December 2019

Network Sampling through Crawling

Katchaguy Areekijseree
Syracuse University

ABSTRACT

In recent years, researchers have increasingly used OSN data to study human behavior. Before such a study can begin, one must first obtain appropriate data. A platform, e.g, Facebook or Twitter, may provide an API for accessing data, but such APIs are often rate-limited, restricting the amount of data that an individual collects in a given amount of time. In order for the data collector to efficiently collect data, she needs to make intelligent use of her limited budget. Therefore, when collecting data, efficiency is extremely important. We consider the problem of network sampling through crawling, in which the data collectors have no knowledge of the network of interest except the identity of a starting node. The data collector can expand the observed sample by querying an observed node. While the network science literature has proposed numerous network crawling methods, 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.

Here, we show that the performance of network crawling methods is highly dependent on the network structural properties. We identify three important network properties: community separation, average community size, and node degree. In addition, we provide guidelines to data collectors on how to select an appropriate crawling method for a particular network. Secondly, we propose a novel crawling algorithm, called **DE-Crawler**, and demonstrate that it performs the best across different network domains. Lastly, we consider the scenario in which there is are errors in the data collection process. These errors then lead to errors in a subsequent analysis task. Therefore, it is important for a data analyst to know if a collected sample is trustworthy. We introduce a robustness measure called *sampling robustness*, which measures how robust a network is under random edge deletion with respect to sampling. We demonstrate that sampling robustness highly depends on the network properties and users can estimate sampling robustness from the obtained sample.

NETWORK SAMPLING THROUGH CRAWLING

By

Katchaguy Areekijseree
B.Eng in Computer Engineering King Mongkut's University of Technology Thonburi, 2012
M.Eng in Computer Engineering King Mongkut's University of Technology Thonburi, 2014

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer & Information Science & Engineering

Syracuse University
December 2019

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my doctoral advisor, Prof. Sucheta Soundarajan, for her endless encouragement, mentorship, enthusiasm, and immense knowledge. Her guidance and support help me in all the time of my study, research and my life in general. I could not imagine having a better doctoral advisor.

Besides my advisor, I would like to thank the rest of my research committee for their encouragement and insightful comments.

I also thank Jeremy Wendt of Sandia National Laboratories for thoughtful comments and conversations throughout my first project.

I thank my fellow labmates in Syracuse University Network Science Laboratory: Ricky Laishram, Pivithuru Wijegunawardana, Humphrey Mensah and Zeinab S. Jalali for their help, support, and all the fun we have had in the last 3-4 years.

Finally, I would like to thank my family and friends for their love. Most importantly, huge thank to my wife Sikana Tanupabrungsun for her full support and encouragement during the hard time.

Contents

List of Tables

List of Figures

CHAPTER 1
INTRODUCTION

In recent years, data analysts and researchers have become interested in the study of complex networks. Networks of interest span many domains including biological networks, the WWW, communication networks and online/offline social networks. For example, by studying social networks, we are able to gain deep insights into societal-scale human behavior, such as understanding how groups form, information spreads, target marketing or friendships dissolve. In bioinformatics, researchers are interested in finding network motifs (e.g., certain small subgraphs which frequently occur). Reserchers gain various intersting results by analyzing these motifs, e.g., the protein-protein interactions [4], hierarchical network decomposition [39] and the analysis of temporal gene expression patterns [40]. Similarly, finding motifs is also an essential task in linguistics. The linguistic networks can be constructed from the corpus (e.g., sentences, a set of words), where nodes represent words and the edges exists between two nodes if words co-occur in the corpus. The linguistic network has many applications. For example, it can be used for assigning the appropriate sense (e.g., word meaning) to a word in a sentence [27], dependency parsing [55] and textual entailment [35]. However, before one can attempt to explore such questions, one must first collect network data.

Over the past few years, many data analysts and researchers have chosen to collect network data from online platforms, such as Facebook or Twitter. These platforms are convenient for data collection, because they typically provide a public application programming interface or API to access the data. Web scraping or web crawling is an alternative way of collecting data from web pages. A data collector can write a computer program, which usually referred to "*spider*" or "*bot*", to access the WWW using Hypertext Transfer Protocol or HTTP. This spider will extract the links (e.g. urls lead to other webpages) and other information from HTML tags. The spider can move to other websites by maintaining the obtained urls.

In other cases, where an API is not valid or not available, the network data can be collected by traditional approaches such as interrogation, interviews, etc. For example, the authors proposed a method for finding people of interest in [77]. In this work, they focus on finding the terrorists that hide among other people on the networks. The network of terrorists can be discovered by capturing and interrogating the suspects.

As we can imagine, the process of collecting data can be expensive and time-consuming, which requiring a significant amount of budget (e.g., time, money, or resources). Therefore, when collecting data, the goal is to collect the data in such a way that it is worth our budget. Here, we refer to this process as **network sampling through crawling**. In this scenario, a data collector has 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 (e.g., a computer program makes queries through web API, an interviewer asks participants questions, a police officer interrogates the suspects), and thus expand the observed network. Throughout this dissertation, we use *network sampling* and *network crawling* interchangeably.

The process of data collection posses several challenges. First and foremost, the data collector has no control and knowledge about the network of interest (except a single seed nodes which will be the starting point of the data collection process), but can expand the sample by querying on the nodes that she obtains so far. Thus, one important question is: how can we make a smart decision or strategy about what would be the next node to be queried which will give us the highest efficiency? Secondly, there are many proposed crawling methods which have been used in the literature, but the research community lacks insight into how well these methods perform, and why. Existing work sometimes mentions that one method may work well on some particular networks but fails on other networks [8]. Thirdly, most of the previous work focus on the undirected networks [3, 9, 29, 69], however, many real networks have directional edges (e.g., Twitter network and the WWW). So, method which works on undirected network may not work well on directed networks [44]. Thus, it is quite unclear to the data collector on which method to use when she wants to collect the network data. Lastly, when collecting data, a data collector may usually face incompleteness from the query response in real scenarios (e.g., missing nodes or edges from each query). This missing information makes the sample incomplete. Then, if one performs an analysis on this sample, the result will lead to an inaccurate result. Thus, once we obtain the network sample, it is useful to know the quality of this sample before performing any further analysis. If the missing data has an effect on the quality of the sample, a data analyst can modify or fix the sample beforehand.

The purpose of this dissertation is to to address these challenges and fill all these gaps mentioned above. We believe that our work will help researchers and others in the field to have more insights about how to collect the network data.

1.1 Dissertation Outline

During the last two decades, researchers have proposed a large number of network crawling algorithms. A brief summary of related work on network sampling are provided in Chapter 2. However, due in part to the large number of choices, it is often difficult for data collectors to select a single crawling technique, and it is rarely clear which crawling method is the best to crawl a given network. Existing work in the literature has typically attempted to determine which crawling method is the 'best' overall; however, there is generally a lack of insight into why these supposed 'best' methods perform well on given networks. The need to understand the behavior of specific crawling algorithms across different network structures motivates us to investigate the network structural factors that affect crawler performance.

In Chapter 3, we investigate and study the effect of network structural properties on the performance of important network crawling algorithms. More specifically, we are interested in those network properties that govern the ability of a crawler being able to move between 'regions' of a graph. To evaluate a crawler's performance, we select the objective of maximizing the number of observed nodes (which we refer to as 'maximizing node coverage'). The objective of this chapter is to analyze, characterize, and categorize the the performance of various network crawling algorithms and demonstrate that these features have a strong effect on the performance of various crawling methods. In addition, we also provide guidelines on how a user can select an appropriate crawling method for a network from a given domain, even before observing specific properties of that network.

In Chapter 4, we propose a novel crawling algorithm, **DE-Crawler**, which is based on our observations from Chapter 3. From Chapter 3, we see that no existing methods work uniformly well across network types, and so we design **DE-Crawler**

to incorporate the best aspects of existing techniques. Our proposed algorithm is capable of seamlessly transitioning to different regions of the network regardless of network structure. The main ingredient that makes **DE-Crawler** successful is the ability of balancing $exploration$ and $exploitation$ when crawling the network. **DE-Crawler** consists of two important stages: *Densification* and *Expansion*. The *Densification* stage aims to discover nodes in the current region, while the *Expansion* stage aims to expand the sampled network by moving to another dense region. We test the performance of **DE-Crawler** on different network types and compare its performance against the performance of other crawlers. As a result, **DE-Crawler** is able to capture the best aspects of the existing algorithms, and achieves outstanding performance across domains.

In Chapter 5, we consider the scenario in which there are errors during the data collection process. These errors may come from a bug in a web crawler, mistakes made by respondents during the interview, or an adversary tampering with and altering the information exchanged between two parties. As one can imagine, these errors may lead to inaccuracy in a subsequent analysis task. In this chapter, we introduce a novel robustness measure called *sampling robustness*. Specifically, it measures how robust the network is under random edge deletion with respect to sampling. To the best of our knowledge, there are many works on network sampling, but none of them focus on robustness respect to network sampling. Here, we demonstrate that sampling robustness is strongly correlated with certain properties of the original network and of the obtained sample. We present four different performance measures for calculating the sampling robustness, depending on the goal of sampling. In addition, we investigate when different crawlers are used to generate the samples. Our goal is to allow data analysts or anyone who is interested in the study of the complex networks to measure the quality of the obtained sample from just a few set of parameters. Therefore, we present a regression model

for estimating sampling robustness of any network by considering the structural properties of the obtained sample. The conclusion is presented in Chapter 6.

The main contributions of this dissertation are: 1) We examine the effect of network structure on the performance of existing crawling algorithms, and show that the ability of a crawler to move between different dense regions of a network is critical to its success 2) Using these observations, we propose an algorithm that performs well across different categories of networks, and 3) We consider the scenario in which there is an error during the data collection which may lead to error or inaccuracy in the further analysis. We define a novel robustness measure called sampling robustness and introduce a model which can be used to estimate the robustness of any network by considering only the obtained samples.

CHAPTER 2
RELATED WORK

Work on 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 web scraping). 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. Due to space constraints and the vast amount of literature in this area, we cannot provide a complete overview of the topic, but refer the reader to the survey in [2] for a more detailed discussion.

2.1 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 [56]. Similarly, Ahn, et al. use a BFS crawler to collect data from CyWorld, a South Korean social networking site, and MySpace [3]. 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 [43]. The BFS crawler is easy to implement, thus, it has been applied in a large number of studies [15, 18, 78, 31, 32]. Other graph-traversal methods like Depth-First Search (DFS) and Snowball sampling have been used for analyzing on online social networks are demonstated in [20] and [3], respectively.

Random Walk crawler is an alternative crawler. It has been used a lot for crawling peer-to-peer networks [33, 74], the WWW [36, 22] and online social networks [64, 41, 30, 66, 60]. However, samples collected by Random Walk crawler are found to be biased towards high degree nodes in the graph. An unbiased approach based on Metropolis-Hastings Random Walks for undirected networks [29] and directed networks [75] are introduced, which they demonstrate that the proposed crawlers can balance the visiting frequencies between low and high degree nodes. Cheierichetti, et al. propose other variations of Random Walk for collecting uniform samples in [19]. In addition, they show a near-tight bound expressed of the algorithms in terms of parameters of the networks., e.g. average degree and the mixing time.

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 crawl-

ing networks to preserve community structure [69]. Likewise, Blenn, et al. present a crawler which nodes belong to the same community are crawled first, thus, the crawler crawls one community after another [11]. The experimental results show that this proposed method is capable of preserving community structure of the networks and its performance is better than BFS and DFS crawler. Bruno, et al. study the ability of different methods for preserving degree distribution [65]

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 [9, 8]. 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 MOD performance significantly outperforms other algorithms like BFS and RW on the task of node coverage. The authors acknowledge that MOD sometimes performs poorly, but leave that discussion for future work. Salamanos, et al. present a Rank Degree method [68]. The idea is similar to MOD crawler in which the nodes with the highest degree are selected. Experimental results show the evidence that the proposed crawler is good for node coverage and the samples preserve the node centrality and also *k*-core of the original network. The OPIC method, presented in [1], adopts a similar idea as MOD, except that it queries the node with the highest PageRank in each step. Experimental results in [1] and [8] show that both MOD and OPIC significantly outperform other methods. A multi-armed bandit crawler is proposed by Madhawa and Murata [50]. The results show that the proposed crawler outperforms other baselines. Laishram, et al. show that MOD does not work well on directed graphs and present PMD crawler [44] that predicts which k nodes are most likely to have the highest number of unobserved nodes. PMD works the best on directed networks.

2.2 Algorithms for Down-Sampling:

Due to the rapid growth of the networks, these networks become larger and larger. The size of these networks is massive, and it is not feasible to analyze. There are a large number of work which focus on reducing the size of network. Sampling is the common technique that has been used.

Network down-sampling method has been used for large network visualization. Many works are presented in [28, 62, 25]. The main goal is to reduce the size of the network such that it is good for visualization. Thus, the samples should capture the abstract view of the networks. Leskovec and Faloutsos study the characteristics of different state-of-the-art algorithms under the down-sampling scenario in [48]. They study and evaluate several sampling algorithms based on how well the sampled graph maintains properties of the original graph, and conclude that a Random Walk sampling method is best. Maiya and Berger-Wolf present a down-sampling algorithm that aims to preseve the community structure of the original network [51, 53]. The main idea behind their proposed algorithm is to select the node with the most neighbors outside the current sample in each iteration. Their results show that their sampled network captures the community structure of the original network when including as few as 15% of the nodes from the complete graph. However, other structural properties of the sample graph are not taken into account in this work.

2.3 Analysis of Sampling Algorithms

There has additionally been a great deal of work on comparing sampling methods. In [47], Lee, et al. study the statistical properties of sampled networks obtained by node, edge and snowball sampling under down-sampling. Leskovec and Falout-

sos also present a similar 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 [48]. They conclude that Random Walk sampling is the best at preserving network properties. Similar to the study in [48] but the main focus is on crawling setting, Ochodkova, et al. compare the properties of a generated network, real-world network and also properties of sample that obtained from different sampling techniques [59]. The experiments show how the distribution of the properties change by using different sampling methods and also how these properties of sampled network differ from the original network.

Likewise, Kurant, et al. analyze BFS crawlers, and demonstrate that such methods are biased towards high degree nodes [42]. Maiya and Berger-Wolf also study the biases of different sampling methods [54]. They argue that these biases of certain sampling method can be advantageous for some specific properties of interest. As a result, they found that bias towards high expansion is good for push the sampling process towards new undiscovered part of the networks.

Ahmed, et al. study several algorithms for both down-sampling and crawling [2]. Several methods are studied and evaluated the capabilities of preserving several properties of the networks. Ye, et al. present an empirical study that focuses on performance, sensitivity, and bias in [79], and Ahmed, et al. provide a framework for classifying sampling algorithms with respect to how well they preserve graph statistics [2].

Saroop and Karnik study the performance of different crawlers on the task of node coverage [70]. They focus on the crawler that is the best to crawl Twitter network. Other properties like *in-* and *out-* degree distributions are also taken into account when network is crawled. Similarly, Baeza-Yates, et al. focus on the crawlers for collecting web pages [10]. Several crawlers are studies based on how

well it find the important pages. The results show that BFS crawler has a bad performance, as compared to other crawlers like PageRank and OPIC crawlers. Hu and Lau present an excellent survey of several popular sampling methods for both down-sampling and crawling [38]. Several important network properties, theoretical studies and different types of evaluation criterion are discussed in detail.

Chapter 3
Network Structural Properties and the Performance of Crawling Approaches

In this chapter, our goal is to understand the interplay between network structure and crawler 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.

To study the effect of network structure, we are interested in the the properties that obstructs the crawler when it transitions from one region to other regions of the network. Three selected network properties are 1) community mixing 2) average node degree and 3) average size of community. We conduct a series the controlled experiments on both generated and real networks. A total of eight popular crawling methods are selected for the study.

Most previous work has assumed that all neighboring nodes are returned in

response to a query [8]. 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 work, we analyze the performance of crawlers on network with different query model, which 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.

Our main contributions in this chapter are as follows:

1. We consider five realistic query models- *complete*, *paginated*, *partial*, *in-out*, and *out* responses- which describe the neighbor data obtained in response to a query on a node.

2. While most of existing work, considers only 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 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.

The rest of this chapter is organized as follows. In Section 3.1, we discuss the problem, preliminaries, details of the network crawling methods, the network

structural properties of interest, and the considered query models. We describe our experiments and discuss their results in Sections 3.3, 3.4 and 3.5, and present conclusions in Section 3.7.

3.1 Network Crawling Overview

3.1.1 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. Here, we consider five different types of query responses, as described in Section 3.1.2. 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 3.1.3. 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.[1]

We consider the case where the crawling goal is to find nodes as many as possible. We refer this task as *maximizing node coverage*[2].

Sampling Goal (Node Coverage): Collect a sample graph $S = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$ so that the number of nodes in V' is maximized.

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 [53], and in [52], use this goal to find influential nodes. There are numerous other crawling goals that one could consider (such as obtaining an unbiased sample), but these goals require funda-

[1] We generalize the problem definition which covers broad scenarios. For example, web crawling, collecting data from OSNs, constructing a terrorist network for the investigation.

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

mentally different approaches, and so we do not consider them here.

3.1.2 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 [6]. 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:

Responses on Undirected Networks

Three types of query response model on undirected network are

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 tof 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.

Responses on Directed Networks

In a directed network, 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: In this query model, 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: In this query model, 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 online social networking sites also provide APIs with similar behavior. For example,

Flickr provides an API call for obtaining a list of users that user A follows, but not the users who follow A.

3.1.3 Closed nodes vs. Open nodes

As the crawler proceeds, the nodes seen so far can be grouped into various categories. First, 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 [67]. Using this technique, 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.

3.1.4 Online Crawling Methods

Based on the literature, we select and consider eight popular crawling methods. These methods have been recently used for web crawling, collecting data from technological, online and offline social networks.

Maximum Observed Degree (MOD) The crawler greedily selects the open node with the highest observed degree [8]. `MOD` substantially outperforms other methods at the node coverage task.

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 [69].

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, as demonstrated in [1].

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 in [56].

Snowball Sampling (SB) The crawler acts similarly to `BFS`, except only p fraction of each node's neighbors are put into the queue (we set p to 0.5). This method can find hub nodes in a few iterations, as presented in [3].

Depth-first Search (DFS) The crawler acts similarly to `BFS`, except that a node is selected in LIFO fashion. Analysis on network samples obtained using a DFS crawl is presented in [20].

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, are only queried if they are still an open node. The results of `Random Walk` crawling came out on top in [48].

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. Under the *paginated* and *partial* query models, the `BFS`, `SB`, `DFS` and `Rand` 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).

Note that the methods mentioned above are referred as "link-tracing" methods in which the crawler follows each link (edges) on the obtained sample in order to get more nodes and expand the current sample. These methods are suitable for the scenarios where we have no or very little knowledge about the population. Unlike surveys, public polls or crowdsourcing approaches where a data collector need decide who will be her target demongraphic (e.g. target group) before sending our those questionaires. These approaches have been used in sociology research [7, 46].

3.1.5 The Effects of Network Structure on Algorithm 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 [79], 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?

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:[3]

Community Separation: A community is a subgraph with dense intra-connections and sparse inter-connections. We find communities using the Louvain method [12], and then use the modularity Q of the detected partition to measure how well-separated the communities are [58]. Modularity Q is defined as

$$Q = \frac{1}{2m} \sum_{vm} \left[A_{vw} - \frac{d_v d_w}{2m} \right] \delta(c_v, c_w),$$

where A, m and d_i are the adjacency matrix, total edges, and degree of node i, respectively. $\delta(c_v, c_w)$ is a delta function which returns one when node v and w are in the same community. Otherwise, it returns zero. The higher the modularity, the better the separation between communities, and so a crawler is more likely to get

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

trapped in a region. Community separation is an important part of our analysis: e.g., Leskovec, et al. suggest that communities are well defined and distinct if they are small, but that large communities tend to mix with one another [49]. 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 next compute the average degree of all nodes in the network. 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. It is defined as

$$d_{avg} = \frac{\sum_{v \in V} d_v}{m},$$

where d_v is a degree of node v and m is a number of total edges.

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. It is defined as

$$CS_{avg} = \frac{\sum_{c_i \in C} |c_i|}{|C|},$$

where C is a set of communities, c_i is the set of nodes in community i and $|\cdot|$ refers to a cardinality of a set.

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: e.g., we can expect to see more dense, near-cliques in social networks than we would expect to see on citation networks, where we expect to see long chain-like structures [57]. One cannot reasonably expect a single crawling method to be the best on every network under different query responses. To address this issue, in Section 3.5, we provide a set of guidelines for users on selecting a suitable crawling method when network domain is known.

3.2 Experimental Setup

We now evaluate the effect of network properties on crawler performance through a series of experiments on synthetic and real experiments. 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.

To generate the synthetic networks, we adopt the LFR network model [45]. 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; d_{avg}, CS_{avg}, 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 (0.1 to 0.9), d_{avg} from 7-200, and CS_{avg} sizes from 100-2500. 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.

3.3 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*.

3.3.1 Responses on Undirected Graphs

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 3.1-3.3.

First, we consider the case where networks have a clear community structure

Table 3.1: 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 3.2.

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**

(high modularity, low community mixing μ: sharp community borders with few edges between communities), and average degree and community size are varied. Results are shown in Figure 3.1.

The outer axes indicates different values of the test properties. The outer x-axis represent the increasing in average community sizes (100-2500 nodes). The outer y-axis represent 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).

