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 a process for extracting a subset of nodes and edges from a network, such that analyses on the sample provide results and conclusions comparable to the ones they would return if run 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 sample in all possible scenarios, thus in our results section we show the cases in which our method performs best and the cases in which it performs worst. Overall, the noise corrected network sampling performs well: it has the best rank average among the tested methods across a wide range of applications.

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), 22 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, their API platforms grant access to their data, and they severely limit the accessible throughput. For this reason, a network scientist needs to select wisely the subset of nodes and edges they access. In other words, they need to perform topological network sampling.

Topological network sampling is a process for extracting 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 of the discovered nodes to explore next. The aim of network sampling is to reconstruct a sample able to produce results and

[1]https://www.theverge.com/2020/1/29/21114130/facebook-q4-2019-earnings-instagram-whatsapp-messenger, date of access: April 1st, 2020.

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

conclusions comparable to the larger whole network. For this reason, many different methods have been defined.

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

Given the importance of sampling and the enormous size of online social networks, developing new and better sampling procedures is an important issue. Current topological sampling methods are limited in the amount of information from the collected sample they use to make their next move. Most sampling methods use almost no information at all: e.g. Breadth and Depth First explorations only check whether the next selected node had been already explored or not. Other methods only use limited local information: e.g. the popular Metropolis-Hastings Random Walk only compares the degree of the next node to explore and of the currently explored node. Here we propose a method that analyzes the entire currently collected sample to inform the decision about the next node to explore.

To do so, we draw on our previous work on noise-corrected network backboning [6]. Our previous work deals exclusively with the network backboning problem: to identify the most significant connections in a dense weighted network to sparsify it for network analysis. 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.

In this paper, we expand our previous work by applying it to the network sampling problem. The difference is that, in network backboning, we filter edges out of a complete observation. In sampling, we are observing an incomplete portion of the network. The information gain from our method guides us towards the edges we should follow, because they are the ones which would allow us to discover more about the whole network. Thus we provide additional contributions such as a new application scenario and new experiments.

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 previous work [7], 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 that always provides the best sample. Instead, when presenting a new sampling strategy, we need 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 experiments section, we show the cases in which the noise corrected sampling performed 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. When performance is averaged across all analyses, network topologies and API systems, it ranks best among the tested alternatives. Its strengths are mostly due to good performance at high crawling budgets. If the researcher has the ability of collecting a reasonably large sample, our proposed approach is expected to return the best samples.

We make our code available for the usage of our sampling method[2]. The provided archive contains also the data and the code on which our results are based are available, for replicability purposes.

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 split in the literature is between static sampling and online sampling of an evolving structure [31]. For this paper, we focus on static network sampling. The reason is that social media are vast, and thus their topological characteristics change slowly, on much larger time scales than most crawling processes.

The second main split is between topological and non-topological sampling. In this paper we focus on topological sampling because non-topological sampling is not compatible with API systems, since it requires to either have some knowledge of the global network structure or it requires accessing nodes in ways that are not supported by API systems – e.g. uniformly at random.

To understand this point, consider one example of non-topological sampling: induced sampling. 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 sampling methods are Partially and Totally Induced Edge Samples (TIES and PIES [1, 2]), while node sample methods include fully uniform, degree (RDN) and PageRank weighted samples (RPN) – see [17] for a discussion of these methods. However, to perform and edge induces sample we must be able to access edges individually, which is something one cannot do with social media APIs, as one can only query nodes. On the other hand, one cannot extract nodes at random because this would imply knowing their IDs, which are usually unpredictable random alphanumeric strings rather than progressive numbers.

Other sampling methods that require information not available via API systems are ones based on knowing in advance the partition of the original network in communities [8]; or exploring the graph via learning automata [32], which require a pre-processing phase to estimate the importance of nodes.

In topological sampling, which is the type of method we explore in this paper, 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 it can be probabilistic – as in random walks [15, 18, 30, 34, 37] and probabilistic modifications of the deterministic approaches such as forest fire [17] or Neighbor Reservoir Sampling [3, 19]. We need to understand these methods in detail, because they represent the main comparison with our new proposed network sampling methodology.

In Breadth-First Search (BFS) [24, 25], 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) [38] is similar to BFS, but uses a Last-In, First-Out (LIFO) queue instead of a FIFO (see Figure 1(b)), exploring children before siblings, rather than the other way around.

