Collecting Influencers: a Comparative Study of Online Network Crawlers

Mikhail Drobyshevskiy[1,2], Denis Aivazov[1,2], Denis Turdakov[1,3],
Alexander Yatskov[1], Maksim Varlamov[1], and Danil Shayhelislamov[2]
[1]*Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow, Russia*
[2]*Moscow Institute of Physics and Technology (State University), Moscow, Russia*
[3]*Lomonosov Moscow State University, Moscow, Russia*
{drobyshevsky,aivazov,turdakov,yatskov,varlamov}@ispras.ru, shayhelislamov.ds@phystech.edu

***Abstract*—Online network crawling tasks require a lot of efforts for the researchers to collect the data. One of them is identification of important nodes, which has many applications starting from viral marketing to the prevention of disease spread. Various crawling algorithms has been suggested but their efficiency is not studied well. In this paper we compared six known crawlers on the task of collecting the fraction of the most influential nodes of graph.**

We analyzed crawlers behavior for four measures of node influence: node degree, k-coreness, betweenness centrality, and eccentricity. The experiments confirmed that greedy methods perform the best in many settings, but the cases exist when they are very inefficient.

***Index Terms*—Network crawling, network sampling, node influence**

I. Introduction

Today there is a growing interest in network data collecting from online sources. Social networks, such as Facebook and Twitter, often provide APIs which help researchers obtain the data. However, several challenges arise here. A large scale of real-world graphs requires a huge amount of resources to crawl them due to bandwidth limits, many sites impose query limitations, etc. Goals of network sampling could be different. Beside collecting the whole graph itself, one is interested in sampling a representative subgraph to use it instead of the original one, estimating network parameters, or estimating node/edge attributes [1].

In many applications it is not necessary to crawl all nodes of a network but its most influential nodes only. Efficient identification of top-k influential nodes is important for detecting key persons in social networks, preventing disease spread, controlling computer worms, and so on. Node influence is associated with its centrality measure in the graph. For instance, node with high betweenness centrality is likely to have a high impact on information spread in a social network or a contagion process in a biological network.

A good network crawler should discover the highest centrality nodes with a minimal number of steps. Usually, a significant fraction of target nodes can be collected in relatively few iterations, especially when the degree distribution is skewed. For example, 5% of nodes being sampled via random walk, cover 80% of the k largest degree nodes [2].

The study was funded by RFBR according to research project 18-07-01211

Literature contains a number of crawling algorithms, while their efficiency depends on multiple factors. The choice of seed node, presence of "protected" users (whose connections can be detected only by incoming links from other observed nodes) [3], and network structure [4] significantly influence the result performance of each method.

In this paper we compare several popular crawlers on the task of collecting a target set of top-10% influential nodes of the graph. The distinctive feature of our study is that we run a crawler until it collects the whole graph, while crawlers are usually analyzed under a limited budget of queries. Our main contributions are the following ones.

1) The decision on the choice of crawling method significantly depends on the given budget. We observed that while a graph is being crawled, the leading algorithm can change several times.
2) Greedy methods, like the one guided by the maximal observed degree (MOD), are better than others in collecting the fraction of nodes with highest degrees, highest k-coreness, and highest betweenness centrality. Nodes with the least eccentricity, in comparison to other centralities, are harder to find for existing crawlers.
3) We confirm that MOD is often the best choice for network crawling, but there exist cases, when it loses to all other algorithms. The same conclusion holds for influential nodes crawling task.

In the next section we formalize the task and describe our experimental methodology. Section III is devoted to the experimental results and their explanation suggestions. Then we provide a brief overview of related works in section IV and, finally, give a conclusion in section V.

II. Problem definition and methodology

A. Problem Definition

In our work, we consider a static unobserved undirected network, represented with a graph $G(V, E)$, where V is the set of nodes and E is the set of edges. A crawler starts with a seed node $v_{seed} \in V$. Two sets, initially empty, are defined and dynamically updated at each step. $V'_c \subseteq V$ is a set of already closed (queried) nodes. $V'_o \subseteq V$ is a set of observed, but not closed nodes. At each iteration, the crawler queries

the next node $v \in V'_o$, which becomes closed, and updates V'_o with newly seen neighbours of v in G. Sampled graph $S_i = (V', E')$ at iteration i consists of all closed and observed nodes $V' = V'_c \cup V'_o$ and all connections between them $E' \subseteq E$. We denote as $deg(v, S)$ the degree of node v within graph S (since S is a sample of V, $deg(v, S) \leq deg(v, V)$) and as $clust(v, S)$ its clustering coefficient.

