DE-Crawler: A Densification-Expansion Algorithm for Online Data Collection

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

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

***Abstract*—Over the past two decades, online social networks have attracted a great deal of attention from researchers. However, before one can gain insight into the behavior or structure of a network, one must first collect appropriate data. Data collection poses several challenges, such as API or bandwidth limits, which require the data collector to carefully consider which queries to make. Many network crawling methods have been proposed; however, their performance depends on network structure. In particular, our previous work in [1] has shown that existing algorithms tend to either (1) Do well at exploring dense areas of a network, but have difficulty in transitioning to new areas of the network, or (2) Easily move between network regions, but fail to fully explore each region. In this work, we introduce DE-Crawler, a novel network crawler that attempts to capture the best of both worlds. DE-Crawler consists of two main stages: *Densification*, in which the crawler aims to find as many nodes as possible in the current dense region (or community), and *Expansion*, in which the crawler tries to escape from its current region and move to another dense region. We show that DE-Crawler performs well across networks with different structural properties, outperforming baseline algorithms by up to 28%.**

1. Introduction

Complex networks, including online social networking sites, the WWW, and communication networks, have gained a great deal of attention from the researchers in a wide range of fields. By studying these networks, we are able to gain deep insight into societal-scale human behavior, such as understanding how groups form, information spreads, or friendships dissolve. However, before one can attempt to answer these questions, one must first collect network data.

Currently, many researchers choose to collect data from Online Social Networking Sites (OSNs), such as Twitter or Facebook, because these sites provide a convenient accessible channel (e.g. API) that allows anyone to obtain the data. However, collecting data though the API may require a significant amount of time or money. While it is possible to purchase faster access, these costs can be prohibitively large. Collecting appropriate data while staying within resource constraints (such as time limitations) can thus be challenging, and one must be careful to use their limited resources in a way that maximizes the quality of data that they obtain.

In this paper, we consider the problem of **online network crawling**, otherwise known as **network sampling through crawling**. To crawl a network, a crawler starts with a single observed node, and has no other information about the underlying network. To obtain information, the crawler performs a series of queries through the API. Here, we assume here that a list of *all* the neighbors (e.g. friends) of the queried node are returned in response to the query. These returned nodes are now considered to be observed, and in each step, the crawler queries a node that was observed but not queried in order to expand the sampled network and learn more information about the network. This process is repeated until the budget (e.g., time or money) are exhausted.

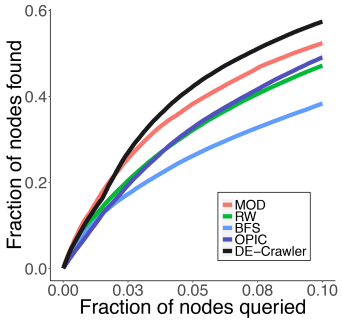


Figure 1: Performance of **DE-Crawler** vs. baseline methods on DBLP-2012, at the task of maximizing node coverage with a maximum query budget of 10% of the total number of nodes. **DE-Crawler** outperforms all the baselines.

When collecting data, there may be many crawling goals of interest, such as finding samples that are unbiased with respect to some property, locating 'important' nodes, or finding a sample that preserves information flow patterns. In this work, we focus on maximizing the total number of nodes observed: i.e., "*node coverage*". We selected this goal because it is closely tied to many important applications, including crawling for census-type applications [2], crawling to preserve communities [3] and crawling to estimate node centrality [4].

The literature contains a large number of proposed crawling algorithms for this task. Some of these methods perform well, but they are rarely consistent across network structures. In our previous work [1], we found that network structural properties have a strong effect on the algorithms performance.

IEEE/ACM ASONAM 2018, August 28-31, 2018, Barcelona, Spain
978-1-5386-6051-5/18/$31.00

More specifically, strong community structure can obstruct a crawler from being able to move from one region to another region of the network. As a result, greedy approaches like `Maximum Observed Degree` (MOD) [5] works very well when the communities overlap with one another, but perform poorly when there are clear borders between regions.