Next, in Figure 3.2, 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 3.3 depicts the case when networks have overlapping community structure (many edges crossing between communities), while average

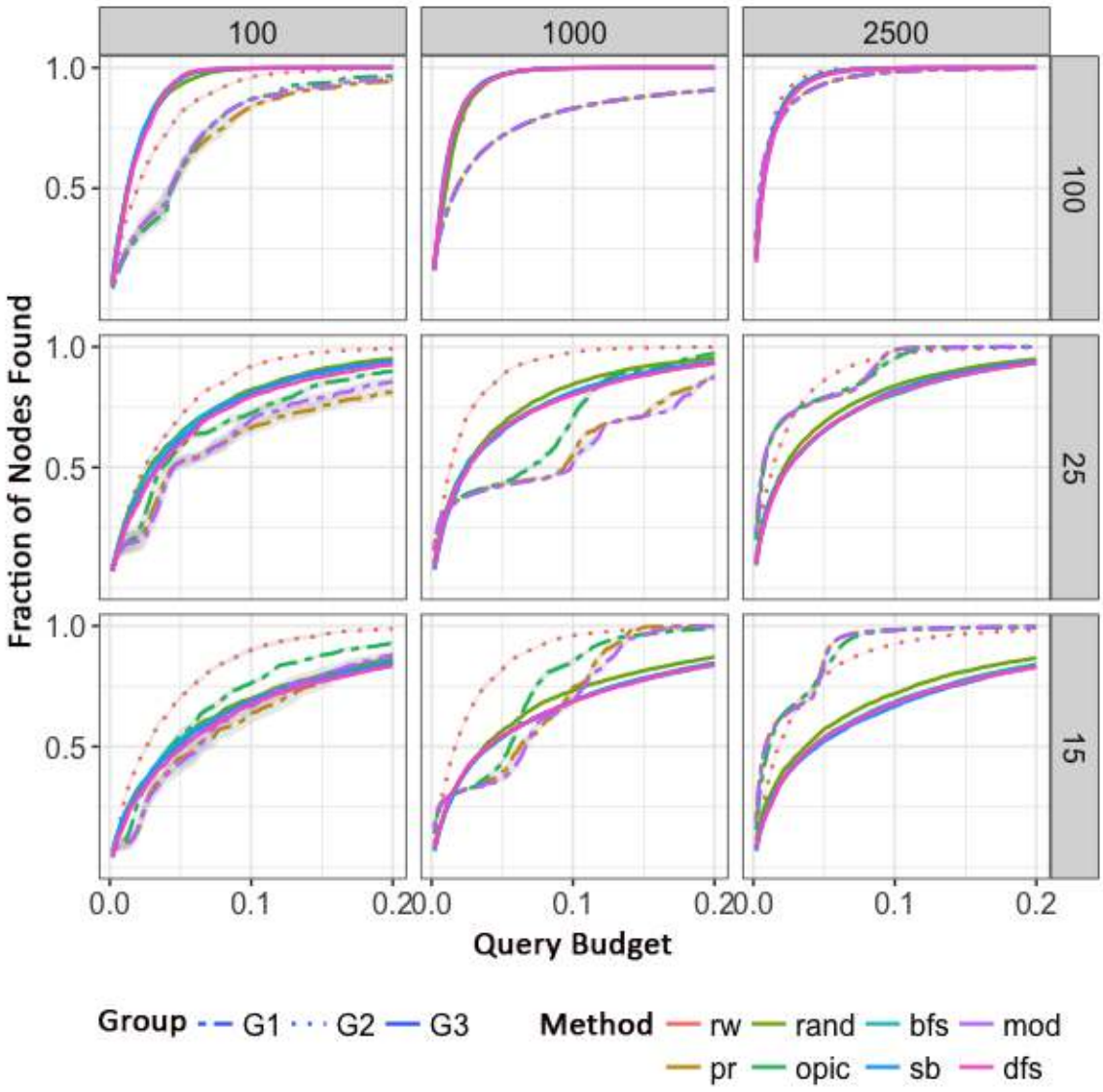


Figure 3.1: **Complete Response**: Results on networks with different values of d_{avg} and CS_{avg} when μ=0.1. G2 shows stable performance. G1 and G3 performance improves when CS_{avg} and d_{avg} increases, respectively.

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 from these results, and summarize in Table 3.1:

G1 - Node Importance-based methods: Methods in this group select a node with high observed centrality (e.g., degree or PageRank). The performance of

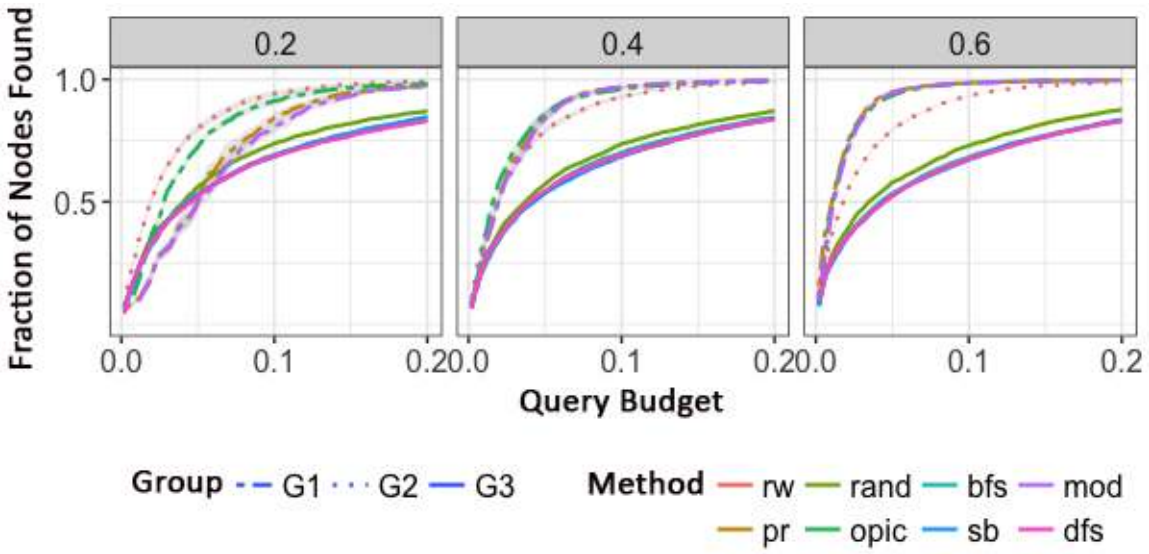


Figure 3.2: **Complete Response:** Results on networks with different values of μ (d_{avg}=15, CS_{avg}=300). G1 methods improve as μ increases.

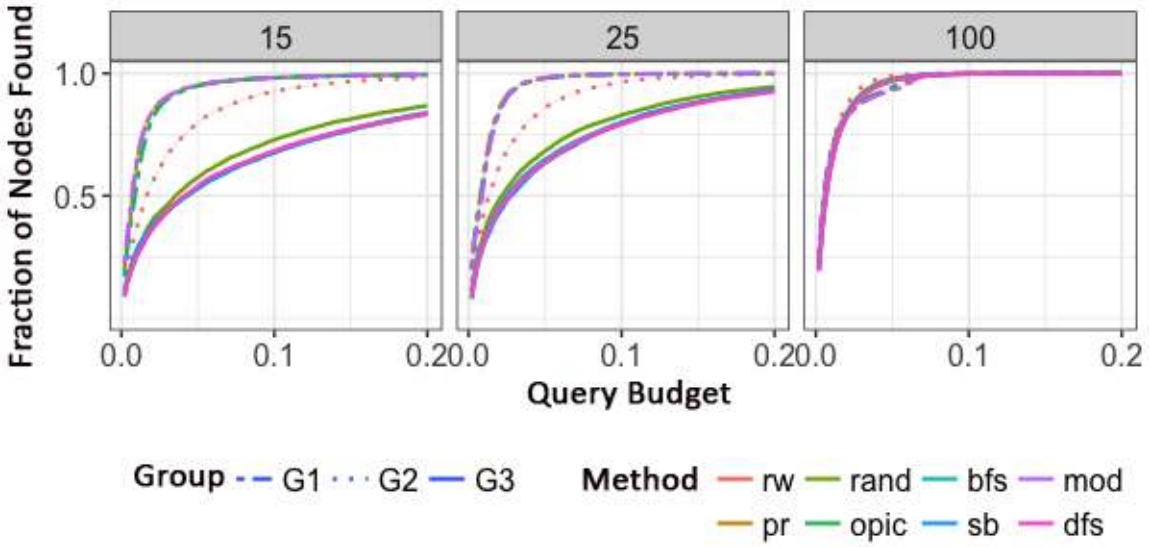


Figure 3.3: **Complete Response**: Results on networks with different values of d_{avg} (μ=0.6, CS_{avg}=300). G1 is the top performer. The performance of G3 methods improve as d_{avg} and CS_{avg} increases.

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 nodes 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 is 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 demonstrated in Figure 3.2 and 3.3.

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 3.1 and 3.3.

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 3.1, G3 performs better when average degree increases (moving up along y-axis) and become top performers on networks with large average degree.

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 3.1.

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 performer in two cases; 1) when communities are overlapping 2) when average community size is high or average degree is extremely low, even if communities are not overlapping. Examples are shown in Figures 3.2 and 3.3.

G2 - Random Walk: As before, the Random Walk 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. From the results, these crawlers have an performance improvement on networks with high average degree. We observe that `Snowball sampling` is the best among this group.

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 the 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 3.1.

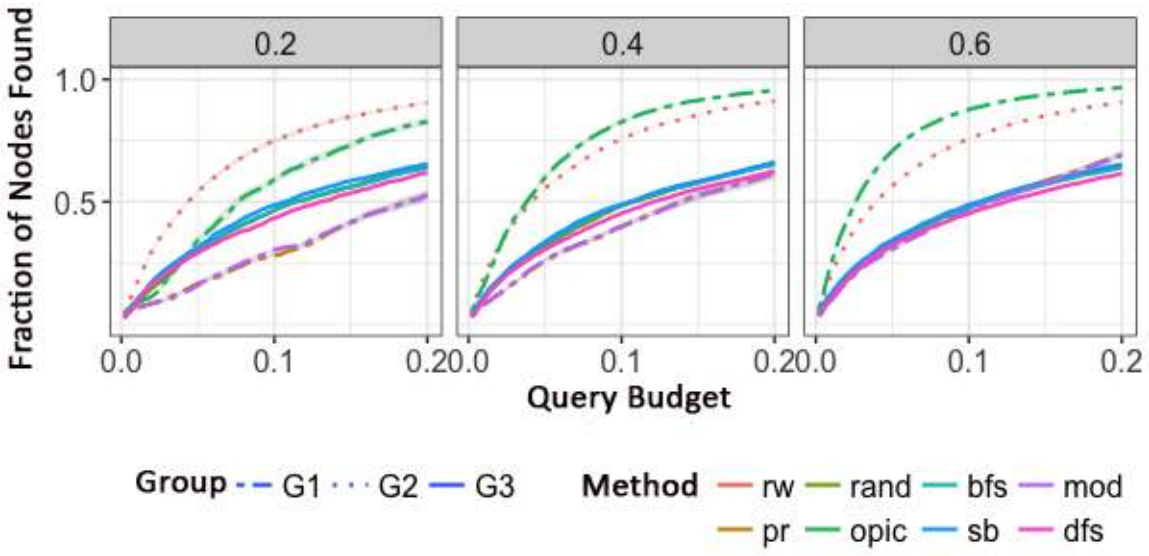


Figure 3.4: **Partial Response:** Results on networks with different values of μ (d_{avg}=15, CS_{avg}=300). G1 methods improve as μ increases, however, methods in this group generally perform similarly to methods in G3.

Under this query model, we observe some differences in performance changes 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.

G1 - Node Importance-based methods: The performance of methods in this group slightly improves when community mixing increases (i.e., overlap increases), as shown in Figure 3.4. Performance also slightly increases as average degree or average community size is increased. However, their overall performance is worse than methods in G3, as shown in Figure 3.5. 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

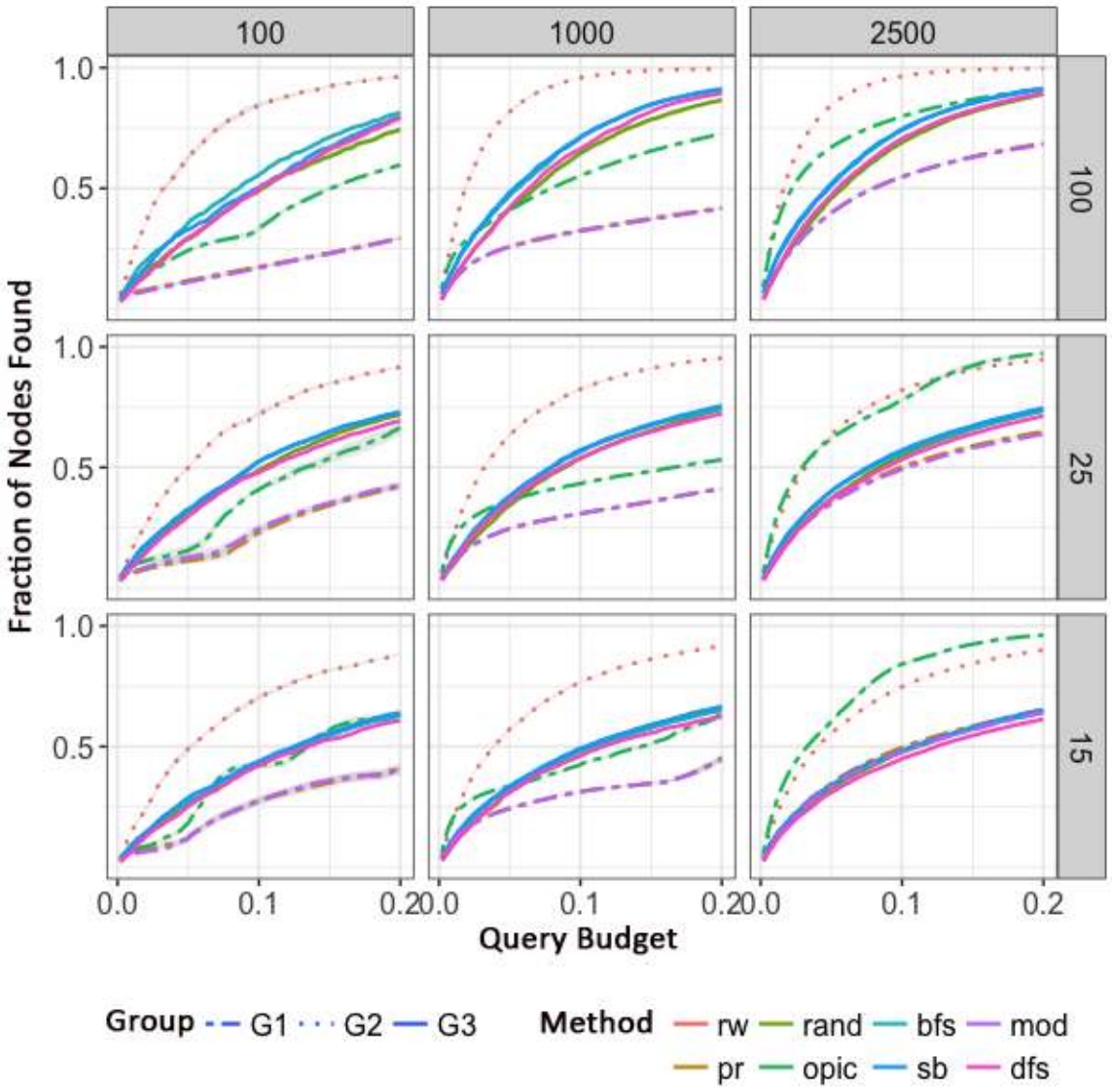


Figure 3.5: **Partial Response:** Results on networks with different values of d_{avg} and CS_{avg} (μ= 0.1). G2 performance is stable. G1 performance improves when CS_{avg} increases, while G3 methods improve when d_{avg} increases. Overall, methods in G1 seem to be the worst performers in general.

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

Table 3.2: 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**

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 G2 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 μ and CS_{avg}.

3.3.2 Responses on Directed Networks

Here, we present the results of our experiments on synthetic directed networks under the *in-out* and *out* response models.

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 3.1.

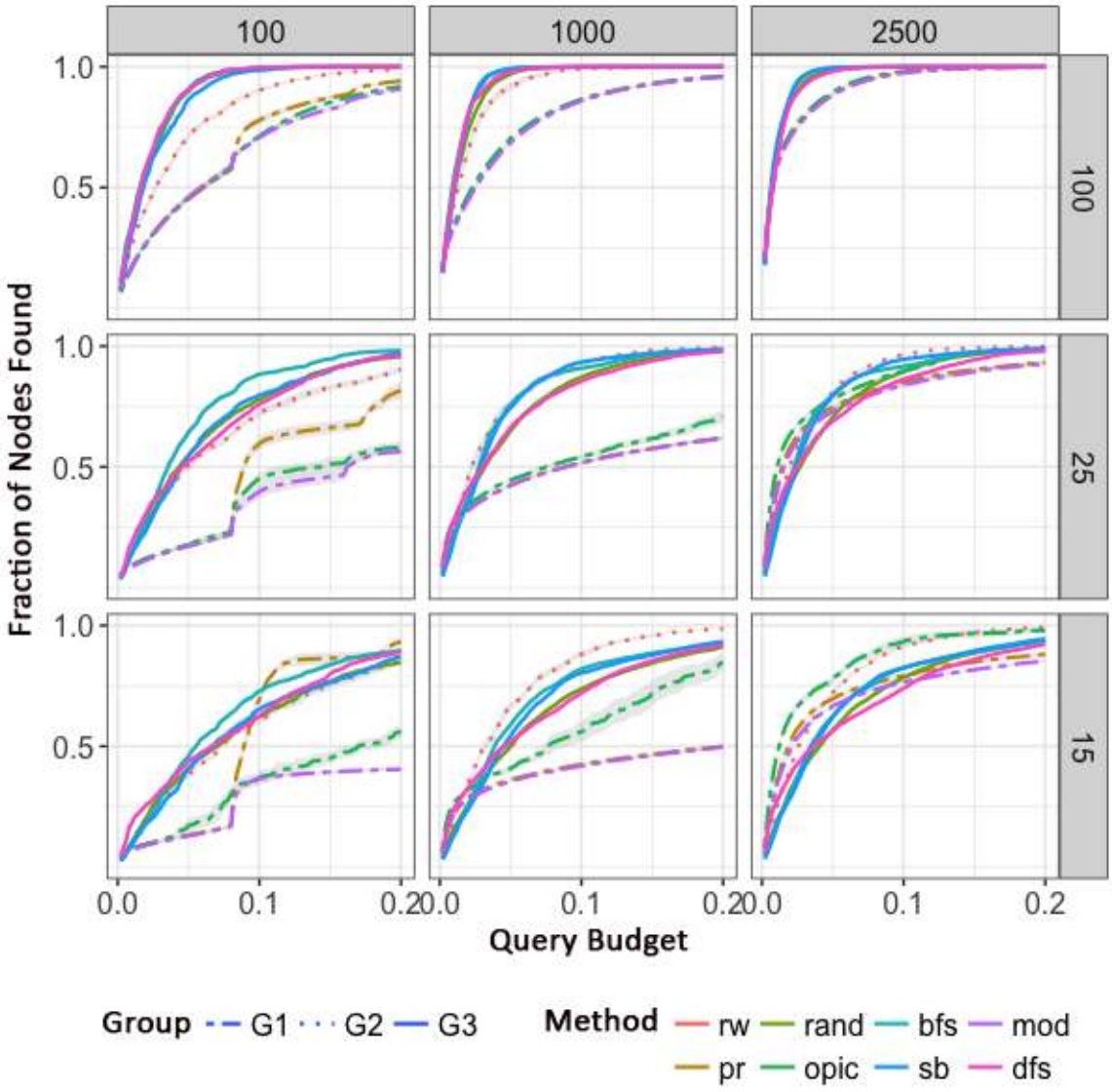


Figure 3.6: **In-Out Response:** Results on networks with different values of d_{avg} and CS_{avg} (μ=0.1). G2 performance is stable. G1 and G3 performance slightly improve as CS_{avg} and d_{avg} increases, respectively.

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 av-

erage degree or average community size is increased on networks with low community mixing, as illustrated in Figure 3.6. 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).

Out Response

Under the *out* query response, 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 3.7. The summary of how structural properties affect algorithms performance is summarized in Table 3.2.

