Predicted Max Degree Sampling: Sampling in Directed Networks to Maximize Node Coverage through Crawling

Ricky Laishram, Katchaguy Areekijseree, Sucheta Soundarajan
Syracuse University, NY 13244-4100
Email: {rlaishra, kareekij, susounda}@syr.edu

***Abstract*—When studying large-scale networks, including the Web and computer networks, it is first necessary to collect appropriate data. There is a large body of literature on web crawling and network sampling in general; however, this work typically assumes that a query on a node either reveals all edges incident to that node (in the case of an undirected graph) or all edges outgoing from that node (e.g., in the case of a web crawler). There is relatively little work on sampling directed networks where incoming and outgoing edges may be obtained with separate queries. This type of sampling is relevant to networks like Twitter, in which the 'Friends' and 'Follower' connections are reciprocal relationships (i.e., if A is a follower of B, then B is a friend of A). In this paper we present *Predicted Max Degree Sampling (PMD)*, a new method of sampling a directed network, with the objective of maximizing the total number of nodes observed. We consider two types of crawling, corresponding to the scenarios in which there is or is not a limit on the number of nodes returned by a query. We compared PMD against three baseline algorithms with three real networks, and saw large improvements vs. baseline sampling algorithms: With a budget of 2000, PMD found 15% to 170.2% more nodes than the closest baseline algorithm for the different datasets considered.**

1. Introduction

Collecting network data is often a time-consuming task: among other things, one must deal with bandwidth limitations or API rate limits. The research literature contains a large amount of work on network sampling, but most of this work has been performed on either undirected networks or on directed networks, where a query returns the edges outgoing from a node (e.g., the web crawling scenario).

In this paper, we consider the setting in which one is sampling a directed network by collecting data through an API that returns some or all of a node's either incoming or outgoing edges with a query. This is relevant to networks such as Twitter, in which the 'Friend' and 'Follower' relationships represent the same edge, but in opposite directions (that is, if A is a friend of B, then B is a follower of A). The Twitter API supports both 'Friend' and 'Follower' requests, so one must make multiple requests to get all edges incident to a node.

The type of sampling we are considering in this paper is *sampling through crawling*. When sampling through crawling, we start with a small sub-graph (or a single node), and the complete network is initially unknown. Information about the network is obtained only through querying observed nodes. Sampling through crawling is critically important when observing certain types of data, such as the Web.

Depending on the behavior of the data source or API, we consider two cases. In the first case, there in no limit to the number of nodes returned per query: that is, a query for the in-neighbors of a node u returns all the in-neighbors of u (and likewise for out-neighbors). In the second case, there is a maximum number of nodes that can be returned in response to a query. Many APIs have such a limitation, so this variation is very important in practice (though often ignored in the literature). In this paper, we will refer to the first case as *Crawling without Limits*, and the second case as *Crawling with Limits*.

For the case of crawling without limits, we propose an algorithm which we refer to as *Predicted Max Degree Sampling (PMD)*. PMD works by using the already queried nodes to predict which un-queried nodes have the most unseen in- or out-neighbors. The appropriate queries are then performed on these open nodes.PMD is described in detail in Section 6.1.

To handle the case of crawling with limits, we propose a modified version of PMD, which we refer to as *Predicted Max Degree Sampling with Limits (PMDL)*. PMDL is described in Section 6.2.

2. Related Works

Sampling can be broadly divided into two categories: representative subgraph sampling, in which the entire graph is known and one must work with a subgraph due to computational limitations (such as memory or efficiency constraints), and sampling through crawling, in which the network data is collected by sampling. Our focus is on the latter scenario.

There are several recent works on sampling a network through crawling [1], [2]. Commonly used sampling algorithms include Breadth First Search and Random Walk and variants of these. Most existing work considers the case of

978-1-5386-2784-6/17/$31.00 ©2017 IEEE

crawling without limits: that is, it is assumed that a query returns all edges incident to a node.

Ye et al. [3] investigated the performance of common sampling algorithms on online social networks. They evaluated the performance of the different algorithms based on node and edge coverage (i.e., fraction of nodes and edges observed).

Avrachenkov et al. [4], proposed an algorithm called Maximum Expected Uncovered Degree (MEUD) for maximizing the node coverage of an undirected graph with a given query budget. The MEUD algorithm assumes that the degree distribution of the underlying graph G is known. If the degree distribution of G is not known, MEUD reduces to the Maximum Observed Degree (MOD) algorithm, a simple greedy method that queries the node with the highest observed degree.