Snowball sampling [11] is almost identical to BFS, but 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 – one can set n to be so high that it does not

[2]http://www.michelecoscia.com/?page_id=1788

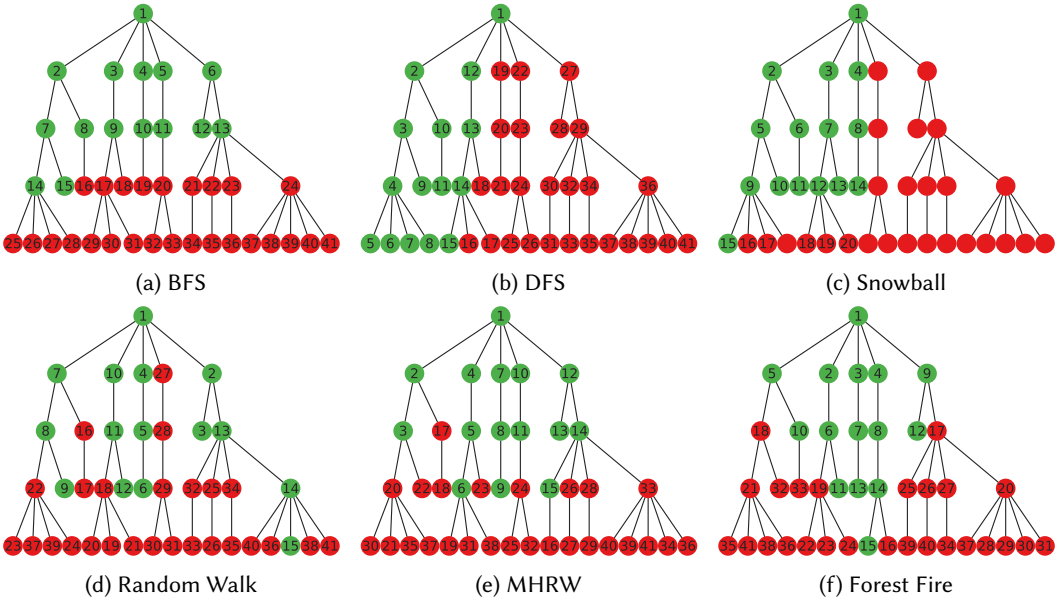


Fig. 1. Examples of how different network sampling strategies explore a given network. Each node is labeled with the order in which it is explored. The node color shows whether the node was sampled (green) or not (red), assuming a budget of 15 units and a constant cost of 1 unit per node. Snowball example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

affect the sampling process any more –, and that is the reason why we do not include it in our experiment section.

Some authors pointed out the benefits of biased network samples [39]. For instance, the highest degree nodes are the most important nodes in a network and thus should be sampled, at the expense of under-sampling the periphery of the network. This philosophy is implemented by the Sample Edge Counts (SEC) method [20]. SEC ranks the neighbors of all the sampled nodes according to their degree and then explores the neighbor with the highest edge count towards already explored nodes.

When sampling a network using random walks (RW), the procedure starts from v and chooses one of its neighbors at random [17, 18]. 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.

A simple extension of RW sampling is m-dependent Random Walk (MRW) [33]. This involves performing m random walks at once. The random walkers are not independent: we choose which of the m random walker to make progress proportionally to the degree of the nodes they are currently visiting. Thus, if there are three random walkers and they are currently on nodes with degrees 3, 2, and 1, we will continue from the first random walker with $3/(3 + 2 + 1) = 0.5$ probability.

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. The Metropolis-Hastings correction for random walk sampling (MHRW) remedies this problem [15, 37]. The network is explored via random walks. However, after selecting

v's neighbor u at random with probability $1/k_v$, the move is accepted with probability k_v/k_u (which is capped to one if $k_v > k_u$), 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 in the random walk family is the re-weighted correction [30, 34] (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, we use the sample to estimate the statistical properties of the original graph. The downside of RWRW is that it can return a correct estimation of the original graph's properties, but not a sample that can be then used as input for any arbitrary algorithm: the extracted sample is equivalent to the one extracted with a vanilla RW. This is why we do not include RWRW in our experiments: we are interested in the actual extracted samples, for which it is equivalent to RW. A similar node attribute reconstruction approach [26] is discarded for the same reason.

In Forest Fire [17] (FF), one starts by performing a BFS exploration of the graph. However, FF introduces a new 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, 19] (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. In the second phase, nodes in the sample are replaced by neighbors of sampled nodes randomly, with a decreasing probability and making sure that the sample retains a single connected component. NRS is by far the most computationally expensive method because at each potential node addition it has to verify that the sample 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 directly is a different operation than sampling it via an API system. 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 them. In previous work, we build a framework to test the effectiveness of network sampling techniques [7]. 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 new sampling technique that is inspired by our work on network backboning [6]. Network backboning is the search for significant connections in a dense weighted network. We present the details of our noise-corrected sampling strategy in Section 3.

Of course, other network backboning techniques could also provide us with strategies to select the next edge to explore in a network sampling process. In High Salience Skeleton [12], for example, 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 [36] one reweights the edges so that each row and column of the adjacency matrix sums to one, and then filters the network according to these new weights. In Disparity Filter [35], 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 [21]. In Section 4 we motivate our choice of focusing on the noise-corrected technique, due to its better performance in reconstructing backbones.

A sister problem of network backboning is error correction [28, 29]. Given a complete network dataset, the objective is to identify which edges are likely to be omitted and which present edges

are likely to be spurious. Just like our noise-corrected backboning, many of these approaches assign to edges an estimation of how much information it gives about the full structure. Thus, some of these methods could also be used for sampling large complex networks. We leave exploring the performance of each alternative for future work.