To tackle this problem, we propose a novel crawling algorithm, **DE-Crawler**, in which the crawler can seamlessly transition to different regions of the network. It consists of two main stages: *Densification* and *Expansion*. The *Densification* stage aims to discover nodes in the current region, while the *Expansion* stage aims to expand the sampled network by moving to another dense region. **DE-Crawler** is able to capture the best aspects of the existing algorithms, and achieves outstanding performance across domains. As an example, see Figure 1 for a result on the DBLP-2012 citation network on the node coverage task. **DE-Crawler** performs better than the baseline method; and moreover, we see this behavior regardless of network structure. The main contributions of this paper are:

1) We present **DE-Crawler**, a novel crawling method, for the task of maximizing node coverage with a fixed query budget. Our experimental results show that **DE-Crawler** outperforms baseline methods by up to 28%.

2) We perform an extensive experimental analysis on networks from diverse categories, including collaboration, Facebook, OSNs, the WWW and technological networks. We show that **DE-Crawler** consistently performs well across different networks types and structures.

2. Related Works

The literature on network sampling can be roughly separated into 1) *down-sampling* and 2) *sampling through crawling*. With down-sampling, one possesses the complete network dataset, and wishes to scale it down to some desired size (e.g. the entire dataset is too large to fit into memory). The sample should maintain the relevant properties and characteristics of the original network. In the crawling scenario, one begins with no information about the network other than knowledge of a single node. We can obtain additional information by performing queries on observed nodes, so the observed sample is expanded from the single initially observed node. The decision on which node to query next is based on the network structure observed so far.

Leskovec and Faloutsos study the characteristics of different algorithms under the down-sampling scenario [6]. They study and evaluate the algorithms based on how well the sampled graph maintains properties of the original graph, and conclude that a Random Walk method is best. Maiya and Berger-Wolf present a down-sampling algorithm that aims to preseve the community structure [3]. The idea is to select the node with the most neighbors outside the current sample. They show that a sample captures the community structure of the original network. However, other network properties of the sampled graph are not taken into account.

A similar idea of maintaining community structure under the crawling scenario was introduced in [7]. Salehi et al. introduce PageRank-sampling(PRS). The algorithm samples nodes to explore the region where the crawler is present, by selecting the node with the highest estimated PageRank. Then, it attempts to escape from the current region by selecting a node with low estimated PageRank but high estimated unobserved neighbors. Experiments show that the sampled graph produced by PRS preserves community structure.

Gjoka et al. propose the Metropolis-Hastings Random Walk algorithm to crawl an unbiased sample of Facebook network [8]. The key idea of the algorithm is to balance the visiting frequencies between high degree and low degree nodes. The results demonstrate that MHRW produces an approximate uniform sample while both BFS and Random Walk produce a sample with a biased degree distribution.

Breadth-first search (BFS) algorithm is widely used for crawling large networks (e.g. OSNs and the WWW). One of the largest OSN crawling studies is presented in [9]. Mislove, et al. study and analyze properties of several OSNs (e.g. Flickr, YouTube and LiveJournal). BFS crawler is used for collecting network data and confirm that these networks have power-law, small-world and scale-free properties. Similarly, Ahn et al. study a large South Korean OSN, CyWorld [10]. A snowball sampling technique is used. They claim that snowball sampling can discover high degree node, which help a crawler expand the search region.

Avrachenkov, et al. present a greedy approach called Maximum Observed Degree (MOD) [5]. The algorithm aims to maximize the node coverage of the sampled graph. The crawler selects a node with largest observed degree in each step. The MOD performance significantly outperforms other algorithms like BFS and RW. They state that sometimes MOD perform poorly, but leave that discussion for future work. Laishram, et al. show that MOD does not work well on directed graphs and present PMD crawler [11] that predicts which k nodes are most likely to have the highest number of unobserved nodes. PMD works the best on directed networks.

While the literature contains many proposed crawling methods, there is very little work that gives insight into these algorithms. One such work is presented in [12]. Ye, et al. present a large-scale empirical study. Two crawling objectives are considered: maximizing node coverage and edge coverage. This work conducts a high-level comparison of sampling methods. Likewise, Areekijseree, et. al. study the effect of network properties on the performance of crawling algorithms [1]. The authors perform an analysis of several crawling approaches with respect to the task of maximizing node and edge coverage. Nine popular algorithms are selected and categorized into three classes: Node importance-based, Graph traversal-based, and Random Walk. They demonstrate that if the network contains strong community structure, a crawler may obstructed by these sharp borders, and have difficulty in moving between regions of the network. As a result, node importance-based methods work well on the large densely connected regions of networks, but get stuck when network has a clear community structure. In contrast, Random Walk is mostly unaffected by strong community structure, but does not fully explore a region before moving on. Graph traversal-based methods (BFS, DFS) are generally the worst with respect to these tasks.