The vast majority of work on maximizing node coverage has been performed on undirected graph, and none consider the case of crawling with limits.

3. Notation and Terminology

In this paper we denote the original directed graph as $G = \langle V, E \rangle$. $G_t^* = \langle V_t^*, E_t^* \rangle$ represents the observed graph after t queries have been made. Additionally, we define two subsets of V_t^*: C_t refers to the set of 'closed nodes', on which at least one query of either type has been made, and and O_t represents the 'open nodes', which have at least one query-type remaining (that is, they may have been in-neighbor queried or out-neighbor queried, but not both). Note that a node may be both closed and open.

The set of nodes on which we have made at least one in-neighbor or out-neighbor query are referred to, respectively, as the In-Closed Nodes C_t^i and Out-Closed Nodes C_t^o. Then $C_t = C_t^i \cup C_t^o$.

We define the set of In-Open Nodes O_t^i and Out-Open Nodes O_t^o as, respectively, $O_t^i = V_t^* \setminus C_t^i$ and $O_t^i = V_t^* \setminus C_t^o$. Then the Open Nodes O_t are defined as, $O_t = O_t^i \cup O_t^o$.

In the rest of this paper, we will use τ to represent either i or o as appropriate. For example, O_τ represents either O^i or O^i depending on the context. In such cases, we will use $\not\tau$ to refer to the opposite of τ. For example, if τ is used to refer to i, $\not\tau$ refers to o.

The sets of all in-neighbors and out-neighbors of a node u are represented by $\Gamma^i(u)$ and $\Gamma^o(u)$, respectively. To observe $\Gamma^i(u)$ and $\Gamma^o(u)$ we must query u, and $\gamma_x^i(u)$ and $\gamma_x^o(u)$ represent the results of the x^{th} query for the in-neighbors and out-neighbors of node $u \in V_t^*$. If $u \notin V_t^*$, both $\gamma_x^i(u)$ and $\gamma_x^o(u)$ are equal to $\emptyset$.

In the case of crawling without limits, $\gamma_x^\tau(u) = \gamma_{x+1}^\tau(u)$ for all x (that is, all queries return the same set, which is equal to the in- or out-neighbors of u). In this case, we omit the subscript and use the notation $\gamma^\tau(u)$.

We use $d^\tau(u)$ to refer to the τ-degree of a node u.

4. Problem Definition

In this paper, we assume that we are given a directed graph G and the only way to explore G is through crawling.

TABLE 1. THE CORRELATION BETWEEN IN-DEGREE AND OUT-DEGREE FOR DIFFERENT NETWORKS WHEN WE CONSIDER ONLY THE HIGHEST DEGREE NODES.

Top %	Wiki-Votes	Twitter-Friends	Web-Stanford
10	-0.07	0.04	-0.01
20	0.08	0.19	0.01
50	0.24	0.36	0.03
100	0.31	0.43	0.04

We are given an small initial sub-graph G_0^* (this can be constructed by a small BFS or random walk crawl, or might consist of only one node) and we can make a total of B queries. Our goal is to maximize $|V_B^*|$ (the total number of observed nodes).

As mentioned in Section 1, we consider two problem settings. In the case of crawling without limits, each query on a node $u \in V_t^*$ returns all in- or out-neighbors of u. That is, $\Gamma^\tau(u) = \gamma_x^\tau(u)$.

When crawling with limits, at most m nodes are returned per query. In this case, the number of queries needed to get all neighbors is $r^\tau(u) = \left\lceil \frac{|\Gamma_\tau(u)|}{m} \right\rceil$. So, $\Gamma^\tau(u) = \bigcup_{x=1}^{r^\tau(u)} \gamma_x^\tau(u)$.

The query budget B depends on how much time or resources the data collector has, and the value of m is set by the data source or API.

Each query $\gamma_x^\tau(u)$ consumes one unit of the budget. For each node can choose to perform in-neighbor queries, out-neighbor queries, both, or none.

5. Challenges

In our problem the network is directed, and this presents challenges not found in undirected networks.

The first challenge is that we need to consider the in-neighbors and out-neighbors of a node separately. Since each of these queries consumes one unit of budget each, we need a way to decide if we want to query the in-neighbors, out-neighbors, or both.