3 METHODS

The algorithm at the basis of noise-corrected sampling is simple. We initialize the system by collecting all the edges incident to the seed node and one of its neighbors at random. Then, we use the noise corrected backboning technique to estimate the z-score of each edge. We then follow the edge with the maximum z-score and we collect all the neighbors of the node connected to it. If two edges have the same z-score, we break the ties randomly. We stop when we spent the entire crawl budget.

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

3.1 Noise-Corrected Sampling

The aim of a sampling method is to collect as little data as possible from the original network while gathering as much information as possible about it. The noise-corrected backbone estimates, for each edge, how much information it gives us about the network. Therefore, it is natural to collect a sample by recursively following the maximum-information edges. The way noise-corrected backboning estimates the amount of information carried by an edge is by estimating its z-score, namely how much surprising its existence is over null expectation. For this reason, we start by explaining how we make this estimation.

Note that what follows in this subsection is not an original contribution of this paper. The noise corrected backbone was originally developed in supplementary materials Section D4.2 of [27] and expanded in our previous work [6]. We refer to these papers for the full details about its rationale.

3.1.1 Estimating the z-score. To clarify what the z-score of an edge would be, we need to summarize how noise-corrected backboning works. This is explained in more detail in our previous work [6]. 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 [35], and thus violate the normality assumption. Thus we transform N_{ij} in two steps. First, we calculate by 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 sum of outgoing weights of i; $N_{.j} = \sum_j N_{ij}$ is the sum of incoming weights of j; and $N_{..} = \sum_{i,j} N_{ij}$ the sum of all 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 we need to estimate in this equation is $V[N_{ij}]$. In this paper, we interpret edge weights as the result of a series of discrete interactions between nodes. Each time two nodes interact with each other, we add one to their edge weight. Thus, N_{ij} is simply the number of times nodes i and j interact. Following this assumption, N_{ij} distributes as a binomial variable 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 the next interaction in the network, 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 edge weights are zero for certain node pairs: $N_{ij} = 0$. 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:

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

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

Choosing a $B[\alpha, \beta]$ beta 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:

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

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 $B[\alpha, \beta]$ distribution, we get:

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

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

Solving for α and β, we get:

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

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

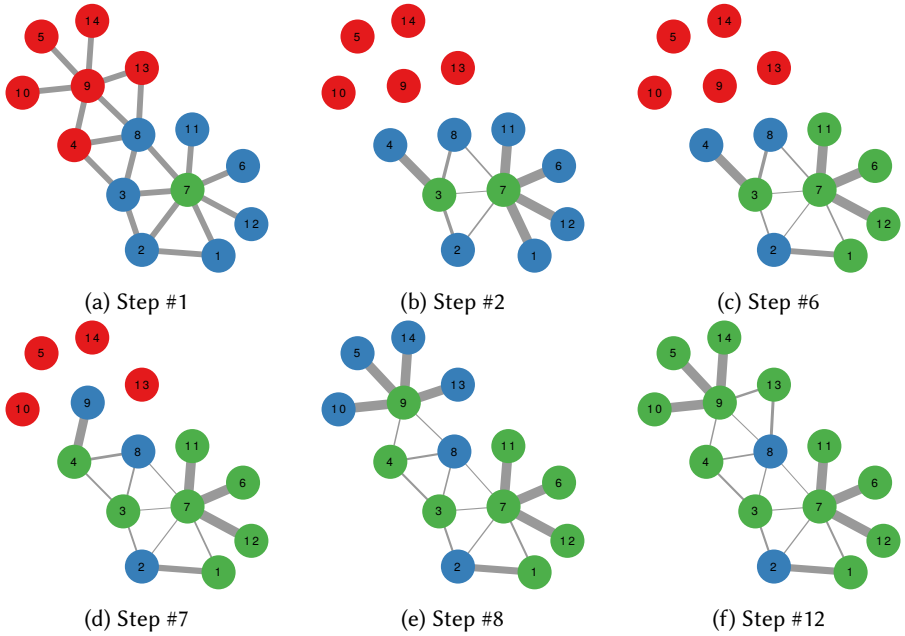
Eqs. 4, 5, 6, 7 and 8 now define a posterior expectation for P_{ij} for each node pair. We can use this posterior expectation of P_{ij} to recalculate variances of edge weights in Eq. 2. Because the posterior expectation of P_{ij} is always strictly larger than zero, variance estimates do not degenerate.

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:

$$z_{ij} = \frac{\tilde{L}_{ij}}{V\left[\tilde{L}_{ij}\right]}.$$

Recall that $\tilde{L}_{ij}$ is centered on zero and already contains a comparison with expectation, thus it can be negative if the observed edge weight N_{ij} is lower than expectation ($E\left[N_{ij}\right]$).

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.



(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).

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 show that edge weights are not necessary. Our example would work also with weighted and directed edges.

During the initialization phase 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 common neighbors (nodes 2 and 8), the interested edges become the least significant edges in the structure and they are 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, since each step follows the same logic: we always explore one of the neighbors of 7. Specifically, we choose the neighbor that has no common neighbors with the already explored nodes – thus skipping nodes 2 and 8.

