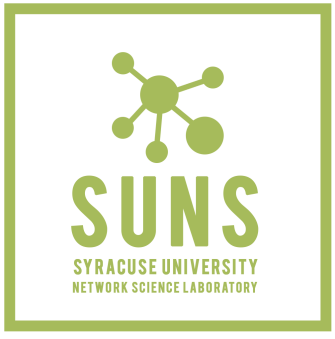
DE-Crawler: A Densification-Expansion Algorithm for Online Data Collection

Katchaguy Areekijseree (Bebe) Sucheta Soundarajan
{kareekij,susounda}@syr.edu

Syracuse University, Syracuse, NY, USA

ASONAM 2018

Syracuse University



Data Collection

- The study of complex networks has gained a lot of attention from researchers.
- A convenient way is to get data from OSNs.
- Many OSNs provide APIs for accessing data.
- **Network Sampling or Network Crawling**
- **Challenge:** The data collection process takes a lot of time.
- For example, they took 6 days to get 8,000 unique users on Twitter in [WWFJS].

Use Case Scenario

- You want to analyze the community structure of the network.
- Before the analysis task, the first thing is to collect the network data - collect as many distinct users as many as possible.
- You have b days (or weeks) to do a data collection.
- **Question: Given a budget (time/money), which nodes should we select and make query such that we obtain distinct users as many as possible?**

Problem Definition

Let $G = (V, E)$ be a static unobserved, undirected network.

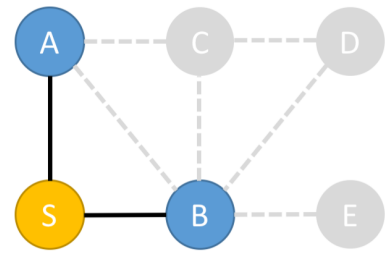
- **Input**: A starting node n_s and a number of query budget b.
- In each step, the crawler queries an observed-but-not-queried node and *ALL neighbors* are returned as a response.
- Repeat b times.
- **Output**: a sample graph $S = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, containing all nodes and edges observed.

Goal: Maximize a number of observed nodes (*node coverage*).

Related Applications

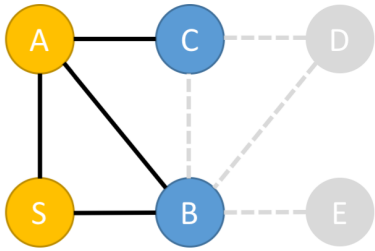
preserving community structure[MBW10a], preserving high centrality nodes[MBW10b], census-liked applications.

Making Query: Toy Example



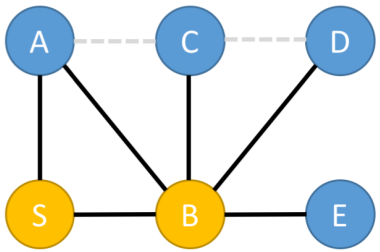
t	|V'| = 3 V' ={S, A, B}

Choice 1: Select **node A** to query



t+1	|V'| = 4 V' ={S, A, B, C}

Choice 2: Select **node B** to query



t+1	|V'| = 6 V' ={S, A, B, C, D, E}

Observed queried node | Observed unqueried node | Unobserved node

Previous Work

Our Previous Study [ALS18]: Network Properties of Interest

The goal of the previous study.

The relationship between network properties and crawling algorithms performance

1. We were interested in 3 properties such as **community separation**, average degree and community size.
2. We selected these properties based on the intuition that a crawler has difficulty in moving between regions.

Our Previous Study [ALS18]: Online Crawling Approaches

We selected nine popular algorithms from the literature and categorized them into three classes (G1-G3) based on the results.

- **G1: Node Importance-based Methods**
  - Maximum Observed Degree [ABN+14]
  - Maximum Observed PageRank [SRR12]
  - Online Page Importance Computation [APC03]
- **G2: Random Walk** [LF06]
- **G3: Graph Traversal-based Methods**
  - Breadth-first Search [MMG+07]
  - Depth-first Search
  - Snowball Sampling [AHK+07]
  - Random Crawling
  - Volatile Multi-armed Bandit [BPSF13]

Our Previous Study [ALS18]

Finding 1

Network properties have a strong effect on the performance of various crawling methods.

