Crawling Complex Networks: An Experimental Evaluation of Data Collection Algorithms and Network Structural Properties

Katchaguy Areekijseree and Sucheta Soundarajan[1]

[1] *Department of Electrical Engineering & Computer Science. Syracuse University, NY USA.*

ABSTRACT

In recent years, researchers and data analysts have increasingly used online social network data to study human behavior. Before such study can begin, one must first obtain appropriate data. This process poses many challenges: e.g. a this platform may provide a public API for accessing data, but such APIs are often rate limited, restricting the amount of data that an individual collect in a given amount of time. Thus, in order for the data collector to e°ciently collect data, she needs to make intelligent use of her limited API queries. The network science literature has proposed numerous network crawling methods, but it is not always easy for the data collector to select an appropriate method: methods that are successful on one network may fail on other networks.

In this work, we demonstrate that the performance of network crawling methods is highly dependent on the struc-tural properties of the network. To do that, we perform a detailed, hypothesis-driven analysis of the performance of eight popular crawling methods with respect to the task of maximizing node coverage. We perform experiments on both directed and undirected networks, under five different query response models: complete, paginated, par-tial, in-out, and out responses. We identify three important network properties: community separation, average community size, and average node degree. We begin by performing controlled experiments on synthetic networks, and then verify our observations on real networks. Finally , we provide guidelines to data collectors on how to select an appropriate crawling method for a particular network.

Keywords: network sampling, network crawling, crawler performance, network structural properties

ISSN 2332-4031; DOI 10.34962/jws-69

1 Introduction

The study of social networks has become a critical aspect of research in a wide range of fields, and study of such networks can give us rich insight into human behavior. In recent years, many researchers have chosen to collect network data from online platforms. These platforms typically provide a public API for accessing the data. In other cases, human network data can be collected through surveys, interviews, etc.

The data collection process can require a significant amount of time, money, or resources. Thus, when collecting data, efficiency is extremely important. In particular, researchers are often interested in obtaining the most data possible given a limited data collection budget. In this paper, we consider the problem of **network sampling through crawling**, in which we have no knowledge about the network of interest except for the identity of a single starting node in the graph. The only way to obtain more information about the network is to query observed nodes for their neighbors, and thus expand the observed network. In this paper, we use *network sampling* and *network crawling* interchangeably.

In network crawling, there are a number of important goals, such as finding samples that are unbiased with respect to some property, locating 'important' nodes, or finding a sample that preserves information flow patterns. Here, we focus on the efficiency of the crawling algorithm itself- i.e., how quickly nodes and edges can be discovered through the crawling process. Our considered goal is maximizing the total number of nodes observed. A multitude of network sampling algorithms (Avrachenkov *et al.*, 2014; Salehi *et al.*, 2012) have been created for this and other goals, and it is often difficult for a user to select an algorithm for her particular application.

The goal of this work is to analyze, characterize, and categorize the the performance of various network sampling algorithms at the node coverage task, over networks with varying structural properties. This paper is an extension of our earlier work (Areekijseree *et al.*, 2018), in which we (1) demonstrated that node importance-based methods (i.e., those that seek to find and query the 'most important' node at each time step) performed well on networks with fuzzy community structure, but tended to get stuck in a region for networks with crisp community borders, and (2) showed that Random walk-based methods were able to transition between regions, regardless of community structure. (3) By knowing network properties or network type, we can suggest the appropriate method that will work the best.

Most previous work, including our earlier work (Areekijseree *et al.*, 2018), has assumed that all neighboring nodes are returned in response to a query (Avrachenkov *et al.*, 2014; Areekijseree and Soundarajan, 2018). However, this assumption is not valid in many real-world scenarios; e.g. Facebook API returns a list of 25 users as a response when request for a list of friends. Therefore, the crawler may need multiple queries to obtain all friends of this particular user. In this paper, we extend our earlier analysis to several other query models, each

motivated by a real-world application. In particular, we define a total of five different responses on both undirected and directed networks; *complete*, *paginated*, *partial*, *in-out* and *out* response. This work extends our earlier work (Areekijseree *et al.*, 2018.) Our main contributions here are as follows:

1. In addition to the 'complete' query response, we consider four other realistic query models- *paginated*, *partial*, *in-out*, and *out* responses- which describe the neighbor data obtained in response to a query on a node.

2. While existing work, including our previous work, considers undirected networks, we perform the analysis on both directed and undirected networks.

3. We observe that the performance of crawling algorithms under the *complete*, *partial*, *paginated*, and *in-out* query models are similar. In contrast, under the *out* query response model, algorithms exhibit different performance. In particular, under the out query response, we see that the performance of G1 methods is not substantially affected as average degree or community size increases. Moreover, all methods tend to have similar performance on networks with high community mixing.

4. We provide guidelines on how a user can select an appropriate crawling method for a network from a given domain, under a specific query model, even before observing specific properties of that network.

2 Related Work

Network sampling can be separated into two main categories. The first is work on *down-sampling*, in which one has full access to or ownership of the network data, but the size of the data makes the network too large to feasibly analyze. The objective in this case is to scale the network down to some desired size. A good sample should maintain the relevant properties and characteristics of the original network so that the results of analysis obtained from the sample should be similar to the results one would have obtained from the original network.

The second category of work is on the problem of *network crawling*. In this case, one has a limited access to the network data, and can retrieve information about the network by performing queries on observed nodes (e.g., through an API or surveys). By repeatedly querying the observed parts of the network, the sample is expanded from the single initially observed node or set of nodes.

Both cases are often broadly referred to as 'sampling'; however, they require fundamentally different approaches. The goal of our work is to analyze and characterize the performance of *network crawling* algorithms. We refer the reader to this survey (Ahmed *et al.*, 2014) for a more detailed discussion.

Algorithms for Network Crawling: Network crawling is frequently used in scientific studies. For example, Mislove, et al. use a BFS crawler to collect data from large networks, including Orkut, Youtube, Live Journal, and Flickr, before analyzing the structures of those networks (Mislove *et al.*, 2007). Similarly, Ahn, et al. use a BFS crawler to collect data from CyWorld, a South Korean social networking site, and MySpace (Ahn *et al.*, 2007). However, it is known that the network samples produced by a BFS crawler contain bias: specifically, the crawler is disproportionately likely to visit hub nodes. Kurant, et al. present a modification of BFS to correct this issue (Kurant *et al.*, 2011), and Gjoka, et al. propose an unbiased approach based on Metropolis-Hastings Random Walks (Gjoka *et al.*, 2009). These latter works demonstrate that the proposed crawlers can balance the visiting frequencies between low and high degree nodes.

There has additionally been a great deal of interest on network crawling for specific applications. Salehi, et al. introduce a method, based on PageRank, for crawling networks to preserve community structure (Salehi *et al.*, 2012), and Avrachenkov, et al. propose a greedy crawling method, called Maximum Observed Degree (MOD), that has the goal of finding as many nodes as possible with a limited query budget (Avrachenkov *et al.*, 2014). MOD operates by selecting the node with the highest observed degree in each step, with the assumption that nodes that have a high sample degree are also likely to have a high true degree. The OPIC method adopts a similar idea, except that it queries the node with the highest PageRank in each step (Abiteboul *et al.*, 2003). Experimental results show that both MOD and OPIC significantly outperform other crawling methods (Avrachenkov *et al.*, 2014).

Analysis of Sampling Algorithms: There has additionally been a great deal of work on comparing sampling methods. Leskovec and Faloutsos present a study of the characteristics of different down-sampling methods, with the goal of determining which method leads to samples with the least bias with respect to various network properties (Leskovec and Faloutsos, 2006). They conclude that Random walk sampling is the best at preserving network properties. Similarly, Kurant, et al. analyze BFS crawlers, and demonstrate that such methods are biased towards high degree nodes (Kurant *et al.*, 2010).

Ye, et al. present an empirical study that focuses on performance, sensitivity, and bias (Ye *et al.*, 2010), and Ahmed, et al. provide a framework for classifying sampling algorithms with respect to how well they preserve graph statistics (Ahmed *et al.*, 2014).

These existing works have generally sought to determine which method is 'best' overall, at a high level. In contrast, our goal is to understand the interplay between *network structure* and *crawler performance*, rather than simply evaluating performance. We wish to understand the underlying reasons behind why certain algorithms are successful on certain types of networks, but may show weaker performance on other networks. These insights give guidance to those who are developing new network crawling algorithms, as well as to those who seek to select a single algorithm for crawling a specific network.

3 Network Crawling Overview

3.1 The Network Crawling Problem

Let $G = (V, E)$ be an unobserved network (either directed or undirected). In this problem, one is given a starting node $n_s \in V$ and a query budget b. To collect data, one may perform a query on a previously-observed node. As a *query response*, a list of neighboring nodes is returned. In this work, we consider five different types of query responses, as described in Section 3.2. In each iteration, the crawler selects one node whose neighborhood has not yet been fully explored to query

(for further discussion of this step, see Section 3.3. The crawler stops once b queries have been made. The output is a sample graph $S = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, containing all nodes and edges observed.

In this work, we consider the case where the crawling goal is to find nodes as many as possible. We refer to this goal as *maximizing node coverage.*[1] We selected this goal because it has been closely tied to several important applications; e.g., Maiya, et al. use the node coverage goal for the task of generating a sample that preserves community structure (Maiya and Berger-Wolf, 2010). There are numerous other crawling goals that one could consider (such as obtaining an unbiased sample), but these goals require fundamentally different approaches, and so we do not consider them here.