The second challenge is that in most real networks, there is very little correlation between the in-degree and out-degree of high degree nodes (Table 4). Because our objective is to maximize the node coverage, the nodes with high τ-degree are most useful, but it is difficult to determine whether a node has high out- (or in-) degree based only on its in- (or out-) degree.

In the case of crawling with limits, there is an additional challenge. In this case, we do not know $|\Gamma_\tau(u)|$ before making all $r^\tau(u)$ queries. So, we need to find a way to ensure that we do not waste queries.

6. Methodology

In this section, we describe our algorithms, *Predicted Max Degree Sampling (PMD)* and *Predicted Max Degree Sampling with Limits (PMDL)*.

Algorithm 1 BestNodes

Input:
C : The set of closed nodes
O : The set of open nodes
p : The threshold accuracy
d_ϕ : The threshold degree
k : The number of nodes to return
Output:
The set of candidate nodes and type of query for each
Initialization :
$N \leftarrow set()$
$score \leftarrow HashTable()$
$s \leftarrow SampleSize(C, d_\phi, p)$
$S^* \leftarrow RandomSample(C, s)$
for $v \in S^*$ **do**
  for $u \in \gamma^i(v) \cap O$ **do**
    $score[(u,' o')] \leftarrow score[(u,' o')] + 1$
  end for
  for $u \in \gamma^o(v) \cap O$ **do**
    $score[(u,' i')] \leftarrow score[(u,' i')] + 1$
  end for
end for
for $i \in \{1, 2, \ldots, k\}$ **do**
  $key, value \leftarrow max(score)$
  $score[key] \leftarrow -1$
  $N \leftarrow N \cup \{key\}$
end for
return N

Algorithm 2 QueryNodes

Input:
B : The budget available
p : The threshold accuracy
ϕ : The threshold percentile
k : The number of nodes to return
G^* : The initial sub-graph
Output:
The sample network G^*
while $cost \leq B$ **do**
  $d_\phi \leftarrow PercentileDegree(C, \phi)$
  $N \leftarrow BestNodes(C, O, p, d_\phi, k)$
  for $(v, \tau) \in N$ **do**
    $\alpha \leftarrow \gamma^\tau(v) \setminus (O \cup C)$
    $O \leftarrow O \cup \alpha$
    $O \leftarrow O \setminus \{v\}$
    $C^\tau \leftarrow C^\tau \cup \{v\}$
  end for
  $a \leftarrow UpdateAccuracy(N, p)$
  $k \leftarrow UpdateK(a, k)$
  $\phi \leftarrow UpdateThershold(a)$
  $cost \leftarrow UpdateCost()$
end while
return G^*

6.1. Predicted Max Degree Sampling (PMD)

For the case of crawling without limits, we present Predicted Max Degree Sampling (PMD). In this case, $O_t^\tau \cap C_t^\tau \equiv \emptyset$.

At a high level, PMD works by selecting the k nodes from O_t with the highest expected number of unobserved in/out degree, and the appropriate query for each of them. This is done by by performing in- and out-queries on a random sample of $u \in C_i \cup C_o$. The intuition is that an open node that is observed frequently during this step is likely to have higher number of in- or out-neighbors.

Our proposed algorithm has two main components - *BestNodes* (Algorithm 1) and *QueryNodes*(Algorithm 2). *BestNodes* is responsible for selecting the set of k best open nodes to query and the type of query to perform on each node. *QueryNodes* then performs the queries and updates the parameters according to the performance.

The sample size s in Algorithm 2 is calculated such that if a sample S^* of size s is selected from C_t, the probability that a node $u \in O_t$ with τ-degree $d^\tau(u) \geq d_\phi$ has an in/out neighbor in S^* is at least p. This value of s can be calculated by solving the inequality 1 (Appendix A).

$$\prod_{i=1}^{d_\phi}(|C_t| + 1 - s - i) \leq (1-p) \cdot \prod_{i=0}^{d_\phi}(|C_t| + 1 - i) \qquad (1)$$
$$s \in \mathbb{Z}_+$$

If N is the set of nodes and query types returned by *BestNodes*, the accuracy is calculated after querying all of these nodes, and it is given by the ratio of nodes that have degree greater than d_ϕ to the number of nodes returned (Equation 2). This accuracy is used to determine the value of k for the next iteration of *BestNodes*. If the accuracy is above the threshold p, the value of k is incremented, otherwise the value of k is decreased.

$$a = \frac{|\{(u, \tau) \in N \colon d^\tau(u) \geq d_\phi\}|}{|N|} \qquad (2)$$

If there are not enough nodes with degree greater than d_ϕ left, the accuracy will remain below p even after adjusting k. So if a fails to increase even after adjusting k, we decrease ϕ and d_ϕ is recalculated.