G1 - Node Importance-based methods: On networks with low community mixing (few connections between communities), the performance of methods in G1 improves when average degree or average community size increases, but methods in G1 do not perform well compared to other methods. From Figure 3.8, we see that the PR crawler is the best (by a small amount) performer 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 im-

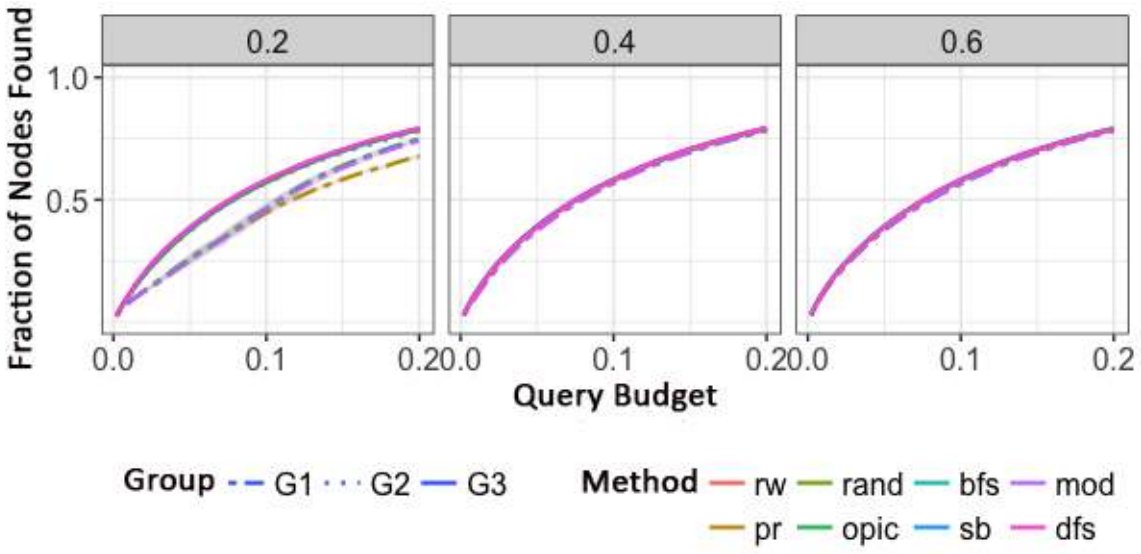


Figure 3.7: **Out Response:** Results on synthetic networks with different values of μ (d_{avg}=15, CS_{avg}=300). G1 methods improve as μ increases. Surprisingly, there is no difference in term of performance for these methods on the networks with high community mixing.

proves when average degree increases. In contrast to previous results, G3 methods perform very well and seem to be top performers.

3.4 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.

3.4.1 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

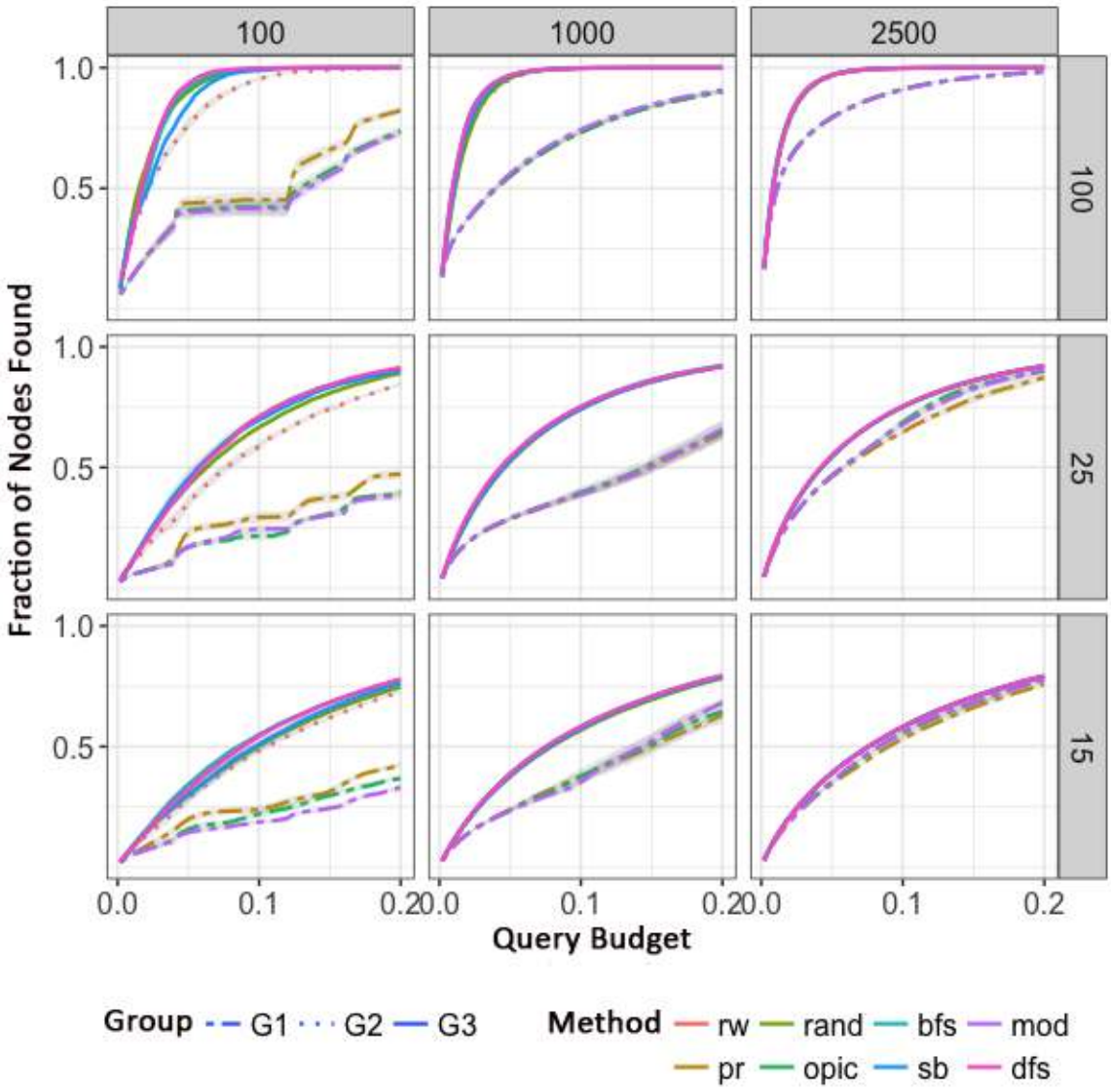


Figure 3.8: **Out Response:** Results on networks with different values of d_{avg} and CS_{avg} (μ=0.1). G2 and G3 performance slightly increase when d_{avg} increases. G1 performance slightly improves as CS_{avg} increases, but it seems to be the worst performer in most cases.

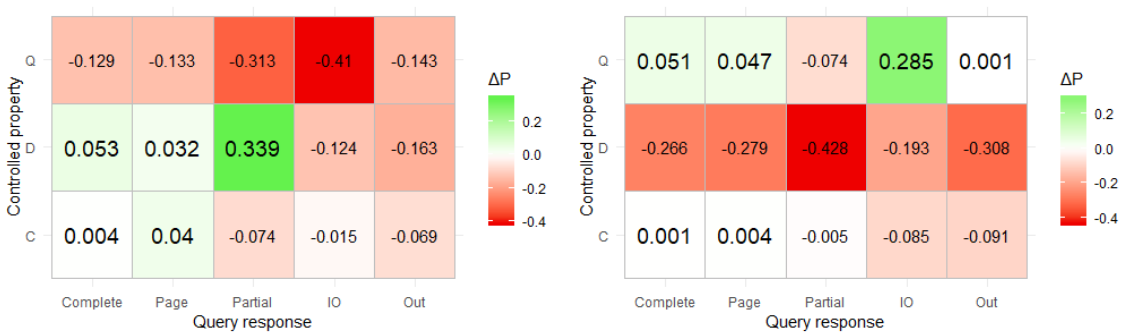
very different with respect to the third; e.g., *Wiki-Vote* and *Twitter* networks have different modularity values (0.42 vs 0.81) but they have similar average degree (Wiki-Vote has d_{avg}=28.51 and Twitter has d_{avg}=33.01) and average community size (Wiki-Vote has CS_{avg}=1177.67 and Twitter has CS_{avg}=1129.25). Within each pair, we refer to the 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 3.3

shows network statistics. The properties on which the networks in the same pair differ are shown in bold. Datasets can be found at ‡ networkrepository.com (NR) and † snap.stanford.edu (SNAP).

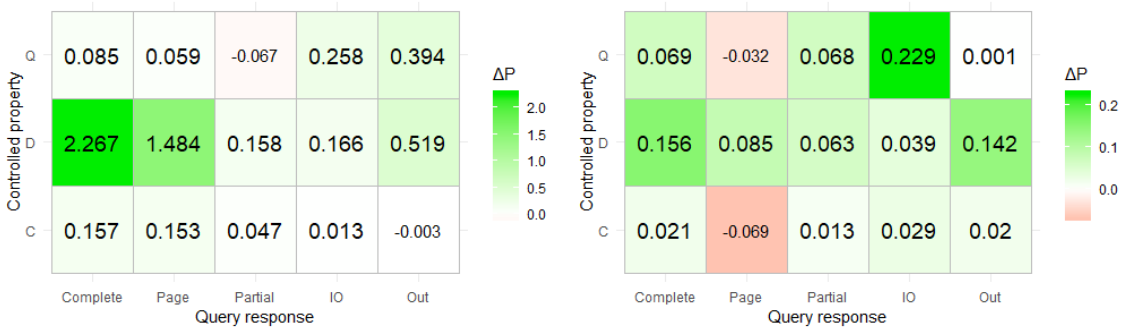
We find communities using the *Louvain method* [12] and measure the strength of the detected communities using the modularity value Q, which we use as a proxy

Table 3.3: 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 3.9: **[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 values indicate an improvement in performance and negative values indicate a performance degradation as controlled property increases. Zeros indicate performance is unchanged.

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, the `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 3.9. 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 $\Delta P_x = P_{hi} - P_{lo}$, 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 number 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 3.9a and 3.9b, 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 A.1 in the Appendix for full details.

3.4.2 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 3.9 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 3.9a (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 3.9a (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: `Random Walk` 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 A.1, 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 3.9b, 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 A.1.

3.5 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 in the same domain tend to have similar properties, and so it is possible to make reasonably accurate generalizations about the relevant structural properties. Here, we will use *network type* and *network domain* interchangeably.

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 domains: 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 3.4. 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 3.5. Full results

Table 3.4: 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	

are show in Table A.2 and A.3.

Newman suggests that networks with modularity $Q \geq 0.3$ have a strong community structure [58]. From Table 3.4, OSNs contain overlapping community structure, indicated by their having the lowest modularity of all considered domains. 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 3.4, all Facebook networks indicate a strong community structure ($Q \geq 0.5$). As expected, G1 methods per-

Table 3.5: 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

Directed Networks		
Type	**Best Method**	
	In-Out	**Out**
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	

form well on these OSNs, because they can freely move between regions. Other network domains have higher modularity (0.4 - 0.9), so, the performance can be determined by average degree and community size.

3.5.1 Undirected Networks

We first consider the network categories with high community separation (high modularity 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 per-

formance of the G2 method outperforms other methods. It is the best method to use on these domains.

3.5.2 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.

3.6 Major Takeaways

We have presented a wide variety of results across different domains 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 [61]. 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 underly-

ing network, but it has been shown that some network properties are self-similar, which means it has same statistical properties at many scales [72, 71]. Although, these samples may not be perfect representations of the underlying network, they have been used to capture the community structure [53, 69], degree distribution [63] or clustering coefficient [48, 63] of the original networks.

3.7 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.

Chapter 4
Expansion-Densification algorithm for the Data Collection

As mentioned in Chapter 2, the literature contains a large number of proposed crawling algorithms. Some of these methods perform well, but they are rarely consistent across network structures. In Chapter 3, we found that network structural properties have a strong effect on the algorithms performance. More specifically, strong community structure can obstruct a crawler from being able to move from one region to another region of the network. As a result, greedy approaches like `Maximum Observed Degree` (MOD) [8] works very well when the communities overlap with one another, but perform poorly when there are clear borders between regions. Conversely, `Random Walk` crawler is capable of moving around different region and works best when communities are disjoint.

To tackle this problem, we propose a novel crawling algorithm, **DE-Crawler**, in which the crawler can seamlessly transition to different regions of the network. It consists of two main stages: *Densification* and *Expansion*. The *Densification* stage

aims to discover nodes in the current region, while the *Expansion* stage aims to expand the sampled network by moving to another dense region. **DE-Crawler** is able to capture the best aspects of the existing algorithms, and achieves outstanding performance across domains.

The main contributions of this chapter are:

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

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

The rest of this chapter is organized as follows. We give an overview of existing work in this area. We formally state our problem definition is described in Section 4.1. The details of our proposed algorithm and key ideas are explained in Section 4.2. Section 4.3 and 4.4 contain our experimental setup and results, and we present conclusions in Section 5.8.

4.1 Problem Definition

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

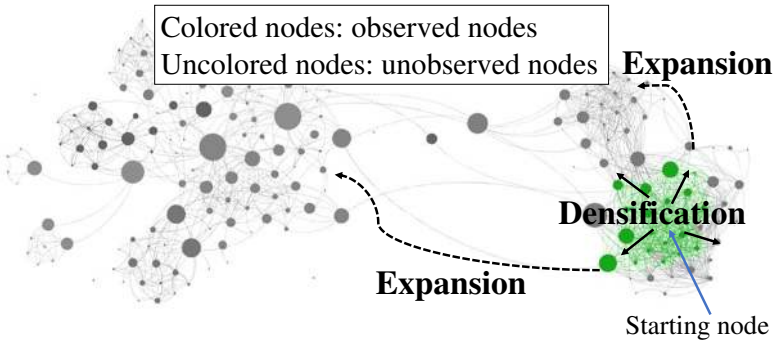


Figure 4.1: The concept of **DE-Crawler** algorithm. *Densification:* The crawler focuses on find as many nodes in a current region. *Expansion:* it tries to escape the current region and finds new unexplored dense regions.

output is a sampled graph $S = (V', E')$, where $V' \subseteq V$ and $E' \subseteq E$, containing all nodes and edges observed. The goal is to discover as many nodes as possible.

4.2 Proposed Method: DE-Crawler

4.2.1 Key Ideas

In Chapter 3, we observed that a major factor in crawler performance is the ability to fully explore regions of a graph, while still being able to move between regions of the graph. In particular, we observe that there are two important classes of crawlers, but their performance depends on network structure. Node Importance-based methods are those that select the node for query based on a centrality measure computed from the observed graph. These methods excel when the underlying network contains overlapping communities and/or community size is relatively large compared to the query budget, because in such networks, the crawler is able to transition between regions. Still, a Random Walk crawler is the best when communities are disjoint.

Node Importance-based methods quickly explore individual regions, but tend to get 'stuck' inside a community when community borders are sharp. As these methods continue querying nodes within the same community, then even though

each node is queried at most once, many of the queried nodes' neighbors will have already been observed; thus, the total number of observed nodes is small. In contrast, the Random Walk crawler is capable of moving freely between regions, but only partially explores each one. Based on these findings, we propose **DE-Crawler**, a novel crawling method that incorporates the best of both worlds.

The basic concept of **DE-Crawler** is illustrated in Figure 4.1. Given a starting node, the crawler aims to discover and fill out the nodes in a dense region (e.g., a community). We refer to this stage as the *Densification* stage. After the crawler discovers most of the nodes in that region, it attempts to expand to another dense region. We refer to this stage as the *Expansion* stage.

4.2.2 DE-Crawler algorithm

Pseudocode for **DE-Crawler** is shown in Algorithms 1, 2 and 3. Users must specify the budget for sample initialization $b^{'}$, total budget b, and the starting node n_s as the input parameters of the algorithm.

Initialization

When it begins, **DE-Crawler** conducts an *Initialize* step (line 2 in Algorithm 1). Here, the crawler collects a small sample so that it can obtain information about the underlying network structure. It uses this stage to initialize certain necessary

Algorithm 1 DE-Crawler

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

Algorithm 2 Densification

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

Algorithm 3 Expansion

1: **function** EXPANSION(S)
2: $V_c^{'}$ = getCandidates($V_o^{'}$)
3: v = random($V_c^{'}$)
4: **return** v

parameters. A small amount of budget b' is used, where $b' << b$. The initial sample can be collected by using any crawling method, and we adopt the Random Walk-based method proposed in [23]. Using this technique, we can estimate the average degree of the network, which will be used for weight adjustment (discussed in the later section). The output is a sampled graph $S^{'}$.

Densification

Here, the crawler selects and queries nodes with the goal of exploring the current region. Pseudocode is shown in Algorithm 2. The intuition here is to quickly find as many nodes as possible, since real networks are known to contain communities that are internally densely connected. Therefore, the sooner the crawler finds the highly-central hub nodes (e.g. high degree nodes), the faster the crawler can observe the remaining nodes in the region. Building on our earlier work in [6], this stage is based on the success of the *Node Importance-based* methods in exploring individual regions [6]. The basic idea is to pick a node with the highest observed centrality (i.e. degree/PageRank centrality) since it is likely that these high centrality nodes are hub nodes. In this way, each query observes many new nodes.

Node Selection: In order to select a node that will give high true degree, **DE-Crawler** identifies candidate query nodes κ as those in the top 20% as ranked by observed degree.[1] For each candidate $v \in \kappa$, a score $\Phi(v)$ is calculated. The score is defined as

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

where d_v^o and $\hat{c_v}$ are, respectively, the normalized observed degree and observed clustering coefficient of node v. This formula is motivated by the observation that hub nodes tend to have high degree and low clustering coefficient [13]. The crawler then selects and queries the node v with the highest score.

Note that the node selection criteria (Φ) can be changed and improved based on the knowledge of the a data collector. In this work, our objective is to prove the idea of the proposed algorithm. We have tried other node selection methods; e.g., selecting a node with highest observed degree and a score proposed in [14]. By several trails, **DE-Crawler** with the proposed node selection criteria performs the best in the given settings.

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

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

At each step t, the s_t^d and s_t^e are calculated (line 4 and 5 in Algorithm 2). These scores have two terms which incorporate the current stage and previous stage of

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

the sample. The *Densification score* s_t^d indicates how many new unseen nodes are found after node v is queried at step t. It is defined as

$$s_t^d = \alpha_1 \cdot \frac{d_v^{new}}{d_v^{ex}} + \beta_2 \cdot s_{t-1}^d,$$

where d_v^{new} is the number of new edges that connect node v to new discovered nodes after the query and d_v^{ex} is the excess degree, which is defined as the difference between the true degree and the observed degree of the node before the request. α and β are the weighting parameters that control the influence of the current score and the previous score.

On the other hand, the *Expansion score* s_t^e measures the amount of nodes that have been seen so far, which is given by the ratio of d^{ext}, the number of new edges that link to already-observed nodes, to excess degree d^{ex}. It is defined as

$$s_t^e = \alpha_2 \cdot \frac{d_v^{seen}}{d_v^{ex}} + \beta_2 \cdot s_{t-1}^e,$$

where d_v^{seen} is the number of new edges that connect node v and nodes that already be in the sample before a request. β_1 and β_2 are parameters that weight the densification and expansion scores of the sample at step $t-1$. We have observed that setting $\beta_1 = \beta_2 = 0.5$ gives us the best results.

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

Generally speaking, if the network is easy to expand, we want the crawler to

spend more time on densification than on expansion. To set α_1, the crawler looks at the initial sample to estimate the expansion factor. We adopt the idea from work on dynamic processes, e.g. epidemic spreading or information diffusion, to estimate the expansion factor. We set α_1 to be the ratio between the maximum degree and average degree of nodes in the initial sample ($\alpha_1 = d_{max}/\hat{d}$). This ratio is closely related to "epidemic threshold" τ, which governs how fast an epidemic can spread, and thus is also related to how quickly the crawler can expand the sample [16].

Expansion

The crawler attempts to move out of the current region and attempts to search for a new unexplored dense region. In the spirit of an explore-exploit algorithm, we use the approach of choosing a node uniformly at random from the list of observed-but-not-queried nodes. In the earlier *Densification* stage, **DE-Crawler** selected a node from the top 20% of candidate nodes as ranked by observed degree; here, **DE-Crawler** selects a random node in the bottom 80% of observed degrees, since these nodes are poorly connected to the sampled network. The pseudocode is displayed in Algorithm 3.

4.3 Experimental Setup

We compare **DE-Crawler** to seven baseline crawling methods `RW`, `BFS`, `MOD`, `OPIC`, `Snowball`, `DFS`, and `Random`. Due to space constraints, we present results only for the following four baselines: `RW`, `BFS`, `MOD` and `OPIC`. We chose these methods because results in Chapter 3 grouped them into three classes. These four methods represent the best of each class. The details of these baseline algorithms are described as follows:

1. *Maximum Observed Degree (MOD)*: A crawler selects the *observed-but-not-queried node* that has the highest observed degree. This method works well when communities are overlapping [8].

2. *Online Page Importance Computation (OPIC):* It is an online algorithm which aims to estimate each node's centrality score with only local updates. OPIC belongs to the same class as MOD, as shown in Chapter 3, and outperforms MOD in a some cases. In each step, the algorithm updates the scores of the most recently queried node and its neighbors. Each node is given an initial score, and the score is distributed evenly to it neighbors after each query. The node with the highest score is selected for the next query [1].

3. *Random Walk (RW):* A crawler transitions to a random neighbor of the latest queried node. Nodes can be visited multiple times but crawler only queries a node if it was not queried before. According to [6], RW is the most *stable* algorithm, performing consistently across network types.

4. *Breadth-first Search (BFS):* Due to its simplicity, BFS is one of the most popular crawling algorithms [56]. Nodes are selected and queried in FIFO fashion.

As we already observed in the previous chapter, networks of the same type tend to have similar structural properties, and there is no single method that performs the best across different types of networks. As we show in Chapter 3, `MOD` and `OPIC` perform the best when 1) the underlying network contains overlapping communities and 2) the size of each community is large compared to the query budget, even if the communities are disjoint and have sharp borders. On the other hand, the `RW` crawler is the best on networks that contain disjoint communities structure. The results show that its performance is not affected by network properties. The `BFS` crawler is generally a weak performer, but we include it here due to its popularity as a crawling algorithm.