3.2 Query Responses

The data that one obtains in response to a query on a node varies depending on domain. For example, some APIs may return all neighbors of the queried node, while others may only return a subset, and so the crawler must query that node repeatedly to obtain all neighbors. If questioning an individual as to the identities of her friends, one may receive a random sample in response. Our earlier work considered only the case where all neighbors are returned in response to a query (Areekijseree *et al.*, 2018). Here, we consider five different responses, motivated by these various real-world settings. First, for undirected networks, we consider three types of query responses: *complete*, *paginated* and *partial*. For directed networks, we consider *in-out*, and *out* responses. Details are as follows:

3.2.1 Responses on Undirected Networks

On undirected networks, we consider three different types of query responses.

Complete Response: In this query model, *all* neighboring nodes are returned in response to a query on a node. This is motivated by settings such as network routing; e.g. the '*netstat*' command in Linux returns all network connections that connect from the machine.

Paginated Response: In the paginated response query model, only k neighboring nodes are returned in response to a query on a node. The neighbors of each node are divided into distinct chunks, or *pages*. Each page contains up to k neighbors (except the last page, which may contain fewer); thus, the crawler may need to query a node more than once to obtain all of its neighbors. This response is common in APIs for online social networks, such as querying photos/albums on Facebook. We assume that the crawler is notified (e.g., by the API) when there are no further pages to be returned.

Partial Response: In the partial response query model, like the paginated response model, k neighbors of a node are returned in response to a query on that node. However, in the partial response case, these neighbors are returned in a random fashion, and different queries may result in duplicate returned neighbors. To observe all neighbors of a node v, the crawler must conduct at least d_v/k queries on v, where d_v is the degree of node v, but even after performing this many queries, one cannot be guaranteed that all neighbors have been observed. This model is motivated by scenarios such as personal interviews, such as surveys or criminal interrogations. In such cases, one may ask a respondent to identify all of her friends, but her memory is likely to be incomplete, with an element of randomness.

3.2.2 Responses on Directed Networks

There are two types of edges incident to any node v; *incoming edges* originate at another node u and terminate at v, while *outgoing edges* originate at u and terminate at another node v. Accordingly, we define two query responses: *in-out* and *out*.

In-Out Response: The crawler can choose to query for either incoming or outgoing edges. To get all of the neighbors of a node, the crawler must perform two queries on that node. This query model can be found in some online social networks- e.g., on Twitter, one can separately query for the followers or friends (followees) of a node.

Out Response: The query for incoming edges is not available, and the crawler can only query for outgoing edges of a node. This response applies to the web scraping scenario: one can quickly identify which websites a particular site is linking *to*, but not which websites it is linked *from*. Some OSN platforms also provide APIs with similar behavior (e.g., Flickr has an API call for obtaining a list of users that user A follows, but not the users who follow A).

3.3 Closed nodes vs. Open nodes

As the crawler proceeds, the nodes seen so far can be grouped into various categories. We refer to all the nodes in a sample as *observed* nodes. An observed node may be either *closed* or *open*. Closed nodes are the nodes whose entire neighborhood is believed to be known, while open nodes are those nodes that are believed to still have unobserved neighbors.

Under the *complete*, *paginated*, *in-out*, and *out* query response models, we know exactly how many queries are required to observe all neighbors of a node. The *complete, in-out*, and *out* models return all [in/out] edges of a node, and for the *paginated response* model, we assume that the crawler is informed when no further pages can be returned (e.g., the API indicates that no more data is available, or provides the total degree of a node). Once all of a node's neighbors are observed, that node will be changed from *open* to *closed*.

The *partial* response model is somewhat more challenging to deal with, because each query returns a random subset of the queried node's neighbors. Information about node's total degree is unavailable, so the crawler must estimate each node's degree. Once the observed degree is equal the estimate degree, the node is switched from open to closed. To perform this estimate, we adopt the "*mark and recapture*" technique from ecology, which is used to estimate population size (Robson and Regier, 1964). The degree of node v can be estimated by $d_v = M \cdot C/R$, where M is the total number of distinct neighbors that have been discovered prior to the current query, C is the total number of number of neighbors discovered after the current query and R is the number of neighbors returned by the current query that had been previously observed.

[1] We also considered the equivalent *edge coverage* goal, and the results were largely similar.

3.4 Online Crawling Methods

Maximum Observed Degree (MOD) The crawler greedily selects the open node with the highest observed degree. It has been shows that `MOD` substantially outperforms other methods at the node coverage task (Avrachenkov *et al.*, 2014).

Maximum Observed PageRank (PR) The crawler acts similarly to `MOD`, except that the PageRank score of every node is used to select the query node. It has been demonstrated that this technique captures the community structure of the network (Salehi *et al.*, 2012).

Online Page Importance Computation (OPIC) OPIC aims to calculate each node's importance score without recalculating it in each step. The crawler updates only the scores of the most recently queried node and its neighbors. Initially, each observed node is given an equal amount of *"cash"*. The crawler queries the node with the highest cash, and this cash is spread equally between the node's neighbors. `OPIC` can quickly compute the importance of nodes (Abiteboul *et al.*, 2003).

Random Crawling (Rand) The crawler randomly selects one open node for the next query.

Breadth-First Search (BFS) The crawler maintains a queue of open node in a FIFO fashion, and queries the first node in the queue. `BFS` crawling is extremely popular, due partly to it its simplicity, but also because the obtained sample contains all nodes and edges on a particular region of the graph. Analysis on network samples obtained using a BFS crawl is presented (Mislove *et al.*, 2007).

Snowball Sampling (SB) The crawler acts similarly to `BFS`, except for only a p fraction of each node's neighbors is put into the queue (we set p to 0.5). This method can find hub nodes in a few iterations (Ahn *et al.*, 2007).

Depth-first Search (DFS) The crawler acts similarly to `BFS`, except that a node is selected in LIFO fashion.

Random Walk (RW) In each step, the crawler randomly moves to a neighbor of the most recently queried node. Nodes may be visited multiple times, but are only queried if they are still an open node. The results of `Random Walk` crawling came out on top (Leskovec and Faloutsos, 2006). On directed graphs, to prevent the crawler from becoming 'trapped', we use a teleport probability of 0.15, allowing the crawler to jump to an already-observed node.

Note that under the *in-out* query model, we assume that a crawler will make double queries on each node. This is because all of the considered algorithms are not designed for directed graphs. To adapt the above algorithms for the in-out query model, we assume that a crawler queries each selected node twice- once for the in-edges and once for the out-edges (Laishram *et al.*, 2017). Under the *paginated* and *partial* query models, the `BFS`, `SB`, `DFS` crawlers keep querying until all neighbors are seen. The `MOD`, `PR` and `OPIC` crawlers select the node with the highest observed centrality with at least k unobserved neighbors remaining (where k is the size of the query response).

3.5 The Effects of Network Structure on Crawler Performance

The overarching goal of our work is to investigate how the structural properties of a network can affect the performance of various crawling algorithms. As demonstrated in Ye *et al.*, 2010, the performance of different crawlers may vary by the network. This variance is surely due to differences in structure; but *which* properties are important, and *how* do they affect crawler performance?

3.5.1 Structural Properties of Interest

We hypothesize that *the performance of crawling methods strongly depends on how well a crawling algorithm can move between different regions of the graph.* At a high level, if a crawler has difficulty in transitioning between regions of the graph, it may become 'trapped' in one area, and repeatedly see the same nodes returned in response to its queries. Because the goal considered in this paper is that of node coverage, this is effectively a waste of budget. To verify our hypothesis, we select three network structural properties:[2]

Community Separation: A community is a subgraph with dense intra-connections and sparse inter-connections. We find communities using the Louvain method Blondel *et al.*, 2008, and then use the modularity Q of the detected partition to measure how well-separated the communities are Newman, 2004. The higher the modularity, the stronger the separation between communities, and so a crawler may be more likely to get trapped in a region.

Average Degree: We compute the average degree of nodes in the network.[3] If average degree is high relative to community size, this indicates that nodes are likely to have many connections outside their own community, making it easier for a crawler to move between regions.

Average Community Size: Finally, we consider the average community size (in terms of number of nodes) of the communities found using the Louvain method. As described earlier, this property is useful when taken together with average degree.

3.5.2 Properties of Real Networks

As we will see, the above three structural properties have a large effect on the comparative performance of the various crawling methods. However, in a real-world setting, one would not know these network properties ahead of time; so how can one use these results in practice?

As is well-known from the network science literature, networks of the same type (e.g., social, hyperlink, etc.) tend to have similar properties (Newman, 2003; Mislove *et al.*, 2007; Bonner *et al.*, 2016; Boccaletti *et al.*, 2006). For example, Boccaletti et al. show that P2P networks have an average degree of around 4-6, while Mislove et al. show that the average degree of OSNs ranges between 15-140. Numerous studies have investigated specific properties of other network categories, including the degree distribution, size of LCC, and diameter of the WWW (Broder *et al.*, 2000; Serrano *et al.*, 2007), the average degree and path length of OSNs (Mislove *et al.*, 2007; Ahn *et al.*, 2007) and community structure of citation networks (Chen

[2] We explored other properties, such as clustering coefficient, but these three emerged as having the greatest effect on performance.

[3] We also look at median degree, since the real networks have degree distribution follows power-law. However, it does not show any different in terms of the results.

and Redner, 2010). One cannot reasonably expect a single crawling method to be the best on every network under different query responses. In Section 7, we provide a set of guidelines for users on selecting a suitable crawling method when network type is known.

4 Experimental Setup

We now evaluate the effect of network properties on crawler performance through a series of experiments on synthetic and real networks. First, we perform a set of experiments on synthetic networks with carefully controlled properties, and investigate how changes in these structural properties affect performance of the crawling algorithms. Next, we perform another set of experiments on real networks, and use the results to validate the observations that we see on synthetic networks. For each set of experiment, we consider the five different query responses: *complete*, *paginated*, *partial*, *in-out* and *out* responses.