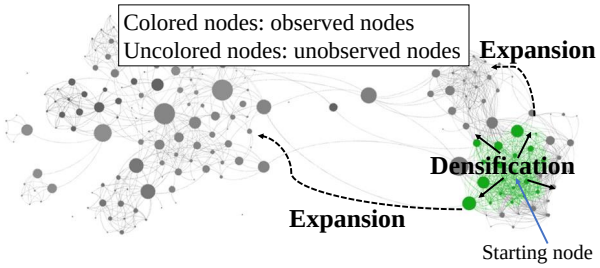


Figure 2: The concept of **DE-Crawler** algorithm. *Densification:* a crawler focuses on find as many nodes in a current region. *Expansion:* it tries to escape the current region and finds new unexplored dense region.

3. Problem Definition

Let $G = (V, E)$ be a static, undirected, unobserved network, where V and E are the set of nodes and edges, respectively. A starting node $n_s \in V$ and a budget b are given. The crawler explores the network by querying one node at a time, up to a total of b queries. To expand the observed sample, the crawler queries an *observed-but-not-queried* node. In response to each query, the algorithm receives all the neighbors of the queried node. These neighbors have now been *observed*. The output is a sampled graph $S = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, containing all nodes and edges observed. The goal is to discover as many nodes as possible.

4. Proposed Method: DE-Crawler

4.1. Key Ideas

In our previous work [1], we observed that a major factor in crawler performance is the ability to fully explore regions of a graph, while still being able to move between regions of the graph. In particular, we observe that there are two important classes of crawlers, but their performance depends on network structure. Node Importance-based methods are those that select the node for query based on a centrality measure computed from the observed graph. These methods excel when the underlying network contains overlapping communities and/or community size is relatively large compared to the query budget, because in such networks, the crawler is able to transition between regions. Still, a Random Walk crawler is the best when communities are disjoint.

Node Importance-based methods quickly explore individual regions, but tend to get 'stuck' inside a community when community borders are sharp. As these methods continue querying nodes within the same community, then even though each node is queried at most once, many of the queried nodes' neighbors will have already been observed; thus, the total number of observed nodes is small. In contrast, the Random Walk crawler is capable of moving freely between regions, but only partially explores each one. Based on these findings, we propose **DE-Crawler**, a novel crawling method that incorporates the best of both worlds.

The basic concept of **DE-Crawler** is illustrated in Figure 2. Given a starting node, the crawler aims to discover and fill out the nodes in a dense region (e.g., a community). We refer to this stage as the *Densification* stage. After the crawler discovers most of the nodes in that region, it attempts to expand to another dense region. We refer to this stage as the *Expansion* stage.

4.2. DE-Crawler algorithm

Pseudocode for **DE-Crawler** is shown in Algorithms 1 and 2. Users must specify the budget for sample initialization $b^{'}$, total budget b, and the starting node n_s as the input parameters of the algorithm.

4.2.1. Initialization

When it begins, **DE-Crawler** conducts an Initialize step (line 2 in Algorithm 1). Here, the crawler collects a small sample so that it can obtain information about the underlying network structure. It uses this stage to initialize certain necessary parameters. A small amount of budget b' is used, where $b' << b$. The initial sample can be collected by using any crawling method, and we adopt the Random Walk-based method proposed in [13]. Using this technique, we can estimate the average degree of the network, which will be used for weight adjustment (discussed in the later section). The output is a sampled graph $S^{'}$.

4.2.2. Densification

Here, the crawler selects and queries nodes with the goal of exploring the current region. Pseudocode is shown in Algorithm 2. The intuition here is to quickly find as many nodes as possible, since real networks are known to contain communities that are internally densely connected. Therefore, the sooner the crawler finds the highly-central hub nodes (e.g. high degree nodes), the faster the crawler can observe the remaining nodes in the region. Building on our earlier work in [1], this stage is based on the success of the *Node Importance-based* methods in exploring individual regions [1]. The basic idea is to pick a node with the highest observed centrality (i.e. degree/PageRank centrality) since it is likely that these high centrality nodes are hub nodes. In this way, each query observes many new nodes.

