SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

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

Katchaguy Areekijseree, Ricky Laishram, Sucheta Soundarajan
Department of EECS, Syracuse University USA.
kareekij, rlaishra, susounda@syr.edu

Problem and Motivation

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.

The 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.
- LinkedIn allows around 1,000 requests for the same interval.

As we can see, data collection is a time consuming task. It takes almost six days to collect all the friends and followers of 8,000 unique users on Twitter.

How should one determine which nodes to query so that the resulting sample is optimal with respect to a desired goal?

Problem Definition

Use case scenario:
- Suppose that we want to obtain data from Twitter.
- We have 24 hours for collecting data.
- We expect to get as many Twitter users as possible.
- Start the data collection process by selecting one known user account.
- Query a list of following or follower users via Twitter API.
- The server returns a list of users. These users are stored in the list.
- Pick a user from the list for next query.
- Repeat the process until run out of time.

Problem Formulation:
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.

Given:
- A starting node - n_{start}
- A number of API requests - *budget*

Goal:
- Collect a sample graph S(V', E') where V'⊆V and E'⊆E.
- Maximize |V'| within a budget.

Overview: Max-Node

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.
- ***Expansion***, which transitions the sampling algorithm to a new region of the graph.

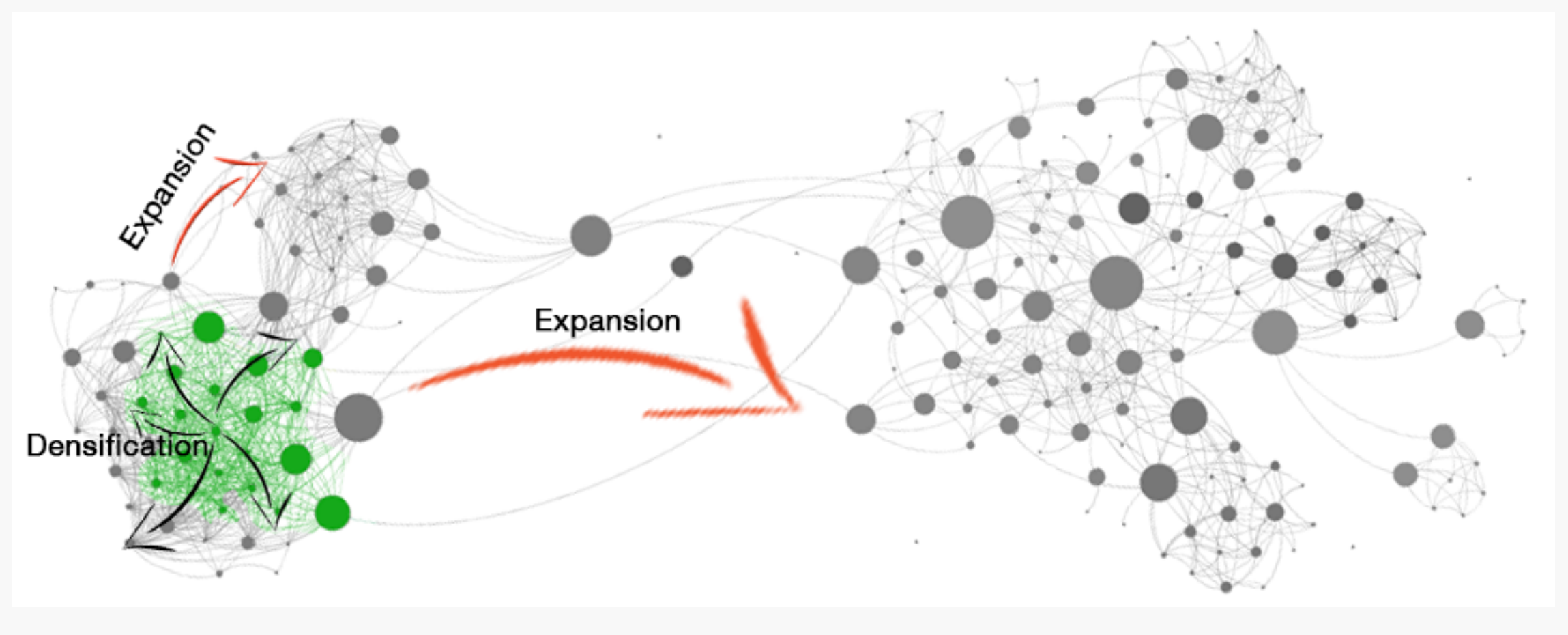


Figure 1. Concept of Expansion-Densification Sampling

Max-Node

Densification:
To expand a sample within a region, we adopt *Maximum Observed Degree* (MOD), as it outperforms other algorithms in the same class.
- In each iteration, the node with maximum degree is selected from S_s^o 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:
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 approach is to pick a node uniformly at random from S^o.
- In our future work, we examine other strategies for Expansion.