We adopt the LFR network model (Lancichinetti *et al.*, 2008) for generating networks. This model allows us to generate undirected or directed networks with desired properties including number of nodes, average degree, power-law exponent, community size, community mixing and etc. We set each generated network to have 5000 nodes with a maximum degree of 300. Then, we vary the value of three network properties; average degree, community size, and community mixing μ (μ has a range between 0 and 1, and indicates the fraction of edges that link to nodes outside the community).

Community mixing μ and modularity Q are related. Networks with high μ will have low *modularity* and vice versa. Higher values of μ indicate overlapping community structure. For our experiments, we vary the value of μ from 0.1 to 0.9, d_{avg} from 7 to 200, and CS_{avg} sizes from 100 to 2500. To reduce the effects of randomness, for each parameter setting, we generate 10 networks. We consider the query budgets up to 1000 queries (20% of total nodes).

We categorize the eight crawling methods into three groups. These groups correspond both to how the methods work and, as we will see, their performance on various networks. The methods in each class are:

G1: (Node Importance-based) `MOD`, `OPIC` and `PR`.
G2: `Random walk`
G3: (Graph Traversal-based) `BFS`, `DFS`, `SB`, `Rand`.

Throughout the figures in this paper, we use colors to represent the different methods, and different linetype to represent the different groups. '*dashed*', '*dotted*' and '*solid*' lines represent G1, G2 and G3, respectively.

5 Experiments on Synthetic Networks

We first analyze crawler performance on undirected networks, under three query response models- *complete*, *paginated*, and *partial*- and then consider directed networks, under two query response models- *in-out* and *out*.

5.1 Responses on Undirected Graphs

5.1.1 Complete Response

Recall that in the complete response query model, all neighboring nodes are returned when a node is queried. We plot results for each method in Figures 1-3.

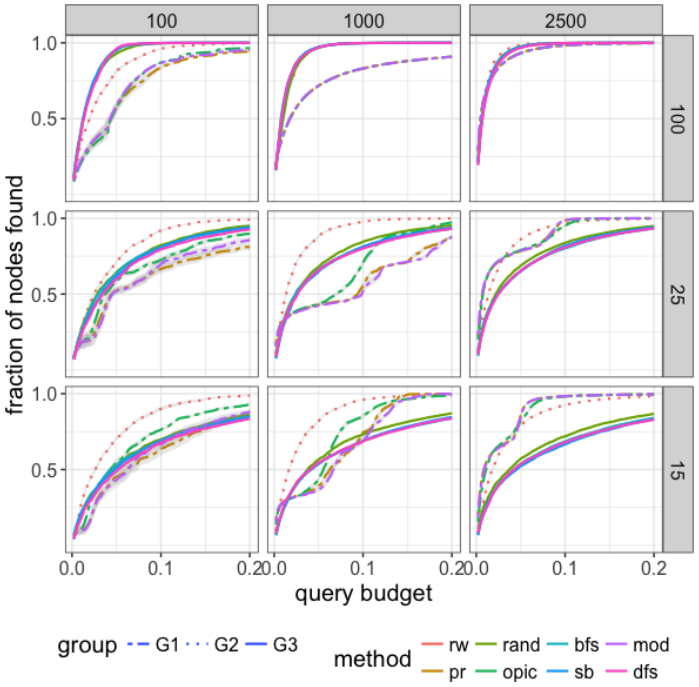


Figure 1: **[Best viewed in color] Complete Resp.**: Results on networks with different values of average degree and community size (μ=0.1). G2 is stable. G1 and G3 performance improves as community size and average degree increases, respectively.

First, we consider the case where networks have a clear community structure (high modularity, low μ: sharp community borders with few edges between communities), and average degree and community size are varied. Results are shown in Figure 1.

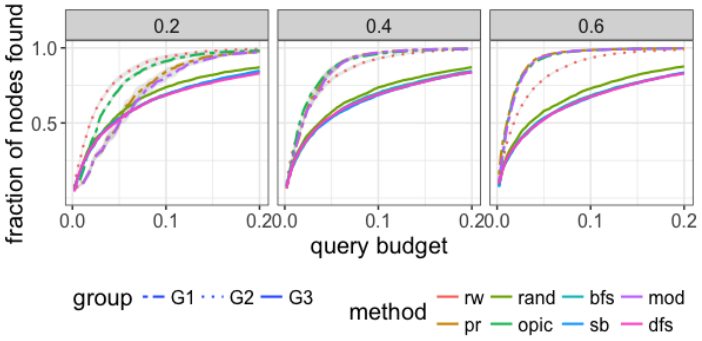


Figure 2: **[Best viewed in color] Complete Resp:** Results on networks with different values of μ (d_{avg}=15, CS_{avg}=300). G1 methods improve as μ increases.

The outer axes indicates different values of the test properties. The outer x-axis represents the increasing in average community sizes (100-2500 nodes). The outer y-axis represents the increasing in average degree (15-100). The axes of the inner plots indicate the fraction of nodes queried (x-axis) and fraction of nodes observed (y-axis).

Table 1: Categorization and summary of the performances of crawling algorithms on networks under the *complete*, *paginated*, *partial* and *in-out* response models. Results for the *out* model are presented in Table 2.

Property	G1: Node Importance-Based	G2: Random Walk	G3: Graph Traversal-Based
Community Separation	Performance improves when community overlap is high, i.e. high μ.	Stable	Stable
Average community size	Strong performance when communities are large if μ is low. Community size does not matter if μ is high.		
Average degree	Strong performance when average degree is extremely low (<10) even if μ is low. Otherwise, stable		Performance improvement when average degree increases.
Best Method in Group			
Complete	**MOD**	**RW**	**BFS**
Paginated	**OPIC/MOD**		**SB**
Partial	**OPIC**		**SB**
In-out	**PR/MOD**		**BFS**

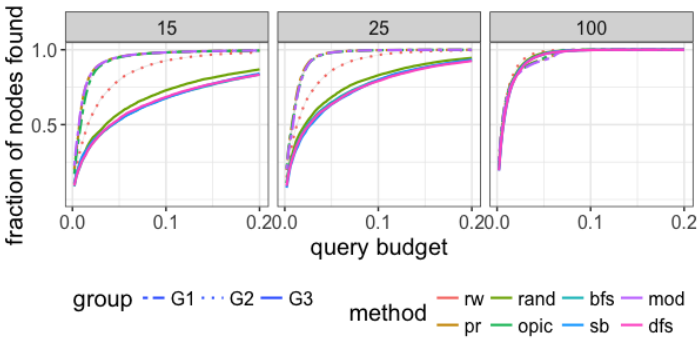


Figure 3: **[Best viewed in color] Complete Response**: Results on networks with different values of average degree (μ=0.6, CS_{avg}=300). G1 is the top performer. The performance of G3 methods improve as average degree and community size increases.

Next, in Figure 2, we demonstrate the case when community mixing is varied, and average degree and community size are fixed at 15 and 300, respectively. Low community mixing indicates that networks contain sharp and clear community structure, while high mixing indicates that networks have overlapping community structure. Finally, Figure 3 depicts the case when networks have overlapping community structure (many edges crossing between communities), while average degree is varied and community size is fixed at 300. For brevity, we cannot show results for all parameter settings, but the depicted results are representative of the full set of results. We draw several conclusions and summarize in Table 1.

G1 - Node Importance-based methods: Methods in this group select a node with high observed centrality (e.g., degree or PageRank). The performance of these methods tends to be similar. As intended by the creators of these algorithms, querying nodes with high observed centrality is likely to make the crawler discover many new nodes, since these are hub nodes. Indeed, the performance of these methods significantly improves as the size of the community is increased: in this case, a crawler can stay in one region of the graph for a long time without exhausting the set of new nodes to observe. However, G1 methods show reduced performance when μ is low (fewer connections between communities), and communities are small. This is because the crawlers tend to get trapped in a single region of the graph, and because there are few edges crossing between communities, the crawlers cannot easily move across to new communities. They thus suffer from diminishing marginal returns: even though they do not query the same node multiple times, they query nodes with similar neighborhoods, and so observe redundant information.

Interestingly, on the networks with extremely low average degree, we observe that G1 methods perform worse than both G2 and G3 for low query budgets, but their performance rapidly increases, and these methods are top performers for high query budgets. We observed this behavior on generated networks with low community mixing and with average degree less than 10.

To conclude, G1 methods are the best performers when μ is high (many edges between communities). These crawlers can easily move between communities as shown in Figure 2 and 3.

G2 - Random Walk: Our results show that this is the most stable method. Its performance seems to be unaffected by the considered properties. It is able to freely move between regions, even if community mixing is low. Sample results are shown in Figure 1 and 3.

G3 - Graph Traversal-based methods: Our results suggest that methods in G3 are not meaningfully affected by community size. The crawler can move between regions of the graph by uniformly expanding the sample frontier. As shown in Figure 1, G3 performs better when average degree increases (moving up along y-axis) and become top performers on networks with large average degree.

5.1.2 Paginated Response

In this section, we describe the results of each crawler on generated networks with paginated response. In the paginated query model, only k neighboring nodes are returned for each query on a node. We observe that results are similar to those of the complete response query model. A summary of how structural properties affect each method is shown in Table 1.

G1 - Node Importance-based methods: These methods tend to exhibit similar behavior as in the case of the complete query model. They exhibit excellent performance and are the top performers in two cases; 1) when communities are overlapping and 2) when average community size is high or average degree is extremely low, even if communities are not overlapping. Examples are shown in Figures 2 and 3.

G2 - Random Walk: As before, this crawler is very stable, and its performance appears to be independent of these properties.

G3 - Graph Traversal-based methods: The performance of methods in this group seems to be unaffected by modularity and average community size, but is affected by average degree. Results show that these crawlers have an performance improvement on networks with high average degree. We observe that `Snowball sampling` is the best among this group.

