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 phases 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 all of the cases.**

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

I. Introduction

Social networking sites gain a lot of attention from users, developers and researchers. It is a gold mine where tons of data is available and generated every single minute. By performing an analysis on these networks, we can study interesting behaviors and phenomena which happen in the real world system. However, a process of data collection need to be done before perform any further analysis.

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 on following/follower while Linkedln allows around 1,000 requests for the same interval. As mentioned in [1], it took almost six days to collect all 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 data should reach the desire goal.

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. Our goal is to design an algorithm that is capable of efficiency collecting the data under a specific amount of budget where the number of distinct users in the sample is maximized.

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 built based expander graphs concept. The results shows that sample is capable of capturing the community structure. Another interesting work is presented in [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 outperforms other algorithms such as BFS, DFS and RW. For the best of our knowledge, MOD is the best algorithm in this class.

III. 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. Suppose we want to collect the data from Twitter. We have 24 hours for collect the data. The goal is to get as many users as possible. The process starts by selecting one user account. Then, we send

a request through the API asking for the followers of this account. Server responses by returning a list of users back. All the users are stored in the list and we randomly select one from the list to query next. The process is repeated until the next day. The output is a set of unique users in the list.

Definition: Suppose there is a network $G(V, E)$ where V is set of nodes (users), E is a set of edges (activities), however, there is no information about G. We are given a starting node(n_{start}) and a number of API requests ($budget$). We allow to send a request asking the neighbors of any node through API. API returns a set of nodes and edges. The goal is to construct an algorithm that is capable of collecting sample graph $S(V', E')$, $V' \subseteq V$ and $E' \subseteq E$, where number of nodes obtained $|V'|$ is maximized.

IV. Proposed Method

In this section 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 phases, *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 inside while the connections outside the communities are very sparse.

The concept of the algorithm is shown in Figure 1. For instance, the data collection process starts at some node in a green cluster. As we can see, the network consists of several communities, illustrated by different colors. The Densification phase aims to collect as many (green) nodes as possible. At some point when the algorithm collects most of the green nodes. The algorithm stops the Densification phase and switchs to Expansion. It find a way to escape from green cluster and decides a direction to go to a nearby cluster. In this case, it can choose either blue or purple cluster. Then, it repeats the Densification phase. The algorithm switches between these two phases until it runs out of the budget. A pseudocode in shown in Algorithm 1 and a list of variables in Table I.

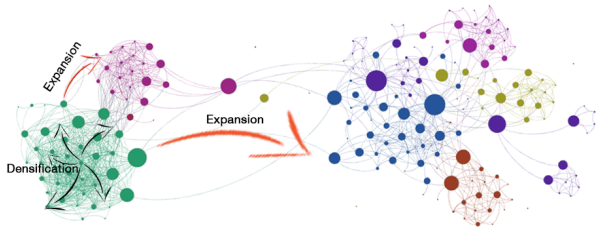


Figure 1: Concept of Expansion-Densification Algorithm

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 *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. Here, we defined two types of nodes, *closed* and *open* node. *Closed node* is a node that we already know all of its neighbors. Basically, 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 runs until it reaches an amount 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}). Finally, S_{sub} 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: $\quad S_s \leftarrow Densification(n_{exp})$s
6: $\quad S \leftarrow$ Merge S_s with S
7: **end while**

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

Table I: Description of each variable

Densifition: We adopt Maximum Observed Degree (MOD) [4] in this phase as it outperforms other algorithms in the same class. The pseudocode is 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}). The Densification stops when the sc_{exp}^t is higher.

Expansion: In the Expansion step, the algorithm wants to

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

escape from the current area of the network. The algorithm selects a node in order to explore on another dense area. In the spirit of explore-exploit algorithm, one naive appoarch is to pick one node uniformly at random from S^o . We refer this Expansion strategy as "$random$".

Switching Phases: Two scores are calculated in each iteration in Densification. Intuitively, a number of closed nodes increase while a number of new nodes decrease over time. The algorithm will find more and more 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 recent request over the number of closed nodes. sc^t_{exp} is a fraction of a number of edges, which one end is a closed node and open node on another end, over number of open nodes in sub-sample. Basically, if the number of edges (e'_{ij}) increases, the number of open node also increase. If not, it means the algorithm already found most of the nodes. These scores give us an approximation of number of nodes left unexplored. Thus, the algorithm can appropiately switch between phases.

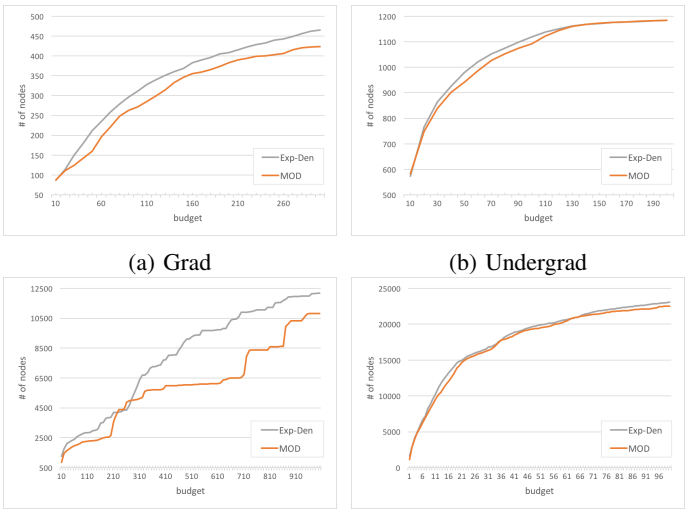
V. Simmulation, Experiment and Discussion

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.

Network	# nodes	# edges	global CC.	Mod.
Grad [5]	503	3256	0.4792	0.6915
Undergrad [5]	1220	43208	0.2980	0.3937
Twitter	12230	50884	0.1117	0.6371
Enron-Email	36692	183831	0.4970	0.5975

Table II: Datasets

We ran 15 experiments on each dataset and ploted the average values in Figure 2. These are the plots between



(a) Grad (b) Undergrad

(c) Twitter (d) Enron-Email

Figure 2: Result of different algorithms on each dataset

amount of budget versus number of nodes obtained. Our algorithm outperforms MOD in every cases. Even a naive approach in Expansion, the result is satisfied. It gives us a strong evidence that Expansion-Densification algorithm is able to collect more nodes than MOD at the same amount of budget. 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 improving a better 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.