Finding 2

`Random Walk (G2)` crawler is most stable algoritm and works the best on networks with distinct community structure.

Finding 3

`Node Importance-based (G1)` crawler works the best on networks with overlapping community structure or large community.

Finding 4

The performance of `Graph Traversal-based (G3)` crawler cannot beat the others in this specific crawling goal.

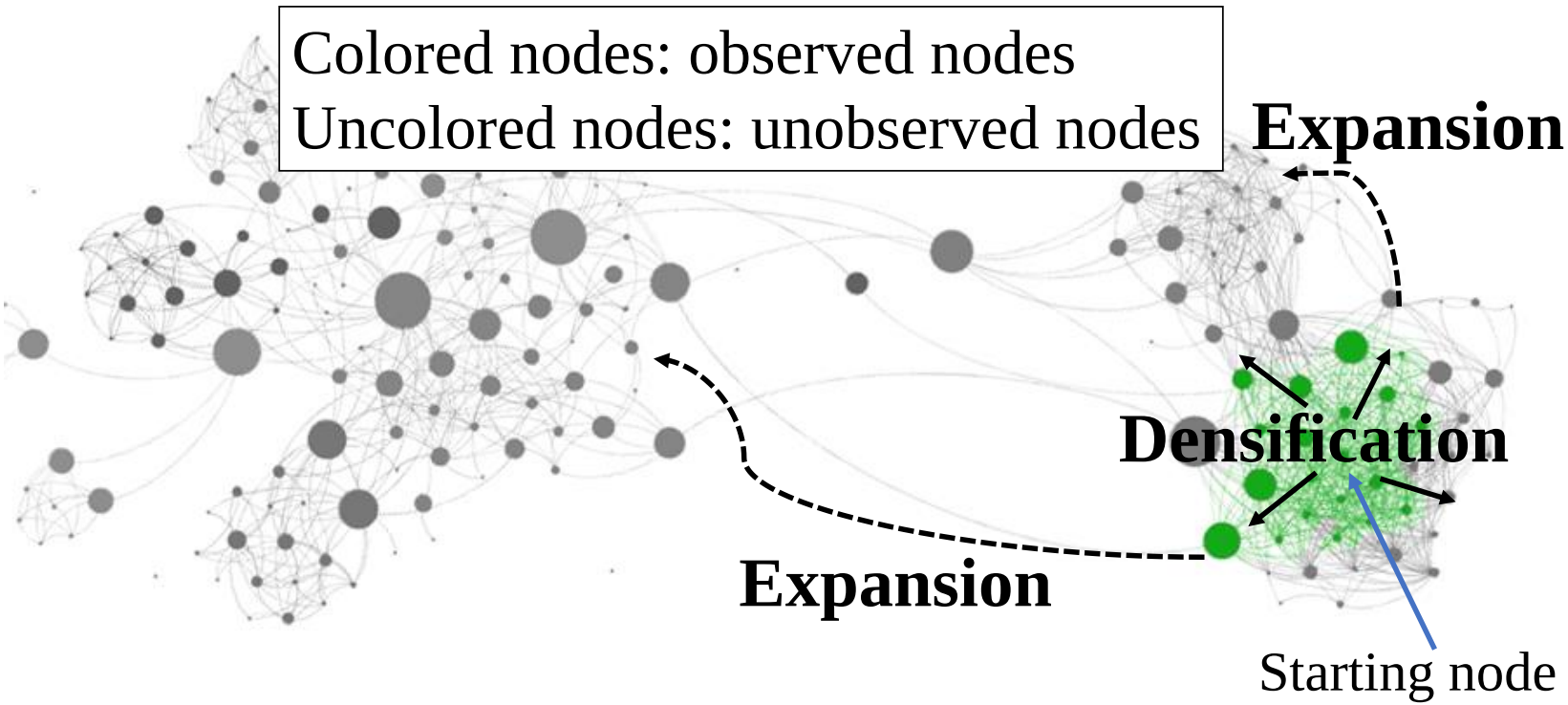
This work: DE-Crawler

Contribution of this work

In this work,

1. We present **DE-Crawler**, a novel crawling method, for the task of node coverage.
2. We perform the experiments on networks from diverse categories (collaboration, FB, OSNs, the WWW and technological).
3. We show that **DE-Crawler** performs the best across different network categories.