6.2. Predicted Max Degree Sampling with Limits (PMDL)

In Section 5, we discussed the additional challenges introduced in crawling with limits. In this section, we describe *Predicted Max Degree Sampling with Limits (PMDL)*.

If the maximum number of nodes that can be returned is m, we define a network model such that:

1) Every node is assumed to be made up of an ordered list of sub-nodes. That is, every node u is made up of $u'_1, u'_2, \ldots, u'_r$.
2) If a node u has $|\Gamma^\tau(u)|$ neighbors, the number of sub-nodes of u is $r^\tau(u) = \left\lceil \frac{|\Gamma_\tau(v)|}{m} \right\rceil$.
3) Every neighbor of a node u has an edge to one sub-node of u. That is for every $v \in \Gamma^\tau(u)$, there is an $i \leq r$ such that $v \in \Gamma^\tau(u'_i)$.
4) No sub-node of the same node have any common neighbor. That is, for any $0 < j < k < r^\tau(u)$, $\Gamma^\tau(u'_j) \cap \Gamma^\tau(u'_k) \equiv \emptyset$.

5) A sub-nodes cannot have more neighbors than one that come before it in the order. $|\Gamma^\tau(u'_i)| \geq |\Gamma^\tau(u'_{i+1})|$.
6) The i^{th} query on node u returns all the neighbors of sub-node u'_i only if $i \leq r'(u)$. If $i > r^\tau(u)$, it returns $\emptyset$.
7) If we have made i queries on node u, we do not know $\Gamma^\tau(u'_j)$ for any $j > i$.
8) If there are $r^\tau(u)$ sub-nodes, at least $r^\tau(u)-1$ have exactly m neighbors.

We then slightly modify the algorithm in Section 6.1.

In *SelectBestNodes*, only edges to un-queried sub-nodes are considered during score calculation. During this step, we do not need to know which of the sub-nodes an observed edges goes to since we cannot query the sub-nodes in random order.

Suppose $E^\tau(S, u)$ is the set of edges from the sub-sample S^* to a node $u \in O_t$, and we have already queried u i times. Then the set of already observed neighors of u is, $\bigcup_{x=1}^{i} \gamma_x^{\not\tau}(u)$. Then the $\not\tau$-score of u is given by,

$$score(u, \not\tau) = |E^\tau(S, u) \setminus \bigcup_{x=1}^{i} \gamma_x^{\not\tau}(u)|$$

In *QueryNodes*, C_t and O_t are not mutually exclusive. The open node $u \in O_t$ is removed from O_t after the x^{th} query only if $|\gamma_x^\tau(u)| < m$.

If the budget B is small this algorithm will have no significant improvement over a naive algorithm that queries a node until all the neighbors have been found. However it can be shown (Appendix B) that if the original network is scale-free, the number of highest degree nodes to query on completely until κ fraction of of the queries become sub-optimal is,

$$f \geq \left(\frac{\kappa(\alpha - 1)d_{min}}{m(\alpha - 2)(1 - \kappa)} \right)^{\alpha - 1} \quad (3)$$

where d_{min} is the minimum degree, α is the exponent of the power-law equation and m is the maximum neighbors returned in each query. In practice, the value of α is between 2 and 3 for most real world networks So, f is very small for most most values of κ.

7. Experimental Setup

In our experiments, we consider three baseline algorithms: Maximum Observed Degree (MOD) [4], which is the current state-of-the-art for maximizing node coverage in undirected networks, Breadth-First Search (BFS), and Random Walk. Because these algorithms were designed for undirected networks, they make two queries - γ^i and γ^o at each step. When crawling with limits, these methods continue querying a node until it has no further edges left to observe.

For experiments on crawling without limits, we use the wiki-Vote, soc-Slashdot0922, soc-Epinions1, web-Stanford and web-Google datasets from SNAP [5]. In addition, we also use a twitter-Friends dataset that we collected through the Twitter API. These datasets are described in Table 7. When crawling with limits, we use the web-Google and web-Stanford datasets. This is because API limits only become relevant in cases where the typical node has a fairly high degree; otherwise, all neighbors will be returned and the comparison reduces to the crawling without limits case.