Node Selection: In order to select a node that will give high true degree, **DE-Crawler** identifies candidate query

Algorithm 1 DE-Crawler

1: **function DE-CRAWLER**($n_s, b, b^{'}$)
2: S = Initialize($n_s, b^{'}$)
3: **for** $t = b^{'}$ to b **do**
4: v_d = Expansion(S)
5: $S^{'}$ = Densification(v_d)
6: S = Merge($S, S^{'}$)
7: **return** S

Algorithm 2 Densification

1: **function** DENSIFICATION(v)
2: **for** $s_t^d < s_t^e$ or $t < b$; $t+ = 1$ **do**
3: $V^{'}$, $E^{'}$ = Query(v)
4: $s_t^d = \alpha_1 \cdot \frac{d_v^{new}}{d^{ex}} + \beta_1 \cdot s_{t-1}^d$
5: $s_t^e = \alpha_2 \cdot \frac{d_v^{seen}}{d_v^{ex}} + \beta_2 \cdot s_{t-1}^e$
6: $v = argmax_{v \in V_o}\{\Phi(v) = \hat{d}_v^o \cdot (1 - \hat{c_v})\}$
7: $S^{'}$ = updateSample($V^{'}$, $E^{'}$)
8: **return** $S^{'}$

nodes κ as those in the top 20% as ranked by observed degree.[1] For each candidate $v \in \kappa$, a score $\Phi(v)$ is calculated. The score is defined as $\Phi(v) = \hat{d_v^o} \cdot (1 - \hat{c_v})$, where d_v^o and $\hat{c_v}$ are, respectively, the normalized observed degree and observed clustering coefficient of node v. This formula is motivated by the observation that hub nodes tend to have high degree and low clustering coefficient [14]. The crawler then selects and queries the node v with the highest score.

Switching criterion: After each query, a crawler must decide whether to keep exploring, or escaping from the current region. To do so, two scores are computed at t-th step: the *densification score* s_t^d and the *expansion score* s_t^e. These scores are used to approximate the number of nodes left unexplored.

The switching happens when $s_t^d < s_t^e$. As the crawler stays in a region, the number of observed nodes increases while the number of new nodes added will start to decrease (diminishing marginal returns). The crawler will initially find many new nodes in the same community, but this amount drops as more and more nodes in the region are queried.

At each step t, the s_t^d and s_t^e are calculated (line 4 and 5 in Algorithm 2). These scores have two terms which incorporate the current stage and previous stage of the sample. The *Densification score* s_t^d indicates how many new unseen nodes are found after node v is queried at step t. It is defined as $s_t^d = \alpha_1 \cdot \frac{d_v^{new}}{d_v^{ex}} + \beta_2 \cdot s_{t-1}^d$, where d_v^{new} is the number of new edges that connect node v to new discovered nodes after the query and d_v^{ex} is the excess degree, which is defined as the difference between the true degree and the observed degree of the node before the request. α and β are the weighting parameters that control the influence of the current score and the previous score.

On the other hand, the *Expansion score* s_t^e measures the amount of nodes that have been seen so far, which is given by the ratio of d^{ext}, the number of new edges that link to already-observed nodes, to excess degree d^{ex}. It is defined as $s_t^e = \alpha_2 \cdot \frac{d_v^{seen}}{d_v^{ex}} + \beta_2 \cdot s_{t-1}^e$, where d_v^{seen} is the number of new edges that connect node v and nodes that already be in the sample before a request.

β_1 and β_2 are parameters that weight the densification and expansion scores of the sample at step $t-1$. We have observed that setting $\beta_1 = \beta_2 = 0.5$ gives us the best results.

However, the algorithm is sensitive to the values of α_1 and α_2. As the sampling process goes on, the number of observed nodes increases while the number of new nodes drops. We observe that s^e increases very quickly, because more and more of the same nodes are observed. So, if α_1 and α_2 are assigned with equal weight, the score will be biased towards expansion. To tackle that, we keep the value of α_2 lower than α_1. With several trials, we found that setting α_2 to 1, and varying the value of α_1 according to the network, works best. We initialize α_1 as follows:

Generally speaking, if the network is easy to expand, we want the crawler to spend more time on densification than on expansion. To set α_1, the crawler looks at the initial sample to estimate the expansion factor. We adopt the idea from work on dynamic processes, e.g. epidemic spreading or information diffusion, to estimate the expansion factor. We set α_1 to be the ratio between the maximum degree and average degree of nodes in the initial sample ($\alpha_1 = d_{max}/\hat{d}$). This ratio is closely related to "epidemic threshold" τ, which governs how fast an epidemic can spread, and thus is also related to how quickly the crawler can expand the sample [15].