5.1.3 Partial Response

Next, we present the results of crawling methods on the generated networks under the partial response model. The partial query model is similar to paginated response, in that only k neighboring nodes are returned after each query. However, nodes are returned randomly, thus, the crawler can see the same neighbor from different queries. A summary of how the structural properties affect each method in this scenario is shown in Table 1.

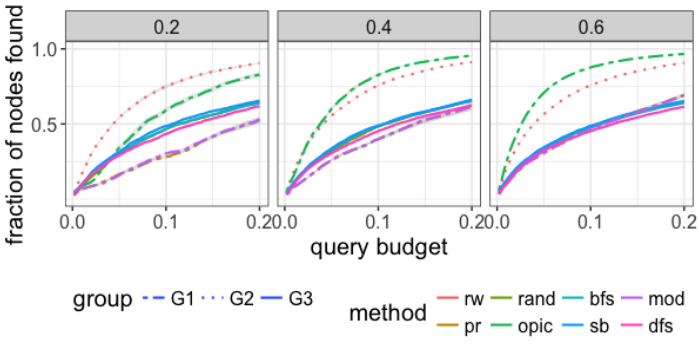


Figure 4: **[Best viewed in color] Partial Response:** Results on networks with different values of μ (d_{avg}=15, CS_{avg}=300). G1 methods improve as μ increases, but they generally perform similarly to methods in G3.

Under this query model, we observe some different behaviors in term of performance of these methods. Firstly, we observe that average degree has a small effect the Random Walk performance. Next, the performance of G1 is affected by community mixing as expected, however, all methods in G1 except OPIC can perform as good as methods in G3. Lastly, we observe that average degree and community size have less effect on G1 performance.

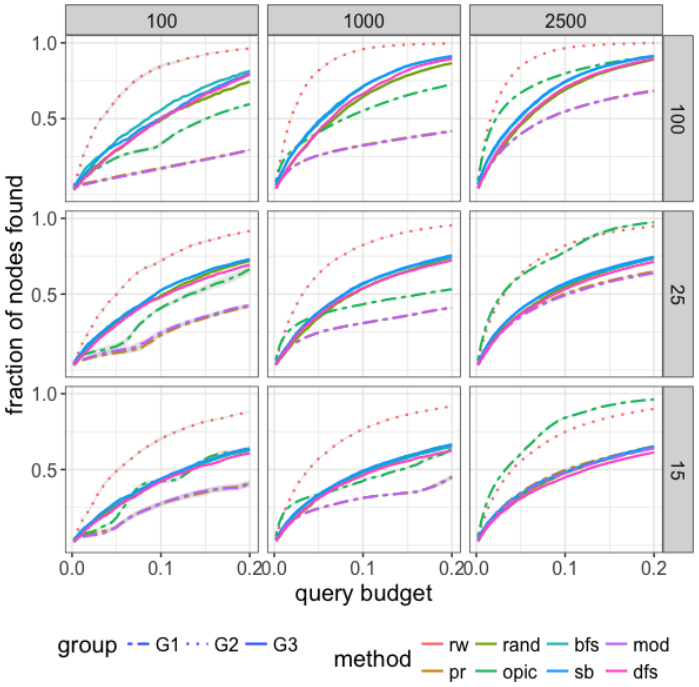


Figure 5: **[Best viewed in color] Partial Resp:** Results on networks with different average degree and community size (μ= 0.1). G3 methods improve when average degree increases. G1 performance improves when community size increases, they seem to be the worst performers.

G1 - Node Importance-based methods: The performance of these methods slightly improves when community mixing increases as shown in Figure 4. In Figure 5, G1 performance slightly increases as average degree or average community size is increased. However, their overall performance is worse than methods in G3. These methods do not perform well because the correlation between the observed centrality measure and actual centrality measure is very low: e.g., a node with high observed degree in the sample does not necessarily have high true degree. When a crawler queries the same node multiple times, it is likely to retrieve duplicate nodes from different queries. We observe that many of the open nodes tend to have the same observed degree, and so due to this low correlation, observed degree is not useful for distinguishing between medium- and high- degree nodes. On one hand, if a crawler happens to query on a node with (true) medium degree, a crawler needs to spend only a few queries, but the payoff (i.e., number of nodes returned) is low. On the other hand, if a crawler happens to query a node with extremely high (true) degree, it will retrieve many neighbors, but extends a large amount of budget because many duplicates are returned.

G2 - Random Walk: As before, the performance of the Random Walk crawler is still stable and unaffected by any of the considered properties. On networks with low community mixing, regardless of the other two properties, this method exhibits good performance, and is generally the top performer.

G3 - Graph Traversal-based methods: Similar to previous results under other query models, these methods show improvement as average degree is increased, but are mostly unaffected by community mixing and community size.

5.2 Responses on Directed Networks

We present the experiment results on synthetic directed networks under the *in-out* and *out* response models.

5.2.1 In-out Response

Under the *in-out* query response model, a crawler must query each node twice to obtain all of its edges: once to obtain its incoming edges and again to obtain its outgoing edges. We observe that results are similar to those on undirected networks under the complete response model. A summary is shown in Table 1.

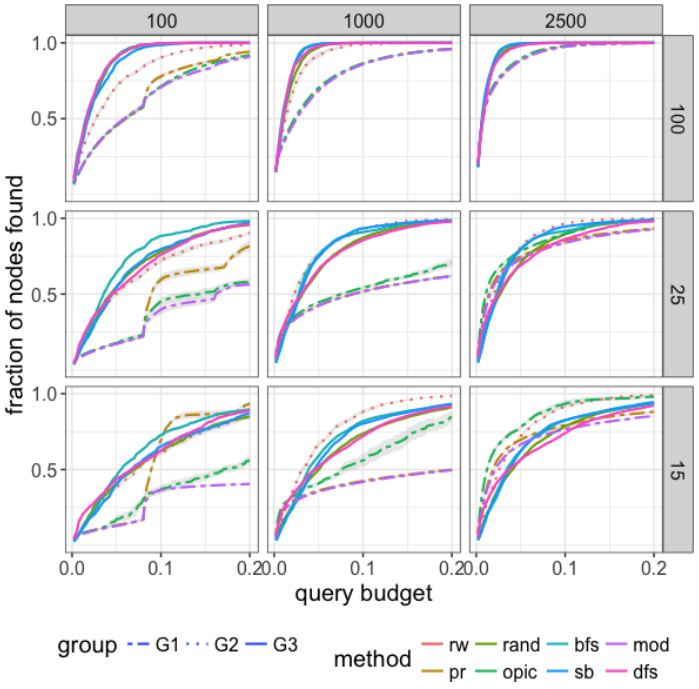


Figure 6: **[Best viewed in color] In-Out Response:** Results on networks with different values of average degree and community size (μ=0.1). G1 and G3 performance slightly improves as community size and average degree increases, respectively.

G1 - Node Importance-based methods: As before, methods in this group show an improvement in performance as community mixing is increased. This is expected, because these methods behave similarly to crawls on undirected networks under the complete response scenario, except that the crawler must query each node twice. We also observe a slight improvement in performance when either average degree or average community size is increased on networks with low community mixing, as illustrated in Figure 6. Surprisingly, these methods are often the worst performers on networks with low community mixing.

G2 - Random Walk: As in previous cases, the performance of the G2 crawler is generally stable, though it slightly improves as average community size increases.

G3 - Graph Traversal-based methods: The performance of the G3 methods is not meaningfully affected by μ and CS_{avg}; however, they are affected by average degree, and tend to perform very well on the networks with high average degree. Surprisingly, G3 performance is as good as or better than G2 performance on networks with high average degree (d_{avg} = 100 in these experiments).

5.2.2 Out Response

Here, a crawler is only able to request the edges outgoing from a node. Results under this model are somewhat different than those observed earlier, though there is no difference in terms of performance for these methods on networks with high community mixing, as illustrated in Figure 7. The summary is shown in Table 2.

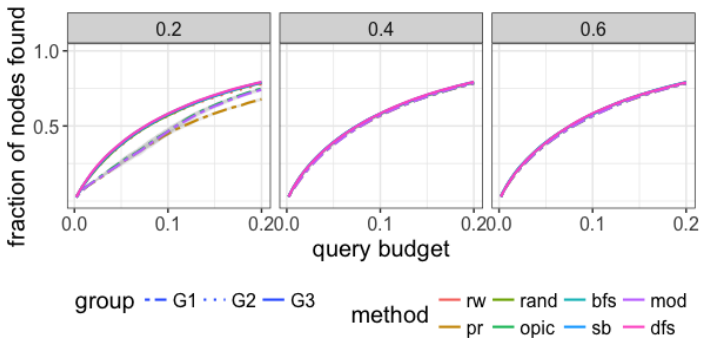


Figure 7: **[Best viewed in color] Out Response:** Results on networks with different μ (d=15, CS=300). There is no difference in terms of performance for these methods on the networks with high community mixing.

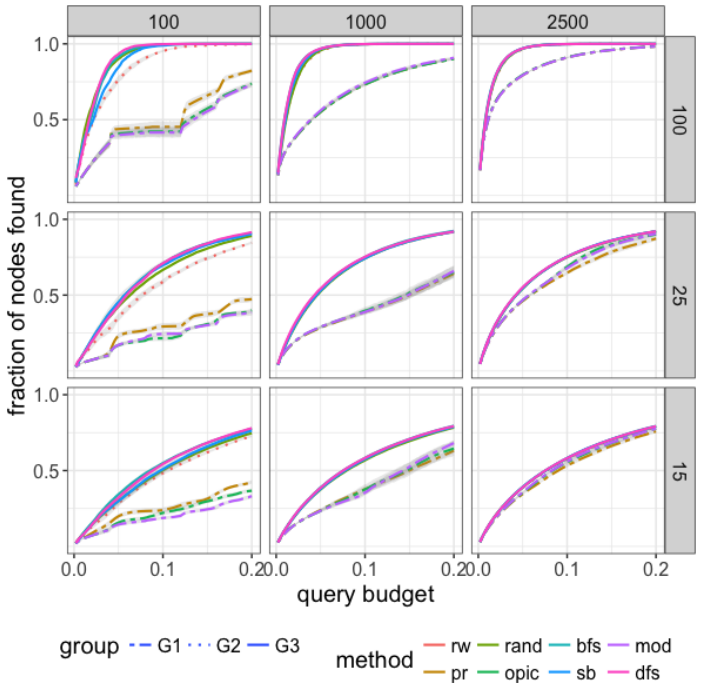


Figure 8: **[Best viewed in color] Out Response:** Results on networks with different values of average degree and community size (μ=0.1). G2 and G3 performance slightly increases when d_{avg} increases. G1 performance slightly improves as CS_{avg} increases, but it seems to be the worst performer in most cases.