TABLE 2. NETWORK DATASETS

Dataset	Nodes	Edges
wiki-Vote	7115	103689
twitter-Friends	12230	674141
soc-Epinions1	75879	508837
soc-Slashdot9022	82168	948464
web-Stanford	281903	2312497
web-Google	875713	5105039

The initial values of the parameters for all experiments are $p = 0.9$ and $\phi = 90$. As the algorithm adjusts these parameters by decreasing them, we assign high initial values. The inital sub-graph is generated through 20 steps of BFS. For crawling with limits, we set m (the maximum number of nodes returned) to 500.

8. Experimental Results

In this section, we present the results of PMD and PMDL against the baseline algorithms described in Section 7. Recall that the objective of PMD and PMDL is to maximize the node coverage, or fraction of nodes that have been observed. Because the total number of nodes in the complete graph G is a constant, it is sufficient to compare only the number of observed nodes, $|O_B \cup C_B|$.

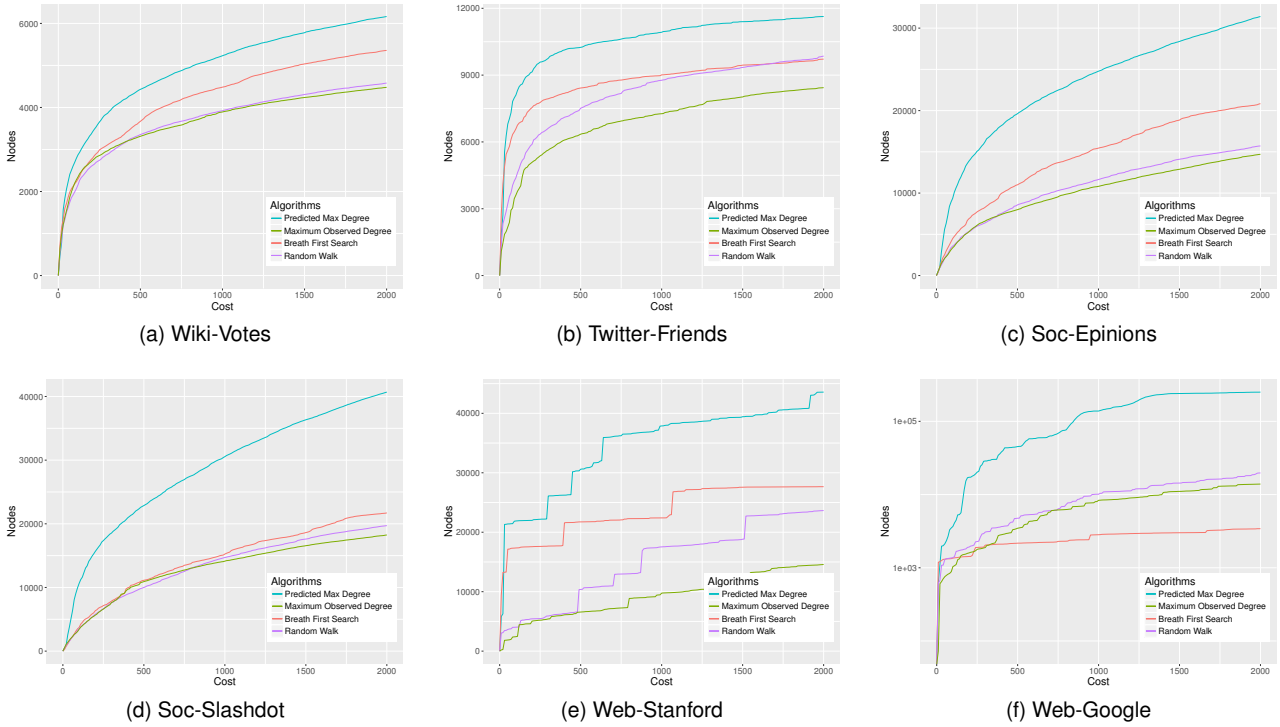
8.1. Results for PMD

For each of the datasets in Section 7, we select 10 nodes randomly as seed nodes, and we consider budgets up to a maximum of 2000 queries. Figure 1 compares the number of nodes observed by PMD and the baselines algorithms against the query budget. Figures show a mean over 10 trials.