The goal of the crawler is to cover a target set of most influential nodes $V^* \subseteq V$ as soon as possible. We consider four different measures of node influence: the degree, k-coreness, betweenness centrality, and eccentricity.

- Node *degree* is the most straightforward measure of importance as the number of friends, subscribers, connections, citations, etc.
- *Betweenness centrality* characterizes how many paths in graph go through the node. High betweenness means high influence on information flows.
- Node *k-coreness* indicates that the node is a part of a connected subgraph where all nodes have degree at least k.
- Node *eccentricity* measures the maximal distance to any other node in the graph. The lower the eccentricity, the faster information/disease spreads from the node to the rest of the graph.

We take 10% top-scored nodes of the graph as a target set in all our experiments. Note that there are 4 different although overlapping target sets, one per each centrality measure. For example, Figure 3 (left) shows 3 target sets from slashdot graph. The intersection of degree and k-coreness is significant, while the eccentricity set has about a half in common with them.

B. Crawlers

In our work, we considered 5 most popular crawling methods (RC, RW, DFS, BFS, MOD) and a recently proposed DE-Crawler. Table I summarizes the used algorithms together with their computational complexities per one iteration (in our implementations). Random Crawler (RC) and Random Walk (RW) algorithms each time go to a random node, selected from the whole observed set V'_o, or from the newly observed neighbours of the previously crawled node, respectively. RW is known to be effective at discovering top-centrality nodes [2]. BFS and DFS implement two well-known search strategies. BFS crawler is usually applied for network analysis [5], although partial BFS crawls are biased towards high-degree nodes and underestimate low-degree nodes [6].

MOD is a greedy method, based on a heuristic that a node v with high observed degree $deg(v, S)$ also has high real degree $deg(v, G)$. This proved to be efficient in terms of node coverage [7].

DE-Crawler was recently suggested by K. Areekijseree and S. Soundarajan [8] as a smart combination of RW and a greedy algorithm. As it was shown in their previous work [4], while MOD outperforms other methods, it gets stuck within communities in graphs with high modularity. At the same time, walking-based methods are good to move between dense regions of the network. DE-Crawler algorithm consists of two main stages: Densification and Expansion. These stages are switched, depending on the result of comparison of certain statistics. At each crawling step, V'_o is sorted by $deg(v, S)$. In the Expansion mode, the next node to crawl is randomly selected from 80% of bottom elements. In the Densificatoin mode, for the top 20% a score $\Phi(v)$ is calculated: $\Phi(v) = \frac{deg(v,S)}{\langle deg(v,S)\rangle}(1 - clust(v, S))$. The next node is the one with a maximal score, which is motivated by the observation that hubs have high degrees and low clustering coefficients [9].

C. Dataset

For the experiments, we collected a dataset of small and medium-size networks from various domains. All graphs, except DCAM, are available at *http://networkrepository.com* or at The Koblenz Network Collection [10]. DCAM was manually crawled from one community (*vk.com/club1694*) by API and contains only open profiles. Since all considered crawlers are designed to operate with connected graphs only, we extracted a giant component from each graph and further analyzed it instead of the whole graph. Short descriptions and parameters of graphs can be found in Table II.

D. Method

We tested the crawlers from Table I on all graphs from the dataset (Table II). As it was mentioned in section II-A, a crawler starts with a randomly chosen seed node and traverses the graph, updating V'_c and V'_o sets. It stops when the whole graph is collected, i. e. $S = G$, $V'_c = V$, and $V'_o = \emptyset$. In order to avoid bias of the seed choice, all results in the next section were averaged over 8 different seeds uniformly chosen from V.

To evaluate the crawling algorithms, we used a classical measure, node coverage $c^{nodes} = |V'|/|V|$. We considered two ways to measure the coverage of a target set. One can count the coverage for all already known nodes, $c^{target} = |V' \cap V^*|/|V^*|$ or only for the closed ones: $c_c^{target} = |V'_c \cap V^*|/|V^*|$. The motivation for the second approach is that one is usually interested in how many influential nodes are collected rather than seen.

