aggregation dimension, we look at the performance of each sampling method on average across a network topology. We use the aggregated score performance, thus we are following the logic of "higher is better".

Figure 6 shows the score evolution for increasing budgets. The network topology on which noise corrected sampling works worst is the Erdos-Renyi random graph (average rank of 3.6 – Figure 6(b)). This is a comforting result, because this is the least realistic network topology we tested, and it is thus unlikely to be a fair representation of the types of topologies one could find in the real world. The noise corrected sampling obtains the best performance for the powercluster network topology (average rank of 1.8 – Figure 6(a)), which is the second most realistic topology we test.



(a) Weighted (b) Unweighted

Fig. 7. The score (y axis) of each sampling method (color) for increasing budget levels (x axis). Result averaged across all topologies, analyses, and APIs. (Higher is better)

Method	AVG Rank
MHRW	2.89
NC	2.91
BFS	3.04
RW	3.05
FF	3.08
DFS	3.21

Table 2. The average rank of each sampling method, across all budget levels, topologies, analyses, and APIs. (Lower is better)

4.4 Global ranks & Use Cases

Finally, we aggregate the performance of the network sampling methods across all dimensions, distinguishing only if we are looking at a weighted or at an unweighted network. Again, for this section, "higher is better".

Figure 7 shows the overall performance. We can see that the noise corrected sampling has a clear edge for high budget situations, while it suffers lower budgets.

Finally, Table 2 sums up the overall rank of each method across all our dimensions, including budget levels. As we can see, the highest ranking method is the Metropolis-Hastings Random Walk sampler, with our proposed method ranking in second. However, the difference between these two methods is small, much smaller than the gap between the second and third ranked method. Moreover, as we saw, the NC sampling method outperforms for high budgets. We could test only a handful of distinct high budgets, due to the required time to run enough simulations to guarantee statistical significance. We find this an appropriate approach, because we should not overweight high budgets, as it is less likely a researcher will have enough time to gather a large sample from an online social media platform.

We conclude this section by summing up the most promising use cases for the noise corrected network sampling method. The NC method should be used when working with low query latency platforms, when the estimation of node-level attributes is important, and for networks with realistic topologies (power law degree distributions and high clustering). If the researcher is not sure about the type of analyses they will run, nor of the underlying topological characteristics of the network, the NC sampling is still a good general choice. It should be preferred over MHRW especially if there is the possibility of extracting a relatively larger sample.

5 CONCLUSION

In this paper, we proposed a new approach to the topological network sampling problem. In topological network sampling, one starts from a seed node and tries to identify which connections to follow, with the aim of creating a small sample out of a larger network. The sample should be built such that the analyses on it will support the same conclusions as if they were to be ran on the whole structure. We use a noise corrected backboning technique to identify the connections with the highest information gain, which should be the ones to be followed to sample a new node. We tested our approach over a number of dimensions: different underlying network topologies, different network analyses, and different API systems – the latter dimension is necessary due to the common scenario of crawling network data from social media platforms. We identified the scenarios in which our noise corrected sampling works best: analysis of node attributes, realistic topologies with power law degree distributions and high clustering, and API systems allowing for fast querying. Overall, our method is the second best performing, with a clear advantage in case of high crawl budgets.

Our paper paves the way for further improvements in the field of network sampling. First, to our knowledge, this is the first time that the network backboning and network sampling problems are shown to be related. We could investigate the performance of other backboning methods in identifying the next node to be sampled. Second, we only directly applied the noise corrected backboning to the sampling problem without adapting its edge-centric approach. The sampling problem is node-centric, thus an adaptation of noise corrected backbone to estimate the node information gain, rather than the edge information gain, could prove promising. Finally, this paper relies exclusively on random simulations. Testing these methods in real world data is challenging, as we would need the entire network to evaluate performance, but it could provide further important information about the sampling methods' performances.

REFERENCES

[1] Nesreen Ahmed, Jennifer Neville, and Ramana Rao Kompella. 2011. Network sampling via edge-based node selection with graph induction. (2011).

[2] Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. 2012. Space-efficient sampling from social activity streams. In *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: algorithms, systems, programming models and applications*. ACM, 53–60.

[3] Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. 2014. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8, 2 (2014), 7.

[4] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[5] Michele Coscia and Frank MH Neffke. 2017. Network backboning with noisy data. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 425–436.

[6] Michele Coscia and Luca Rossi. 2018. Benchmarking API costs of network sampling strategies. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 663–672.

[7] P Erdős and A Rényi. 1959. On random graphs. *Publicationes Mathematicae Debrecen* 6 (1959), 290–297.

[8] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[9] Leo A Goodman. 1961. Snowball sampling. *The annals of mathematical statistics* (1961), 148–170.

[10] Daniel Grady, Christian Thiemann, and Dirk Brockmann. 2012. Robust classification of salient links in complex networks. *Nature communications* 3 (2012), 864.

[11] Petter Holme and Beom Jun Kim. 2002. Growing scale-free networks with tunable clustering. *Physical review E* 65, 2 (2002), 026107.

[12] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. 2008. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*. ACM, 19–24.

[13] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008), 046110.

[14] Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 631–636.

[15] László Lovász. 1993. Random walks on graphs. *Combinatorics, Paul erdos is eighty* 2, 1-46 (1993), 4.
[16] Xuesong Lu and Stéphane Bressan. 2012. Sampling connected induced subgraphs uniformly at random. In *International Conference on Scientific and Statistical Database Management*. Springer, 195–212.
[17] Riccardo Marcaccioli and Giacomo Livan. 2019. A Pólya urn approach to information filtering in complex networks. *Nature communications* 10, 1 (2019), 1–10.
[18] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.
[19] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.
[20] Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 29–42.
[21] Edward F Moore. 1959. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*. 285–292.
[22] Amir Hassan Rasti, Mojtaba Torkjazi, Reza Rejaie, Nick Duffield, Walter Willinger, and Daniel Stutzbach. 2009. Respondent-driven sampling for characterizing unstructured overlays. In *INFOCOM 2009, IEEE*. IEEE, 2701–2705.
[23] Amir H Rasti, Mojtaba Torkjazi, Reza Rejaie, D Stutzbach, N Duffield, and W Willinger. 2008. Evaluating sampling techniques for large dynamic graphs. *Univ. Oregon, Tech. Rep. CIS-TR-08* 1 (2008).
[24] Matthew J Salganik and Douglas D Heckathorn. 2004. Sampling and estimation in hidden populations using respondent-driven sampling. *Sociological methodology* 34, 1 (2004), 193–240.
[25] M Ángeles Serrano, Marián Boguná, and Alessandro Vespignani. 2009. Extracting the multiscale backbone of complex weighted networks. *Proceedings of the national academy of sciences* 106, 16 (2009), 6483–6488.
[26] Paul B Slater. 2009. A two-stage algorithm for extracting the multiscale backbone of complex weighted networks. *Proceedings of the National Academy of Sciences* 106, 26 (2009), E66–E66.
[27] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, and Walter Willinger. 2009. On unbiased sampling for unstructured peer-to-peer networks. *IEEE/ACM Transactions on Networking (TON)* 17, 2 (2009), 377–390.
[28] Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1, 2 (1972), 146–160.
[29] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* 11, Oct (2010), 2837–2854.