Switching Phases:
Intuitively, in each step, the number of closed nodes increases while a number of new nodes added decreases over time (diminishing marginal returns). Two scores are calculated in each iteration of Densification, sc^t_{den} and sc^t_{exp}. These scores give us an approximation of number of nodes left unexplored.
- When $sc^t_{exp} > sc^t_{den}$, the algorithm switches from Densification to Expansion phase.

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.

Note:
- A *closed node* is a node that has already been queried.
- An *open node* is a node that has been observed, but not queried.

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:     $n_{exp} \leftarrow Expansion(S)$
5:     $S_s \leftarrow Densification(n_{exp})$
6:     $S \leftarrow$ Merge $S_s$ with $S$
7: end while
```

Algorithm 2 Densification (n_{cur})

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

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^o_s	a set of open nodes $n \in V_s$
S^c_s	a set of closed nodes $n \in V_s$
e'_{ij}	$e_{ij} \in E_s, (i \in S^c_s \wedge j \in S^o_s) \vee (i \in S^o_s \wedge j \in S^c_s)$
sc^t_{exp}	an expansion score at t^{th} iteration
sc^t_{den}	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

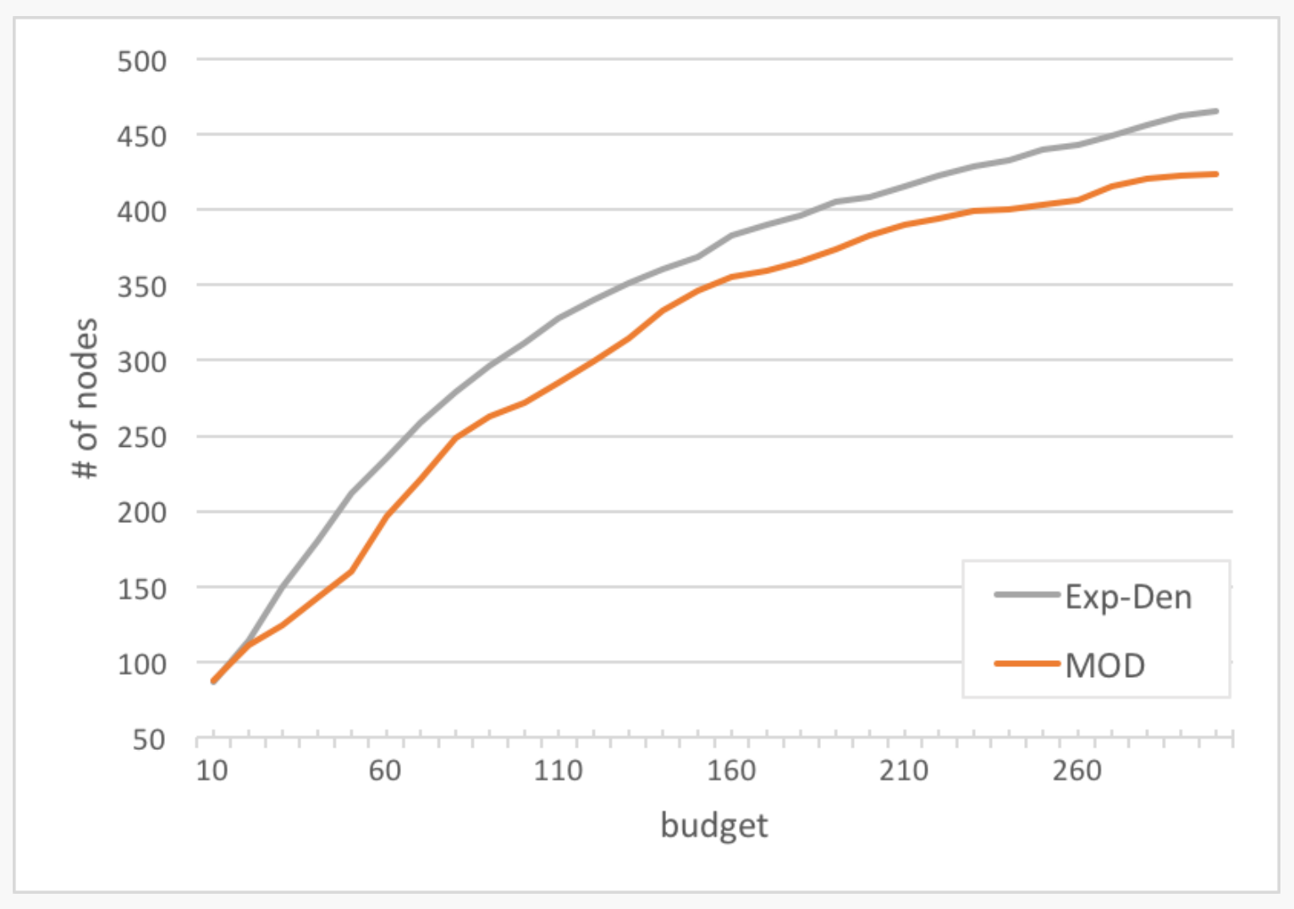
Experiments

To mimic the process of querying an API, we simulate sampling from an existing network dataset. We use four different datasets, described in Table 2.
- *Grad* and *Undergrad* are the Facebook networks.
- *Enron-Email* is an email communication network.
- Twitter is a friend-follower network that we collected via Twitter API.