In other words, we preferentially explore discovered nodes with degree equal to one. Once we run out of this type of node around 7, we discover another one of such nodes: 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 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 methods like random walks or DFS, because at each

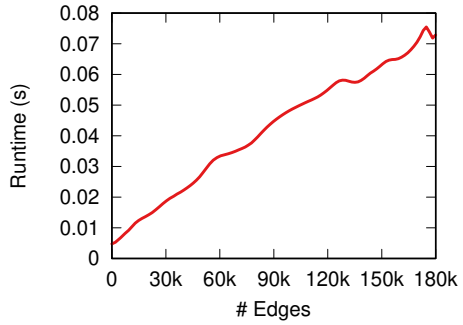


Fig. 3. The amount of time it takes to calculate the noise corrected backbone (y axis) for a network with a given number of edges (x axis).

sampling step their complexity is $O(1)$, given that they exclusively use local information about the currently explored nodes, rather than information coming from the entire current sample.

However, we argue that this is not a major issue, for 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 illustrate this point in Figure 3. The figure shows that the relationship 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, less than one tenth 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 from the server.

Moreover, as we see in the next section, social media platforms 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 sampler runs for a given amount of time, after which it terminates. This simulates the data gathering phase of a research project. Thus, the unit we use to measure a budget is the number of available seconds we have to perform the crawl.

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 we developed in previous work [7], which allows us to simulate calls to an API system. This, in turn, allows us to estimate the time cost of sampling a 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

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.

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 medium 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 [7] 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 characteristic 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 to 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 [22] 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 [23]. 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 an 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 binary attributes: one assortative and one disassortative.

Community similarity. We want to group sampled nodes into 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 [41]. In this scenario, a good sample is one that has a high normalized mutual information with the original graph. This test is only performed for networks that actually have communities, i.e. the LFR benchmarks we present in the next section.

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 synthetic topologies and real world networks. For the synthetic networks, we range from least to most realistic by examining: Erdős-Rényi random graphs (both unipartite and bipartite), 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 [9]. 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 an unrealistic degree distribution and clustering coefficient, and it lacks communities, a commonly observed network property. We also make a bipartite version of it, splitting nodes in two classes – V_1 and V_2 – and then extracting random links running exclusively between nodes of unlike classes.

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 [14] works exactly like the preferential 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 [16] 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.

All synthetic networks have $10k$ nodes and approximately $50k$ edges. While this is significantly smaller than real world networks, we decide to strike a compromise between size and statistical robustness: we need to perform the experiments multiple times from different starting seeds to make sure that the results are robust. This is not possible for larger networks.

We also test on two real world network. The first is a social media network based with geolocation metadata [5]. The weight of the edge is determined by the number of check-ins the two friends have made in the same location plus one – to include friends who never checked in the same location. This network has $56k$ nodes and $213k$ edges. The second network is bipartite, connecting

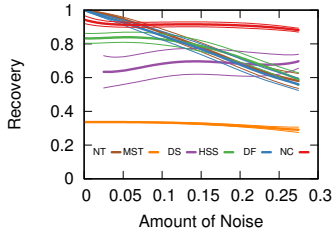


Fig. 4. Recovery of the true backbone of synthetic Barabasi-Albert networks.

pathogens to the locations they were sampled on [42]. The weight of the edge is the number of times the connection was observed in a sample or in a publication. This network has $74k$ nodes and $157k$ edges.

4 MOTIVATION

In this section we reproduce the main experiments from the original noise corrected backboning paper [6]. The aim of this section is to motivate our choice of the noise corrected backbone over the alternative most used backboning strategies: High Salience Skeleton [12], Doubly-Stochastic Transformation [36], and Disparity Filter [35]. The idea is that any backboning method could be used to drive the sampling process, and we focus on noise corrected backboning because it is the one performing the best among the currently available alternatives.

4.1 Synthetic Networks

In this section we test the performance of each method in recovering the backbone of synthetic networks. In this scenario, we have perfect information about which edge is part of the actual network and which edge reflects noise. We generate several Barabasi-Albert random networks with average degree 3 and 200 nodes. We set η as our noise parameter. Each actual edge in the Barabasi-Albert network carries the following weight:

$$N_{ij} = (k_i + k_j) \times \mathcal{U}(\eta, 1),$$

where k_i is the degree of node i, and $\mathcal{U}(\eta, 1)$ is a number extracted from a uniform distribution with minimum η and maximum 1. In practice, we use a fraction of at least η of the sum of the degrees of the connected nodes. In this way, we ensure broad edge weight distributions locally correlated with the network topology. Then, the complement of the adjacency matrix is filled with noisy edge weights, which are defined with the same formula, only changing the uniform element with $\mathcal{U}(0, \eta)$. In practice, a noisy edge can have at most a fraction η of the degrees of i and j.

For all methods we set the parameters (if any) so that the backbone will return the same number of edges as the underlying actual network. Our quality target is the Jaccard coefficient between the set of edges of the original non-noisy network and the backbone. It is equal to one if the two edge sets are identical, and to zero if they do not share a single common edge.

