INFORMATION SAMPLING FROM ONLINE SOCIAL NETWORKS

BY

SUHANSANU KUMAR

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Associate Professor Hari Sundaram, Chair
Associate Professor Hanghang Tong
Assistant Professor Sanmi Koyejo
Assistant Professor Meng Jiang

ABSTRACT

Data sampling from online social networks is a pre-requisite step for several downstream applications. Further, the massive size of the online social networks coupled with several API limitations and restrictions to the social information makes sampling a challenging problem. This thesis addresses some of the sampling challenges by proposing novel samplers for sampling attributes (content), hidden attributes (population), and network from online social networks.

Specifically, we first propose an information-based sampler in Chapter 3 for sampling content from online social networks. We leverage the surprise of content to direct our sampler towards the informative content. The surprise-based sampling strategy allows us to sample the cluster shape and boundary of content clusters efficiently, which is crucial for several data-mining tasks, including clustering, classification, regression, and attribute-discovery. We demonstrate our proposed sampler's efficacy on a suite of thirty real-world networks and four data-mining tasks. We further show through empirical counterfactual analysis that network structure does not hinder the performance of surprise-based link-trace sampler in many real-world datasets.

Next in Chapter 4, we propose a novel attributed search based sampler to sample hidden populations. We use a decision-tree-based search strategy to query the attribute-search space systematically. Our proposed decision-tree Thompson sampler follows the exploration and exploitation strategy to sample hidden populations from social networks. We demonstrate our sampler's efficacy over a suite of fourteen sampling tasks on three online social sites and five offline datasets. Furthermore, we show the impact of several factors, like page-size, missing information, and noise, affecting hidden population sampling in real-world social networks.

Finally, in Chapter 5, we propose a novel framework for learning network samplers. First, we show through theoretical and empirical proof that there exists no universal network sampler that can preserve all the topological properties of the underlying graph in the sample. To address the non-existence issue, we propose a reinforcement learning framework that learns high-quality sampling policies according to application needs. We demonstrate the efficacy of our proposed sampling framework through extensive experiments across ten different graph families and seven diverse tasks.

In summary, this thesis develops several sampling strategies for sampling information (attribute, hidden attribute, network) from online social networks while being cognizant

of API restrictions' constraints. We propose adaptive samplers that can cater to different application needs.

To my teachers and parents for teaching me everything I know.

ACKNOWLEDGMENTS

Throughout the process of my Ph.D., I received a great deal of support and assistance.

First and foremost, I would like to express the sincerest gratitude to my advisor, Professor Hari Sundaram, for his constant support and guidance. He taught me to how to read and write academic papers and the art of presentation. He helped me appreciate the project's intricate details and encouraged me to explore the problem and solutions in great depth. While he provided me with the independence of choice of the problems, he guided me through the problem selection and solution search, thereby guiding me to do better research.

I am also grateful to Professor Meng Jiang for mentoring me and teaching me how to ask the right questions in research and where to look for the possible solutions. Through the heterogeneous modeling project, he helped me understand the various aspects of research, including literature review and hypothesis testing. I am also indebted to Professor Sanmi Koyejo for guiding me through the theory of sampling. He encouraged me to think about the broader impact of my research that led to a better understanding of my proposed methods and their limitations. I am also thankful to Professor Hanghang Tong for teaching me the importance of fundamental research and to connect different ideas.

I feel fortunate to be advised by exceptional mentors in my career. A big shout out to Professor Animesh Mukherjee, Professor Tanmoy Chakraborty, and Professor Niloy Ganguly at my undergraduate institution, Indian Institute of Information Technology (IIT) Kharagpur, for igniting the flame of research curiosity in my formative years. During IIT, my initial experiences helped me appreciate the joys and importance of research and motivated me to pursue it as a full-time career.

I would also like to acknowledge my colleagues at Crowd Dynamics Lab for their invaluable suggestions and sympathetic ear. All of my friends outside of work deserve a special mention to help me maintain some semblance of a personal life outside of my research.

Finally, I am forever indebted to my family for their continued love and support.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

As public information becomes *online*, huge volumes of social data are now accessible for research and analysis. Social data is a rich source of information capturing several dimensions of our society, including the demography and political affiliation of the population, the public opinion about different topics, and the organization of our social connections. Unsurprisingly, several disciplines actively use social data for their studies. For example, advertisers [1, 2] employ social networks like Facebook and Twitter to advertise their products to segments of the population that have a higher likelihood of adopting their products. Similarly, social scientists have shifted their focus from labor-intensive offline surveys to online surveys to reach out to "hard-to-reach" populations such as people with mental illnesses [3], jazz musicians [4] and sex workers [5]. Several applications of social data exist: prediction of election results [6], detection of outbreaks of influenza [7, 8], estimation of revenue of released movies [9], disaster management [10], to name a few.

Sampling information from online social networks is a necessity for these social data-driven applications. Downstream applications typically sample social data to make inferences on the sampled data because of the massive size of the online social networks and the limited access to social information through application programming interfaces (API), e.g., Twitter API interface [1] and Facebook API interface [2]. First, online social networks such as Facebook, Twitter, and Weibo are extremely large-scale datasets and growing at an exponential rate. For example, Facebook has over 2.6 billion active monthly users; Twitter has over 353 million active monthly users; Pinterest houses over 416 million active users according to a recent survey [11]. Consequently, it is practically infeasible to sample these social networks' content in its entirety. Furthermore, these networks often exhibit exponential growth in their content, thereby necessitating re-sampling of newly generated content. For instance, Twitter generates over 500 million tweets every day. Secondly, application programming interfaces (API) of online social networks often restrict access to the social dataset for privacy and security reasons. For example, Twitter allows only 60 API calls in an hour, and each query returns at most 100 entities in a single API call. Thus, the massive size of these social networks and the limited access implies that the downstream applications employ samplers that can sample social data for their use in a reasonable time (typically a few days or weeks).

However, sampling information from online social networks is a challenging task. First, uniform (or random) sampling of social information is typically not available on online social networks, including Twitter, Facebook, and Flickr. These social networks do not

[1] `https://developer.twitter.com/en/docs/twitter-api`

[2] `https://developers.facebook.com/docs/graph-api/`

allow random access to social information to protect their propriety dataset. Consequently, several classical statistical samplers that depend on random sampling, such as stratified sampling, cluster sampling, and systematic sampling, are not feasible for sampling from online social networks. Alternatively, we employ social network APIs such as search API and graph API to access the information from these large-scale social networks using local exploration. For instance, search APIs like Twitter search API and GitHub search API allow us to sample the social content using a combination of attributes like keywords, location, and time. Similarly, graph APIs like Twitter friend API and Facebook graph API allow us to traverse the network using follower and friendship links. Second, different social networks often exhibit different social structures and organizations. For example, Mislove et al. [12] showed that social networks like Orkut, Youtube, and Flickr exhibit remarkably different organizations than web networks. Therefore, it is not trivial to develop a sampler that can be universally applied across different social networks. Furthermore, different applications typically focus on different aspects of the social data, such as the user-generated content, the user population, or the network structure of social relationships. E.g., Twitter comprises tweets (content), accounts (users), and follower-followee relationships among users (network). In this thesis, we seek samplers that can cater to diverse research needs while handling the API limitations and restrictions.

In order to understand the sampling problem more concretely, we represent the online social network as an attributed network as shown in Figure 1.1. An attributed network comprises the users represented as nodes and the relationships (e.g., friendship, following) between users represented as links or edges of the network. Further, note that the users have attributes such as their designation, affiliation, place of residence, conference preference, etc. An attributed network can be further decomposed into three separate components: observable attributes, hidden attributes, and network as shown in Figure 1.2. As described before, different applications may be interested in sampling different components of the attributed network. For example, on Twitter, an advertiser may be interested in understanding user interest in her products. Similarly, a health-care specialist may be interested in mining the social behavior of hidden populations (having the hidden attribute) like people diabetes, depression, and HIV. Finally, sociologists may be interested in just the network structure component of the attributed network.

This thesis's goals are then necessarily much more modest and limited in scope–we shall design samplers for sampling the three components of the social networks, namely observable attributes (or content), hidden attributes, and the network. Our samplers are therefore designed to solve three sampling sub-problems corresponding to each of three components of the attributed network: *attribute sampling problem*, *hidden attribute or population sam-*

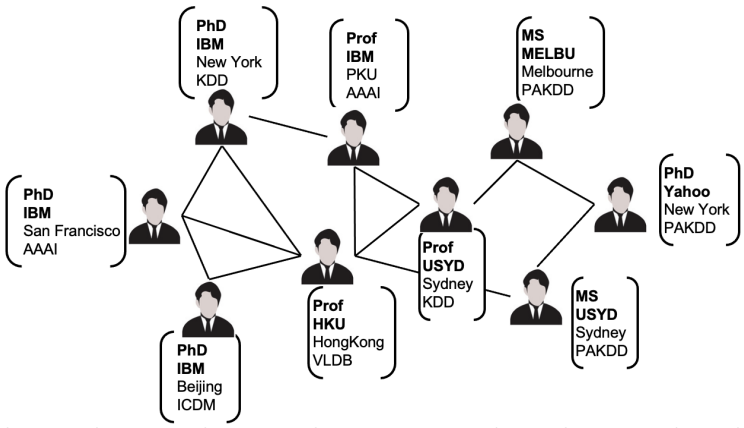


Figure 1.1: Figure shows the social networks as an attributed network. The attributed network comprises of users as nodes, relationship between users as edges and the user attributes like user's job affiliation, designation and preferences.

pling problem, and *network sampling problem*. These three sub-problems are by no means comprehensive of all aspects of the social data. However, they pave a viable path to develop better information sampling methodologies in the future. Since the field of sampling is vast, there are numerous unsolved challenges for each dimension and every specific social network.

The rest of the chapter is organized as follows: In the next section, we shall discuss the three sampling sub-problems in detail while outlining the prior work for each of the three problems. Thereafter in Section 1.2, we present novel samplers for each of the three sampling problems. Finally, we provide the outline of this thesis in Section 1.3.

1.1 SAMPLING PROBLEMS

We shall now describe the three sampling problems and provide a landscape of prior work relating to each problem.

1.1.1 Attribute Sampling Problem

Social attribute (or content) is one of the crucial aspects of social data. Social networks like Twitter, Facebook, and Pinterest comprise rich user-content in the form of attributes such as gender, location, and political affiliation. Note that when we refer to "attributes", we specifically refer to content attributes such as gender, location, political affiliation, etc., distinct from attributes derived from network structure (e.g., node degree, clustering coefficient).

Content sampling is of special significance to social scientists and researchers who may wish to perform data-mining algorithms on social content. We now list few examples from

the several applications involving the social content,

- *Attribute discovery task*: What are the different types of "jobs" in the US on the LinkedIn network?
- *Clustering task*: How many different clusters of "Chicago Cub fans" exist in the Twitter network?
- *Classification task*: what attributes differentiates a "Beattles' fan" from a "Pink Floyd's fan" in the Facebook network?
- *Regression task*: What are the most prominent "attributes" that indicate "a user's age" in the Pokec network?

There are two popular approaches to sampling content in literature. One, the classical statistical samplers such as uniform sampling, stratified sampling, and cluster sampling require random access to the content. Given that the social content is typically accessible through graph API, such statistical samplers are not feasible. The second approach is to sample content by crawling through social network links (edges such as friendship and retweet) using graph API or link-trace sampling [13]. However, link-trace samplers have historically focused on preserving the network structure and largely ignored content [14]. Prior samplers have explicitly tried to preserve the network topology and made an implicit assumption that topology and content co-vary; thus, preserving topology should also lead to the preservation of content. From various social network studies [15], we know that this co-variation should occur. However, there are enough instances where it does not [16]. Thus, we seek content samplers that can sample social content irrespective of the network-content correlation.

1.1.2 Hidden Attribute Sampling Problem

Entities such as people and businesses constitute the social network population. For example, the Facebook network comprises over 2.7 billion user entities and over 100 million advertised business entities as of 2020. Studying user (entity) behavior of the social networks is of special interest to several research studies. For example, social scientists often query social networks to find target populations such as people with mental illness [3] and jazz musicians [4]. These users or entities of interest are often inaccessible to the researchers through direct query search. More generally, the researchers' goal is to find people that satisfy a certain property, but *crucially*, the property itself can not be directly queried. That

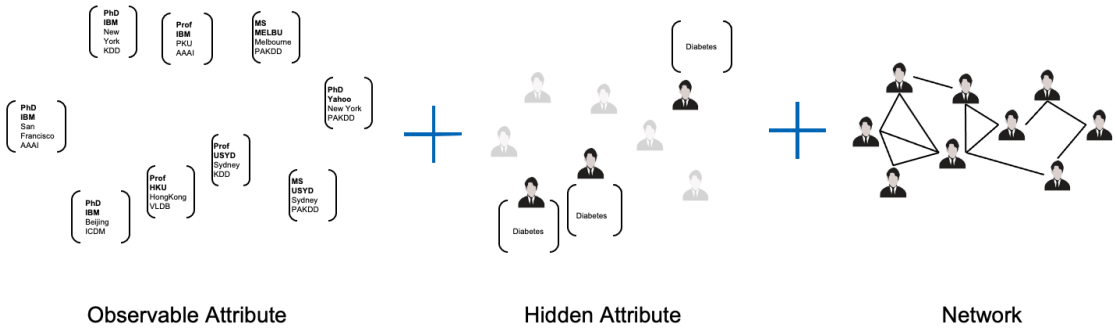


Figure 1.2: Figure shows that an attributed network comprises of three distinct components: observable attributes like user affiliation and preferences, hidden attributes like diabetes and mental health, and network structure.

is, for example, one cannot use the phrase "mental illness" to identity people on Twitter potentially suffering from depression because they rarely use that phrase in any of their tweets to self-describe. However, experts who examine content posted by such populations can more readily do so.

There are several potential strategies to sample hidden populations from an online social network. One strategy is to exploit the graph structure as in Respondent Driven Sampling [17] or web-crawling [18, 19]. At a high-level, the key limitation of a graph-based navigation strategy is that the local graph structure can limit our efforts to traverse the entire graph. One could also view the problem as reconstructing the underlying entity database [20, 21] of the social network. Unlike [20, 21], the population sampling problem is much more restricted—we aim to obtain only a subset of the database. Query reformulation [22, 23] is another promising approach. Query reformulation systems typically use query log data to rewrite a query to maximize the number of relevant documents returned, where relevance is typically computed using the similarity of the query to the document. However, hidden properties are not directly accessible from the document text, making query reformulation challenging. Thus, we seek a hidden population sampler that can exploit social network APIs like search API to query entities using content (entity attributes) directly, and hence sample entities present anywhere on the graph.

1.1.3 Network Sampling Problem

One of the most studied aspects of online social networks is their network (or graph) structure, e.g., the friendship graph's structural organization and communication patterns in the communication graph. Social networks comprise several types of relationships between its users, such as friendship relation in the Facebook network, follower-followee relation in the Twitter network, and subscription relation in the Youtube network. Understanding and

estimating the topological structure of such networks is of crucial value for not just social science but also several other disciplines, including physics and computer science. For instance, researchers study protein interaction networks in biology [24], species interaction networks in ecology [25], and individual relationship networks in sociology [17]. These representative sampled social graphs find use in a wide range of tasks, including the study of cascading patterns of communication [26], the influence propagation of individuals [27] and structure of communities [28]. Furthermore, not only do the graphs from which to sample vary in topology, but also in access cost—for example, some graphs like Twitter and Facebook disallow random access to nodes and edges, or social network APIs may set rate limits.

In the light of a wide range of tasks and graphs topologies, we ask two fundamental questions:

- *Universal sampler*: Can we design a universal graph sampler that one can apply in these different scenarios?
- *Adaptive sampler*: If one cannot design a universal sampler, how can we learn application-suited samplers for sampling graphs for different applications?

Two broad strategies are in use to address the problem of representative graph sampling. The first strategy is to sample graphs to support a *specific* task. Examples include sampling subgraphs from a specific topic [29] or a specific target population using a biased sampler. E.g., focused crawlers [18] sample target web-sites from web-graphs, and respondent-driven sampling [17] samples hidden populations. For tasks that require preserving node label distributions, we could use MHRW [30], or we can de-bias snowball samples [31]. A major drawback of this strategy is that the handcrafting of task-specific samplers requires considerable expert time and effort, making this option difficult to scale through numerous tasks and graph topologies.

The second strategy is to design a *universal* graph sampler that preserves in the sampled graph all properties of the original graph. The motivation is that if one could preserve the original graph's properties in the sample, we could decouple the graph sampling technique from the downstream data-analysis task. Thus the sampler would be *universal*. For example, Forest Fire sampler [32] preserves well, degree, clustering coefficient, and hop distributions in citation and internet networks, Expansion sampler [13] is efficient at discovering community structures, and Rank Degree sampler [33] is effective for sampling centrality measures in social networks.

1.2 THESIS CONTRIBUTIONS

In the previous section, we outlined three prominent social data sampling sub-problems: attributed sampling problem, hidden population sampling problem, and network sampling problem. In this dissertation, we propose a few foundational works for each of the sampling sub-problems. We now summarize the samplers for each sub-problem: Surprise-based Information sampler for content sampling, Decision-Tree Thompson sampler for hidden population sampling, and Graph-family and Task-specific sampler for network (or graph) sampling.

1.2.1 Surprise-Based Information Sampler for Content Sampling

For content sampling, we propose a novel information theory-based surprise sampler that samples social content for data-mining tasks. The key insight is our sampler picks points around the boundary of content class that are more informative for determining the class boundary. Since the identification of class boundary is critical for data-mining tasks such as clustering and classification, we show that our sampler is suitable for sampling content for such applications.

We now enumerate the list of our contributions,

- We propose a task-independent, attribute-aware *link-trace* sampler grounded in Information Theory. The proposed sampler greedily adds to the sample the node with the most informative (i.e., surprising) neighborhood. The sampler tends to rapidly explore the attribute space, maximally reducing the surprise of unseen nodes.
- We prove that content sampling is an NP-hard problem. Further, we show that our proposed definition of familiarity F (which is the converse of surprise) is monotone and sub-modular and that the sampling problem is a constrained maximization of F.
- We show through empirical counterfactual analysis that network structure does not hinder the performance of surprise-based link-trace samplers in many real-world datasets.

1.2.2 Decision-Tree Thompson Sampler for Hidden Population Sampling

For hidden population sampling, we propose a decision-tree Thompson sampler that maximizes the sampling of hidden population entities. The key insight for hidden population sampling is to identify high-quality queries and issue them multiple times. First, we address the problem of combinatorial attributed search space by hierarchically organizing the query space in the form of a tree. Subsequently, we use a *decision-tree* based search strategy

to systematically explore the query space by expanding along high yielding decision-tree branches. Second, we address the problem of black-box API by using the returned set of results to estimate the quality of not just the issued query but also related queries sharing one or more attribute combinations. We employ a reward function to estimate the unique un-sampled hidden entities that can be obtained by issuing a query. Our unified reward function takes into account the stochastic feedback and re-sampling effect while allowing for an exploration-exploitation among queries. We use reinforcement learning-based *Thompson sampling* to define the reward function.

We now enumerate the list of our contributions,

- We address the problem of hidden population sampling problem in online social platforms using attributed search for the first time.
- Our proposed sampling strategy applies to diverse web-forms having a variable number and type of attributes with varying attribute cardinalities.
- We perform a comprehensive set of experiments over a suite of twelve sampling tasks on three online web-query platforms: Twitter, RateMDs, and GitHub, and three offline entity datasets: Patent, Adult, and Auto. Our proposed sampler outperforms all baseline samplers.
- Through an extensive ablation study, we identify the different sampling factors that impact the hidden population sampling. We find page-size, attribute cardinality, the number of queryable attributes, and the correlation between queryable attributes and the hidden property are the prominent factors that affect sampling.

1.2.3 Imitation Learning-Based Learner for Learning Graph Samplers

For network (graph) sampling, we first prove that there is no "universal graph sampler". Subsequently, we propose a reinforcement learning framework that learns different sampling policies for different application contexts. We use two key insights to address the challenges involved with learning an optimal sampling policy. First, we transform the graph sampling problem into a sequence prediction problem (of nodes). This transformation allows us to search for the optimal sampling policy in a continuous policy space rather than a discrete policy space. Second, we exploit the feedback from the user-defined objective function to efficiently explore and identify high-quality policies from the exponentially-large policy space.

We now enumerate the list of our contributions,

- We show through a simple theoretical proof and an exhaustive empirical survey that there exists no universal graph sampler that works independently of context. We prove by contradiction that a sampler cannot simultaneously preserve two different properties (breadth and depth) of a stylized tree graph.

- To the best of our knowledge, we are the first to propose learning of context-aware samplers. In contrast, we propose a reinforcement learning algorithm that learns a high-quality sampling policy for any user-provided context.

- Through extensive experiments across ten different graph families and seven diverse real-world tasks, we show the robustness of our proposed sampling framework, which outperforms the state-of-the-art samplers by a margin of 10% to 318%.

1.3 THESIS OUTLINE

The rest of the thesis is organized as follows. In the next chapter, we shall discuss the prior work in sampling as well as review the areas related to sampling. In Chapter 3, we describe our proposed surprised-based sampler for content sampling. In Chapter 4, we describe the decision-tree Thompson sampler for hidden population sampling. In Chapter 5, we prove the non-existence of a universal graph sampler followed by a proposed universal learning framework that learns adaptive samplers for different application context. Finally, we present the thesis conclusion along-with a list of open problems in sampling in Chapter 6.

CHAPTER 2: LITERATURE REVIEW

Before we delve into the details of our proposed sampling work in this thesis, we discuss the rich prior work of information sampling from online social networks. Unsurprisingly, information sampling arises in several diverse areas. In this chapter, we provide a broad overview of the different sampling methodologies and their applications in the literature. First, we discuss the prior work relating to a social network's three components: observable attribute, hidden attribute, and network, as described in Chapter 1. Finally, we discuss various tools, including machine learning and reinforcement learning algorithms, to tackle different social sampling problems.

2.1 CONTENT SAMPLING

Content sampling is a well-studied problem in classical statistics. However, the social network APIs restrict random and full access to the data necessitating researchers to develop network locality-constrained sampling methodologies. We discuss both the classical and link-trace sampling approaches in this section.

2.1.1 Classic Statistical Sampling

Many sampling algorithms exist in the classical statistics literature, including uniform sampling, stratified sampling, and cluster sampling [34, 35] that samples content randomly from a content repository. However, few social networks allow random access to the content. An exception, Twitter's streaming API provides access to a curated subset of the content. However, several works have pointed out the bias and lack of representativeness of black box APIs such as streaming API [36, 37] for sampling content. These social network API restrictions and black-box API access often restrict the application of traditional statistical algorithms in practice. In this thesis, we address the shortcomings of the traditional sampling methods by adapting these sampling methodologies like information-based sampling for sampling from graph API.

2.1.2 Network-Constrained Content Sampling

Over the last decade, social scientists have designed and developed several sampling techniques to sample from attributed networks. Some of the samplers are cognizant of both the

content (attribute) and the network structure of the online social networks. For example, Li et al. [38] studied five different sampling strategies for node-type and link-type distribution preservation. Yang et al. [39] proposed a semantic sampling strategy, Relational Profile sampling, that preserves the semantic relationship types in a heterogeneous network. Park et al. [40] remarked about the inefficiency of the existing network samplers in estimating node attributes. Wagner et al. [16] showed the sensitivity of existing samplers while sampling attributes from attributed networks. In contrast to the existing works that are limited to specific objectives such as attribute distribution, node-type preservation, frequency estimation [41, 42], we propose a novel content sampler in this thesis that samples content useful for several data-mining tasks such as attribute discovery, clustering, classification, and regression.

2.2 HIDDEN POPULATION SAMPLING

Researchers from several diverse disciplines are interested in mining hidden populations from social networks. Further, many social networks, including Twitter and GitHub, allow for searching user populations via attributed query, in addition to a text query. For example, we can also specify time and location attributes on Twitter in addition to text. In contrast to attributed search, text search has received considerable attention in the IR community. In this section, we discuss various approaches to sample hidden populations via attributed search in Database research, via text search (or query reformulation) in Information Retrieval research, and via link crawling in Network Science research.

2.2.1 Database Sampling

Database sampling or hidden web crawling is an active area of research in the Database field that tries to gather the entire population or database entities by efficiently querying or crawling via the database's attributed search interface. Raghavan et al. [20] first proposed a task-specific hidden web crawler called Hidden Web Exposer that crawled the hidden web forms by maintaining the Label Value Set table used for filling out the forms. Wu et al. [43] proposed attribute value graph traversal based heuristics to crawl the hidden databases. Several other works, such as [44] tries to sample hidden databases entirely in the fewest number of queries. Sheng et al. [21] showed optimal algorithms for crawling the entire hidden web database from form-based interfaces. Given the voluminous size of the online social networks, these algorithms are often practically infeasible to obtain the entire user population. In contrast to the previous studies in database sampling that aim to discover

the *entire* hidden database, we design a hidden population sampler that effectively samples the *target* hidden population.

2.2.2 Keyword-Based Sampling

Query reformulation is another line of research that tries to identify queries having higher recall of target documents in text retrieval systems. Existing retrieval systems in literature are predominantly designed to search for new queries that yield higher reward [22]. Query reformulation systems [23] typically rewrites a query to find a new query that maximizes the number of the relevant document returned. However, users' hidden properties in online social networks are not directly accessible from the document text, making query reformulation challenging. In contrast to the query retrieval systems that retrieve documents usually expressed by rich textual information, we focus on the entity retrieval problem where discovered entities provide very limited information in the form of few attributes. The query interface is limited to queryable attributes of the entities.

2.2.3 Web Crawlers

Focused crawling is a widely used sampling technique used for web crawling where a web crawler tries to maximize the coverage of a given target topic such as "semiconductor-related web-pages" by traversing web-links. Chakraborti et al. [18] proposed a focused crawler that iteratively explores web-links that are more likely to fetch topic web-pages. Similar works on crawling are focused on exploiting the information of web-page such as its content link structure, URL, and metadata [45, 46] to efficiently crawl target pages. Note that focused crawling and respondent-driven sampling [17] rely on graph interface to sample user populations. One of the key drawbacks of the web-crawlers is that they are able to sample only populations that are well-connected to the general population, thus making them unsuitable for sampling "hard-to-reach" hidden populations such as HIV patients, prostitutes, and jazz musicians [4] that are often not strongly connected to the general population. In contrast to the focused crawling that uses a graph-based interface, we develop samplers in this thesis that use a form-based interface to iteratively query for the hidden population.

2.3 GRAPH SAMPLING

Graph sampling is a well-studied problem. As the size of social networks grew, it became increasingly intractable and non-practical to study the sociological networks at scale. The seminal works by Granovetter [47] and Frank [48] were the first works that tried to sample from networks for specific properties such as the mean degree estimation and density estimation of the graph. They used snowball crawling or breadth-first search to estimate these properties. Spurred by the sociological research, the network sampling grew to more complex objectives such as sampling hidden populations such as injection drug users, the homeless, and artists from large social networks.

With the advent of online social networks, large sociological networks became prevalent and easily accessible. Researchers shifted their focus to preserving more complex properties of the graph, such as the centrality of the nodes and the distributional characteristics of networks. For example, Costebander [49] studied the effect of sampling on a suite of eleven node centrality measures, and Lee et al.[50] studied sampling effect on topological properties such as degree distribution and clustering coefficient of graphs. Further development of network sampling led to three prominent lines of research of network sampling: *unbiased node sampling*, *representative network sampling*, and samplers designed for estimating specific *graph properties*.

2.3.1 Unbiased Node Sampling

Unbiased sampling aims to obtain an unbiased sample of network nodes prominently to estimate node-based properties like node degree and nodal values like demography distributions. In the beginning, research focused on understanding the biases of existing samplers like snowball sampling and ways to obtain *uniform samples*. Kurant et al. [51] quantified the degree bias for several network samplers such as breadth-first sampler and proposed new ways to correct them. Gjoka et al. [30] implemented the unbiased link-trace samplers on the very massive Facebook network to estimate several node properties. Chiericetti et al. [52] proposed an efficient random walk sampling strategy for sampling according to a prescribed distribution. Recent works [53] in unbiased sampling show that combining various estimators can be adaptively modified to yield better estimates of the network properties in different networks.

2.3.2 Representative Network Sampling

Unlike unbiased sampling, representative or "universal" network sampling aims to construct a sampled subgraph that has a *network structure* very similar to that of the original network. Forest Fire [32] preserved several key network structure characteristics such as degree and clustering coefficient distributions. Hubler et al. [54] showed via the Metropolis algorithm that prior knowledge of the network can help obtain better representative samples. Arun et al. [13] proposed expansion-based sampling that was very efficient at preserving the network properties such as community structures. Subsequently, Voudiagari et al. [33] proposed a rank degree sampler that preserves the degree and clustering coefficient distribution and several centrality measures. However, one of the key limitations of the above representative network sampling is that they are limited to specific graph domains such as sociological and technological networks. Krishnamurthy et al. [55] showed that the size of a network could be reduced by 70% by either deletion, contraction, or exploration methods while preserving important network properties such as power-law degree distribution, hop-plot, and spectral characteristics. Lee et al. [50] showed analytically the statistical properties such as degree and betweenness centrality distribution, average path length, assortativity, and clustering coefficient of the sampled networks in contrast to the original network for different random sampling strategies. We show that universal graph sampling is impossible for sampling from diverse graph domains and diverse user needs.

2.3.3 Property Estimation

Another line of network research focuses on the estimation of specific graph properties such as the network size (number of nodes in the network), mean degree or density of the network, and certain population statistics of the network. Several experiments have studied evaluating small portions of the network to approximate global properties by using random walks. [56, 57, 58]. In order to approximate a variety of graph-theoretical properties, including the average clustering coefficient, degree distribution, degree correlation, and network size, Hardiman et al. [56] employed the return times and properties of nodes visited during random walks. Kurant et al. [59] showed alternative ways of estimating the graph size and density by sampling induced edges of the graph. Bhuiyan et al. develop graph samplers for estimating the frequency of motifs in large graphs [42]. However, property-specific samplers are often limited to few properties and cannot be easily extended to handle new user objectives. In contrast to property-specific samplers, we propose a universal learning framework that allows for the learning of samplers suited for any user-specified properties.

2.4 APPLICATIONS RELATED TO SAMPLING

We now discuss alternative methods of sampling, including graph generation, sparsification, and compression. Similarly, we show that recent advances in reinforcement learning and optimization can be employed to solve some of the long-standing sampling problems.

2.4.1 Graph Synthesis, Sparsification, and Compression

There exists several complementary approaches to graph sampling including: a) graph compression [60, 61] to speed up the graph algorithms; b) graph sparsification [62, 63] to reduce the size of large graphs to manageable size; and c) graph generation models [64, 65, 66] to generate synthetic samples of the original graph. Graph compression algorithms try to obtain a smaller or compressed representation of the network by using different compression techniques, including clique partitioning [60] and virtual node creation [61]. Graph sparsification tries to handle the study of very large graphs by removing edges [67] or vertices [68]. Finally, graph models are widely used to generate a family of random graphs as a substitute for the original graph. Popular models include Erdos Renyi model [69], Watts-Strogatz model [70] and Barabasi-Albert model [71] that capture different properties of the real-world social networks: the density, high clustering coefficient and scale-free degree distributions respectively. However, these methods require complete or random access to the underlying graph or the underlying graph properties. Since complete access to online social networks like Facebook and Twitter is typically not possible for researchers, we restrict ourselves to graph crawlers or graph samplers in this thesis.

2.4.2 Brain Parcellation

The brain is a particularly complex organization of neurons. Neuroscientists increasingly employ neuroimaging tools, including functional magnetic resonance imaging (fMRI) and diffusion MRI (dMRI), to understand the brain networks' complex functional and structural organization. Such studies [72, 73, 74, 75, 76] typically represent the distinct regions of the brain as the 'nodes' of the brain graph, and the connection (e.g., white-matter fiber bundles) or interaction (e.g., correlation in blood-oxygen-level) between the regions as the 'edges' of the brain graph. These brain networks have been shown to very important for several medical tasks such as identifying neuro-psychiatric disorders like Alzheimer's disease [24], depression [77], social anxiety disorder [78] autism spectrum disorder [79], Parkinson's disease [80] and schizophrenia [81]. These brain networks have also been shown to correlate

with individual's phenotype like gender and age [75, 82], behavioral traits [83] and emotional measures [84]. However, given the large size of the brain network and the noise in the measurement of the brain data often leads the researchers to work with smaller brain representation or parcellated brain networks.

Labeling of the brain regions (or brain parcellation) defines the nodes of the brain graph. It is, therefore, a critical step in the formation of brain networks and subsequent analysis and diagnosis. There are two prominent strategies to parcellate the brain. One, the most commonly used strategy is to employ the brain atlas that maps the regions of the brain into a set of ROIs [85, 86, 87]. The representative examples employed to create the atlas often do fit well with the data, and these atlases are mutually inconsistent [88]. Further, atlases limit the scope of some neurological studies that may warrant subdividing the brain into smaller regions with more precise functional roles. Another strategy is to parcellate the brain into a set of non-overlapping regions that show some homogeneity within the fMRI activation data [89, 90, 91], anatomical information [92, 93], such as gyrosulcal anatomical data [94], autoradiography for cyto-architecture [95], anatomical connectivity with diffusion imaging [96]. Despite a plethora of brain parcellation strategies, it is not clear if there exists a representative brain parcellation that can be used across all populations (adults, children, diseased) and for all studies (development, disease, phenotype identification). Although few works [85, 97, 98, 99] have implicated the need for adaptive brain parcellation, this area has received little attention. In this thesis, we extend the adaptive sampling framework to adaptively learn brain parcellations suited for different tasks and populations.

2.4.3 Reinforcement Learning

Even though seemingly distant, reinforcement learning has a high similarity with graph samplers. In particular, we employ reinforcement learning algorithms in this thesis to solve two sampling problems: hidden population sampling in Chapter 4 and adaptive graph sampling in Chapter 5. We now briefly describe some of the popular reinforcement learning algorithms.

Multi-Armed Bandit A multi-armed bandit (MAB) problem is when an agent has a finite number of actions (or arms) and an underlying reward associated with each action. Over a limited number of trials, the agent tries to draw different actions and figure out the expected reward for each action. Multi-armed bandits have been used in a number of diverse applications including clinical trials [100], adaptive routing [101], financial portfolio management [102] and recommendation [103]. Several sampling-related works, including

targeted node samplers [104], web-crawlers [18] and graph explorers [105] have applied the idea of MAB. We consider various MAB algorithms while designing samplers in this thesis.

Markov Decision Process Owing to its generality, Markov Decision Process (MDP) [106] is a widely tool applied across several diverse domains including game theory [107], information theory [108], multi-agent systems [109] and natural language processing [110]. Recently, MDPs are successfully used to solve several graph combinatorial optimization problems [111] like traveling salesperson problem [112], minimum vertex cover problem [113] and resource-constrained scheduling problem [114]. Sampling from graphs and attributed networks bears several similarity traits with the existing Markov decision problems. For example, graph sampling can be reduced to an optimization problem where the search space is constrained by a local exploration of the graph, and actions are local decisions regarding which node to sample. However, learning an optimal graph sampler differs from learning an optimal optimization algorithm due to some unique challenges covered in detail in Chapter 5. First, unlike existing graph optimizers, we only have local access to the frontier nodes, and the access to the entire graph is inaccessible. Second, we are constrained to use local features of frontiers nodes, and as such, cannot use the node embeddings. Third, experiments show that Q-learning [115] and policy gradient [116] are sample in-efficient for sampling large-scale graphs (e.g., 1K nodes and above).

Imitation Learning During the last decade, imitation learning has gained tremendous popularity in reinforcement learning, owning to its simplicity and extensibility. An imitation learner tries to mimic an expert's action for any given task. The imitation learner makes mapping between the observation and expert's action by learning from expert demonstration. Imitation learning is extensively used in navigation problems such as navigating flying vehicles [117] and automated cars [118]. It is also employed in controlling robotic movements [119] and in computer games [120]. One key advantage of imitation learning over MDPs is imitation learning algorithms' ability to perform well under an induced distribution of states [120, 121]. In this thesis, we employ the data-aggregation technique of imitation learning techniques to design efficient graph samplers.

CHAPTER 3: CONTENT SAMPLING

In this chapter, we present our sampling work for sampling the content (or observable attributes) of a social network. When we refer to "attributes", we specifically refer to content attributes such as gender, location, etc., distinct from attributes derived from network structure (e.g., node degree, clustering coefficient). Several downstream applications are interested in analyzing the content; for example, advertisers may be interested in understanding the user preferences of their product [2], the sociologists may be interested in estimating the demography of a population [30], and journalists may be interested in identifying the latest trends in fashion [122].

To address the above interest in content sampling, we introduce a novel task-independent sampler for attributed networks. The content sampling problem is important because while data-mining tasks on network content are common, sampling on internet-scale networks is costly. Link-trace samplers such as Snowball sampling, Forest Fire, Random Walk, and Metropolis-Hastings Random Walk are widely used for sampling from networks. The design of these attribute-agnostic samplers focuses on preserving salient properties of network structure and are not optimized for tasks on node content. In this work [123], we propose a novel attribute-aware *link-trace* sampler, `SI` that is grounded in Information Theory. Our sampler greedily adds to the sample the node with the most informative (i.e., surprising) neighborhood, thus efficiently exploring the attribute space. Experimental results over 18 real-world datasets reveal that our proposed surprise-based sampler `SI` is sample efficient and outperforms the state-of-the-art attribute-agnostic samplers by a wide margin (e.g., 45% performance improvement in clustering tasks).

The rest of the chapter is organized as follows. In the next section, we provide an overview of the content sampling problem. We present the challenges involved with content sampling, the idea of surprise-based sampling to tackle the content sampling issue, and a summary of contributions. Then, in Section 3.2, we describe our proposed `SI` sampler in detail. In Section 3.3, we compare our proposed sampler with several state-of-the-art samplers on a suite of datasets over four different data-mining tasks: attribute discovery, clustering, classification, and regression. In Section 3.4, we present a discussion of issues raised in the chapter. We extend `SI` sampler for handling noise and missing information in Section 3.5. Finally, we conclude this chapter in Section 3.6.

3.1 OVERVIEW

We shall now provide an overview of the content sampling problem and its importance, followed by an intuitive explanation of our contribution to the content sampling problem.

3.1.1 Why content sampling?

Sampling is critical for data analysis from internet-scale graphs (e.g., Facebook has over a billion nodes) since the entire dataset is too large to analyze in its entirety. Social networks (e.g., Twitter) allow access to their network through rate-limited `API` calls (e.g., Twitter allows 60 API requests per hour), implying that creating a large, representative sample to train data mining algorithms takes significant time.

Ideally, we would like a single framework for sampling content that works well across a range of downstream data mining tasks, to avoid re-sampling the original graph for each task. One strategy to ensure task independence for content analysis is to ensure that the underlying attribute distributions in the original graph are well represented in the sample. Sampling the graph uniformly at random works well to provide an unbiased estimate of the attribute value distributions. However, while random access is possible with offline data, online social networks (e.g., Facebook, Pinterest) prevent random access to their network, necessitating researchers to use *link-trace* sampling [13]. In a link-trace sampler, we start with a seed node, and each new node added to the sample has a neighbor in the current sample.

Widely used link-trace samplers are designed to preserve structural properties of the network, *not* content. That is, they are *attribute-agnostic*. Well known methods include snowball sampling, and Expansion Sampling (`XS`) [13], stochastic samplers such as Random Walk (`RW`), Forest Fire [32] (`FF`) and Metropolis-Hastings Random Walk (`MHRW`[1]) [54]. These samplers focus on preserving the structural properties of the network (e.g. diameter, edge densification, degree distribution) in the sample [30, 32, 54].

The questions in the content analysis of social networks are different from that of analysis of graph structure. While we can use samplers such as `RW` and `MHRW` to help answer questions like the average number of friends on Twitter (via degree distribution), the average degree of separation (via effective diameter), we are interested in helping data scientists answer different sets of questions: how many different religions have a presence on a social network (attribute discovery)? Identify co-located fans of various sports teams (a clustering problem). Ask if a social network participant is interested in fashion [124] (a classification problem).