Table 2: Categorization and summary of algorithm performance on directed networks under the *out* response model.

Property	G1: Node Importance-Based	G2: RW	G3: Graph Traversal-Based
Community Separation	Performance improves when community mixing is high.	Stable	Stable
Average community size	The performance slightly improves when average degree increases.		
Average degree	The performance slightly improves when average degree increases.	Performance improvement when average degree increases.	
Best Method	**MOD/PR**	**RW**	**BFS**

G1 - Node Importance-based methods: In Figure 8, the performance of methods in G1 improves when average degree or average community size increases on networks with low μ (few connections between communities), however, methods in G1 do not perform well compared to other methods. We see that the `PR` crawler is the best (among the crawlers in this group).

G2 - Random Walk: The G2 Random Walk crawler performance seems to be stable. We observe a slight improvement when average degree increases on networks with low μ (i.e., few edges between communities).

G3 - Graph Traversal-based methods: The performance of methods in G3 improves when average degree increases. In contrast to previous results, G3 methods perform very well and seem to be top performers.

6 Real World Networks

The previous experiments show that the major factor in the performance of each method is the ability to transition between different regions of the graph. Here, we consider the main observations from the previous section, and evaluate the extent to which they hold on real networks.

6.1 Experimental Setup

To validate our observations, we perform three sets of controlled experiments. Each set contains two pairs of networks ('P_1' and 'P_2'), and each network pair consists of networks that are similar with respect to two of the three properties but very different with respect to the third; e.g., *Wiki-Vote* and *Twitter* networks have different modularity values (0.42 vs 0.81) but they have similar average degree and average community size (28.51 and 1177, respectively). Within each pair, we refer to one network with the higher value of the test property as the *'High'-valued network (Hi)* and the other one as the *'Low'-valued network (Lo)*. Table 3 shows network statistics. The properties on which the networks in the same pair differ are shown in bold. Dataset can be found at ‡ networkrepository.com (NR) and † snap.stanford.edu (SNAP).

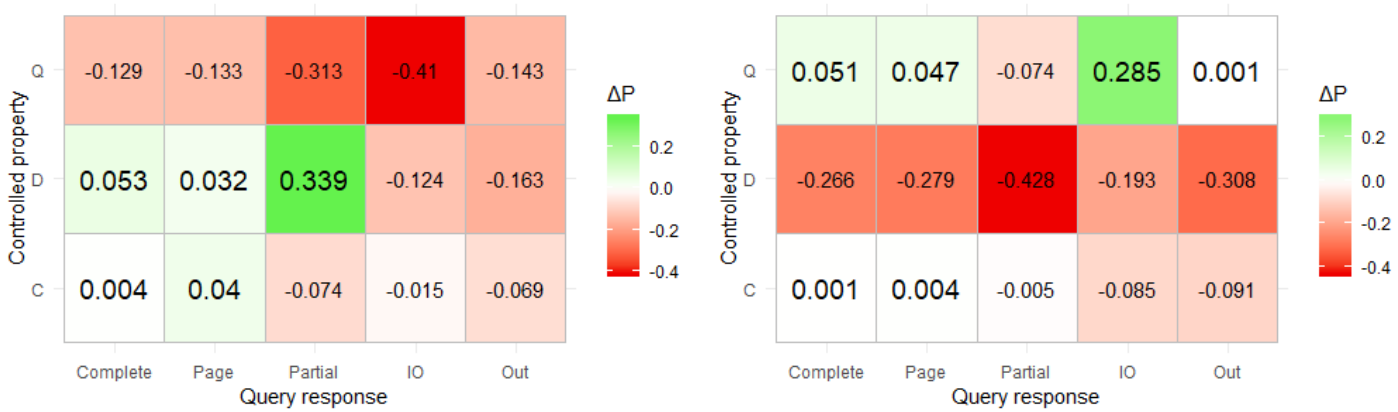
We find communities using the *Louvain method* Blondel *et al.*, 2008 and measure the strength of the detected communities using the modularity value Q, which we use as a proxy for community mixing. Note that the networks with high modularity values have low community mixing and vice versa ($\uparrow Q \Leftrightarrow \downarrow \mu$). The query budget is set to be 10% of the total nodes, as opposed to considering a fixed budget, because the selected networks may have different sizes.

In each experiment, we perform 10 trials and report the average result. We use the results of the best method in each group as a representative for each group. According to our earlier experiments, `Random walk` crawler is the least affected by these properties, we use it as a reference point to normalize the results of the other methods.

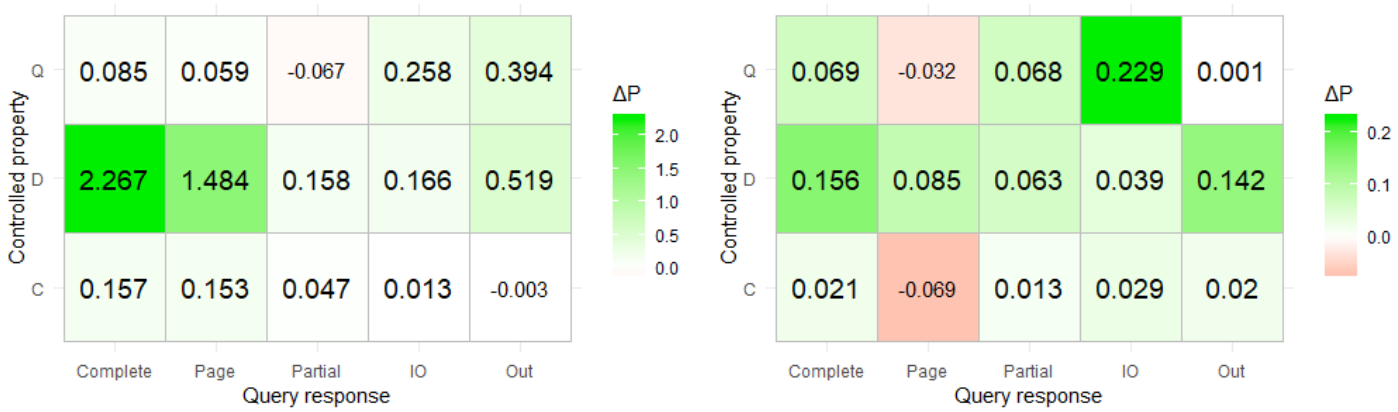
The results of these experiments are shown in Figure 9. Each row corresponds to a controlled property, and contains results on network pairs that differ with respect to that property. The columns represent different query responses. For each cell, the value indicates the changes in performance of the method x, ΔP_x, on *low-* vs. *high-* valued networks, defined as

Table 3: Network statistics of real-world networks used in the controlled experiments.

Test Prop.	Pair	Network	d_{avg}	CS_{svg}	Q
Undirected Networks (Complete, Page, Partial resp.)					
Q	P_1	**(Lo)** Wiki-Vote †	28.51	1,177.67	**0.42**
		(Hi) Ego-Twitter †	33.01	1,129.25	**0.81**
	P_2	**(Lo)** Brightkite ‡	7.51	274.10	**0.68**
		(Hi) MathSciNet ‡	4.93	594.09	**0.80**
CS_{avg}	P_1	**(Lo)** Github ‡	7.25	**83.68**	0.43
		(Hi) P2P-gnutella ‡	4.73	**1,276.76**	0.50
	P_2	**(Lo)** Shipsec1 ‡	24.36	**4,117.50**	0.89
		(Hi) Shipsec5 ‡	24.61	**5,252.15**	0.90
d_{avg}	P_1	**(Lo)** Amazon †	**2.74**	272.44	0.99
		(Hi) UK-2005 ‡	**181.19**	157.13	1.00
	P_2	**(Lo)** P2P-gnutella ‡	**4.73**	1,276.76	0.50
		(Hi) Bingham ‡	**72.57**	1,250.13	0.45
Directed Networks (In-out, Out resp.)					
Q	P_1	**(Lo)** bitcoinalpha ‡	7.48	157.29	**0.47**
		(Hi) Indochina-2004 ‡	8.38	147.50	**0.94**
	P_2	**(Lo)** rt-islam ‡	2.05	74.95	**0.63**
		(Hi) rt-obama ‡	2.13	82.36	**0.91**
CS_{avg}	P_1	**(Lo)** p2p-Gnutella25 †	4.82	**552.76**	0.49
		(Hi) p2p-Gnutella31 †	4.73	**1303.35**	0.50
	P_2	**(Lo)** p2p-Gnutella24 †	4.93	**662.45**	0.47
		(Hi) p2p-Gnutella31 †	4.73	**1303.35**	0.50
d_{avg}	P_1	**(Lo)** bitcoinalpha ‡	**7.48**	157.29	0.47
		(Hi) web-spam ‡	**15.68**	176.56	0.50
	P_2	**(Lo)** p2p-Gnutella30 †	**4.82**	814.36	0.51
		(Hi) Cit-HepTh ‡	**25.70**	782.86	0.65



(a) Performance changes ΔP of G1 methods (Left: Pair P_1 Right: P_2).