1. This strategy is based on the *law of the vital few* or the *80/20 rule*.

4.2.3. Expansion

The crawler attempts to move out of the current region and attempts to search for a new unexplored dense region. In the spirit of an explore-exploit algorithm, we use the approach of choosing a node uniformly at random from the list of observed-but-not-queried nodes. In the earlier *Densification* stage, **DE-Crawler** selected a node from the top 20% of nodes as ranked by observed degree; here, **DE-Crawler** selects a random node in the bottom 80% of observed degrees, since these nodes are poorly connected to the sampled network.

5. Experimental Setup

We compare **DE-Crawler** to seven baseline crawling methods `RW`, `BFS`, `MOD`, `OPIC`, `Snowball`, `DFS`, and `Random`. Due to space constraints, we present results only for the following four baselines: `RW`, `BFS`, `MOD` and `OPIC`. We chose these methods because our previous work in [1] grouped them into three classes. These four methods represent the best of each class. The details of these baseline algorithms are described as follows:

1) *Maximum Observed Degree (MOD)*: A crawler selects the *observed-but-not-queried node* that has the highest observed degree. This method works well when communities are overlapping [5].

2) *Online Page Importance Computation (OPIC):* It is an online algorithm which aims to estimate each node's centrality score with only local updates. OPIC belongs to the same class as MOD in [1], and outperforms MOD in a some cases. In each step, the algorithm updates the scores of the most recently queried node and its neighbors. Each node is given an initial score, and the score is distributed evenly to it neighbors after each query. The node with the highest score is selected for the next query [16].

3) *Random Walk (RW):* A crawler transitions to a random neighbor of the latest queried node. Nodes can be visited multiple times but crawler only queries a node if it was not queried before. According to [1], RW is the most *stable* algorithm, performing consistently across network types.

4) *Breadth-first Search (BFS):* Due to its simplicity, BFS is one of the most popular crawling algorithms [9]. Nodes are selected and queried in FIFO fashion.

As described in [1], networks of the same type tend to have similar structural properties, and there is no single method that performs the best across different types of networks. As shown in [1], `MOD` and `OPIC` perform the best when 1) the underlying network contains overlapping communities and 2) the size of each community is large compared to the query budget, even if the communities are disjoint and have sharp borders. On the other hand, the `RW` crawler is the best on networks that contain disjoint

communities structure. The results show that its performance is not affected by network properties. The BFS crawler is generally a weak performer, but we include it here due to its popularity as a crawling algorithm.

In our experiments, we use a total of eighteen networks from different categories. The statistics of each network are provided in Table 1[2]. We perform 10 runs on each network and report the average fraction of nodes observed by each crawler. We set the query budget b to be 10% of the total nodes in the network and for **DE-Crawler**, we set the initialization budget b' to be 15% of the total budget.

TABLE 1: The statistics of realworld networks used.

Type	Network	# Nodes	# Edges	$\hat{d}$	$\hat{CS}$	Q
1. Collab.	Astro	17903	196973	22.00	436.66	0.63
	CondMat	21363	91287	8.55	374.79	0.72
	HepPh	11204	117619	21.00	238.38	0.65
	Citeseer	227320	814135	71.6	988.34	0.89
2. FB100	Bingham	10001	362893	72.57	1250.13	0.45
	JohnsHopkins	5157	186573	72.36	515.70	0.45
	WashU	7730	367527	95.09	966.25	0.47
	Yale	8561	405441	94.72	856.10	0.43
3. OSN	Anybeat	21250	66892	6.30	259.15	0.48
	Slashdot	70068	358648	10.24	173.86	0.36
	Hamsterster	2000	16097	16.10	66.67	0.54
4. Web	Google	1299	2774	4.27	34.18	0.93
	IndoChina	11358	47607	8.38	153.49	0.94
	Webbase-2001	16062	25594	3.19	232.78	0.93
5. Tech.	RL-caida	190914	607611	6.36	856.11	0.86
	PGP	10680	24316	4.55	106.80	0.88
	Router-rf	2113	6633	6.28	88.04	0.69
	WhoIs	7476	56944	15.23	276.89	0.56

To obtain a fair comparison across networks, we compare the performance of **DE-Crawler** and the baseline methods against the greedy oracle, Maximum Excess Degree (MED). MED assumes that each node's true degree is known, and in each step, queries the node with the highest excess degree (the difference between true degree and observed degree). With this oracle, we can compute the *regret* r of each crawler, as $r = (y_o - y_x)/y_o$, where y_o is the number of nodes discovered by the oracle, and y_x is the number of nodes discovered by crawler x. Lower values of regret indicate higher performance.