[1]The stationary distribution for `MHRW` is uniform over the network nodes.

How does income vary with age and gender (a regression problem)?

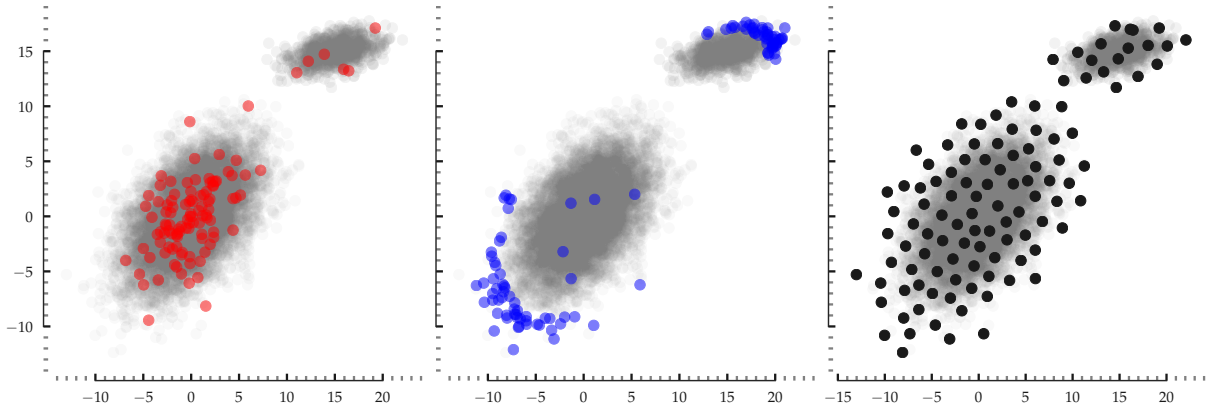


Figure 3.1: We sample two skewed classes (gray) with continuous 2D attributes, distributed over a stylized complete graph. The figure shows the effect of using node sampling performed using `MHRW` (equivalent to uniform sampling) in graphs (left, red), a sampler that focuses on extreme nodes (middle, blue), and a desired sample set of nodes (right, black) as obtained by our proposed link-trace sampler based on surprise (`SI`). Our sampler first captures informative samples at the class boundary and then samples the class interior, whereas the uniform sampler (red) captures samples from the center of the distribution and the extremal sampler samples extrema nodes. Notice that the samples obtained from both the extrema sampler (blue) and the uniform sampler (red) are well separated, but these samples do not cover the ground-truth class boundary, hinting at reduced generalization performance for classifiers trained on these samples.

Data scientists implicitly assume that samplers designed for preserving network properties are "good enough" for sampling network content. However, as [16] point out in recent work, accuracy in tasks (actor position; group visibility) is sensitive to the sampler (they compared edge sampling, random walk, and snowball sampling) used for gathering attributed data.

Designing a single task-independent link-trace sampler for content is hard. While sampling a graph uniformly at random is essential for characterizing the distribution of attribute values, uniform samplers (obtained, for example, through the use of `MHRW`) will pick points around that part of the underlying probability density with the highest concentration of probability mass (see Figure 3.1). From a clustering or a classification standpoint, the points around the boundary of the class are more informative—sampling the center of the density is not helpful for determining the class boundary. Thus, uniform sampling is not suitable for attributed graphs because it *ignores* the arrangement of samples in the underlying feature space, potentially missing out on informative samples useful for tasks like classification or clustering.

3.1.2 Summary of Contributions

Our contributions are as follows:

Surprise-based sampler: We propose a task-independent, attribute-aware *link-trace* sampler (`SI`) grounded in Information Theory. In contrast, well-known prior work on link-trace sampling (e.g. [13, 30, 32, 54]) ignore nodal content because they were explicitly designed to preserve graph structural properties (e.g., degree distribution; diameter) in the sampled graph. The `SI` sampler greedily adds to the sample the node with the most informative (i.e., surprising) neighborhood. The sampler tends to rapidly explore the attribute space, maximally reducing the surprise of unseen nodes.

NP-Hardness: We prove that content sampling is an NP-hard problem. We do this by showing that familiarity F, the converse of surprise, is monotone and sub-modular and that the sampling problem is a constrained maximization of F. A well-known greedy algorithm [125] (denoted as $\texttt{SI}^\star$) best approximates the optimization solution but requires full access to the graph; full access is available for offline data but unavailable for many online social networks. `SI` is equivalent to $\texttt{SI}^\star$, when full access is available, or when the graph is a complete graph.

Counterfactual analysis: *If our proposed link trace sampler (`SI`) can examine only the neighbors of current sample, how does `SI` compare to $\texttt{SI}^\star$ that can access any node?* We show through empirical counterfactual analysis that in many real-world datasets, network structure does not hinder the performance of surprise-based link-trace samplers; they work as well as $\texttt{SI}^\star$.

For standard data mining tasks—clustering and classification—the Information Theoretic sampler (`SI`) strongly outperforms baselines (`ES, RW, XS`). For example, for clustering, there is an average of 45% improvement over `RW` at a sample size of 5%. For classification the average improvement over `RW` is $5 - 10\%$. `SI` is more efficient: for example, in a Patent network, 5.7% of the patents sampled by `SI` achieves the same clustering performance as 10% of the patents collected uniformly `RW` from the dataset. This performance improvement translates to saving over 100K nodes while sampling. `SI` outperforms baselines ($5 - 44\%$ over `RW`) in discovering unique tuples in the data. The performance for `SI` on attribute distribution preservation is surprising: in theory, we expect `RW` to outperform all baselines. In practice, for some datasets `SI` is indistinguishable from `RW`, whereas for some others, `RW` outperforms all baselines as expected.

Significance: Our sampler will impact the work of data scientists who deal with the practical realities of sampling large attributed graphs for their work. Our sampler is simple

to use and more efficient: it requires fewer samples than state-of-the-art baselines to achieve the same clustering and classification accuracy.

3.2 PROPOSED SAMPLER

In this section, we formally define the problem of sampling from attributed networks, followed by a detailed description of our proposed sampler.

3.2.1 Problem Statement

We seek to sample large attributed graphs $G = (V, E)$ in a task-independent manner. As a reminder, our focus is on sampling nodal attributes related to content (e.g., gender), not in network attributes (e.g., clustering coefficient). Nodal attributes include self-reported characteristics (e.g., gender, location) or maybe the result of a classifier (e.g., political affiliation; interests in fashion) operating on the content associated with a person (e.g., tweets).

In this work, we focus on *link-trace* samplers. We define link trace sampling as follows: given an integer z and an initial seed node $v \in V$ which initializes the sample $\mathbb{S}$, a link trace sampler adds nodes v to $\mathbb{S}$ such that there exists a node $w \in \mathbb{S}$ where $(w, v) \in E$. The sampler stops when $|\mathbb{S}| = z$.

The goal of this work is to develop a *task-independent* link-trace sampler that generates graph samples $\mathbb{S}$ from a given static attributed graph $G = (V, E)$ with an aim to support data-mining tasks on node content.

Assumptions: We make two assumptions. First, we assume that we sample static graphs; the assumption works well in practice when the attributes are either immutable (e.g., ethnicity) or slowly varying (e.g., political views). Second, we assume that the cost c_i of acquiring attributes (e.g., from an `API` call; the result of a classifier) for any node i is constant. Thus the total cost C incurred by *any* link-trace sampler will be proportional to z, the desired sample size; that is, $C = O(z)$. Since z is common to all link-trace samplers considered in this work, we ignore attribute collection costs.

3.2.2 Attribute Aware, Surprise-based Samplers

In this subsection, we introduce a specific attribute-aware, surprise-based link-trace sampler, grounded in Information Theory, to sample the graph. Attribute-aware samplers use node attributes (content) to determine the next node v to add to the current sample set $\mathbb{S}$,

by checking the content of this node against the content of the nodes in the current sample. We abbreviate the phrase 'Surprising Information Sampler' as `SI`.

At each step, `SI` adds to $\mathbb{S}$, one optimal node $v \in N(\mathbb{S})$. We assume that for each $v \in N(\mathbb{S})$, we have access to the content of the neighbors of v. We denote δv as the set of neighbors of v, that do not belong to $\mathbb{S}$; we define the set $\Delta v \equiv \delta v \cup v$. We refer to Δv as the candidate set.

In the rest of this section, we show how to sample networks with discrete attributes, followed by sampling networks with continuous attributes. We will use the Pareto-Optimal frontier to identify optimal samples for networks with discrete and continuous attributes.

The Discrete Case: The surprise-based sampler picks a node v to add to $\mathbb{S}$, such that the corresponding candidate set Δv is most surprising.

Balanced sampling (`BAL`) is the simplest surprise-based sampler that adds one node at a time from the frontier $N(\mathbb{S})$, without looking at the neighbors of the node in $N(\mathbb{S})$. That is, in the balanced case, $\Delta v \equiv v$, where $v \in N(\mathbb{S})$. The optimal node v is such that its attributes have the lowest probability of occurrence in the sample $\mathbb{S}$.

In the more general case of the `SI` sampler, $\Delta v \equiv v \cup \delta v$. We define the surprise $I_{\Delta v}$ of a candidate set Δv (conditioned on $\mathbb{S}$, for any single attribute) as follows:

$$I_{\Delta v} = \frac{-\ln P(\Delta v|\mathbb{S})}{|\Delta v|}. \tag{3.1}$$

Where, $P(\Delta v|\mathbb{S})$ is the probability of generation of attributes in Δv given the distribution of attribute values in set $\mathbb{S}$. Assuming that the attribute values of the nodes $v_i \in \Delta v$ are independent (explained in more detail at the end of this section):

$$P(\Delta v|\mathbb{S}) = \prod_{v_i \in \Delta v} P(v_i|\mathbb{S}) \tag{3.2}$$

After algebraic manipulation, we can express Equation (3.1) after combining it with Equation (3.2) as follows:

$$I_{\Delta v} = -\sum_{i=1}^{r} p_{\Delta v}(i) \ln p_{\mathbb{S}}(i) \tag{3.3}$$

where, r is the number of distinct attribute values, $p_{\Delta v}(i)$ is the probability of attribute value i in the candidate set Δv, and $p_{\mathbb{S}}(i)$ is the probability of the attribute value i in the sample set $\mathbb{S}$. Both $p_{\Delta v}(i)$ and $p_{\mathbb{S}}(i)$ are Maximum-Likelihood estimates. Notice that unseen attribute values (i.e. $p_{\mathbb{S}}(i) = 0$) in $\mathbb{S}$ will cause Equation (3.1) to diverge; in general, `SI`

prioritizes discovery of unseen attribute values. Note that `SI` reduces to balanced sampling when $\Delta v \equiv v$.

We can interpret Equation (3.3) as proportional to the distance $d(\boldsymbol{p}_{\Delta v}, \boldsymbol{p}_{\mathbb{S}})$ of the point $\boldsymbol{p}_{\Delta v}$ from the plane $\sum_{i=1}^{r} x_i \ln p_{\mathbb{S}}(i) = 0$. This is because we want to add a node $v \in N(\mathbb{S})$, we compare every candidate set Δv for nodes $v \in N(\mathbb{S})$, against the *same* sample set $\mathbb{S}$. Thus, we pick the optimal node v^* as follows:

$$d\left(\boldsymbol{p}_{\Delta v}, \boldsymbol{p}_{\mathbb{S}}\right) = \frac{-\sum_{i=1}^{r} p_{\Delta v}(i) \ln p_{\mathbb{S}}(i)}{|| \ln \boldsymbol{p}_{\mathbb{S}} ||}, \tag{3.4}$$

$$I_{\Delta v} \propto d(\boldsymbol{p}_{\Delta v}, \boldsymbol{p}_{\mathbb{S}}),$$

$$v^* = \arg\max_{v \in N(\mathbb{S})} d(\boldsymbol{p}_{\Delta v}, \boldsymbol{p}_{\mathbb{S}}). \tag{3.5}$$

That is, we pick v^* such that the candidate set Δv^* is maximally surprising given our current knowledge $p_{\mathbb{S}}$. Where, we use $\boldsymbol{p}_{\Delta v}$ to refer to the distribution of attribute values in the candidate set Δv, and where $|| \ln \boldsymbol{p}_{\mathbb{S}} ||$ in Equation (3.4) is the L_2 norm of the natural log of the distribution of attribute values $\boldsymbol{p}_{\mathbb{S}}$.

To determine surprise when nodes have multiple attributes, we assume that attributes are independent, a simplifying assumption that works well in practice. Thus Equation (3.5) generalizes to:

$$v^* = \arg\max_{v \in N(\mathbb{S})} \sum_{A \in \mathbb{A}} \frac{d(\boldsymbol{p}_{\Delta v}, \boldsymbol{p}_{\mathbb{S}} \mid A)}{|\mathbb{A}|}. \tag{3.6}$$

Where $d(\boldsymbol{p}_{\Delta v}, \boldsymbol{p}_{\mathbb{S}} \mid A)$ is the distance of the set Δv to the sample set $\mathbb{S}$ with respect to attribute A. Equation (3.6) says that the surprise of a neighborhood Δv is the average surprise over all attributes.

The Continuous Case: We compute surprise for continuous attributes, using a Normal kernel density [126, 127] to estimate the continuous probability density $P(\mathbb{S})$. We compute the probability of generating Δv from the sample set $\mathbb{S}$ for multiple continuous attributes as follows:

$$P(\Delta v \mid \mathbb{S}) \propto \prod_{y \in \Delta v} \sum_{x \in \mathbb{S}} \frac{1}{|\mathbb{S}|} \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\Big(-\frac{1}{2} \Delta_{x,y}^T \Sigma^{-1} \Delta_{x,y} \Big), \tag{3.7}$$

where, $\Delta_{x,y} = f_x - f_y$ and where f_x and f_y refer to the vector of continuous feature values corresponding to nodes $x \in \mathbb{S}$ and $y \in \Delta v$ respectively. Equation (3.7) says that the conditional probability of the set Δv given $\mathbb{S}$ for a specific attribute is proportional to product

of the conditional likelihoods of each node $y \in \Delta v$ with each likelihood computed via the kernel density estimate of $P(f_y \mid \mathbb{S})$.

Now using a standard kernel density estimation heuristic $\Sigma = \hat{\Sigma}/|\mathbb{S}|$, where $\hat{\Sigma}$ is the *diagonal* of the sample covariance matrix in $\mathbb{S}$; the heuristic reduces the influence of any single sample point on the estimated density. Then, Equation (3.7) simplifies to:

$$P(\Delta v \mid \mathbb{S}) \propto \prod_{y \in \Delta v} \sum_{x \in \mathbb{S}} \frac{1}{\sqrt{2\pi |\hat{\Sigma}||\mathbb{S}|}} \exp\Big(- \frac{|\mathbb{S}| d^2(x, y; f, \hat{\Sigma})}{2} \Big), \tag{3.8}$$

where $d(x, y; f, \hat{\Sigma})$ is the weighted Euclidean distance measure for feature f, where we weight components of feature f by the sample variance $\hat{\Sigma}$; we only use the diagonal terms of the sample covariance matrix.

For any $y \in \Delta v$, we can compute $d_{\min}$, the minimum of the distances between y and the elements of the set $\mathbb{S}$. We take the negative log on both sides of Equation (3.8) and then divide by $|\Delta v|$, and thus $-\log P(\Delta v \mid \mathbb{S})/\Delta v \propto$:

$$\frac{|\mathbb{S}| d^2_{\min}}{2} + \frac{\log(2\pi |\mathbb{S}||\hat{\Sigma}|)}{2} - \frac{1}{|\Delta v|} \sum_{y \in \Delta v} \log \sum_{x \in \mathbb{S}} \exp \frac{-|\mathbb{S}| \left(d^2_{x,y} - d^2_{\min} \right)}{2}. \tag{3.9}$$

Notice that by definition $d_{x,y} \geq d_{\min}(y, \mathbb{S})$. In general, for large values of $|\mathbb{S}|$ all except one term inside the summation tends to zero, implying that the third term goes to 0. In practice, the third term is negligible when $|\mathbb{S}| \geq 10$. Thus, Equation (3.9) simplifies to:

$$I(\Delta v \mid \mathbb{S}) \propto \frac{|\mathbb{S}| d^2_{\min}}{2} + \frac{\log(2\pi |\mathbb{S}||\hat{\Sigma}|)}{2} \tag{3.10}$$

Equation (3.10) says that the surprise of a set Δv is well approximated by the minimum of the distances of the elements belonging to the set Δv to the set $\mathbb{S}$. Notice that for any sample set $\mathbb{S}$ in Equation (3.10), comparing surprise values across elements $v \in V \setminus \mathbb{S}$ only involves $d_{\min}$. Thus, we define surprise as:

$$I_{\Delta v} = I(\Delta v \mid \mathbb{S}) \equiv \min_{x \in \Delta v, y \in \mathbb{S}} d(x, y), \tag{3.11}$$

$$v^* = \arg\max_{v \in N(\mathbb{S})} \min_{x \in \Delta v, y \in \mathbb{S}} d(x, y), \tag{3.12}$$

where we add node v^* with maximum surprise to the sample $\mathbb{S}$.

We combine the surprise from the discrete and continuous attributes using a Pareto optimal framework. We rank the sets $\Delta v, \forall v \in N(\mathbb{S})$ based on Equation (3.12) and using Equa-

tion (3.6) separately. We identify the set Δv, $v \in N(\mathbb{S})$ on the Pareto-optimal frontier that maximizes surprise from both continuous and discrete attributes and add the corresponding optimal node $v \in N(\mathbb{S})$ to the sample $\mathbb{S}$. The pseudo-code (Algorithm 3.1) summarizes the `SI` algorithm.

Algorithm 3.1 Pseudo-code for generalized `SI` sampler defined for an attributed network having both discrete and continuous attributes

Input: Attributed Graph G, Budget z
Output: Sampled nodes $\mathbb{S}$

$\mathbb{S} = \phi$ ▷ sampled nodes
$\mathbb{F} = \phi$ ▷ frontier nodes
for $k = 1$ to z **do**
  for $v \in F$ **do**
    $\Delta v = \Big(N(v) \setminus \mathbb{S}\Big) \cup v$
    $I_v^{discrete} = I(\Delta v|\mathbb{S})$ ▷ Use Equation (3.1)
    $I_v^{continuous} = I(\Delta v|\mathbb{S})$ ▷ Use Equation (3.8)
    $I_v = (I_v^{discrete},\ I_v^{continuous})$
  $v^* = arg_v \text{Pareto-optimal}(I_v)$ ▷ breaking the ties randomly
  $\mathbb{S} = \mathbb{S} \cup v^*$
  $\mathbb{F} = \mathbb{F} \cup \Big(N(v^*) \setminus \mathbb{S}\Big)$

Attribute Independence: We use a diagonal sample covariance $\hat{\Sigma}$ in Equation (3.8), out of concerns for stability of the covariance matrix. In real-world networks, the attributes of a node will co-vary and are not independent. We will also see co-variation across neighbors due to homophily. The challenge lies in *incrementally* estimating covariance (or equivalently the joint distribution for the discrete case) amongst attributes given the current sample $\mathbb{S}$. If we could estimate covariance effectively, then we could use a variant of the familiar Mahanalobis distance, which incorporates the covariance matrix, to compute the combined distance. In practice, we observe that when we use the full covariance matrix (or the full joint for the discrete case), the estimates of the off-diagonal elements of the covariance matrix do not stabilize unless the sample set $\mathbb{S}$ is large. The effect is to "push" the sampler in the wrong direction when $|\mathbb{S}|$ is small due to poor on-line covariance estimates; the estimates are worse for skewed attribute distributions.

In this section, we discussed attribute-agnostic and attribute-aware sampling schemes. We introduced the idea of surprise, grounded in Information Theory, as a framework to develop sampling schemes. Our surprise based sampler `SI` adds one node $v \in N(\mathbb{S})$ with the most surprising candidate set Δv to $\mathbb{S}$.

3.3 EXPERIMENTS

In this section, we describe eighteen real-world attributed networks used for evaluating our proposed and baseline samplers. We consider three widely-used content-related tasks: cluster discovery task in Section 3.3.2, content coverage task in Section 3.3.3, and classification and regression tasks in Section 3.3.4.

3.3.1 Datasets

Table 3.1: The eighteen real-world networks used in this paper for empirical analysis differ in size, and in key network parameters: degree distribution, diameter and clustering coefficient. N: number of nodes, E: number of edges, DS: # of discrete attributes, CT: # of continuous attributes, d_V: average node degree, CC: clustering coefficient, DIA: diameter

Networks	N ($\times 10^3$)	E ($\times 10^3$)	DS	CT	d_V	CC	DIA
Facebook	4	88.2	3	0	43.69	0.27	8
Enron	36	183	0	7	10.02	0.72	13
Patent ($\times 6$)	147-403	700-1,340	3	3	6.5-10.9	0.05-0.11	13-18
NSF ($\times 5$)	2.2-4.7	4.2-10	5	1	3.64-4.64	0.37-0.45	21-36
Pokec	1,100	14,900	2	0	13.16	0.05	11
Wikipedia ($\times 4$)	0.433-1.6	5-26	7,900-18,600	0	20.2-53.1	0.37-0.68	7-8

We consider an assortment of eighteen real-world network datasets summarized in Table 3.1 from varied domains: Facebook social network; six bibliographic networks from the Patent dataset; Enron communication network; Pokec social network; four information networks from Wikipedia and five co-authorship networks within the five prominent US National Science Foundation (NSF) organizations. We provide a full description of the real-world datasets, including inclusion criteria [2], and algorithms to generate synthetic datasets in the supplementary information section.

Due to a similar performance of samplers across the individual networks in the Patent, Wikipedia, and NSF datasets, we show performance on a representative network: 'Chemical' (Patent); 'Philosophers' (Wikipedia) and 'Computer' (NSF); we provide a complete summary of performances over each network (for NSF, Patent datasets) in the supplementary section.

Having introduced the datasets used for experiments, we next discuss results for cluster discovery.

[2]Besides the above datasets, we consider other publicly available datasets like Twitter and Google+ [28]. However, we couldn't include them in our experiments due to the poor quality of the datasets, i.e., more than 25% of nodes in the graph have missing attributes.

3.3.2 Cluster Discovery

In this subsection, we examine the effects of link-trace sampling on cluster discovery. We first describe the evaluation metrics and clustering algorithms used for discovering content clusters. Next, we present the experimental results.

Given the plethora of distance metrics and algorithms used for defining clusters in content, we use the standard well-known metrics and algorithms for simplicity and illustrative reasons. We use the standard distance measures (Euclidean distance for continuous variables and the Jaccard distance for nominal variables) and use a well-known k-prototype clustering algorithm [128]. This clustering algorithm is a generalization of the k-means and k-modes algorithms, for content with continuous and discrete attributes.

To understand the sampling effect independent of cluster size and quantity, we also vary the number of clusters (k) in our experiments. For each such value of k, we cluster the ground truth data. Then, after we sample the data with `SI` and the other baselines (`XS, FF, ES, RW`), we evaluate the fraction of original clusters that are present in the sample. The fraction of original clusters captured in the sample shows how good a sampler is at discovering the original content clusters. Figure 3.2 shows the results, averaged over 100 runs.

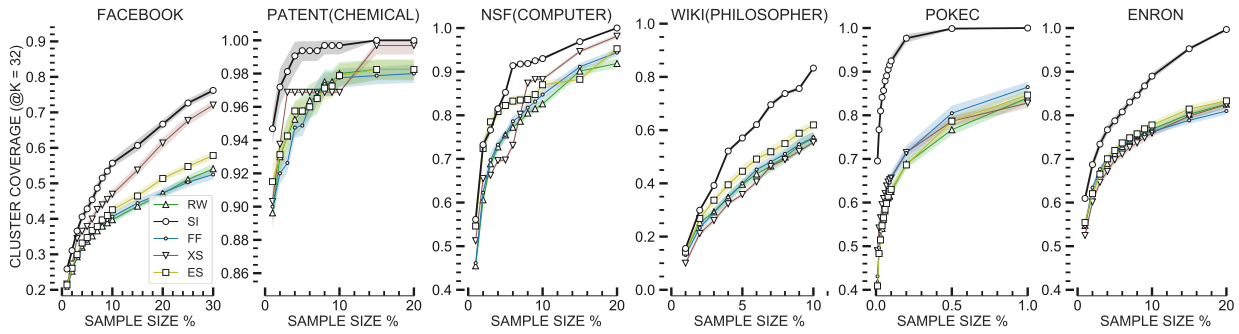


Figure 3.2: Cluster coverage for $k = 32$ clusters, on the Facebook, Patent, NSF, Wikipedia and Enron datasets shows that `SI` outperforms competing samplers (`RW, ES, FF, XS`) at nearly all sampling sizes. Bands indicate 95% CI.

We see from Figure 3.2 results for cluster coverage when the number of clusters k is 32, the surprise based sampler (`SI` outperforms baseline samplers—re-weighted random walk (`RW`), forest fire (`FF`), expansion sampling (`XS`) and edge sampling (`ES`). The main reason is that `SI` sampler rapidly explores the attribute value space, allowing us to cover niche clusters. However, the attribute-agnostic samplers are heavily influenced by the distribution of the attributes over the network and the skewness in the clusters' size. In datasets such as Wikipedia, Pokec, and Enron, clusters show high skew, thus contributing to the relatively high-performance improvement of `SI` over baselines. Among the attribute-agnostic samplers,

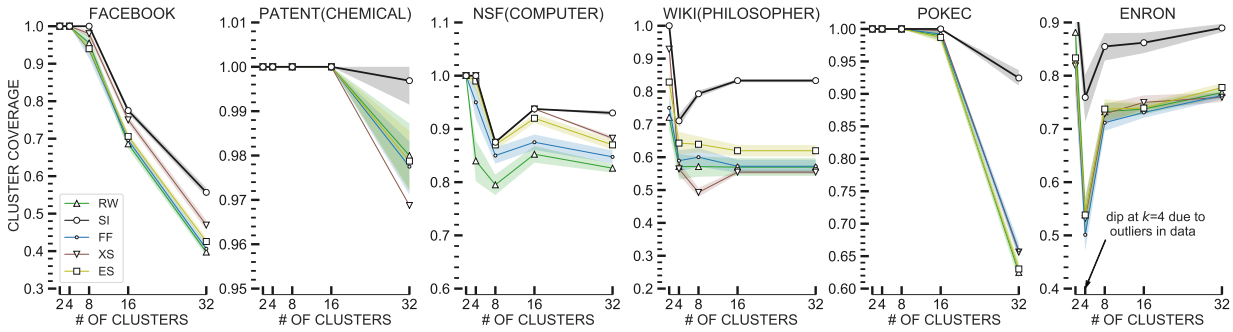


Figure 3.3: Cluster coverage at 5% sample size by varying k, the number of clusters, averaged over 100 runs. `SI` has the best performance over all cluster sizes; the cluster coverage falls with increasing k because the sample size is fixed at 5%. Outliers distort the ground truth clusters and are responsible for the "dip" (NSF; Wikipedia; Enron).

`XS` notably performs well when the attributes are correlated with the community structure in networks such as Patent and NSF. Thus, `XS` which is known for its high community coverage [13] also achieves relatively better cluster coverage. On average, `SI` is moderately better than the second-best baseline samplers by and on Patent (3%) and NSF (6%) respectively, much better on Enron and Facebook (12%) and significantly better on Wikipedia (28%) and Pokec (37%).

Let us examine the weaker performance of `RW`, `FF` and `ES` link-trace samplers on the Facebook dataset. This dataset contains attributes with high assortativity, increasing the chance that nearby nodes share the same attribute value as the current node. Since `XS, ES` and `RW` are *attribute agnostic* samplers that use only the network structure for exploration, high assortativity influences the quality of clusters discovered. Thus these samplers will tend to over-sample similar tuple combinations, missing out on smaller clusters.

Next, we examine how changing the number of clusters k alters the cluster coverage. We examine cluster coverage results at 5% sample size and vary cluster sizes from 2 to 32. Figure 3.3 shows the results. We see that in all cases the surprise-based sampler outperforms baselines at all values of k. The cluster coverage results are close for $k \leq 4$. This result is unsurprising since these datasets are large and since we set the sample size to be 5%, covering $k \leq 4$ clusters is easy for all samplers. As we examine finer clusters, `SI` sampler performance is superior to other baselines. The cluster coverage falls with k since the sample size is fixed at 5%.

Of interest is the odd dip for $k = 4$ in several datasets (e.g. Enron at $k = 4$) for all samplers. We examined the ground truth data and found that the presence of large outliers explains the dip. These outliers distort the ground truth clusters (e.g., Enron at $k = 4$) obtained using the k-mode algorithm; For larger values of k, the k-mode algorithm groups

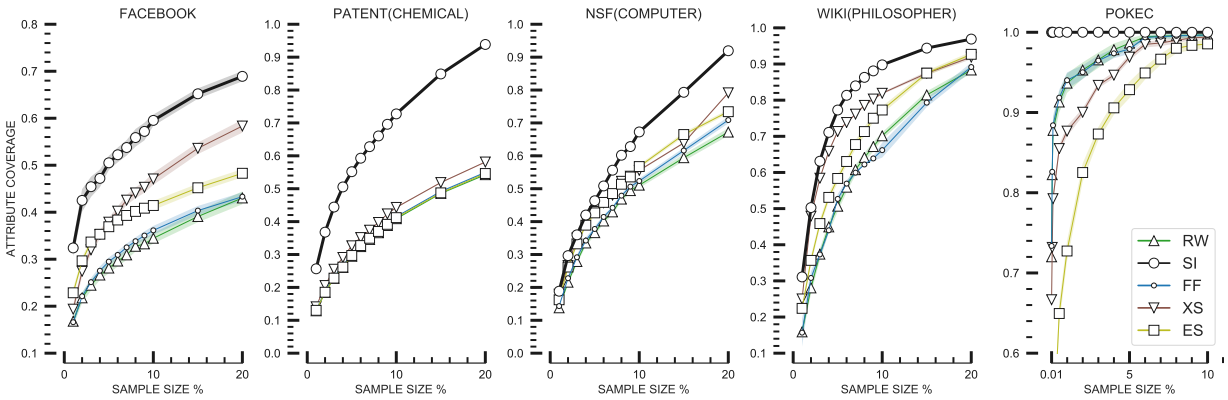


Figure 3.4: Attribute tuple coverage on five real-world datasets—Facebook, Patent, NSF, Wikipedia and Pokec. The figures with 95% confidence interval bands, show that the surprise based sampler (`SI`) outperforming all other samplers.

these outliers into a separate cluster.

In this subsection, we evaluated `SI` against baseline samplers. `SI` outperforms baselines for cluster coverage, notably with increasing dataset skew.

3.3.3 Content Coverage

In this subsection, we evaluate how different samplers perform with respect to attribute tuple coverage and attribute distribution.

Tuple coverage or unique entity discovery is a desirable goal in any empirical data analysis: to ensure that all the unique entities in the data are present in the sample. For example, consider a corporation trying to allocate its resources based on a demographic study is expected to sample from all possible demographics of its customers or unique entities in its user study. To evaluate the unique entity coverage for each sampler, we compute the fraction of unique tuples (for datasets with discrete attributes) present in the sample as the metric. Thus, the Enron dataset, which comprises only continuous attributes, is not considered in this study.

Figure 3.4 shows the results. First, across all datasets, the surprise-based sampler (`SI`) outperforms baselines. Second, none of the samplers reach 100% coverage even for large sample sizes (∼20%) for the Facebook, Patent, and Wikipedia datasets. This is because all three of these datasets have high attribute cardinality, and these attributes exhibit skew. Notice further that the attribute-agnostic samplers fares poorly in comparison to `SI` in some datasets—for example, `SI` outperforms the baseline samplers by 26% in the Facebook dataset and by 14% in the Patent dataset. High attribute skew is the best explanation—since

the baseline samplers are agnostic to attributes, they are more likely to sample attribute values that appear more often, penalizing tuples that contain attribute values that appear infrequently. For the Pokec dataset, all samplers have good attribute value coverage since the attribute cardinalities are lower than the other two datasets. Averaged over all real-world datasets, `SI` outperforms the state-of-the-art samplers by 14%.

Samplers vary in how well they preserve attribute distribution. In Figure 3.5, we compare the $K - S$ statistic, averaged across the different attributes, for each of the six datasets. A small $K - S$ statistic implies that the sampler preserves the underlying distribution well in the sample. Across all the datasets, unsurprisingly, the uniform attribute sampler would have performed the best since `UNI` is an unbiased estimator of the attribute distribution. In general, random walk (`RW`) based samplers are the best link-trace samplers, since asymptotically, the probability of visiting each node in the graph is uniform. The surprise-based sampler (`SI`) gives a mixed performance for capturing the attribute distribution. We don't expect `SI` to perform well since the sampler goal is to maximize familiarity. Interestingly, the differences between `RW` and `SI` are negligible on the Wikipedia dataset. Expansion sampling (`XS`) works well on the Facebook, Enron, and Wikipedia datasets, but not on the Patent dataset. We note that Patent has a significantly lower clustering coefficient (c.f. Table 3.1) than the other networks. Since `XS` rapidly explores the network helped by the presence of clusters, networks with low-clustering coefficients may hinder rapid exploration.

3.3.4 Classification

In this subsection, we present our results for classification and regression. First, we describe the experimental setup used for the experiments, followed by the results.

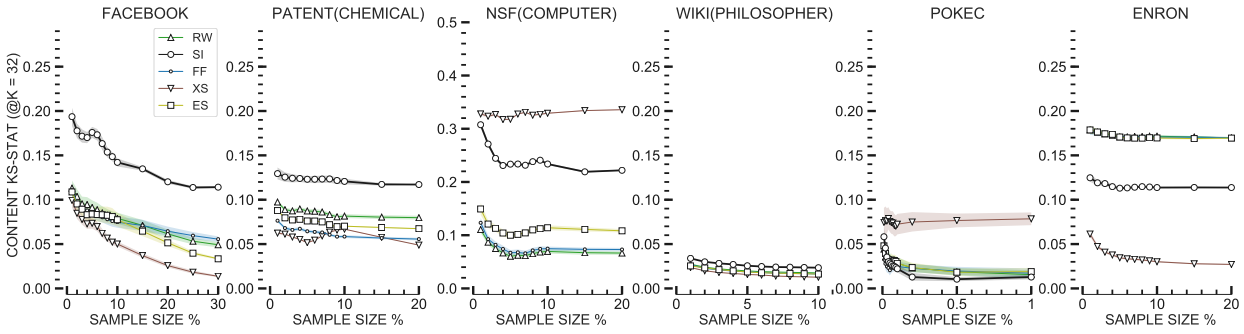


Figure 3.5: Attribute Distribution: The $K-S$ statistic (lower is better), averaged across attributes, for different datasets. Notice that while uniform sampling based link-trace sampler `RW` performs the best, the differences are negligible on the Wikipedia dataset. `SI` is better than `RW` or `ES` for the Enron dataset.

We use different evaluation metrics for predicting discrete and continuous target attributes. We evaluate the prediction of discrete attributes through the weighted F_1 score, and we use Pearson's R^2 coefficient to evaluate the prediction of continuous attributes.

We pick attributes to predict for the classification and regression tasks using the following principles. First, across the three real-world datasets, we pick attributes of varying cardinality to predict target attributes to help us understand the effect of cardinality. Second, as would be natural in any classification task, we pick attributes to predict that co-varied with the features that are used as input to the classifier or regressor. Thus, for the Facebook dataset, we predict "gender" using feature set "locale", "education type". For the Patent dataset, we predict attribute "country" as the discrete attribute and "number of claims" as the continuous attribute on which we regress using the rest of the attributes as features. For the NSF dataset, we predict "duration" of NSF awards awarded to the NSF investigators using investigator attributes such as "number of awards won" and "number of project's PI".

The results in Figure 3.6 reveal that `SI` is a better choice for sampling networks for content than the state-of-the-art link-trace samplers for classification tasks (Facebook and Patent datasets); `SI` performance is indistinguishable from other samplers for the regression task (Patent dataset; number of claims). For the Facebook dataset, `SI` achieves an 18% relative gain with the weighted F_1 score over `RW` variants. For the Patent dataset, we note that `SI` is better than baseline samplers by a margin of over 2% for discrete attribute "country". The overall weighted F_1 performance of almost all samplers is high due to the skewed distribution of the target attribute (i.e. "country" attribute skew = 0.70). We show the prediction of "number of claims" in the Patent dataset as an example of regression task—there is no significant difference among the samplers. For the NSF dataset, `SI` outperforms its competition by over 10% at predicting the "cumulative duration" of NSF awards awarded

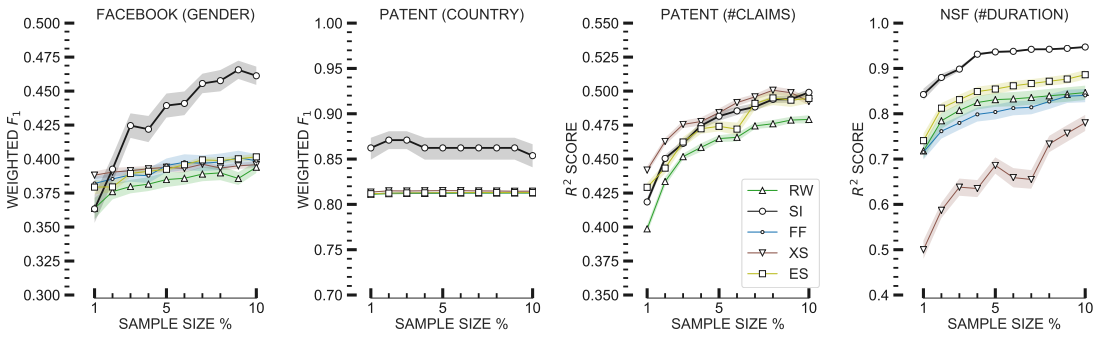


Figure 3.6: Classification performance on different datasets. The `SI` sampler has significantly better classification performance than baseline samplers. There is no significant difference in performance of samplers for regression task in the Patent dataset. Bands show CI = 95%.

to NSF investigators.

We show additional results in in the supplementary section of the paper [123]. We show that the SI sampler consistently outperforms state of the art link-trace samplers such as XS, RW, FF and ES for four content tasks: clustering, classification, regression, and attribute-value discovery.

3.4 DISCUSSION

In this section, we begin in Section 3.4.1 by analyzing the performance of the surprise based sampler (SI) and then in Section 3.4.2, we examine the issue of "network resistance." Finally, we discuss limitations of SI sampler in Section 3.4.3.

3.4.1 Why does SI work well?

The surprise-based sampler SI outperforms baselines for classification and cluster discovery. To understand why, we examine Figures 3.1 and 3.4. Figure 3.1 shows that the surprise based sampler tends to uniformly cover the underlying density, whereas the baseline samplers tend to pick data near the center of the density (where the tuples are more likely). Figure 3.4 compares sampler tuple coverage and shows that the SI sampler, consistent with its bias, tends to pick up less common tuples. Taken together, both Figures 3.1 and 3.4 imply that the SI sample will tend to cover the boundary of the class conditional density first. For the same number of samples, SI sampler will have covered the less common examples of a class, whereas the attribute-agnostic samplers would cover the more common instances. Our work indicates (c.f. Figure 3.1) that link-trace samplers that attempt to sample content uniformly in an effort to mimic UNI (e.g., RW, MHRW) may be weaker for clustering and classification tasks since they ignore the underlying geometry (i.e., the arrangement of samples in the underlying metric space) which is important to identify the most informative samples for these tasks. Thus, we ought to expect that classifiers that use data from the SI sampler to exhibit lower generalization error than the uniform sampler (and samplers like RW, MHRW). A similar explanation holds for cluster coverage results in Figure 3.2.