Table 4.1: The statistics of real-world networks used.

Type	Network	# Nodes	# Edges	$\hat{d}$	$\hat{CS}$	Q
Collaboration	Astro	17903	196973	22.00	436.66	0.63
	CondMat	21363	91287	8.55	374.79	0.72
	HepPh	11204	117619	21.00	238.38	0.65
	Citeseer	227320	814135	71.6	988.34	0.89
Facebook100	Bingham	10001	362893	72.57	1250.13	0.45
	JohnsHopkins	5157	186573	72.36	515.70	0.45
	WashU	7730	367527	95.09	966.25	0.47
	Yale	8561	405441	94.72	856.10	0.43
Online Social Network	Anybeat	21250	66892	6.30	259.15	0.48
	Slashdot	70068	358648	10.24	173.86	0.36
	Hamsterster	2000	16097	16.10	66.67	0.54
Web	Google	1299	2774	4.27	34.18	0.93
	IndoChina	11358	47607	8.38	153.49	0.94
	Webbase-2001	16062	25594	3.19	232.78	0.93
Techology	RL-caida	190914	607611	6.36	856.11	0.86
	PGP	10680	24316	4.55	106.80	0.88
	Router-rf	2113	6633	6.28	88.04	0.69
	WhoIs	7476	56944	15.23	276.89	0.56

In our experiments, we use a total of eighteen networks from different categories. The statistics of each network are provided in Table 4.1[2]. We perform 10 runs on each network and report the average fraction of nodes observed by each crawler. We set the query budget b to be 10% of the total nodes in the network and for **DE-Crawler**, we set the initialization budget b' to be 15% of the total budget.

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

Note that this problem is a NP-hard problem; thus, the true optimal solution is

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

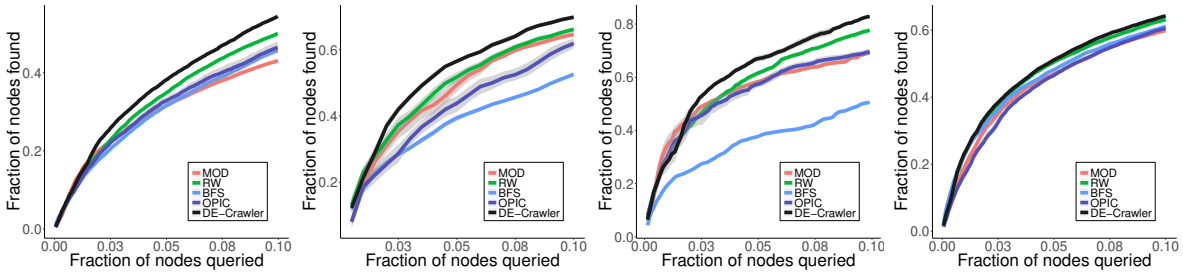
unknown. The MED crawler can produce a near-optimal solution, meaning that the actual regret may be equal or lesser than the regret calculated against MED. So, the calculated regret which mentioned above indicates the upper bound of the actual values.

An alternative way of implementing oracle can be done by using Monte Carlo Tree Search (MCTS), which we assume that this oracle has the full knowledge of the network. Each node in the search tree represents a possible obtained sample in each time-step. MCTS is an algorithm which will figure out the best strategy out of all possible moves (e.g. queries) in order to find the final solution which may be closer to the optimal solution. However, this method is time-consuming since MCTS is an iterative method and there are several possible paths that needs to be considered (e.g., number of nodes for the next query in each time-step is really extremely large). There can be exponentially many such paths as the sample size increases.

4.4 Experimental Results

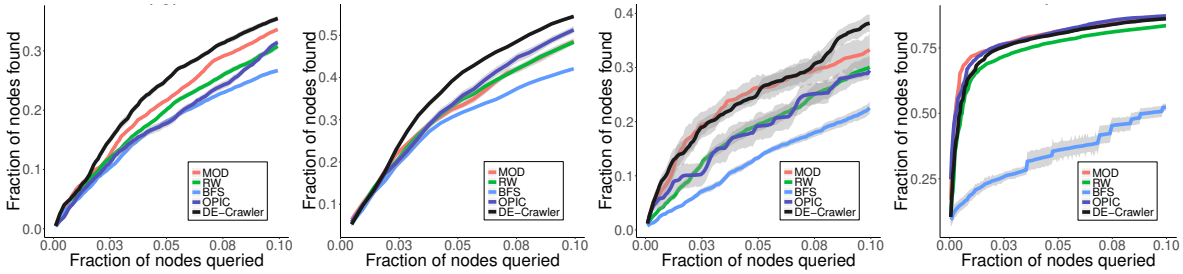
In this section, we present the performance of our proposed algorithm **DE-Crawler** against other baseline methods, as described in Section 4.3. We evaluate these methods with respect to the *node coverage* task (discover as many nodes as possible). Note that, we compare our algorithm to many algorithms; `RW`, `BFS`, `MOD`, `OPIC`, `Snowball`, `DFS` and `Random`, but these four baselines (`RW`, `BFS`, `MOD`, `OPIC`) were best. So, we present the results of these four baselines.

Results are presented in Figure 4.2, 4.3 and Table 4.2. In Figures 4.2 and 4.3, the x-axis represents the query budgets, and the y-axis represents the fraction of nodes observed in the sample. Table 4.2 shows the overall $regret$ of each method, as compared to the oracle. Our results show that **DE-Crawler** is the best of both



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

Figure 4.2: **DE-Crawler** consistently outperforms or matches the best baseline method on networks that RW outperforms MOD.



(a) AstroPh (b) AstroPh (c) AstroPh (d) AstroPh

Figure 4.3: **DE-Crawler** consistently outperforms or matches the best baseline method on networks that MOD outperforms RW.

worlds: it performs consistently well across all network types, regardless of community structure. In all but one of the considered networks, **DE-Crawler** is the best performer.

As discussed earlier, RW and MOD are excellent methods, but the choice of which is best depends on the network structure. E.g., Figure 4.2 demonstrates the case where RW is better than MOD: these networks contain dense, distinct communities, and MOD has trouble transitioning between regions. Figure 4.3 illustrates the case where MOD outperforms RW: these networks have overlapping communities with fuzzy borders, allowing MOD to move freely between regions.

But in both cases, as we can clearly see, the performance of **DE-Crawler** substantially outperforms all the baselines. By switching between expansion (moving to new regions) and densification (exploring the current region), **DE-Crawler** is

Table 4.2: The average $regret$ of **DE-Crawler** and baseline algorithms (lower value means better performance).

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
Online Social Networks	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
Technology	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

able to gain an improvement of up to 28% as compared to the best baseline methods. The results in Table 4.2 also show that **DE-Crawler** performs achieves a low regret, indicating that it performs close to the optimal greedy method. **DE-Crawler** has the lowest average regret of approximately 0.22, which is dramatically better than `RW`, the next best method.

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

4.5 Conclusion

We considered the problem of **online network crawling** with the goal of maximizing node coverage. In Chapter 3, we demonstrated that the performance of existing crawling methods is heavily affected by network properties [6]. Intuitively, a strong community structure can obstruct a crawler from moving between regions. Based on that observation, we introduced **DE-Crawler**, which consists of two main stages: *Densification*, which explores the current region, and *Expansion*, in which the crawler transitions to a new region. Our results over 18 datasets show that **DE-Crawler** outperforms all other baselines, with an improvement of up to 28% over the next best baseline. Moreover, **DE-Crawler** is consistently the best over all considered network types, and the results also show that **DE-Crawler** performance is close to the performance of the optimal greedy crawling algorithm.

CHAPTER 5
SAMPLING ROBUSTNESS

As discussed in Chapter 2, studying the structure of complex networks has become an interesting and important task. Data analysts can gain many insights by analyzing real-world networks. For example, one can study how fast information flows through a network, how people form a community, or identify products to recommend to consumers. However, before performing any graph analysis task, one must collect appropriate network data.

The method used to collect network data is largely dependent on the network domain or the preference of the data analysts. For example, one can collect *offline* social data through pen-and-paper questionnaires or by interviewing subjects. On the other hand, for *online* social networks, one can write a computer program which queries through a provided API. In many scenarios, a data analyst or a data collector may have no initial knowledge about a network except the identity of a single seed node (e.g. the first person that she will interview). A network sample can be expanded by querying already-observed nodes to learn their neighbors.

For simplicity, our work so far has assumed that the data collector gets a *complete* response when he performs a query (i.e., all neighboring nodes are returned). However, in real world application scenarios, we may encounter errors during the

data collection process. When a data collector performs a query, a list of neighboring nodes is returned in response, but this list may be incomplete. Such errors can occur for many reasons: for example, a participant who answers a questionnaire may make a mistake in their answer, a web crawler may have a bug and fail to extract links from web pages, or an adversary may tamper with the API and alter the information exchanged between two parties. These errors may then lead to inaccuracy in a subsequent network analysis. Therefore, it is important for a data analyst to know if a collected sample is trustworthy. If not, a data analyst will get some sense of the inaccuracy and fix the sample before performing the analysis task.

In this chapter, we introduce a new network robustness measure, which we call *sampling robustness*. To the best of our knowledge, while there are many ways to evaluate network robustness in general, there is no existing work that measures a network's robustness with respect to sampling. For a crawler of choice C, the sampling robustness of network G, denoted by $R_p(G, C)$, is defined as the expected similarity between two samples: one produced by crawler C on an *error-free* version of G, and one produced by C on a version of G in which each edge is missing with probability p. Intuitively, if a network is robust to the error, the performance of a crawler C will be mostly unaffected by missing edges when it crawls network G.

In this work, we model the error as *random edge deletion*, though our definition could easily be adapted to other types of error. Here, our goal is to investigate how the sampling robustness of a network with respect to random edge deletion, and analyze network sampling robustness using the network's structural properties, allowing a data analyst to predict whether a network will have high or low sampling robustness by measuring only a small set of parameters. We observe that sampling robustness is strongly correlated with the network's leading eigenvalue,

average node degree, and global clustering coefficient, and with these simple measurements, one can estimate a network's robustness.

Our contributions can be summarized as follows:

1. We introduce and define a new robustness measure, *sampling robustness*, which measures the robustness of a network with respect to sampling by a crawler when edges from the original network are dropped at random. Each edge has probability p that it will be removed from a list of returned edges after each query.

2. To measure sampling robustness, we demonstrate 4 different types of performance measure, which depend on the sampling goal and output type of the measure.

3. We show that sampling robustness is highly dependent on the network structure. Networks of different types have different level of sampling robustness.

4. We observe that sampling robustness is highly correlated with the leading eigenvalue, average degree and average clustering coefficient calculated from sampled networks.

5. We show that sampling robustness decreases as the error probability p increases.

6. We present regression models for estimating sampling robustness given a sampled network.

The rest of this chapter is organized as follows. In Section 5.1, we give an overview of existing work for network robustness. We formally state our problem definition and definition of sampling robustness in Section 5.2. Experiments and key observations are explained in Section 5.3 - 5.6. Lastly, we present a regres-

sion model for estimating sampling robustness of any network given an obtained sample in Section 5.7. The conclusion is presented in Section 5.8.

5.1 Related Work on Network Robustness

In this section, we explore previous literature that relates to network robustness. To the best of our knowledge, there is no existing work on network robustness relating to network crawling or data collection. However, there are numerous studies on other forms of network robustness.

Work on network robustness has a long history, and has been heavily studied by researchers from different backgrounds, including computer science, biology, physics and mathematics. In general, network robustness is defined as the ability of a network to keep functioning when there is a random failure or targeted attack [5]. For example, a telecommunications network is considered to have high robustness if the network continues its functions and services when some devices fail. Intuitively, robustness is all about back-up possibilities or alternatives paths [24]. Interest in network robustness was sparked by the study of Albert *et al.* in [5]. They study the effect of random failures and targeted attacks. They measure network robustness by the diameter of the network and size of the largest connected component, and show that scale-free networks have a high degree of tolerance against random failure, as opposed to random networks. However, scale-free networks are very sensitive to targeted attacks. The diameter of the network drastically increases and the network breaks into several components when the hub nodes are attacked. Cohen *et al.* are interested in finding the critical point (exact fraction of nodes to be removed which causes the networks to break into isolated fragments) under a targeted attack in [21]. Other measures for capturing network robustness are proposed, including shortest-path [37], path diversity [73],

eigengap [26], spectral radius [17].

5.2 Sampling Robustness

In this section, we begin by giving a brief description of the data collection process and random error. Next, we provide the details of different crawling techniques. Lastly, we define sampling robustness of a network G.

5.2.1 Network Data Collection

Let $G = (V, E)$ be an undirected, unobserved network, where V is the set of nodes and E is the set of edges. At the start of the data collection process, only the identity of a single seed node is available. A data collector collects a network sample using a crawling algorithm C. Given a seed node and a query budget, the crawler expands the sample by iteratively querying observed nodes. For each query response, all the neighboring nodes and incident edges are returned and added to the sampled network. The crawler selects another next query node from the list of observed nodes in the obtained sample and repeats until the query budget is exhausted. We assume that the data collector queries each observed node at most once. A crawler C thus generates a sample network $S = (V', E')$ where $V' \subseteq V$ and $E' \subseteq E$ are a list of nodes and edges observed, respectively.

5.2.2 Random Error

Error can originate from many different sources, such as mistakes or missing answers from survey respondents, a misreading of instruments by the data collector, or a bug in the data collection program. In this work, we consider the case of error modeled by *random edge deletion*. With this type of error, each query misses some fraction of edges. To model this type of error, each returned edge has a probability

p that it will be removed from a list of returned edges after each query. If there is an edge between node A and B, this edge may be missed when a crawler queries on node A, but possibly discovered when B is queried.

5.2.3 Network Crawling Technique

In this section, we describe each crawling method in detail. We consider three popular crawling algorithms: BFS, Random walk and MOD. These crawlers were selected as they represent three important categories of crawling algorithms [6] as we already seen in Chapter 3.

To collect a network sample, each crawler is given the same seed node and the same total query budget (in our experiments, we set a budget to be 10% of the total nodes of a network).

Breadth-first search (BFS): The BFS crawler selects the node that has been in the list of unqueried nodes the longest (First-in, First-out). After each query, all of the neighboring nodes that have not been queried are added to the queue. The BFS crawler uniformly expands its frontier and is good at capturing a complete view of the network.

Random walk (RW): In each iteration, the crawler transitions to a random neighbor of the node that was just queried. The crawler performs a query if it lands on an unqueried node. The random walk crawler is capable of finding many nodes from different regions (e.g. communities). Here, the random walk crawler cannot teleport.

Maximum observed degree (MOD): This crawler selects the unqueried node with the highest observed degree. The MOD crawler can quickly finds hub nodes in a few iterations [8].

For completeness, we also consider our proposed algorithm **DE-Crawler**, which we discussed in Chapter 4. The results of **DE-Crawler** are largely similar to the re-

sults of Random walk crawler. For full results of **DE-Crawler**, please see Figure A.1 and A.2 in Appendix.

5.2.4 Measuring Sampling Robustness

We define a novel network measure, *sampling robustness*, which measures the extent to which a sample generated by a crawling algorithm in the presence of errors (either in the original network- e.g., a communications network in which edges flicker in and out of existence- or in the crawling process itself- e.g., errors in the crawling process) is representative of a sample generated by the same algorithm without errors.

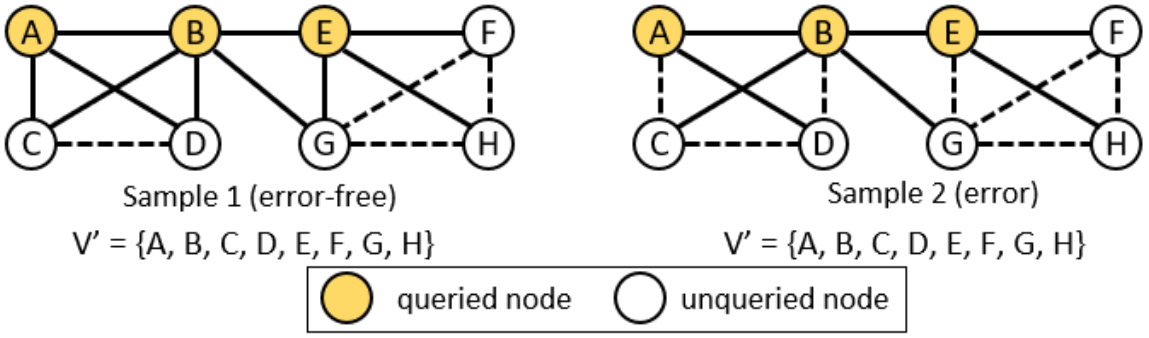


Figure 5.1: A toy example of network with high sampling robustness, showing two samples generated from the same crawler C on network G. The crawler queries node A, B and C, respectively. (Left) An *error-free* sample, S. (Right) A sample containing some errors, $S^{'}$. Edges (A, C), (B, D) and (E, G) are missing. These two samples contain the same set of observed nodes.

If a network G has high sampling robustness, the performance of a crawler C on network G will be consistent even if there are errors in the original network or in the data collection process. Here, we assume such errors take the form of edges missing uniformly at random, but the definitions and analysis that we present can easily be generalized to other types of errors.

A toy example is illustrated in Figure 5.1, which illustrates a case of a network with high sampling robustness. Nodes A, B and C are queried, respectively. The left figure depicts a sample generated from a crawler C when there is no error

(*error-free* sample). The right figure depicts a sample generated from a crawler C where some of the edges are missing, (namely, (A, C), (B, D) and (E, G)). Assuming our sampling goal is to find as many nodes, we can clearly see that both samples contains the same set of observed nodes, $V_1^{'} = V_2^{'} = \{A, B, C, D, E, F, G, H\}$. This indicates that network G has high sampling robustness since the performance of a crawler C is consistent even though some edges are missing. A formal definition of sampling robustness is shown in Definition 5.2.1.

Definition 5.2.1. Sampling Robustness

$$R_p(G, C) = \frac{sim(M(S), M(S^{'}))}{\bar{R}_0}$$

We denote the sampling robustness of G when p fraction of edges are missing uniformly at random as $R_p(G, C)$, as shown in Definition 3.1. Here, we let S represent the *error-free* sample (i.e., the sample produced by running a crawler C on network G), and let $S^{'}$ represents the sample obtained by the crawler C with errors (i.e., the sample produced by running a crawler C on network G with missing edges). The numerator is defined by computing the similarity between two samples, S and S', produced by a crawler C: the first on the original network G without errors, and the second on a version of G in which p fraction of edges have been removed at random.

In the denominator, we account for potential randomness in sampling (including the choice of seed node from which the crawler begins). We normalize this value by $\bar{R}_0$, which represents the average similarity between two samples in the case where there are no missing edges (p = 0). To calculate this, we generate multiple *error-free* samples and compute the average similarity of these samples against each others.

Table 5.1: Examples of different performance measures $M(\cdot)$ and appropriate similarity measures $sim(\cdot, \cdot)$.

Performance Measure	Output Type	Similarity measure
number of nodes or edges found	number	$1 - d_{canberra/L_1/L_2}$
distinct nodes in the sample	a set	Jaccard similarity
communities membership	a set of set	NMI, Partition distance [34]
degree distribution of the sample	distribution	1-KS statistic

Performance Measure and Similarity

In Definition 5.2.1, $M(S)$ is an application-specific function which characterizes the performance of the crawler C when it generates a sample (e.g., if one is interested in the sampling robustness of a network for the community detection application, $M(S)$ could represent the set of communities detected on S). Note that $M(\cdot)$ can be any function, as depends on the sampling goal. This means that different types of outputs can be returned by $M(\cdot)$. Some examples are as follows:[1]

- Numbers - e.g. the number of nodes or edges found
- A set - e.g. the distinct nodes in the sample.
- A set of sets - e.g. communities in the sample.
- A distribution - e.g. degree distribution of the sample.

Thus, the appropriate *similarity* measure depends on the output of $M(\cdot)$. The examples of different performance and similarity measures are shown in Table 5.1. For example, if a data collector wants to collect a network sample for a census-type application, she can use *the number of nodes found* as a performance measure, since the goal of sampling is to collect distinct people as many as possible, which the robustness can be calculated by the distance function. On the other hand, if the data

[1]Our code and implementation can be found at https://github.com/kareekij/sampling-robustness.

collector wants to collect the sample such that it contains population from several groups, the appropriate performance measure would be a *community membership of nodes*. Thus, she would use the partition distance as a similarity measure for calculating the robustness of a network.

5.3 Sampling Robustness and Network Type

Our definition of sampling robustness requires access to the original network G. Note that this is something of a contradiction: if one has the entire original network G, one need not concern oneself with sampling, or even with robustness! So in practice, if one has collected a sample, with errors, from a graph G, how can one determine whether that sample is likely to be a good representation of the sample one would have obtained had the sampling process not contained errors?

We hypothesize that *the sampling robustness of a network depends on that network's properties*. We verify our hypothesis by conducting the following experiment. Naturally, networks of the same type tends to have similar properties, as shown in Chapter 3. For example, the average degree of collaboration network (e.g. number of names appears on the manuscript) is around 5 while the average degree of the social network (e.g. number of friends) is around 20.