(b) Performance changes ΔP of G3 methods (Left: Pair P_1 Right: P_2).

Figure 9: **[Best viewed in color]** Results of controlled experiment. Each cell shows the changes in performance (ΔP) of G1 and G3 methods on *low*-valued and *high*-valued network. Positive value indicates an improvement in performance and negative value indicates a performance degradation as controlled property increases. Zero indicates performance is unchanged.

$\Delta P_x = P_{hi} - P_{lo}$, where P is the number of nodes found by method x divided by the number of nodes found by a `Random Walk` crawler. Positive values of ΔP_x indicate that the amount of nodes found by method x is greater on the high-valued network than the low-valued network, and negative values indicate the opposite. The differences in performance of G1 and G3 on both pairs of networks are shown in Figure 9a and 9b, respectively. We also report the percentage improvement above (or below) the number of nodes found by `Random Walk`, including the summary of all observations. Please see Table 6 in the Appendix for full details.

6.2 *Experimental Results*

Obs1: The effect of structural properties on crawlers' performance is similar for all types of queried responses, with the exception of *out* response.

Figure 9 shows the change in each crawler's performance across properties, for the different query models. The value indicates how the performance changes when there is a change in controlled properties. We can clearly see the changes of *complete*, *partial*, *paginated* and *in-out* query responses are similar.

Obs2: Methods in G1 have excellent performance on networks with overlapping communities.

As expected, G1 methods generally perform well when Q is low. The results of P_1, when modularity is a controlled property indicate that the performance of G1 methods drop when modularity increases, shown in Figure 9a (left).

Obs3: Methods in G1 perform well on networks with extremely low average degree even if Q is high.

On the other hand, G1 methods also perform very well on networks with extremely low average degree ($d_{avg} < 10$) even if modularity is high. We can see the consistent results for every pair P_2 when modularity is a test property on all responses in Figure 9a (right).

Obs4: The performance of methods in G1 improves on undirected networks with larger community size even if modularity is high.

On undirected networks with complete and paginated response, we observed that community size affects the performance of G1. Networks with larger community size seems to improve the G1 performance. However, this property does not seem to affect G1 performance on networks with other responses.

Obs5: `RW` crawler is the best under *partial* response.
The Random Walk crawler seems to be the best method on networks with a partial response, as we observed from synthetic networks. This also holds on real-world networks. As illustrated in Table 6, G1 and G3 have negative normalized performance, meaning that these crawlers' perform worse than Random Walk.

Obs6: Average degree affects the performance of G3.
In Figure 9b, G3 methods show a performance improvement on networks with higher average degree under every query response.

Obs7: G3 methods are generally the weakest.
As we observed on the synthetic networks, the performance of methods in G3 comes in last. This also holds on real-world network, as seen in Table 6.

7 Guidelines for Users

When collecting network data, the structural properties of the network are not known in advance. How can a data collector decide which crawler to use?

Here, we demonstrate that we can select a crawling method by using the network domain. Networks of the same type tend to have similar properties, and so it is possible to make reasonably accurate generalizations about the relevant structural properties. In addition, our guidelines cover different query responses from real application scenarios; e.g., most of the APIs provided by OSNs return paginated results, while only outgoing neighbors can be obtained when crawling web pages. For the sake of completeness, we include all combinations of network type and query response.

We categorize 21 networks into six network types: scientific collaboration networks, recommendation networks, Facebook100 networks, Web (hyperlink) networks, and technological networks (router-level network topology). Although the Facebook100 networks are online social networks, we consider them as a separate categories due to the restricted nature of the Facebook100 networks. These networks represent early versions of the Facebook network, dating to the period when universities each had separate Facebook networks. All nodes are thus members of the same university population, as opposed to modern online social networks, which include a much more diverse population. Due to this membership restriction, the Facebook100 networks exhibit very strong community structure, in contrast to the fuzzier structure one would expect from an OSN. All networks statistics are listed in Table 4. Datasets are taken from SNAP (†) and NR (‡).

Again, the maximum query budget is set to be 10 percent of the total number nodes. For standardization, we set the number of returned nodes for paginated and partial response to be the mean of the average degree across networks in that group. 10 trials are performed for each method and depict the mean and standard deviation of the percentage of nodes. A summary is shown in Table 5. Full results are show in Table 7 and 8.

Newman suggests that networks with $Q \geq 0.3$ have a strong community structure (Newman, 2004). From Table 4, OSNs contain overlapping community structure, indicated by their having the lowest modularity of all considered types. This is because people can be part of several groups in real life; e.g., group of friends, family, co-workers, etc. As shown in the Table 4, all Facebook networks indicate a strong community structure ($Q \geq 0.5$). As expected, G1 methods perform well on these OSNs, because they can freely move between regions. Other network types have higher modularity (0.4-0.9), so, the performance can be determined by average degree and community size.

Table 4: Categories of the real-world networks and their structural characteristics.

Type	Network	d_{avg}	CS_{avg}	Q	Properties
Undirected Networks (Complete, Page, Partial)					
Collab.	Citeseer ‡	7.16	988.35	0.90	Low degree, medium-sized and clear communities
	Dblp-2010 ‡	6.33	739.91	0.86	
	Dblp-2012‡	6.62	1248.35	0.82	
	MathSciNet ‡	4.93	594.09	0.80	
Recmnd.	Amazon ‡	2.74	272.44	0.99	Low degree, small clear communities.
	Github‡	7.25	83.68	0.43	
FB100	OR ‡	25.77	1074.44	0.63	High degree, large and clear communities
	Penn ‡	65.59	2186.11	0.49	
	WestOhio ‡	25.77	856.65	0.63	
OSNs.	Themarker ‡	29.87	458.90	0.31	High degree, small-to-medium-sized and fuzzy communities
	BlogCatalog ‡	47.15	1455.48	0.32	
	Catster ‡	73.22	1294.14	0.38	
Directed Networks (In-Out, Out)					
Web.	Arabic-2005 ‡	21.36	115.86	1.00	High degree, medium-sized and fuzzy communities
	Italycnr-2000 ‡	17.36	1134.34	0.91	
	Sk-2005 ‡	5.51	338.22	0.99	
	Uk-2005 ‡	181.19	157.13	1.00	
Tech.	P2P-gnutella ‡	4.73	1276.76	0.50	Low degree, large clear communities
	RL-caida ‡	6.37	856.12	0.86	
OSNs. (directed)	Slashdot ‡	10.24	173.87	0.36	High degree, small-to-medium-sized fuzzy communities
	Ego-Twitter †	90.93	2038.33	0.51	
	Wiki-Vote †	28.51	1009.43	0.42	

Table 5: Summary of algorithm performance. Algorithms perform similarly within the same category.

Type	Best Method		
	Comp.	Page	Part.
Undirected Networks			
Collaboration: low d_{avg}, medium CS_{avg}, high Q	G1	G1	G2
Recommendation: low d_{avg}, low CS_{avg}, high Q			
FB100: high d_{avg}, high CS_{avg}, high Q	G2	G2	
OSNs: high d_{avg}, lo-med CS_{avg}, low Q	G1	G1	G1

Type	Best Method	
	In-Out	Out
Directed Networks		
Technological: low d_{avg}, high CS_{avg}, high Q	G1	G3
Web: high d_{avg}, medium CS_{avg},low Q	G2	G2
OSNs (directed): high d_{avg}, high CS_{avg}, high Q	G1	

7.1 *Undirected Networks*

We first consider the network categories with high community separation (high Q). Here, we examine collaboration and recommendation networks. Both of these categories exhibit a

large average community size of at least approximately 50 times larger than their average degrees ($d_{avg} < 10$). These networks have clear community structure and low average degree, and as expected from earlier experiments, G1 methods perform very well under the *complete* and *paginated* models. In contrast, Facebook networks have communities only 30 times larger than their average degrees. On networks with smaller communities, the performance of methods in G2 are the best under all query response models. On networks with partial response, as suggested by our earlier experiments, the performance of the G2 method outperforms other methods. It is the best method to use on these types.

7.2 *Directed Networks*

On technological networks, the ratio of community size to average degree is approximately 200, indicating large communities. As expected, the performance of G1 is the best on networks under the *in-out* response model. In contrast, web networks have small communities- only approximately 35 times larger than average degree. As predicted by our earlier results, the G2 method works the best in this case. In addition, the G2 method is also the best on web and online social networks under the *out* response model. Finally, methods in G3 seems to perform slightly better than others on technological networks under the *out* response model. All the results are consistent with the results in previous experiments.

8 Major Takeaways

We have presented a wide variety of results across different types of networks and query models. We make several common observations. First and foremost, community mixing has a strong effect on the performance of methods in G1, which query nodes with high observed centrality (degree or other). The G1 methods are able to quickly discover a large number of nodes, but when μ is low, these methods risk becoming trapped inside a community. This occurs because even if a node from another community is observed, it is on the periphery of the observed sample, and so has low centrality. These methods repeatedly query nodes in the same region, but if the network has low mixing between communities, the queried nodes are likely to have similar neighborhoods. This results in the same nodes being observed over and over again, leading to diminishing marginal returns and thus reduced node coverage, and much of the budget is spent before the crawler moves to a new region. In contrast, if μ is high, then nodes from outside the starting community can reach high observed centrality, and are then queried. An exception to this observation generally occurs if the average community size is high relative to the average degree. This behavior is demonstrated on the real OSNs, which contain community structure with low modularity values. On these networks, the G1 methods tend to be best.

Secondly, regardless of the query response model, the considered properties have little effect on the performance on Random Walk, which is consistently a good performed. The Random Walk crawler, unlike the G1 methods, is able to easily escape dense regions of the network, because the crawler selects the next query node randomly from the neighbors of the current visited node. We see this behavior on the Facebook networks ($Q \geq 0.5$), as well. Finally, average degree has the most effect on methods in G3. Higher average degree tends to increase the performance of G3 methods, which do not encounter difficulty in moving between regions, because the crawler uniformly expands the sample frontier.

