Max-Node Sampling: an Expansion-Densification Algorithm for Data Collection

Katchaguy Areekijseree
Department of EECS
Syracuse University
Syracuse, NY U.S.A.
kareekij@syr.edu

Ricky Laishram
Department of EECS
Syracuse University
Syracuse, NY U.S.A.
rlaishra@syr.edu

Sucheta Soundarajan
Department of EECS
Syracuse University
Syracuse, NY U.S.A.
susounda@syr.edu

***Abstract*—In this work, we propose *Max-Node* sampling, a novel sampling algorithm for data collection. The goal of Max-Node is to maximize the number of nodes observed in the sample, given a budget constraint. Max-Node is based on the intuition that networks contain many densely connected regions (i.e., communities), that may be only weakly connected to another, and to maximize the number of nodes observed, it is critical to transition between communities. The two key phases of our algorithm are Expansion and Densification. The goal of the Expansion phase is to transition to unobserved regions, while the Densification phase aims to collect as many nodes in the current community. We conduct experiments on several real networks, and show an improvement of up to 40% vs. the baselines.**

***Index Terms*—Network Sampling; Data Collection; Data Crawling, Large Graph, Complex Network, Algorithms;**

1. Introduction

The rise of online social networking sites in recent years has produced a gold mine of data. By analyzing these networks, researchers can understand the interesting behaviors and phenomena which happen in real world systems. However, before data can be analyzed, it must first be collected.

These social networking platforms provide a channel for collecting the data through their APIs. Unfortunately, the APIs come with a limitation. For example, Twitter allows only 15 requests per 15 minutes for crawling following/follower relationships, while Linkedln allows around 1,000 requests for the same interval. As described in [1], it took almost six days to collect all the friends and followers of 8,000 unique users on Twitter. Data collection is a time consuming task and poses a challenge. Given that collecting data is a time-consuming process, how should one determine which nodes to query so that the resulting sample is optimal with respect to a desired goal?

In this paper, we introduced *Max-Node* sampling, a novel sampling algorithm for the task of collecting data with the goal of maximizing the total number of nodes observed given a limited query budget. The intuition behind Max-Node comes from the observation that networks consist of many communities, which are internally tightly connected. When sampling, it is necessary to transition between these communities in order to observe as many nodes as possible. Our experiments on real networks demonstrate that Max-Node performs up to 40% better than comparison strategies.

2. Related Works

Because network data is growing rapidly, there is a large literature on network sampling. Sampling is necessary for two reasons: (1) Current computing power cannot always handle the sizes of available data, and (2) Collecting data is time-consuming. For example, it is impossible to collect the whole Facebook network within a reasonable amount of time.

Accordingly, network sampling can be separated into two main scenarios. The first scenario is referred to as "scaling-down" or "down-sampling". In this scenario, the entire network data is available, and the goal is to scale down the network to some desired size, and the sample graph should preserve the network properties as a representative of the original network. The second scenario, and the focus of this paper, is a data collection scenario where we have a limited view of the network. Here, decisions about how to grow the sample are based only on the data that has been observed so far.

In [2], the authors study the characteristics of different sampling algorithms. They evaluate how well the output graphs from different algorithms capture network properties. Similarly, Maiya and Berger-Wolf present an algorithm that aims to preseve the community structure of the original network [3]. This algorithm is built on the concept of expander graphs. The results shows that sample is capable of capturing the community structure.

The most relevant work to this paper is presented in [4]. This work introduces a greedy sampling approach called Maximum Observed Degree (MOD). The goal of this algorithm is to maximize the network coverage, which is as same as our objective. In MOD, in each step the algorithm selects the node with the largest observed degree in the sample. Their experimental results show that MOD outperforms other algorithms such as BFS, DFS and RW. To the best

of our knowledge, MOD is currently the best algorithm in this class.

3. Problem Definition

In this work, we focus on sampling under the network crawling scenario. For example, suppose that we want to obtain data from Twitter, and we have only 24 hours to collect the data. The goal is to get as many users as possible. The process starts by selecting one known user account. Then, we send a request through the API asking for the followers of this account. The server responses by returning a list of users back. All the users are stored in the list, and our next query must be selected from this list. At each step, the node we query must be one observed from a previous query. The output is a set of unique users in the list.

Definition: Suppose there is a true, underlying undirected network $G(V, E)$, where V is set of nodes (users), E is a set of edges (activities). We assume that we have no information about G. We are given a starting node(n_{start}) and a number of API requests $(budget)$. We are allowed to send a request to an API asking for neighbors of the specified node. The API returns all neighbors of the queried node. Our goal is to collect a sample graph $S(V', E')$, $V' \subseteq V$ and $E' \subseteq E$, where the number of nodes in $|V'|$ is maximized.