Key Concept of the **DE-Crawler**



DE-crawler

INPUT: a starting node, total budget, initialize budget.
OUTPUT: a sampled network S=(V', E').

Algorithm 1 Densification-Expansion

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: **end for**
8: **return** S
9: **end function**

DE-crawler: Initialization

Key Idea

Collect a small sample and initialize parameters.

- The crawler collects a small sample, so, it obtain information about the underlying network structure.
- This step can be done by any crawling technique, we adopt Random walk-based [DKS14].
- Initialize parameters: $\alpha_1, \alpha_2, \beta_1, \beta_2$.

DE-crawler: Densification

Key Idea

Explore the current region and quickly find as many nodes as possible

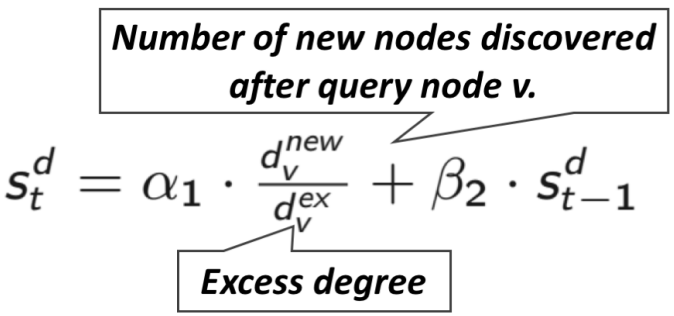
- Find hub nodes (high degree nodes)
- Node selection: For every node v,

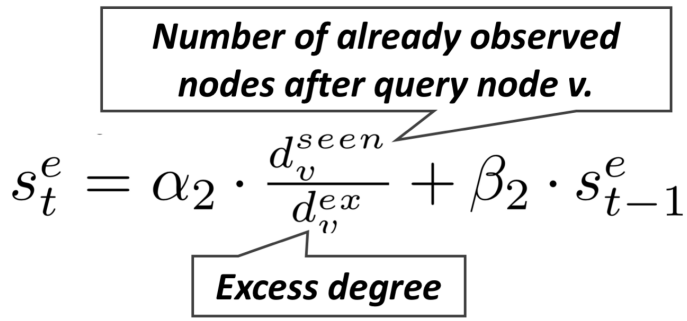
Normalize observed degree of node *v*. **Clustering coefficient of node *v*.**

$$\Phi(v) = \hat{d}_v^o \cdot (1 - \hat{c}_v)$$

DE-crawler: Densification

Switching criteria: $s_t^d < s_t^e$





- If the *densification score* (s_t^d) is high, it means there are more nodes left unexplored in the current region of the network.
- If the *expansion score* (s_e^d) is high, it means a crawler seems to see same nodes over and over.

DE-crawler: Expansion

Key Idea

The crawler wants to move to an unexplored region.

In the spirit of explore-exploit algorithm, we use the approach of choosing a node uniformly at random from the list of observed-but-not-queried nodes.