Therefore, using only the network type, we can roughly classify networks by their properties. In our experiments, we computed the sampling robustness of 15 networks on 5 different categories. To verify our hypothesis and investigate the level of robustness, we define the performance measure $M(\cdot)$ as the size of the sample (node coverage), which is defined as

$$M(S) = |\{v \in V_s^{'}, V_s^{'} \subseteq V\}|$$

This function measures the performance of a crawler in terms of the number

Table 5.2: Statistics of a network. $|V|$ is the number of nodes, $|E|$ is the number of edges, $\bar{d}$ is the average degree, $\bar{cc}$ is the average clustering coefficient and λ_1 is the leading eigenvalue of adjacency matrix A of the network. These networks can be downloaded from www.networkrepository.com.

Type	**Network**	$\|V\|$	$\|E\|$	$\bar{d}$	$\bar{cc}$	λ_1
CA	Erdos992	4991	7428	2.977	0.08352	15.13
	HepTh	8638	24806	5.743	0.4816	31.03
	GrQc	4158	13422	6.456	0.5569	45.62
BIO	CE-GN	2215	53680	48.47	0.1843	96.22
	CE-PG	1692	47309	55.92	0.4467	152.6
	SC-GT	1708	33982	39.79	0.3491	109.9
SOCFB	Amherst41	2235	90954	81.39	0.3104	137.1
	Colgate88	3482	155043	89.05	0.2673	141.9
	Bowdoin47	2250	84386	75.01	0.289	124.2
SOC	Hamsterster	2000	16097	16.1	0.54	50.02
	Advogato	5054	39374	15.58	0.2526	70.51
	Wiki-Elec	7066	100727	28.51	0.1418	138.1
Tech	PGP	10680	24316	4.554	0.2659	42.44
	Router-RF	2113	6632	6.277	0.2464	27.67
	WhoIS	7476	56943	15.23	0.4889	150.9

of nodes discovered after a crawler performs some specified number of queries. To compare the similarity of two samples S and $S^{'}$ (each produced with the same number of queries), we use the Canberra Distance, since the output is normalized between 0 and 1. So, the similarity between S and $S^{'}$ is defined as

$$sim(M(S), M(S^{'}) = 1 - d_{canberra}(|V_s^{'}|, |V_{s'}^{'}|)$$

Statistics of each network are listed in Table 5.2 and the results are illustrated in Figure 5.2. In this figure, each point represents the sampling robustness of a network, as computed and aggregated over 10 trials. The x-axis represents the network type and the y-axis represents the sampling robustness. The error probability

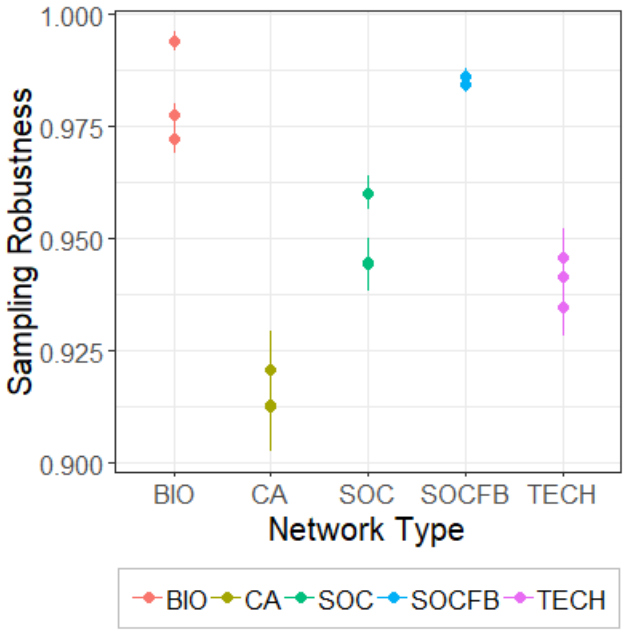


Figure 5.2: Aggregate results of $R_p(G, BFS)$ when p is ranging between 0.1-0.5. Each point represents a network from various categories (along x-axis). Sampling robustness highly depends on network type.

p is varied between 0.1 and 0.5. A BFS crawler is used to collect samples from these networks. Similar results were obtained for other tested crawlers (Random walk and MOD, specifically).

As we can clearly see in Figure 5.2, networks of different types tend to have different levels of sampling robustness and networks in the same category have similar sampling robustness. This supports our hypothesis. As a result, biological and Facebook networks tend to be the most robust, as opposed to collaboration networks, which have the lowest sampling robustness.

5.4 Characterizing Sampling Robustness

As noted in the previous sections, computing sampling robustness requires generating an error-free sample S. In real-world applications, obtaining this sample is not practical. In the previous section, we showed that one can roughly estimate

sampling robustness from the network category. In this section, we will demonstrate that $R_p(G, C)$ highly depends on the structural properties of both the original network G as well as the obtained network samples S'.

In Chapter 3, our work demonstrated that the structural properties of the network play the important role in network sampling [6]. Specifically, certain network properties enhance (or degrade) the efficiency of a crawler.

When sampling error occurs, certain edges may be invisible to the crawler. The ability of the crawler to expand the sampled network may thus drop, because the crawler makes its query decisions based on the nodes and edges in the sampled network that it has observed so far. What, then, are the properties that make a network robust to sampling?

In this section, we investigate three properties that we believe support a crawler in expanding a sample's boundary; 1) the largest eigenvalue of the adjacency matrix 2) average node degree and 3) average clustering coefficient.

The largest (or leading) eigenvalue λ_1 and average node degree are closely related, since λ_1 is bounded by the degree of a network [76]. The key idea is that a network is robust when a crawler can quickly find hub nodes (here, a node with degree larger than average). Intuitively, if the average degree of a network is k and p percent of edges are missing, this means that a crawler will discover a maximum of around $k \times (1 - p)/100$ nodes for each query on average (the number of discovered nodes will be lower if there is a lot of redundancy). If the average degree is low, the crawler will require more time to get to these hub nodes, since it is less likely that one (or more) of the neighbors of the current queried node will be a hub. However, when the average degree is high, the chance that the crawler will discover these hub nodes will increase. Thus, a network with a high average degree helps a crawler in expanding its sample and tends to be more robust against the missing edges.

Similarly, a network with high average clustering coefficient indicates dense connections between nodes. This indicates that nodes have many redundant edges (or paths) which connect from other nodes to itself. Thus, even some edges are missing, a crawler should be able to visit nodes easily because of these alternative paths, making the network more robust to the missing edges.

In this experiment, we calculate sampling robustness by using four different performance measures, which are described in subsection 5.2.4. Three crawlers, i.e. BFS, Random walk and MOD, are used to collected the samples from networks listed in Table 5.2. Error probabilities are varied between 0.1 to 0.5 and the results are aggregated over 10 runs.

As we will see, sampling robustness is highly dependent on these network properties, as computed in both the original network as well as the sample generated with errors. Figure 5.3, 5.5 and 5.7 illustrate the correlation between sampling robustness and each property on original network G. While Figure 5.4, 5.6 and 5.8 demonstrate the correlation between sampling robustness and each property on the obtained sample $S^{'}$.

5.4.1 The largest eigenvalue of the adjacency matrix

The *largest* or leading eigenvalue of the adjacency matrix A, denoted by λ_1, plays an important role in forecasting epidemic spreading processes.

As shown in [76], the leading eigenvalue λ_1 is related to the epidemic threshold τ, which governs how quickly disease can spread through a network via the SIR model. It has been shown that $\tau = \frac{1}{\lambda_1}$, so, one can predict whether an epidemic will die out on any given network by considering only a single parameter. In the SIR model, where β is the birth rate and γ is the curing rate, the epidemic will die out iff $\frac{\beta}{\gamma} < \tau$.

The epidemic process and network data collection process have similar dynam-

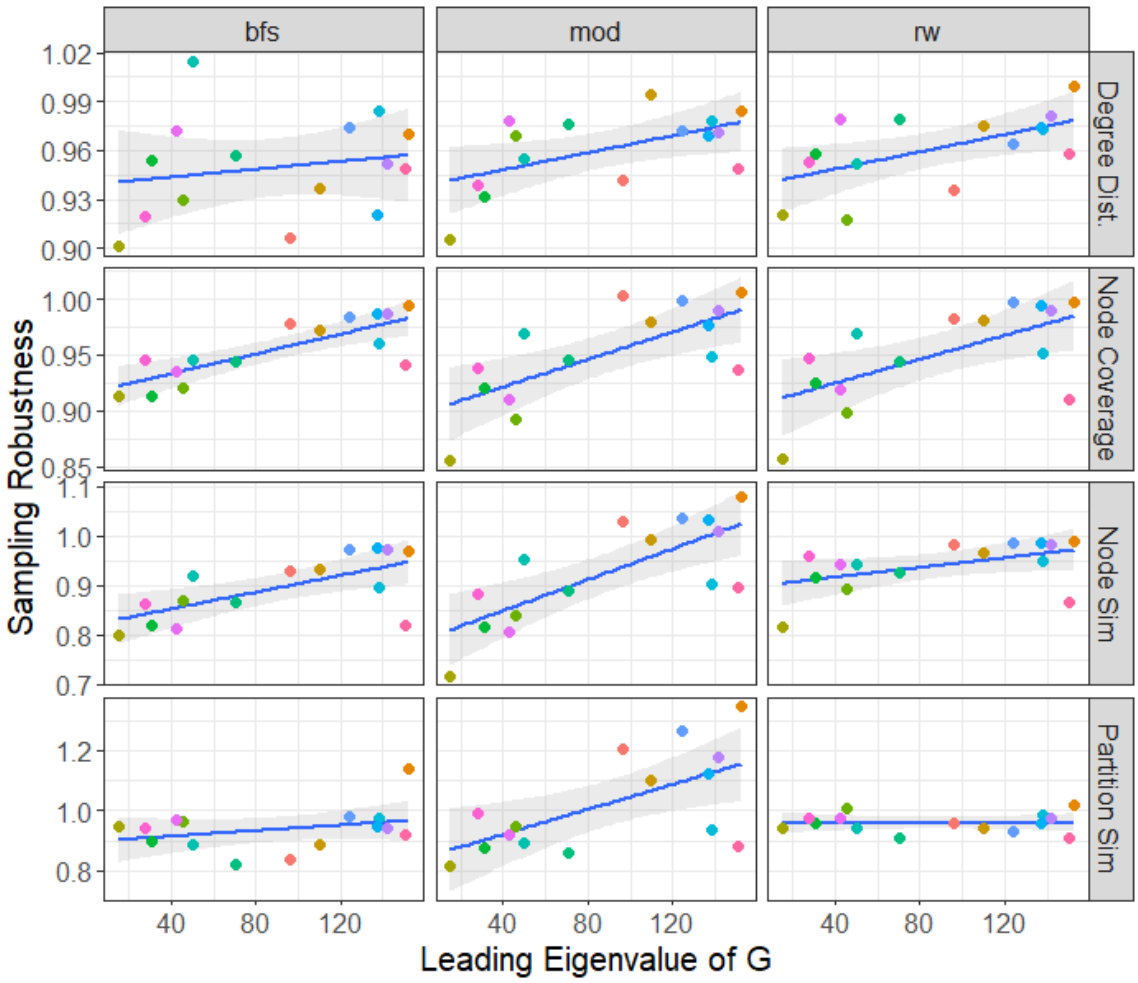


Figure 5.3: Aggregate results of $R_{p:0.1\to0.5}$ against leading eigenvalue λ_1. Each point represents a network. Sampling robustness highly depends on λ_1.

ics. Both start from a single seed node and gradually expand outwards. In both cases, we look at the population of interest after t time steps: i.e., how many people get the diseases or how many distinct users a crawler discovers through crawling.

We can consider network crawling to be a simpler version of epidemic model, where γ is a constant and β is a *crawl rate* (e.g., number of requests per second, number of new nodes added to sample). So, we can use λ_1 to indicate how fast the crawler can expand the sample. Since λ_1 is bounded by the average degree, the larger λ_1 means the higher average degree. As we will see in Figure 5.5 and 5.6,

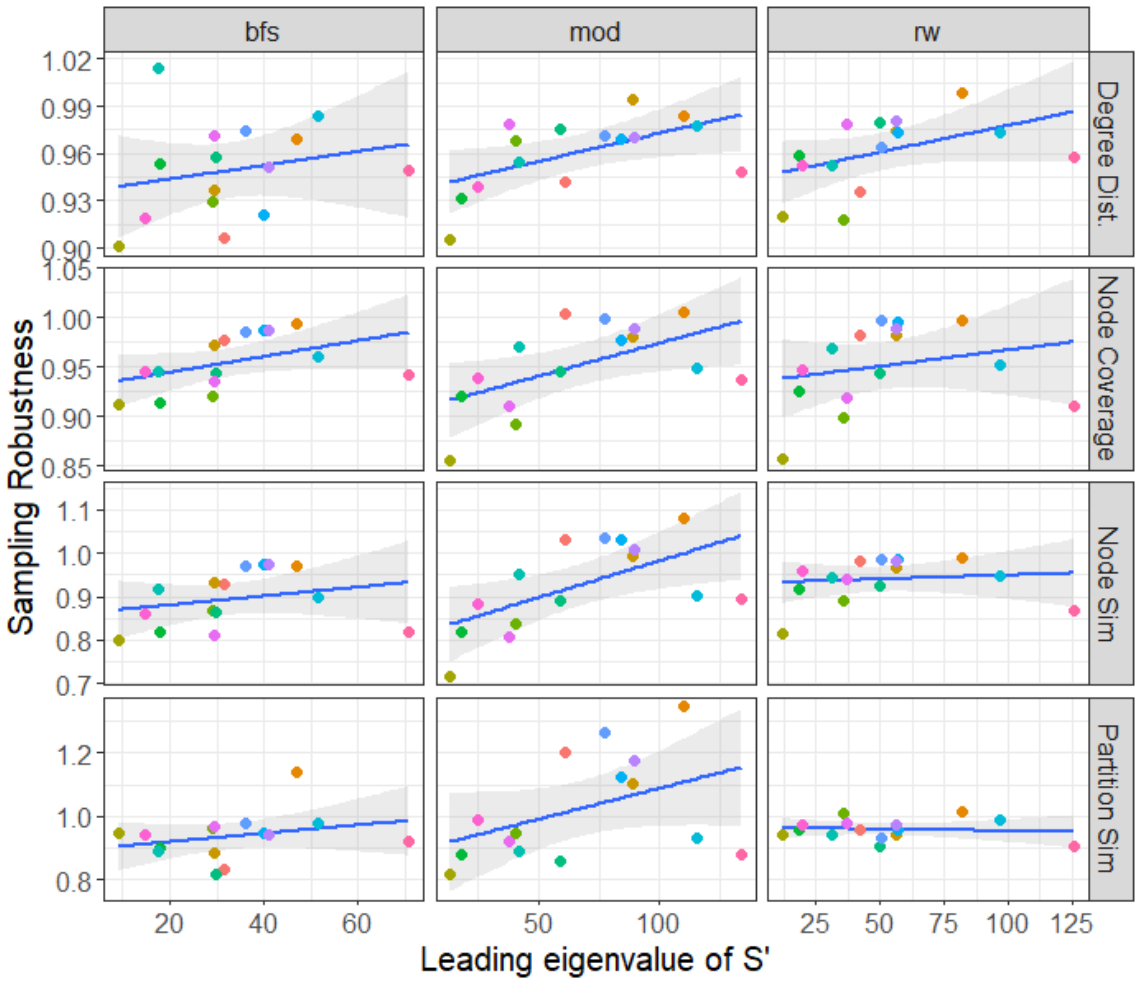


Figure 5.4: Aggregate results of $R_{p:0.1\to0.5}$ against leading eigenvalue λ_1. Each point represents a network. The results illustrates that sampling robustness highly depends on leading eigenvalue λ_1.

higher average degree indicates that a crawler can easily expand its sample.

In Figure 5.3, we plot sampling robustness against the leading eigenvalue λ_1 of the adjacency matrix A of the network G. The leading eigenvalues of the error-containing sample $S^{'}$ are shown in Figure 5.4. In these figures, each point is the sampling robustness of a network, computed as an average over 10 experiments. Each column illustrates a case when different crawling technique is used and each row represent the the sampling robustness when different performance measure $M(\cdot)$ is used.

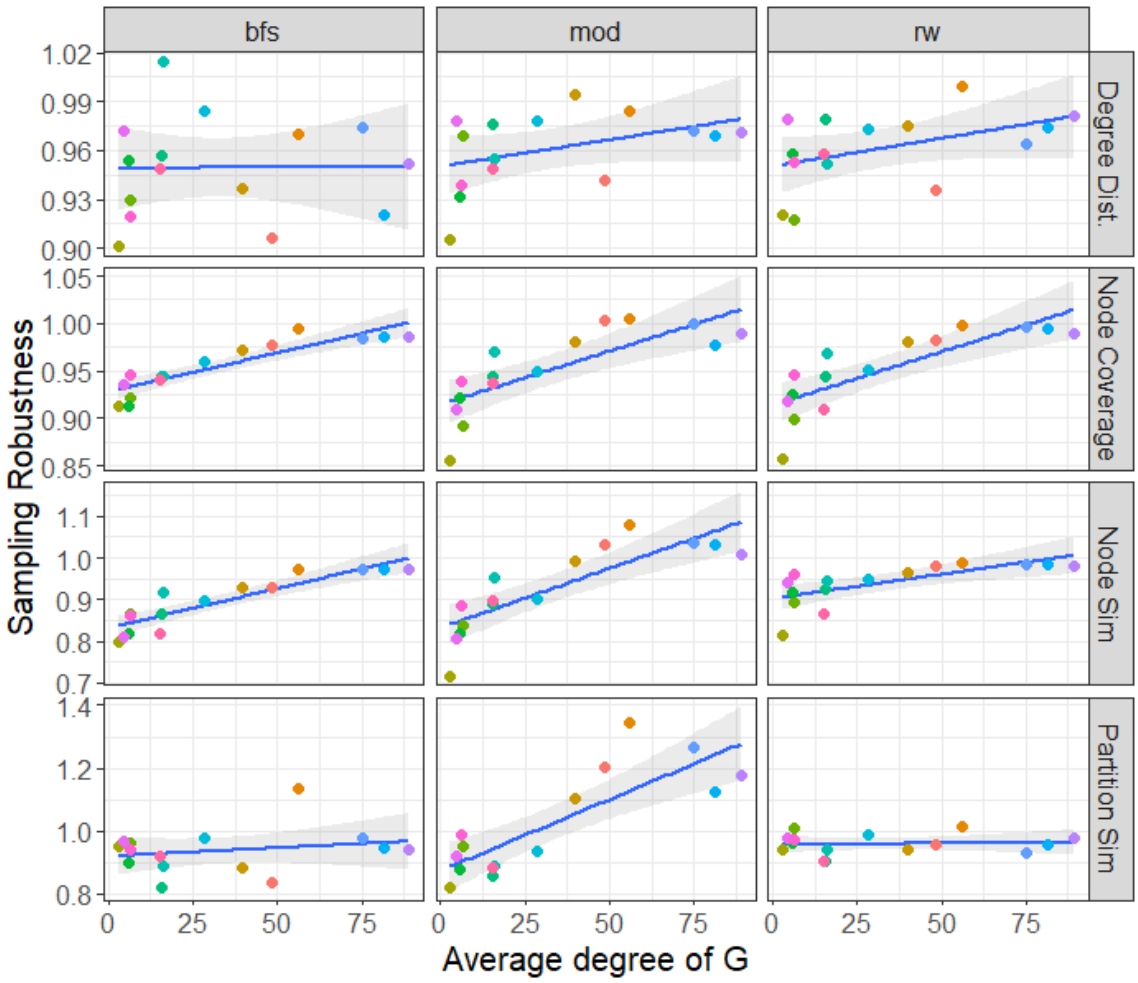


Figure 5.5: Aggregate results of $R_{p:0.1\to0.5}$ against average degree. Each point represents a network. Sampling robustness highly depends on average degree of network G.

Sampling robustness and leading eigenvalue are highly correlated. As expected, we observe that a network with higher leading eigenvalues (lower epidemic threshold τ) is more robust. This indicates that a crawler can easily expand its sample even the edges are missing.

