Expansion-Densification Algorithm for Data Collection

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

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

***Abstract*—In this work, we propose the novel sampling algorithm for data collection, called Expansion-Densification. The intuition comes from the nature of the networks where it consists of many densely connected parts, refer to communities. Our objective is to maximize a number of nodes obtained in the sample under a budget constraint. Two keys procedures of our algorithm are Expansion and Densification. Expansion aims to find a direction which leads to other unexplored areas. While the Densification aims to collect as many nodes in the community. We conducted the experiments on real-world networks. The results show that our algorithm outperforms the existing algorithm in most of the cases.**

***Keywords*-Network Sampling; Data Collection; Algorithms;**

I. Introduction

Nowadays, social network sites are very popular and gain a lot of attention from users, developers and researchers. There are many social network sites that serve different purposes on the internet. For instance, Twitter, Facebook, Linkedln and etc. These are the virtual worlds where people can connect with friends, family, co-workers. People can easily communicate and share their stories to others. It is a place that contains a lot of information from the users.

The networks can be represented by graph G(V,E) where V is set of vertices and E is a set of edges. A vertex represents a person and an edge represents a relationship, e.g. frienship, following, follower, or an activity between two users. By performing an analysis on these graphs, researchers can study interesting behaviors that happen on the network. However, a process of data collection need to be done before perform a further analysis.

These social network platforms provide a channel for collecting these data through their APIs. Unfortunately, these APIs come with a limitation. For instance, Twitter allows only 15 requests per 15 minutes for crawling on follower edges while Linkedln allows around 1,000 requests for the same interval. As mentioned in [1], it took almost six days to collect all of the friends and followers of 8,000 unique users on Twitter. As we can see, data collecting is a time consuming task and poses a challenge. With the limited information of the network, how can we decide which node is worth for the request and the sample is reach the desire goal. In this work, we focus on algorithm that is capable of efficiency collecting the data under a specific budget where the number of users in the sample is maximized.

In this paper, we introduced a novel sampling algorithm called *Expansion-Densification* algorithm. The intuition comes from an observation that the networks consist of many communities. The existing algorithms tends to get a same set of users at some point during the data collection, since those users are densely connected. The goal of this work is to create an sampling algorithm that is capable of creating a sample graph which maximize node coverage under a given budget.

II. Related Works

There are many works that focus on the network sampling because network data is getting larger and larger every day. The current computing power cannot handle this such big data in order to analyse and produce the results in time. Collecting data is also another issue. It is impossible to collect the whole Facebook network within the reasonable amount of time. These are the reasons why network sampling is important and gain a lot of attention.

Network sampling can be separated into two main scenarios. The first one is a data collection scenario where we have a limited view of the network, the decision is making based on the sample. Another scenario is called "scale-down" or "down-sample". In this scenario, we have the information of the entire network. The goal is to scale down the network to some desired size and the sample graph should preserve the network properties as it is a representative of the original network.

In [2], they study the characteristics of different sampling algorithms. They evaluate how well the output graphs from different algorithms capture network properties. Similarly, the work which aims to preseve the community structure of the original network is introduced in [3]. The algorithm is build based on a concept of expander graphs. The results shows that sample is capable of capturing the community structure. Another work on sampling technique [4], they introduce 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. The idea is to select a node with the largest

observed degree in the sample in each step. The experiment shows that MOD outperform other algorithms such as BFS, DFS and RW.

III. Proposed Method

A. Problem definition

As mentioned in the previous section, we focus on the network crawling scenario, which is a case where we have a limited information about network G. We are given some amount of budget b. In our case, it is a number of requests which we can send through API. We want to construct a sample graph $S(V, E)$ where number of nodes obtained $|V|$ is maximized.

B. Proposed Algorithm

We introduce Expansion-Densification algorithm. Again, the goal of this algorithm is to maximize the node coverage of the network under a given budget. The key idea of this algorithm consists of two steps, *Densification* and *Expansion*. The intuition comes from the characteristic of the networks where the networks consist of many communities. Community refers to a subgraph where the nodes are densely connected while the connections outside the communities are very sparse.

The Densification step aims to find all possible nodes that are densely connected to each others in the particular area of the network. On the other hand, Expansion step focuses on finding a way to escape from the dense area which already explored, and it should lead to another unexplored dense area. The pseudocode of the algorithm is shown in Algorithm 1. A list of variables is shown in Table I.

Table I: Description of variables

	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 close 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 close 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_{c_i}	a set of nodes in i^{th} community
d_{c_i}	a diameter of i^{th} community
$\lvert \cdot \rvert$	a cardinality of a set

Algorithm 1 starts by collecting some small sample. The initial sample can be collected by any crawling techniques. In our case, we adopt BFS crawling as shown in line 2. Start from any starting node(n_{start}), the algorithm requests the API for node's neighbors and adds all neighbors that have not been queried to an *unvisited* 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. Here, we defined two types of nodes, *close* and *open* node. *Close node* is a node that we already know all of its neighbors. Generally speaking, we already made a request asking for its neighbors. *Open node* is a node where we know some of its neighbors and it has not been yet queried.

The main loop (line 3-7), the algorithm switches between $Expansion$ and $Densification$ and it goes until it runs out of budget. The input for $Expansion$ is a sample graph that obtained so far. It returns a single node(n_{exp}). $Densification$ acquires n_{exp} as its input and try to explore on this new area on the network. It returns sub-sample graph(S_{sub}). Then, S_{sub} is merged with S.