6. Experimental Results

In this section, we present the performance of our proposed algorithm **DE-Crawler** against other baseline methods, as described in Section 5. We evaluate these methods with respect to the *node coverage* task (discover as many nodes as possible). Note that, we compare our algorithm to many algorithms; RW, BFS, MOD, OPIC, Snowball, DFS and Random, but these four baselines (RW, BFS, MOD, OPIC) were best. So, we present the results of these four baselines.

Results are presented in Figure 3, 4 and Table 2. In Figures 3 and 4, the x-axis represents the query budgets, and the y-axis represents the fraction of nodes observed in the sample. Table 2 shows the overall $regret$ of each method, as compared to the oracle. Our results show that **DE-Crawler** is the best of both worlds: it performs consistently well across all network types, regardless of community structure. In all but one of the considered networks, **DE-Crawler** is the best performer.

As discussed earlier, RW and MOD are excellent methods, but the choice of which is best depends on the network structure. E.g., Figure 3 demonstrates the case where RW is better than MOD: these networks contain dense, distinct communities, and MOD has trouble transitioning between regions. Figure 4 illustrates the case where MOD outperforms RW: these networks have overlapping communities with fuzzy borders, allowing MOD to move freely between regions.

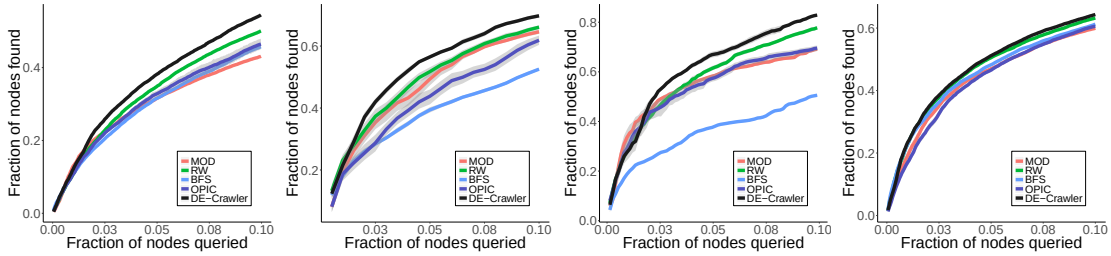
But in both cases, as we can clearly see, the performance of **DE-Crawler** substantially outperforms all the baselines. By switching between expansion (moving to new regions) and densification (exploring the current region), **DE-Crawler** is able to gain an improvement of up to 28% as compared to the best baseline methods. The results in Table 2 also show that **DE-Crawler** performs achieves a low regret, indicating that it performs close to the optimal greedy method. **DE-Crawler** has the lowest average regret of approximately 0.22, which is dramatically better than RW, the next best method.

From the results, it is clear that **DE-Crawler** outperforms all the baselines at the task of maximizing node coverage. It has a very *stable* performance across the network categories, suggesting that network structural properties have little effect on it. By using a mix of the *expansion* and *densification* strategies, and transitioning between phases when densification begins to exhibit diminishing marginal returns, **DE-Crawler** is able to achieve outstanding performance.

TABLE 2: The average $regret$ of **DE-Crawler** and baseline algorithms (lower value means better performance).