5.4.2 Average node degree

Next, the average degree indicates the average number of neighbors of each node (e.g. average number of friends of users on social networks, average number of

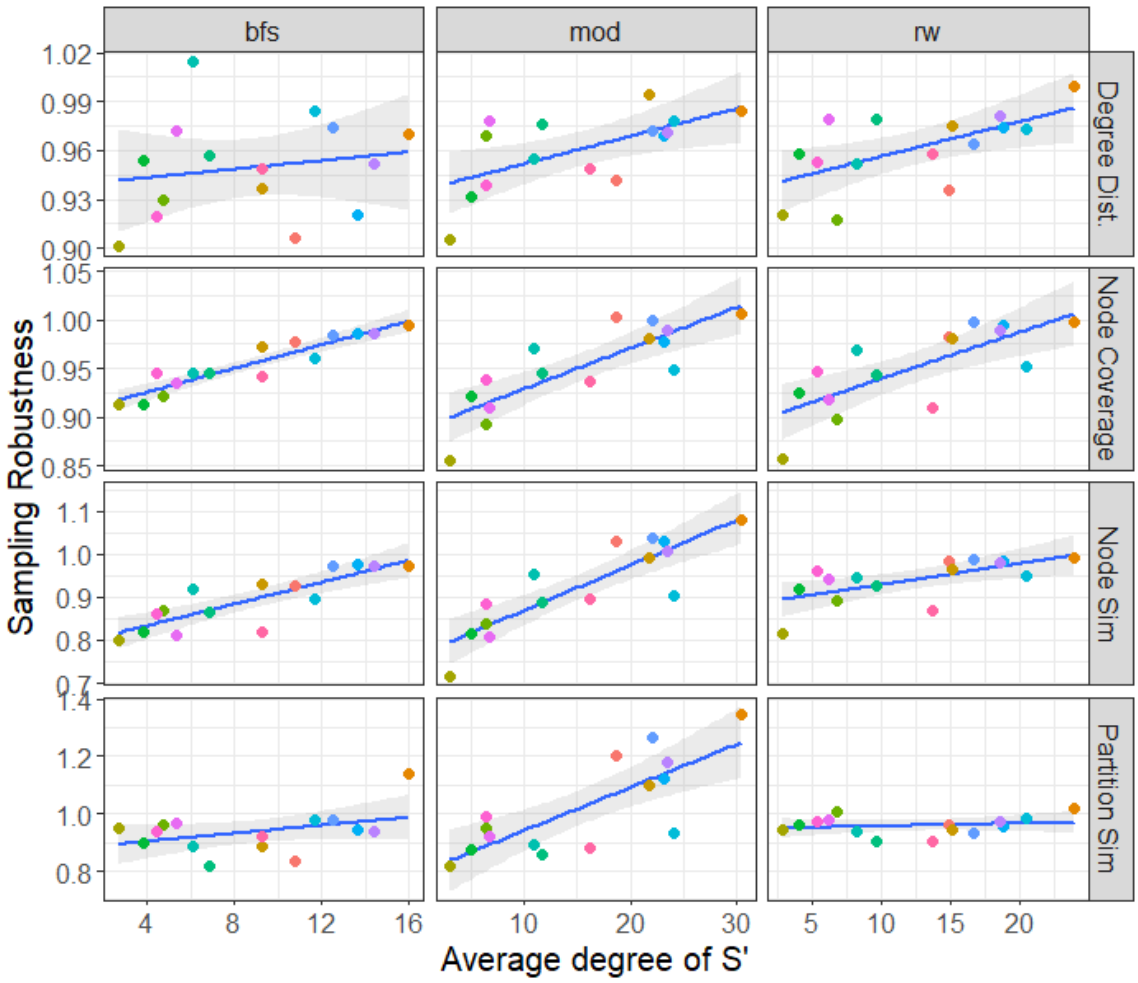


Figure 5.6: Aggregate results of $R_{p:0.1\to0.5}$ against average degree. Each point represents a network. The results illustrate that sampling robustness highly depends on average degree of the sample.

co-authors on collaboration networks). Intuitively, a crawler can quickly expand its sample if the average degree is large, and can continue to do so even if some of the edges are lost.

Figure 5.5 illustrates the average sampling robustness against average degree of a network. The average observed degree of samples are shown in Figure 5.6. The results are aggregated over 10 runs when p is varied from 0.1 to 0.5. Each column represents results from different types of crawler and each row represent when different performance measure $M(\cdot)$ is used. As we expected, the sampling

robustness is also highly correlated with the average degree of the networks as well as the average observed degree of the obtained samples.

5.4.3 Average clustering coefficient

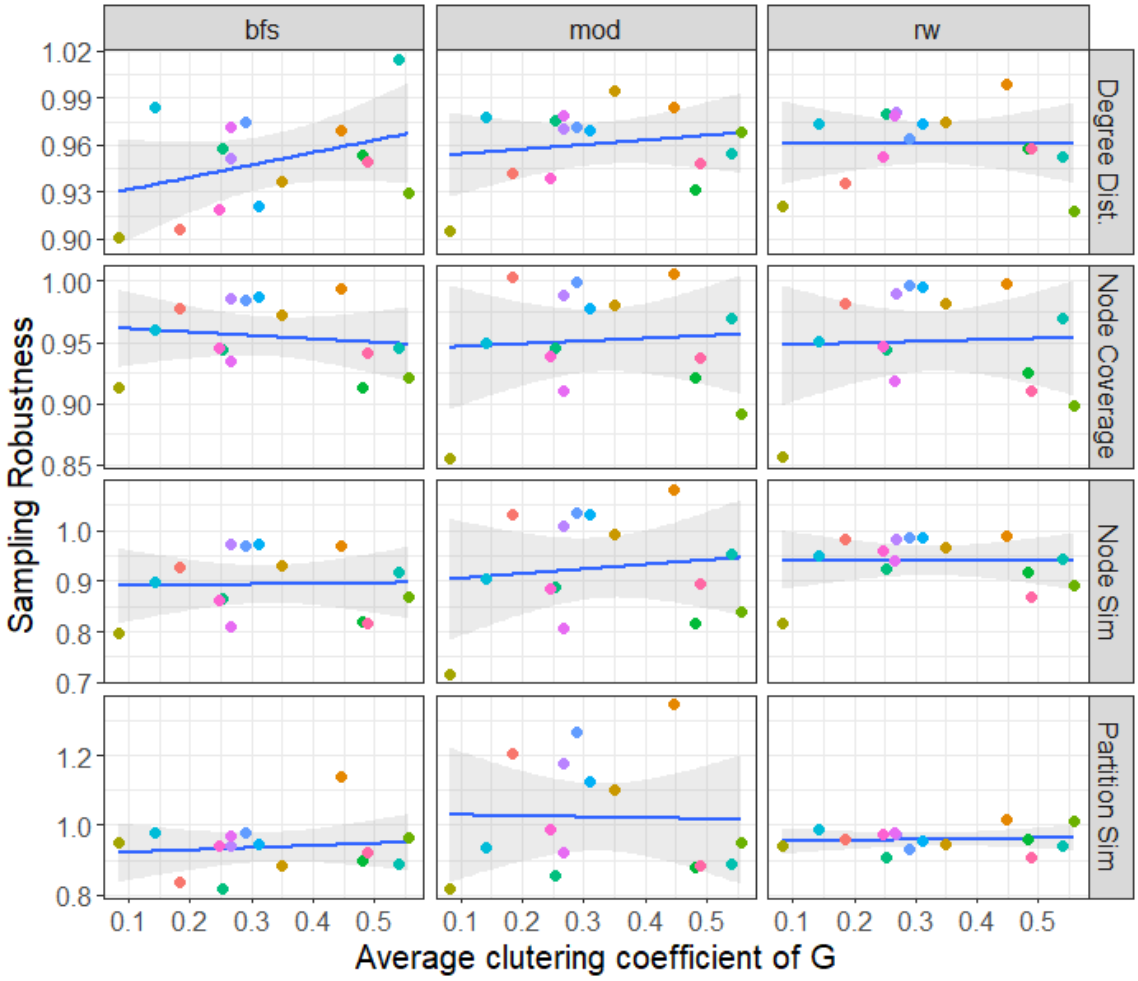


Figure 5.7: Aggregate results of $R_{p:0.1\to0.5}$ against average clustering coeficient. Each point represents a network. Sampling robustness seems not to depends on average clustering coefficient of the network G.

Lastly, we consider the average clustering coefficient of the original network G and the obtained samples S'. This property measures how well nodes are connected. A higher clustering coefficient indicates that neighboring nodes are densely

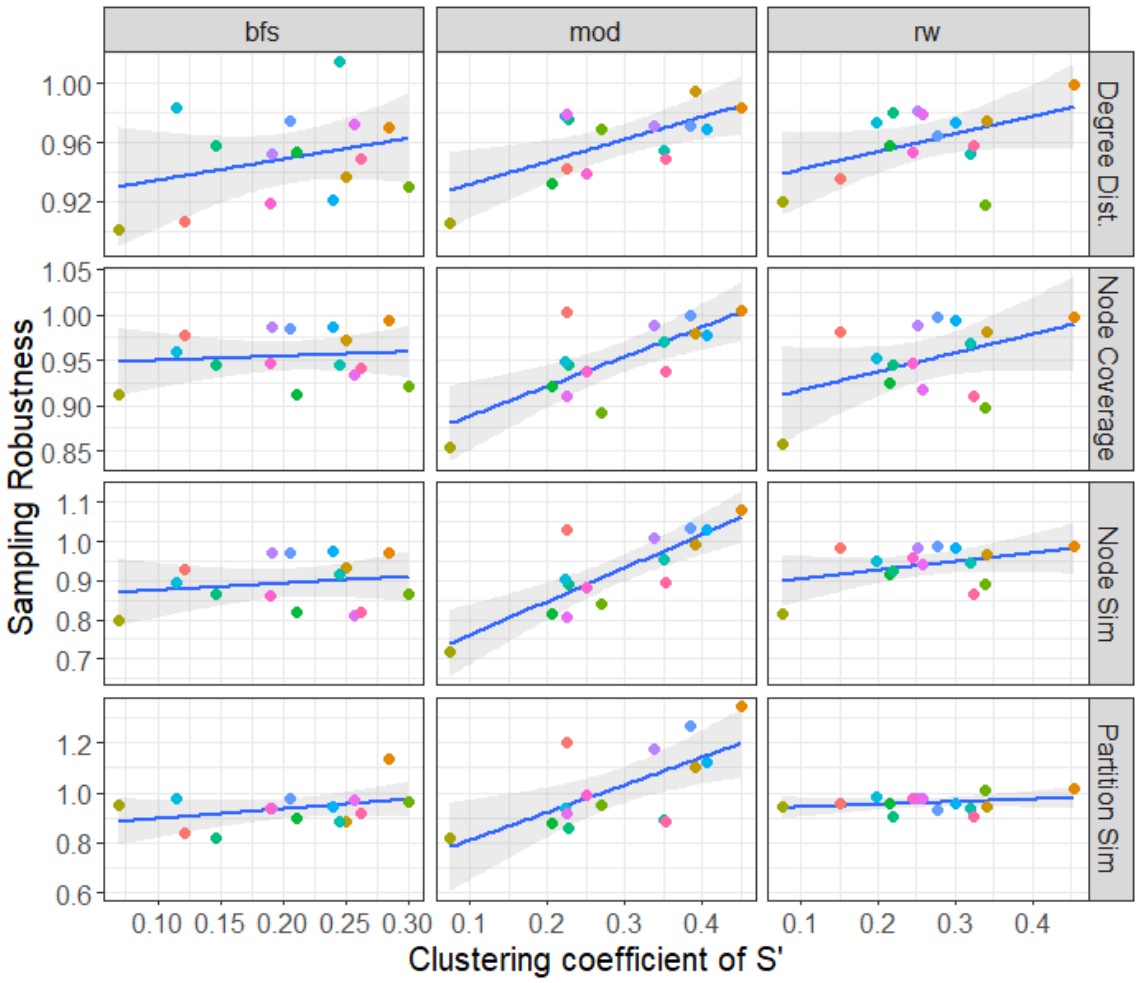


Figure 5.8: Aggregate results of $R_{p:0.1\to0.5}$ against average clustering coefficient. Each point represents a network. The results illustrate that sampling robustness highly depends on average clustering coefficient of the sample.

connected to each others. Intuitively, when nodes are densely connected (near clique structure), the crawler will discover nodes quickly, and is more robust against missing edges.

In Figure 5.7, we observe that the clustering coefficient of a network G is not correlated with its sampling robustness.

However, we do observe that the clustering coefficient of a sample S' is correlated with the sampling robustness, as illustrated in Figure 5.8. This may be because a large portion of the nodes in network G have degree 1 due to the power-law

distribution, and these nodes bring down the average clustering coefficient overall. On the other hand, our selected crawlers are known to be biased toward hub nodes [42, 29], thus, the sampled networks contain nodes with high degree connecting to each others. The sampled network represents the inner-core structure of the network rather than the periphery, which is a better indicator for measuring robustness.

5.5 Sampling Robustness and Error Probability

In this section, we investigate the sampling robustness when error probability p is varied. In this experiment, we varied the error probability between 0.1 and 0.5. Again, four performance measures $M(\cdot)$ are used to calculate the value of sampling robustness and three crawlers are used to collect the network samples. These results are shown in Figure 5.9.

In Figure 5.9, each row represents results when different performance measure is used and each column represents different network type. For each subplot, x-axis represents the error probability p (ranging from 0.1 to 0.5), while y-axis represents sampling robustness. The results are aggregated over 10 runs.

We can clearly see that sampling robustness decreases as error probability p increases, which is as expected. However, we observe that some network types tend to be more robust than others. For example, we can clearly see that sampling robustness of the collaboration networks drops faster than the value of Facebook100 networks when error probability p increases from 0.1 to 0.5. Similarly, sampling robustness of technological networks drops faster than sampling robustness of biological networks.

As a result, we observe that collaboration and technological networks are sensitive to random edge deletion. The robustness of collaboration and technological

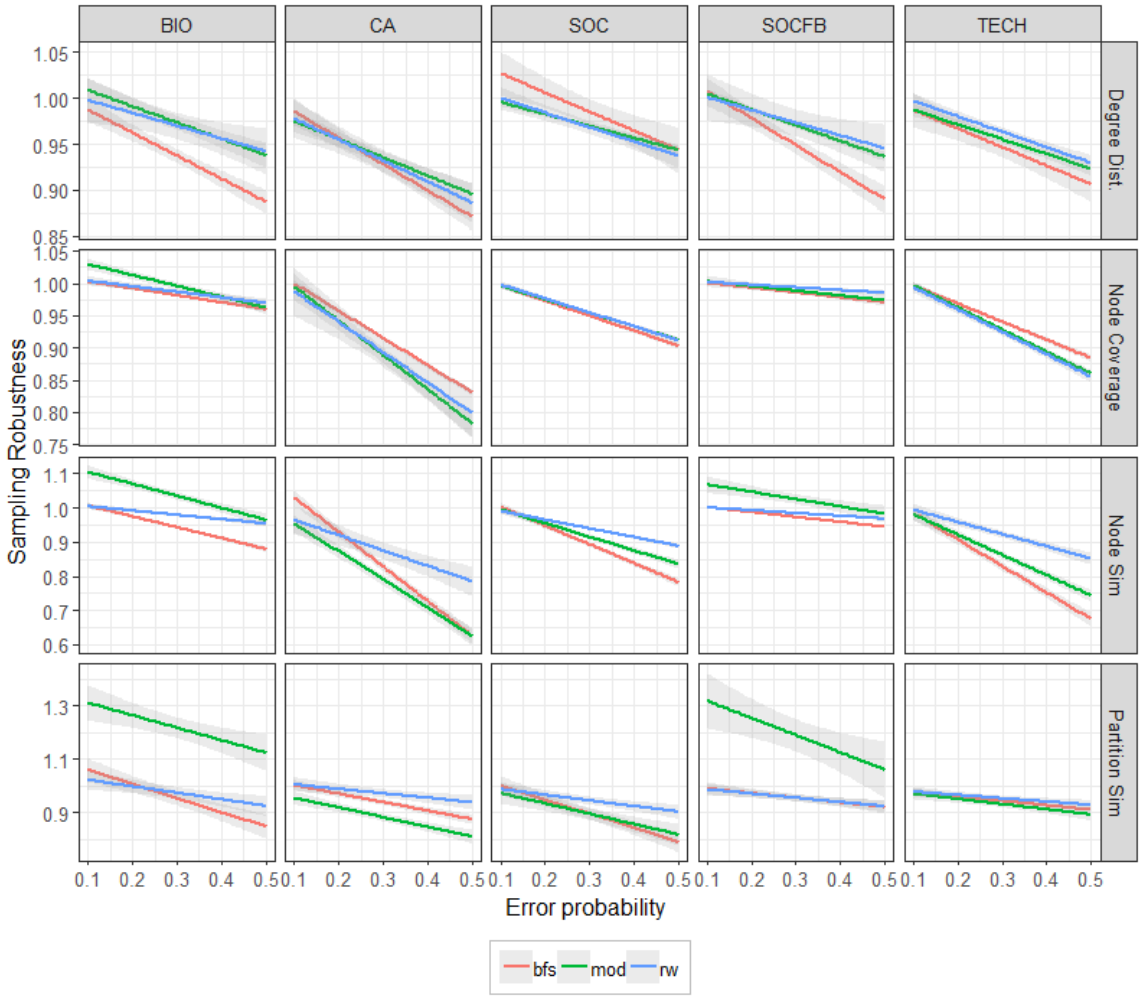


Figure 5.9: Aggregate results of sampling robustness (y-axes) when p is ranging between 0.1 - 0.5 (x-axes). Each row represents when different performance measures $M(\cdot)$ are used and each column represents the results on networks in different types.

networks drop to around 0.65 and 0.7, respectively, in the worst case. This is because networks in these categories tend to have low average degree (the average degree is around 5). Thus, when edges are missing, a crawler has some difficulty in moving to different parts on the network which degrades the performance of a crawler.

On the other hand, Facebook100 networks seems to be the most robust as compared to networks in other types. The value of sampling robustness slightly drops as error probability increases. These networks are very robust to missing edges,

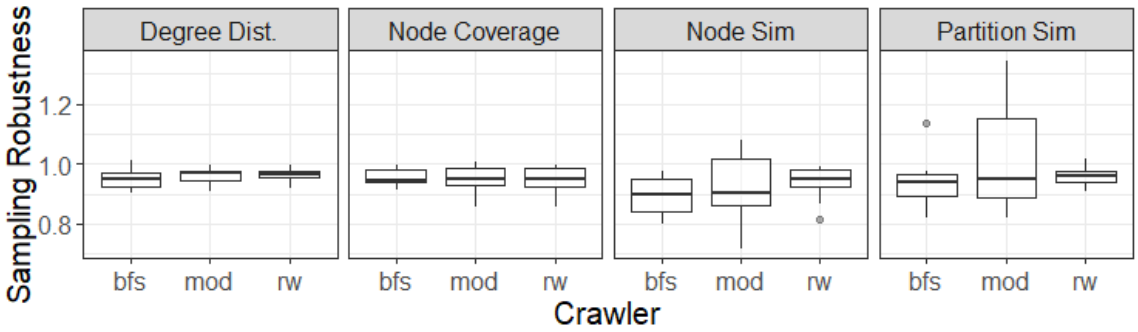


Figure 5.10: Aggregate results of sampling robustness when p is ranging between 0.1 - 0.5. Each plot represents the results when calculated by different performance measure $M(\cdot)$. The results illustrate that sampling robustness does not depend on crawling technique.

its robustness is around 0.9 even 50% of edges are missing. This is because these networks have very high average degree ($\bar{d} \approx 80$). So, even half of the edges are missing, the crawler is still capable of moving between various regions on these networks, which seems not to affect the performance of the crawler.

5.6 Sampling Robustness and Different Crawling Techniques

In this section, we investigate sampling robustness when different crawlers are used for collecting the samples from networks. Here, we focus on three different crawlers: BFS, Random walk and MOD crawler. We measure the sampling robustness of networks listed in Table 5.2. The error probability is varied between 0.1 and 0.5. The aggregate results are from 10 runs and are illustrated in Figure 5.10. For each subplot, it represents results when different performance measure is used. The x-axis represents different crawlers, while y-axes represent the level of sampling robustness.

As we can see from this figure, the levels of sampling robustness seems to be similar regardless of the type of the crawler and performance measure. We can ex-

pect to see more variation on networks with MOD crawler and partition similarity are used.

5.7 Sampling Robustness Estimation

In this section, we introduce a regression model which we can use to estimate a sampling robustness of any network given an obtained sample. We describe how to estimate error probability p and show how we construct our models for estimating sampling robustness in subsection 5.7.1. Then, we evaluate the models and report the results in subsection 5.7.2.

5.7.1 Building a Model

Previously, we observed that there is a relationship between sampling robustness and properties of the collected samples. In addition, we clearly see that sampling robustness also decreases as error probability p increases. These are the four main factors which have effect on the sampling robustness. Thus, we can use them as predictor variables in our model.

To estimate the robustness of the original graph G, we only have a sample $S^{'}$ which obtained by a crawler C. We build a linear regression model using an error probability p and three network properties (as we described in the previous section) as the dependent variables. This model represents the relationship between the response variable ($\hat{R}_p$) and predictor variables ($\bar{d}^{'}$, $\bar{cc}^{'}$, $\lambda_1^{'}$, p). However, before the estimation, uses need to estimate the error probability p.

Estimating error probability: The probability p can be estimated by performing multiple queries on the same node and counting the number of times a particular edge is duplicated. Let k be the number of times a crawler queries node u, and e be one of the edges incident to node u in G. So, p can be estimated by

Table 5.3: The summary statistics of the constructed model. This linear model has moderate value of R-square and the model is statistically significant since the p-value of the model and the p-value of the individual (including their interaction terms) are less than 0.01.

	Coefficient	Std. Error	t-value	p-value
(Intercept)	0.9696104226	1.600e-03	606.036	< 2e-16
p	-0.1915773060	2.457e-03	-77.957	< 2e-16
$\bar{d}$	0.0061236904	1.016e-04	60.297	< 2e-16
λ_1	-0.0004802188	2.823e-05	-17.009	< 2e-16
$\bar{cc}$	0.0224243976	5.001e-03	4.484	7.43e-06
Residual standard error: 0.02765 on 7495 degrees of freedom				
Multiple R-squared: 0.6589, Adjusted R-squared: 0.6587				
F-statistic: 3619 on 4 and 7495 DF, p-value: < 2.2e-16				

$p = 1 - \frac{k_e}{k}$, where k_e is the number of times edge e is seen after k queries on u. Users can estimate p with a small k. In our analysis, we assume that these multiple queries are performed using a small amount of budget, and is done after obtaining the samples.