3.4.2 Examining "Network Resistance"

Submodularity of familiarity $F(v \mid \mathbb{S})$ is a key concept. We remind the reader that since F is submodular and monotone, the sampling problem is an NP-hard optimization problem. The greedy algorithm (SI*), that adds to $\mathbb{S}$ the most surprising node *in the entire graph*,

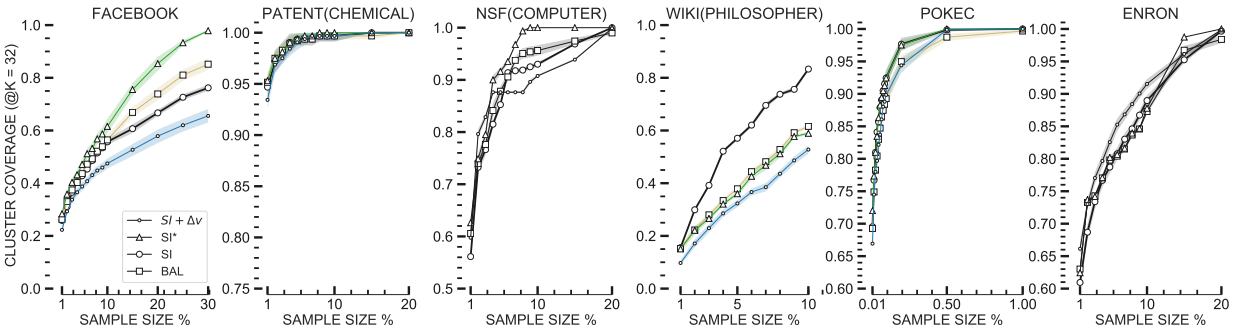


Figure 3.7: Comparing the counterfactual case when all nodes are accessible (i.e., SI*) against the case when the sampler needs to traverse the network (i.e., link-trace). In general, when random access is possible (counterfactual, SI*), the results are better than with link-trace. Interestingly, this is not always true (e.g., Wikipedia). This is because the submodular optimization problem is task-agnostic: it maximizes familiarity but does not optimize for the clustering task.

approximates the optimal $\mathbb{S}^*$ within a factor of $1 - 1/e$ [125]. The SI* algorithm is useful for offline graph datasets where we have random access or when the graph is a complete graph. Since online social networks disallow random access, data scientists use link-trace samplers.

A counterfactual: What if the SI sampler *could* pick any node from $V \setminus \mathbb{S}$ like the SI* algorithm and not be restricted to pick a node from $N(\mathbb{S})$, the neighborhood of the current sample $\mathbb{S}$?

We examine the consequence empirically by comparing the counterfactual case (i.e. SI*) against three baseline link-trace based samplers. The first baseline is SI, the one extensively examined in this paper. The second baseline, we term balanced ('BAL') computes the surprise of nodes $v \in N(\mathbb{S})$, without examining $N(v)$, the network neighborhood of v. The final baseline, which we term $SI + \Delta v$, examines the neighborhood $\Delta v, \forall v \in N(\mathbb{S})$, and *adds the entire neighborhood* Δv to the sample $\mathbb{S}$ corresponding to $v^* \in N(\mathbb{S})$.

Figure 3.7 shows the results. The SI* algorithm is better for Facebook, marginally better for NSF, marginally worse for Enron, and much worse for Wikipedia. Notably, the difference in performance is not statistically significant for Enron, Patent, Pokec, and NSF, implying that link-trace samplers do not suffer a loss of performance for these networks. The disparity for Facebook is more salient, implying that the network structure prevents link-trace samplers from finding the optimal sample set.

Wikipedia is a highly unusual dataset. It has a large number of binary attributes (7,969) compared to just 1,564 nodes. Furthermore, the attributes in Wikipedia are highly asymmetric binary variables (on average, there are only five attributes with val=TRUE, per node). As a consequence, the surprise created by almost every node $v \in N(\mathbb{S})$ in the frontier set is

maximally surprising, implying that the SI* algorithm picks a node at random from $N(\mathbb{S})$. Since SI uses the neighborhood Δv, there are fewer nodes in $N(\mathbb{S})$ that are maximally surprising, leading to better samples for the clustering task. We want to remind the reader that the definition of surprise is task agnostic. Thus, it should not be odd that the SI* algorithm, which approximates the solution to the optimization goal within $1 - 1/e$, may yield samples less suited to the clustering task on some datasets.

3.4.3 Limitations

Now, we discuss four limitations of this work. First, the theoretical analysis assumes that we have no missing values; while SI works in practice when nodes have missing values (one can use the expected value, for example), a noise model to estimate the error in surprise in networks with missing values will be helpful. Second, the time and space complexity of SI is higher than random walk-based samplers such as RW. The incremental update complexity is $O(\mu \log |\mathbb{S}|)$, where μ is the mean degree of the network, while it is $O(1)$ for RW. We can mitigate this issue in two ways. One way is to adopt a "lazy evaluation" for submodular maximization [129] that shows up to 700× speed-up in real-world datasets. Another approach is to use data structures that include information about neighbors in each node, an approach used in decentralized peer-to-peer networks. Third, our model of link-trace sampling is limited: many social networks allow us to make queries on the content and return network nodes that satisfy the query. Extending our sampling framework to incorporate a more rich query model would be interesting. Finally, by design, we examine only the 'content' attributes of a node, not the network properties; investigating samplers that sample for nodal network properties in addition to nodal content would be significant for data mining problems requiring both local network properties and node content.

3.5 EXTENSION: BAYESIAN SURPRISE INFORMATION SAMPLING

In this section, we consider an extension of SI sampler using a Bayesian update method to compute the information surprise. More specifically, we model categorical attributes in the network using categorical distribution and discrete distribution using Poisson distribution. We refer to our sampling algorithm as Bayesian surprise information (BSI) sampling algorithm.

BSI follows the same surprise-based formulation to sample the most informative frontier node as described in Equation (3.5). However, unlike SI, we use prior probability distribution to model the information content in the sampled set $\mathbb{S}$. Subsequently, we add the frontier

node to the sample at each sampling step and update the posterior distribution of the sample's content. We now describe the two surprise score computations using Bayesian modeling of the content: Dirichlet distribution to model categorical content and Gamma distribution to model discrete attributes.

For categorical attributes, we represent the attributes using categorical distribution. For all nodes $v \in V$ and the attribute A_v is a categorical attribute such as gender, education level, and location, the categorical distribution $Cat(p_1, p_2, \ldots, p_K)$ with support $\{1, 2, \ldots, K\}$ (attribute values) is defined as

$$Pr_{x \sim Cat(p1,p2,\ldots pK)}(x = i) = p_i \quad \forall i \in 1, 2, \ldots, K, \ where \sum_{i=1}^{K} p_i = 1 \tag{3.13}$$

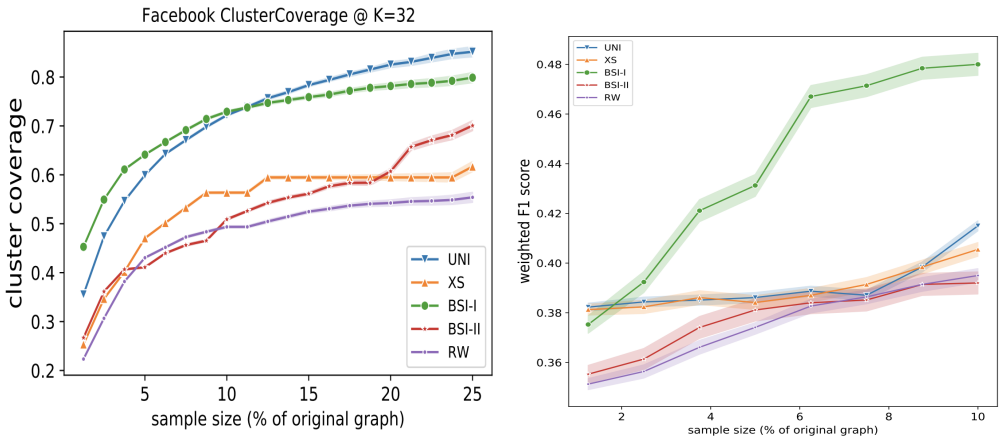
Further, we assume that the probability values $p_1, p_2, \ldots, p_K$ are drawn from Dirichlet distribution $Dir(\alpha_1, \alpha_2, \ldots, \alpha_K)$(prior distribution). Using the fact that Dirichlet distribution is conjugate prior of categorical distribution, we easily update the posterior distribution of the categorical attribute as given below,

$$\begin{aligned} P_t &= Dir(\alpha_1^{(t)}, \alpha_2^{(t)}, \ldots, \alpha_K^{(t)}) \\ where, \ \alpha_i^{(t)} &= \alpha_i^{(t-1)} + \sum_{v \in S(t)} 1_{A_v=1} = i \quad \forall i \in 1, 2, \ldots, K \end{aligned} \tag{3.14}$$

For discrete attributes, we represent the attributes using Poisson distribution. Consider an attribute $A_v \sim Poisson(\lambda)$ is a discrete non-categorical attribute, e.g., age in a social network graph with users as its nodes, number of claims in a patent network with patents as its nodes, etc. We assume support of these attributes to be non-negative integers. Further, we assume that the rate λ of Poisson distribution is drawn from gamma distribution $Gamma(\alpha, \beta)$. Using the fact that gamma distribution is conjugate prior of Poisson distribution; we easily update the posterior distribution of the discrete attribute as given below,

$$\begin{aligned} P_t &= Gamma(\alpha^t, \beta^t) \\ where, \ \alpha_t &= \alpha_{(t-1)} + \sum_{u \in S(t)} A_u \text{ and } \beta_t = \beta_{(t-1)} + |S(t)| \end{aligned} \tag{3.15}$$

Using the above described setup, we model the information content of the sample set $\mathbb{S}$ (i.e., $\boldsymbol{p}_{\mathbb{S}}$) and frontier node (i.e., $\boldsymbol{p}_{\Delta v}$) using the above probability distributions P_t. Finally, we compute the surprise of a frontier node (induced by the frontier node and its neighbors)



(a) Cluster coverage task (b) Classification task

Figure 3.8: Performance of Bayesian surprise samplers and baseline samplers on Facebook dataset over two tasks: cluster coverage and classification.

with respect to the sample content by substituting the KL divergence as distance metric d used in Equation (3.5). For multiple attributes, we compute the KL divergence of each attribute independently and add them to get the overall KL divergence.

Next, we consider two variants of Bayesian information sampler (BSI): a) `BSI-I` adds only one node to be added to the sampled set, i.e. frontier node $v^\star$ at every sampling step t, b) `BSI-II` include the entire neighbor of the most surprising frontier node ($\Delta v^\star$) at every sampling step t. [130] details the algorithm.

We show the performance of the samplers on the representative Facebook network in Figure 3.8 on cluster coverage and classification task. Note that `BSI` samplers perform better than the other link-trace samplers.

3.6 CONCLUSION

In this chapter, we introduced a novel task-independent sampler for attributed networks. We showed that content sampling is essential because while data-mining tasks on network content are common, sampling on internet-scale networks is expensive. While uniform sampling-based link-trace samplers `RW` and `MHRW` are attractive, it provides an unbiased estimate of the attribute distribution; however, the estimate is only achievable at asymptotic limits. Hence, link-trace samplers are widely used. However, these samplers are attribute agnostic and focus on preserving salient properties of network structure, not node content. Next, we introduced `SI`, an attribute-aware sampler grounded in Information Theory. We proved that the sampling problem was NP-hard by showing that familiarity, the converse of

surprise, was monotone and submodular. Further, we showed via an empirical counterfactual analysis that in many real-world datasets, `SI` performs (on a clustering task) as well as the best-known approximation [125] to the NP-hard problem. We showed strong experimental results for a variety of datasets, demonstrating that surprise-based samplers are sample efficient and outperform both random sampling and baseline attribute-agnostic samplers by a wide margin. Given that social networks having noisy and missing information, we extended `SI` sampler to bayesian surprise sampler `BSI` for handling the given issue. We considered observable attributes in this chapter. However, user attributes' like diabetes and mental health are often not observable (or hidden) but infer-able. We show how to sample such hidden attributes in the next chapter.

CHAPTER 4: HIDDEN POPULATION SAMPLING

In this chapter, we present our sampling work for sampling hidden attributed entities or hidden populations from online social networks. Typically, researchers are interested in a subset of the population (hidden population) which are not directly queryable. Examples of hidden population include: people with mental illnesses [3], sex workers [5], cyber bullying [131], hacked accounts [132] to name a few. More generally, the researchers' goal is to find people that satisfy a certain property, but *crucially*, API does not provide direct querying of the hidden property. For example, one cannot use the phrase "mental illness" on Twitter to identify people potentially suffering from depression because they rarely use that phrase in any of their tweets to self-describe. Thus when we refer to "hidden populations," we are more concretely referring to populations with a non-queryable property.

To sample the hidden population, we propose a hierarchical Multi-Arm Bandit (`DT-TMP`) sampler that uses a decision tree coupled with reinforcement learning to query the combinatorial attributed search space by exploring and expanding along high yielding decision-tree branches. More specifically, our sampler exploits the correlation between queryable attributes and the population of interest and hierarchically orders the query space to efficiently search for the hidden populations. A comprehensive set of experiments over a suite of twelve sampling tasks on three online web platforms and three offline entity datasets reveals that `DT-TMP` outperforms all baseline samplers by up to a margin of 54% on Twitter and 48% on RateMDs. An extensive ablation study confirms `DT-TMP`'s superior performance under different sampling scenarios.

The rest of the chapter is organized as follows. In the next section, we provide an overview of the hidden population sampling problem. We present the challenges involved with sampling, insights to tackle the challenges and a summary of contributions. Then, in Section 4.2, we describe our proposed `DT-TMP` sampler in detail. In Section 4.3, we compare our proposed sampler with several state-of-the-art samplers on five offline and three online datasets. In Section 4.4, we present a discussion of issues raised in the chapter. Finally, we present the conclusion of this chapter in Section 4.5.

4.1 OVERVIEW

In this section, we provide an overview of the hidden population sampling problem, followed by the sampling challenges. Next, we present an insight into our proposed solution that tackles the sampling challenges. Finally, we present the summarized list of our contri-

butions.

4.1.1 Why hidden population sampling?

Social networks have made available large amounts of public information online. This colossal information has led to the growth of industries such as social media marketing and management, online advertisements. Further, it has led to scientific advancements in the analysis and mining of social information by several organizations, including ICWSM and ASONAM.

However, with the enormous size of crowds on social networks such as Twitter and Facebook, we face difficulty in mining information concerning a target group of the population. For instance, there are over 330 million monthly active Twitter users and over 2.3 billion monthly active users in Facebook as of December 2018. Typically, researchers are interested in a subset of the population (hidden population) which are not directly queryable. Entities of such target population satisfy a property that is typically identified using an oracle (human or a pre-trained classifier). Examples of hidden population include: people with mental illnesses [3], sex workers [5], cyber bullying [131], hacked accounts [132] to name a few.

In this work, *we focus on identifying hidden populations through attributed search.* Many social networks allow for searching via attributed query, in addition to textual query. For example, we can also specify time and location attributes on Twitter (see Figure 4.2) in addition to text. In contrast to attributed search, text search has received considerable attention in the IR community. While there exist significant work on web crawling [18, 19, 133], we are missing such technologies to mine the entities, e.g., users on Twitter or GitHub) using their social attributes (e.g., for Twitter: their tweets, location; for GitHub: programming language). To address this gap, we explore the problem of hidden population sampling from online social platforms using attributed search for the first time.

4.1.2 Challenges in Hidden Population Sampling

There are two prominent reasons why attributed search of hidden populations on social networks is challenging:

Combinatorial search space: A combination of queryable attributes expresses a query issued to the application programming interface (API). Thus, the size of query space is the product of the attribute cardinalities (the number of possible values for each attribute) and grows exponentially as the number of attributes increases. Since social networks employ rate limits affecting the number of queries (e.g., Twitter API allows only 60 API calls in an

Figure 4.1: The entity population is represented by a cube (color indicates the distribution of hidden entities). `DT-TMP` iteratively explores the entity population using a drill down approach. It first finds the best query plane (general query), followed by the next best subquery within the best general query and so on.

hour) and search result limits (e.g., Twitter API returns at most 100 entities in a single API call) affecting the number of results obtained in a single query, a naive hidden population sampler is likely to remain in exploration phase after it exhausts the query budget.

Black-box API: The internal mechanism of online APIs is often propriety information unavailable to users. The sampler interacts with API using a query to get a subset of entities (returned result) matching the query. Some of the challenges associated with an unknown query system are: a) *stochastic feedback* i.e., different pages of a query very likely have a different number of hidden entities, b) *re-sampling of a hidden entity*, i.e., the API returns the same hidden entity for two different queries when the entity satisfies both queries, c) *variable size of returned results*, i.e., the API fewer than the page-size of entities when there are not enough matching entities in the population. Thus, it becomes difficult for online samplers to efficiently discover high-quality queries that would lead to the sampling of a high number of hidden entities.

The key insight for hidden population sampling is to identify high-quality queries and issue them multiple times. First, we address the problem of combinatorial search space by hierarchically organizing the query space in the form of a tree. Subsequently, we use a *decision-tree* based search strategy to systematically explore the query space by expanding along high yielding decision-tree branches. Second, we address the problem of black-box API by using the returned set of results to estimate the quality of not just the issued query but also related queries sharing one or more attribute combinations. We employ a

reward function to estimate the unique un-sampled hidden entities that can be obtained by issuing a query. Our unified reward function takes into account the stochastic feedback and re-sampling effect while allowing for an exploration-exploitation among queries. We use reinforcement learning-based *Thompson sampling* to define the reward function.

We put together the above insights to propose a Decision-Tree Thompson sampling (`DT-TMP`) algorithm [134]. We illustrate the working of `DT-TMP` using a 3D toy model in Figure 4.1 comprised of three queryable attributes represented by three dimensions. The `DT-TMP` algorithm maintains a query pool from where it queries the API. We initialize the query pool with the most general query, i.e., the algorithm searches from the entire population. Then, after an epoch time, the algorithm identifies the most promising query (i.e., attribute and the corresponding value)—the one with the highest reward. The query pool expands to add new queries, as shown in the second subplot. That is, `DT-TMP` issues a query with this attribute value as a predicate to identify the next attribute and corresponding attribute value to use as a conjunct. Thus, the algorithm over iterations converges to explore among the high hidden entity density regions of the population. Based on the returned set feedback, `DT-TMP` updates reward function corresponding to every query in the query pool. Finally, the algorithm terminates when we have exhausted the query budget.

4.1.3 Contributions

Our contributions are as follows:

Novel sampler: We sample hidden populations from online social platforms using attributed search for the first time. We show that our proposed sampler `DT-TMP` for hidden population sampling is robust to several sampling scenarios such as missing information and the classifier accuracy.

Extensive experimentation: We show extensive empirical results on five real-world offline datasets and three online datasets over a suite of fourteen hidden populations that our proposed sampler outperforms the state-of-the-art samplers.

Theoretical analysis: We show theoretical proofs justifying the usage of combinatorial querying system and proof efficiency of our proposed algorithm `DT-TMP` over baseline samplers (`TMP`).

Ablation study: We perform an extensive ablation study to understand the impact of different sampling factors. We find page-size, attribute cardinality, the number of

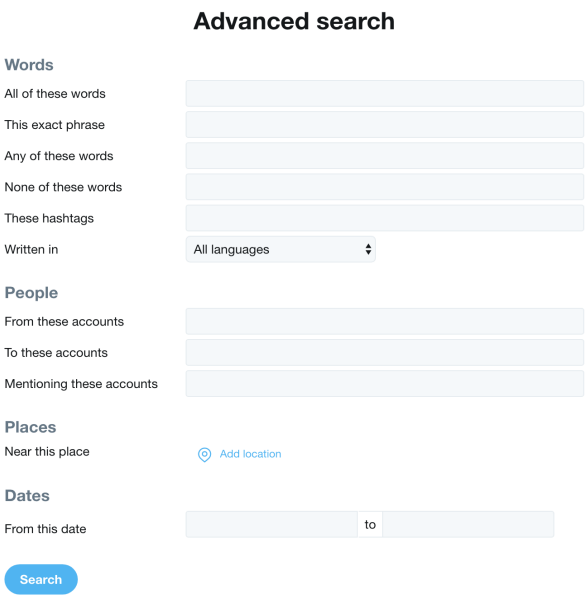


Figure 4.2: Twitter query interface shown as a typical example of online social platform interface comprising several queryable attributes.

queryable attributes, and correlation between queryable attributes and the hidden property are the prominent factors that affect sampling.

4.2 PROPOSED SAMPLER

In this section, we motivate hidden population sampling through a real-world example and a formal definition of the sampling problem. Next, we describe our proposed sampler, Decision tree-based Thompson sampler (`DT-TMP`), as a decision-making agent.

4.2.1 Problem Illustration

Consider a scenario in which a healthcare expert or a researcher is interested in reaching out to the depressed population on Twitter. Assume that the expert has designed a classifier for identifying whether a Twitter user is depressed or not based on the user's profile description and activity [135, 136]. The expert's objective is then to retrieve a maximum number of Twitter users that have the *hidden property* of depression. However, it is not trivial under present circumstances to sample the depressed population from Twitter due to several bottlenecks. The database of Twitter accounts is accessible only through Twitter's application programming interface (API). Twitter allows its user accounts to be queried by only some specific *attributes* such as time, location, and text. Besides, Twitter permits

only 180 API calls in a 15-minute window. Notice that the depressed population is distributed across the entire Twitter population necessitating the sampler to consider all types of queries. Furthermore, the distribution of the depressed population across different queries is unknown, making it difficult to frame queries that would yield a high discovery of the depressed population within a limited *budget* of API calls.

To explain the sampling framework, we describe two major features of online social platform services like Twitter API: *query interface* and *returned-result set*. Query interface as shown in Figure 4.2 lets the expert query Twitter by setting attribute-values to the queryable attributes. For example, the expert may set the location attribute to 'New York' and text-attribute to 'mental health' and time attribute to '*' (or ignoring it). In other words, the *query* can be interpreted as a conjunction over queryable attributes say A_i $(i = 1, 2, \ldots r)$ where attribute-values z_i of the attributes are obtained from their respective attribute domains $z_i \in d_i$ where $d_i = dom(A_i) \cup \{*\}$. Formally, we shall represent a query involving r queryable attributes by $\bigwedge_{i=1}^{r} z_i$. In this work, we consider only the conjunctive combination of attributes as a query.

On issuance of a query, Twitter API by default returns a result page comprising m (=20) entities and a pointer to the next page of results. The sampler may obtain the subsequent m results for the same query by issuing another API call or get another m results by issuing a different query. Thus, the sampler incurs a unit cost of communication for each query issued. Subsequently, the profiles of entities returned by the API are analyzed to identify whether they satisfy the hidden property of depression or not. Since the cost incurred in determining the entity's hidden property is directly proportional to the API call cost, we use API cost as the sole cost constraint of the problem.

4.2.2 Problem Statement

Now, we formally define the hidden sampling problem as follows.

Problem definition: *Suppose entities on an online social platform are queryable using a conjunctive combination of r queryable attributes A_i for $i = 1, 2, \ldots r$. Further, consider that there exists a target subpopulation satisfying a hidden property that is verifiable by an oracle (usually a classifier). Given a budget B of API calls, the sampler's objective is to maximize the count of sampled entities satisfying the hidden (target) property.*

Problem hardness: The sampling of the hidden population is challenging in the real-world setting due to several reasons. One, the number of possible queries that can be constructed using the queryable attributes $(\prod_i |d_i|)$ is exponential in the size of attribute domains. Given a fixed API budget, query selection from an exponential query space makes

the problem particularly challenging. Two, the API returns m or fewer results, and when similar queries are used, it often leads to re-sampling of the same entities. It, therefore, becomes pertinent to handle the re-sampling issue so that the maximum number of distinct hidden entities can be sampled. Three, the limited API calls necessitate that the sampler manages an exploration-exploitation tradeoff. Given that the sampler is oblivious of the correlation between the queryable attributes and the hidden property, it has to tradeoff the API calls between learning the correlation and issuing the highly correlated queries. In addition, a hidden population sampler is intended to be used for sampling a diverse set of entities such as books, restaurants, products, or people through the web interface of online platforms such as the library, Yelp, Amazon, and LinkedIn, respectively. Given the variety of online platform settings comprising different sorts of queryable attributes and an unspecified hidden property, an ideal sampler should exhibit the following characteristics.

- *Simplicity*: The model should be applicable to web-forms with different numbers and types of attributes with varying attribute cardinality.
- *Online*: The model can be updated using only the feedback obtained by the returned result analysis.
- *Unsupervised*: The lack of training examples is usually the prime motivation behind hidden population sampling. Prior distributional information about the hidden population is unavailable to the sampler; thus necessitating the algorithm to be unsupervised.
- *Task-independence*: The sampler is oblivious of the hidden property, and it could therefore be used for sampling any well-defined hidden population, i.e., the sampler should be able to adapt when plugged-in with a different black-box classifier used as the oracle.
- *Perpetual*: Ideally, a sampler is expected to be efficient both when the budget is limited and when the budget is asymptotically large.
- *Flexibility*: The model could be easily extended when extra information such as attribute semantics or attribute correlations is partially available.

In the following section, we propose a hidden population sampler that exhibits every aforementioned characteristic. Table 4.1 lists some of the frequently used terms and their definitions in this chapter.

Table 4.1: Terms and their definitions.

Term	Notation	Definition
Page size	m	Maximum number of results returned in a single API call.
Query precision	p_q	Fraction of target entities in the population matching a given query.
Attribute cardinality	$\|dom(A_i)\|$	Number of attribute-values of a given attribute.
Attribute domain	$dom(A_i)$	Set of all possible attribute-values of a given attribute.

4.2.3 Decision Problem

We show that the process of sampling hidden population from online social platforms as described in Section 4.2.2 is primarily a decision problem. The sampler continuously decides which query to issue to the API such that the sampler obtains a maximum possible number of entities from the hidden population within the given API budget. Based on the sampled entities, the sampler maintains a probabilistic model of the entity database that gets updated over time. The model is used to construct a query. The returned-results obtained from issuing the constructed query are subsequently used to update the model. This cycle of query construction, returned-result analysis, and model updation continues until the API budget runs out. We deliberate upon each component of the cyclic process separately.

We maintain the model of the entity database using a set of probabilistic parameters. In the absence of any prior semantics or syntactic information about the attributes, the model treats each queryable attribute such as location, time, and keywords in Twitter as independent variables. Furthermore, we model the attribute-values of every attribute independently, i.e., 'New York', 'Los Angeles' and 'Chicago' corresponding to location attribute is modeled independently as well. The above assumptions concerning the entity database allow our model to be applicable across a suite of online social platforms. The probability model of the entity database is used to not only estimate the utility of issuing each possible query but also to construct the next query.

The sampler interacts with the online social platform via the query interface. As stated in Section 4.2.2, we represent the query as a conjunctive combination of discrete attributes. In compliance with our problem formulation, we approximate continuous attributes by discretizing them into different bins and handle text search by an expert-based selection of a few relevant textual phrases. Note that the number of possible queries that can be constructed using the queryable attribute is still exponential, i.e., $\prod_i |d_i|$ which is typically very high. The exponential number of choices makes the decision problem even more challenging.

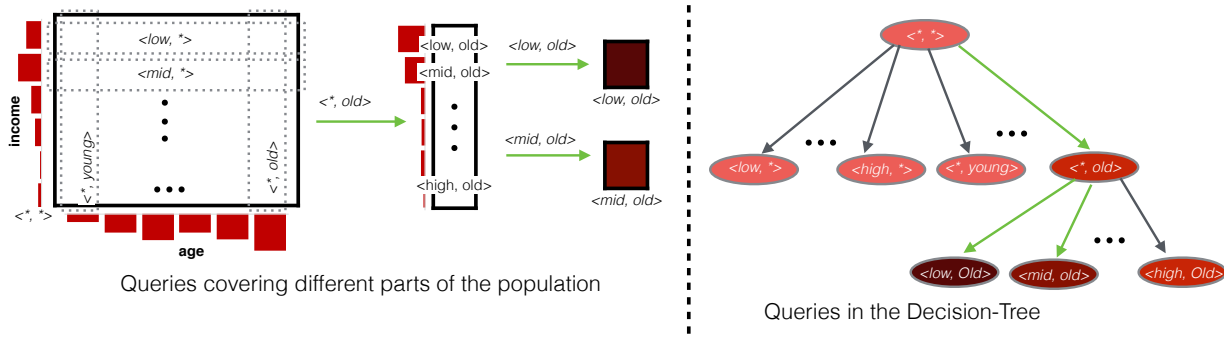


Figure 4.3: Model description of `DT-TMP`. For hidden population of 'mental illness' (represented in red color), the `DT-TMP` searches the population for the best combinatorial query comprising of two queryable attributes: income and age. It first uses `<*, *>` query to find the best single attributed query from queries such as `<Low, *>` and `<*, Young>`. Subsequently, it finds the best query `<Low, *>` along which it expands its query search. The decision tree on the right shows the query expansion withe the query expansion along green links.

On issuance of an attributed query, the online social platform returns a list of entities: 'returned result set.' As shown in the Twitter example, the same attributed query can be used to gather more results by traversing over the next pages. The returned results act as feedback for the sampler, which is used to update the model. The number of entities belonging to the hidden population indicates the quality of the query. Thus, the core objective of the hidden population sampler is to find high-quality queries and to issue those queries repeatedly.

Next, we describe a detailed solution to the cyclic process of decision-making for hidden population sampling by employing a decision-tree-guided multi-armed bandit algorithm. In the later sections, we show the utility of our proposed sampler over a range of tasks.

4.2.4 Proposed `DT-TMP` Algorithm

In this section, we present solutions to the three prominent challenges encountered by a hidden population sampler in the real-world setting. The challenges are exponential query space, unconventional reward feedback, and an unknown correlation between queryable attributes and hidden property. Next, we fully describe the proposed Decision-Tree Thompson sampler (`DT-TMP`).

First, as noted in the problem statement, any hidden population sampler constructs its queries by choosing attribute-values z_i from d_i of each queryable attribute A_i. The size of query space is therefore exponential in the attribute cardinality $\prod_i |d_i|$. We deal with the issue of exponential query space by hierarchically ordering the queries from the most general to the least general (or the most specific) query. Figure 4.3 shows the hierarchical

organization of queries. For instance, a query where location attribute is set to 'Chicago' and text attribute is ignored (or set to '*') is a generalization of the query where location is set to 'Chicago' and text attribute is set to '#Cubs' since the former includes all entities matching the later. In principle, a query q_1 is a generalization of query q_2 if the set of population entities matching query q_1 is a superset of the set of entities matching q_2. Hence, the most general query is one where z_i is set to '*' for every attribute.

Second, the analysis of the returned result set by the API is a non-trivial task because of the partial information available during sampling and the re-sampling issue. In each API call, the sampler obtains partial information in the form of the returned set of m or fewer results. For illustration, consider that a query where location attribute is set to 'Chicago' yields 5 entities from the depressed population out of 20 returned entities on the first result page. We shall assume the query precision or the fraction of hidden entities matching this query is $5/20 = 0.4$. In other words, it is very likely to obtain another 5 hidden entities when a new API call is made for the next result page of the same query. More generally, we model the query precision probabilistically using a Beta distribution which is typically used to model the probability of probabilities [137]. Furthermore, a query where location is set to 'Chicago' is very likely to lead to the entities that also satisfy queries where the text attribute is '#Cubs.' The sampler, therefore, needs to update the quality metric of queries where text attribute is set to '#Cubs' so that it avoids making redundant queries that lead to re-sampling of same hidden entities. We avoid the re-sampling by estimating the expected number of distinct unseen entities to be discovered by issuing a query q. Without making any assumption about the ordering of results for a specific query, we assume that the results are returned either via sampling with or without replacement from the set of entities matching the given query. For sampling with replacement, we derive a reward function as follows.

Assume that the number of entities in the database satisfying a query q is N_q. Further, assume that the sampler has already observed n_q distinct entities satisfying the query q out of which there are S_q number of target entities and F_q number of non-target entities. Therefore, from Standard Probability Theory, we obtain the following expected reward r_q when query q is executed.

$$\mathbb{E}[r_q] = \underbrace{\frac{S_q}{S_q + F_q}}_{\text{expected \# targets}} \cdot \underbrace{\frac{N_q - n_q}{N_q}}_{\text{new}} \cdot \underbrace{\left(1 - \left(1 - \frac{1}{N_q}\right)^m\right)}_{\text{unique}} \tag{4.1}$$

where m is the maximum number of results returned by the web API in response to a single query. The expected reward is the estimated number of new distinct hidden entities

that are likely to be obtained by re-issuing the query q.

For sampling with replacement, we update the reward function by dropping the third term since unique entities within the result page of a query is guaranteed (i.e. $\frac{S_q}{S_q+F_q}\frac{N_q-n_q}{N_q}m$). When N_q is unknown, we assume that $N_q >> n_q$, therefore the reward function approximates to just the first term (i.e. $\frac{S_q}{S_q+F_q}m$).

The query precision for any query is an unknown measure to an unsupervised hidden population sampler. If the precision values are known, the sampler would straightforwardly formulate its queries using only the high precision queries. In order to obtain an unbiased estimate of the precision, the sampler needs to explore over the combinatorially large query space. While a naive exploration of queries is beneficial for formulating better future queries learned from the unbiased estimates, it leads to poor immediate results. We, therefore, employ Thompson sampling for handling this exploration-exploitation tradeoff of queries. Thompson sampling is a well-known optimal MAB algorithm that achieves the lower regret bound of the MAB problem [138]. Notice that Thompson sampling is consistent with the independence assumption of attributes and attribute-values made in Section 4.2.3.

Now, we generalize the intuitions presented above to propose a simple yet effective hidden population sampler: Decision-Tree Thompson sampler (`DT-TMP`).

Description of algorithm: `DT-TMP` is a unique combination of a standard Decision Tree [139] and Thompson sampling [140]. `DT-TMP` maintains a query pool $\mathcal{Q}$ comprising of queries explored by the algorithm. The query pool is initialized with the most general query. Typically, the most general query can be represented using the queryable attributes as $\bigwedge_{i=1}^{r} *$. The most general query is initially used to sample from the entire population. `DT-TMP` expands the query pool by adding more specific queries.

For every query $q \in \mathcal{Q}$, `DT-TMP` tries to predict the future reward that would be obtained when query q is issued. Based on the prediction, `DT-TMP` chooses the best query to issue. A query issued to the API yields a result page comprising m entities. Each returned entity is evaluated as a success or failure depending on whether it belongs to the hidden population or not. We model each success and failure of every returned entity as a random sample drawn from an unknown Beta distribution of the query that the model learns over iterations. We use a non-informative uniform prior $Beta(1, 1)$ as the starting state for every query. This choice of Beta distribution permits us to efficiently update the posterior distribution upon receiving the returned results.

We now show how to update the posterior distribution of any query $q' \in \mathcal{Q}$ when another query q is issued. If q is a generalization of q', we increment the success or failure parameter of Beta distribution by one depending on whether the returned entity is in target populace or not, and the returned entity matches query q. In the other case, when the specific query

q accounts for only a fractional part of the general query q', we update the Beta distribution of the general query proportionately. That is, if q is a specific version of q', we increment the success or failure parameter of q by the ratio of population size matching query q to population size matching query q'. We are able to estimate this fraction directly from the returned result since the query pool is expanded hierarchically from the most general to the most specific queries.

Analysis of DT-TMP algorithm: At each step of the iteration, `DT-TMP` employs Thompson sampling to select the best query among the query pool. Note that the query pool is fixed over epoch time h to ensure that enough entities are sampled before expanding the query pool. The query pool is expanded by adding new specific queries corresponding to the best query in $\mathcal{Q}$. We prove using Lemma 4.1 that expansion of a general query always leads to an equally good or a higher precision specific query. Thus, `DT-TMP` continues to find the highest quality query until the budget is finished.

Lemma 4.1. The query precision of the specific queries is centered around the query precision of their corresponding general query. Further, there exists a leaf node of the decision tree with the highest quality precision.

Proof. Lets assume that the query precision of a general query q_g is p_g, and it has n immediate specific queries q_i as children in the decision tree. Assume that the query precision of the i-th specific query q_i is p_i where $i \in \{1, 2, \ldots n\}$. Further, we denote f_i as the ratio of the size of the population matching query q_i to the size of population matching the general query q_g. Since the disjoint specific queries cover the general query, $\sum_i f_i = 1$.

By preservation of hidden entities, the query precision of the general query p_g can be expressed as the weighted average of the specific queries's precision. That is,

$$p_g = p_1 f_1 + p_2 f_2 + \ldots p_n f_n \tag{4.2}$$

Since p_g is a weighted average, it is bounded by the maximum and minimum of the specific queries' precision.

Following the above argument, we note that there always exists a specific query whose precision is strictly greater or equal to the query precision of the general query. Since the leaf nodes are the most specific queries in the decision tree of `DT-TMP`, it follows that one of the leaf nodes of the decision tree has the highest precision. QED.

Algorithm 4.1 summarizes the `DT-TMP` algorithm via pseudo-code. We provide a diagrammatic representation of `DT-TMP` algorithm in Figure 4.4 in a stylized sampling environment

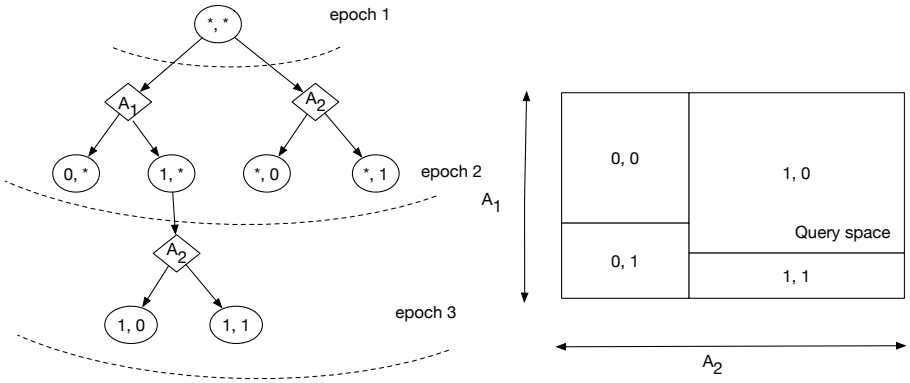


Figure 4.4: Decision tree diagram shows how new attribute-values pairs are added for exploration as DT-TMP samples queries from two binary attributes A_1 and A_2. The query at root of the tree covers the entire database represented by box in right subplot. General queries are situtated at higher levels of the tree while specific queries that cover only smaller subsets of population are situated at lower levels of the tree.

where there are only two binary queryable attributes. Next, we perform a detailed analysis of the DT-TMP algorithm.

Complexity analysis: Even though DT-TMP models the complex relationship among queryable attributes and hidden attributes while keeping an exploration-exploitation tradeoff among different queries, it is surprisingly easy to implement and has a linear space and quadratic time complexity. For practical reasons, we assume the number of attributes r and the page size m to be constant. In each iteration in the outer-loop, the maximum number of queries added to the query pool is limited by n where $n = \sum_{i=1}^{r} |dom(A_i)|$; the budget B limits the number of iterations. Furthermore, the decision tree takes $\mathcal{O}(\mathcal{Q})$ space which is bounded by $\mathcal{O}(nB)$. Every query in $\mathcal{Q}$ uses a constant space parameter set to estimate the reward distribution. Thus, the overall space complexity of the DT-TMP is $\mathcal{O}(nB)$. A similar analysis implies the time complexity of DT-TMP is $\mathcal{O}(n^2B)$ since each iteration (sampling from and updating of Beta distributions is performed in constant time) involving updation of specific and general query combinations take $\mathcal{O}(n)$ time. Finally, we notice that DT-TMP can easily be parallelized by having the worker nodes use the preceding iteration's reward distributions while having the master node maintain the central decision tree.

Lastly, we note that at limiting budgets, DT-TMP behaves as TMP when the query pool expands to the entire query space. Note that Thompson sampler has optimal regret bound for multi-armed bandits [141]; thus TMP is an optimal sampler at limiting budgets when all queries have been sufficiently queried.