The target set V^* is formed by the top $p = 10\%$ of nodes sorted by one of four measures: node degree, betweenness centrality, k-coreness and eccentricity. For eccentricity we took the top nodes with the lowest value. For all cases we measured the coverage c depending on the number of nodes crawled i, iterating i from 1 to $|V|$.

III. Experiments

For the experimental evaluation we implemented a framework with all six crawler methods: RC, RW, DFS, BFS, MOD, and DE. To the best of our knowledge, no public implementation is available, therefore we used our own. The computational complexities of one iteration of each algorithm are presented in Table I.

In all methods at each step we observe neighbours of the current node and add them to V'_o. This operation gives the lower bound of complexity $O(\langle deg(v, G)\rangle)$.

TABLE I

NETWORK CRAWLING ALGORITHMS WITH A SHORT DESCRIPTION AND COMPUTATION COMPLEXITY PER ONE ITERATION.

Name	Method	Node selection strategy	1 step complexity
RC	Random Crawling	At each step, selects a random node from V_o is selected. Does not depend on previously crawled node.	$O(\langle deg(v,G)\rangle)$
RW	Random Walk	Random neighbour of previously crawled node	$\gtrapprox O(\langle deg(v,G)\rangle)$
DFS	Depth First Search	Traverses the graph in depth-first manner	$O(\langle deg(v,G)\rangle)$
BFS	Breadth First Search	Traverses the graph in breadth-first manner	$O(\langle deg(v,G)\rangle)$
MOD	Maximum Observed Degree	At each step selects a node from V'_o with maximal degree	$O(\log \lvert V'_o\rvert \cdot \langle deg(v,S)\rangle)$
DE	Densification-Expansion crawler	Switches between RW (expansion phase) and MOD analogue (Densification phase) strategies depending on statistics of the sampled graph [4].	$O(\lvert V'_o\rvert \cdot \langle deg(v,S)^2\rangle)$

TABLE II

DATASET OF UNDIRECTED NETWORKS. ALL PARAMETERS CORRESPOND TO THE GIANT COMPONENT USED IN EXPERIMENTS.

Name	Description	$\lvert V\rvert$	$\lvert E\rvert$	$\langle deg(v,G)\rangle$
hamsterster	friendship graph of Hamsterster	2 000	16 097	16
DCAM	community subgraph from VKontakte	2 752	68 741	50
facebook	contains friendship data of Facebook users (2009)	63 392	816 886	26
slashdot	reply network of technology website Slashdot	51 083	131 175	5.1
github	membership network of the software development hosting site Github	120 865	439 858	7.3
dblp2010	co-authorship network	226 413	716 460	6.3