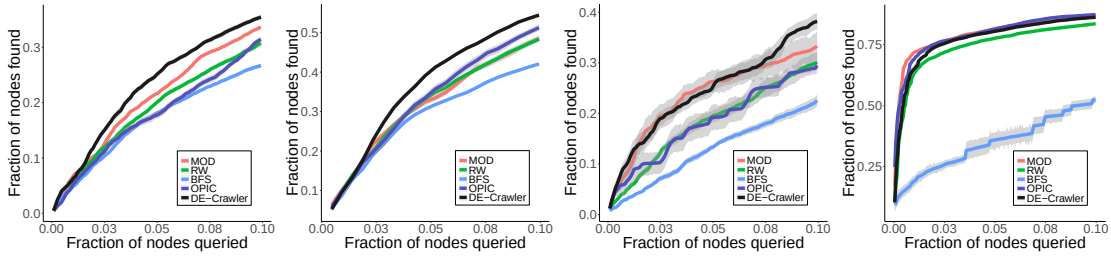
	Network	DE	RW	MOD	OPIC	BFS
1.	AstroPh	**0.144**	0.159	0.202	0.194	0.185
	CondMat	**0.292**	0.349	0.440	0.396	0.406
	HepPh	**0.158**	0.246	0.350	0.205	0.270
	Citeseer	**0.359**	0.467	0.452	0.458	0.557
2.	Bingham	**0.023**	0.024	0.130	0.145	0.026
	JohnsHopkins	**0.034**	0.041	0.129	0.148	0.047
	WashU	**0.012**	0.013	0.149	0.163	0.027
	Yale	**0.007**	0.020	0.080	0.107	0.023
3.	Anybeat	0.082	0.110	0.079	**0.070**	0.442
	Slashdot	**0.045**	0.129	**0.045**	0.046	0.419
	Hamsterster	**0.119**	0.165	0.184	0.218	0.336
4.	Google	**0.450**	0.676	0.471	0.582	0.612
	Indochina	**0.522**	0.623	0.583	0.631	0.718
	Webbase	**0.730**	0.764	**0.730**	0.781	0.764
5.	RL-caida	**0.359**	0.370	0.372	0.449	0.419
	PGP	**0.383**	0.465	0.416	0.453	0.536
	Routers-RF	**0.219**	0.307	0.304	0.265	0.397
	WhoIs	**0.130**	0.184	0.274	0.270	0.469
	Average	**0.226**	0.284	0.299	0.310	0.370

2. All of these networks can be found at www.networkrepository.com.



(a) CondMat (b) Hamsterster (c) WhoIS (d) AstroPh

Figure 3: **DE-Crawler** consistently outperforms or matches the best baseline method on networks that RW outperforms MOD.



(a) PGP (b) Routers-RF (c) IndoChina-2004 (d) Anybeat

Figure 4: **DE-Crawler** consistently outperforms or matches the best baseline method on networks that MOD outperforms RW.

7. Conclusion

We considered the problem of **online network crawling** with the goal of maximizing node coverage. Our previous work has demonstrated that the performance of existing crawling methods is heavily affected by network properties [1]. Intuitively, a strong community structure can obstruct a crawler from moving between regions. Based on that observation, we introduced **DE-Crawler**, which consists of two main stages: *Densification*, which explores the current region, and *Expansion*, in which the crawler transitions to a new region. Our results over 18 datasets show that **DE-Crawler** outperforms all other baselines, with an improvement of up to 28% over the next best baseline. Moreover, **DE-Crawler** is consistently the best over all considered network types, and the results also show that **DE-Crawler** performance is close to the performance of the optimal greedy crawling algorithm.

8. Acknowledgements

We thank Jeremy Wendt of Sandia National Laboratories for thoughtful comments and conversations. This material is based upon work supported in part by the U. S. Army Research Office under grant number #W911NF1810047.

References

[1] 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.

[2] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *19th international conference on World wide web*, 2010.

[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] ——, "Online sampling of high centrality individuals in social networks," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 91–98.

[5] K. Avrachenkov, P. Basu, G. Neglia, B. Ribeiro, and D. Towsley, "Pay few, influence most: Online myopic network covering," in *Computer Communications Workshops*, 2014.

[6] 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.

[7] M. Salehi, H. R. Rabiee, and A. Rajabi, "Sampling from complex networks with high community structures," *Chaos*, vol. 22, no. 2, 2012.

[8] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Unbiased sampling of facebook," *preprint arXiv*, vol. 906, 2009.

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

[10] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in *International conference on WWW*, 2007.

[11] R. Laishram, K. Areekijseree, and S. Soundarajan, "Predicted max degree sampling: Sampling in directed networks to maximize node coverage through crawling," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 940–945.

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

[13] A. Dasgupta, R. Kumar, and T. Sarlos, "On estimating the average degree," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 795–806.

[14] 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.

[15] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 4, p. 1, 2008.

[16] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive on-line page importance computation," in *Proceedings of the 12th international conference on World Wide Web*, 2003.