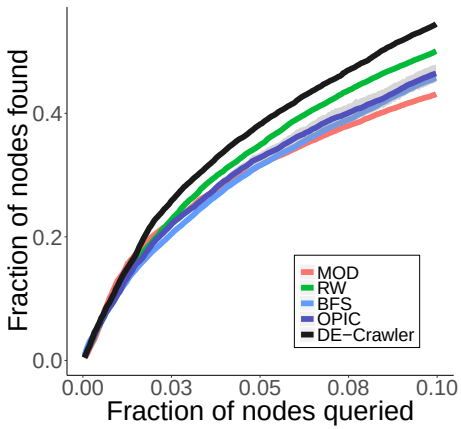
Experiments and Results

Experiment Setup

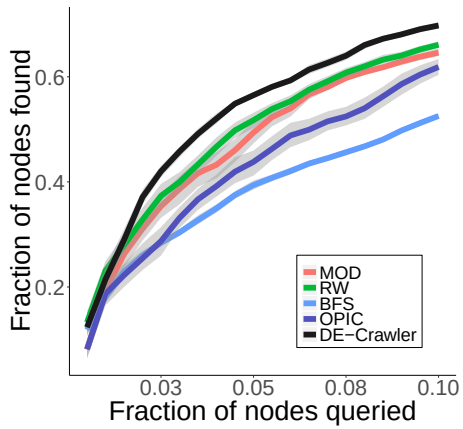
- We compare **DE-Crawler** to seven baseline crawling methods.
- Perform the experiments on eighteen networks from five categories.
- Perform 10 runs on each network and report the average.
- Set total budget b to be 10% of the total nodes and b' to be 15% of the total budget.

Results 1

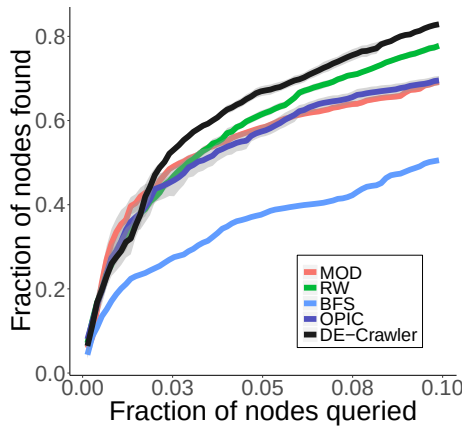
DE-Crawler consistently outperforms or matches the best baseline method on networks that `G2` outperforms `G1`.



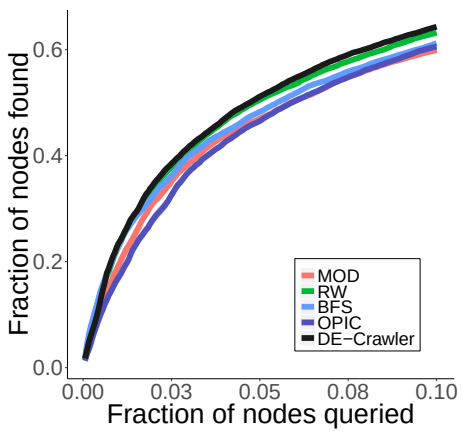
(a) CondMat



(b) Hamsterster



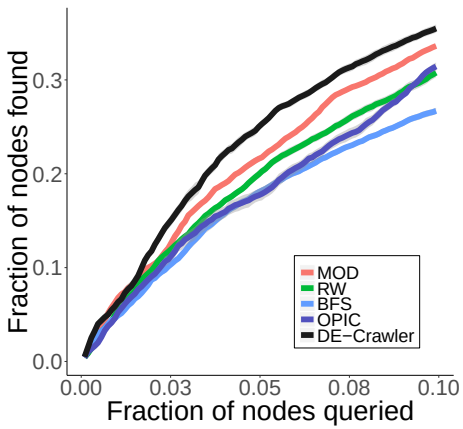
(c) WhoIS



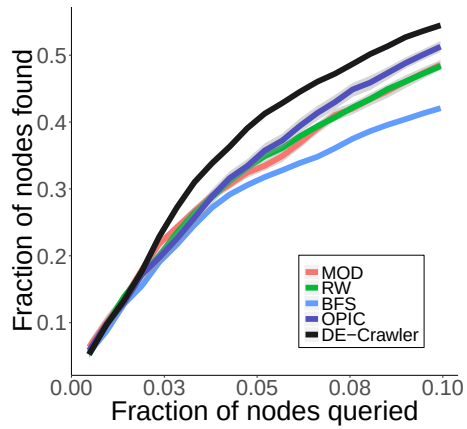
(d) AstroPh

Results 2

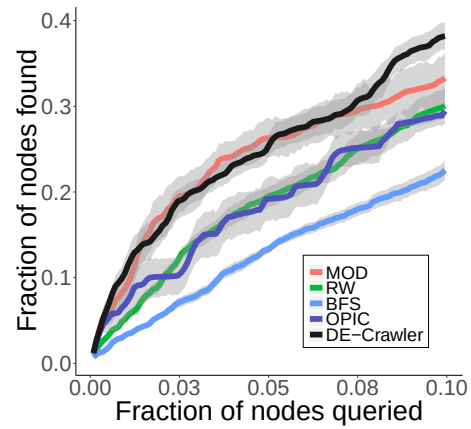
DE-Crawler consistently outperforms or matches the best baseline method on networks that `G1` outperforms `G2`.