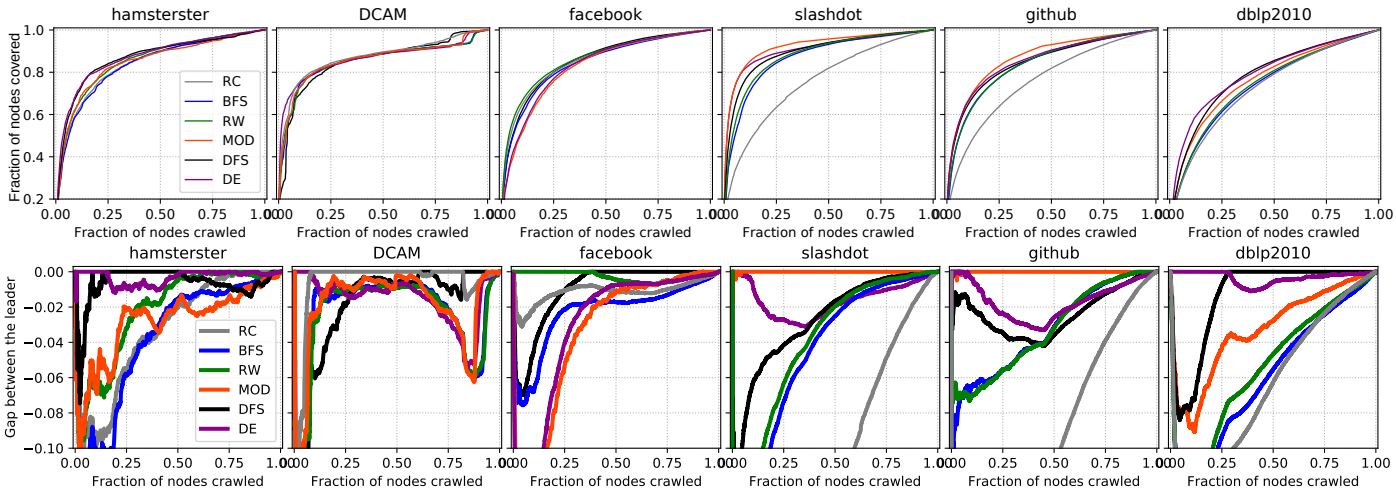

Fig. 1. Top: node coverage $c^{nodes} = |V'|/|V|$. Bottom: same results, the gap between current method and the best result at each point (i.e. the higher the better) for node coverage.

Random Crawler just takes a random node from V'_o which costs O(1). Random Walk crawler each time goes to a friend of the current node, which requires O(1). However, it could surf among already closed nodes for an uncertain time until finds a node from V'_o.

BFS and DFS algorithms require to keep the queue and stack, respectively. Getting a new node from those requires $O(1)$.

The MOD method at each step requires a node from V'_o with the maximal degree. We used sorted lists in which every node is sorted by its degree to speed up our implementation. One step complexity is $(2 \cdot deg(v,S)+1) \cdot \log |V'_o|$. So choosing the next node at each step became much more complex operation, because $deg(v,S)$ increases as size of S increases.

Finally, DE is the most computationally complex algorithm. At each step, for each node in the top 20% by degree in $|V'_o|$, it calculates statistics involving the clustering coefficient with complexity $deg(v,S)^2$. After that, it picks the node with the highest clustering coefficient as the next node. To improve the sorting step we also used sorted lists. The complexity to keep that list sorted at each step is $(2 \cdot deg(v,S)+1) \cdot O(\log |V'_o|)$. The statistics require to find the average degree in $|V'_o|$ for calculating coefficients that decide on the mode switching. So, its total complexity is $O(|V'_o|) \cdot (\langle deg(v,S)^2\rangle + 1) + O(\log |V'_o|) \cdot (2 \cdot deg(v,S)+1) \approx O(|V'_o| \cdot \langle deg(v,S)^2\rangle)$, and it is much higher than that of all others.

A. Nodes coverage

We measured the node coverage $c^{nodes} = |V'|/|V|$ for all 6 crawler methods on 6 graphs. Results are presented at Figure 1. The X-axis corresponds to the fraction of crawled nodes, from

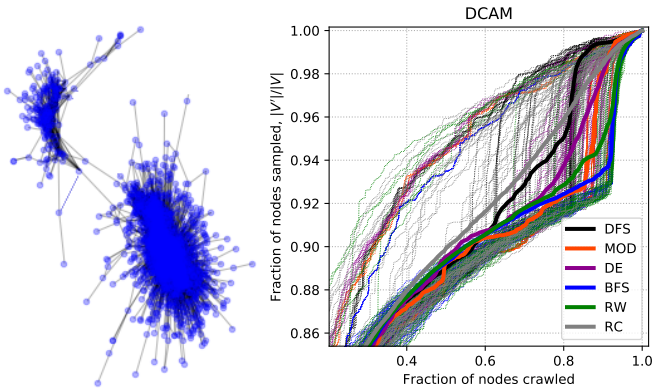

Fig. 2. DCAM graph. Left: network structure. Right: node coverage with variation for several seeds. Dotted liens correspond to individual seeds, bold lines correspond to averaged values.

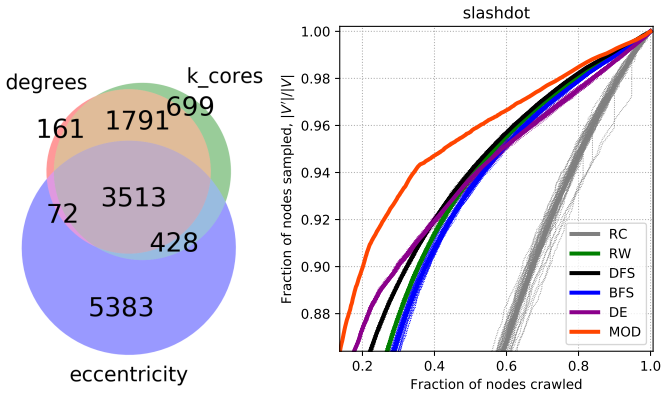

Fig. 3. Slashdot graph. Left: Venn diagram for top centrality nodes sets. Right: node coverage with variation for several seeds. Dotted liens correspond to individual seeds, bold lines correspond to averaged values.