It can be observed from Figure 1 that PMD outperforms all considered baseline algorithms over all budgets. In the case of the wiki-Votes network after 2000 queries, PMD was able to obtain 15% more nodes than the nearest baseline. In the larger networks, this difference is much higher. In soc-Slashdot network, the node coverage of PMD after 2000 queries is 87.4% higher than the nearest baseline, and in web-Google network, the difference is 170.2%.

8.2. Results for PMDL

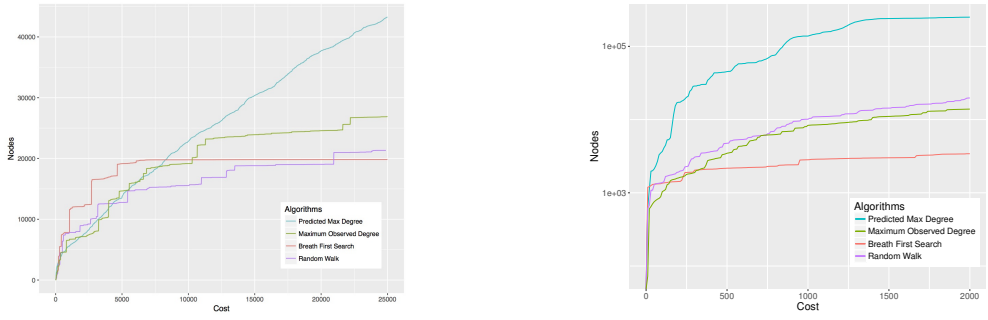
For the experiments with PMDL, we consider only the datasets with large number of nodes- web-Stanford and web-Google- as the degrees in the other datasets were too low for the crawling with limits scenario to make sense. For each of these datasets, we select 10 nodes randomly and use them as starting nodes for the algorithms.



(a) Wiki-Votes (b) Twitter-Friends (c) Soc-Epinions

(d) Soc-Slashdot (e) Web-Stanford (f) Web-Google

Figure 1. Comparison between PMD and the baseline algorithms, showing the mean node coverage after 10 trials of each method.



(a) Web-Stanford (b) Web-Google

Figure 2. The comparison between PMDL and the baseline algorithms. The lines are the mean of 10 experiments.

Figure 2 shows the comparison between Predicted Max Degree Sampling with Limits against the baseline algorithms. As expected, the algorithms that do not account for query limits initially perform well. However, as discussed in Section 6.2, our algorithm overtakes these baselines and ultimately outperforms all baselines.

9. Conclusion

In this paper, we presented the problem of sampling a directed network through crawling with the goal of maximizing node coverage. Although there has been similar work done on undirected networks, directed networks present additional challenges (Section 5). To address this problems, we presented *Predicted Max Degree (PMD) Sampling*, a novel algorithm to maximize node coverage in directed graph when crawling without limits.

We also discussed the real world problem of crawling with limits (Section 4). In Section 6.2, we showed how PMD can be modified by treating the nodes as ordered lists of sub-nodes. We call this modification *Predicted Max Degree Sampling with Limit (PMDL)*.

In Section 8, we showed that both PMD and PMDL obtain samples with significantly higher node coverage than baseline algorithms.

References

[1] A. S. Maiya and T. Y. Berger-Wolf, "Benefits of bias: Towards better characterization of network sampling," in *Proceedings of the 17th ACM*

SIGKDD international conference on Knowledge discovery and data mining, pp. 105–113, ACM, 2011.

[2] S. A. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Crawling facebook for social network analysis purposes," in *Proceedings of the international conference on web intelligence, mining and semantics*, p. 52, ACM, 2011.

[3] S. Ye, J. Lang, and F. Wu, "Crawling online social graphs," in *Web Conference (APWEB), 2010 12th International Asia-Pacific*, pp. 236–242, IEEE, 2010.

[4] K. Avrachenkov, P. Basu, G. Neglia, B. Ribeiro, and D. Towsley, "Pay few, influence most: Online myopic network covering," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pp. 813–818, IEEE, 2014.

[5] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." http://snap.stanford.edu/data, June 2014.

Appendix A.
Sub-Sample Size Calculation

Consider a node, $v \in O_t$ with out-neighbors $\Gamma_o(v)$ and $d_o = |\Gamma_o(v)|$. Assume that there are $|C_t|$ closed nodes, and we want to select s closed nodes, such that there is a p probability that the selected sample of size s contains at least one neighbor of n. There are $(|C_t| - d)$ nodes that are not in $\Gamma_o(n)$.