Lemma 4.2. At asymptotic limits of the budget, DT-TMP tends to a naive Thompson sam-

Algorithm 4.1 Decision tree- Thompson sampler

$S_q = 1, F_q = 1, \forall q$. ▷Successes and failures of query q
$\mathcal{Q} = \{\bigwedge_{i=1}^{r} *\}$ ▷ query pool
$\mathcal{S} = \phi$ ▷ sample set of entities
for $t = 1, 2, \ldots B/h$ **do** ▷ Communication rounds
    for $j = 1, 2, \ldots, h$ **do**
        for $q \in \mathcal{Q}$ **do**
            $r_q \sim$ Beta $(S_q, F_q) \times \frac{N_q - n_q}{N_q} \times N_q\Big(1 - \big(1 - \frac{1}{N_q}\big)^m\Big)$
                ▷ for sampling with replacement
        $q^* = argmax_{q \in \mathcal{Q}}\ r_q$
        $R =$ Execute query q^* ▷ Returned result
        $\mathcal{S} = \mathcal{S} \cup R$
        $s_{q^*} =$ Number of successes in R
        $S_{q^*}, F_{q^*} = S_{q^*} + s_{q^*}, F_{q^*} + |R| - s_{q^*}$
        # update Beta parameters of other queries
        for $q' \in descendant(q^*) \cap \mathcal{Q}$ **do**
                ▷ Descendent nodes in DT are specific queries of q^*
            $S_{q'}$+= # success in R matching q'
            $F_{q'}$+= # failures in R matching q'
        for $q' \in ancestor(q^*) \cap \mathcal{Q}$ **do**
                ▷ Ancestor nodes in DT are general queries of q^*
            $\rho =$ Est. population size of q / population size of q'
            $S_{q'}$+= $\rho\times$ # success in R
            $F_{q'}$+= $\rho\times$ # failures in R
    $\mathcal{Q} =$ Query-expansion$(\mathcal{Q}, q^*, \mathcal{S})$ ▷ adds new queries to the query pool

Algorithm 4.2 Query-expansion$(\mathcal{Q}, q^*, \mathcal{S})$

for $j = 1, 2, \ldots, r$ **do**
    if $A_j == *$ **then**
        for $v \in dom(Q_j) \wedge v \in \mathcal{S}_j$ **do** ▷ add all possible attribute-values observed in sample
            $\mathcal{Q} = \mathcal{Q} \cup q^*(A_j = v)$ ▷ We add new specific query for unexplored attribute-attribute value pair corresponding to the best query q^*
return $\mathcal{Q}$

pler.

Proof. Given that every branch is initialized with a query precision $Beta(1, 1)$, there is a non-zero probability of selecting any branch using best query selection of `DT-TMP` (line 9 of Algorithm 4.1). `DT-TMP` will therefore explore every branch of the search tree at asymptotic limits of budget. Since, at asymptotic budget limits, all branches of the decision tree will be explored; hence the query pool will expand to cover the entire query space. When the query pool $\mathcal{Q}$ covers the entire query space, `DT-TMP` and `TMP` are identical. QED.

4.3 EXPERIMENTS

In this section, we present experimental findings by executing different hidden population sampling strategies over offline and online real-world datasets.

4.3.1 Offline Experiments

We now describe in detail the offline experiments, including the description of the datasets and the query interface that allows samplers to access the entities within these datasets. Next, we discuss the evaluation metric used to compare the efficacy of the baseline and proposed samplers. Thereafter, we show the results of the samplers' performances and interpret the results.

Datasets Now, we present a summary of three real-world datasets used for offline experiments, along with a description of the query interface (API).

In similarity to the offline experiments in [21], we simulate a typical online social platform using data from five real-world entity datasets. The datasets—Patent [142], Auto[1], and Adult [143], Auction [144], and IMDB [145]—are obtained from notably diverse domains. The datasets vary in the number of attributes, their attributes' cardinality, and the distribution of arm sizes of the attributes (i.e., number of records across attribute values). The local server allows us to vary the sampling parameters such as the result size, the attribute cardinalities, and the attribute correlation values. We query the local server using queryable attributes and evaluate our algorithms based on the coverage of the hidden population entities. Table 4.2 summarizes the statistics of the five datasets concerning the respective hidden target population as: patents authored by Japanese researchers, automobiles that

[1] `https://www.kaggle.com/orgesleka/used-cars-database`

have less than $40K$ kilometers mileage, adults who earn more than \$50K per annum, auctions that are successfully sold out and movies having adult content. We use all queryable attributes specified in the table for searching the corresponding hidden target population in the datasets.

Hidden target selection. For experimental validation, we choose the hidden property of the hidden population as the attributes that are not expressible as a combination of one or more queryable attributes. For example, the queryable attributes such as category, subcategory, and class of a patent cannot be used to search for authors with a specific nationality (hidden property). More importantly, the choice of hidden property for offline datasets was motivated by real-world scenarios. We note that academic search engines such as PubMed, Google Scholar, and Microsoft Search are not searchable using properties such as the author's nationality and writing style. Similarly, mileage information is a non-queryable property of automobiles in the popular advertisement website Craigslist. Lastly, income is often a hidden non-queryable property of users in existing search interfaces of popular sites such as LinkedIn and Angelist but can be easily inferred given their job description. In our datasets, the target population comprises 18.73%, 5.58%, 23.93%, 30.84%, 3.33% of the population size in Patent, Auto, Adult, Auction and IMDB datasets, respectively.

Patent dataset is a collection of two million patent records from US patent records. The queryable attributes include categorical attributes such as category and subcategory of patents. Given that the nationality of the inventor is a hidden property, we choose the patents that are invented by Japanese researchers as the target entities in the dataset. Patents authored by Japanese researchers were chosen for only illustrative reasons. We observe similar empirical performance across samplers when the target population is set to patents authored by researchers from other countries such as China and India.

Auto dataset is a collection of 371,469 auto-vehicle records crawled from eBay-Kleinanzeigen. The set of queryable attributes allows users to search for cars by searching over a variety of attributes such as car type, model, and brand. For this dataset, we define the hidden target population of automobiles by setting an arbitrary threshold of their travel miles.

Adult dataset is a collection of 48,842 records of adults extracted by Barry Becker from the 1994 Census database. Similar to online social networks such as Facebook and Pokec, the users can be searched within the entity database based on their public attributes such as education, marital status, and gender. For this dataset, we consider income as the private or hidden attribute of an individual.

Auction dataset is a collection of 258,588 records of sport-related auctions [144]. eBay is one of the largest online platforms that allows sellers to auction their products. The auctions can be searched based on attributes such as auction creation time and price. For

Table 4.2: Description of real-world dataset: Patent, Auto and Adult and their respective queryable attributes with cardinalities in parenthesis, and the target hidden attribute.

Dataset	Queryable attributes	Hidden property
Patent	category (6), subcategory (26), assignee type (7), nclass (417)	inventor's nationality
Auto	vehicle type (9), model (252), brand (40), fuel type (8), repairing (3)	mileage
Adult	class (9), education (16), marital status (7), occupation (15), relationship (6), sex (2)	income
Auction	price range (3), number of Items listed (2), category (45), Hall of Fame product (2)	old
IMDB	title (10), year (3), genres (28)	adult content

this dataset, we consider the hidden auctions as the auctions which were successfully sold.

Internet Movie Database (*IMDB*) comprises over 5.5 million items, including movies, television programs, and home videos. IMDB items can be searched based on queryable attributes like title, genre, and release year. For this dataset, we consider the hidden movies as the movies which were marked as having 'adult' content.

In the absence of a well-defined query interface system, we simulate the query interface in the following way. The query interface to the datasets allows only conjunctive queries formed using the queryable attributes listed in Table 4.2. For each query, the user incurs a unit cost in the API call. The query interface returns a set of k random results (default value of page size is set to 10 unless otherwise stated) drawn from the query matching entities with replacement. We observe similar empirical results across samplers when the query interface returned k random results drawn from the matching entities without replacement.

Evaluation We assess the performance of a sampler by the number of distinct entities of the hidden target population that the sampler collects within a given query budget B. Alternative definitions include coverage, query harvest rate, and precision [146]. Owing to the similarity in performance of samplers across the aforementioned evaluation metrics, we focus on the simplest evaluation metric: recall. Formally, recall (R) of a sampler after making budget B API calls is calculated as follows,

$$R = \frac{\#(\text{hidden target entities retrieved})}{\#(\text{target entities in dataset})} = \frac{N_S}{N_D} \tag{4.3}$$

We observe that recall R is a very small quantity under limited budget constraint because of the large number of N_D target entities in datasets like Patent. Since it is possible to

sample only a limited N_S number of target entities to be sampled within a limited budget, the overall recall value across all samplers is pretty low ($\frac{N_S}{N_D} << 1$). Therefore, we normalize the recall values by the theoretically maximum recall attainable at a budget B, which is same as getting all target entities in every result of API call, i.e., (page-size $\times$ B) unique target entities.

All evaluation results are reported over 100 independent runs.

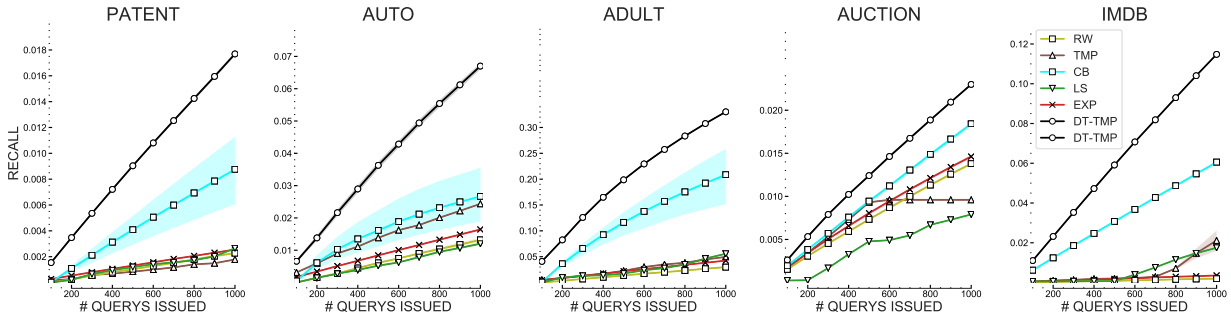


Figure 4.5: Sampling performance of baseline and proposed `DT-TMP` sampler on offline datasets. `DT-TMP` is shown to be the best sampling strategy. Decision tree based search allows `DT-TMP` to explore high yielding queries in a combinatorial query space while simultaneously exploring-exploiting high yielding queries. The bands indicate 95% two-sided confidence interval.

Baselines We now enumerate different sampling strategies to compare against our proposed sample.

- Uniform sampling over an entire database (`UNI`). As evident from the name, this sampler samples the hidden population by repeatedly issuing the most generic query until it exhausts the budget B. Thus, the sample obtained using this sampler is a uniform sample over the entire population.

- Uniform query sampling from the query space or pure exploration sampling (`EXP`). At each time step, EXP queries the web API by randomly sampling with replacement a single query from all possible queries. Thus, this method performs exploration for B rounds, hence named exploration sampling or EXP.

- Thompson sampling (`TMP`) is a standard Thompson sampler [140] where the reward from each arm and draws the best-expected arm in each draw. In other words, it is a `DT-TMP` sampler without a decision tree.

- Lazy slice cover search (`LS`) [21] is an optimal algorithm for retrieving the entire entity set from the online social platforms while minimizing the number of queries.

- Content-based search (CB) [147] is a greedy query design algorithm that uses the sampled entities to construct new high yielding queries that are unique to the hidden population using the *tf-idf* (term frequency inverse document frequency) ranking.

- Random Walk (RW) [148] is an efficient algorithm for randomly sampling from the entity database by creating queries via random walk approach over the space of queries.

Several works in reinforcement learning literature [149] exist, such as Successive Elimination, UCB, UCT, and Thompson sampler, which are known to be optimal for handling exploration-exploitation in MABs, but we exclude them from this study since it is not the focus of this work.

Finally, we set the epoch h of DT-TMP sampler by default to 10 for all datasets. This setting ensures that the sampler uses feedback gained from $10m$ (typically 100) new observations to expand the query pool appropriately.

Results on Offline Datasets We now present a detailed description of experimental findings obtained from experiments performed on real-world datasets.

Now, we present the performance of baseline and proposed samplers under a variable query budget (ranging from 100 to 1K API calls) on the three real-world datasets—Patent, Auto, and Adult. Figure 4.5 shows the recall value for different sampling strategies across varying query budgets. We observe similar performances of naive MAB based samplers, EXP and TMP along-with UNI. The low performance of naive MAB-based samplers is expected given the exponential query space. The naive samplers have to explore among 2600, 11314, and 5970 non-empty queries (queries that have at least one matching entity in the dataset) in Patent, Auto, and Adult datasets, respectively. Thus, the exponential size of the query space causes these samplers to get stuck in the exploration phase. Notice that there are 630K, 2M, 181K possible queries obtained from the combinatorial combination of attribute-values corresponding to the queryable attribute ($\prod_{i=1}^{r} d_i$) but only a few non-empty queries. In our experiments, we allow the non-empty queries to be used as the arms of baseline samplers; DT-TMP however, lacks this information. However, we assume that DT-TMP obtains N_q number of matching entities corresponding to a query q when the query is issued as observed in real-world APIs like Google, Amazon, Facebook, and LinkedIn API. The UNI sampler that samples random entities from the entire population performs significantly better than naive MAB samplers; however, note that UNI is used as a theoretical baseline and not a feasible sampler for several real-world online platforms like Facebook, LinkedIn, and Twitter. Similar to UNI, RW and LS samplers are not target specific samplers. The greedy-based CB suffers from the problem of getting stuck in local, high-quality queries, whereas DT-TMP by virtue of

Thompson sampling maintains an optimal exploration-exploitation tradeoff. Decision tree-based MAB sampler (DT-TMP) significantly outperforms the second-best baseline sampler by a margin of 101.86%, 151.16%, 58.30%, 29.10% and 90/45% in Patent, Auto, Adult, Auction and IMDB dataset respectively. High recall performance of DT-TMP is due to the fact that it exploits the hierarchical structure of the combinatorial action space to greedily explore the attribute combinations (query) that yield a high number of hidden target entities.

In this section, we discussed the experimental outcomes of offline, real-world experiments. We used state-of-the-art MAB sampler (TMP) and hidden database samplers (RW, LS and CB) as the baseline samplers. DT-TMP is shown to perform remarkably well over all hidden population tasks across three different datasets by exploiting the structure in combinatorial query space. In the next section, we observe the sampling performances on online real-world datasets.

4.3.2 Online Experiments

In this section, we describe the details of deploying different sampling strategies on three real-world online web-query platforms—Twitter, RateMDs, and GitHub.

Twitter Twitter is a popular micro-blogging website that allows users to interact with each other via short messages called "tweets". As of December 2017, Twitter reported an estimate of 330 million monthly active users generate half a billion tweets every day [150]. Given the enormous content size and the API limitations, it is infeasible to sample the entire content. Thus, we need to design effective sampling strategies to sieve just the relevant information relating to the hidden population.

Twitter allows combinatorial queries for very few attributes; we use location and hashtags as the two queryable attributes. We consider the hashtags of top 10 National Football League (NFL) teams in USA [2] as the first queryable attribute. The second queryable attribute is the home cities corresponding to the 10 NFL teams.

Twitter REST API is used to gather tweets corresponding to all possible combinatorial attributes (query). Since Twitter API allows only seven days of data to be accessed, we collected all possible queryable tweets between a fixed time frame of 6 days (26 April 2018 till May 1, 2018). Note that Twitter API [3] returns by default of 20 tweets per API call and a maximum of 100 tweets. We vary the result-size from 10 to 100 and observe a negligible impact of the page size on the relative performance of the samplers. Further, the API

[2] https://www.usatoday.com/sports/nfl/rankings/

[3] https://developer.twitter.com/en/docs

returns tweets ordered by tweet's creation time, i.e., for a fixed query, Twitter returns the most recent tweets as the first page and older tweets as the subsequent pages.

We consider three hidden population sampling tasks on Twitter. We employ properties of Twitter users that are not queryable via Twitter API to define three hidden populations: female users, users who have verified Twitter accounts, and early adopters of Twitter based on when the users created their account. We use an off-the-shelf gender predictor [151] as our ground truth classifier for predicting a user's gender; the other two properties are available from user profile information. Consider a sports advertiser interested in reaching out to potential football fans who are female on Twitter. Such an advertiser would want to maximize the coverage of female Twitter users in their sample.

We use throughput-rate as the online sampling evaluation metric. In the absence of information about the size of the underlying target population, it is not possible to use recall as the evaluation metric. In the context of hidden population sampling, we define the throughput rate as the ratio of the number of unique target entities sampled to the theoretical maximum possible target entities that can be sampled. Thus, the throughput rate (TR) of a sampler after making budget B API calls is defined as,

$$TR = \frac{\#(\text{hidden target entities retrieved})}{B \times k} \tag{4.4}$$

where k is the page size. Note that throughput rate penalizes queries that yield results of size less page size. For example, consider for page size $k = 20$, a query q_1 that returns just a set of 5 target entities has the same throughput rate as another query q_2 that returns 5 target and 15 non-target entities, since the API cost incurred by both queries is same, and they both discover 5 target entities.

Due to API restrictions, we can implement only certain samplers on Twitter. Since Twitter API doesn't support uniform sampling, UNI could not be implemented. Therefore, we compare our sampler with five baseline sampling strategies—`EXP`, `LS`, `RW`, `CB` and `TMP`. Further, non-random ranking and absence of query size restrict the implementation of `DT-TMP`. This lack of information about matching results in Twitter of a given query led us to modify the expected reward to the fraction of target entities discovered in each API call. Furthermore, note that the individual attributes, i.e., hashtags and locations are the most general queries on Twitter.

Figure 4.6 reveals that `DT-TMP` is the predominant sampling strategy. It outperforms the second-best sampler `TMP` by a margin of 42.7% when the task is to sample female users. It shows a similar improvement of 39.83% and 79.24% over the second-best samplers when the hidden target population is set to users that have verified accounts and when the target

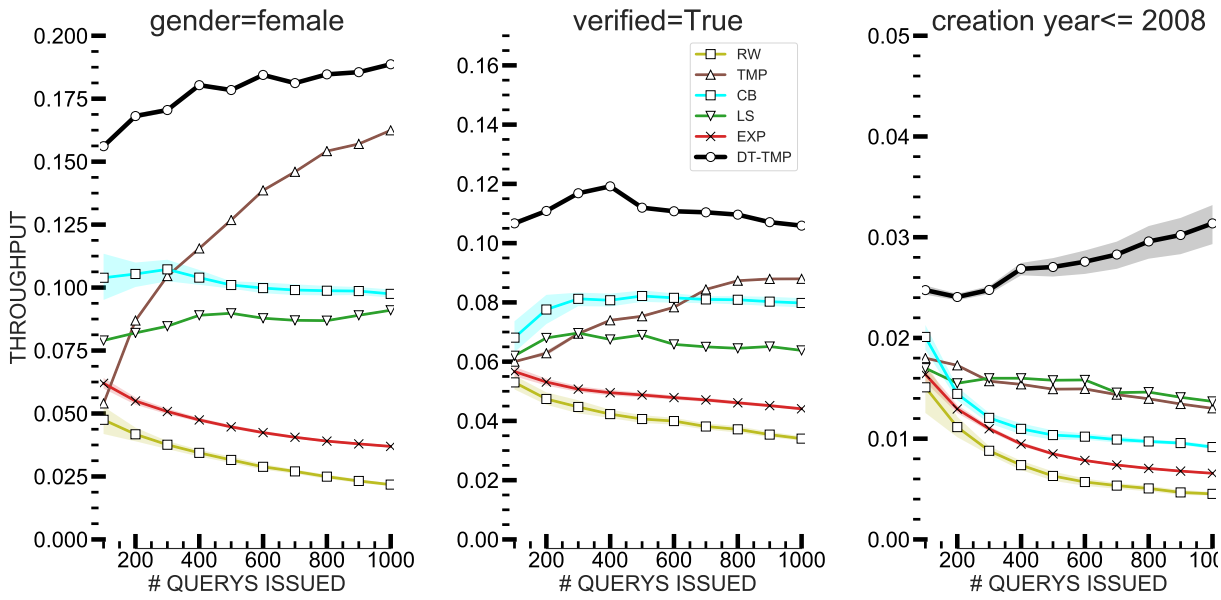


Figure 4.6: Throughput rate for different sampling strategies in Twitter. Combinatorial MAB (`DT-TMP`) does the best.

population of early Twitter adopters, respectively. Consistently superior of `DT-TMP` across different tasks demonstrates the usefulness of the algorithm.

RateMDs RateMD (`https://ratemds.com`) is a free healthcare website that allows users to read and submit reviews about doctors. According to the website, more than 100 million potential patients use RateMDs for information before making important healthcare decisions. There are four queryable attributes—gender, specialties, verified, and patient acceptance—to search for doctors. For gender information, the users have two options, male and female. For specialties, the users can specify one of the 57 specialties of doctors, including dentist, pathologist, and pediatrician. For patient acceptance, the users can restrict the result to doctors who currently accept new patients. For doctor verification, the users can restrict the search to only doctors verified by RateMD. Each query returns one page of 10 doctors, and the user can also specify which page to retrieve. We obtained the dataset by querying 10 pages for each of the attribute-value combinations in August 2018. Each doctor page is a profile consisting of user ratings, credentials, and acceptable insurance.

We consider three hidden populations of doctors: doctors with a 5-star rating, doctors who received more than 10 ratings, and doctors who accept at least three insurance. The experiments are conducted on the baseline and the proposed samplers; the results are evaluated using the same metric as the aforementioned online experiments. We observe the superior

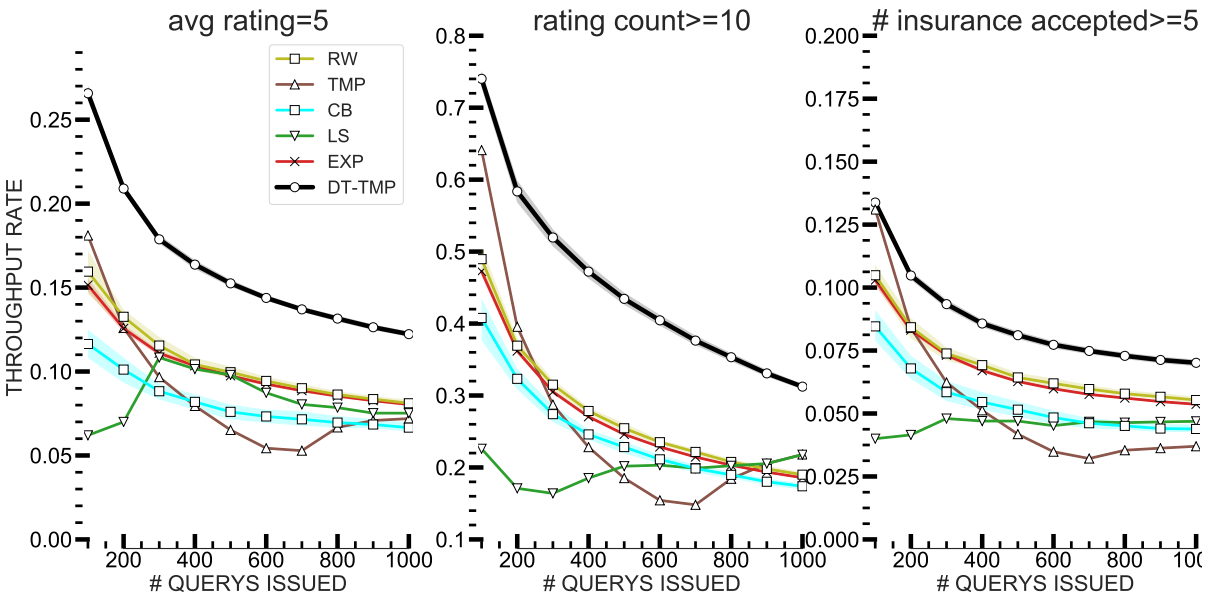


Figure 4.7: Throughput rate for different sampling strategies in RateMD. Combinatorial MAB (`DT-TMP`) does the best.

performance of `DT-TMP` in Figure 4.7 across different tasks. It outperforms the competition by a margin of 55.8%, 64%, and 25.7% over the three different tasks. Overall the throughput rate falls with the sampling budget. Since the number of doctors is limited, the rate at which newer target doctors are found decreases over time.

GitHub GitHub is the most popular open-source version control system that allows individuals to manage and collaborate on software-related projects. As of April 2017, Github reported an estimate of 20 million users and 57 million repositories [152]. Unlike Twitter, GitHub allows a number of user attributes to query for users. We use three queryable attributes. For the first queryable attribute, we use the ten most popular programming languages used on GitHub [4] to search for users using these languages in their projects. For the second queryable attribute, we discretize the "number of followers" attribute into three queryable ranges: ≤ 10, > 100, and otherwise. For the third queryable attribute, we discretize the number of repositories of a user into two queryable ranges: ≤ 10 and > 10. Similar to Twitter, Github returns by default a set of 20 users for each query and 100 users at maximum. Furthermore, for a given query, GitHub API returns a maximum possible result of 1000. We use the default ranking of API to get the results of a query.

For the first task, we consider GitHub users whose nationality is China as the first hidden

[4] `http://githut.info/`

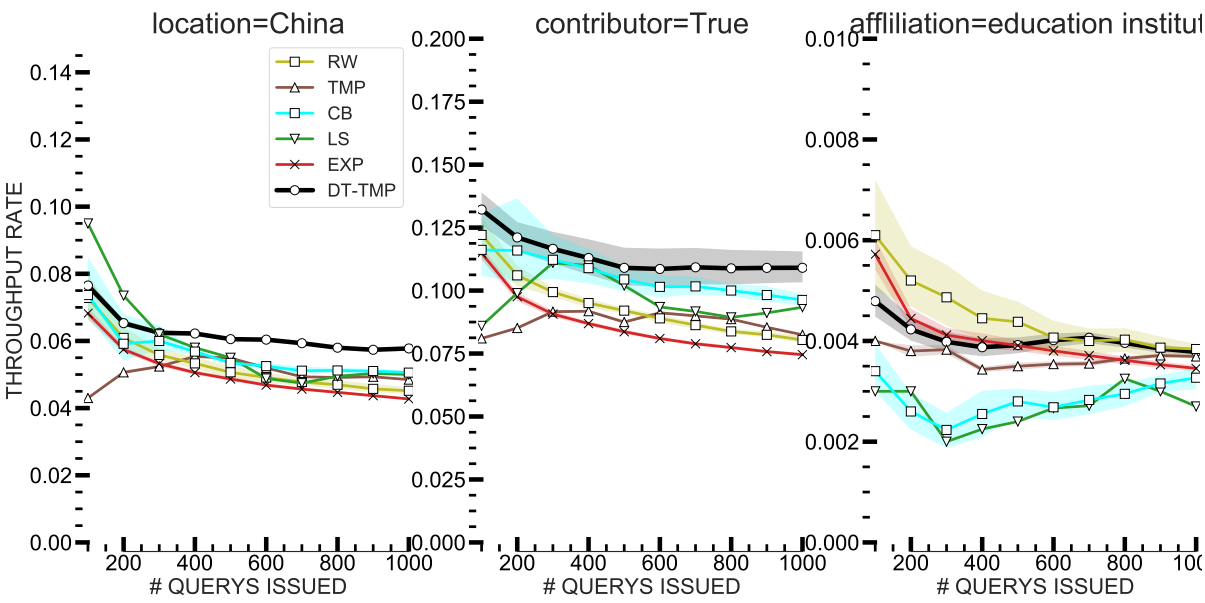


Figure 4.8: Throughput rate for different sampling strategies in Github. Combinatorial MAB (`DT-TMP`) does the best.

target population. For identifying Chinese users, we employ the location information in the users' profiles to predict their nationality. For the next two tasks, we set hidden target population to committed GitHub users (users who contributed to projects on more than 50% of days in the year 2017) and users who work or study at educational institutions (inferred by their email address). The experiments are conducted on the baseline and the proposed samplers; the results are evaluated using the same metric as the aforementioned online experiments.

Figure 4.8 reveals that `DT-TMP` is the best sampling strategy. It is statistically the best sampler for the first two tasks at a confidence interval of 95%. For the last task, since the overall number of accounts declaring education affiliation is very low, the feedback is very weak, leading to similar poor performance across all tasks.

4.4 DISCUSSION

In this section, we discuss why combinatorial querying works for finding hidden populations in Section 4.4.1. Next, in Section 4.4.2, we discuss the implications of `DT-TMP` sampling in real-world scenarios by showing the impact of various parameters on sampler's performance. Finally, in Section 4.4.3, we discuss the limitations of our work.

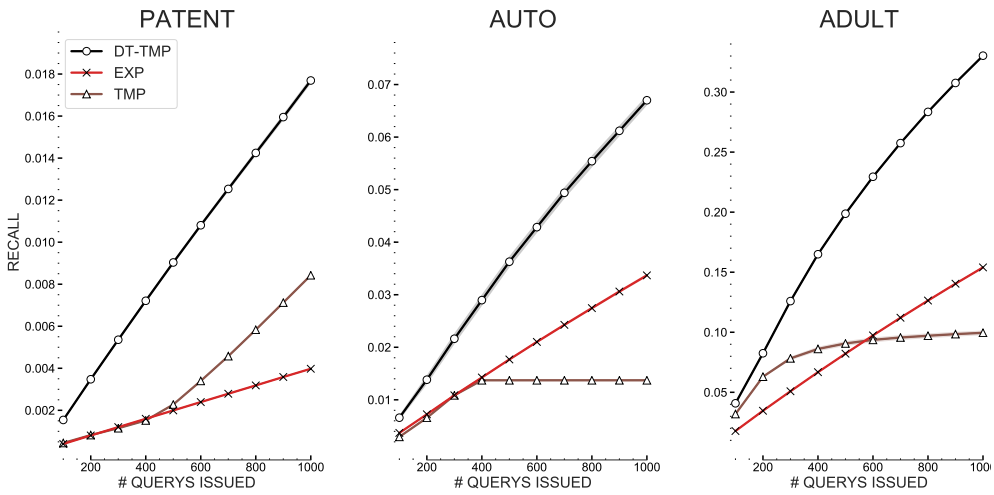


Figure 4.9: Comparing the recall value of non-combinatorial query based baseline samplers (`TMP`, `EXP`) and combinatorial sampler (`DT-TMP`) in real-world datasets. Combinatorial sampler outperforms all non-combinatorial sampler by 48.88% (AUC measure) average overall datasets.

4.4.1 Why does combinatorial querying work?

We observe that querying online APIs via a combination of one or more attributes leads to higher coverage of hidden target population than querying via individual attribute at a time. In a non-combinatorial querying system, only one attribute is used to define a query. Therefore, each queryable attribute A_i contributes d_i (A_i's cardinality) different queries to the non-combinatorial querying space. In contrast, a combinatorial query space is defined by the conjunction of one or more queryable attributes. The size of combinatorial query space is exponential. One of the advantages of a non-combinatorial querying system over a combinatorial querying system is its limited size of query space. This facilitates existing reinforcement learners to efficiently explore-exploit high-yielding queries in non-combinatorial query systems. However, we shall show via `DT-TMP` sampler that correlation between attributes and hierarchical structure within combinatorial query space can be exploited to design even more efficient sampling strategies.

Furthermore, using a specific sampling scenario, we prove the advantage of using a combinatorial query system over a non-combinatorial query system.

Lemma 4.3. For a sufficiently large dataset and a query system defined over a uniformly distributed attribute, `DT-TMP` requires a fewer number of queries to find the best query when the universal query ('*') is available than `TMP`.

Proof. Lets consider a query system defined over a uniformly distributed attribute with

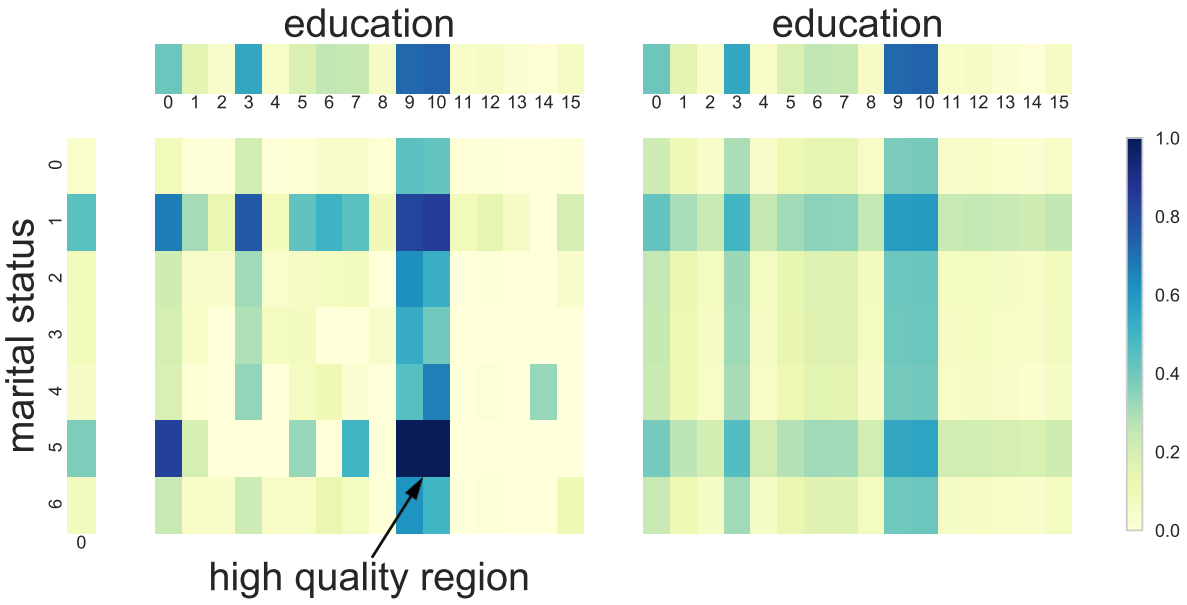


Figure 4.10: Comparison between combinatorial queries and disjoint combination of queries shown via heat-map of two attributes, "marital status" and "education", in Adult dataset. Darker shade represents higher quality arms.

$1, 2, \ldots k$ attribute values. Further, consider that m samples are returned in a single result page for each API call. Further, assume that only m' samples from each attribute value are needed to discern the best attribute value at a given confidence interval δ [153].

Thus, a naive query system would require $\lceil \frac{m'}{m} \rceil$ API calls corresponding to each attribute value to figure out the best query. Thus the total number of API calls issued by the non-combinatorial query system is $k\lceil \frac{m'}{m} \rceil$.

However, consider another query system that allows '*' query. In other words, for each API call, we obtain m samples that are drawn from any of the k attribute values. Given that the attribute is uniformly distributed, only $\lceil k\frac{m'}{m} \rceil$ expected number of API calls are required to obtain m' samples for each attribute value.

Since, $\lceil k\frac{m'}{m} \rceil < k\lceil \frac{m'}{m} \rceil$, therefore a combinatorial query requires fewer number of queries in expectation than a non-combinatorial query system. QED.

Figure 4.9 shows our proposed `DT-TMP` sampler that uses combinatorial querying system outperforms all non-combinatorial based samplers. Owing to the efficient utilization of hierarchy in query space, `DT-TMP` outperforms its competition. At a query budget of 1000, `DT-TMP` outperforms the best non-combinatorial query sampler, `TMP`, by margin of 112.75%, 23.25% and 10.64% on the datasets—Patent, Auto and Adult respectively.

Figure 4.10 shows that when arms of two attributes are combined to generate a new query, it performs better than the two corresponding arms that are queried disjointly. The

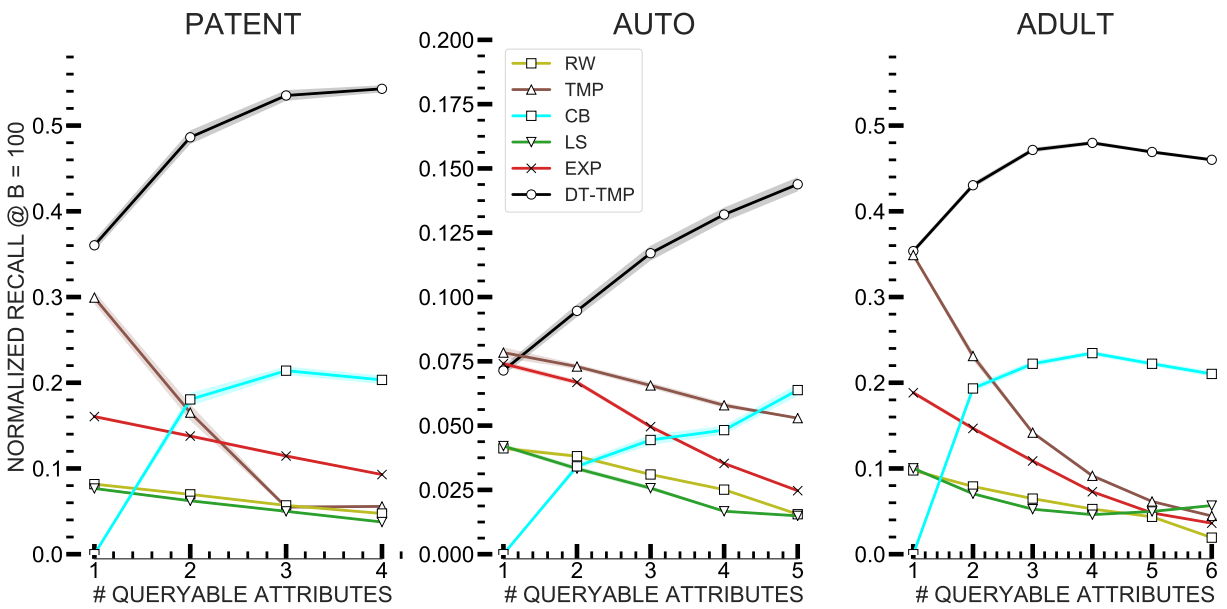


Figure 4.11: Recall value of different samplers at budget of 100 API calls. As the number of queryable attributes increases, `DT-TMP`'s performance increases due to it's flexibility in exploring over large query space. Large number of attributes creates exponential more arms to explore for naive MAB samplers, thereby causing a drop in their performance.

leftmost and the topmost sliders show the quality of arms of individual attributes. The left matrix is the combinatorial queries of the two attributes: "education" and "marital status." The right matrix is the combination of two arms that are queried disjointly, and their quality is measured via the average quality of the two corresponding arms. Observe that the real-world arm combinations help us explore very high-quality combinatorial arms (darker high-quality regions defined by setting "education" to 9 or 10 and marital status set to 5). Notice that such types of correlation between attributes help recover high-yielding arms by the decision tree. However, when the queryable attributes are independent, the decision tree works identical to a naive MAB sampler.

4.4.2 Digging Deeper: Factors Affecting Sampling

We now explore three prominent factors that impact hidden population sampling. This analysis will help users to be mindful of different factors that may affect their hidden population sampling.

Effect of Page Size Page size is directly proportional to the performance of a sampler. A higher page size means a larger number of samples obtained in every API call. More samples

lead to higher recall values. Table 4.3 depicts the percentage improvement in recall value as a consequence of increasing the page size across various samplers. We observe that recall of DT-TMP is better than baseline samplers by a significant factor. Furthermore, we note DT-TMP has the second-best improvement in recall value. UNI has the best improvement in recall value as the page size increases. This is due to the fact that UNI avoids the over-sampling issue, as discussed later. However, it should be noted that UNI is typically not possible in practice since it is not supported by APIs of most real-world data sources such as Facebook, Twitter, and LinkedIn.

For MAB samplers, we observe that increasing page size from 5 to 10 leads to *diminishing returns*, which is used to refer to the phenomenon that the recall value increases by lesser than $\alpha\%$ as the page size increase by $\alpha\%$. The diminishing returns take place due to two reasons. One, at high page size, generic queries are more likely to over-sample the same entities that are sampled by their specific queries. Two, we observe the non-uniform skewed distribution of the population over queryable attributes. Thus, non-uniform query sizes lead to the under-sampling of entities. Under-sampling of entities refers to the scenario when the query size is less than the page size, forcing the API to return less than page size results.