0 to the 1 (the whole graph). The Y-axis shows a fraction of $|V'|$ to $|V|$. We also plot the gap between current method and the best result at each point, i.e. the higher the better (bottom plot at Figure 1). Such a visualisation could help to see which method is the leader. The main result is that the leader changes depending on the number of nodes crawled (budget size), for all graphs. If the budget is limited to 5-10% of $|V|$, DE crawler outperforms the others at hamsterster, DCAM, github, and dblp2010. But when the task is to collect the majority of the graph, DE crawler is not the optimal choice. For github and slashdot that would be MOD, while for dblp2010 and facebook — DFS is the best choice. Results on small graphs hamsterster and DCAM are not so stable as on larger graphs: the leader changes several times depending on the budget.

Another observation concerns crawling curves on DCAM graph. When about 80-90% of nodes are crawled, BFS, RW, and MOD curves demonstrate a hop (see Figure 2, right). The results are averaged over 50 random seeds; the averaged curve is plotted in bold, individual seeds are dotted lines. The hopping behavior happens due to a specific structure of DCAM network. It consists of two well separated communities connected with a small bridge (Figure 2, left). MOD crawler is known to get stuck within communities with high modularity, while RW was reported to be able to transition between them [4]. However, we see that in average RW also gets stuck within the community. DE crawler has a smoother curve and outperforms MOD, proving its ability to escape the community. Finally, according to this DCAM experiment, RC and DFS strategies are the best strategies for getting outside a dense community.

1) Seed choice influence: Although an initial seed is expected to affect the crawling process, its influence is negligible at larger graphs. For example, at DCAM graph with 2.7K nodes, the seed choice could have a significant affect to crawler performance (see variability of dotted curves at Figure 2, right). But for slashdot graph with 51K nodes, results are already little dependent on the seed choice (see Figure 3, right). Nevertheless, it should be noted that for slashdot at early crawling stage, when budget is comparable to the size of DCAM, a similar high variability is observed. The same holds for larger graphs from the dataset.

B. Influential nodes coverage

We measured the target set coverage for 6 crawler methods on 6 graphs of our dataset. The results of the node coverage are presented in Figure 1, the target set coverage for 4 measures is shown at Figure 4 (top). We show only c_c^{target} measure results since it reflects a more realistic picture of collecting most influential nodes. The other reason is that for c^{target} measure, method lines goes too close to each other on the plots to distinguish them.

The lower plot of Figure 4 shows the gap between current method and the best result at each point, i.e. the higher the better.

The first thing one can see is that the results vary a lot for different measures at different graphs. RC often resembles a straight line but is convex at bigger graphs. This means that even such a simple strategy of randomly selecting one of the observed nodes works better than if we would pick a random node of the graph (which would give a straight line on the plot).

1) Degree and k-coreness: For collecting top-degree nodes, as one could expect, greedy algorithms MOD and DE are better than the others. Comparing MOD and DE (see lower plot of Figure 4), one can see that DE does not outperform MOD. Moreover, MOD outperforms DE at slashdot, github, and dblp2010, while DE is strongly better only at facebook.

Quite a similar picture one can see for k-cores centrality at first 5 graphs. For both centralities, MOD and DE are leaders, RC performs the worst, while RW, BFS, and DFS are in the middle almost all the time.

A surprising behavior one can see at dblp2010, where DE is much worse. We assume that the reasons are the unusual topological structure (connected stars) of the dblp2010 graph and the default coefficients in DE statistics formulas. Switching between densification and expansion modes did not occur at the moment when it was intended. It is likely that a