The number of ways to select s nodes out of the $(|C_t| - d)$ nodes is $\binom{|C_t| - d_o}{s}$. The number of ways to select s nodes out of the $|C_t|$ nodes is $\binom{|C_t|}{s}$.

If p is the probability that at-least one of the s nodes is in $\Gamma_o(n)$,

$$p \leq 1 - \frac{\binom{|C_t| - d_o}{s}}{\binom{|C_t|}{s}}$$

$$\prod_{i=1}^{d_o}(n+1-s-i) \leq (1-p) \cdot \prod_{i=0}^{d_o}(n+1-i)$$

We take the log of both sides to make it computationally easier to solve and generalize for in-neighbors and out-neighbors by replacing o with τ.

$$\sum_{i=1}^{d_\tau} log(|C_t| + 1 - s - i) \leq log(1-p) + \sum_{i=1}^{d_\tau} log(|C_t| + 1 - i)$$

We then solve the system of inequalities for $s \geq 0$ and $s \in \mathbb{Z}$.

If s_x and s_y are the sample sizes for threshold degree d_x and d_y respectively, and $d_x < d_y$, it can be shown numerically that, $s_x > s_y$. So it is enough to solve s for,

$$d_\phi = min(d_i, d_o)$$

Appendix B.
Predicted Max Degree Sampling with Limits

For a node v, let the number of sub-nodes be represented by $r(v)$. If $\Gamma_\tau(v)$ is the number of τ-neighbors of node v and m is the maximum number of nodes a query returns,

$$r(v) = \left\lceil \frac{N(n)}{m} \right\rceil$$

It is guaranteed that $r(v) - 1$ nodes return m nodes. So the maximum relative number of sub-optimal queries is $\frac{1}{r(v)}$. If we consider nodes $v_1, v_2, v_3, \ldots, v_l$, it can be shown that the maximum relative number of sub-optimal queries is,

$$SQ = \frac{l}{\sum_{i=1}^{l} \left\lceil \frac{N(v_i)}{m} \right\rceil}$$

Assume the networks is scale-free, $p(x) = c \cdot x^{-\alpha}$
For some x_i, the number of nodes such that $x > x_i$ is,

$$P(x > x_i) = \int_{x_i}^{\infty} p(x)dx = \frac{c}{(\alpha - 1)\, x_i^{\alpha-1}}; \alpha > 1$$

The number of sub-nodes for nodes such that $x > x_i$,

$$Q(x > x_i) = \int_{x_i}^{\infty} \left\lceil \frac{x}{m} \right\rceil c x^{-\alpha} dx$$

If we define,

$$\begin{aligned} Q_u(x > x_i) &= \int_{x_i}^{\infty} \left(\frac{x}{m} + 1\right) c x^{-\alpha} dx \\ &= \frac{c}{m(\alpha-2)x_i^{\alpha-2}} + \frac{c}{(\alpha-1)x_i^{\alpha-1}}; \alpha > 2 \end{aligned}$$

Then,

$$SQ_l = \frac{P(x > x_i)}{Q_u(x > x_i)} = \frac{m(\alpha-2)x_i^{\alpha-2}}{(\alpha-1)x_i^{\alpha-1} + m(\alpha-2)x_i^{\alpha-2}}$$

If we want to achieve a minimum of SQ of κ at $x_i = x_l$, it can be shown that,

$$x_l \geq \frac{m(1-\kappa)(\alpha-2)}{\kappa(\alpha-1)}$$

If we assume that the minimum value of x is x_{min}, the fraction of high-degree nodes after which κ queries are sub-optimal is,

$$\begin{aligned} f &= \frac{P(x > x_l)}{P(x > x_{min})} \\ &\geq \frac{c\kappa^{\alpha-1}(\alpha-1)^{\alpha-2}}{m^{\alpha-1}(\alpha-2)^{\alpha-1}(1-\kappa)^{\alpha-1}} \cdot \frac{(\alpha-1)x_{min}^{\alpha-1}}{c} \\ &\geq \frac{\kappa^{\alpha-1}(\alpha-1)^{\alpha-1}x_{min}^{\alpha-1}}{m^{\alpha-1}(\alpha-2)^{\alpha-1}(1-\kappa)^{\alpha-1}} \\ &\geq \left(\frac{\kappa(\alpha-1)x_{min}}{m(\alpha-2)(1-\kappa)}\right)^{\alpha-1} \end{aligned}$$