Figure 4 reports the results. For very low amount of noise, the two best performing solutions are the Disparity Filter and the naive thresholding. However, our Noise-Corrected backbone is more resilient to increasing noise with the best overall performance, while also performing very well in low-noise environments. As noise increases, there is no significant difference between DF and naive thresholds.

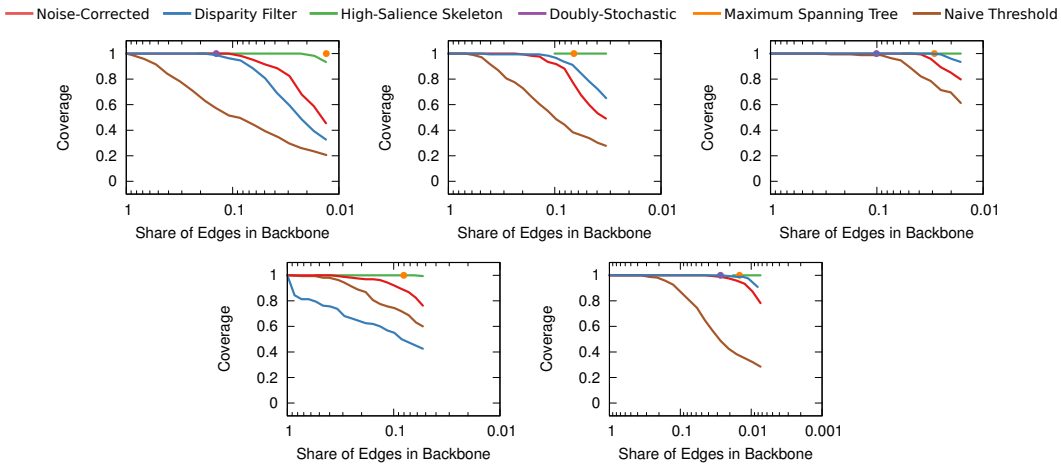


Fig. 5. Coverage per backbone for varying threshold values. From left to right: (top) Country Space, Flight, Migration; (bottom) Ownership, Trade.

4.2 Real World Data

Our test set includes five networks. In all networks, nodes are countries and connections are relationships among them, calculated in five different ways. The five networks, in alphabetical and discussion order, are as follows.

- **Country Space**: two countries are connected with the number of products they both export in significant quantities. This is an undirected co-occurrences network, observed in the years 2011, 2012 and 2013. Trade data comes from [10]. To determine whether exports are significant we use the same criterion of [13], based on the concept of Revealed Comparative Advantage.
- **Flight**: two countries are connected through the existing passenger capacity in flights from airports in one country (origin) to another (destination). This is a directed flow network, observed in the years 2010 and 2014. Proprietary data from OAG[3].
- **Migration**: two countries are connected through the number of total migrants from one country (origin) currently living in another (destination). This is a directed stock network, observed in the years 1990, 2000, 2010 and 2013. Data from the UN [40].
- **Ownership**: two countries are connected through the number of total establishments in a country (destination) reporting to a global headquarter in a different country (origin). This is a directed stock network, observed in the years 2008, 2011 and 2014. Proprietary data from Dun & Bradstreet[4].
- **Trade**: two countries are connected through the total dollar value of all products exported by one country (origin) and imported by another (destination). This is a directed flow network, observed in the years 2011, 2012 and 2013. Trade data have been cleaned with the same procedure outlined in [13].

As the first quality criterion, we focus on the topology of the backbone. In general, backbones ideally isolate as few nodes as possible: each node dropped by the backbone is a node for which

[3]http://www.oag.com/
[4]http://www.dnb.com/

Method	Country Space	Flight	Migration	Ownership	Trade
Doubly Stochastic	2.0975	n/a	1.5153	n/a	0.9287
Naive Threshold	0.6834	0.5196	1.1616	1.2384	0.3935
Disparity Filter	1.4082	0.8569	2.0715	0.5374	0.9024
High Salience Skeleton	1.6549	0.9447	1.2597	0.9744	0.8662
Maximum Spanning Tree	1.9180	0.7981	1.0036	0.9288	0.9532
Noise-Corrected	**2.2437**	**1.4676**	**2.1493**	**1.4165**	**1.1037**

Table 2. The improvement in predictive power when using backbones in our five networks.

we will not have any result from the network analysis. Thus, we define the Coverage as the ratio between non-isolated nodes in the backbone over non-isolated nodes in the original network, or:

$$Coverage = \frac{|\mathcal{V}| - |I_G^*|}{|\mathcal{V}| - |I_G|}.$$

Figure 5 reports the number of nodes preserved in each backbone as a function of the share of edges that was preserved. Note that the Doubly Stochastic method is present only for the networks Country Space, Migration and Trade, as for the other networks the stochastic transformation was not possible. Note that DS and MST do not require any parameter, so they appear in the plot as a point rather than as a line.