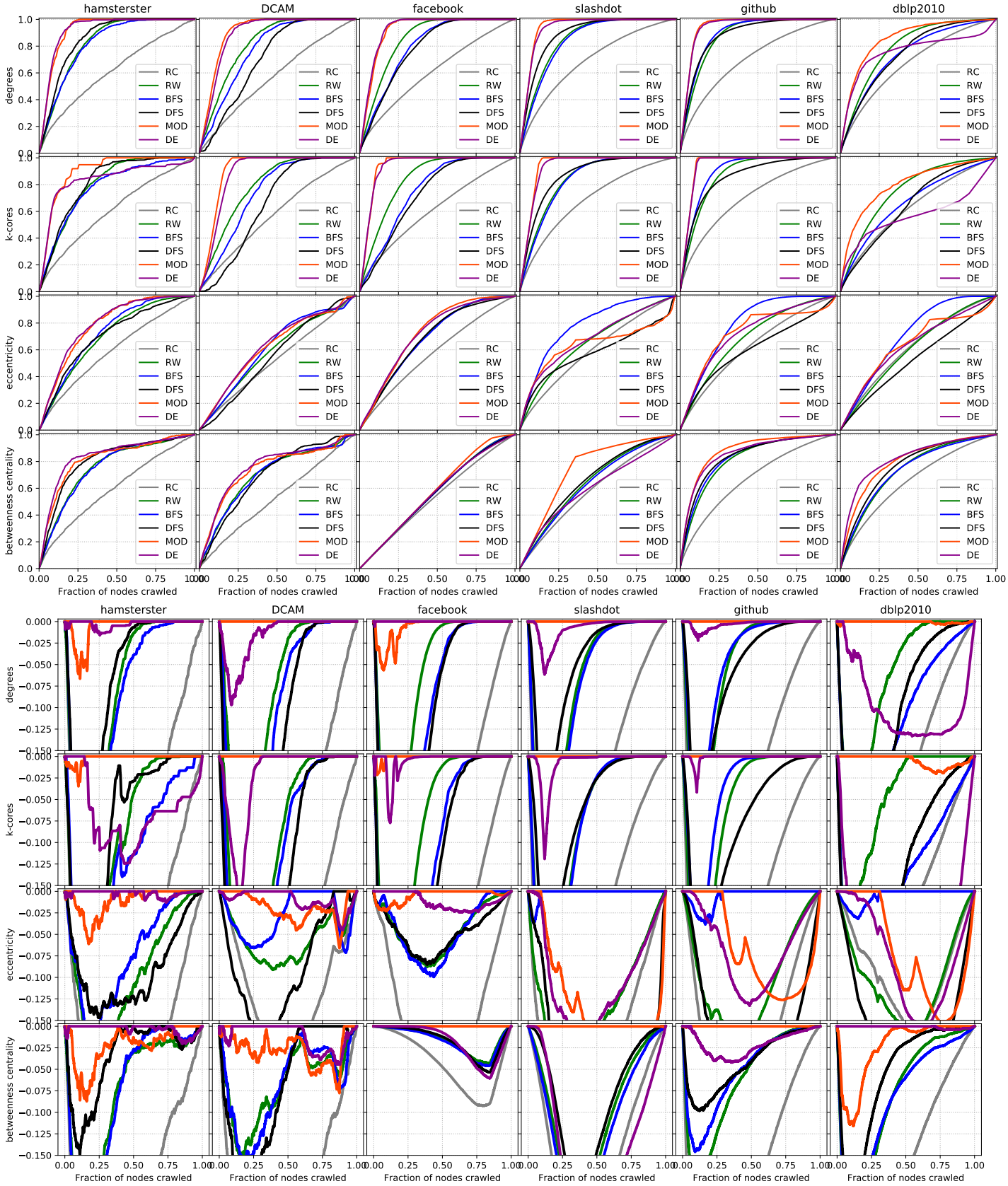

Fig. 4. Target set coverage $c_c^{target} = |V_c' \cap V^*|/|V^*|$ depending on the fraction of nodes crawled. The lower plot shows the gap between current method and the best result at each point (i.e. the higher the better).

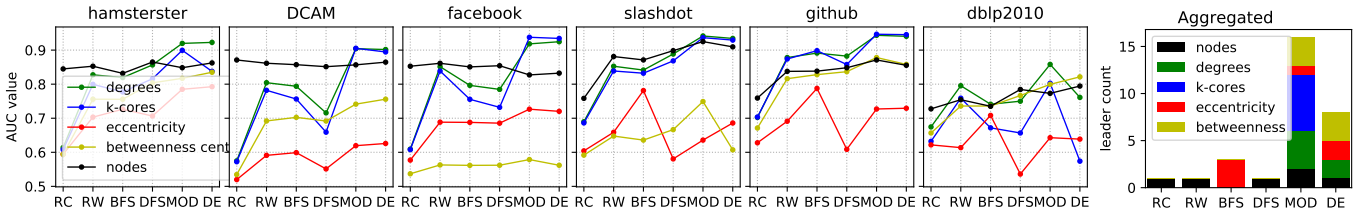

Fig. 5. AUC values computed for node coverage c^{nodes} and target set coverage $c_c^{target} = |V_c' \cap V^*|/|V^*|$. Right: winners aggregated over the graphs. Colored bars' heights denote how many times this crawler was the best one at specific measures.

more accurate tuning of these coefficients is needed to make DE-crawler behave as expected by its authors.