4. Proposed Method

In this section, we introduce the novel sampling algorithm *Max-Node*. The current state-of-the-art algorithm, Maximum Observed Degree, in each step queries the node with the highest observed degree, with the assumption that this node has high unobserved degree as well. Max-Node is based on the intuition that real networks exhibit community structure, and so if one queries the node with the highest observed degree, one may get 'stuck' in a community. Max-Node thus consists of two phases: *Densification*, which queries nodes in the observed region to fill out that region, and *Expansion*, which transitions the sampling algorithm to a new region of the graph.

For example, suppose the data collection process starts at a node in the bottom-left cluster in Figure 1. As we can see, this network has several communities. The green area is already explored, and the sample obtained so far is from this region. The rest are the nodes have not been seen yet. The Densification phase aims to collect as many nodes that are densely connected. When the algorithm collects most of the nodes in that region, the algorithm switches from Densification to Expansion. The Expansion phase aims to escape from the current region. It picks an appropiate node that will lead to a new region. The algorithm switches between these two phases until it runs out of the budget. Pseudocode is shown in Algorithm 1 and a list of variables in Table 1.

Algorithm 1 starts by collecting a small sample. The initial sample can be collected by any crawling technique.

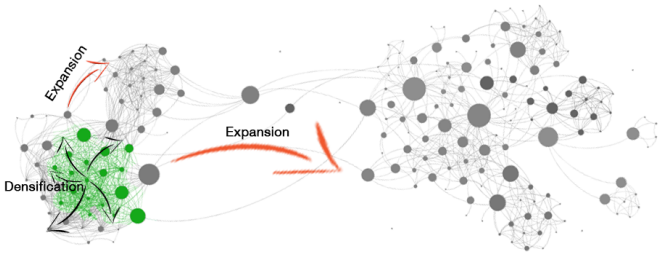


Figure 1: Concept of Expansion-Densification Algorithm

In our case, we adopt BFS crawling as shown in line 2. BFS begins from any node n_{start}, queries the API for the node's neighbors, and adds all neighbors that have not been queried to an *unqueried* queue. The first node in the queue will be a next selected node. BFS is repeated for $budget_{bfs}$ times. All the nodes and edges found are kept as the initial sample. We define two types of nodes, *closed* and *open* node. A *closed node* is a node that has already been queried. An *open node* is a node that has been observed, but not queried.

The main loop (line 3-7) of the algorithm switches between $Expansion$ and $Densification$, and it runs until it reaches a specified amount of budget. The input for $Expansion$ is the sample graph that obtained so far, and it returns a single node (n_{exp}), with the hope that n_{exp} leads to a new region. $Densification$ acquires n_{exp} as its input and tries to explore this new area of the network. It returns the sub-sample graph(S_s). Finally, S_s is merged with S.

Algorithm 1 Exp-Den $(budget, budget_{bfs}, n_{start})$

1: $cost \leftarrow 0$
2: $S \leftarrow bfs(budget_{bfs}, n_{start})$
3: **while** $cost \leq budget$ **do**
4: $\quad n_{exp} \leftarrow Expansion(S)$
5: $\quad S_s \leftarrow Densification(n_{exp})$
6: $\quad S \leftarrow$ Merge S_s with S
7: **end while**

TABLE 1: Description of each variable

	Description
N_q	a set of nodes returned from the API request
E_q	a set of edges returned from the API request
S	a sample graph $S(V', E')$
S^o	a set of open nodes $n \in V'$
S^c	a set of closed nodes $n \in V'$
S_s	a sub-sample graph $S_s(V_s, E_s)$
S_s^o	a set of open nodes $n \in V_s$
S_s^c	a set of closed nodes $n \in V_s$
e_{ij}'	$e_{ij} \in E_s, (i \in S_s^c \wedge j \in S_s^o) \vee (i \in S_s^o \wedge j \in S_s^c)$
sc_{exp}^t	an expansion score at t^{th} iteration
sc_{den}^t	a densification score at t^{th} iteration
n_{exp}	a selected node from Expansion phase
w_1, w_2	weight
$\lvert \cdot \rvert$	a cardinality of a set

Algorithm 2 Densification (n_{cur})

$S_s \leftarrow empty$, $sc^0_{den} \leftarrow 1$, $sc^0_{exp} \leftarrow 0$
while $sc^t_{exp} < sc^t_{den}$ or $cost < budget$ **do**
  $N_q, E_q \leftarrow$Make a query on n_{cur}
  $sc^t_{den} \leftarrow w_1 \times \frac{|n \in N_q \setminus (S^o \cup S^o_s)|}{|S^c_s|} + w_2 \times sc^{t-1}_{den}$
  $sc^t_{exp} \leftarrow w_1 \times \frac{|e^{'}_{ij}|}{|S^o_s|} + w_2 \times sc^{t-1}_{exp}$
  $S_s \leftarrow$ Add N_q, E_q to sub sample
  $n_{cur} \leftarrow$node with the highest degree in S_s
  $cost \leftarrow cost + 1$
end while
  return S_s

Densification: To expand a sample within a region, we adopt Maximum Observed Degree (MOD) [4], as it outperforms other algorithms in the same class. Pseudocode is shown in Algorithm 2. In each iteration, the node with maximum degree is selected from S^o_s and the algorithm requests its neighbors through the API. Nodes (N_q) and edges(E_q) are returned and added to sub-sample (S_s).