Many data points overlap because in many instances all methods were able to preserve the entirety of the node set. However, it is easy to detect the cases in which a particular method was not able to achieve perfect coverage. MST, DS and HSS achieve perfect coverage by definition (the latter fails only for very strict parameter choices). There is no clear winner between NC and DF, as in some networks one achieves better coverage than the other, while the converse is true for others. However, the DF is the only method underperforming the naive method in one case (the Ownership network), which is a critical failure.

As a second test, we argue that a good backbone lets the underlying properties of the data emerge from the noisy data. To prove this point we create a series of models for OLS regressions. These models all have the same structure:

$$\log(N_{ij} + 1) = \beta X_{ij} + \epsilon_{ij}.$$

Here, N_{ij} is the edge weights of the network, X_{ij} is a collection of variables that are supposedly good predictors of N_{ij}, and ϵ is the error term. In practice, we propose a model to explain the connection strength between countries. Then, for each backbone methodology we run two regressions. In the first regression M_{full}, we use the complete set of observations. In the second model M_{bb}, we restrict the observations edges that are contained in the network backbone. Quality is then defined as:

$$Quality = \frac{R^2_{M_{bb}}}{R^2_{M_{full}}},$$

which is the ratio of the quality prediction (the R^2 of the OLS model) obtained using the backbone to restrict observations over the baseline quality that uses all edges. A value of 1 means that the two regressions have equivalent predictive power, while a value higher than 1 means we improve over the full network.

To allow for a fair comparison of different backbone methodologies, we fix the number of edges we include in the backbone. We usually choose the number of edges obtained with low threshold

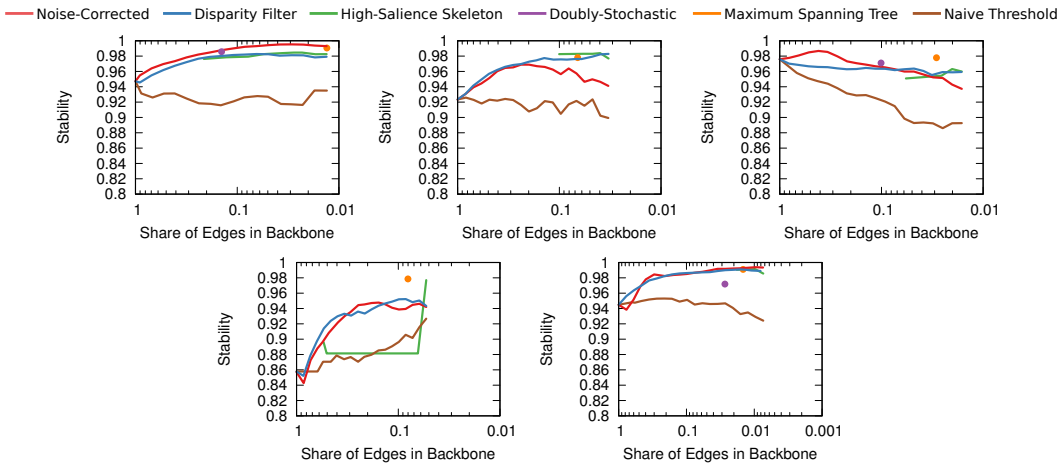


Fig. 6. Stability per backbone for varying threshold values. From left to right: (top) Country Space, Flight, Migration; (bottom) Ownership, Trade.

values for the High Salience Skeleton, because it is the most strict backbone methodology in our collection, always returning the fewest number of edges. Note that this does not apply to the MST and DS backbones, since they do not have parameters and thus the number of edges cannot be tuned.

Table 2 reports the results. We highlight the best performing methodology in boldface. Note that in some cases the doubly stochastic transformation was not possible, thus we label these instances as "n/a". In all cases, the NC backbone performs better than any other backbone. What is more, the NC backbone is also the only method that always returns a quality value higher than one, meaning that the backbone outperforms the original, unfiltered, network in all cases.

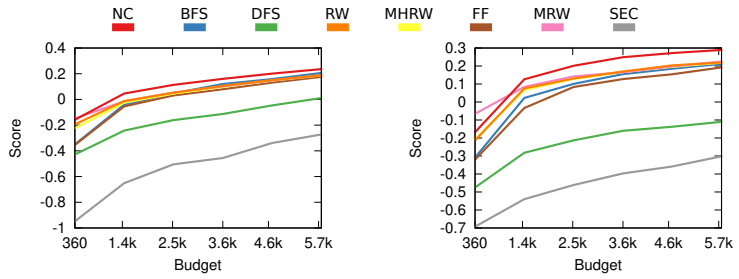
Finally, we are interested in the stability of a backbone. Because most of our networks should be relatively stable, we consider wild year-on-year fluctuations in an edge's weight as a sign that the edge weight is imprecisely measured. Our backbone should contain fewer noisy edges and therefore be more stable than the original network. The stability of a backbone method can be calculated as:

$$Stability = corr(N_{ij}^{t}, N_{ij}^{t+1}),$$

where *corr* is the Spearman correlation coefficient between the two vectors. In principle, any distance metric is appropriate for this task, but we prefer the nonparamatric nature of the Spearman correlation, which mimics our task of ranking edges according to their significance. We calculate the correlation using only the edges present in the backbones. This means that a perfectly stable backbone will have a stability of 1, while a value of 0 implies that there is no relation between the backbones of time t and $t + 1$. We calculate stability across different thresholds.