2) Eccentricity: A different result we observed for eccentricity measure. For slashdot, github, and dblp2010, the BFS strategy significantly outperforms the others. Moreover, MOD loses its leading position after 10-20% of steps and then loses all other methods.

Interestingly, its coverage curve is not smooth and have several turning points. We believe MOD eventually gets stuck within communities composed of vertices with high eccentricity. For any vertex, difference of its eccentricity and eccentricities of neighbouring vertices does not exceed one. Therefore, if a vertex has high eccentricity, its neighbours also have high eccentricity, and vice versa, if a vertex belongs to target set of vertices with lowest eccentricity, its neighbours probably belong there too. Thus there are communities with lots of vertices from target set and there are communities with none of them. When MOD gets stuck in community with high eccentricity, its coverage curve goes flat. Finally, for these three graphs the DFS algorithm also works bad, even worse than RC for most of the time.

3) Aggregated results: We also computed area under curve (AUC) for the results mentioned before, in order to compare crawlers performance independently of budget (Figure 5, left). One can see that in most cases low eccentricity nodes are detected worse than other centralities. Results for top degree nodes and top k-core nodes correlate well, since these measures are related. This is also consistent with Venn diagram (Figure 3, left) showing a significant overlap between the corresponding sets of nodes.

We aggregated AUC results over all graphs, counting how many times each crawler was the best one for a specific measure (Figure 5, right). MOD appeared to get the highest score of 16, two times more than DE. MOD is also the best for k-core nodes collecting for all 6 graphs. BFS is good at lowest eccentricity nodes crawling, possibly due to very bad results of MOD and DE on slashdot, github, and dblp2010 graphs. However, this observation needs a more detailed analysis. Finally, note that results on node coverage are the most uncertain: 5 of 6 methods were the winner at least once.

IV. Related work

There are many studies in literature on the network crawling problem. For a detailed review, please refer to a survey [1].

Ye Shaozhi and co-authors in [3] investigate the network crawling problem on four social networks: Orkut, Youtube, Live Journal and Flickr. They analyze several crawlers depending on seed choice, network structure, and presence of protected profiles. Among their findings are the following ones. A small number of steps is enough to obtain a large part of the target graph: 10% of nodes being crawled provide 49–74% of nodes of the graph. Node/link coverage does not depend much on the choice of seeds: by choosing high degree seeds the improvement did not exceed 5%. Greedy algorithms, like MOD, achieve high node/link coverage faster, but are less robust than BFS.

Several works aim at sampling top centrality nodes. Yeonsup Lim et al. in their work [2] try to estimate nodes with top-k centrality, namely degree, betweenness, and closeness. Experiments on various networks show that RW crawler quickly discovers a major fraction of the top-degree nodes: 5% of the sampled nodes contain over 80% of the top-10. Node degree correlates well with other centralities and thus could be used as an approximation for them. For betweenness and closeness measures, RW strategy outperforms more complex strategies. Two kinds of error are introduced, sampling (collection) error and identification error. A sampling error means that a node from the top-k is missing in the sample, while an identification error occurs when the correct node is sampled but not recognized as such. The authors discover that sampling error is higher when the network has slightly skewed degree distribution. When the degree centrality has low correlation to betweenness and closeness in a network, identification error become larger.

A challenging task could be to detect top-k centrality nodes in a known large graph, for example, in a crawled sample. Since the calculation of betweenness and closeness centralities requires search for all-pairs shortest paths, which complexity is $O(|V||E|)$ for an undirected graph, their exact computation could be too expensive. A series of works employ compressive sensing approach to detect top centrality nodes, e.g. [11]–[14]. The idea is to use a part of the known graph for estimations based on compressive sensing theory. K. Avrachenkov, N. Litvak, and coauthors suggested a sublinear algorithm to finding list of nodes with maximal degree [15]. A random walk based approach allows to approximately find such nodes faster than in $O(|V|)$ iterations.