Expansion: In the Expansion step, the algorithm tries to escape from the current region of the network. The algorithm selects a node that will lead to another dense area. In the spirit of explore-exploit algorithms, one naive appoarch is to pick a node uniformly at random from S^o. We refer this Expansion strategy as "$random$" (in our future work, we examine other strategies for Expansion).

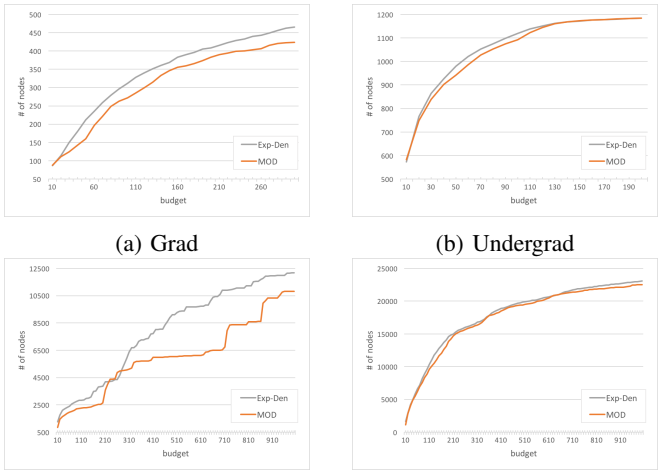
Switching Phases: Two scores are calculated in each iteration of Densification. Intuitively, in each step, the number of closed nodes increases while a number of new nodes added decreases over time (diminishing marginal returns). The algorithm will find many nodes in the same community at the beginning and this amount drastically drops when most of them are found. sc^t_{den} measures how many new nodes are added to the sample after a request, divided by the number of closed nodes. sc^t_{exp} is the fraction of the number of edges ($e^{'}_{ij}$) connecting a closed node to an open node, divided by the number of open nodes in sub-sample. If the number of edges (e'_{ij}) increases, the number of open nodes also increases. If not, it means the algorithm already found most of the nodes. These scores give us an approximation of number of nodes left unexplored. Densification switches to Expansion when sc^t_{exp} is higher than sc^t_{den}. Thus, the algorithm can appropiately switch between phases.

5. Experiments and Discussion

To mimic the process of querying an API, we simulate sampling from an existing network dataset. We compare our algorithm with the MOD algorithm [4]. For simplicity, we assume all networks are undirected. We use four different datasets, described in Table 2. *Grad* and *Undergrad* are the Facebook networks [5]. *Enron-Email* is an email communication network. $Twitter$ is a friend-follower network that we collected via Twitter API.

TABLE 2: Names and statistics of datasets used in our work.

Network	# Nodes	# Edges	Global CC.	Mod.
Grad	503	3256	0.4792	0.6915
Undergrad	1220	43208	0.2980	0.3937
Twitter	12230	50884	0.1117	0.6371
Enron-Email	36692	183831	0.4970	0.5975



(a) Grad (b) Undergrad

(c) Twitter (d) Enron-Email

Figure 2: Experimental results on each dataset

We ran 15 experiments on each dataset, and plotted average values in Figure 2. These plots depict the between amount of budget used versus the number of nodes obtained. Our algorithm outperformed MOD in every case. This gives us strong evidence that Max-Node algorithm is able to collect more nodes than MOD at the same amount of budget. Interestingly, on the Twitter dataset, we clearly see that MOD becomes trapped in a region before being forced into a new region (indicated by steps in the result curve). A large budget is spent, but few nodes are added to the sample.

With a budget constraint, Max-Node performs well. Our future work includes improving the Expansion strategy with different switching criteria.

References

[1] J. D. Wendt, R. Wells, R. V. Field Jr, and S. Soundarajan, "On data collection, graph construction, and sampling in twitter."

[2] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 631–636.

[3] A. S. Maiya and T. Y. Berger-Wolf, "Sampling community structure," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 701–710.

[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*. IEEE, 2014, pp. 813–818.

[5] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: inferring user profiles in online social networks," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 251–260.