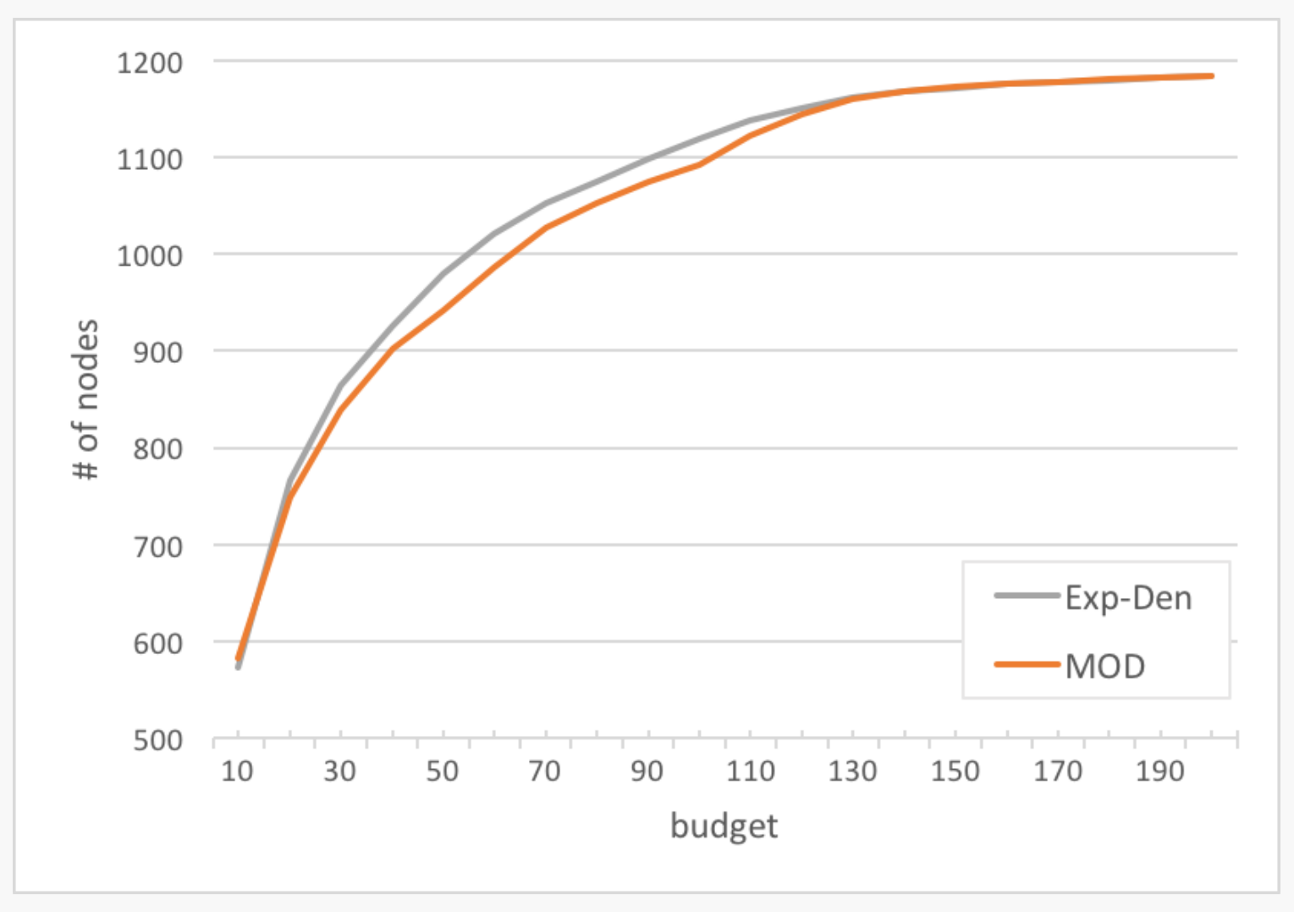
Datasets:

Network	# nodes	# edges	Global CC.	Modularity
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

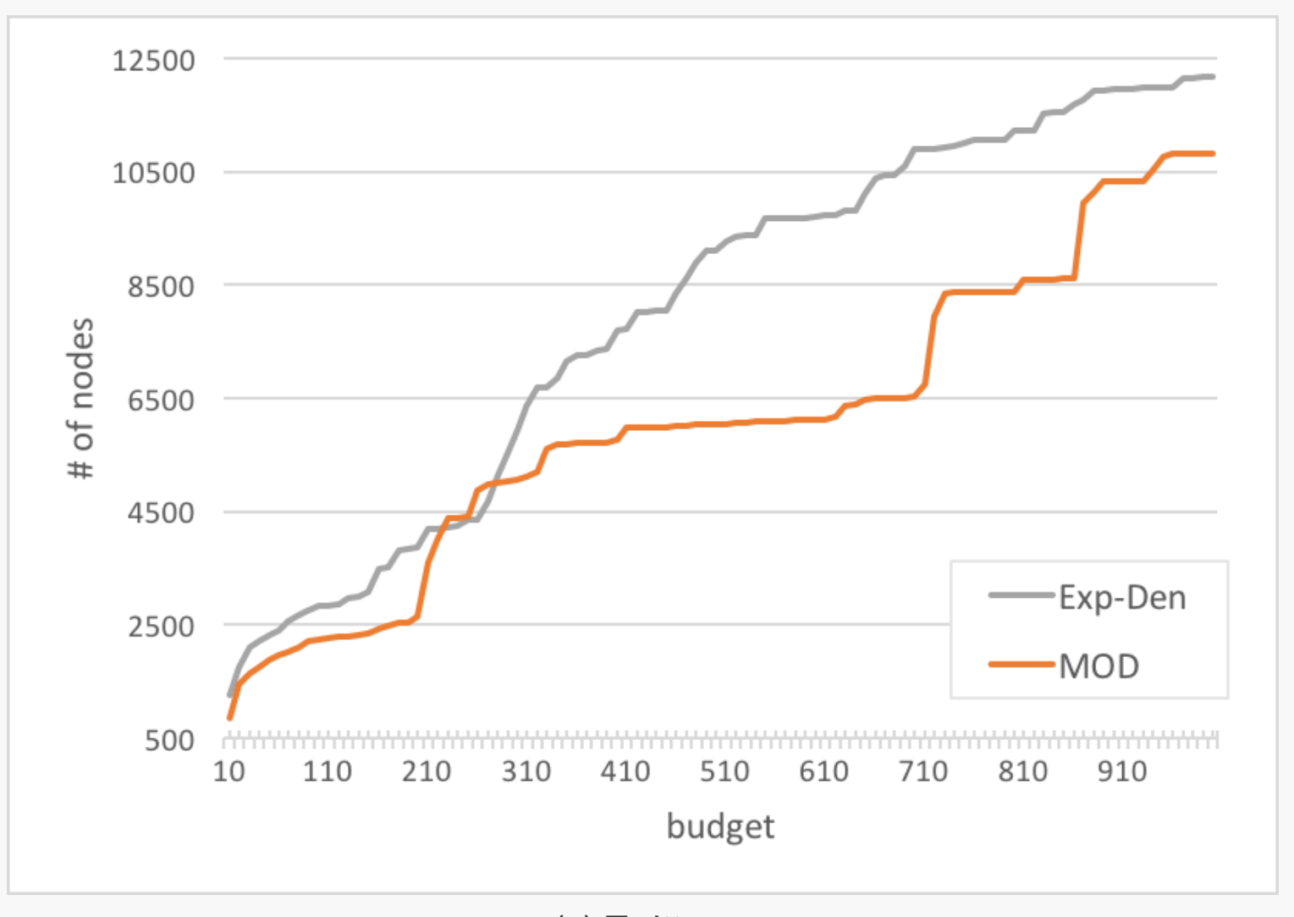
Results



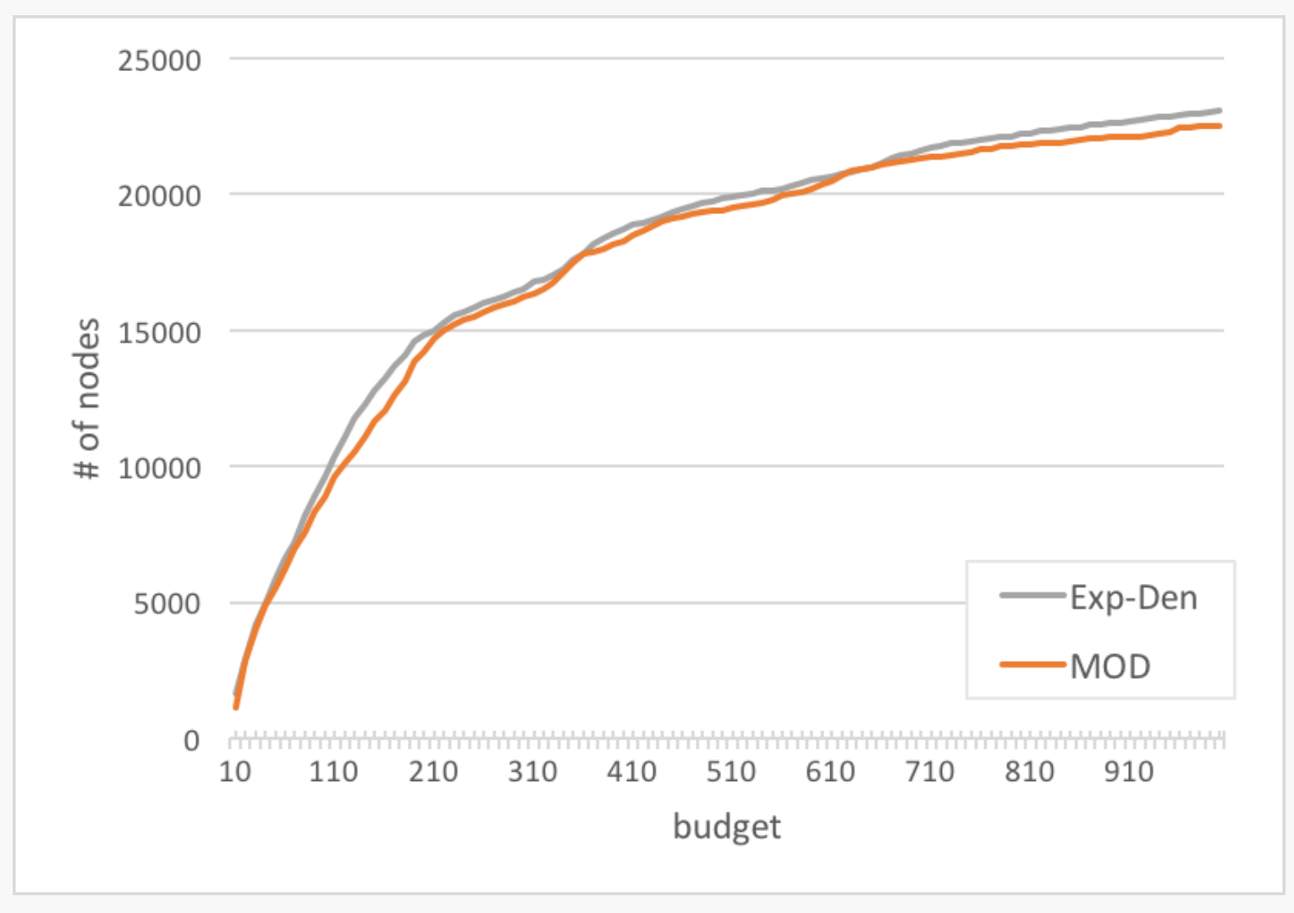
(a) Grad



(b) Undergrad



(c) Twitter



(d) Enron-Email

The results show an improvement of up to 40% vs. the baselines.

Conclusion

This gives us strong evidence that Max-Node algorithm is able to collect more nodes than MOD at the same amount of budget.

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