Figure 6 depicts the results. There is no clear winner in this quality criterion. All backbones are very stable, with stability always exceeding .84.

In conclusion, we can say that the Noise-Corrected backbone performs well on three critical evaluation criteria. NC backbones can handle low and high amounts of noise, as shown in synthetic network experiments. NC backbones have comparable coverage and stability with the state of the art. NC backbones are of high quality, being able to improve the performance of edge weight predictive models, also in real-world scenarios, as our case study shows.



(a) SPLL (b) LPHL

Fig. 7. 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)

5 EXPERIMENTS

Summing up the Methods section, we are performing tests across: (i) four different API systems (SPHL, SPLL, LPHL, and LPLL); (ii) six network characteristic analyses; and (iii) seven network topologies. This is a total of $4 \times 6 \times 7 = 168$ tests. Rather than reporting on all of the tests, we aggregate them across two dimensions, leaving one dimension to be examined at a time.

When aggregating over the measures presented in Section 3.2.2, we have the problem that they are expressed in different units. Moreover, some are similarities (e.g. degree correlations) and some are distances (e.g. mean absolute error). Thus we need to reduce them to comparable units. 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".

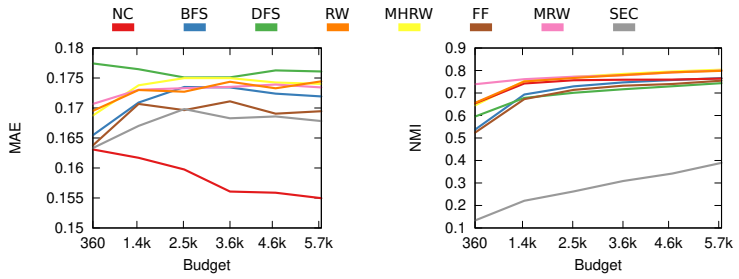
Note that the aim of this section is not to show that the noise corrected sampling is better than the alternatives in all scenarios. No method is. Rather, the objective of this section is to find out *in which* scenarios noise corrected sampling is better and in which scenarios it is worse. This is the reason why, for each subsection, we only show two plots: the one corresponding to the test in which the noise corrected sampling performed best (relative to the alternatives) and the one corresponding its worst relative performance. The supplementary materials contain all plots.

Here we test the noise corrected sampling method against: DFS, BFS, RW, MHRW, MRW, FF, and SEC. Since in total we have eight methods, an average rank lower than 4 indicates a better-than-average performance.

5.1 By API Features

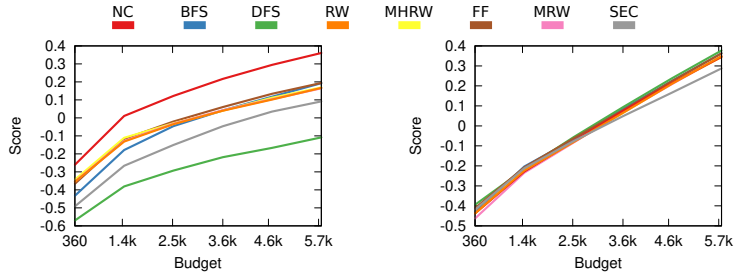
The first dimension we analyze is the features of the API system. The question we ask here is: under which combination of page size and query latency does noise corrected work better than the alternatives? To answer this question, we calculate the rank of the noise corrected sample among all tested methods,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

Figure 7 shows the score evolution for increasing budgets – recall that, in this test, we are looking at the aggregated score for which "higher is better". According to the average rank method, the noise corrected sampling works best for the SPLL API (average rank of 1.16 – Figure 7(a)) and worst for LPHL (average rank of 2.66 – Figure 7(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



(a) Disassortativity (b) Communities

Fig. 8. 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. 9. 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)

almost all budget levels, while in LPHL it underperforms for low budgets and it is best for high budgets.

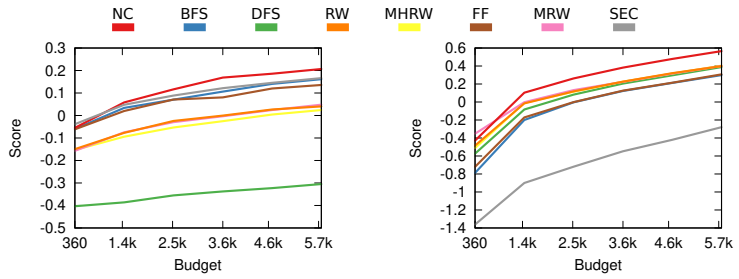
5.2 By Analysis

Here, we repeat the analyses from the previous section, but aggregating results not across API systems, but across all network analyses.