Table 4.3: Percentage improvement in recall value when page size increase from 5 to 10. UNI that samples uniformly over the database shows an expected improvement of nearly 100%. DT-TMP is the best performing sampler. However, we observe the effect of diminishing results. At very high page size, all samplers behave similarly.

	Patent		*Auto*		*Adult*	
Samplers	R_5	$\Delta R_{5\to10}\%$	R_5	$\Delta R_{5\to10}\%$	R_5	$\Delta R_{5\to10}\%$
UNI	1.39E-03	96.94	4.02E-04	93.39	5.43E-02	93.03
TMP	4.96E-04	89.24	5.02E-04	58.20	1.92E-02	37.08
EXP	7.50E-04	90.30	3.37E-04	49.85	1.69E-02	39.68
RW	6.13E-04	87.78	3.00E-04	22.95	1.04E-02	44.56
LS	5.20E-04	138.12	1.91E-04	72.36	1.43E-02	69.58
CB	2.17E-03	107.59	3.60E-04	147.54	7.89E-02	49.89
DT-TMP	5.34E-03	83.13	1.32E-03	60.47	1.22E-01	65.85

Effect of Number of Queryable Attributes We now explore the effect of the number of queryable attributes on the hidden target population's discoverability. We try different subsets of queryable attributes described in Table 4.2 for the offline datasets to observe this effect. We average the results of attribute combinations (subsets) where an equal number of queryable attributes are used. Figure 4.11 shows the efficacy of DT-TMP over baseline

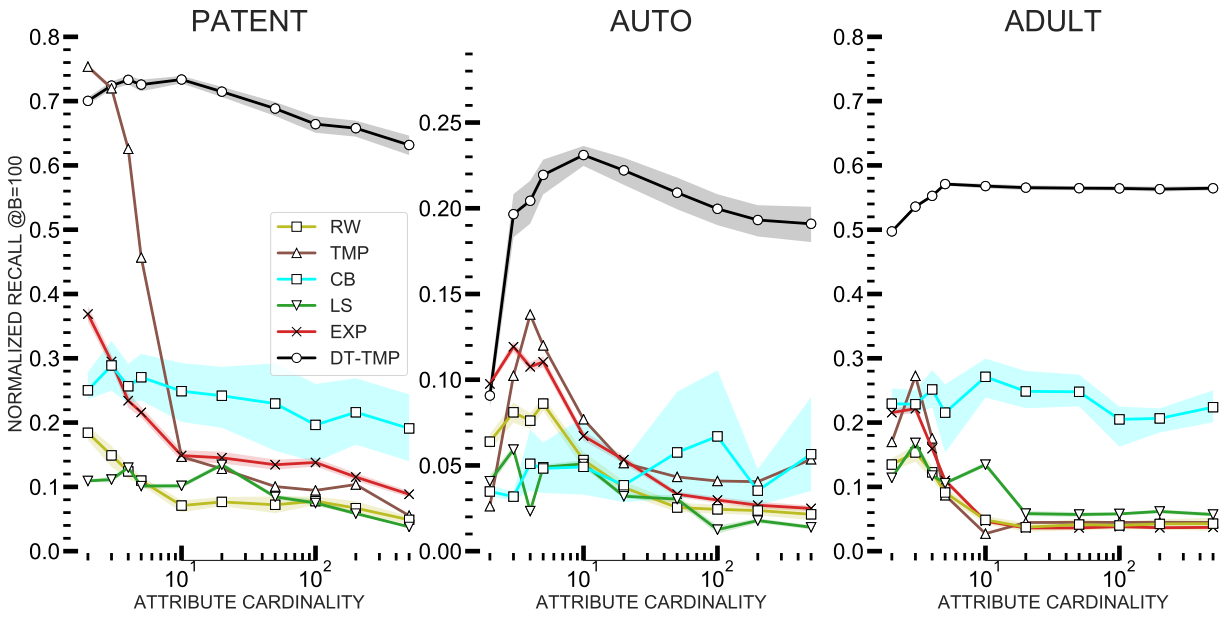


Figure 4.12: Recall value for different sampling strategies at budget of 100 API calls across variable attribute cardinality. Combinatorial MAB (`DT-TMP`) does increasingly better over large attribute space created by high attribute cardinalities.

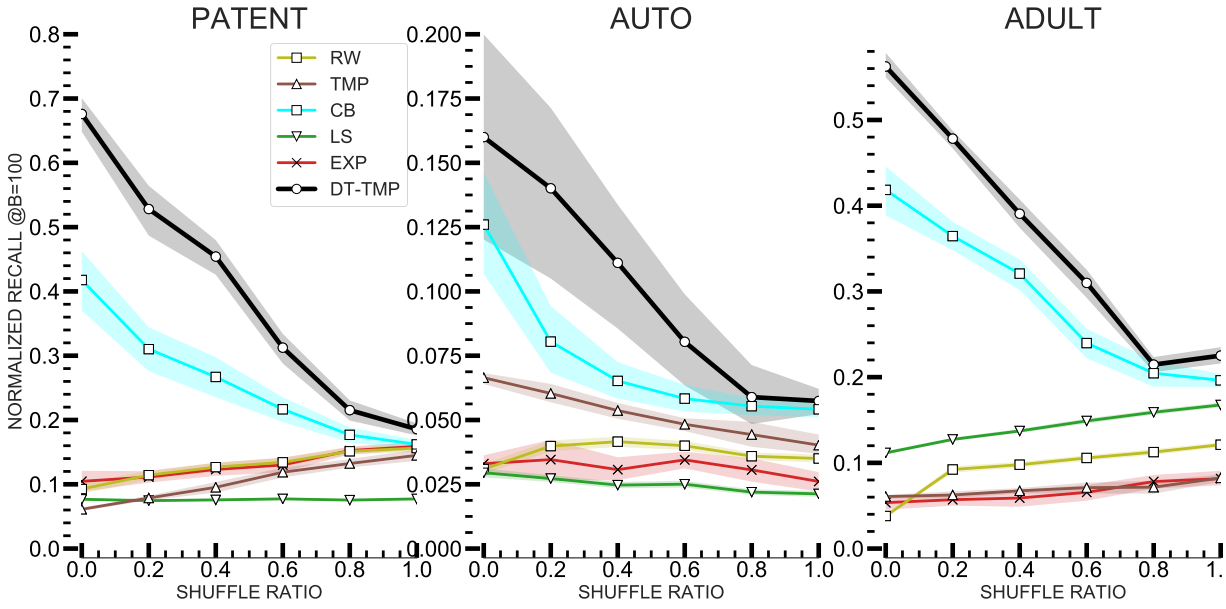


Figure 4.13: Recall value for different sampling strategies at budget of 100 API calls across different shuffle rates. The sampling performance falls with increasing shuffle rate. The performance depicts that `DT-TMP` is better at learning the correlation between hidden property and queryable attributes.

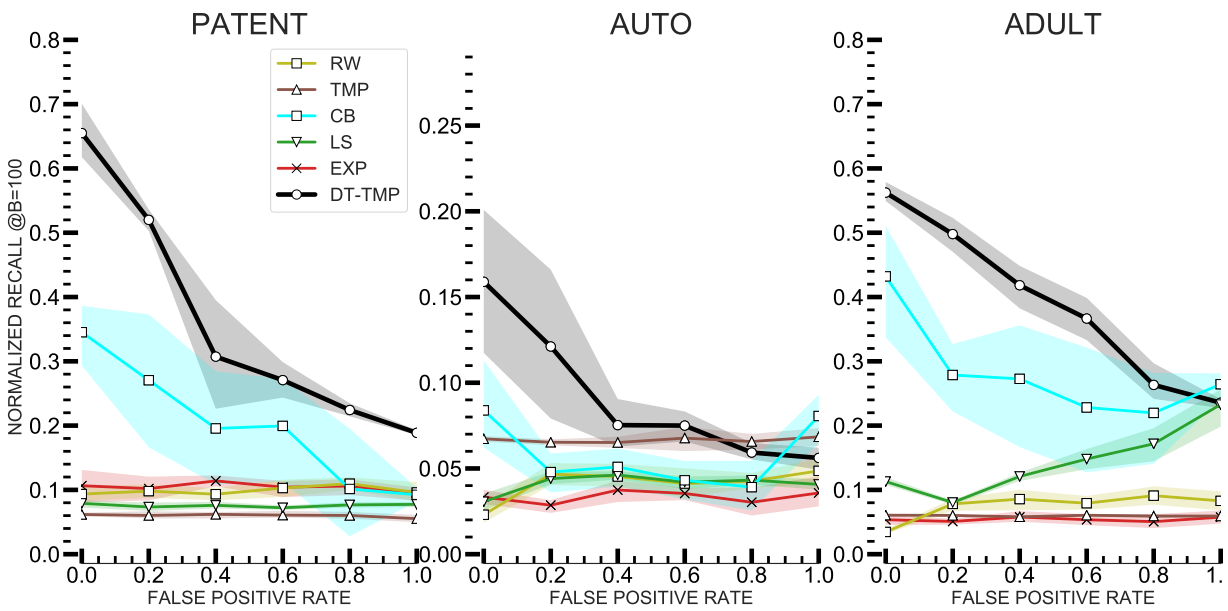


Figure 4.14: Recall value for different sampling strategies at budget of 100 API calls across different false positive classification error rates. Across all datasets, we observe that false positive errors have significantly higher impact on sampling performance than false negative rate. This difference is due to the fact that the target population is minority (below 50% of population size), and hence error in the identification of target population leads to loss in sampling performance.

samplers as the number of queryable attributes increases. Existing samplers suffer from high exploration space associated with a large number of queryable attributes. `DT-TMP` applies the decision tree-based search to preferentially explore high yielding queries in large query spaces. It, therefore, performs significantly better than baseline samplers. However, when the number of queryable attributes is one, it is observed in the "auto" subplot of Figure 4.11 that `DT-TMP` performs slightly worse than baseline samplers; for all other subplots, the samplers are statistically indistinguishable at a confidence interval of 95%. This happens because `DT-TMP` trades off the explore-exploit phase in typical MABs to greedily explore new attribute combinations. It, therefore, performs slightly poorly compared to baseline MAB samplers when the number of combinations is very few. Since UNI is independent of attributes, we do not include it in Figures 4.11 and 4.12.

Effect of Queryable Attributes' Cardinality Attribute cardinality controls the size of query space—higher attribute cardinality implies larger query space. We simulate the attribute cardinality in real-world datasets by modifying the attribute cardinalities in Table 4.2 to a fixed number, say c. For each attribute, we preserve the top-yielding $c-1$ attribute-

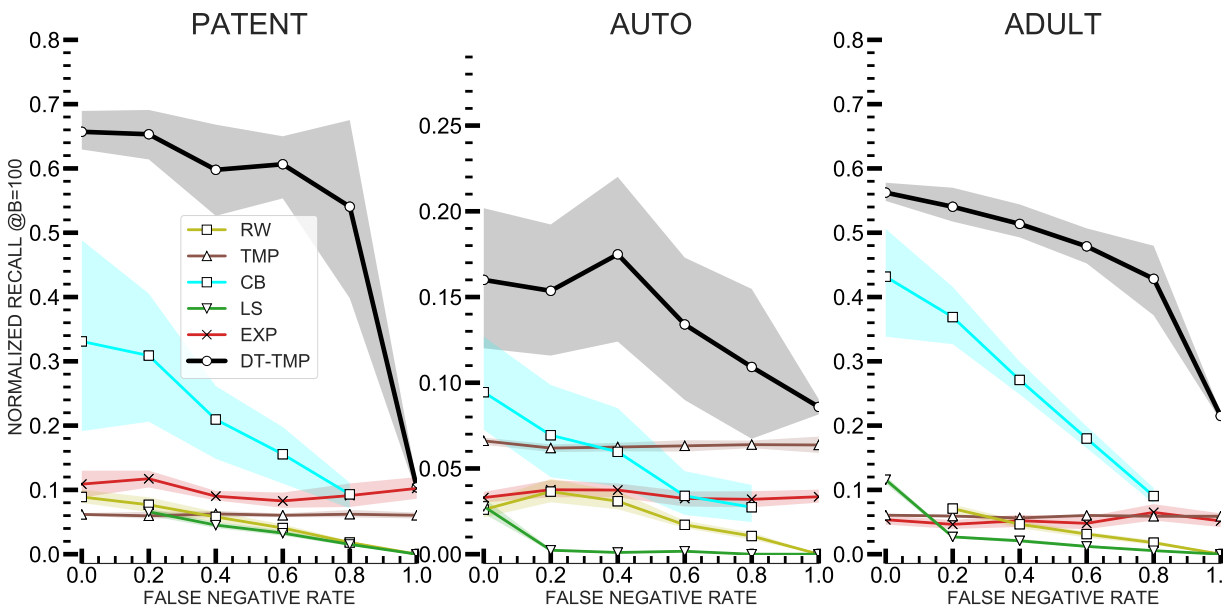


Figure 4.15: Recall value for different sampling strategies at budget of 100 API calls across different false negative classification error rates. When the target population size is above 50%, we observe false negative rate to be the more critical factor than false positive rate. Its effect is minimal on the sampling performance when the target population is minority.

values and merge the low-yielding remaining attribute-values into a new attribute-value. The merging preserves the discover-ability of hidden entities in the dataset over each attribute. We observe the effect of varying attribute cardinality in Figure 4.12. `DT-TMP` owing to its search tree strategy, is very effective at exploring and exploiting rewards distributed over large query space, which is induced by high cardinality attributes.

Effect of Correlation Between Queryable Attributes and the Hidden Property Correlation between the queryable attributes and the hidden property is one of the most important metrics. Highly correlated queryable attributes help form queries that yield a high number of hidden population entities and vice versa. We control the vary between queryable attributes and hidden attributes by randomly shuffling a fractional (shuffle ratio) subset of the dataset. Figure 4.13 shows that the performance of all samplers falls as the shuffle ratio increases. This happens because, at a high shuffle rate, the hidden population entities are uniformly distributed across different queries of the population, thus leading to poor recall values. The relatively higher performance of `DT-TMP` over the baseline samplers at an even higher shuffle ratio is on account of the fact that `DT-TMP` avoids re-sampling of entities.

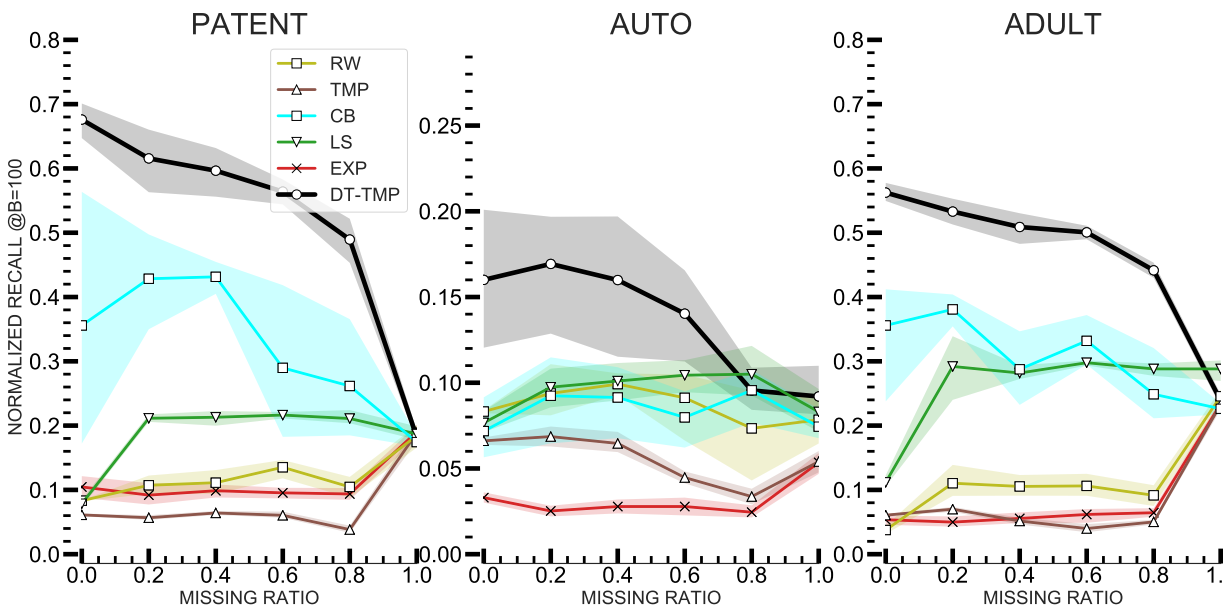


Figure 4.16: Recall value for different sampling strategies at budget of 100 API calls across different missing data rates. The sampling performance depreciates as the missing data increases. Interestingly, `DT-TMP` does really well even when the missing attributes are around 20-40% showing that it learns the correlation from the available sample efficiently. At very high missing information rates, all samplers tend to uniform sampling.

Effect of Classification Accuracy Classifier or the black-box identifier is one of the most critical components of sampling hidden populations from online social networks. An inaccurate classifier would be unable to identify the hidden entity as a hidden entity (false negative) or would identify a non-hidden entity as the hidden entity (false positive). We control the classifier accuracy by varying the false positive rate and the false-negative rate of the classifier by using an erroneous classifier in our offline experiments. Figures 4.14 and 4.15 show the effect of the classifier accuracy on the sampler's performance. As the false-positive rate and false-negative rate increase, it becomes increasingly difficult for the sampler to sample the ground truth hidden population due to erroneous feedback. When the classifier has a high false-positive rate, non-hidden entities are judged as hidden entities; thereby, the sampler over-estimates the quality of queries. Similarly, the samplers under-estimate the quality of queries when the false-negative rate is high. When choosing between two classifiers, we observe that classifier choice should be made in accordance with the estimated fraction of the target population. When the target population is a minority, we should prefer a classifier having a lesser false positive rate (type I) error. Conversely, we should prefer a classifier with a lesser false-negative rate (type II) error when the target population is in the majority.

Effect of Missing Data Missing and noisy data are common in online social networks. Often entities and users in online social platforms share only partial information due to privacy and security concerns. It is therefore important for the samplers to be impervious to the effect of missing data. We control the missing information in our datasets by removing the attributes from a fractional subset ((missing rate) of the dataset. Figure 4.13 shows that the performance of all samplers falls as the missing rate increases. This happens because a high missing rate makes it difficult for the samplers to distinguish the quality of different queries. At a very high missing rate above 80%, all samplers tend to behave similarly as a uniform sampler.

4.4.3 Limitations

Now, we discuss four limitations of this work. First, our algorithms are agnostic to the ranking function and, therefore, the ordering of the results. However, when the ranking function is correlated with the hidden target attribute, agnostic samplers may not perform well. We note that non-stationary reinforcement learners should be used to handle the effect of non-stationary rewards induced by unknown ranking functions. Second, our algorithms rely upon a classifier for identifying the target population when the hidden target attribute is implicit. Our future work is to discern the effect of classifier(s) in hidden population sampling. Third, our MABs handle continuous and infinite valued attributes by discretizing the attribute to ensure that there are finite arms of MAB. Third, much of our work assumes content to be static. In the future, it will therefore be useful to modify the `DT-TMP` sampling strategy to handle streaming datasets.

4.5 CONCLUSION

In this chapter, we proposed a novel algorithm for sampling hidden target populations from online social networks. Next, we showed that sampling individuals from hidden populations is hard due to API rate limits, limited access methods (or the limited number of queryable attributes), and combinatorial query space to search from. To address these challenges, we modeled the sampling problem as a Multi-Armed Bandit problem. Thereafter, we proposed a state-aware `DT-TMP` that exploited structure in combinatorial query space to discover high yielding queries. We showed that our proposed sampler is better than the competing samplers by a factor of 0.9-1.5× on offline, real-world datasets where query size is returned by the API. Our samplers out-performed baseline samplers on online social platforms by a margin of 54% on Twitter hidden population tasks and 49% on RateMD experiments. Together

in the previous and this chapter, we considered the sampling of observable and hidden attributes from social networks. However, attributed network comprises both the attribute and network structure. We shall focus on the third aspect of an online social network, i.e. network structure sampling, in the next chapter.

CHAPTER 5: LEARNING NETWORK SAMPLERS

In this chapter, we present our sampling work for sampling the network (or graph) of a social network. Social networks like Twitter can comprise several types of networks arising due to different interactions among the users, such as follower-followee interaction, re-tweet interaction, and reply interaction. Several downstream applications are interested in sampling information about these networks, such as Twitter's re-tweet network for early disaster warning [10], estimating Twitter users' characteristics based on follower-followee network [154], and discovering social circles in friendship networks [155]. The success of these downstream applications relies upon an efficient sampling of the target social network.

To address the above interest in network sampling, we propose a novel data-driven learning approach to sample graphs. In contrast to the prior works that aimed to develop a universal network samplers [13, 32, 33], we show the non-existence of a universal network sampler. By conducting an extensive empirical study and using theoretic examples, we show that trade-offs in some graph topological properties can cause different samplers to be more effective on different tasks and graph structures. To alleviate the non-existence of universal graph sampler, we propose a reinforcement learning framework that would allow researchers to design new samplers based on the need (or *context*), which is defined as the user-defined *task* and the *graph family*. We show both empirically and theoretically that our learned network sampler, `GTS` (graph family & task aware sampler) is more effective than the state-of-the-art network samplers. Our proposed sampler outperforms the existing samplers by up to 3× over a suite of ten different graph families and seven diverse tasks.

The rest of the chapter is organized as follows. In the next section, we provide a high-level overview of the network sampling problem. We present the challenges involved with network sampling, the limitations of the current sampling approaches, an insight into our proposed sampler, and finally, a summary of contributions. Then, in Section 5.2, we describe in detail our proposed learning framework that learns `GTS` sampler. In Section 5.3, we compare our proposed sampler with several state-of-the-art samplers on a suite of ten graph datasets over four seven network tasks. In Section 5.4, we present a discussion of issues raised in this chapter. We use the idea of adaptive graph samplers to learn adaptive brain parcellations of brain networks in Section 5.5. Finally, we present the conclusions of this chapter in Section 5.6.

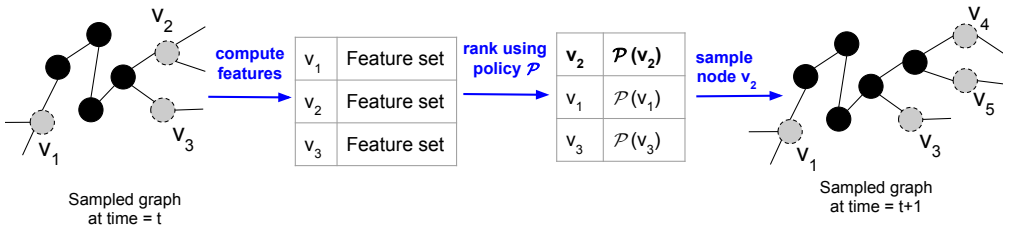


Figure 5.1: Figure shows a sampling policy $\mathcal{P}$ used for sampling nodes from a graph. The sampled nodes are colored black, and the frontier nodes (that are neighbors of the sampled nodes) are colored gray. At every step, $\mathcal{P}$ chooses one node from the frontier nodes to add to the sample.

5.1 OVERVIEW

In this section, we provide a high-level overview of the graph sampling problem and the limitations of the current sampling approaches. To address the limitations, we propose a learning framework to learn network samplers in Section 5.1.1. Next, we present our insights to address the challenges in learning graph sampler in Section 5.1.2, followed by a summary of contributions in Section 5.1.3.

5.1.1 Why learn network samplers?

Identifying a representative graph (or network) from a larger graph is a fundamental pre-processing task of interest across disciplines, including computer science [30], medicine [24], ecology [25], and the social sciences [17]. These representative sampled graphs find use in a wide range of tasks, including inferring behavioral characteristics of a node [156], identifying hidden populations [17] and diagnosing whether an individual has Alzheimer's disease [24]. Furthermore, not only do the graphs from which to sample vary in topology, but also in access cost—for example, some graphs like Twitter and Facebook disallow random access to nodes and edges, or social network APIs may set rate limits. With the wide range of tasks and graph topologies, we ask two fundamental questions in this chapter: ***Q1:*** *Can we design a universal graph sampler that one can apply in these different scenarios?* ***Q2:*** *If one cannot, what are the next best principled options?*

Two broad strategies are in use to address the problem of representative sampling. The first strategy is to sample graphs to support a *specific* task. Examples include sampling sub-graphs from a specific topic [29] or from a specific target population using a biased sampler. E.g., focused crawlers [18] sample target web-sites from web-graphs, and respondent-driven sampling [17] samples hidden populations. For tasks that require preserving node label distributions, we could use MHRW [30], or we can de-bias snowball samples [31]. A major

drawback of this strategy is that the handcrafting of task-specific samplers requires considerable expert time and effort, making this option difficult to scale through numerous tasks and graph topologies.

The second strategy is to design a *universal* graph sampler that preserves in the sampled graph all properties of the original graph. The motivation is that if one could preserve the original graph's properties in the sample, we could decouple the graph sampling technique from the downstream data-analysis task, and thus the sampler would be *universal*. For example, Forest Fire sampler [32] preserves well, degree, clustering coefficient, and hop distributions in citation and internet networks, Expansion sampler [13] is efficient at discovering community structures, and Rank Degree sampler [33] is effective for sampling centrality measures in social networks.

We answer the first research question *Q1* by showing an impossibility result. Through an extensive empirical survey and theoretical analysis (Section 5.4.1), we show an important impossibility result: **it is impossible to design a universal graph sampler**. An intuitive interpretation of this result is to think of different graph samplers as sampling policies with specific biases—they need the right context in which their bias is an asset but work less well in other contexts. For example, Forest Fire and Rank Degree samplers, owing to their bias towards high-degree nodes, are well-suited to scale-free networks [157] but fail to generalize to stochastic block models [158] and grid-like traffic networks.

We address the second research question *Q2* by proposing a principled framework (Section 5.2) to learn a graph sampler given an application context. The *framework is universal*–we can learn the sampling policy for any *specified* application context. In this work, the application context comprises the user-defined task (via an objective function) and the topological structure (i.e., few example graphs from the application context).

Our framework thus strikes a balance between two extremes: a) handcrafting an application-specific (or task-specific) graph sampler; b) using an existing universal graph sampler, which may potentially be sub-optimal for an application. For example, consider an applied data-scientist interested in understanding the social influence of her corporation's product on Twitter; her goal would be to identify a Twitter subgraph that preserves the social influence of nodes, say, degree centrality. Therefore, she would use our framework to learn a graph sampling policy suited for her target task (i.e., influence preservation specified via an objective function) and the target graph family (i.e., few example graphs from Twitter). Similarly, any other expert can specify a target task and target graph-family of their interest to learn a suitable graph sampler.

Based on the above inferences, we now present a formal definition of the problem of a

learning graph sampler. Given a user-specified graph family[1] and objective function, we seek a graph sampler that maximizes the objective function on the graph family. E.g., the data-scientist from the above example could input 'Twitter networks' as her target graph family and 'degree preservation' as her target objective function and learn a high-quality graph sampler using our proposed framework. She can then use the learned sampler for sampling her graphs.

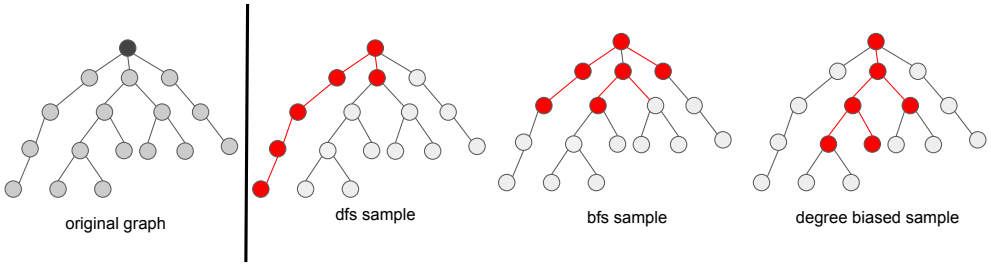


Figure 5.2: Different sampling strategies lead to graph samples capturing different topological properties of the graph.

An example of the learned sampler is demonstrated in Figure 5.2: on a 'tree' graph-family, the framework would pick the breadth-first sampling policy or the depth-first sampling policy depending on whether the user's objective is to preserve the branching factor or the depth of trees.

5.1.2 Challenges in learning a network sampler

Learning an optimal sampling policy is a challenging problem. Firstly, **the non-linear and discrete structure** of graphs makes traditional optimization functions like gradient descent un-suitable for learning an optimal graph sampler. Secondly, graph sampling is a well-studied NP-hard problem [54], where a particular sampling policy selects a subset of network nodes. Therefore, the learning framework has to determine the optimal sampling policy from a **combinatorially-large policy space**.

We use two key insights to address the challenges involved with learning an optimal sampling policy. First, we transform the graph sampling problem into a sequence prediction problem (of nodes), as shown in Figure 5.1. This transformation allows us to search for the optimal sampling policy in a continuous policy space rather than a discrete policy space. Second, we exploit the feedback from the user-defined objective function to efficiently explore and identify high-quality policies from the exponentially large policy space.

[1] a collection of example graphs sharing similar topological organization, e.g., Twitter graphs, proteins graphs, web graphs, etc.

We use these insights to propose a reinforcement learning framework that identifies high-quality sampling policies using a data-aggregation technique of imitation learning [121]. It learns a graph family and task aware sampler: `GTS`. Specifically, the learner begins with a random sampling policy. The learner executes the sampling policy on different graphs of the input graph family and thereafter updates its sampling policy w.r.t. the objective function feedback. The learner systematically iterates through different sampling policies and converges to a fixed sampling policy, `GTS`.

5.1.3 Contributions

We now summarize our contributions as follows,

Absence of universal sampler: We show through a simple theoretical proof and an exhaustive empirical survey (Section 5.4.1) that there exists no universal graph sampler that works independently of context. Even though few works [25, 159] have speculated the absence of a universal sampler, their experiments are either inconclusive or lack theoretical proof. We prove by contradiction that it is impossible for a sampler to *simultaneously* preserve two different properties for a stylized tree graph. In addition, our experiments on six different graph families and thirty-five different tasks show significant differences in sampling performance across contexts (graph family and task). The significance: for every problem context, we need a sampler tuned to that context.

Learning an adaptive sampler: To the best of our knowledge, we are the first to propose *learning of* context-aware samplers. Existing works [13, 32, 33] focus on general-purpose samplers, which in practice work well only in some limited contexts. In contrast, we propose a reinforcement learning algorithm that learns a high-quality sampling policy for any user-provided context. Our framework is significant: instead of handcrafting a sampling policy for every new context, our algorithm, by learning a context-dependent sampler, provides a unified way of solving multiple sampling-related problems, including scaling down internet graphs [160], visualizing social graphs [161], and conducting sociological surveys [17].

Robust empirical findings: We conducted extensive experiments across ten different graph families and seven diverse real-world tasks. The empirical results show the robustness of `GTS`, which outperforms the state-of-the-art samplers by a margin of 10% to 318%.

5.2 PROPOSED SAMPLING FRAMEWORK

In this section, we first formally define the supervised problem of learning graph samplers. Next, we describe any link-trace sampler as a decision-making agent in Section 5.2.2 followed by the challenges in learning the optimal agent in Section 5.2.3. Finally, we propose our proposed learner by addressing the challenges in Section 5.2.4.

5.2.1 Problem Statement

Consider a graph family $\mathcal{G}$ [2] and a user-defined task expressed through a quality function Q. A network sampler $\mathcal{S}$ samples a subset of nodes $\mathbb{S}$ from a graph $G(V, E) \in \mathcal{G}$ to form an induced subgraph $G_{\mathbb{S}}$ where $G_{\mathbb{S}} \subset G$. In this work, we focus on commonly used *link-trace* network samplers or crawlers [13, 30, 32, 33]. Given a sampling budget B and an initial seed node $v \in V$ which initializes the sample $\mathbb{S}$, a link trace sampler $\mathcal{S}$ adds nodes v to $\mathbb{S}$ such that there exists a node $w \in \mathbb{S}$ where $(w, v) \in E$. The sampler stops when the sampling budget is reached, i.e., $|\mathbb{S}| = B$. For this study, we consider undirected graphs that are connected.

The link-trace sampler $\mathcal{S}$ is evaluated for a specific task using a quality function Q that quantifies how good the sampled graph $G_{\mathbb{S}}$ is with respect to the original graph G. Therefore, an ideal sampler $\mathcal{S}$ maximizes the quality of the sampled graph or $Q(G, G_{\mathbb{S}})$.

We formally present the problem statement as:

> For a given graph family $\mathcal{G}$ and a given task $\mathcal{T}$; and a task-specific quality function Q, we want to find the sampler $\mathcal{S}^\star$ that maximizes the quality of the sampled graph. That is,
>
> $$\mathcal{S}^\star = \arg\max_{\mathcal{S}} \sum_{G \in \mathcal{G}} Q(G_{\mathbb{S}}, G) \tag{5.1}$$

In this section, we solve the sampling problem by first posing the link-trace sampler as a decision-making agent. Thereafter, we address the challenges involved in learning an optimal decision-making agent. Finally, we describe a simple and efficient reinforcement learning algorithm that learns high-quality sampling policies within a limited number of iterations.

[2] We assume the collection of example graphs from the application context exhibits distinct topological characteristics. Note that the topological structure also matters in developing the adaptive sampler. For instance, a data scientist could not simply train a sampler on one graph family, say protein networks, and apply it to another graph family (social networks) as the topological organizations of the graph families are totally different. More specifically, we exploit the valuable information available in graph repositories—Network repository [162] and SNAP [163] categorize hundreds of graphs from different disciplines, e.g., retweet graphs, protein interaction graphs, or web graphs—to draw example graphs.

5.2.2 Sampler as a Decision-Making Agent

We now describe a link-trace sampler, shown in Figure 5.1, as a decision-making agent. The sampling framework at any sampling instance comprises a sampled part of the underlying graph G as subgraph $G_{\mathbb{S}}$, where $\mathbb{S}$ is the set of sampled nodes. The sampler chooses a frontier node v_i from the set of frontier nodes $v_1, v_2, \ldots, v_n$ (un-sampled nodes that are neighbors of the sampled nodes) based on its current sampling policy $\mathcal{P}$. The sampler adds the chosen frontier node to the sample. We express the sampling policy $\mathcal{P}$ as a function that assigns a preference score to each frontier node with respect to the given task-based quality function Q and the graph family $\mathcal{G}$. Intuitively, this function attempts to assign the highest score to that frontier node that will maximally increase the quality of the sampled graph. For example, a depth-first sampling policy $\mathcal{P}$ in Figure 5.2 would assign a higher preference score to the frontier nodes that are many hops away from the seed node than to nodes that are few hops away from the seed node. Thus, at every sampling step, the sampler adds the frontier node $v^\star$ having the highest score,

$$v^\star = \arg\max_{v_i} \mathcal{P}(v_i | v_1, v_2, \ldots v_n, G_{\mathbb{S}}) \tag{5.2}$$

Alternatively, any link-trace sampler $\mathcal{S}$ can be interchangeably represented by decision-making agent having a preference function $\mathcal{P}$. For example, rank-degree [33] sampler's preference function is solely based on the frontier nodes' degree. Similarly, the breadth-first sampler's preference function assigns higher scores to frontier nodes at fewer hops away from the seed node. In summary, the preference function decides the direction of sampling and hence the quality of the sampled graph.

Learning an optimal sampling policy (or preference function) $\mathcal{P}$ is a non-trivial task. Firstly, the learner gets a single quality feedback after $B(\gg 1)$ frontier node selections (actions). In other words, the learner on executing a policy $\mathcal{P}$ for B steps gets a sampled graph $G_{\mathbb{S}}$ that subsequently gets evaluated for the task's quality function Q; hence one reward feedback. Additionally, the feedbacks are stochastic since we get different sampled graphs for different initial seed nodes, and the sampling policy $\mathcal{P}$ can be stochastic. Thus, *sparsity* and *stochasticity* of the feedback makes it hard for the learner to find optimal sampling policies efficiently. Secondly, it is not clear how to distinguish among the frontier nodes and choose the best frontier node to sample (action). Furthermore, once a frontier node is selected and added to the sample, the same action cannot be retaken. Thus, the *in-distinguishability* and *spontaneity* of actions make it difficult for the learner to identify optimal actions (and hence optimal policies). Thirdly, the learner encounters an exponentially-large state-space (intermediate sampled graphs) during sampling. For a sampling budget B on a graph of

N nodes, there are $\mathcal{O}(N^B)$ combinations of intermediate sampled graphs, and hence the exponentially-large state space. Furthermore, the intermediate sampled graphs are not independent of each other. The *interdependency* and *exponentially-large* state-space make the search for the optimal sampling policy even more challenging.

5.2.3 Learning a Sampling Policy

Now, we present approaches we used for tackling the above challenges involved in learning a high-quality sampling policy.

Short-sighted reward feedback: We address the problem of sparse reward feedback by evaluating a sampling policy $\mathcal{P}$ after every action rather than waiting for single feedback after B actions using a surrogate reward feedback system. The reward feedback of adding a frontier node v to the sample $\mathbb{S}$ is evaluated as the change in the quality of the sampled graph at the end of sampling budget B if the node v is added to the sample $\mathbb{S}$ at time t in contrast to the scenario that node v is not added to the sample $\mathbb{S}$ at time t. Thus, the reward R_v for adding a node v to sample $\mathbb{S}$ is,

$$R_v = Q(G, G_{\mathbb{S}\cup v, B-1-|\mathbb{S}|}) - Q(G, G_{\mathbb{S}, B-|\mathbb{S}|}) \tag{5.3}$$

$$\approx Q(G, G_{\mathbb{S}\cup v}) - Q(G, G_{\mathbb{S}}) \tag{5.4}$$

$$\approx \Delta Q_v(G, G_{\mathbb{S}}) \tag{5.5}$$

where $G_{x,y}$ stands for the sampled graph obtained by running the sampling policy $\mathcal{P}$ on a sampled node-set x for y steps. Hence, a high-quality sampling policy picks frontier nodes v that would lead to a high-reward feedback R_v, and vice versa. Later, we shall use the above reward feedback to design high-quality sampling policies.

Note that the above reward feedback analysis can also be viewed as a simplification of the counterfactuals. For computational efficiency, we approximate the effect of selecting node v by considering the immediate effect (short-sighted), as shown in Equation (5.5). We ignore the future effect of adding a node v due to high computational overhead. Surprisingly, in our experiments, we find that considering the immediate impact of adding node v is sufficient for learning high-quality samplers.

Anonymous action representation: We address the problem of selecting frontier nodes by representing the nodes via a set of features. Note that we are constrained to pick node features, such as the node's degree and node's time of discovery, which are anonymous [164] or independent of the graph. This anonymity constraint allows our sampling policy to be

universally applicable across all graphs of the input graph family. Further, this representation provides a compact way of representing $\mathcal{O}(|V|)$ unique frontier nodes in a low-dimensional feature space.

Data aggregation policy: We address the exponential state space by generalization principle [106] and the inter-dependency in sampled states by using a data-aggregation-based iterative policy learning algorithm. First, we use the sampled graph's topological features like the mean degree and average clustering coefficient to represent the state of the sampler. This generalization of state to a low-dimensional feature vector (smaller space) allows efficient learning of high-quality sampling policies. Second, we note that the states observed by the learner are *not* independent. The interdependence of states makes Markov Decision Process-based reinforcement learning algorithms [106] like Q-learning and Policy gradients highly inefficient. We adapt the idea from imitation learning, which has been shown to learn policies that perform well under an induced distribution of states [121]. Thus, our learner employs the data-aggregation principle to learn high-quality sampling policies.

Next, we combine the solutions presented above to propose a data-aggregation-based learning framework that learns high-quality samplers.