Limitations: Our experiments are conducted on networks downloaded from SNAP and Network Repository. For the most part, these networks themselves represent samples of larger networks. To the best of our knowledge, the FB100 dataset, which contains all friendships between users from different universities in 2005, is the only set of 'complete' networks. This data was provided directly by Facebook (Porter, 2011). However, because these networks are from an early point in Facebook's history, they may not be accurate representations of the current Facebook network.

Other networks are collected by crawling the original networks, where the crawling method is often not publicly stated. The properties of these collected network may not accurately reflect the the actual properties of the whole underlying network, but it has been shown that some network properties are self-similar, which means it has same statistical properties at many scales (Song *et al.*, 2005; Serrano *et al.*, 2008). Although, these samples may not be perfect representations of the underlying network, they have been used to capture the community structure (Maiya and Berger-Wolf, 2010; Salehi *et al.*, 2012), degree distribution (Rezvanian and Meybodi, 2015) or clustering coefficient (Leskovec and Faloutsos, 2006; Rezvanian and Meybodi, 2015) of the original networks.

9 Conclusion

We evaluated the performance of crawling algorithms on the goal of maximizing node coverage with respect to three network properties: community separation, community size, and average degree. We defined five query responses based on real data collection scenarios. We performed a set of controlled experiments on synthetic and real networks. We demonstrated that the performance of crawling methods highly depends on the network properties. In particular, their performance is largely dependent on the ease with which the method is able to transition between different regions of the graph. Lastly, we showed how a user can select an appropriate crawling method based on the network type and queried response.

Appendix A

Table 6 shows the percentage improvement above (or below) the number of nodes found by `Random Walk` crawler of both pairs 'P_1' and 'P_2' from the experiments in Section 6. Each row corresponds to a network property, and contains results on network pairs that differ with respect to that property. The columns represent different query responses. Each cell shows the performance improvement of G1 and G3 as compared to the performance of `Random Walk`. The arrow indicates how the performance changes ($\Uparrow$ improves or $\Downarrow$ degrades) when the considered property changes from low value to higher value.

As we expected, the performance of G1 drops when modularity increases as we can observe in pair P_1. However, in pair P_2, the performance of G1 improves even when modularity increases because the selected network pairs have extremely low average degree (all of them have the average degree less than

Table 6: Results of the controlled experiments. An arrow indicates how the performance changes when test property changes from Low to High (⇑: improve, ⇓: degrade, ≈: unchanged). In $\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right]$, x and y indicate the percentage improvement of Low- and High- valued networks, respectively ('+': outperform RW, '-': underperform RW).

	Networks $\left[\begin{smallmatrix}Lo\\Hi\end{smallmatrix}\right]^{undir}$ $\left[\begin{smallmatrix}Lo\\Hi\end{smallmatrix}\right]^{dir}$	**Comp**	**Page**	**Part**	**In-Out**	**Out**	**Observation**
Improvement of G1 vs. RW							
Q	P_1 $\left[\begin{smallmatrix}Wiki\\Twitter\end{smallmatrix}\right]\left[\begin{smallmatrix}Bitcoin\\Indo\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}7.62\%\\-5.24\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}7.19\%\\-6.13\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}-5.84\%\\-65.66\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}34.77\%\\-15.07\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}22.53\%\\12.11\%\end{smallmatrix}\right]$	The performance degrades on networks with higher Q. Excellent performance when d is extremely low even Q is high.
	P_2 $\left[\begin{smallmatrix}Brightkite\\MathSciNet\end{smallmatrix}\right]\left[\begin{smallmatrix}Islam\\Obama\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}12.47\%\\19.81\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}8.71\%\\13.41\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-26.05\%\\-14.15\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-1.73\%\\25.72\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}-0.01\%\\-2.45\%\end{smallmatrix}\right]$	
CS	P_1 $\left[\begin{smallmatrix}Github\\P2P\end{smallmatrix}\right]\left[\begin{smallmatrix}P2P25\\P2P31\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-71.52\%\\-70.32\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-66.57\%\\-66.18\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}-78.22\%\\-80.50\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}20.76\%\\12.23\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}12.05\%\\14.13\%\end{smallmatrix}\right]$	Performance improves when CS increases, but is unchanged on directed networks.
	P_2 $\left[\begin{smallmatrix}Shipsec1\\Shipsec5\end{smallmatrix}\right]\left[\begin{smallmatrix}P2P24\\P2P31\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}12.04\%\\13.47\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}12.33\%\\16.30\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}0.92\%\\0.95\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}12.92\%\\12.23\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}12.53\%\\13.13\%\end{smallmatrix}\right]$	
d	P_1 $\left[\begin{smallmatrix}Amazon\\UK\end{smallmatrix}\right]\left[\begin{smallmatrix}Bitcoin\\Webspam\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-0.40\%\\6.25\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}0.20\%\\3.38\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-23.20\%\\510.67\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}38.66\%\\14.05\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}22.53\%\\-4.14\%\end{smallmatrix}\right]$	Performance improves when d increases.
	P_2 $\left[\begin{smallmatrix}P2P\\Bingham\end{smallmatrix}\right]\left[\begin{smallmatrix}P2P30\\HepTh\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}10.14\%\\-14.38\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}15.58\%\\-12.32\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}16.84\%\\-31.84\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}4.37\%\\-15.78\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}24.45\%\\-25.11\%\end{smallmatrix}\right]$	
Improvement of G3 vs. RW							
Q	P_1 $\left[\begin{smallmatrix}Wiki\\Twitter\end{smallmatrix}\right]\left[\begin{smallmatrix}Bitcoin\\Indo\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-22.44\%\\-13.97\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}17.77\%\\-11.97\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-5.77\%\\-0.56\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-46.56\%\\-20.96\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-44.63\%\\-9.47\%\end{smallmatrix}\right]$	Performance is worse than RW. Performance slightly improves as Q increases.
	P_2 $\left[\begin{smallmatrix}Brightkite\\MathSciNet\end{smallmatrix}\right]\left[\begin{smallmatrix}Islam\\Obama\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-28.27\%\\-17.64\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}-17.64\%\\-20.54\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}0.20\%\\-2.42\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-9.01\%\\3.48\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}0.12\%\\-3.07\%\end{smallmatrix}\right]$	
CS	P_1 $\left[\begin{smallmatrix}Github\\P2P\end{smallmatrix}\right]\left[\begin{smallmatrix}P2P25\\P2P31\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}-20.53\%\\-27.68\%\end{smallmatrix}\right]$	⇓ $\left[\begin{smallmatrix}-19.58\%\\-27.71\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}-57.25\%\\-56.12\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}-15.75\%\\-12.32\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}-2.67\%\\-3.08\%\end{smallmatrix}\right]$	Performance is worse than RW. The performance seems to be unchanged overall.
	P_2 $\left[\begin{smallmatrix}Shipsec1\\Shipsec5\end{smallmatrix}\right]\left[\begin{smallmatrix}P2P24\\P2P31\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-31.64\%\\-15.91\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-23.36\%\\-7.49\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}-39.40\%\\-38.45.\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}-14.89\%\\-12.32\%\end{smallmatrix}\right]$	≈ $\left[\begin{smallmatrix}1.36\%\\1.12\%\end{smallmatrix}\right]$	
d	P_1 $\left[\begin{smallmatrix}Amazon\\UK\end{smallmatrix}\right]\left[\begin{smallmatrix}Bitcoin\\Webspam\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-15.18\%\\-0.87\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-8.25\%\\0.00\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-16.45\%\\-2.16\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-11.97\%\\-9.43\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-2.99\%\\6.80\%\end{smallmatrix}\right]$	Performance is worse than RW. Better performance when average degree increases.
	P_2 $\left[\begin{smallmatrix}P2P\\Bingham\end{smallmatrix}\right]\left[\begin{smallmatrix}P2P30\\HepTh\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}2.09\%\\334.34\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-95.20\%\\133.98\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-36.62\%\\-31.44\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-46.56\%\\-24.32\%\end{smallmatrix}\right]$	⇑ $\left[\begin{smallmatrix}-44.63\%\\14.77\%\end{smallmatrix}\right]$	

Table 7: **Undirected Networks -** Summary of the network characteristics and performance of algorithms.