Model 1: Multiple Linear Regression

As mentioned earlier, we consider four variables in this model ($\bar{d}^{'}$, $\bar{cc}^{'}$, $\lambda_1^{'}$, p). Thus, the relationship between sampling robustness and the predicted variables can be shown as

$$\hat{R}_p = \beta_1 \cdot p + \beta_2 \cdot \vec{d}^{'} + \beta_3 \cdot \lambda_1^{'} + \beta_4 \cdot \bar{cc}^{'} + b,$$

where $\beta_{1...i}$ are the coefficients, p is an error probability, $\vec{d}^{'}$, $\lambda_1^{'}$ and $\bar{cc}^{'}$ are the average degree, leading eigenvalue and average clustering coefficient of a sample $S^{'}$, respectively.

In order to build the model, we train our model from the sampled networks which we obtained in the previous experiment. In total, there are 7,500 data points in the training set. The statistics of the constructed model are listed in Table 5.3.

Table 5.3 shows the coefficient of each predictor variable, standard error, and its significant level. As we can see, this model shows a statistically significant since the model p-value is less than 0.01. In addition, all the predicting variables are also significant (p-value < 0.01). However, the R-squared and Adjusted R-squared of this model are around 0.658, which indicates some room for improvement.

Model 2: Multiple Linear Regression with Interaction

To improve the predicting power of the previous model, we will consider the interaction between variables. We hypothesize that *each property affects sampling robustness differently on different level of error probability*. Therefore, we add three interaction terms to the model. The model can be described as follows,

$$\hat{R}_p = \beta_1 \cdot p + \beta_2 \cdot \vec{d}^{'} + \beta_3 \cdot \lambda_1^{'} + \beta_4 \cdot \bar{c}\bar{c}^{'} + \beta_5 \cdot p \cdot \bar{d} + \beta_6 \cdot p \cdot \bar{c}\bar{c} + \beta_5 \cdot p \cdot \lambda_1 + b$$

Again, we construct the model from the sampled networks listed in Table 5.2. The summary statistics of the model are in Table 5.4. This model also shows statistical significance, since the model p-value is very low (less than 0.01) and all the predicting variables including the interaction terms are significant (p-value < 0.01). In addition, this model gives R-squared and Adjusted R-squared of 0.8649 and 0.8648, respectively. All of the variables are significant by the t tests. This indicates that this model can explain the variation in response very well. These statistical tests give us a strong evidence that this model significantly improves the model over the previous one by adding the interaction terms.

Table 5.4: The summary statistics of the constructed model. This linear model has a great value of R-square and the model is statistically significant since the p-value of the model and the p-value of the individual (including their interaction terms) are mostly less than 0.05.

	Coefficient	Std. Error	t-value	p-value
(Intercept)	1.0470999071	1.680e-03	623.099	< 2e-16
p	-0.4744817968	4.915e-03	-96.547	< 2e-16
$\bar{d}$	-0.0049934093	1.354e-04	-36.867	< 2e-16
λ_1	0.0003848386	3.838e-05	10.028	< 2e-16
$\bar{cc}$	-0.0117558437	6.712e-03	-1.752	0.07989
$p \times \bar{d}$	0.0457005403	4.877e-04	93.705	< 2e-16
$p \times \bar{cc}$	0.0689145597	2.200e-02	3.133	0.00174
$p \times \lambda_1$	-0.0037103293	1.327e-04	-27.953	< 2e-16
Residual standard error: 0.01741 on 7492 degrees of freedom				
Multiple R-squared: 0.8649, Adjusted R-squared: 0.8648				
F-statistic: 6851 on 7 and 7492 DF, p-value: < 2.2e-16				

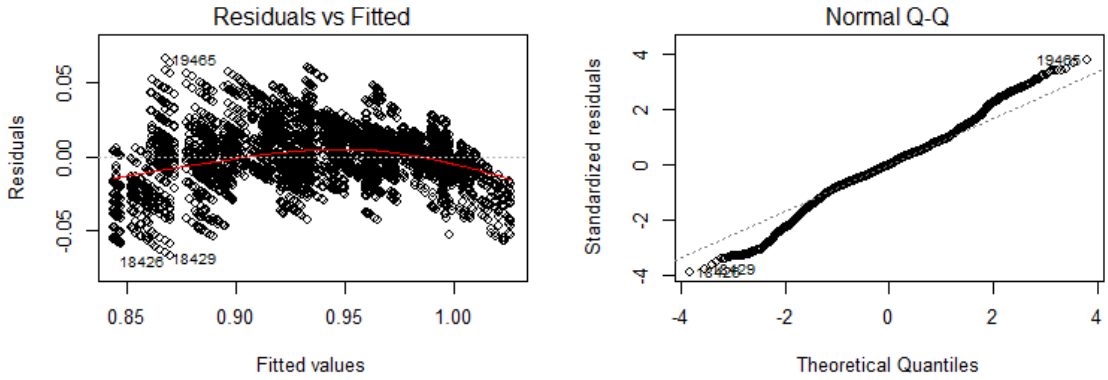


Figure 5.11: Regression diagnostic plots. (Left) Residuals vs. Fitted values plot. Residuals are scattered around 0, which indicates a linear pattern. (Right) Q-Q plot of the residuals of model 2. This shows that the residuals are normally distributed.

Validity of the Model

Although, this model gives a high R-squared (0.8649), we also investigate the model further by performing a residuals analysis. We analyze the residuals and

Table 5.5: Statistics of network used for model testing. We generate around 600 samples networks in total by using BFS, MOD and Random walk crawlers.

Network	$\|V\|$	$\|E\|$	$\bar{d}$	$\bar{cc}$	λ_1
Hamilton46	2312	96393	83.38	0.2983	135.93
Trinity100	2613	111996	85.72	0.2903	135.83
Epinion	26588	100120	7.53	0.1351	66.206
Caida2007	26475	53381	4.03	0.2082	69.643

illustrate in Figure 5.11.

The left plot in Figure 5.11 shows residuals and fitted values. As we can see, the residuals are equally scattered around horizontal line (dotted-line at 0). This indicates the linear relationship between predictor variables and response variable. It shows a good evidence that this model meets the *linearity* and *homoscedasticity* assumptions for linear regression model.

On the right of Figure 5.11, it illustrates the normal Q-Q plot. This plots demonstrates that residuals are normally distributed (residuals follows a diagonal straight line). This gives us a strong evidence that the model satisfies the *normality* assumption.

5.7.2 Model Evaluation

To test our model, we use samples generated from the networks listed in Table 5.5. These networks were not used in generating the regression model. We use these networks as the original networks G, and use the BFS, random walk, and MOD crawlers to generate samples. The error probability p ranges from 0.1 and 0.5. For each network, ten network samples are generated using each crawler, for each of p. In total, we have around 600 sampled networks.

We evaluate our models through min-max accuracy and mean absolute percentage error (MAPE). The accuracy is calculated from the correlation between

the actual and predicted values. Higher accuracy indicates that the actual and predicted values have similar trend (i.e. predicted values increase as actual values increase), so, higher is better. While MAPE is a measure of how accurate the model is (the lower is better).

As a result, the accuracy of this model is 0.9012342, while MAPE is 0.02499224. The evaluation results show a very high accuracy and low error. This indicates that our proposed model is capable of estimating sampling robustness of a network G from a sample $S^{'}$.

Through this method, users can estimate the sampling robustness of any network given an obtained sample. We evaluate the model and the results show that our model have great R-squared and it estimates sampling robustness with high accuracy and a small error. This lets the user understand whether the results of an analysis performed on a particular sample with errors are a good representation of the results one would have gotten from analyzing a sample without errors.

5.8 Conclusion

In this chapter, we presented a novel network robustness measure called “sampling robustness", which measures how much the performance of a network crawler changes when the edges are missing during the data collection process. We present four different performance measures that can be used for calculating sampling robustness. The performance measure can be varied and it is dependent on the sampling goal. We demonstrated that different network types have different level of robustness, and that sampling robustness is highly dependent on the structural properties of the original graph. In addition, it is also correlated with the structural properties calculated from the obtained network samples. We presented a linear model for estimating sampling robustness from properties of the obtained

sample. As a result, our proposed model has high R-squared and it is capable of predicting sampling robustness with high accuracy with low error.

Chapter 6
Conclusion

In this dissertation, we considered the problem of network sampling through crawling, in which the data collector have no knowledge about the network of interest except the identity of a seed node (e.g. the first person that the data collector will start to query). The data collector can obtain more information about the network by querying observed nodes that obtained so far under a given budget. We began by presenting and summarizing many important crawling techniques from the network science literature. However, the existing literature contained no clear conclusion on which method should be used for the task of data collection. As observed in this dissertation, a method may work the well on one type of network but fail on other types of network. Therefore, our first goal was to understand the behavior of these crawlers when across networks. In particular, we were interested in examining the factors which can improve (or degrades) the performance of the crawler.

In the first chapter, we examined the interplay between crawler performance and the structure of the network on many synthetic and real networks. We studied the performance of the crawler under the goal of maximizing a number of nodes observed. We focused on three properties which we hypothesized to have

an effect on crawler performance: community mixing, average degree and community size. These properties govern the ability of a crawler to move between regions of the network. We considered five different types of query responses on both undirected and directed networks, and chose eight commonly used crawling techniques from the literature, which we grouped them into three categories as based on their performance. We found that greedy methods (i.e., MOD, OPIC and PageRank) perform the best on networks with overlapping communities. These methods are capable of finding nodes in dense regions in a few iterations, but their performance is obstructed by sharp community borders. On the other hand, a random walk crawler performs the best on networks that have disjoint community structure. This crawler has the ability to discover many partial regions (e.g. communities) of the networks. We performed extensive experiments on networks from different domains. Finally, we provided guidelines on how a data collector can select an appropriate method for network crawling when type of network of interest is known even before observing properties of that specific network.

Next, we presented a novel crawling technique called **DE-Crawler**, which combines methods that can quickly explore nodes in an individual region and those that can move freely between regions. **DE-Crawler** consists of two stages: the *densification* stage aims to find nodes in a particular region with a minimal amount of budget, and the *expansion* stage aims to find a new region which has not been explored yet. The algorithm automatically switches between these two stages based on the sample observed so far. We compared the performance of our proposed algorithm against baselines on many diverse networks. Experimental results showed that the performance of **DE-Crawler** is consistently the best over all types of networks, with a performance improvement of up to 28% over the next baseline.

Finally, we focused on the scenario where there is an error during the data

collection process. Specifically, we examined the scenario when there are some missing edges on the response after each query, which we modeled as *random edge deletion*. These errors may lead to inaccuracy in further network analysis. Thus, it is important for a data analyst to measure the trustworthiness of the obtained samples. We presented a novel robustness method called "*sampling robustness*", which measures the extent to which a sample generated by a crawling algorithm in the presence of errors is a representative of a sample generated by the same algorithm without errors. We illustrate that sampling robustness is strongly correlated with network properties and some network types tends to be more robust than others when edges are missing. Specifically, it is highly correlated with leading eigenvalue, average degree, and clustering coefficient of the network and the obtained samples. By considering a small set of these properties, we presented a regression model for estimating the robustness of the networks. Our proposed model has an R-squared of 0.8649, indicating that our model includes only relevant predictor variables and explains the variation very well. All predictor variables are statistically significant with p-value of less than 0.01. The experimental results also show that our model can achieve high prediction accuracy (0.90) with minimal error (MAPE ≈ 0.024).

Appendices

APPENDIX A

Table A.1 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 3.4. 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 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 3.5. We provide the mean and standard deviation of the percentage of node coverage in Ta-

Table A.1: Results of the controlled experiments. An arrow indicates how the performance changes when test property changes from Low to High ($\Uparrow$: improve, $\Downarrow$: degrade, $\approx$: unchanged). In $\begin{bmatrix} x \\ y \end{bmatrix}$, x and y indicate the percentage improvement of Low- and High- valued networks, respectively ('+': outperform RW, '-': underperform RW).

		Comp	**Page**	**Part**	**In-Out**	**Out**
Improvement of G1 vs. RW						
Q	P_1	$\Downarrow \begin{bmatrix} 7.62\% \\ -5.24\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 7.19\% \\ -6.13\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} -5.84\% \\ -65.66\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 34.77\% \\ -15.07\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 22.53\% \\ 12.11\% \end{bmatrix}$
	P_2	$\Uparrow \begin{bmatrix} 12.47\% \\ 19.81\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} 8.71\% \\ 13.41\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -26.05\% \\ -14.15\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -1.73\% \\ 25.72\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} -0.01\% \\ -2.45\% \end{bmatrix}$
CS	P_1	$\Uparrow \begin{bmatrix} -71.52\% \\ -70.32\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -66.57\% \\ -66.18\% \end{bmatrix}$	$\approx \begin{bmatrix} -78.22\% \\ -80.50\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 20.76\% \\ 12.23\% \end{bmatrix}$	$\approx \begin{bmatrix} 12.05\% \\ 14.13\% \end{bmatrix}$
	P_2	$\Uparrow \begin{bmatrix} 12.04\% \\ 13.47\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} 12.33\% \\ 16.30\% \end{bmatrix}$	$\approx \begin{bmatrix} 0.92\% \\ 0.95\% \end{bmatrix}$	$\approx \begin{bmatrix} 12.92\% \\ 12.23\% \end{bmatrix}$	$\approx \begin{bmatrix} 12.53\% \\ 13.13\% \end{bmatrix}$
d	P_1	$\Uparrow \begin{bmatrix} -0.40\% \\ 6.25\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} 0.20\% \\ 3.38\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -23.20\% \\ 510.67\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 38.66\% \\ 14.05\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 22.53\% \\ -4.14\% \end{bmatrix}$
	P_2	$\Downarrow \begin{bmatrix} 10.14\% \\ -14.38\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 15.58\% \\ -12.32\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 16.84\% \\ -31.84\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 4.37\% \\ -15.78\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 24.45\% \\ -25.11\% \end{bmatrix}$
Improvement of G3 vs. RW						
Q	P_1	$\Uparrow \begin{bmatrix} -22.44\% \\ -13.97\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 17.77\% \\ -11.97\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -5.77\% \\ -0.56\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -46.56\% \\ -20.96\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -44.63\% \\ -9.47\% \end{bmatrix}$
	P_2	$\Uparrow \begin{bmatrix} -28.27\% \\ -17.64\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} -17.64\% \\ -20.54\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 0.20\% \\ -2.42\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -9.01\% \\ 3.48\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} 0.12\% \\ -3.07\% \end{bmatrix}$
CS	P_1	$\Downarrow \begin{bmatrix} -20.53\% \\ -27.68\% \end{bmatrix}$	$\Downarrow \begin{bmatrix} -19.58\% \\ -27.71\% \end{bmatrix}$	$\approx \begin{bmatrix} -57.25\% \\ -56.12\% \end{bmatrix}$	$\approx \begin{bmatrix} -15.75\% \\ -12.32\% \end{bmatrix}$	$\approx \begin{bmatrix} -2.67\% \\ -3.08\% \end{bmatrix}$
	P_2	$\Uparrow \begin{bmatrix} -31.64\% \\ -15.91\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -23.36\% \\ -7.49\% \end{bmatrix}$	$\approx \begin{bmatrix} -39.40\% \\ -38.45.\% \end{bmatrix}$	$\approx \begin{bmatrix} -14.89\% \\ -12.32\% \end{bmatrix}$	$\approx \begin{bmatrix} 1.36\% \\ 1.12\% \end{bmatrix}$
d	P_1	$\Uparrow \begin{bmatrix} -15.18\% \\ -0.87\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -8.25\% \\ 0.00\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -16.45\% \\ -2.16\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -11.97\% \\ -9.43\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -2.99\% \\ 6.80\% \end{bmatrix}$
	P_2	$\Uparrow \begin{bmatrix} 2.09\% \\ 334.34\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -95.20\% \\ 133.98\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -36.62\% \\ -31.44\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -46.56\% \\ -24.32\% \end{bmatrix}$	$\Uparrow \begin{bmatrix} -44.63\% \\ 14.77\% \end{bmatrix}$

bles A.2 and A.3 for undirected and directed networks under five different responses.

For undirected networks, G1 methods are the best method to crawl these networks (except Facebook-typed networks) under complete and paginated responses. However, Random walk crawler is the best for crawling networks under partial responses.

For directed networks, G1 methods are appropriate for crawling technological and online social network under In-Out response, while Random walk crawler is an appropriate method for networks under out responses.

Figure A.1 and A.2 demonstrates the aggregate results, when **DE-Crawler** is

Table A.2: **Undirected Networks -** Summary of the network characteristics and performance of algorithms.

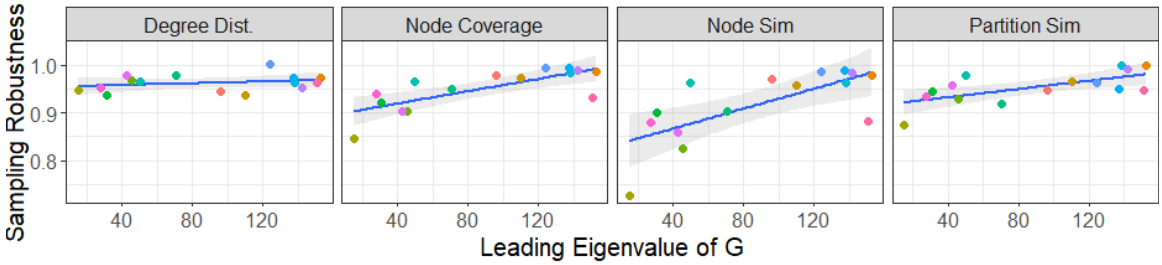
Type	Network	G1	G2	G3
		Complete Response		
Collab	Citeseer	**39.78 ± 0.85**	38.94 ± 0.46	32.29 ± 0.22
	Dblp-2010	**45.73 ± 0.07**	40.8 ± 0.37	28.84 ± 0.11
	Dblp-2012	**52.39 ± 0.08**	46.88 ± 0.2	38.26 ± 0.12
	MathSciNet	**51.15 ± 0.08**	44.94 ± 0.15	36.56 ± 0.13
Rec	Amazon	**11.35 ± 0.18**	11.25 ± 0.09	11.64 ± 0.2
	Github	**66.15 ± 0.02**	58.34 ± 0.16	39.88 ± 0.13
FB	OR	52.89 ± 1.06	**67.41 ± 0.30**	63.4 ± 0.06
	Penn	82.36 ± 0.72	**89.35 ± 0.29**	88.32 ± 0.04
	Wosn-friends	53.04 ± 1.13	**67.61 ± 0.36**	63.37 ± 0.1
OSNs	BlogCatalog	**94.99 ± 0.01**	94.21 ± 0.12	57.6 ± 0.53
	Themarker	**93.61 ± 0.00**	91.66 ± 0.14	58.4 ± 0.22
	Catster	**94.74 ± 0.01**	94.25 ± 0.05	69.77 ± 3.24
		Paginated Response		
Collab	Citeseer	**39.71 ± 0.13**	38.86 ± 0.33	34.09 ± 0.21
	Dblp-2010	**43.03 ± 0.17**	41.29 ± 0.24	31.99 ± 0.18
	Dblp-2012	**49.08 ± 0.14**	47.27 ± 0.17	40.7 ± 0.15
	MathSciNet	**43.05 ± 0.33**	45.01 ± 0.15	39.83 ± 0.25
Rec	Amazon	**11.66 ± 0.12**	11.39 ± 0.15	1.2 ± 1.21
	Github	**65.41 ± 0.07**	58.97 ± 0.12	45.11 ± 0.13
FB	OR	53.33 ± 3.62	**66.78 ± 0.50**	63.86 ± 0.21
	Penn	80.67 ± 3.26	**89.35 ± 0.28**	88.5 ± 0.06
	Wosn-friends	52.97 ± 4.47	**66.60 ± 0.59**	63.93 ± 0.18
OSNs	BlogCatalog	**96.12 ± 0.03**	93.65 ± 0.21	63.58 ± 0.77
	Themarker	**93.82 ± 0.02**	90.86 ± 0.10	63.56 ± 0.61
	Catster	**94.93 ± 0.01**	94.82 ± 0.07	81.54 ± 0.77
		Partial Response		
Collab	Citeseer	7.06 ± 1.4	**12.84 ± 0.23**	5.23 ± 0.76
	Dblp-2010	9.96 ± 0.68	**13.53 ± 0.14**	5.44 ± 0.21
	Dblp-2012	12.38 ± 0.14	**14.18 ± 0.1**1	7.57 ± 0.20
	MathSciNet	12.97 ± 0.70	**14.56 ± 0.07**	8.59 ± 0.27
Rec	Amazon	7.23 ± 0.11	**9.32 ± 0.46**	4.94 ± 1.91
	Github	22.59 ± 0.26	**45.50 ± 0.20**	26.8 ± 0.11
FB	OR	41.43 ± 5.48	**63.29 ± 0.65**	48.65 ± 0.17
	Penn	59.62 ± 4.62	**80.18 ± 0.50**	56.67 ± 1.50
	Wosn-friends	42.98 ± 4.58	**62.99 ± 0.52**	48.39 ± 0.31
OSNs	BlogCatalog	**29.72 ± 0.10**	28.18 ± 0.13	19.18 ± 0.81
	Themarker	**33.99 ± 0.15**	32.83 ± 0.12	21.69 ± 0.31
	Catster	**48.1 ± 8.81**	30.87 ± 0.17	10.92 ± 0.62

Table A.3: **Directed Networks -** Summary of the network characteristics and performance of algorithms.