Algorithm 5.1 Offline learning of `GTS` Sampling Policy

Input: Graph family $\mathcal{G}$, task's quality function Q
Output: Sampling policy $\mathcal{P}^\star$

Initialize $\mathcal{D} \leftarrow \phi$
Initialize $\mathcal{P}_1$ randomly
for $i = 1, 2, \ldots 50$ **do** ▷ Training rounds or until convergence
    $G \leftarrow Draw(\mathcal{G})$
    $\mathbb{S} \leftarrow \phi$ ▷ Sampled node set
    for $j = 1, 2, \ldots B$ **do** ▷ Sampling budget
        $\mathcal{F} \leftarrow \mathbb{N}(\mathbb{S}) \setminus \mathbb{S}$ ▷ Frontier set
        $v^\star \leftarrow \arg\max_{v \in \mathcal{F}} \mathcal{P}_i(v)$
        $\mathcal{D} \leftarrow \{\mathbb{F}(v), \Delta Q_v(G, G_\mathbb{S})\}, \quad \forall v \in \mathcal{F}$
        $\mathbb{S} \leftarrow \mathbb{S} \cup \{v^\star\}$
    Re-train P_i on $\mathcal{D}$ ▷ using `lr`, `rr`, `svr`, `nr` in Section 5.2.4
return best P_i on validation

5.2.4 Proposed Learner

Before we dive into the details of our learner, we justify the key parameters used by the learner. We represent the frontier node and the state of sampled graph as a multi-dimensional

Algorithm 5.2 Online sampling by GTS

Input: Graph G, `GTS`'s sampling policy $\mathcal{P}$
Output: Sampled graph $G_{\mathbb{S}}$

1: $\mathbb{S} \leftarrow \phi$ ▷ Sampled node set
2: **for** $j = 1, 2, \ldots B$ **do** ▷ Sampling budget
3: $\mathcal{F} \leftarrow \mathbb{N}(\mathbb{S}) \setminus \mathbb{S}$ ▷ Frontier set
4: $v^\star \leftarrow \arg\max_{v \in \mathcal{F}} \mathcal{P}(v)$
5: $\mathbb{S} \leftarrow \mathbb{S} \cup \{v^\star\}$
return $G_{\mathbb{S}}$

feature vectors $[x_1, x_2 \ldots x_{10}]$ using feature transformation function $\mathbb{F}$, where x_i are the values of features f_i as shown in Table 5.1. Our feature selection process ensures that the features are generalizable across different graphs of the graph family. For computation reasons, we pick features that are incremental and can be updated in constant time. Interestingly, these constraints on the feature representation enable us to learn straightforward and efficient sampling policies that perform very well in real-world experiments. For simplicity, we use four standard regression algorithms—Linear Regression (`lr`), Ridge Regression (`rr`), Support Vector Regression (`svr`) and Neural Network with a single hidden layer (`nn`) [165]—to train the sampling policy $\mathcal{P}$ based on reward feedback in Equation (5.5). The regressors use the $[x_1, x_2 \ldots x_{10}]$ as input to predict the preference score of the frontier node as output. Our learning framework learns the best preference function or sampling policy $\mathcal{P}^\star$ by evaluating the different sampling policies on the validation set of graphs. The final learned sampling policy is named `GTS` (Graph family and Task aware Sampler).

Algorithm 5.1 describes the working of the learner that learns `GTS`. The learner simulates multiple episodes of sampling on the training graphs from the input graph family (line 4). The learner initializes the learning process with a random sampling policy (exploration). Afterward, each sampling run provides the learner with a set of reward feedback for each frontier node (line 9), which is then used by the regressor to train a new sampling policy $\mathcal{P}_i$ (line 11). The new sampling policy, in turn, samples new high-quality frontier nodes (line 8). To avoid the mistakes made in the previous runs, we aggregate past observations with new observations to learn a new updated policy (data-aggregation, line 9). We continue this cyclic process of policy updating and node sampling until the sampling policy converges or the training iterations run out. Given that running multiple iterations of a sampler is costly, we limit the training iteration count to 50 as often used in literature [121].

Finally, Algorithm 5.2 shows the working of the learned sampler `GTS` in the online scenario. In every sampling step, `GTS` samples the best frontier node according to its learned sampling

Table 5.1: Feature table.

		Description
Frontier Node	f_0	degree
	f_1	sample induced degree
	f_2	$f_1/\ f_0$
	f_3	max neighbor's timestamp
	f_4	#hops from seed
	f_5	#hops from last added node
	f_6	expansion degree
	f_7	random value $\in [0, 1]$
Graph	f_8	graph size
	f_9	mean degree
	f_{10}	average clustering coefficient

policy $\mathcal{P}$.

Complexity analysis: Online `GTS` sampler as described in Algorithm 5.2 is a quadratic time and linear space complexity algorithm. With each new node added to the sampled set, the frontier set expands by at most λ, the maximum degree of the graph (line 3). Further, the sampling policy of `GTS` takes constant time to evaluate every frontier node. Hence, the overall time complexity of the algorithm after sampling B nodes is $\mathcal{O}(\lambda B^2)$. Note that `GTS`'s complexity is similar to the complexity of some of the state-of-the-art samplers like `XS` [13]. Finally, the space required to maintain the frontier nodes and sampled graph, or the space complexity is $\mathcal{O}(\lambda B)$.

The offline learning complexity of the `GTS` sampler scales linearly with the number of training iterations with an additional cost of training the classifier after each iteration that is linear in the number of training points (or actions).

5.3 EXPERIMENTS

In this section, we evaluate our proposed sampling framework through the following research questions:

Q1: Effectiveness of proposed sampler: Can our proposed sampler, `GTS`, sample more efficiently than the state-of-the-art (SOTA) samplers? How does the performance of samplers vary across different graph families and tasks?

Q2: Adaptability of sampler: How does `GTS` sampling strategy change depending on the task and the graph family? Is the proposed sampler flexible to divergent tasks and graph

families?

Q3: Interpretability of proposed sampler and connection with existing samplers: What can we understand from the learned sampling policy? How does `GTS` differ from the baseline samplers?

Q4: Extensibility of the learner: Can `GTS` be extended to learn content samplers for attributed networks?

Q5: Parameter sensitivity of the learner: How do different parameters of the learner affect the learned `GTS` sampling policy?

5.3.1 Experimental setup

We now summarize the three essential components of our experiment design: datasets (graph family), tasks, and evaluation metrics.

Graph family We now describe the different graph families used in this work.

Erdos Renyi [166] (ER) generates a family of random graphs where each edge of the graph is drawn randomly from the Bernoulli process.

Watts Strogatz [70] generates random graphs that capture the small world phenomenon, including a high clustering coefficient and short average path lengths.

Stochastic Block Model [158] generates graphs with communities or subgraphs with high graph density. It captures the macroscopic property of the graph, like community structure and small-world phenomenon.

Lattice or mesh generates graphs that have regular patterns in the form of tiles.

Barabasi Alberta [167] generates scale-free networks where nodes exhibit power-law degree distribution. It famously captures the commonly observed phenomenon of the 'rich getting richer' effect in social networks.

Core Periphery generates graphs that have densely connected groups of nodes called core and a sparsely connected periphery. The graph structure of the world wide web is found to exhibit a core-periphery structure [168].

ADHD [169] is a collection of brain functional networks (nodes are the regions of the brain and edges the correlation between blood-oxygen levels) of the patients having Attention Deficit Hyperactivity Disorder (ADHD).

Regular is a collection of resting-state brain functional networks of regular people without any chronic disease.

BNU [170] is a collection of the structural brain networks from fifty subjects. We utilize the MRI-cloud datasets hosted on neurodata.io. We conduct our analysis on graphs con-

Table 5.2: Dataset statistics of several diverse graph families.

Graph-family	$\|\mathcal{G}\|$	$\|V\|$	$\|E\|$
Erdos	50	5K-10K	25K-51K
Watts	50	5K-10K	28K-58K
SBM	50	5K-10K	25K-51K
Lattice	50	5K-10K	30K-62K
Coperi	50	5K-10K	24K-50K
Barabasi	50	5K-10K	25K-49K
ADHD	20	413-444	3103-14522
Regular	20	419-444	3556-11872
BNU	47	990-1156	13643-21733
Retweet	8	2139-6288	2786-7977

structed from the BNU3 diffusion MRI data and pre-processed using pre-processed ndmg. The data repository consists of 47 subjects; for each subject, ndmg down-samples the voxel-wise graphs.

Retweet networks [162] are communication networks where nodes represent users, and edges represent retweets between users.

Note that we generate all synthetic networks at a fixed parameter of N ranging between 5K and 10k with a uniform probability, $k = 10$, $beta = 0.25$, and $lamda = 10$ using the Igraph package. The implementation shall be released with the code. Table 5.2 details the datasets' statistic.

Tasks We evaluate the efficacy of our proposed sampler, GTS, against the eight state-of-the-art or SOTA samplers (BFS: Breadth-first search, DFS: Depth-first search[13], RW: Random walk[30], MHRW: Metropolis Hastings RW[30], FS: Frontier sampling[171], FF: Forest Fire[32], XS: Expansion sampling[13], RD: Rank degree [33]) over seven diverse network tasks. In addition to the existing samplers, we consider two variants of GTS sampler—GREEDY (exploitation sampler which learns only from the best actions) and EXP (exploration sampler which samples frontier nodes randomly)—as baselines.

We evaluate the samplers by the extent to which they preserve different topological properties of the graph as listed in Table 5.3. The corresponding task for each of the listed property are: task T1 for degree, task T2 for influence, task T3 for clustering coefficient, task T4 for community labels, task T5 for average path-length, task T6 for network coverage, and task T7 for assortativity [15]. Finally, the quality function Q is defined as the inverse of distance D between the sampled graph property and the underlying graph property. For the distance metric, we use KS statistic for tasks T1, T3 and T5; mean absolute error for task

T7; and 1−coverage for tasks T2, T4 and T6. For example, for task T1, the quality function is $1/D$ where $D = max_k|F(k) - F'(k)|$, where k is over the range of the node degree, and F and F' are the cumulative distribution of node degrees in sampled graph $G_{\mathbb{S}}$ and original graph G respectively.

We now summarize the seven network tasks and their applications.

T1: **Degree similarity task**: Several real-world graphs such as internet networks, web-graphs, and social networks exhibit characteristic scale-free behavior [172]. Nonetheless, there is an abundance of non-scale-free networks like biological and technological networks that exhibit non-skewed degree distributions [157]. It is, therefore, desirable for samplers to preserve the degree characteristic for several studies. We use the Kolmogorov–Smirnov (KS) statistic to evaluate the degree similarity between the sampled graph's and the original graph's degree distributions.

T2: **Influence coverage task**: Sampling influential nodes of the network is a task of interest for several applications. Christakis and Fowler [173] employed network sampling to identify highly connected nodes to prevent disease outbreaks in social networks. Several areas, including viral marketing [174], expert finding [175], and malware propagation monitoring, require sampling of highly influential nodes of the networks. We evaluate the samplers based on the fraction of highly influential nodes (top 10% degree nodes) sampled for this task.

T3: **Clustering coefficient preservation task** Clustering coefficient is a widely used metric of a node [176] that captures how a node clusters together with other nodes in the network. Various tasks such as link recommendation [177], anomaly detection systems [178], and essential protein identification [156] apply clustering coefficient in their analysis. Therefore, preservation of the clustering coefficient is a pertinent issue for such tasks. We evaluate a sampler's efficacy at preserving the clustering coefficient using the KS statistic between the sampled and original graph's clustering coefficient distributions. We observed similar experimental results for preserving the average local and global clustering coefficient of the network.

T4: **Community discovery task** Real-world graphs such as social, biological, and chemical networks have characteristic modular structures or communities that correspond to the real social groups, functional groups, or similarity. Constructing samples from these diverse groups is an important goal for several applications. For example, marketing surveys seek to obtain stratified samples from different communities [179]. In this task, we evaluate the samplers based on the coverage of communities in the sampled graph as described in [13].

T5: **Average path length preservation task** "Six degrees of separation" or the small-world phenomenon [180] is a widely occurring phenomenon in social and web networks,

including Twitter and Facebook. Studies involving the design of efficient routing protocols [181] and understanding the transmission of diseases like HIV [182] rely upon the small-world property of the networks. Preserving the small-world property is, therefore, a salient feature of a sampler for such applications. For this task, we evaluate a sampler's efficacy by measuring the difference between the sampled and the original graph's average path length.

T6: **Graph exploration task** Landmark-based methods are a general class of algorithms to compute distance-based metrics in large networks quickly [183]. The basic idea is to select a small sample of nodes (i.e., the landmarks), compute offline the distances from these landmarks to every other node in the network, and use these pre-computed distances at runtime to approximate distances between pairs of nodes. As noted in [183], for this approach to be effective, landmarks should be selected so that they cover significant portions of the network. For such tasks, we evaluate a sampler's ability at graph exploration as the fraction of network reachable from the sample within one hop.

T6: **Graph exploration task** Landmark-based methods including web-search [184] and social search [185] seek to identify the landmark nodes in the dataset. The landmarks or point-of-interest nodes are important since they facilitate the shortest distance computation between nodes. Nodes use their inter-distances from the landmark nodes to compute an approximate estimate of their shortest path distance. In the context of graphs, a good landmark sampler preferentially samples the landmark nodes of the graphs or nodes that are easily reachable from other nodes in the graph. For this task, we evaluate a sampler's ability at landmark sampling or graph exploration as the fraction of original network nodes reachable from the sampled nodes within one hop.

T7: **Network relationship preservation task** Wagner et al [16] showed that sampling biases could cause a disproportionate representation of the relationship between population demographics. To ensure fairness of inference results, a sampler should preserve the node and group relationship accurately. For this task, we use the assortativity [15] metric to capture the relationship of a node with other nodes (degree correlation). The absolute difference of the assortativity value between the sampled graph and the original graph is used for evaluation.

Evaluation We evaluate our samplers at a fixed sampling budget of 10% of the graph size for 50 independent runs. We observed similar relative performance of the samplers at sampling budgets 15% and 20%. Lower distance values (mean absolute error, KS statistic, 1-coverage) indicate better sampling performance. Finally, we split the collection of graphs in the graph family into the training/validation/test set with a split ratio of 60/20/20. We use training graphs for learning different sampling policies and validation graphs for choosing

(a) Performance across graph family

(b) Performance across tasks

Figure 5.3: Averaged performance of samplers across different graph families (subplot a) and different tasks (subplot b). Lower distance values in each cell indicate better sampling performance. `GTS` outperforms all baseline samplers (shaded gray). * indicates `GTS` outperforms the sampler at 95% statistical confidence.

the best sampling policy `GTS`. For evaluation on test graphs, all samplers are initialized alike with the same seed set. Due to brevity of space, we skip the results of `FS` and `MHRW` as they perform very similar to `RW`.

5.3.2 Task Performance Results

Figure 5.3 shows the cumulative performance of different samplers (along the columns) across different graph families and different tasks, respectively. We observe that `GTS` outperforms (at significance level $\alpha = 0.05$) all baseline samplers by a margin of 54-75% on different graph families and by a margin 51-69% on different tasks.

Across different graph families, we observe GTS performs notably better than the SOTA samplers on graph families having non-skewed degree distribution such as Erdos, Watts and SBM. The existing samplers like FF and RD, which are biased towards high degree nodes, fare poorly on such graph families. Interestingly, we note a high variance in the performance of SOTA samplers across different graph families. For example, XS does significantly better on functional brain networks than structural brain networks. We note that GTS by adapting its sampling strategy according to the context achieves consistently high sampling performance and low-performance variability.

We observe a similar high variability in the performance of baseline samplers across different tasks T1 to T7. For instance, XS and RD samplers perform well on tasks such as graph exploration (T6) due to their exploration bias, but they significantly over-estimate average path-length due to their expansive nature. On the other hand, BFS works well in preserving the average path length (T5), but it does poorly when preserving the community structure (T4) due to its locality bias [51]. Further, GTS does almost as good as XS at the community discovery task (T4), which is known to be an optimal sampling strategy for this task [13]. Thus, GTS, by modifying its sampling policy according to the task, achieves superior sampling performance across all tasks.

Finally, we note that GREEDY sampler gets stuck in local, high-quality sampling policies. Under imitation learning [121], GTS maintains an optimal exploration-exploitation trade-off when searching for globally high-quality policies and avoids getting stuck in locally high-quality policies. The above results show the effectiveness of the proposed context-based sampler and thus helps answer *Q1*.

5.3.3 Adaptability Results

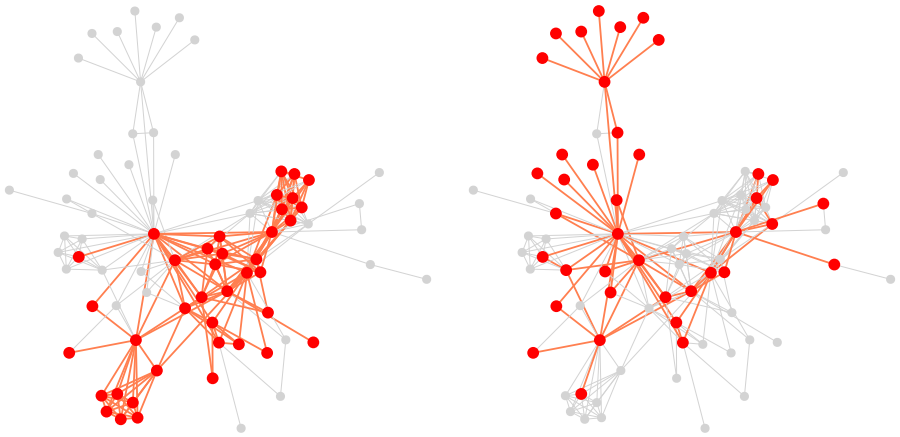
In this subsection, we answer question *Q2* by testing the adaptability of our learned sampler both qualitatively and quantitatively.

Figure 5.4 shows through the Les Miserables co-occurrence network [162] the adaptability of GTS sampler in preserving clustering coefficient CC of the graph (task T2) versus and the network coverage NC (task T6). When preserving the clustering coefficient in the graph (CC = 0.57), GTS preferentially samples the clustered nodes (triangles), as shown in subplot (a). On the other hand, GTS trade-offs clustered nodes with well-distributed nodes in the graph when sampling for network coverage, as shown in subplot(b). These graph visuals elucidate the flexibility of our proposed sampler.

Quantitatively, we observe how the GTS adapts according to the topological structure and task by observing the variation in its sampling performance value across different graph

Table 5.3: Description of graph properties.

Property	Definition	Formula
Degree	Number of neighboring nodes	$\lvert\mathcal{N}(v)\rvert$
Influence	Fraction of hub-nodes $\mathbb{H}$ in the sampled set $\mathbb{S}$ in one hop	$\frac{\lvert\mathbb{H}\cap\mathbb{S}\rvert}{\lvert\mathbb{H}\rvert}$
Clustering coefficient	Density of connection in the node neighborhood	$\frac{2\lvert E(G_{\mathcal{N}(v)}\rvert}{\lvert\mathcal{N}(v).\mathcal{N}(v)-1\rvert}$
Community discovery	Fraction of communities [13] in the sampled set	$\frac{\sum_{c\in\mathcal{C}}\mathbb{I}(c\in\mathbb{S})}{\lvert\mathcal{C}\rvert}$
Path length	Distance from node v to every other node in graph	$\frac{\sum_{u\in V} dist(v,u)}{\lvert V\rvert}$
Graph exploration	Fraction of nodes reachable from the sampled set $\mathbb{S}$ in one hop	$\frac{\lvert\mathbb{S}\cup\mathcal{N}(\mathbb{S})\rvert}{\lvert V\rvert}$
Assortativity	Correlation between nodes' degree connected by edge	$(u,v)\in E,\ \rho(d_u,d_v)$



(a) CC= 0.56, NC= 84% (b) CC= 0.34, NC= 100%

Figure 5.4: Sampling trade-off between clustering coefficient (CC) and network coverage (NR) in Les Miserables network.

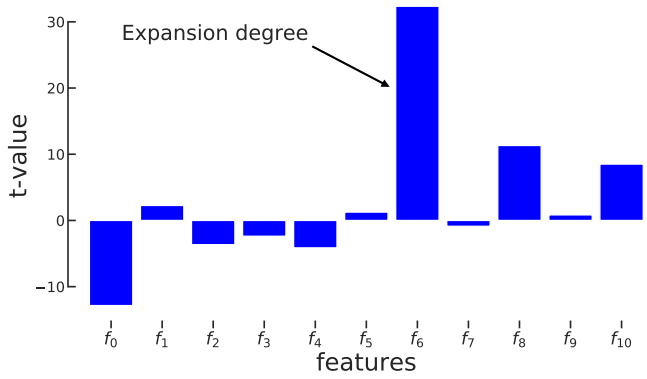


Figure 5.5: GTS preferentially samples nodes having high expansion degree for community coverage task (T4). Thus, it behaves as XS sampler.

Table 5.4: The baseline samplers can be treated as a special case of GTS. θ_i=0 for unmentioned features.

Sampler	Coefficients
BFS	$\theta_4 = -1$
DFS	$\theta_3 = 1, \theta_4 = 1$
RW	$\theta_3 \approx 1$
XS	$\theta_6 = 1$
RD	$\theta_0 = 1$

families and tasks. For example, for degree distribution task (T1), GTS adapts to BFS-like sampling strategy in Lattice and SBM graphs. This is because Lattice and SBM have local node connectivity pattern replicated across the entire graph; thus, BFS based local sampling strategy works well. On the other hand, GTS adapts to XS-like and FF-like sampling strategies in scale-free Barabasi and fMRI brain networks. Preferential sampling works best in scale-free networks since most of the edges are concentrated at few high degree nodes. Finally, GTS adapts to DFS-like sampling strategy in star-shaped graph families, e.g., Retweet. Detailed results for individual graph-family and task is available in Appendix(Appendix A.3).

5.3.4 Interpretability Results

We now answer *Q3* by showing that GTS is interpretable, and it can be understood as a generalization of the baseline samplers.

GTS's preference for different frontier nodes helps reveal the nature of GTS sampler. When using linear regression (lr) to learn GTS's preference function, the sampler picks the frontier node with highest preference score ($=\sum_i \theta_i x_i + c$), where θ_i is the coefficient and x_i is the

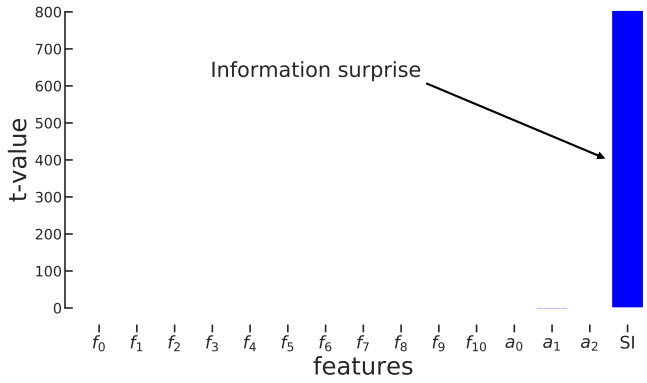


Figure 5.6: GTS preferentially samples nodes having high information surprise [123] for cluster coverage task in attributed networks. Thus, it behaves as SI sampler. The features used for picking the frontier nodes are structural features $f_1, f_2, \ldots, f_{10}$ (as listed in Table 5.1), the node attributes gender(a_1), locale (a_2), education (a_3), and node's information surprise as described in Equation (3.1) of Chapter 3.

value corresponding to feature f_i listed in Table 5.1. By setting different values of θ_i, GTS sampler can act as any of the baseline samplers. Table 5.4 shows the specific values of feature coefficients that causes GTS to act as one of the SOTA samplers. Note, that some baseline samplers like FF are combination of BFS and DFS; likewise FS and MHRW are variants of RW. In summary, GTS achieves superior sampling performances by varying the feature coefficients suitably for different graph families and tasks.

Now, we show how to interpret GTS sampler. In Figure 5.5, we show the bias of GTS towards different frontier nodes through the significance value (t-value) of the feature coefficients. As seen in the figure, GTS is biased towards frontier nodes with a high expansion degree when sampling for community structures in SBM graphs. In other words, GTS acts as XS sampler. Likewise, we find that GTS tends to behave as RW sampler in Barabasi networks for degree preservation tasks. Thus, our sampler's interpretability allows us to re-discover past network sampling theory [13, 159] through a data-driven approach.

5.3.5 Extensibility Results

In this subsection, we answer question *Q4* by showing that GTS can also be used to sample attributes from the attributed network.

In Figure 5.6, we show that GTS greedily picks frontier nodes with highest surprise (feature SI) defined as the content surprise of a frontier node in Equation (3.1) of Chapter 3 for attributed networks. By sampling the highly surprising nodes, the sampler preferentially samples the cluster shape, and boundary described in Section 3.4. In other words, GTS

acts as SI sampler for sampling content for cluster-coverage task in Facebook. Thus, our sampler's extensibility to attributed networks and for content tasks show that GTS is not only suitable for sampling network structure but also network content. In the future, we would like to extend GTS for more complex tasks, including node classification and node embedding.

5.3.6 Parameter Sensitivity

GTS learner has only one free parameter, which is the choice of regressor to train the preference function. We train the preference function using four regressors separately as described in Section 5.2.4.

The learner's ability to learn the optimal sampling policy (or the preference function) is affected by the regressor's policy space. When the policy space is large, the learner has to explore a larger policy space; hence the convergence time increases, and it is more likely for the learner to get stuck in local optima (overfitting). In Figure 5.7, we observe when using the svr model, the learner experiences overfitting, and it gets stuck in locally optimal sampling policies within a few training iterations. Further, we note that regularization can alleviate the problem of overfitting (see rr). Thus, the regressor's policy space helps determine the complexity of the learned sampling policy and thereby affects the sampling performance.

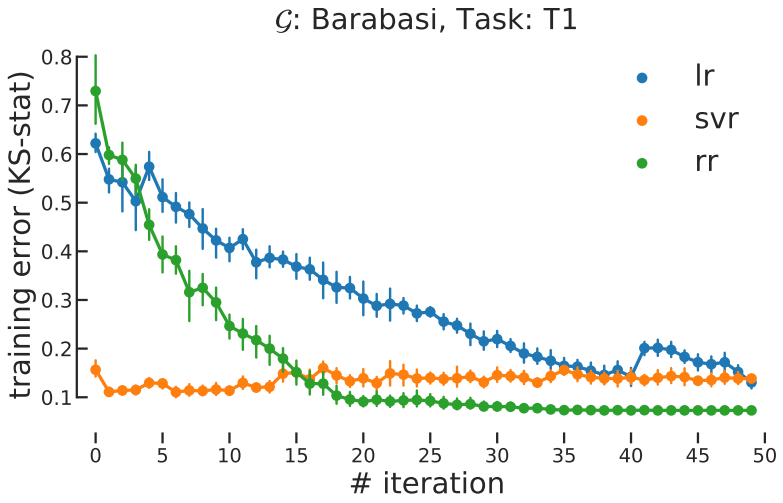


Figure 5.7: The figure shows the convergence of the learner (Algorithm 5.1) when using different classifiers (lr, rr, svr) to learn the preference function.

Table 5.5: Offline training time to learn GTS sampler on 10K-sized graphs. Tasks such as path-length take more time due to the high complexity of computing properties like path-length. Terms a and b in '$a(\pm b)$' are the mean and standard deviation values.

Task	Property	Time (minutes)
Task-T1	degree	20.78 ± 6.93
Task-T2	influence	21.86 ± 5.60
Task-T3	clustering	27.13 ± 7.84
Task-T4	community	28.13 ± 1.22
Task-T5	path-length	43.42 ± 4.41
Task-T6	coverage	30.76 ± 5.82
Task-T7	assortativity	36.83 ± 4.88

5.3.7 Time Complexity

As noted in Section 5.2, the learning sampling framework comprises offline training (Algorithm 5.1) and online testing (Algorithm 5.2). In this subsection, we compare both the offline time cost and the online time cost of GTS in comparison to the state-of-the-art samplers.

Offline training is unique to GTS since it is a learned sampling policy. The traditional samplers such as Forest Fire [32] and Expansion Sampler [13] were handcrafted by researchers. In contrast to the extensive time involved with designing a new sampler, we rely upon a few graph examples from the graph family and the user-defined quality function to adaptively learn the sampler. Furthermore, in practical applications, typically, users don't pay much attention to samplers' offline training costs since it is a one-time task and can be carried out on large-scale servers. Since it is not possible to objectively define the time required to handcraft a new sampler for a new problem, we report the offline empirical time taken to learn GTS sampler in Table 5.5.

We compare the empirical training time on 10K-sized synthetic graph-families (G1-G6), and at a 10% sampling budget, it takes around an hour to train GTS. We implemented all described algorithms in Python using the Igraph package [186]. All tests were performed on a computer with 16 GB RAM and a 3.2 GHz Intel Core i5 processor (4 cores).

For online testing, GTS as described in Algorithm 5.2 queries both the sampled set ($\mathbb{S}$) and the frontier set ($N(\mathbb{S})$) to decide the best frontier node, thus the total API calls require $\mathcal{O}$ ($|\mathbb{S} \cup N(\mathbb{S})|$) time, and is independent of the size of the graph. Likewise, recent samplers like expansion sampling (XS) and rank-degree (RD) that uses a policy to sample the best frontier node require an equivalent number of API calls as GTS. In contrast, naive samplers such as BFS, DFS and Rw use only the local graph exploration to add the frontier nodes to the sample; thus require $\mathcal{O}$ ($|\mathbb{S}|$) time. The online time for GTS and baselines for averaged

Table 5.6: Online time taken for different samplers on 10K sized graphs at 10% sampling budget. GTS takes more time than the baseline samplers due to the time taken to compute the preference of the frontier nodes in each iteration. Terms a and b in '$a(\pm b)$' are the mean and standard deviation values.

Sampler	Time (seconds)
BFS	4.11 ± 0.44
DFS	4.01 ± 0.50
RW	4.18 ± 0.52
FF	4.10 ± 0.48
XS	5.39 ± 1.64
RD	4.38 ± 0.47
GTS	57.29 ± 6.11

across 100 runs are reported in Table 5.6. Note, GTS takes more time than the baselines due to the time taken to compute the preference of the frontier nodes in each iteration (line 4 of Algorithm 5.2). In general, we empirically observe that GTS is only a constant factor $(10 - 15\times)$ slower than other baseline samplers for 10K-sized graphs.

5.4 DISCUSSION

In this section, we provide empirical and theoretical evidence suggesting the non-existence of a universal sampler. Next in Section 5.4.2, we discuss why simple extensions for learning optimal sampler don't work well in practice. Finally, in Section 5.4.3, we discuss the limitations and future extensions of this work.

5.4.1 Why not a Universal sampler?

In this section, we show the non-existence of a universal graph sampler in theory due to tradeoffs in topological properties. Next, we show the impact of topological properties' tradeoff on practical applications by conducting an extensive empirical survey.

Theorem 5.1. There exists no optimal sampler $\mathcal{S}$, which preserves all properties of a network better than all possible samplers $\mathcal{S}'$, such that $\mathcal{S}' \neq \mathcal{S}$.

Proof. We prove by contradiction using two tradeoffs: breadth vs. depth preservation in a stylized tree graph, and triangle count vs. exploration maximization in a stylized line graph with repeating triangles. We assume there is a universal sampler that can preserve *all* graph topological properties at the same time. Next, we show two special cases contradicting our

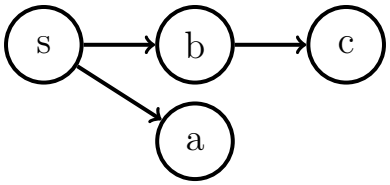


Figure 5.8: An example tree to show the sampler's trade-off in maintaining the breadth and depth of the tree.

assumption in Example 5.1 and Example 5.2. This leads to the conclusion that there is no universal sampler. QED.

Example 5.1. There exists no link-trace sampler $\mathcal{S}$ that can preserve both the depth and the breadth (or branching factor) of the 'tree' graph, as shown in fig. 5.8 at a sample size of 3. Further, assume that 's' is the seed node of any sampler.

Proof. Let us assume that there exists a universal sampler $\mathcal{S}^\star$ which preserves both the **depth** and **breadth** (or branching factor) of the tree graph for a sample size of 3.

As shown in Figure 5.8, seed node 's' as the starting node in the sampled set. Any link-trace sampler having a sampling budget of 3 can query two more nodes. Thus, any sampler can sample either $\{a, b\}$ or $\{b, c\}$ as the two other nodes. Notice that the former sampled set, i.e., $\{s, a, b\}$ has a branching factor of 2 while a depth of 1. On the other hand, the later sampled set, i.e., $\{s, b, c\}$, has a branching factor of 1 while a depth of 2. Thus, we observe any sampler can either preserve either the branching factor, i.e., 2, or the depth, i.e., 2, of the original tree graph.

The above argument shows by contradiction that there exists no universal sampler due to the tradeoff between breadth and depth of a tree graph.

QED.

We now show the tradeoff between maximizing two topological properties of a graph: **triangle** and **expansion**. The triangle maximization task tries to maximize the number of triangles $T(\mathbb{S})$ in the sampled graph formed by the sample set $\mathbb{S}$. Expansion $E(\mathbb{S})$ or graph coverage task [13] tries to maximize the number of nodes that can be reached from the sampled set in at most one hop, i.e. $|\mathbb{S} \cup N(\mathbb{S})|$. The triangle count is important for several real-world tasks such as motif-based community detection [187] and graph classification [188]. On the other hand, graph exploration tasks are relevant to tasks like outbreak detection [173] and landmark discovery [183].

Example 5.2. There exists no link-trace sampler $\mathcal{S}$ that can maximize both the triangle and the expansion property of a graph as shown in fig. 5.9 at a sample size of $\leq N/3$, where N is the size of the graph.

Proof. Let us assume that there exists a universal sampler $\mathcal{S}^\star$ that maximises the two topological properties: **expansion** and **triangle** on a stylized graph as shown in Figure 5.9, at the same time.

Consider, the nodes of the graph are partitioned into two sets U (bottom nodes) and V (top nodes) as labeled in the Figure 5.9.

For any sampled set $\mathbb{S}$ of size n ($\leq N/3$), assume the number of sampled nodes from partition V is n_V and the number of sampled nodes from partition U is n_U. Therefore,

$$n_U + n_V = n \tag{5.6}$$

Given that any link-trace sampler obtains *connected* samples, the induced sampled graph from the sampled nodes in U should form a straight line or line-graph. [3]

Ignoring the sampled graph endpoints [4], the number of triangles in $\mathbb{S}$ would be n_V, given that all nodes in U surrounding the sampled node in $\mathbb{S}_V$ has to be sampled for connectedness reason. That is,

$$T(\mathbb{S}) = n_V \tag{5.7}$$

Similarly, we can see that the number of un-sampled neighboring nodes of $\mathbb{S}$ (i.e., $N(\mathbb{S})\setminus\mathbb{S}$) is $\frac{n_U}{2} - n_V$. Therefore, the expansion of sample $\mathbb{S}$ will be $n + \frac{n_U}{2} - n_V$. That is,

$$E(\mathbb{S}) = n + \frac{n_U}{2} - n_V = \frac{3}{2}n_U \tag{5.8}$$

We can combine the above Equations (5.6) to (5.8) to show that the weighted sum of expansion $E(\mathbb{S})$ and number of triangles $T(\mathbb{S})$ is a constant,

$$2E(\mathbb{S}) + 3T(\mathbb{S}) = 3n \tag{5.9}$$

From the above constraint, we see that for any link-trace sampler can either maximize $E(\mathbb{S})$ or maximize $T(\mathbb{S})$. Thus, any sampler having high expansion will have a low triangle count and vice versa. Thus, there exists no link-trace sampler $\mathcal{S}$ that can maximize both the properties at the same time.

QED.

Empirical Survey: We evaluate the eight baseline samplers (`BFS`, `DFS`, `RW`, `MHRW`, `RW`, `FS`, `FF`, `XS`, `RD`) over the six synthetic graph families (`Erdos`, `Watts`, `SBM`, `Lattice`, `Coperi`,

[3] If the sampled nodes in U are not connected, it will lead to the sampled graph being disconnected; and hence contradict the condition that the sampled graph should be connected.

[4] We ignore the two (left and right end-point) conditions of the sample, since it will have a small insignificant change in the value of $T(\mathbb{S})$ and $E(\mathbb{S})$.

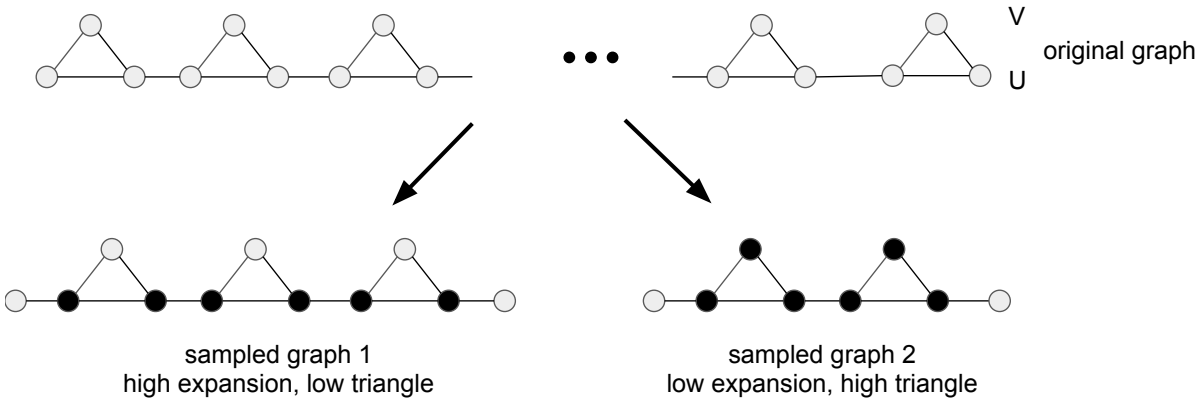


Figure 5.9: Sampling from a stylized line graph with repeating triangles as shown in the top. At the same sample size, we show two sampled graphs exhibiting a tradeoff in maximizing two graph properties: the triangle count and the expansion.

`Barabasi`) and under the same experimental setup as used in Section 5.3. We compare the samplers' performances over thirty-five tasks such as degree, between, and closeness centrality preservation tasks employed in this biological study [189].

Figure 5.10 shows the relative performance of the samplers at a sample size of 10% over different tasks and graph families averaged over 50 independent runs. It shows that no sampler performs superbly across all inference tasks or all graph families. For example, while degree-biased samplers like `FF` are very good at sampling the degree distribution in `Barabasi` graphs due to their skewed degree distributions, they fare poorly at sampling from non-skewed distribution graphs like `Lattice`. We note similar observations in past literature. For example, Wang et al. [159] noted the sampling tradeoff in degree and clustering coefficient-based tasks. Aguiar et al. [25] showed for ecological interaction networks that a sampler for estimating an underlying network property should be dependent on the underlying network topology. Thus, based on the above experiments and observations, we claim that different sampling contexts necessitate different sampling strategies.

Graph family: We test using six synthetically-generated graph families generated by Igraph package [186], and with the corresponding parameters as follows: `G0`: `Barabasi` ($N = 10000, m = 5$), `G1`: `Coreperi` ($N = 10000$, preference matrix=[[0.0020, 0.0010], [0.0010, 0.0010]], block sizes=[2000, 8000]), `G2`: `Erdos` ($N = 10000, p = .001$), `G3`: `Lattice` ($dim = [100, 100], nei = 2$), `G5`: `SBM` ($n = 10000, lamda = 10, k = 10, beta = 0.25$), `G6`: `Watts_Strogatz` ($dim = 2, size = 101, nei = 2, p = .25$).

Task: We test our samplers on thirty-five topological property preservation tasks as used in this biological study [189]. The preservation tasks are evaluated as: Tasks `T0-T3` are

the mean, maximum, minimum and standard deviation for betweenness centrality distributions between the sampled graph and the original graph; tasks `T4-T7` are for clustering coefficient distribution; tasks `T8-T11` are for closeness centrality distribution; tasks `T12-T15` are for network constraint distribution; tasks `T16-T19` are for degree centrality distribution; tasks `T23-T26` are for eigenvector centrality distribution; tasks `T27-T30` are for node levels distribution; tasks `T31-T34` are for effective network size distribution; task `T20` is the absolute difference average pairwise distance between nodes between sampled graph and original graph; task `T21` is the absolute difference for network density between sampled graph and original graph; and task `T22` is the absolute difference for global efficiency of the sampled graph and original graph.