V. Conclusion

We tested six network crawlers on several graphs from various domains. 5 crawlers are well known, RC, RW, BFS, DFS, MOD, and 1 more DE-crawler was proposed recently. We run the algorithms until they collect the whole graph and measured the coverage of a target set of top-10% influential nodes. We obtained the following results.

There is no superior crawling method neither for classical nodes coverage measure, nor for influential nodes coverage. Furthermore, the leading algorithm for a particular graph can change several times depending on the fraction of graph nodes is collected. MOD, DE, RW, BFS, and DFS methods all have been the leader for several times in our experiments. This is in contrast with result obtained by running crawlers with a limited budget, e.g. [4], [8]. Our results on a whole graph imply that the leading crawler do change depending on the budget.

Greedy strategy MOD is often the optimal one, but not always, which is consistent with known results from literature. This is also true for collecting of influential nodes. For example, at slashdot, github, and dblp2010 graphs, MOD outperforms the others at lowest eccentricity nodes crawling at early stages (until 10-25% of nodes of graph are collected). But further it loses to all other considered methods. Surprisingly, BFS showed the best coverage of lowest eccentricity nodes.

The previous conclusion also holds for DE crawler. Moreover, experiments do not evidence that DE outperforms MOD, there are cases when it loses to many other methods. Taking into consideration its high computational complexity, it is not currently a good choice. Probably, a more accurate parameters tuning could improve its performance.

Concerning the task of collecting centrality measures, MOD is the best one for k-coreness. MOD and DE are better than the other 4 methods in collecting top degree and betweenness centrality nodes. As for lowest eccentricity nodes, the best method could not be reliably defined. All 6 crawlers detect this kind of nodes worse (relatively slower) than the other types of influential nodes.

In one experiment with DCAM graph consisting of two well separated communities, we observed that RC and DFS are less prone to getting stuck within highly modular communities, than RW, MOD, DE, and BFS.

The influence of the seed choice is not crucial for comparing crawlers, when graph size is more than 50K nodes. This confirms the results of Ye Shaozhi et al [3].

References

[1] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 2, p. 7, 2014.

[2] Y.-s. Lim, D. S. Menasché, B. Ribeiro, D. Towsley, and P. Basu, "Online estimating the k central nodes of a network," in *2011 IEEE Network Science Workshop*. IEEE, 2011, pp. 118–122.

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

[4] K. Areekijseree, R. Laishram, and S. Soundarajan, "Guidelines for online network crawling: A study of data collection approaches and network properties," in *Proceedings of the 10th ACM Conference on Web Science*. ACM, 2018, pp. 57–66.

[5] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 29–42.

[6] S. H. Lee, P.-J. Kim, and H. Jeong, "Statistical properties of sampled networks," *Physical review E*, vol. 73, no. 1, p. 016102, 2006.

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

[8] K. Areekijseree and S. Soundarajan, "De-crawler: A densification-expansion algorithm for online data collection," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 164–169.

[9] M. Bloznelis *et al.*, "Degree and clustering coefficient in sparse random intersection graphs," *The Annals of Applied Probability*, vol. 23, no. 3, pp. 1254–1289, 2013.

[10] J. Kunegis, "Konect: the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 1343–1350.

[11] H. Mahyar, "Detection of top-k central nodes in social networks: a compressive sensing approach," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM, 2015, pp. 902–909.

[12] H. Mahyar, R. Hasheminezhad, E. Ghalebi, A. Nazemian, R. Grosu, A. Movaghar, and H. R. Rabiee, "Identifying central nodes for information flow in social networks using compressive sensing," *Social Network Analysis and Mining*, vol. 8, no. 1, p. 33, 2018.

[13] ——, "Compressive sensing of high betweenness centrality nodes in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 497, pp. 166–184, 2018.

[14] H. Mahyar, R. Hasheminezhad, and H. E. Stanley, "Compressive closeness in networks," *arXiv preprint arXiv:1906.08335*, 2019.

[15] K. Avrachenkov, N. Litvak, M. Sokol, and D. Towsley, "Quick detection of nodes with large degrees," *Internet Mathematics*, vol. 10, no. 1-2, pp. 1–19, 2014.