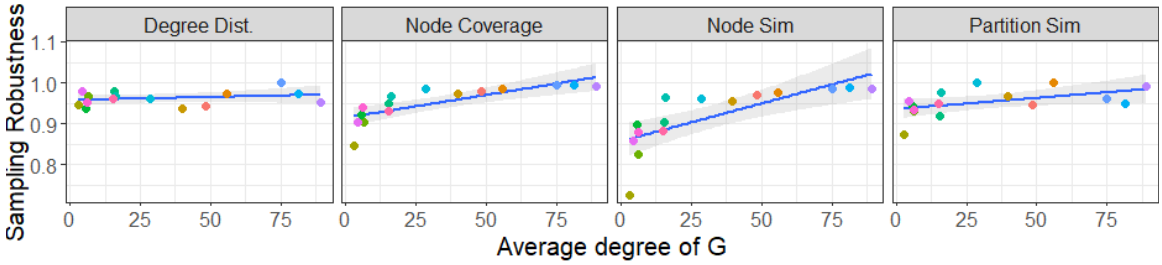
Type	Network	G1	G2	G3
		In-Out Response		
Tech	P2P-gnutella	**36.04 ± 0.12**	31.97 ± 0.24	27.73 ± 0.23
	RL-caida	**36.38 ± 0.18**	29.45 ± 0.48	26.83 ± 0.11
Web	Arabic-2005	9.33 ± 1.01	**9.90 ± 1.45**	8.58 ± 1.14
	Italycnr-2000	10.07 ± 1.92	**19.34 ± 4.5**	17.01 ± 3.43
	Sk-2005	9.30 ± 0.78	**10.01 ± 0.4**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

used as a crawler and error probability p is varied from 0.1 to 0.5, against network properties of network G and obtained sample S', respectively. Three network properties are considered, leading eigenvalue, average average degree and clustering coefficient. For each plot, x-axis shows the value of the considered properties and y-axis shows the value of sampling robustness. Each point represents each network in Table 5.2. We consider four performance measures as mentioned in Table 5.1.

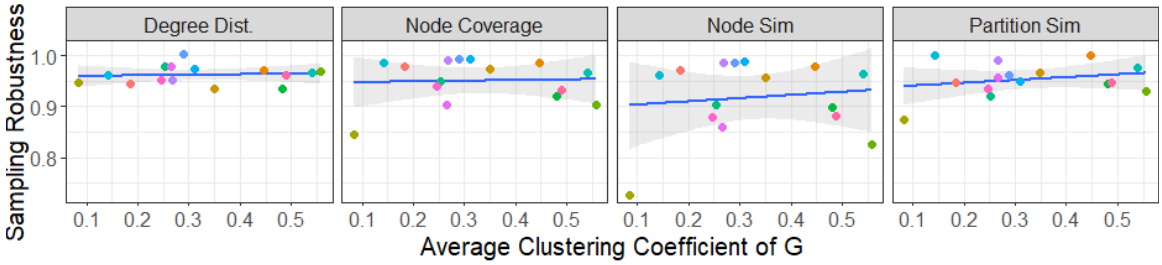
We can clearly see in Figure A.1, the sampling robustness is highly correlated with leading eigenvalue and average degree of the original network G, but it seems to be uncorrelated with average clustering coefficient of network G.



(a) Sampling robustness vs. leading Eigenvalue of an original network.



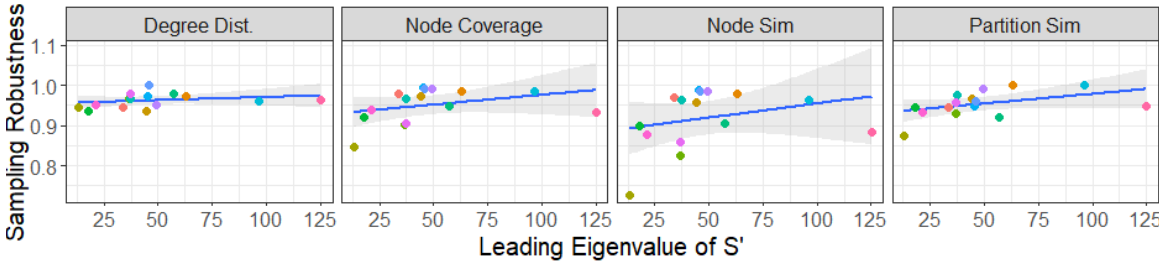
(b) Sampling robustness vs. average degree of an original network.



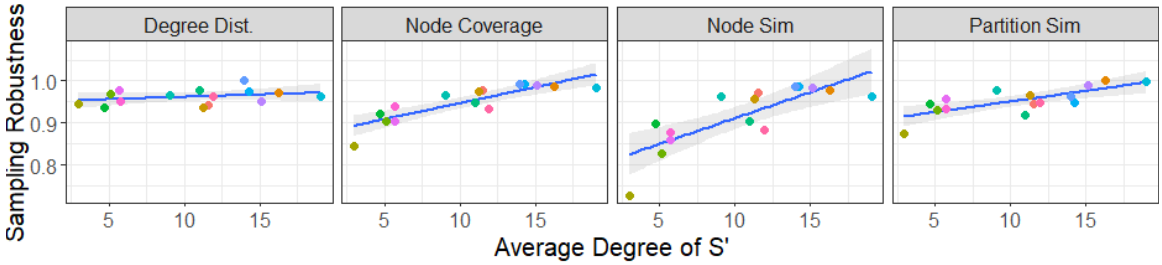
(c) Sampling robustness vs. average clustering coefficient of an original network.

Figure A.1: Aggregate results of $R_{p:0.1\to0.5}$ against three properties of the original network G when **DE-Crawler** is used as a crawler. Each point represents a network. Sampling robustness highly depends on λ_1 and average degree, but not average clustering coefficient.

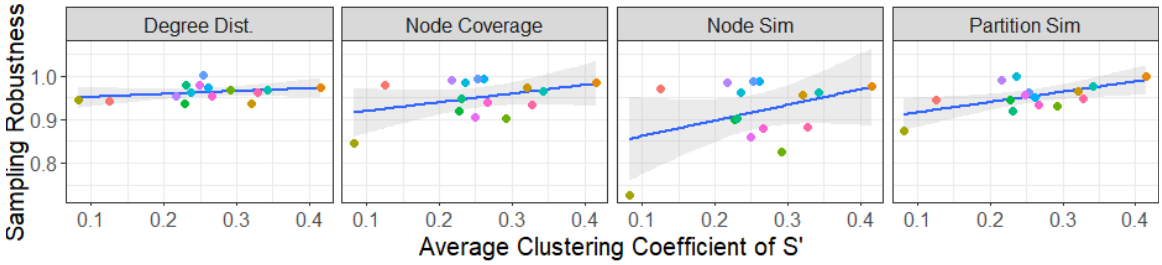
However, when we consider the properties of the obtained sample S' when **DE-Crawler** is used to generate these samples, the sampling robustness is strongly correlated with all these three properties of sample S'. The results are illustrated in Figure A.2. Thus, the networks with higher leading eigenvalue, average degree and clustering coefficient are more robust against missing edges.



(a) Sampling robustness vs. leading Eigenvalue of the sample.



(b) Sampling robustness vs. average degree of the sample.



(c) Sampling robustness vs. average clustering coefficient of the sample.

Figure A.2: Aggregate results of $R_{p:0.1\to0.5}$ against three properties of the obtained sample S' when **DE-Crawler** is used as a crawler. Each point represents a network. Sampling robustness highly depends on all these three properties.

BIBLIOGRAPHY

[1] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive on-line page importance computation," in Proceedings of the 12th International Conference on World Wide Web. ACM, 2003, pp. 280–290.

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

[3] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in Proceedings of the 16th International Conference on World Wide Web. ACM, 2007, pp. 835–844.

[4] I. Albert and R. Albert, "Conserved network motifs allow protein–protein interaction prediction," Bioinformatics, vol. 20, no. 18, pp. 3346–3352, 2004.

[5] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," nature, vol. 406, no. 6794, p. 378, 2000.

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

[7] P. J. Aspinall, "Operationalising the collection of ethnicity data in studies of the sociology of health and illness," Sociology of health & illness, vol. 23, no. 6, pp. 829–862, 2001.

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

[9] ——, "Online myopic network covering," arXiv preprint arXiv:1212.5035, 2012.

[10] R. Baeza-Yates, C. Castillo, M. Marin, and A. Rodriguez, "Crawling a country: better strategies than breadth-first for web page ordering," in Special interest tracks and posters of the 14th international conference on World Wide Web. ACM, 2005, pp. 864–872.

[11] N. Blenn, C. Doerr, B. Van Kester, and P. Van Mieghem, "Crawling and detecting community structure in online social networks using local information," in International Conference on Research in Networking. Springer, 2012, pp. 56–67.

[12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, p. P10008, 2008.

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

[14] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Bandit algorithms for social network queries," in 2013 International Conference on Social Computing. IEEE, 2013, pp. 148–153.

[15] S. A. Catanese, P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Crawling facebook for social network analysis purposes," in Proceedings of the international conference on web intelligence, mining and semantics. ACM, 2011, p. 52.

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

[17] H. Chan and L. Akoglu, "Optimizing network robustness by edge rewiring: a general framework," Data Mining and Knowledge Discovery, vol. 30, no. 5, pp. 1395–1425, 2016.

[18] D. H. Chau, S. Pandit, S. Wang, C. Faloutsos, and C. Faloutsos, "Parallel crawling for online social networks," in Proceedings of the 16th international conference on World Wide Web. ACM, 2007, pp. 1283–1284.

[19] F. Chiericetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós, "On sampling nodes in a network," in Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2016, pp. 471–481.

[20] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: human, bot, or cyborg?" in Proceedings of the 26th annual computer security applications conference. ACM, 2010, pp. 21–30.

[21] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin, "Breakdown of the internet under intentional attack," Physical review letters, vol. 86, no. 16, p. 3682, 2001.

[22] C. Cooper and A. Frieze, "Crawling on web graphs," in Proceedings of the thiry-fourth annual ACM symposium on Theory of computing. ACM, 2002, pp. 419–427.

[23] A. Dasgupta, R. Kumar, and T. Sarlos, "On estimating the average degree," in Proceedings of the 23rd International Conference on World Wide Web. ACM, 2014, pp. 795–806.

[24] W. Ellens and R. E. Kooij, "Graph measures and network robustness," arXiv preprint arXiv:1311.5064, 2013.

[25] G. Ellis and A. Dix, "A taxonomy of clutter reduction for information visualisation," IEEE transactions on visualization and computer graphics, vol. 13, no. 6, pp. 1216–1223, 2007.

[26] E. Estrada, "Network robustness to targeted attacks. the interplay of expansibility and degree distribution," The European Physical Journal B-Condensed Matter and Complex Systems, vol. 52, no. 4, pp. 563–574, 2006.

[27] M. Galley and K. McKeown, "Improving word sense disambiguation in lexical chaining," 2003.

[28] A. C. Gilbert and K. Levchenko, "Compressing network graphs," in Proceedings of the LinkKDD workshop at the 10th ACM Conference on KDD, vol. 124, 2004.

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

[30] ——, "A walk in Facebook: Uniform sampling of users in online social networks," arXiv preprint arXiv:0906.0060, 2009.

[31] ——, "Walking in Facebook: A case study of unbiased sampling of osns," in 2010 Proceedings IEEE Infocom. Ieee, 2010, pp. 1–9.

[32] ——, "Practical recommendations on crawling online social networks," IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1872–1892, 2011.

[33] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in IEEE INFOCOM 2004, vol. 1. IEEE, 2004.

[34] D. Gusfield, "Partition-distance: A problem and class of perfect graphs arising in clustering," Information Processing Letters, vol. 82, no. 3, pp. 159–164, 2002.

[35] A. D. Haghighi, A. Y. Ng, and C. D. Manning, "Robust textual inference via graph matching," in Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2005, pp. 387–394.

[36] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork, "On near-uniform url sampling," Computer Networks, vol. 33, no. 1-6, pp. 295–308, 2000.

[37] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks," Physical review E, vol. 65, no. 5, p. 056109, 2002.

[38] P. Hu and W. C. Lau, "A survey and taxonomy of graph sampling," arXiv preprint arXiv:1308.5865, 2013.

[39] S. Itzkovitz, R. Levitt, N. Kashtan, R. Milo, M. Itzkovitz, and U. Alon, "Coarse-graining and self-dissimilarity of complex networks," Physical Review E, vol. 71, no. 1, p. 016127, 2005.

[40] S. Kalir, J. McClure, K. Pabbaraju, C. Southward, M. Ronen, S. Leibler, M. Surette, and U. Alon, "Ordering genes in a flagella pathway by analysis of expression kinetics from living bacteria," Science, vol. 292, no. 5524, pp. 2080–2083, 2001.

[41] M. Kurant, M. Gjoka, C. T. Butts, and A. Markopoulou, "Walking on a graph with a magnifying glass: stratified sampling via weighted random walks," in Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems. ACM, 2011, pp. 281–292.

[42] M. Kurant, A. Markopoulou, and P. Thiran, "On the bias of bfs," arXiv preprint arXiv:1004.1729, 2010.

[43] ——, "Towards unbiased bfs sampling," IEEE Journal on Selected Areas in Communications, vol. 29, no. 9, pp. 1799–1809, 2011.

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

[45] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," Physical Review E, vol. 78, no. 4, p. 046110, 2008.

[46] J. Lawrence and U. Tar, "The use of grounded theory technique as a practical tool for qualitative data collection and analysis," Electronic Journal of Business Research Methods, vol. 11, no. 1, p. 29, 2013.

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

[48] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2006, pp. 631–636.

[49] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," Internet Mathematics, vol. 6, no. 1, pp. 29–123, 2009.

[50] K. Madhawa and T. Murata, "Exploring partially observed networks with nonparametric bandits," arXiv preprint arXiv:1804.07059, 2018.

[51] A. S. Maiya and T. Y. Berger-Wolf, "Expansion and search in networks," in Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010, pp. 239–248.

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

[53] ——, "Sampling community structure," in Proceedings of the 19th International Conference on World Wide Web. ACM, 2010, pp. 701–710.

[54] ——, "Benefits of bias: Towards better characterization of network sampling," in Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011, pp. 105–113.

[55] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič, "Non-projective dependency parsing using spanning tree algorithms," in Proceedings of the conference on

Human Language Technology and Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2005, pp. 523–530.

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

[57] M. E. Newman, "Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality," Physical Review E, vol. 64, no. 1, p. 016132, 2001.

[58] ——, "Fast algorithm for detecting community structure in networks," Physical Review E, vol. 69, no. 6, p. 066133, 2004.

[59] E. Ochodkova, M. Kudelka, and D. Ivan, "Sampling as a method of comparing real and generated networks," in The Euro-China Conference on Intelligent Data Analysis and Applications. Springer, 2017, pp. 117–127.

[60] M. Papagelis, G. Das, and N. Koudas, "Sampling online social networks," IEEE Transactions on knowledge and data engineering, vol. 25, no. 3, pp. 662–676, 2013.

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

[62] D. Rafiei, "Effectively visualizing large networks through sampling," in VIS 05. IEEE Visualization, 2005. IEEE, 2005, pp. 375–382.

[63] A. Rezvanian and M. R. Meybodi, "Sampling social networks using shortest paths," Physica A: Statistical Mechanics and its Applications, vol. 424, pp. 254–268, 2015.

[64] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multi-dimensional random walks," in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010, pp. 390–403.

[65] ——, "On the estimation accuracy of degree distributions from graph sampling," in 2012 IEEE 51st IEEE Conference on Decision and Control (CDC). IEEE, 2012, pp. 5240–5247.

[66] B. Ribeiro, P. Wang, F. Murai, and D. Towsley, "Sampling directed graphs with random walks," in 2012 Proceedings IEEE INFOCOM. IEEE, 2012, pp. 1692–1700.

[67] D. Robson and H. Regier, "Sample size in petersen mark–recapture experiments," Transactions of the American Fisheries Society, vol. 93, no. 3, pp. 215–226, 1964.

[68] N. Salamanos, E. Voudigari, and E. J. Yannakoudakis, "Deterministic graph exploration for efficient graph sampling," Social Network Analysis and Mining, vol. 7, no. 1, p. 24, 2017.

[69] M. Salehi, H. R. Rabiee, and A. Rajabi, "Sampling from complex networks with high community structures," Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 22, no. 2, p. 023126, 2012.

[70] A. Saroop and A. Karnik, "Crawlers for social networks & structural analysis of twitter," in Internet Multimedia Systems Architecture and Application (IMSAA), 2011 IEEE 5th International Conference on. IEEE, 2011, pp. 1–8.

[71] M. A. Serrano, D. Krioukov, and M. Boguná, "Self-similarity of complex networks and hidden metric spaces," Physical review letters, vol. 100, no. 7, p. 078701, 2008.

[72] C. Song, S. Havlin, and H. A. Makse, "Self-similarity of complex networks," Nature, vol. 433, no. 7024, p. 392, 2005.

[73] J. P. Sterbenz, E. K. Çetinkaya, M. A. Hameed, A. Jabbar, S. Qian, and J. P. Rohrer, "Evaluation of network resilience, survivability, and disruption tolerance: analysis, topology generation, simulation, and experimentation," Telecommunication systems, vol. 52, no. 2, pp. 705–736, 2013.

[74] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," IEEE/ACM Transactions on Networking (TON), vol. 17, no. 2, pp. 377–390, 2009.

[75] T. Wang, Y. Chen, Z. Zhang, P. Sun, B. Deng, and X. Li, "Unbiased sampling in directed social graph," ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 401–402, 2011.

[76] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: An eigenvalue viewpoint," in 22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings. IEEE, 2003, pp. 25–34.

[77] P. Wijegunawardana, V. Ojha, R. Gera, and S. Soundarajan, "Seeing red: Locating people of interest in networks," in International Workshop on Complex Networks. Springer, 2017, pp. 141–150.

[78] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in Proceedings of the 4th ACM European conference on Computer systems. Acm, 2009, pp. 205–218.

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

VITA

NAME OF AUTHOR: Katchaguy Areekijseree

PLACE OF BIRTH: Bangkok, Thailand

DATE OF BIRTH: November 15, 1989

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

King Monkut's University of Technology Thonburi (KMUTT), Thailand

DEGREES AWARDED:

M.Eng., 2014, KMUTT, Thailand

B.Eng., 2012, KMUTT, Thailand

PROFESSIONAL EXPERIENCE:

2015-2019 *Research Assistant, Syracuse University, USA* - Research Assistant under the supervision of Prof. Sucheta Soundarajan. The work is in the area of network mining and graph analysis. The main research topic is network sampling through data crawling.

2012-2014 *Software Engineer, Novitat Co., Ltd., Thailand* - Developed a medical imaging system for visualizing DICOM images (CT/MRI images) which is integrated with PACS in the hospitals.

2012-2014 *Software Engineer, Freelancer, Thailand* - Developed a make-to-order software for local business in Thailand. e.g. customers support tracking system, POS system.

2012 *Researcher (Intern), SIT, Japan* - Developed a module for detecting an irregular heartbeat by using RF-ECG sensor.

2011 *Software Developer/Tester (Intern), Innosoft, Thailand* - Developed a web-based application for student registration service of Kantana Institute.

2011 *Technical Support Engineer (Intern), Smart Tech Solution, Thailand* - The responsibility was to address customer questions and concerns regarding their company's products.

PUBLICATIONS:

1. Katchaguy Areekijseree and Sucheta Soundarajan. "Crawling Complex Networks: An Experimental Evaluation of Data Collection Algorithms and Network Structural Properties" The Journal of Web Science, 2019.
2. Katchaguy Areekijseree and Sucheta Soundarajan. "Measuring the Sampling Robustness of Complex Networks" 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2019.
3. Katchaguy Areekijseree, Yuzhe Tang and Sucheta Soundarajan. "Computing Node Clustering Coefficients Securely" 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2019.

4. 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.

5. Katchaguy Areekijseree, Yuzhe Tang, Ju Chen, Shuang Wang, Arun Iyengar and B. Palanisamy. "Secure and Efficient Multi-Party Directory Publication for Privacy-Preserving Data Sharing." SecureComm 2018.

6. Katchaguy Areekijseree and Sucheta Soundarajan. "DE-Crawler: A Densification-Expansion Algorithm for Online Data Collection" 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2018.

7. Ricky Laishram, Katchaguy Areekijseree, and Sucheta Soundarajan. "Predicted max degree sampling: Sampling in directed networks to maximize node coverage through crawling." Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on. IEEE, 2017.

8. Katchaguy Areekijseree, Ricky Laishram, and Sucheta Soundarajan. Max-Node Sampling: An Expansion-Densification Algorithm for Data Collection. IEEE BigData. 2016.

9. Ricky Laishram, Katchaguy Areekijseree, and Sucheta Soundarajan. Predicted Max Degree Sampling : Sampling in Directed Networks to Maximize Node Coverage through Crawling. IEEE BigData. 2016.

10. Katchaguy Areekijseree, and Tiranee Achalakul. "Volunteered mobile sourcing with multi-objective ant colony optimization." 2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE, 2014.

AWARDS AND HONORS:

2013 Winner of the Thailand Imagine Cup 2013 (Innovation category).

2013 Third place winner of the Imagine Cup Worldwide Final 2013, Russia (Innovation category).

2013 Asia Pacific ICT Awards, Hong kong.

2013 Winner of the National Software Contest 2013, Thailand (Science and Technology).

2013 Winner of the Thailand ICT Awards 2013.

2012 Winner of Student Design Challenge Competition 2012, Singapore.

2012 Winner of the Thailand Imagine Cup 2012 (Software Design).

2012 NSTDA Full Scholarship for Graduate Study