Type	**Network**	**Complete**			**Paginated**			**Partial**		
		G1	**G2**	**G3**	**G1**	**G2**	**G3**	**G1**	**G2**	**G3**
Collab: Low d_{avg}, Medium CS_{avg}, High Q	Citeseer	**39.78 ± 0.85**	38.94 ± 0.46	32.29 ± 0.22	**39.71 ± 0.13**	38.86 ± 0.33	34.09 ± 0.21	7.06 ± 1.4	**12.84 ± 0.23**	5.23 ± 0.76
	Dblp-2010	**45.73 ± 0.07**	40.8 ± 0.37	28.84 ± 0.11	**43.03 ± 0.17**	41.29 ± 0.24	31.99 ± 0.18	9.96 ± 0.68	**13.53 ± 0.14**	5.44 ± 0.21
	Dblp-2012	**52.39 ± 0.08**	46.88 ± 0.2	38.26 ± 0.12	**49.08 ± 0.14**	47.27 ± 0.17	40.7 ± 0.15	12.38 ± 0.14	**14.18 ± 0.11**	7.57 ± 0.20
	MathSciNet	**51.15 ± 0.08**	44.94 ± 0.15	36.56 ± 0.13	**43.05 ± 0.33**	45.01 ± 0.15	39.83 ± 0.25	12.97 ± 0.70	**14.56 ± 0.07**	8.59 ± 0.27
Rec: Low d_{avg}, Low CS_{avg}, High Q	Amazon	**11.35 ± 0.18**	11.25 ± 0.09	11.64 ± 0.2	**11.66 ± 0.12**	11.39 ± 0.15	1.2 ± 1.21	7.23 ± 0.11	**9.32 ± 0.46**	4.94 ± 1.91
	Github	**66.15 ± 0.02**	58.34 ± 0.16	39.88 ± 0.13	**65.41 ± 0.07**	58.97 ± 0.12	45.11 ± 0.13	22.59 ± 0.26	**45.50 ± 0.20**	26.8 ± 0.11
FB: High d_{avg}, High CS_{avg}, High Q	OR	52.89 ± 1.06	**67.41 ± 0.30**	63.4 ± 0.06	53.33 ± 3.62	**66.78 ± 0.50**	63.86 ± 0.21	41.43 ± 5.48	**63.29 ± 0.65**	48.65 ± 0.17
	Penn	82.36 ± 0.72	**89.35 ± 0.29**	88.32 ± 0.04	80.67 ± 3.26	**89.35 ± 0.28**	88.5 ± 0.06	59.62 ± 4.62	**80.18 ± 0.50**	56.67 ± 1.50
	Wosn-friends	53.04 ± 1.13	**67.61 ± 0.36**	63.37 ± 0.1	52.97 ± 4.47	**66.60 ± 0.59**	63.93 ± 0.18	42.98 ± 4.58	**62.99 ± 0.52**	48.39 ± 0.31
OSNs: High d_{avg}, Low-to-medium CS_{avg}, Low Q	BlogCatalog	**94.99 ± 0.01**	94.21 ± 0.12	57.6 ± 0.53	**96.12 ± 0.03**	93.65 ± 0.21	63.58 ± 0.77	**29.72 ± 0.10**	28.18 ± 0.13	19.18 ± 0.81
	Themarker	**93.61 ± 0.00**	91.66 ± 0.14	58.4 ± 0.22	**93.82 ± 0.02**	90.86 ± 0.10	63.56 ± 0.61	**33.99 ± 0.15**	32.83 ± 0.12	21.69 ± 0.31
	Catster	**94.74 ± 0.01**	94.25 ± 0.05	69.77 ± 3.24	**94.93 ± 0.01**	94.82 ± 0.07	81.54 ± 0.77	**48.1 ± 8.81**	30.87 ± 0.17	10.92 ± 0.62

Table 8: **Directed Networks -** Summary of the network characteristics and performance of algorithms.

Type	**Network**	**In-Out**			**Out**		
		G1	**G2**	**G3**	**G1**	**G2**	**G3**
Tech: Low d_{avg}, High CS_{avg}, High Q	P2P-gnutella	**36.04 ± 0.12**	31.97 ± 0.24	27.73 ± 0.23	12.64 ± 1.62	12.48 ± 1.55	**13.16 ± 2.28**
	RL-caida	**36.38 ± 0.18**	29.45 ± 0.48	26.83 ± 0.11	**5.09 ± 0.00**	4.72 ± 0.00	**5.09 ± 0.00**
Web: High d_{avg}, Medium CS_{avg}, Low Q	Arabic-2005	9.33 ± 1.01	**9.90 ± 1.45**	8.58 ± 1.14	**0.51 ± 0.00**	**0.51 ± 0.00**	**0.51 ± 0.00**
	Italycnr-2000	10.07 ± 1.92	**19.34 ± 4.5**	17.01 ± 3.43	15.97 ± 0.11	**18.32 ± 1.80**	15.35 ± 0.33
	Sk-2005	9.30 ± 0.78	**10.01 ± 0.4**	8.38 ± 0.42	**0.39 ± 0.00**	**0.39 ± 0.00**	**0.39 ± 0.00**
OSNs: High d_{avg}, High CS_{avg}, High Q	Slashdot	**72.85 ± 0.01**	60.32 ± 0.21	39.5 ± 0.37	**79.03 ± 0.03**	**79.47 ± 0.37**	47.97 ± 0.28
	Ego-Twitter	**86.84 ± 2.30**	**86.26 ± 1.07**	77.21 ± 1.08	53.65 ± 12.83	**78.99 ± 7.15**	61.37 ± 3.45
	Wiki-Vote	**66.22 ± 1.21**	60.71 ± 1.00	46.95 ± 0.52	29.13 ± 0.77	**31.2 ± 0.26**	30.61 ± 0.11

10). When average community size is a controlled property, we see small changes for the *complete*, *paginated* and *partial* query models. This is because the selected networks have too large of a community size relative to the given query budget. Thus, we cannot observe substantial changes in performance here. Lastly, as expected, the performance of G3 methods increases as average degree increases.

Next, we show complete results of the experiments from Section 7. We provide the mean and standard deviation of the percentage of node coverage in Tables 7 and 8 for undirected and directed networks.

Acknowledgments

This work was supported by the U.S. Army Research Office under grant #W911NF1810047. The authors thank Jeremy Wendt of Sandia National Lab. for helpful conversations.

References

Abiteboul, S., M. Preda, and G. Cobena (2003). "Adaptive online page importance computation". In: *Proceedings of the 12th International Conference on World Wide Web*. ACM. 280–290.

Ahmed, N. K., J. Neville, and R. Kompella (2014). "Network sampling: From static to streaming graphs". *ACM Transactions on Knowledge Discovery from Data (TKDD)*. 8(2): 7.

Ahn, Y.-Y., S. Han, H. Kwak, S. Moon, and H. Jeong (2007). "Analysis of topological characteristics of huge online social networking services". In: *Proceedings of the 16th International Conference on World Wide Web*. ACM. 835–844.

Areekijseree, K., R. Laishram, and S. Soundarajan (2018). "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. 57–66.

Areekijseree, K. and S. Soundarajan (2018). "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. 164–169.

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

Blondel, V. D., J.-L. Guillaume, R. Lambiotte, and E. Lefebvre (2008). "Fast unfolding of communities in large networks". *Journal of Statistical Mechanics: Theory and Experiment*. 2008(10): P10008.

Boccaletti, S., V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang (2006). "Complex networks: Structure and dynamics". *Physics reports*. 424(4-5): 175–308.

Bonner, S., J. Brennan, G. Theodoropoulos, I. Kureshi, and A. S. McGough (2016). "Deep topology classification: A new approach for massive graph classification". In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. 3290–3297.

Broder, A., R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener (2000). "Graph structure in the web". *Computer networks*. 33(1-6): 309–320.

Chen, P. and S. Redner (2010). "Community structure of the physical review citation network". *Journal of Informetrics*. 4(3): 278–290.

Gjoka, M., M. Kurant, C. T. Butts, and A. Markopoulou (2009). "Unbiased sampling of facebook". *preprint arXiv*. 906.

Kurant, M., A. Markopoulou, and P. Thiran (2010). "On the bias of BFS". *arXiv preprint arXiv:1004.1729*.

Kurant, M., A. Markopoulou, and P. Thiran (2011). "Towards unbiased BFS sampling". *IEEE Journal on Selected Areas in Communications*. 29(9): 1799–1809.

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

Lancichinetti, A., S. Fortunato, and F. Radicchi (2008). "Benchmark graphs for testing community detection algorithms". *Physical Review E*. 78(4): 046110.

Leskovec, J. and C. Faloutsos (2006). "Sampling from large graphs". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 631–636.

Maiya, A. S. and T. Y. Berger-Wolf (2010). "Sampling community structure". In: *Proceedings of the 19th International Conference on World Wide Web*. ACM. 701–710.

Mislove, A., M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee (2007). "Measurement and analysis of online social networks". In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. ACM. 29–42.

Newman, M. E. (2003). "The structure and function of complex networks". *SIAM review*. 45(2): 167–256.

Newman, M. E. (2004). "Fast algorithm for detecting community structure in networks". *Physical Review E*. 69(6): 066133.

Porter, M. (2011). "Facebook100 Dataset". http://masonporter.blogspot.com/2011/02/facebook100-data-set.html.

Rezvanian, A. and M. R. Meybodi (2015). "Sampling social networks using shortest paths". *Physica A: Statistical Mechanics and its Applications*. 424: 254–268.

Robson, D. and H. Regier (1964). "Sample size in Petersen mark–recapture experiments". *Transactions of the American Fisheries Society*. 93(3): 215–226.

Salehi, M., H. R. Rabiee, and A. Rajabi (2012). "Sampling from complex networks with high community structures". *Chaos: An Interdisciplinary Journal of Nonlinear Science*. 22(2): 023126.

Serrano, M. A., D. Krioukov, and M. Boguná (2008). "Self-similarity of complex networks and hidden metric spaces". *Physical review letters*. 100(7): 078701.

Serrano, M. A., A. Maguitman, M. Boguñá, S. Fortunato, and A. Vespignani (2007). "Decoding the structure of the WWW: A comparative analysis of Web crawls". *ACM Transactions on the Web (TWEB)*. 1(2): 10.

Song, C., S. Havlin, and H. A. Makse (2005). "Self-similarity of complex networks". *Nature*. 433(7024): 392.

Ye, S., J. Lang, and F. Wu (2010). "Crawling Online Social Graphs". In: *Web Conference (APWEB), 2010 12th International Asia-Pacific*. IEEE. 236–242.

Katchaguy Areekijseree is currently a PhD student in the Department of Electrical Engineering and Computer Science at Syracuse University. His research mainly focuses on network sampling and network robustness. He received his BEng in 2012 and MEng. in 2014 from KMUTT.

Sucheta Soundarajan is an Assistant Professor in the Department of Electrical Engineering and Computer Science at Syracuse University. Her research interests include data mining and complex network analysis. She studies a variety of problems related to social network analysis, including network crawling, community and core structures, and adversarial social networks. She received her BS from The Ohio State University in 2005, her PhD from Cornell University in 2013, and held a postdoctoral position at Rutgers University.