Evaluation: We evaluate our samplers at a fixed sampling budget of 10% of the graph size for 50 independent runs. Figure 7 (in the main text) shows the relative performance of the eight state-of-the-art samplers over the six graph-families (`G0-G5`) and thirty-five biological tasks (`T0-T34`).

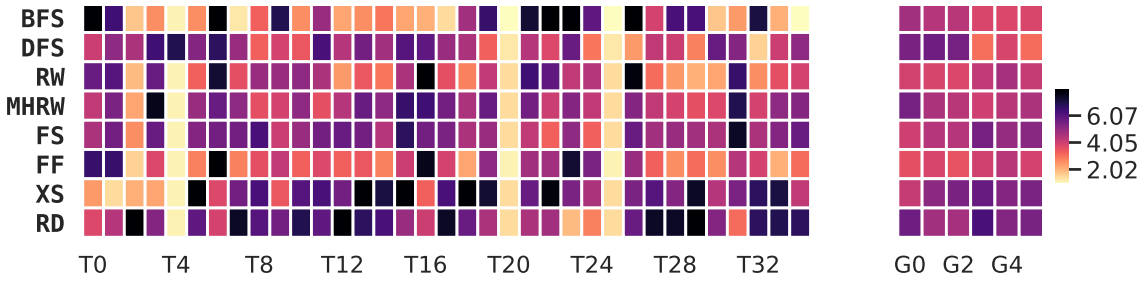


Figure 5.10: Samplers' performance across different tasks (left subplot) and different graph families (right subplot). We observe no state-of-the-art sampler (row) performs best (light yellow) across all tasks or graph families. The value in each cell corresponds to the rank of the sampler. The rows of the plot represent the samplers while the columns correspond to the tasks (T0-T35) and graph families (G0-G6).

The above empiric and theoretic proofs support our claim about the non-existence of a universal graph sampler.

5.4.2 Why simple extensions for learning sampler doesn't work?

We now list the different logical extensions of the state-of-the-art samplers for learning an adaptive network sampler and their limitations.

E1 One approach could be to choose the best sampler from one of the existing sampling methods (`BFS`, `DFS`, `RW`, `FS`, `FF`, `RD`, `XS`) on the training examples. However, most of the existing methods have been designed for few real-world networks and tested for few

network properties such as degree and clustering coefficient distributions. They fail to generalize on unseen graph examples and user-defined tasks. It is therefore imperative that we need a more adaptive framework for learning the sampler.

E2 Another approach would be to define a parameterized sampler such as learning the optimal value of the parameter that yields high sampling performance. For example, we could learn p of Forest Fire sampler, p, q parameter of Node2vec based random walker [190] and ρ value of Rank Degree sampler. However, we note that the objective function of the learner trying to optimize Equation (5.1) would be similar to black-box optimization. Further, each feedback to the optimizer would involve $\mathcal{O}(B)$ for sampling query every time. This is both a highly inefficient and non-scalable solution.

E3 Another possible approach could be to apply standard reinforcement learning algorithms such as Q-Learning and Policy-gradient [106]. However, due to the sequential mechanism of sampling, the sampled sets are not independent. Further, as discussed in Section 5.2, the graph sampling poses unique challenges that make the direct implementation of these algorithms highly infeasible and inefficient.

In summary, learning to sample from networks by applying traditional learning approaches presents several non-trivial challenges.

5.4.3 Limitations and Future Extensions

We now describe some of the major limitations of this work. One, as practitioners, the learning framework requires some example graphs from a graph family for learning the optimal sampling policy. We posit that this problem might be mitigated by using some expected random graph models of the graph family [191] in the training phase. Another option would online learn the sampling policy on some known property of the graph (for example, Facebook and Twitter often share statistics of their dataset [5]), and then applying transfer learning techniques to adapt the sampler for the user-defined task. Another limitation of the learning framework is the computational cost involved during training and a constant factor (dependent on the average degree of the graph) increase in the sampler's time during testing. Overall, we posit that the increase in resource cost in training is often negligible compared to the API cost savings.

For future work, we want to apply that our proposed sampler to new graph families, including heterogeneous and attributed networks and black-box inference tasks. Further, newer

[5] `https://research.fb.com/category/data-science/`

dimensions of sampling such as random node sampling or preferential seed selection could make the sampling policy richer and allow the learner to learn even better samplers. Finally, we note that the proposed learner uses immediate reward feedback. When more computation resources are available, we expect to learn better sampling policies by considering the future reward of adding frontier nodes. Additionally, we considered model-free learning in this work which allowed us to learn generalized sampler for any graph family. When prior information about the task and graph-family is available to the learning framework designer, model-based learners might prove to be more efficient.

Finally, we note that we purposefully choose simplistic features and policy spaces for this study. We note that even simple features and policy space allow us to learn significantly efficient samplers. Exploring more complex features and policy spaces is another possible line of future work.

5.5 EXTENSION: LEARNING CONTEXT-SPECIFIC BRAIN PARCELLATIONS

In this section, we extend context-specific samplers to learn context-specific brain parcellations (or atlases). Labeling of the brain regions (or brain parcellation) defines the nodes of the brain graph. It is, therefore, a critical step in the representation of brain networks and subsequent analysis and diagnosis. Like network samplers, we first show there exists no universal brain parcellation through empirical and theoretical arguments. Next, we develop an adaptive learning framework to learn brain parcellations according to the downstream task and population.

5.5.1 No universal parcellation

In this section, we show the non-existence of a universal brain parcellation using empirical and theoretical arguments.

First, we show that the performance of state-of-the-art brain parcellations varies greatly according to the context. We used nine different atlases (having ROIs between 10 and 200) on six diverse brain classification tasks described in Table 5.7. We observe that different parcellations focus on different brain regions, hence capture different brain features. These brain features are thereafter useful for different classification tasks. For example, brain signals from specific regions like the left hippocampal region are helpful for discerning ADHD disease [73]. At the same time, widened sulci and cortical atrophy is an often useful signal for discerning the biological age of a person [85]. We test the efficacy of different brain atlases using the benchmark classification pipeline found in Dadi et al. [192], i.e., logistic regression

Table 5.7: Detailed description of the brain datasets alongwith the task. We allow enumerate the different widely used state-of-the-art parcellations techniques used for parcellating the brain networks.

Dataset	Task description
COBRE [194]	gender, diagnosis (schizophrenia or not)
NYU [195]	gender
BNU3 [196]	gender, eye-status (open or closed)
Parcellation(size)	**Description**
msdl [197] (39)	Atlas based method, 3d+t volumetric data
basc_multiscale [93] (7, 64, 122, 197)	Atlas based method, 3d+t volumetric data
harvard_oxford [94] (48)	Atlas based method, 3d+t volumetric data
craddock [91] (249)	Atlas based method, 3d+t volumetric data
aal [87] (112)	Atlas based method, 3d+t volumetric data

with L2 norm. The different datasets and tasks used for this empirical survey are described in Table 5.7. In order to compare the performance across different datasets and tasks, we rank the atlases by classification performance obtained as a result of atlas-based brain parcellation. Figure 5.11 shows the relative performance of the atlas-based classification. We observe that there exists no atlas-based parcellation that does best across all datasets and tasks. Thus, based on the above experiments and observations, we claim that different contexts (dataset and task pair) necessitate different parcellation strategies.

Next, we note that it is theoretically impossible to optimal achieve a universal brain parcellation due to the no-free lunch principle of clustering [193]. Thus, in contrast to prior efforts on developing a universal population-level or individual-specific brain parcellation, we propose a universal learning framework that learns population and task-specific brain parcellation that is suited for the given context.

5.5.2 Learning Brain Parcellations

In this subsection, we summarize two adaptive brain parcellation frameworks for adaptively learning the best brain parcellation according to a context.

We formally define the problem of learning a brain parcellation as learning the best clustering of graph nodes. Consider a set of weighted, labeled brain graphs $\mathcal{G}$ $(G_1, G_2, \ldots, G_n)$, where each G_i is composed of vertex set V, edge set $E \subset V \times V$ and weight of edges $W : E \to \mathbb{R}$ and represented as (V, E, W). Further, each brain network is associated with a task-based label Y $(y_1, y_2, \ldots, y_n)$. Thus, given the population $\mathcal{G}$ and task-based label Y,

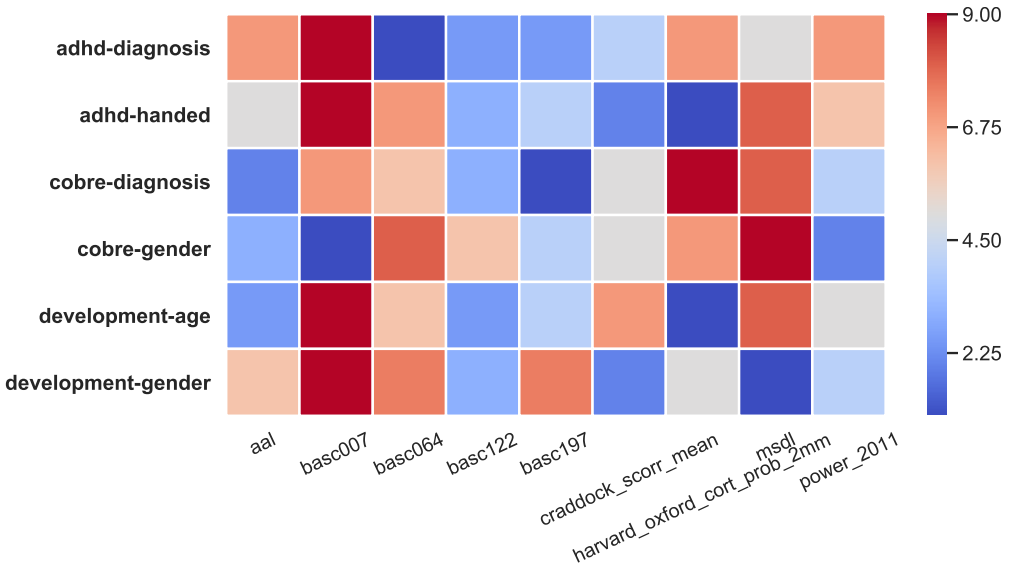


Figure 5.11: Classification performance when using different atlases for parcellating the brain network. We observe no atlas (row) performs best (blue) across all dataset and task pairs. The value in each cell corresponds to the rank of the atlas-based parcellation. The rows of the plot represent the dataset and task pair, while the columns correspond to the atlases.

we seek a parcellation $\mathcal{P} : V \to \{1, 2, \ldots, k\}$ of the brain graph into k groups that preserves the informative parts of the graph for label prediction. Thus, the parcellation $\mathcal{P}$ reduces the number of nodes in the graph from V to k. We create a compressed, parcellation graph G' corresponding to each original graph G.

To state formally, for a given population of brain networks $\mathcal{G}$, a given task-based label Y, and a loss function $\mathbb{L}$, we seek an optimal parcellation that parcellates the brain into a coarser graph while minimizing the task-based loss function.

$$\mathcal{P}^\star = \arg\min_{\mathcal{P}} \mathbb{L}(P(G), Y) \tag{5.10}$$

We now briefly describe two adaptive approaches to learn the best brain parcellation: a) matrix optimization approach, and b) discriminative heuristic.

Optimization approach In this approach, we use matrix optimization to learn the best node-cluster assignment. We express Equation (5.10) that learns both the optimal clustering and classification parameters simultaneously as follows,

$$Loss = \sum_{i=1}^{n} \log(1 + \exp(-y \cdot \text{sum}(W \odot (P^\top \cdot X \cdot P)))) \tag{5.11}$$

We use stochastic gradient descent to learn the best clustering parameter P and classifi-

cation parameter W that minimizes the logistic loss.

Thus, the gradient for W can be expressed as,

$$\begin{aligned}\frac{\partial L}{\partial W} = -(y \cdot \exp(-y \cdot \text{sum}((W^\top \odot ((X \cdot P)^\top \cdot P)) \cdot \text{vector}(1))))/ \\ (1 + \exp(-y \cdot \text{sum}((W^\top \odot ((X \cdot P)^\top \cdot P)) \cdot \text{vector}(1)))) \cdot P^\top \cdot X \cdot P\end{aligned} \tag{5.12}$$

Similarly, the gradient for P can be expressed as,

$$\begin{aligned}\frac{\partial L}{\partial P} = -((y \cdot \exp(-y \cdot \text{sum}((W \odot (P^\top \cdot X \cdot P)) \cdot \text{vector}(1))))/ \\ (1 + \exp(-y \cdot \text{sum}((W \odot (P^\top \cdot X \cdot P)) \cdot \text{vector}(1)))) \cdot X \cdot P \cdot W^\top + \\ (y \cdot \exp(-y \cdot \text{sum}((W^\top \odot ((X \cdot P)^\top \cdot P)) \cdot \text{vector}(1))))/ \\ (1 + \exp(-y \cdot \text{sum}((W^\top \odot ((X \cdot P)^\top \cdot P)) \cdot \text{vector}(1)))) \cdot X^\top \cdot P \cdot W)\end{aligned} \tag{5.13}$$

Discriminative heuristic In the discriminative heuristic approach, we find the clustering for the discriminative features. In order to learn the discriminative parcellation, we split the labeled graphs $\mathcal{G}$ according to the label. For example, for binary classification tasks (with labels $+1$ and -1), we split the graphs into $\mathcal{G}_+$ and $\mathcal{G}_-$ based on the graph labels. Next, we construct a unified graph based on the discriminative score for every edge (independent t-test[6] or logistic regression coefficient). Thereafter, we run spectral clustering on the unified discriminative graph to obtain clusters that preserve the discriminative edges and combines the non-discriminative edges for a specific context.

5.5.3 Experiments and Results

We test the adaptive parcellation algorithms proposed in the previous subsections with the existing clustering algorithms used in literature for parcellation brain networks. For the experiments, we consider five different brain datasets and classification tasks as enumerated below:

- `BNU1` [196] is an fMRI brain connectivity network dataset with each graph corresponding to a subject and comprising 437 nodes (DS00446 atlas). The classification task is gender prediction.
- `NYU` [195] is an fMRI brain connectivity network dataset with each graph corresponding to a subject and comprising 437 nodes (DS00446 atlas). The classification task is predicting whether the subject is above 30 years old or not.

[6] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html`

- `Cobre` [194] is an fMRI brain connectivity network dataset with each graph corresponding to a subject and comprising of 444 nodes (basc_multiscale_2015 atlas). The classification task is to predict whether the subject has ADHD or not.

- `Development` [198] is an fMRI brain connectivity brain network dataset with each graph corresponding to a subject and comprising of 444 nodes (basc_multiscale_2015 atlas). The classification task is to predict whether the subject is an adult or a child.

- `BNU3` [196] is a dMRI brain connectivity network dataset with each graph corresponding to a subject and comprising of 444 nodes (basc_multiscale_2015 atlas). The classification task is gender prediction.

We compare the following parcellation algorithms to parcellate the above networks to small $k = 20$ sized graphs:

- `Ward` [199] is a hierarchical agglomerative clustering algorithm that starts with each node in a separate cluster and merges clusters according to the weights between the clusters. The merging stops when k. We consider different variations of Ward clustering, including variance minimization, single linkage, complete linkage, and average linkage.

- `K-means` [199] is a partitioning clustering algorithm that partitions nodes into k clusters such that the sum of squared differences between the nodes and their representative cluster centroids is minimized.

- `Spectral` [199] clustering algorithm uses the spectrum (eigenvalues) of the graph weights to split the nodes into k groups.

- `Adaptive-Mat` is matrix optimization algorithm as described in Section 5.5.2 that minimizes the logistic loss while simultaneously learning the optimal parcellation of Equation (5.11).

- `Adaptive-Discr` is a discriminative clustering algorithm as described in Section 5.5.2 computes the discriminative score (independent t-test) of node edges followed by a spectral clustering on the discriminative graph.

Table 5.8 shows the classification performance of different parcellation methods. We observe that adaptive parcellations consistently outperform the fixed parcellation methods.

Table 5.8: Performance of different parcellation algorithms. Our proposed adaptive parcellation algorithms outperform the baselines for all datasets and classification tasks.

Parcellations	BNU1	NYU	Development	Cobre	BNU3
k-means	0.511	0.625	0.931	0.547	0.511
Ward-variance	0.596	0.542	0.919	0.440	0.596
Ward-complete	0.596	**0.667**	0.906	0.413	0.596
Ward-average	0.532	0.583	0.913	0.480	0.532
Ward-single	0.553	0.542	0.869	0.553	0.553
Spectral	0.553	**0.667**	0.919	0.420	0.553
Adaptive-Mat	0.489	**0.667**	0.556	0.533	0.489
Adaptive-Discr	**0.702**	**0.667**	**0.950**	**0.611**	**0.702**

5.6 CONCLUSION

In this chapter, we showed that universal network sampling is unattainable. Extensive experiments and theoretic examples suggested the non-existence of a universal network sampler that can simultaneously preserve multiple graph properties. Thus, we proposed a contextual learning framework that learns different sampling policies for different contexts. Next, we proposed a learning framework that is efficient and straightforward but learns a very high-quality sampling policy: GTS. Experiments involving ten different graph families and seven diverse tasks showed GTS significantly outperforms existing samplers by a margin of up to 3×. Furthermore, we showed that the learned sampler is interpretable and can be treated as a generalization of baseline samplers. We extended the adaptive graph sampler idea to learn adaptive brain parcellations.

CHAPTER 6: CONCLUSION AND FUTURE WORK

In this chapter, we first summarize the contributions of this dissertation in Section 6.1 and then discuss open problems in sampling and possible directions for future work in Section 6.2.

6.1 RESEARCH CONTRIBUTIONS

Sampling data from online social networks lie at the heart of several social data-driven applications. To meet different applications' needs, we need to build sampling techniques that can sample social data in a fast and efficient manner while ensuring that the data remains free from biases. However, it is challenging to achieve such a sampler given the diversity of applications, restrictions imposed by the network APIs, and the resource limitations of time and space. In this thesis, we addressed some of the sampling challenges and developed new sampling methodologies.

More specifically, we proposed novel sampling algorithms to sample social data from online social networks. First, we proposed a *surprise-based* sampler grounded in Information Theory to sample node content for data-mining tasks such as clustering, classification, and regression. Next, we proposed a multi-arm bandit-based sampler for sampling hidden populations from the social networks via an attributed search interface. Finally, we advanced the understanding of network sampling from online social networks by showing the non-existence of a universal graph sampler. To address the non-existence of a universal sampler, we proposed a reinforcement learning framework that learns a sampling policy suited for a specified user-application context. We shall now expand on the implication of our contributions and their impact on future work.

6.1.1 Content Sampling for Data Mining Applications

In Chapter 3, we proposed a novel task-independent sampler for attributed networks. While the existing works relied upon network sampling to provide the sample for content, we primarily focused on content sampling. We showed through strong experimental results for a variety of datasets, demonstrating that surprise-based samplers are sample efficient and outperform both random sampling and baseline attribute-agnostic samplers by a wide margin. Our sampler will impact the work of data scientists who deal with the practical realities of sampling large attributed graphs for their work. It will facilitate downstream data analysis tasks like clustering, classification, and regression. Our sampler is simple to

use and more efficient: it requires fewer samples than state-of-the-art baselines to achieve the same clustering and classification accuracy. Further, we showed that our surprise-based sampler could be extended to handle the noise and missing information in Section 3.5.

6.1.2 Hidden Population Sampling

In Chapter 4, we proposed a novel algorithm for sampling hidden target populations from online social networks. The existing works on hidden population sampling were limited in their applications: a) web crawlers can only sample nodes connected by relationship links, b) hidden database samplers aim at sampling the entire database content, which is infeasible for the social networks due to their sheer size, and c) query reformulation engines rely on the rich textual data to discover hidden content which is often not available for users in social networks. In contrast, we proposed a state-aware sampler that exploited structure in combinatorial query space to discover high-yielding queries. Extensive experiments show that our proposed sampler is better than the competing samplers by a factor of 0.9-1.5$\times$ on offline, real-world datasets. Similarly, our proposed sampler outperforms baselines by a margin of 54% on Twitter hidden population tasks and 49% on RateMD experiments. Our sampler will aid social scientists in sampling target hidden populations such as people with mental illnesses [3], jazz musicians [4] and sex workers [5]). Further, our work is of significance to applications such as social segmentation and profiling [200, 201], online advertising, and social mining [131, 132].

6.1.3 Reinforcement Learning Framework To Learn Application-Suitable Subgraph Sampler

Finally, in Chapter 5, we showed that universal network sampling is unattainable. Thus, we proposed a learning framework that learns adaptive network samplers dependent on the application requirement.

We proved that universal graph samplers are not possible using proof by contradiction. We show a trade-off between two topological properties, i.e., depth and breadth, in a stylized tree graph. Further, we show that the trade-off in topological properties exists even at asymptotic limits of the budget in a stylized line graph. Next, we show that the impossibility rule holds not only in theory but also in real-world scenarios by conducting an extensive empirical survey. Our experiments on six different graph families and thirty-five different tasks show significant differences in sampling performance across contexts.

Finally, we addressed the lack of a universal graph sampler by proposing a universal

learning framework that learns appropriate sampler for specified user applications. The empirical results showed our proposed sampler's robustness, which outperforms the state-of-the-art samplers by a margin of 10% to 318%. Our framework is significant: instead of hand-crafting a sampling policy for every new context, our algorithm, by learning a context-dependent sampler, provides a unified way of solving multiple sampling-related problems, including scaling down internet graphs [160], visualizing social graphs [161], and conducting sociological surveys [17].

6.2 FUTURE WORK

There are multiple avenues of future work that can improve and expand upon the samplers mentioned in this dissertation.

6.2.1 Learning Optimal Sample Size

Convention samplers for online social networks, especially the network samplers, operate under the scenario where the appropriate sampling budget required for the downstream application is unknown. Generally, the sampling techniques do better with greater sampling budgets, and in some cases, the sampling budget is provided apriori. In contrast, classical statistical samplers such as random sampling and stratified sampling provide sufficient statistics guarantees [202, 203], and hence provide a sufficient condition for the determination of appropriate sampling budget. The primary bottlenecks impeding the development of such guarantee-supplying samplers are the unknown nature of underlying social data distribution and the dynamic nature of the social networks. However, a sampler that is cognizant of the above factors might be able to provide confidence bounds and optimal sampling size.

6.2.2 Theoretical Sampling Guarantees for Real-World Networks

Most of the sampling strategies for online social networks including Respondent Driven sampler [17], Forest Fire sampler [204], Rank Degree sampler [33], focused crawler [18], and the samplers proposed in Chapters 3 to 5 provide limited theoretical guarantees for the real-world networks. While we prove several sampling guarantees with stylized and synthetically generated datasets in this thesis, it is unclear if the same guarantees will hold for real-world social networks that may significantly differ from the synthetic cases. In the future, samplers that will be able to provide theoretical (sampling) guarantees, even under adversarial conditions, would be very beneficial for downstream applications.

6.2.3 Sampling From Complex Graphs

Most sampling methodologies in literature are either limited to a specific type of social network like undirected and connected graphs or often designed for such graphs. While we tried to develop new sampling methodologies for sampling content from attributed networks in Chapter 3, we noted that there is no sampler for sampling network-content relationships such as assortativity. Few studies have systematically explored the impact of sampling on complex networks such as attributed, weighted, heterogeneous, dynamic, streaming, and multiplex graphs. While we tried to extend our sampling framework for sampling attributes in attributed networks in Chapter 5, much work is needed. We posit that designing more generalized sampling frameworks capable of learning sampling policies suited for different application needs is a promising future research direction.

6.2.4 Exploiting Rich Query Interface

Despite several query and API restrictions imposed by social networks, some social networks like Twitter provide unique application programming interfaces such as streaming API. Streaming API provides a unique opportunity for researchers to collect and analyze the Twitter dataset. For example, Han et al. [122] used Twitter's follow API to estimate the fashion graph's size before sampling fashion users using the faster and more efficient streaming API. Thus, while the new sampling interfaces raise new challenges to the existing sampling mechanisms but they also provide new opportunities to sample data that would otherwise have not been possible. Similarly, we exploit the attributed search to sample users with specific hidden property in Chapter 4, which would not be possible with text search or graph search. Combining multiple query interfaces to address the sampling problems is an exciting area for future study.

6.2.5 Sampler for Advanced Deep-Learning Based Applications

While deep graph learning algorithms like graph convolution network [205] and graph transformers [206] have now become ubiquitous, it is not clear how bias and noise in the sampled data impact the downstream model's capability and inferences. In Chapter 5, we provide for the first time a mechanism for incorporating the application need into the design of the sampler. In the future, we want to develop better sampling mechanisms that will be aware of the user needs and the application context, i.e., the type of social network to be sampled from. A better understanding of the sampler's impact is required to ensure

that data bias in the downstream task is minimized. Given that graph neural networks are widely used in several applications, including fraud detection, friend suggestion, and product recommendation, it is important to ensure that the sampling mechanism used to obtain the training datasets of such a model is free from any bias.

6.2.6 Handling Noise and Missing Information in Sampling Design

Typically, samplers often tend to overlook the noise and missing information in the data by pruning strategy or otherwise. However, most social network datasets abound with noisy and missing information. In Chapter 3, we designed a Bayesian surprised-based information sampler for sampling content from noisy datasets with missing attributes. In the future, a better modeling of the noise and missing information shall lead to better sampling techniques. In Chapter 4, we show the impact of missing information and noise on our sampler. Therefore, it is important to test the robustness of our samplers to such aberrations in the data. A more principled testing framework for testing the robustness of samplers can be a viable direction of future work.

6.2.7 Parallel Samplers

Most social network samplers in literature including random-walk based samplers [30, 54], focused crawlers [18], are sequential samplers. Sequential samplers typically sample one node or few nodes at a time. The state of a sequential sampler is updated sequentially, and the current sampling actions depend on the accumulated state. Sequential sampling is a very slow process for sampling from large-scale social networks that can scale from several million to billions of nodes. Parallel computation is a natural solution to address this issue. Using multiple seed nodes, e.g., frontier samplers [171] or using approximate greedy sampling [129] are natural choices to scale up traditional samplers. In Chapter 3, we attempted to scale the sampler by sampling the neighborhood of the frontier node along-with the frontier node and observe that the sampler's performance is worsened slightly in comparison to a constant factor improvement in speed.

6.2.8 Learning sampler using few examples

In Chapter 5, we showed that there exists no universal sampler and provided a learning framework to adaptively learn appropriate sampler according to application need. Notice that learning an application-specific sampler requires graph examples to train the sampling

policy on the example graphs. Further, we note that the training graphs are distributed non-uniformly across different applications (or domains). For example, Network Repository houses several hundred graphs from social, brain, and chemical domains but fewer than ten or twenty graphs from ecology and citation domains. Learning to learn idea [207] has been successfully used to solve several learning tasks using only a small number of training examples. Applying the above idea of meta-learning to learn high-quality samplers from a few training example graphs will be very beneficial for domains like ecology and citation, where the example graphs are few.

APPENDIX A: APPENDIX

A.1 DETAILED ANALYSIS OF SI SAMPLER

A.1.1 Generation of Synthetic Datasets

We use the Lancichinetti-Fortunato-Radicchi (LFR) [208] algorithm to generate artificial networks of size $N = 1000$, with mixing coefficient $\mu = 0.1$. This value of the mixing coefficient synthesizes real-world networks with a strong community structure. We refer to such networks as LFR (μ =0.1).

Three essential characteristics are important for generating synthetic attributed networks: content skew, purity, and assortativity. Cluster skew refers to the skew in attribute cluster sizes. We use entropy $H(\mathbb{C})$ over the cluster size to measure cluster skew, where, $\mathbb{C} = \{C_1, C_2, \ldots, C_k\}$. The purity p of the cluster refers to the separability or the degree of overlap of the attribute distributions among clusters. Assortativity measures the degree to which links between nodes with similar attributes differ from the same set of nodes but with random edges [15].

To generate the desired attributed graph, we first generate the "bare" network without attributes. Then, for a specified skew, we generate content clusters. Finally, given an assortativity value, we map the synthesized data to nodes in the network through a label propagation algorithm that terminates when the algorithm achieves the target assortativity. We shall refer the interested readers to the code provided for implementation.

A.1.2 Experiments on Synthetic Networks

We briefly summarize our extensive experiments on synthetic networks. We conducted analyses using standard synthetic network benchmarks (LFR $\mu = 0.1$) [208]), generated with different skew, purity, and assortativity parameters.

We use a bi-cluster map Figure A.1 to show the classification results on synthetic networks. The bi-clustering map helps us identify similar performances across samplers and identify conditions where these similarities occur.

The SI sampler outperforms all baselines on synthetic networks. Notably, SI outperforms baseline samplers with increasing skew, and in particular, by 30-40% at a high skew. The table shows that skew and purity play a more significant role in sampler classification performance than assortativity. We see a pronounced effect of cluster purity on the performance

Table A.1: SI's performance (± standard deviation) over the current state-of-the-art samplers (ES, RW, FF, XS) over a span of data-mining tasks—clustering, classification, regression, unique attribute-value discovering—measured over a three network suites: NSF, Patent, and Wikipedia at 5% sampling rate. A column named ΔX indicates relative performance of sampler X over SI. Thus, the column Δ ES reports for example, the value $100 \times \frac{\text{ES}-\text{SI}}{\text{SI}}$. SI is superior to (RW, ES, FF, XS) over all sub-networks belonging to these three datasets.

Dataset	Task	SI	Δ ES	Δ RW	Δ FF	Δ XS
NSF	Attribute coverage	0.41 (±0.07)	↓20.41%(±2.14)	↓26.37%(±0.86)	↓25.00%(±1.01)	↓ 7.08%(±2.57)
	Cluster coverage	0.95 (±0.03)	↓ 9.68%(±1.41)	↓10.19%(±2.10)	↓ 8.81%(±1.93)	↓10.94%(±3.29)
	Regression	0.92 (±0.02)	↓ 4.90%(±1.46)	↓ 6.03%(±1.79)	↓ 6.21%(±2.36)	↓76.50%(±39.56)
Patent	Attribute coverage	0.67 (±0.01)	↓69.67%(±1.68)	↓72.79%(±1.78)	↓68.47%(±1.47)	↓62.96%(±1.77)
	Cluster coverage	0.99 (±0.00)	↓ 1.08%(±0.50)	↓ 1.10%(±0.51)	↓ 1.21%(±0.56)	↓ 1.54%(±0.81)
	Classification	0.86 (±0.04)	↓16.93%(±1.12)	↓17.27%(±1.06)	↓16.60%(±1.08)	↓16.14%(±1.21)
	Regression	0.48 (±0.02)	↓ 1.57%(±0.32)	↓ 3.53%(±0.27)	↓ 2.21%(±0.13)	↑ 0.54%(±0.12)
Wikipedia	Attribute coverage	0.95 (±0.01)	↓43.46%(±5.30)	↓47.73%(±4.97)	↓37.63%(±1.43)	↓19.55%(±5.12)
	Cluster coverage	0.39 (±0.07)	↓ 2.80%(±1.75)	↓ 7.30%(±0.74)	↓ 6.39%(±2.45)	↓28.79%(±12.88)

of all samplers. Furthermore, we can see while SI dominates, XS is better on average than random walk-based samplers (RW) and edge sampling (ES). It is not surprising that SI consistently outperforms attribute-agnostic samplers. This is because SI, to some extent, solves the "class balancing problem" via stratified sampling of objects from each class.

A.1.3 Network structure preservation

This section examines the effects of link-trace sampling on preserving the network structure by observing the extent to which various topological properties are preserved in the sampled graph. We first describe the topological properties used for analysis, followed by a description of the evaluation metrics. Finally, we report the experimental results.

Preserving topological properties of the graph such as degree distribution and clustering coefficient is a desirable goal for several representative samplers in literature [13, 16, 32, 209]. We examine how the attribute-agnostic and attribute-aware samplers preserve several graph topological properties. We evaluate the samplers on preserving the four widely-used topological properties—degree, clustering coefficient, path-length, and assortativity [15]. To evaluate the samplers' performance, we apply the KS statistic between the sampled graph and the original graph's distribution of topological properties (for the degree, clustering coefficient, and path-length); we use the mean absolute difference to evaluate assortativity. For example, the degree distribution property is evaluated using $D = max_k|F(k) - F'(k)|$, where k is over the range of the node degree, and F and F' are the cumulative distribution of node degrees in the sampled graph $G_{\mathbb{S}}$ and original graph G respectively.

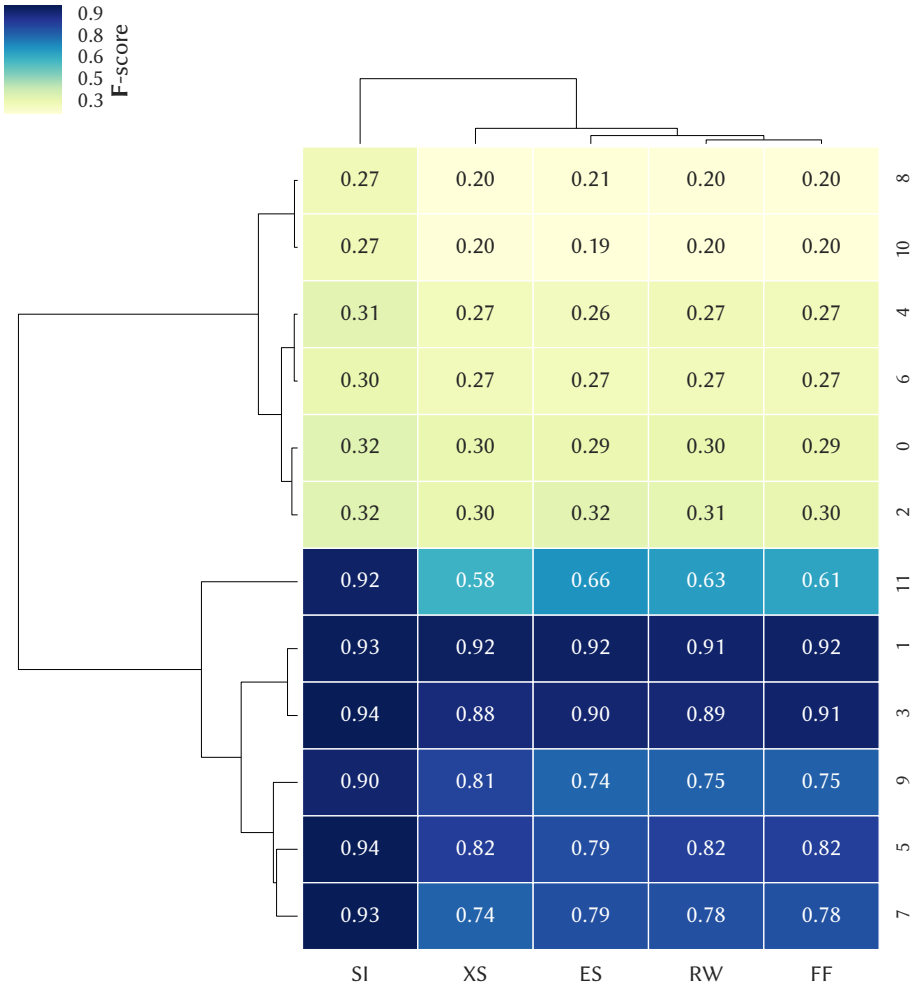


Figure A.1: Classification performance (weighted F_1 score) of network samplers where we show the results using a bi-cluster map. Each row shows the weighted F_1 score for all samplers on a synthetic LFR ($\mu = 0.1$) network generated with a particular skew, purity and assortativity parameters. We use two purity levels, two assortativity levels and three skew levels. High purity: all odd rows; low-purity: all even rows. Low assortativity: rows 0-1, 4-5, 8-9. High assortativity: 2-3, 6-7, 10-11. Low-skew: rows 0-3; Medium skew: rows 4-7; High-skew: rows 8-11.

Table A.2: Performance of attribute-agnostic (`ES, RW, FF, XS`) and attribute-aware sampler (`SI`) over a span of network structure preservation tasks for topological properties—degree, clustering coefficient, path-length, and assortativity—averaged over the network datasets at 5% sampling rate. Smaller values mean higher topological property preservation in the sample and hence better sampling performance. Terms a and b in '$a(\pm b)$' the mean and standard deviation values.

Dataset	Topological property	SI	ES	RW	FF	XS
Facebook	Degree	0.454 (± 0.03)	0.403 (± 0.11)	0.334 (± 0.01)	0.336 (± 0.01)	0.666 (± 0.11)
	Clustering coefficient	0.299 (± 0.07)	0.295 (± 0.04)	0.257 (± 0.03)	0.234 (± 0.00)	0.522 (± 0.09)
	Path length	0.326 (± 0.02)	0.781 (± 0.04)	0.636 (± 0.01)	0.546 (± 0.03)	0.100 (± 0.08)
	Assortativity	0.141 (± 0.01)	0.088 (± 0.02)	0.091 (± 0.01)	0.081 (± 0.07)	0.122 (± 0.00)
Patent	Degree	0.051 (± 0.01)	0.074 (± 0.01)	0.080 (± 0.00)	0.185 (± 0.03)	0.114 (± 0.04)
	Clustering coefficient	0.088 (± 0.01)	0.075 (± 0.00)	0.080 (± 0.00)	0.188 (± 0.08)	0.040 (± 0.00)
	Path length	0.479 (± 0.10)	0.688 (± 0.11)	0.442 (± 0.07)	0.916 (± 0.22)	0.554 (± 0.27)
	Assortativity	0.003 (± 0.00)	0.081 (± 0.01)	0.077 (± 0.02)	0.067 (± 0.01)	0.092 (± 0.00)
NSF	Degree	0.221 (± 0.04)	0.232 (± 0.08)	0.332 (± 0.06)	0.201 (± 0.01)	0.198 (± 0.10)
	Clustering coefficient	0.210 (± 0.04)	0.202 (± 0.01)	0.276 (± 0.07)	0.227 (± 0.06)	0.343 (± 0.08)
	Path length	0.655 (± 0.20)	0.554 (± 0.31)	0.423 (± 0.22)	0.645 (± 0.33)	0.421 (± 0.16)
	Assortativity	0.120 (± 0.05)	0.177 (± 0.07)	0.212 (± 0.06)	0.199 (± 0.05)	0.186 (± 0.05)
Wikipedia	Degree	0.398 (± 0.11)	0.362 (± 0.15)	0.332 (± 0.09)	0.335 (± 0.22)	0.300 (± 0.16)
	Clustering coefficient	0.225 (± 0.11)	0.338 (± 0.16)	0.216 (± 0.08)	0.211 (± 0.05)	0.209 (± 0.02)
	Path length	0.626 (± 0.32)	0.552 (± 0.21)	0.405 (± 0.17)	0.336 (± 0.18)	0.396 (± 0.21)
	Assortativity	0.597 (± 0.00)	0.597 (± 0.00)	0.597 (± 0.00)	0.597 (± 0.00)	0.597 (± 0.00)
Pokec	Degree	0.513 (± 0.10)	0.123 (± 0.01)	0.235 (± 0.04)	0.246 (± 0.07)	0.219 (± 0.10)
	Clustering coefficient	0.260 (± 0.01)	0.118 (± 0.03)	0.157 (± 0.01)	0.168 (± 0.03)	0.271 (± 0.02)
	Path length	0.588 (± 0.21)	0.277 (± 0.18)	0.275 (± 0.11)	0.309 (± 0.10)	0.766 (± 0.10)
	Assortativity	0.030 (± 0.01)	0.076 (± 0.01)	0.025 (± 0.01)	0.020 (± 0.03)	0.213 (± 0.02)
Enron	Degree	0.367 (± 0.05)	0.342 (± 0.04)	0.391 (± 0.04)	0.388 (± 0.05)	0.151 (± 0.01)
	Clustering coefficient	0.303 (± 0.04)	0.295 (± 0.07)	0.324 (± 0.09)	0.335 (± 0.10)	0.122 (± 0.04)
	Path length	0.610 (± 0.20)	0.679 (± 0.39)	0.423 (± 0.21)	0.387 (± 0.11)	0.251 (± 0.07)
	Assortativity	0.033 (± 0.02)	0.024 (± 0.02)	0.013 (± 0.01)	0.012 (± 0.00)	0.013 (± 0.01)

Table A.2 shows the results. As expected, attribute-agnostic samplers, including `FF` and `XS` designed specifically for preserving network structure, perform well on preserving properties like the degree and clustering coefficient. Interestingly, `SI` performs comparatively similar to some of the attribute-agnostic samplers like `BFS` and is much better than some samplers like `ES`. We note that this work (`SI`) focuses on preserving content; as the future work, it will be interesting to include topological properties in the design of `SI`.