We adopt Maximum Observed Degree(MOD) [4] for

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: $\quad S_{sub} \leftarrow Densification(n_{exp})$s
6: $\quad S \leftarrow$ Merge S_s with S
7: **end while**

Algorithm 2 Densification (n_{cur})

1: $S_s \leftarrow empty$, $sc_{den}^0 \leftarrow 1$, $sc_{exp}^0 \leftarrow 0$
2: **while** $sc_{exp}^t < sc_{den}^t$ or $cost < budget$ **do**
3: $\quad N_q, E_q \leftarrow$Make a query on n_{cur}
4: $\quad sc_{den}^t \leftarrow w_1 \times \frac{|n \in N_q \setminus (S^{open} \cup S_{sub}^{open})|}{|S_{sub}^{close}|} + w_2 \times sc_{den}^{t-1}$
5: $\quad sc_{exp}^t \leftarrow w_1 \times \frac{|e_{ij}^{'}|}{|S_s^o|} + w_2 \times sc_{exp}^{t-1}$
6: $\quad S_s \leftarrow$ Add N_q, E_q to sub sample
7: $\quad n_{cur} \leftarrow$node with the highest degree in S_s
8: $\quad cost \leftarrow cost + 1$
9: **end while**
return S_{sub}

Algorithm 3 Expansion (S)

1: **for** c_i in C **do**
2: $\quad score_i \leftarrow \frac{|n_{c_i}|}{|V|} \times \frac{1}{d_{c_i}}$ ▷ score = gain x spread
3: **end for**
4: $C_{target} = argmax(score_i)$
5: **for** n in S^o **do**
6: $\quad d_n \leftarrow$ shortest path length from n to C_{target}
7: **end for**
8: $n_{best} = argmin(d_n)$
return n_{best}

$Densification$ as shown in Algorithm 2. In each iteration, a node with maximum degree is selected from S_s^o and it is requested for its neighbors through API. Nodes(N_q) and edges(E_q) are returned are added to sub-sample(S_{sub}). Two scores are calculated in each iteration. sc_{den}^t measures how many new nodes are added to the sample comparing to number of close nodes in sub-sample after last request. sc_{exp}^t denotes the expansion score which is a fraction of a number of edges which one end is a close node and another end is an open node over number of open nodes in sub-sample. These scores give us an approximation of how much we can perform Expansion and Densification. The Densification stops when the sc_{exp}^t is higher.

In the Expansion step, we want to escape from the current part of the network. The algorithm selects a node in order to explore on another dense area. One naive appoarch is to pick one node uniformly at random from S^o, we refer this Expansion strategy as "$random$". Moreover, we introduce another appoarch, we refer as "*Oracle*". We assume that Oracle has the information of the entire graph. The pseudocode is shown in Algorithm 3. Firstly, Oracle selected a target community(C_t) by considering the score (line 1-3). A community with the highest score is selected. The score is a multiplication of $gain$ and $spread$. $Gain$ measures a number of nodes left unexplored. $Spread$ measures how fast we can reach all the nodes in the community. We define spread is inversely proportional to the diameter of community(d_{c_i}). After target community is selected, Oracle calculates the shortest path length from every open nodes(S^o) to the target community. A node with the smallest distance is selected. Note that, we can improve the performance here by calculating clustering coefficient of all nodes in S^o and ignore all node that the clustering coeffocient is equal to one. It is likely that these nodes are in the same community since the neighbors are tightly connected.

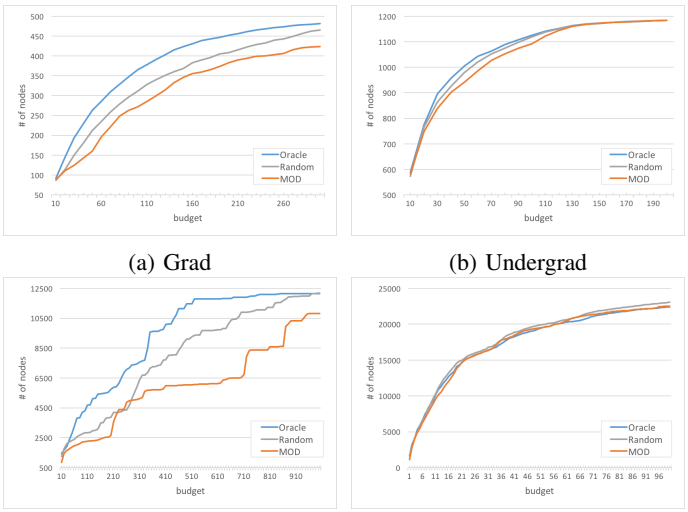
IV. Simmulation, Experiment and Discussion

We simulate the crawling process instead of directly requesting to the API. The process is mimicked by using the static network that we obtained beforehand and pretend that we have no information about the network. We compare our algorithm with pure MOD algorithm [4]. For simplicity, we assume all networks are undirected. We use four different datasets and the details are shown in Table II.

Table II: Datasets

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

We run 15 experiments for each algorithm and reported the average values in Figure 1. These are the plots between



(a) Grad (b) Undergrad

(c) Twitter (d) Enron-Email

Figure 1: Result of different algorithms on each dataset

amount of budget versus number of nodes obtained in the sample. We can clearly see that our algorithm outperform MOD in every cases. It gives us a strong evidence that Expansion-Densification algorithms is able to gain more nodes than MOD at the same amount of budget. Only the Enron-Email dataset that all three algorihms perform the same. Picking a node at random in Expansion step also gives a better sample than MOD. Interestingly, MOD is being trapped in Twitter dataset. Large budget is spent, but a few of nodes is added to sample.

With a budget constraint, Expansion-Densification performs well for data collection. Our future work includes removing Oracle and replacing it with a model that learns from the available information in the sample thus far.

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] S. Soundarajan and J. E. Hopcroft, "Use of local group information to identify communities in networks," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 3, p. 21, 2015.