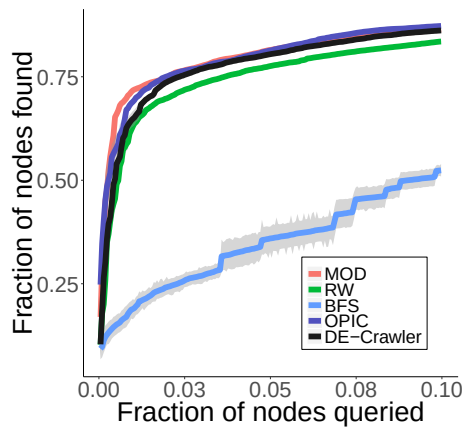
(a) PGP



(b) Routers-RF



(c) IndoChina-2004



(d) Anybeat

Results 3.1 - Compare against optimal greedy algorithm

- Optimal greedy algorithm: maximum excess degree (MED) algorithm.
  - excess degree = true deg - observed degree
- The average *regret* of **DE-Crawler** and baseline algorithms.
  - $regret = \frac{p_{optimal} - p_x}{p_{optimal}}$.
  - Lower regret, better in performance.

Results 3.2 - Compare against optimal greedy algorithm

Type	Network	DE	RW	MOD	OPIC	BFS
Collaboration	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
Facebook100	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
OSN	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
Web	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
Tech.	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

* *Lower is better*

Conclusion

- We considered the problem of **online network crawling** - maximize node coverage.
- We proposed **DE-Crawler**, a *Densification-Expansion* algorithm.
- **DE-Crawler** outperforms other baselines up to 28% improvement.
- **DE-Crawler** performance is consistent over all considered network types.

Thank You

Questions?

kareekij@syr.edu

References I

- Konstantin Avrachenkov, Prithwish Basu, Giovanni Neglia, Bruno Ribeiro, and Don Towsley, *Pay few, influence most: Online myopic network covering*, Computer Communications Workshops, 2014.
- Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong, *Analysis of topological characteristics of huge online social networking services*, International conference on WWW, 2007.
- Katchaguy Areekijseree, Ricky Laishram, and Sucheta Soundarajan, *Guidelines for online network crawling: A study of data collection approaches and network properties*, Proceedings of the 10th ACM Conference on Web Science, ACM, 2018, pp. 57–66.

References II

- Serge Abiteboul, Mihai Preda, and Gregory Cobena, *Adaptive on-line page importance computation*, Proceedings of the 12th international conference on World Wide Web, 2003.
- Zahy Bnaya, Rami Puzis, Roni Stern, and Ariel Felner, *Bandit algorithms for social network queries*, 2013 International Conference on Social Computing, IEEE, 2013, pp. 148–153.
- Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos, *On estimating the average degree*, Proceedings of the 23rd international conference on World wide web, ACM, 2014, pp. 795–806.
- Jure Leskovec and Christos Faloutsos, *Sampling from large graphs*, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2006, pp. 631–636.

References III


Arun S Maiya and Tanya Y Berger-Wolf, *Online sampling of high centrality individuals in social networks*, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2010, pp. 91–98.


———, *Sampling community structure*, Proceedings of the 19th international conference on World wide web, ACM, 2010, pp. 701–710.


Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee, *Measurement and analysis of online social networks*, 7th ACM SIGCOMM conference on Internet measurement, ACM, 2007, pp. 29–42.


Mostafa Salehi, Hamid R Rabiee, and Arezo Rajabi, *Sampling from complex networks with high community structures*, Chaos **22** (2012), no. 2.

References IV


Jeremy D Wendt, Randy Wells, Richard V Field Jr, and Sucheta Soundarajan, *On data collection, graph construction, and sampling in twitter*.