Figure 8 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 – Figure 8(a), note that this is a mean absolute error, so lower is better) and worst to recover the network's communities (average rank of 4 – Figure 8(b), not that this is normalized mutual information so higher is better). In general noise corrected allows precise estimation of node-level properties (disassortativity, assortativity, and centrality all have average rank < 3) while it performs poorly for meso-level analyses such as the ones involving communities and degree distributions (average rank > 3).

5.3 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".



(a) Weighted (b) Unweighted

Fig. 10. 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)

Figure 9 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 5.16 – Figure 9(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 – Figure 9(a)), which is the second most realistic synthetic topology we test.

In the case of real world networks, noise corrected sampling performed well: the average ranks were 1.16 and 2.3 for the social media and the bipartite network, respectively, well below the average rank of 4.

5.4 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 10 shows the overall performance. We can see that the noise corrected sampling has a clear edge for high budget situations, it suffers – while still being close to the top of the lot – under lower budgets.

Finally, Table 3 sums up the overall rank of each method across all our dimensions, including budget levels. As we can see, the highest ranking method is our proposed method. However, one should be cautious in taking these results at face value. We have shown cases in which noise corrected performed poorly, and should thus be avoided in case those analytic scenarios are the ones targeted by the analyst. One cannot take Table 3 as a proof that NC sampling is best in all cases: rather, it is best only in the general case where there is no knowledge of the underlying network topology, API characteristics, or of the final analysis to be performed on the network.

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.

Method	AVG Rank
NC	3.52
DFS	3.74
FF	3.95
MRW	3.98
RW	3.99
MHRW	4.06
BFS	4.13
SEC	4.28

Table 3. The average rank of each sampling method, across all budget levels, topologies, analyses, and APIs. (Lower is better)

6 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 run 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 (in case network data are being crawled 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. When averaging across different analytic scenarios, our method is the best performing, with a clear advantage in case of high crawling 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, is promising. Finally, this paper relies on small test networks. Testing these methods on real world data is challenging, as we would need the entire network to evaluate performance, but it would provide further 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] Eunjoon Cho, Seth A Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1082–1090.

[6] 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.

[7] 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.

[8] Xiaolin Du, Dan Wang, Yunming Ye, Yan Li, and Yueping Li. 2019. SGP: a social network sampling method based on graph partition. *International Journal of Information Technology and Management* 18, 2-3 (2019), 227–242.

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

[10] Guillaume Gaulier and Soledad Zignago. 2010. Baci: international trade database at the product-level (the 1994-2007 version). (2010).

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

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

[13] Ricardo Hausmann, César A Hidalgo, Sebastian Bustos, Michele Coscia, Sarah Chung, Juan Jimenez, Alexander Simoes, and Muhammed Yildirim. 2014. *The atlas of economic complexity: Mapping paths to prosperity*. MIT Press.

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

[15] 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.

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

[17] 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.

[18] László Lovász. 1993. Random walks on graphs. *Combinatorics, Paul erdos is eighty* 2, 1-46 (1993), 4.

[19] 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.

[20] Arun S Maiya and Tanya Y Berger-Wolf. 2011. Benefits of bias: Towards better characterization of network sampling. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 105–113.

[21] Riccardo Marcaccioli and Giacomo Livan. 2019. A Pólya urn approach to information filtering in complex networks. *Nature communications* 10, 1 (2019), 1–10.

[22] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46, 253 (1951), 68–78.

[23] 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.

[24] 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.

[25] Edward F Moore. 1959. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*. 285–292.

[26] Yohsuke Murase, Hang-Hyun Jo, János Török, János Kertész, and Kimmo Kaski. 2019. Sampling networks by nodal attributes. *Physical Review E* 99, 5 (2019), 052304.

[27] Frank MH Neffke. 2019. The value of complementary co-workers. *Science advances* 5, 12 (2019), eaax3370.

[28] M Newman. 2018. Network reconstruction and error estimation with noisy network data. *arXiv preprint arXiv:1803.02427* (2018).

[29] Tiago P Peixoto. 2018. Reconstructing networks with unknown and heterogeneous errors. *Physical Review X* 8, 4 (2018), 041011.

[30] 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.

[31] 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).

[32] Alireza Rezvanian, Behnaz Moradabadi, Mina Ghavipour, Mohammad Mehdi Daliri Khomami, and Mohammad Reza Meybodi. 2019. Social network sampling. In *Learning Automata Approach for Social Networks*. Springer, 91–149.

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

[34] 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.

[35] 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.
[36] 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.
[37] 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.
[38] Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing* 1, 2 (1972), 146–160.
[39] Sho Tsugawa and Hiroyuki Ohsaki. 2020. Benefits of Bias in Crawl-Based Network Sampling for Identifying Key Node Set. *IEEE Access* 8 (2020), 75370–75380.
[40] UNPD. 2013. Trends in International Migrant Stock: Migrants by Destination and Origin, The 2013 Revision. (2013).
[41] 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.
[42] Maya Wardeh, Claire Risley, Marie Kirsty McIntyre, Christian Setzkorn, and Matthew Baylis. 2015. Database of host-pathogen and related species interactions, and their global distribution. *Scientific data* 2, 1 (2015), 1–11.