For preserving assortativity, we observe that no sampler distinctly outperforms others. Recent work [16] has explored the tradeoff of different samplers at preserving the network-content relationship. However, it remains an open problem to design an efficient sampler that can preserve network-content relationship over a wide range of networks.

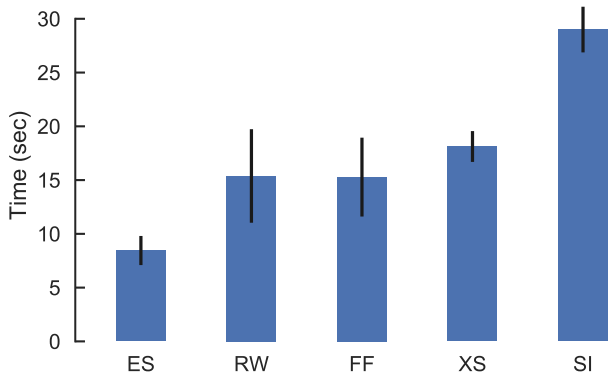


Figure A.2: Figure shows the runtime of different link-trace samplers on the Facebook network at 10% sampling rate. `SI`, being attribute-aware, processes both the network structure and surprise due to content (surprise), and hence incurs a higher computational cost. Attribute-agnostic samplers only process the network structure.

A.1.4 Runtime analysis

In Figure A.2, we observe the runtime for different samplers [1] of attribute-agnostic and attribute-aware samplers on the Facebook network. As noted in Section 3.4.3, `SI` experiences higher time complexity due to the processing of the attributes and the frontier nodes in addition to the sampled nodes. `ES` randomly samples edges from the graph and is the fastest sampler. Attribute-agnostic samplers like `XS` which maximally explore the network structure by sampling the maximal degree node from the frontier set, incur the highest time complexity among the attribute-agnostic samplers. In summary, we observe that `SI` sampler, even though a constant factor (2-3×) slower than the attribute-agnostic samplers, is very useful for sampling content.

A.2 THEORETICAL GUARANTEES OF DT-TMP SAMPLER

Different number of attributes, attribute cardinalities, attribute distributions and the use of decision-based search tree structure makes the analysis of `DT-TMP` difficult. Furthermore, it is non-trivial to extend the standard regret analysis used for analyzing MABs to the `DT-TMP` algorithm. First, unlike MAB, which has just one optimal arm (or query), a standard `DT-TMP`'s optimality involves a set of queries. Second, the underlying quality of a query is fixed in MABs while `DT-TMP` has unconventional reward feedback as described in Section 4.2.4. For sampling with replacement, issuing the same query causes entities to

[1] We implemented all described algorithms in Python using Igraph package [210]. All tests were performed on a computer with 16 GB RAM and a 2.5 GHz Intel Core i7 processor. We performed 100 runs for each sampler.

be repeated, while for sampling without replacement, it sometimes leads to the depletion of the query matching results. Third, on the issuance of a query, the MABs get one result while `DT-TMP` gets the result set R that can be of size anywhere between 0 and m.

In the following lemma, we show that when a specific query's quality is correlated with the general query's quality, `DT-TMP` based sampling strategy is more efficient than naive `TMP` sampling.

Lemma A.1. For a sufficiently large dataset, when the query precision of specific queries within one general query are more similar to each other than specific queries of another general query (clustering effect), `DT-TMP` requires fewer number of queries to find the optimal query than `TMP`.

Proof. For this proof, we shall consider three scenarios of the clustering of the precision values of specific queries centered around their corresponding general queries. 1) when the clusters are well separated, or the general queries are disjoint, 2) when any two clusters among the clusters are disjoint except few specific queries that are common to both. 3) when the clusters corresponding to the general queries are overlapping.

First, consider that there are n disjoint general queries $q_1, q_2, \ldots q_n$ and their corresponding query precision be $p_1, p_2, \ldots p_n$. Further, assume that there are n_i specific queries, $q_{i,j}$ where $j \in \{1, 2, \ldots n_i\}$ within a general query q_i. From Lemma 4.1, the query precision of specific queries $q_{i,j}$ are clustered around q_i's precision, i.e. the specific query $q_{i,j}$'s precision $p_{i,j}$ lies in the range $[p_i - \Delta p_i, p_i + \Delta p_i]$ where $\Delta p_i > 0$. Since, the clusters are well separated we note that the query precision of the specific queries satisfy the following condition: $|p_i - p_j| > \Delta p_i + \Delta p_j$ for any pair of general queries q_i and q_j. The above condition would imply that the all specific queries corresponding to a general query are well-separated. First, we shall proof the separation of clusters composed of precision values of specific queries under the aforementioned condition. Next, we shall proof the efficacy of `DT-TMP` over `TMP` when the clusters satisfy the condition.

We shall now prove that the precision range of specific queries corresponding to a general query q_i is disjoint from the range of specific queries' precision corresponding to another general query q_j where $i \neq j$.

Without loss of generality, we assume that $p_i > p_j$. From the clustering condition stated above,

$$p_i - p_j > \Delta p_i + \Delta p_j \tag{A.1}$$

By re-arranging the terms, we get

$$p_i - \Delta p_i > p_j + \Delta p_j \tag{A.2}$$

Since, $\Delta p_i, \Delta p_j > 0$, thus

$$p_i + \Delta p_i > p_i - \Delta p_i > p_j + \Delta p_i > p_j - \Delta p_j \tag{A.3}$$

It follows form the above inequalities that,

$$[p_i - \Delta p_i, p_i + \Delta p_i] \cap [p_j - \Delta p_j, p_j + \Delta p_j] = \phi \tag{A.4}$$

Thus, we observe that the precision range of specific queries belonging to a general query q_i is $[p_i - \Delta p_i, p_i + \Delta p_i]$ which doesn't overlap with the precision range induced by any another general query q_j.

We shall now use the above clustering condition of the precision ranges of the specific queries to show that the `DT-TMP` requires a lesser number of samples than `TMP` to find the best query in the query space.

Without loss of generality, assume that $p_1 > p_2 > \cdots > p_n$. Further, assume the query precision of a specific query $q_{i,j}$ within the general query q_i are ordered by their precision values $p_{i,j}$. We shall now show the sample complexity of `DT-TMP` for identifying the best query $p_{1,1}$ is lesser than the sample complexity of `TMP`. Note that it follows from Lemma 4.1 that $p_{1,1} \geq p_1$.

From sampling theorem [211], we know that the number of samples from sub-optimal query $q_{i,j}$ needed to discern the best query $q_{1,1}$ with a confidence interval of δ is $\mathcal{O}(\frac{1}{\epsilon^2} \ln \frac{2}{\delta})$, where $\epsilon = p_{1,1} - p_{i,j}$.

`TMP` searches among the specific queries corresponding to all general queries to identify the best query $q_{1,1}$. Thus, `TMP` would have a sample complexity of $\sum_{i=1}^{n} \sum_{j=1}^{n_i} \mathcal{O}(\frac{1}{\epsilon_{i,j}^2} \ln \frac{2}{\delta})$, where $[i,j] \neq [1,1]$ to find the best query $q_{1,1}$.

On the other hand, `DT-TMP` is a two phase sampler. In the first phase, it identifies the best cluster or general query. In the second phase, it identifies the best specific query within the best chosen general query q_1 (cluster). In the second phase, `DT-TMP` explores among only the specific queries corresponding to q_1 to identify $q_{1,1}$. Thus, the sample complexity of `DT-TMP` to identify the best specific query $q_{1,1}$ is $\underbrace{\mathcal{O}(\sum_{i=2}^{n} \frac{1}{(p_1 - p_i)^2} \ln \frac{2}{\delta})}_{\text{phase 1}} + \underbrace{(\sum_{j=2}^{n_1} \frac{1}{(p_{1,1} - p_{1,j})^2} \ln \frac{2}{\delta})}_{\text{phase 2}}$.

Note, that the phase two of `DT-TMP` matches with the best query finding among the specific queries of query q_1. However, we note that the sample complexity for searching among the specific queries $q_{i,j}$ where $i \neq 1$ is lesser than the sample complexity involved in finding the best clusters. Assuming that Δp_i's are similar (i.e., $\Delta p_i \approx \Delta p_j, \forall i, j$), we note that `TMP`

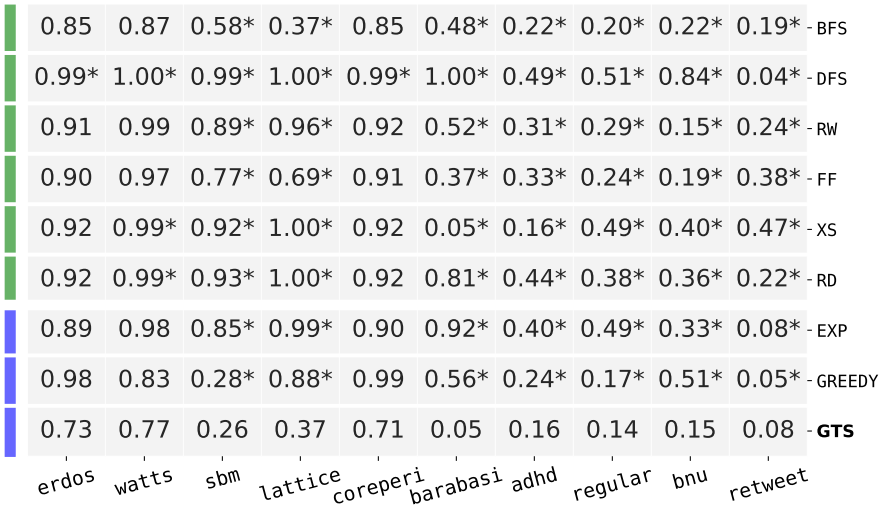


Figure A.3: `Task T1:` Performance of samplers measured using KS statistics between sampled graph's and original graph's degree distribution.

would require around $\mathcal{O}(\sum_{j=1}^{n} n_j \frac{1}{(p_1-p_i)^2} \ln \frac{2}{\delta}))$ sample complexity in-comparison to `DT-TMP`'s $\mathcal{O}(\frac{n_1 n}{(p_1-p_i)^2} \ln \frac{2}{\delta})$.

Thus, we observe that as the number of queryable attributes n increases and the attribute cardinalities of the attributes n_i increases, `DT-TMP`'s relative improvement over `TMP` increases.

Second, we note that when the specific clusters are shared, it can be reduced to the disjoint case by considering two sub-cases: a) when $q_{1,1}$ is a shared specific query, thus even if `DT-TMP` chooses a sub-optimal query will lead to $q_{1,1}$. b) when $q_{1,1}$ is not shared, it can still be proved that the query precision of q_1 is highest, and thus `DT-TMP` finds the optimal query. Third, when the clusters are overlapping, it is harder to prove the efficacy of `DT-TMP` since the best specific query may belong to a sub-optimal general query. We leave the proof of overlapping cases for future work.

QED.

A.3 DETAILED EMPIRICAL RESULTS OF GTS SAMPLER

We now show the detailed empirical experiment results of the baseline and proposed sampler `GTS` introduced in Chapter 5 across different tasks and graph families along with the statistical confidence of 95% shown using *.

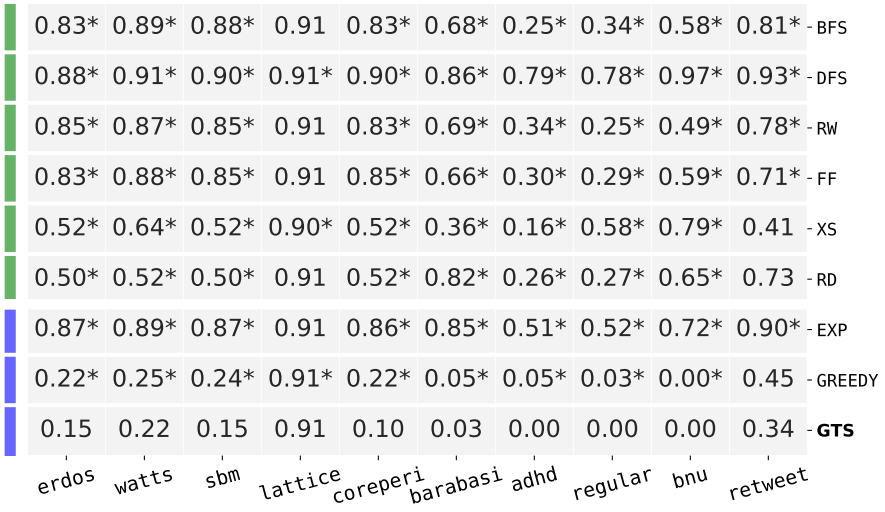


Figure A.4: `Task T2:` Performance of samplers measured using 1 - coverage of influential nodes in the graph. Lower values means better sampling performance.

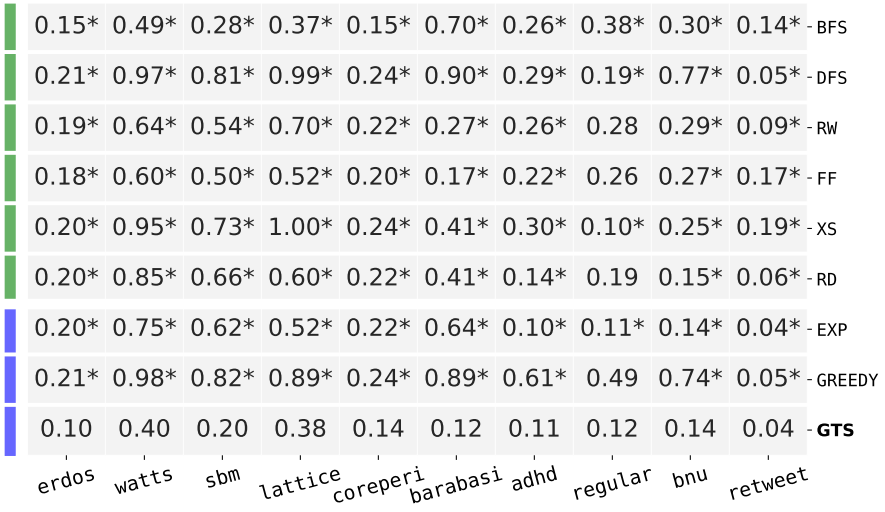


Figure A.5: `Task T3:` Performance of samplers measured using KS statistics between sampled graph's and original graph's clustering coefficient distribution.

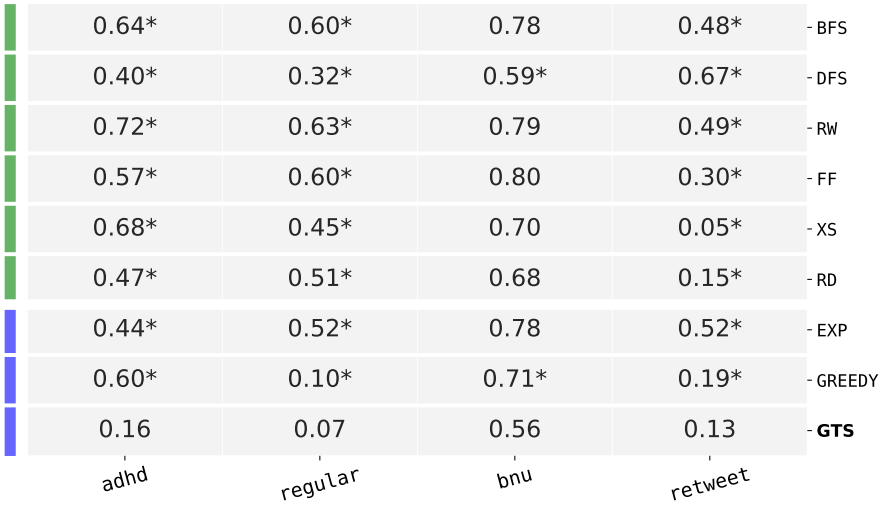


Figure A.6: `Task T4:` Performance of samplers measured using 1 - fraction of communities covered by the sampled node set. Lower value means better sampling performance. Note that some of the graph families like `Erdos` and `Barabasi` graphs are known to have no community structure. We do not conduct the community coverage task for such graph families.

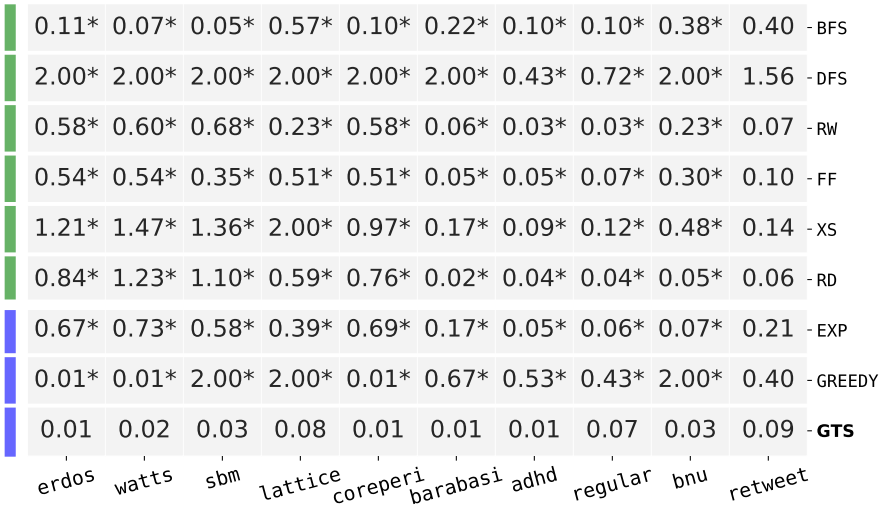


Figure A.7: `Task T5:` Performance of samplers measured using mean absolute difference between sampled graph's and original graph's degree average path-length values. The worst sampling performance are limited to 2 to allow comparison across different tasks. Lower value means better sampling performance.

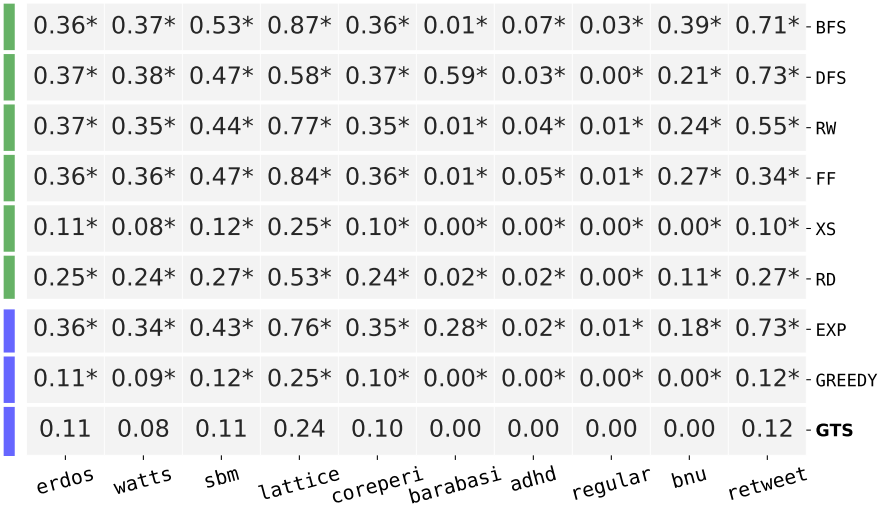


Figure A.8: `Task T6:` Performance of samplers measured using 1- fraction of the network nodes reachable from the sampled set within one hop. Lower value means better sampling performance.

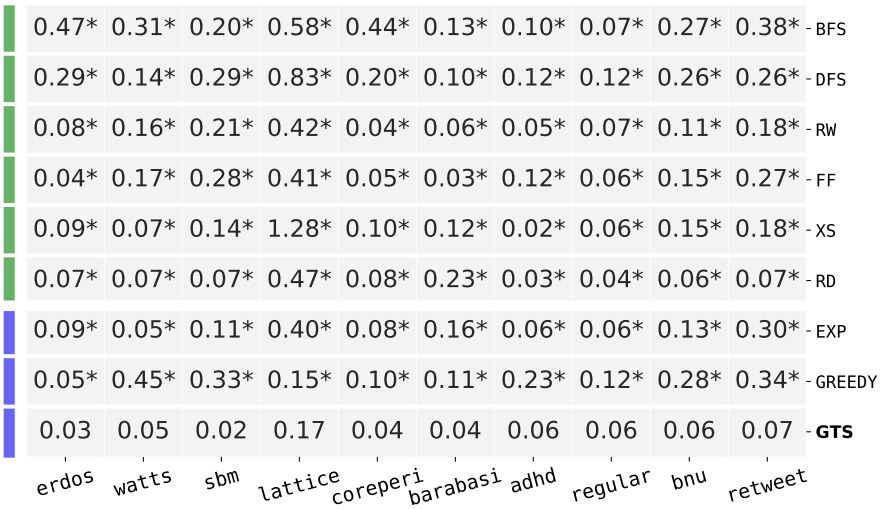


Figure A.9: `Task T7:` Performance of samplers measured using mean absolute difference between sampled graph's and original graph's degree assortativity values. Lower value means better sampling performance. Note that the maximum error can be 2.

REFERENCES

[1] H. A. Voorveld, G. Van Noort, D. G. Muntinga, and F. Bronner, "Engagement with social media and social media advertising: The differentiating role of platform type," *Journal of advertising*, vol. 47, no. 1, pp. 38–54, 2018.

[2] A. A. Alalwan, "Investigating the impact of social media advertising features on customer purchase intention," *International Journal of Information Management*, vol. 42, pp. 65–77, 2018.

[3] M. De Choudhury, S. S. Sharma, T. Logar, W. Eekhout, and R. C. Nielsen, "Gender and cross-cultural differences in social media disclosures of mental illness," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW '17. New York, NY, USA: ACM, 2017, pp. 353–369.

[4] D. D. Heckathorn and J. Jeffri, "Finding the beat: Using respondent-driven sampling to study jazz musicians," *Poetics*, vol. 28, no. 4, pp. 307–329, 2001.

[5] M. Malekinejad, L. G. Johnston, C. Kendall, L. R. F. S. Kerr, M. R. Rifkin, and G. W. Rutherford, "Using respondent-driven sampling methodology for hiv biological and behavioral surveillance in international settings: a systematic review," *AIDS and Behavior*, vol. 12, no. 1, pp. 105–130, 2008.

[6] N. Anstead and B. O'Loughlin, "Social media analysis and public opinion: The 2010 uk general election," *Journal of computer-mediated communication*, vol. 20, no. 2, pp. 204–220, 2015.

[7] V. Martin, X. Zhou, E. Marshall, B. Jia, G. Fusheng, M. A. FrancoDixon, N. DeHaan, D. U. Pfeiffer, R. J. S. Magalhães, and M. Gilbert, "Risk-based surveillance for avian influenza control along poultry market chains in south china: the value of social network analysis," *Preventive veterinary medicine*, vol. 102, no. 3, pp. 196–205, 2011.

[8] T. Kuchler, D. Russel, and J. Stroebel, "The geographic spread of covid-19 correlates with the structure of social networks as measured by facebook," *Journal of Urban Economics*, p. 103314, 2021.

[9] M. Mestyán, T. Yasseri, and J. Kertész, "Early prediction of movie box office success based on wikipedia activity big data," *PloS one*, vol. 8, no. 8, p. e71226, 2013.

[10] A. Chatfield and U. Brajawidagda, "Twitter tsunami early warning network: a social network analysis of twitter information flows," *ACIS*, 2012.

[11] J. Clement, "Global social networks ranked by number of users 2020," *Statista. com*, 2020.

[12] 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*, 2007, pp. 29–42.

[13] A. S. Maiya and T. Y. Berger-Wolf, "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.

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

[15] M. E. Newman, "Mixing patterns in networks," *Physical Review E*, vol. 67, no. 2, p. 026126, 2003.

[16] C. Wagner, P. Singer, F. Karimi, J. Pfeffer, and M. Strohmaier, "Sampling from social networks with attributes," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1181–1190.

[17] D. D. Heckathorn, "Respondent-driven sampling: a new approach to the study of hidden populations," *Social problems*, vol. 44, no. 2, pp. 174–199, 1997.

[18] S. Chakrabarti, M. Van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," *Computer networks*, vol. 31, no. 11-16, pp. 1623–1640, 1999.

[19] C. Olston, M. Najork et al., "Web crawling," *Foundations and Trends® in Information Retrieval*, vol. 4, no. 3, pp. 175–246, 2010.

[20] S. Raghavan and H. Garcia-Molina, "Crawling the hidden web," Stanford, Tech. Rep., 2000.

[21] C. Sheng, N. Zhang, Y. Tao, and X. Jin, "Optimal algorithms for crawling a hidden database in the web," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1112–1123, 2012.

[22] C. Li, P. Resnick, and Q. Mei, "Multiple queries as bandit arms," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 1089–1098.

[23] S. Y. Rieh et al., "Analysis of multiple query reformulations on the web: The interactive information retrieval context," *Information Processing & Management*, vol. 42, no. 3, pp. 751–768, 2006.

[24] A. Khazaee, A. Ebrahimzadeh, and A. Babajani-Feremi, "Identifying patients with alzheimer's disease using resting-state fmri and graph theory," *Clinical Neurophysiology*, vol. 126, no. 11, pp. 2132–2141, 2015.

[25] M. A. de Aguiar, E. A. Newman, M. M. Pires, J. D. Yeakel, C. Boettiger, L. A. Burkle, D. Gravel, P. R. Guimarães Jr, J. L. O'Donnell, T. Poisot et al., "Revealing biases in the sampling of ecological interaction networks," *PeerJ*, vol. 7, p. e7566, 2019.

[26] K. Subbian, B. A. Prakash, and L. Adamic, "Detecting large reshare cascades in social networks," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 597–605.

[27] M. De Choudhury, Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher, "How does the data sampling strategy impact the discovery of information diffusion in social media?" *ICWSM*, vol. 10, pp. 34–41, 2010.

[28] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics reports*, vol. 659, pp. 1–44, 2016.

[29] J. Tang, C. Zhang, K. Cai, L. Zhang, and Z. Su, "Sampling representative users from large social networks," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[30] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: A case study of unbiased sampling of osns," in *INFOCOM*. Ieee, 2010, pp. 1–9.

[31] M. Kurant, A. Markopoulou, and P. Thiran, "Towards unbiased bfs sampling," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1799–1809, 2011.

[32] 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.

[33] E. Voudigari, N. Salamanos, T. Papageorgiou, and E. J. Yannakoudakis, "Rank degree: An efficient algorithm for graph sampling," in *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE Press, 2016, pp. 120–129.

[34] S. Thompson, *Sampling*, ser. CourseSmart. Wiley, 2012. [Online]. Available: https://books.google.com/books?id=-sFtXLIdDiIC

[35] M. Q. Patton, *Qualitative research*. Wiley Online Library, 2005.

[36] F. Morstatter, J. Pfeffer, H. Liu, and K. Carley, "Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 7, no. 1, 2013.

[37] F. Morstatter, J. Pfeffer, and H. Liu, "When is it biased? assessing the representativeness of twitter's streaming api," in *Proceedings of the 23rd international conference on world wide web*, 2014, pp. 555–556.

[38] J.-Y. Li and M.-Y. Yeh, "On sampling type distribution from heterogeneous social networks," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2011, pp. 111–122.

[39] C.-L. Yang, P.-H. Kung, C.-T. Li, C.-A. Chen, and S.-D. Lin, "Sampling heterogeneous networks," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 1247–1252.

[40] H. Park and S. Moon, "Sampling bias in user attribute estimation of osns," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 183–184.

[41] M. Gjoka, E. Smith, and C. T. Butts, "Estimating subgraph frequencies with or without attributes from egocentrically sampled data," *arXiv preprint arXiv:1510.08119*, 2015.

[42] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan, "Guise: Uniform sampling of graphlets for large graph analysis," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 91–100.

[43] P. Wu, J.-R. Wen, H. Liu, and W.-Y. Ma, "Query selection techniques for efficient crawling of structured web sources," in *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*. IEEE, 2006, pp. 47–47.

[44] Q. Zheng, Z. Wu, X. Cheng, L. Jiang, and J. Liu, "Learning to crawl deep web," *Information Systems*, vol. 38, no. 6, pp. 801–819, 2013.

[45] F. Menczer and R. K. Belew, "Adaptive retrieval agents: Internalizing local context and scaling up to the web," *Machine Learning*, vol. 39, no. 2, pp. 203–242, 2000.

[46] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur, "The shark-search algorithm. an application: tailored web site mapping," *Computer Networks and ISDN Systems*, vol. 30, no. 1, pp. 317–326, 1998.

[47] M. Granovetter, "Network sampling: Some first steps," *American Journal of Sociology*, pp. 1287–1303, 1976.

[48] O. Frank, "Sampling and estimation in large social networks," *Social networks*, vol. 1, no. 1, pp. 91–101, 1978.

[49] E. Costenbader and T. W. Valente, "The stability of centrality measures when networks are sampled," *Social networks*, vol. 25, no. 4, pp. 283–307, 2003.

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

[51] M. Kurant, A. Markopoulou, and P. Thiran, "On the bias of bfs (breadth first search)," in *Teletraffic Congress (ITC), 2010 22nd International*. IEEE, 2010, pp. 1–8.

[52] F. Chiericetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlos, "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.

[53] X. Wang, R. T. Ma, Y. Xu, and Z. Li, "Sampling online social networks via heterogeneous statistics," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2587–2595.

[54] C. Hubler, H.-P. Kriegel, K. Borgwardt, and Z. Ghahramani, "Metropolis algorithms for representative subgraph sampling," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 283–292.

[55] V. Krishnamurthy, M. Faloutsos, M. Chrobak, J.-H. Cui, L. Lao, and A. G. Percus, "Sampling large internet topologies for simulation purposes," *Computer Networks*, vol. 51, no. 15, pp. 4284–4302, 2007.

[56] S. J. Hardiman, P. Richmond, and S. Hutzler, "Calculating statistics of complex networks through random walks with an application to the on-line social network bebo," *The European Physical Journal B*, vol. 71, no. 4, pp. 611–622, 2009.

[57] L. da Fontoura Costa and G. Travieso, "Exploring complex networks through random walks," *Physical Review E*, vol. 75, no. 1, p. 016102, 2007.

[58] M. Zhong and K. Shen, "Random walk based node sampling in self-organizing networks," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 3, pp. 49–55, 2006.

[59] M. Kurant, C. T. Butts, and A. Markopoulou, "Graph size estimation," *arXiv preprint arXiv:1210.0460*, 2012.

[60] T. Feder and R. Motwani, "Clique partitions, graph compression and speeding-up algorithms," *Journal of Computer and System Sciences*, vol. 51, no. 2, pp. 261–272, 1995.

[61] A. Apostolico and G. Drovandi, "Graph compression by bfs," *Algorithms*, vol. 2, no. 3, pp. 1031–1044, 2009.

[62] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1913–1926, 2011.

[63] W.-S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi, "A general framework for graph sparsification," *SIAM Journal on Computing*, vol. 48, no. 4, pp. 1196–1223, 2019.

[64] P. Robles, S. Moreno, and J. Neville, "Sampling of attributed networks from hierarchical generative models," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1155–1164.

[65] Y. Murase, H.-H. Jo, J. Török, J. Kertész, and K. Kaski, "Sampling networks by nodal attributes," *Physical Review E*, vol. 99, no. 5, p. 052304, 2019.

[66] J. J. Pfeiffer III, S. Moreno, T. La Fond, J. Neville, and B. Gallagher, "Attributed graph models: Modeling network structure with correlated attributes," in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 831–842.

[67] A. A. Benczúr and D. R. Karger, "Randomized approximation schemes for cuts and flows in capacitated graphs," *SIAM Journal on Computing*, vol. 44, no. 2, pp. 290–319, 2015.

[68] A. Moitra, "Vertex sparsification and oblivious reductions," *SIAM Journal on Computing*, vol. 42, no. 6, pp. 2400–2423, 2013.

[69] P. Erdős and A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[70] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[71] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[72] O. Sporns, C. J. Honey, and R. Kötter, "Identification and classification of hubs in brain networks," *PloS one*, vol. 2, no. 10, p. e1049, 2007.

[73] K. Ota, N. Oishi, K. Ito, H. Fukuyama, S.-J. S. Group et al., "A comparison of three brain atlases for mci prediction," *Journal of neuroscience methods*, vol. 221, pp. 139–150, 2014.

[74] A. Zalesky, A. Fornito, I. H. Harding, L. Cocchi, M. Yücel, C. Pantelis, and E. T. Bullmore, "Whole-brain anatomical networks: does the choice of nodes matter?" *Neuroimage*, vol. 50, no. 3, pp. 970–983, 2010.

[75] N. U. Dosenbach, B. Nardos, A. L. Cohen, D. A. Fair, J. D. Power, J. A. Church, S. M. Nelson, G. S. Wig, A. C. Vogel, C. N. Lessov-Schlaggar et al., "Prediction of individual brain maturity using fmri," *Science*, vol. 329, no. 5997, pp. 1358–1361, 2010.

[76] A. Riaz, M. Asad, S. M. R. Al Arif, E. Alonso, D. Dima, P. Corr, and G. Slabaugh, "Deep fmri: An end-to-end deep network for classification of fmri data," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, 2018, pp. 1419–1422.

[77] A. Anand, Y. Li, Y. Wang, J. Wu, S. Gao, L. Bukhari, V. P. Mathews, A. Kalnin, and M. J. Lowe, "Activity and connectivity of brain mood regulating circuit in depression: a functional magnetic resonance study," *Biological psychiatry*, vol. 57, no. 10, pp. 1079–1088, 2005.

[78] A. Hahn, P. Stein, C. Windischberger, A. Weissenbacher, C. Spindelegger, E. Moser, S. Kasper, and R. Lanzenberger, "Reduced resting-state functional connectivity between amygdala and orbitofrontal cortex in social anxiety disorder," *Neuroimage*, vol. 56, no. 3, pp. 881–889, 2011.

[79] A. Nair, J. M. Treiber, D. K. Shukla, P. Shih, and R.-A. Müller, "Impaired thalamocortical connectivity in autism spectrum disorder: a study of functional and anatomical connectivity," *Brain*, vol. 136, no. 6, pp. 1942–1955, 2013.

[80] C. D. Hacker, J. S. Perlmutter, S. R. Criswell, B. M. Ances, and A. Z. Snyder, "Resting state functional connectivity of the striatum in parkinson's disease," *Brain*, vol. 135, no. 12, pp. 3699–3711, 2012.

[81] S. V. Kalmady, R. Greiner, R. Agrawal, V. Shivakumar, J. C. Narayanaswamy, M. R. Brown, A. J. Greenshaw, S. M. Dursun, and G. Venkatasubramanian, "Towards artificial intelligence in mental health by improving schizophrenia prediction with multiple brain parcellation ensemble-learning," *npj Schizophrenia*, vol. 5, no. 1, pp. 1–11, 2019.

[82] C. D. Good, I. Johnsrude, J. Ashburner, R. N. Henson, K. J. Friston, and R. S. Frackowiak, "Cerebral asymmetry and the effects of sex and handedness on brain structure: a voxel-based morphometric analysis of 465 normal adult human brains," *Neuroimage*, vol. 14, no. 3, pp. 685–700, 2001.

[83] N. I. Eisenberger, M. D. Lieberman, and A. B. Satpute, "Personality from a controlled processing perspective: an fmri study of neuroticism, extraversion, and self-consciousness," *Cognitive, Affective, & Behavioral Neuroscience*, vol. 5, no. 2, pp. 169–181, 2005.

[84] S. J. Banks, K. T. Eddy, M. Angstadt, P. J. Nathan, and K. L. Phan, "Amygdala–frontal connectivity during emotion regulation," *Social cognitive and affective neuroscience*, vol. 2, no. 4, pp. 303–312, 2007.

[85] A. C. Evans, A. L. Janke, D. L. Collins, and S. Baillet, "Brain templates and atlases," *Neuroimage*, vol. 62, no. 2, pp. 911–922, 2012.

[86] J. Mazziotta, A. Toga, A. Evans, P. Fox, J. Lancaster, K. Zilles, R. Woods, T. Paus, G. Simpson, B. Pike et al., "A probabilistic atlas and reference system for the human brain: International consortium for brain mapping (icbm)," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 356, no. 1412, pp. 1293–1322, 2001.

[87] N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, and M. Joliot, "Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain," *Neuroimage*, vol. 15, no. 1, pp. 273–289, 2002.

[88] J. W. Bohland, H. Bokil, C. B. Allen, and P. P. Mitra, "The brain atlas concordance problem: quantitative comparison of anatomical parcellations," *PloS one*, vol. 4, no. 9, p. e7200, 2009.

[89] B. T. Yeo, F. M. Krienen, J. Sepulcre, M. R. Sabuncu, D. Lashkari, M. Hollinshead, J. L. Roffman, J. W. Smoller, L. Zöllei, J. R. Polimeni et al., "The organization of the human cerebral cortex estimated by intrinsic functional connectivity," *Journal of neurophysiology*, 2011.

[90] T. Blumensath, T. E. Behrens, and S. M. Smith, "Resting-state fmri single subject cortical parcellation based on region growing," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2012, pp. 188–195.

[91] R. C. Craddock, G. A. James, P. E. Holtzheimer III, X. P. Hu, and H. S. Mayberg, "A whole brain fmri atlas generated via spatially constrained spectral clustering," *Human brain mapping*, vol. 33, no. 8, pp. 1914–1928, 2012.

[92] X. Artaechevarria, A. Munoz-Barrutia, and C. Ortiz-de Solórzano, "Combination strategies in multi-atlas image segmentation: application to brain mr data," *IEEE transactions on medical imaging*, vol. 28, no. 8, pp. 1266–1277, 2009.

[93] P. Bellec, P. Rosa-Neto, O. C. Lyttelton, H. Benali, and A. C. Evans, "Multi-level bootstrap analysis of stable clusters in resting-state fmri," *Neuroimage*, vol. 51, no. 3, pp. 1126–1139, 2010.

[94] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman et al., "An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest," *Neuroimage*, vol. 31, no. 3, pp. 968–980, 2006.

[95] S. B. Eickhoff, D. Bzdok, A. R. Laird, C. Roski, S. Caspers, K. Zilles, and P. T. Fox, "Co-activation patterns distinguish cortical modules, their connectivity and functional differentiation," *Neuroimage*, vol. 57, no. 3, pp. 938–949, 2011.

[96] P. Roca, A. Tucholka, D. Riviere, P. Guevara, C. Poupon, and J.-F. Mangin, "Inter-subject connectivity-based parcellation of a patch of cerebral cortex," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2010, pp. 347–354.

[97] A. W. Toga, P. M. Thompson, S. Mori, K. Amunts, and K. Zilles, "Towards multimodal atlases of the human brain," *Nature Reviews Neuroscience*, vol. 7, no. 12, pp. 952–966, 2006.

[98] A. W. Toga and P. M. Thompson, "What is where and why it is important," *NeuroImage*, vol. 37, no. 4, pp. 1045 – 1049, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1053811907001206

[99] S. S. Ghosh, A. Keshavan, and G. Langs, "Predicting treatment response from resting state fmri data: comparison of parcellation approaches," in *2013 International Workshop on Pattern Recognition in Neuroimaging*. IEEE, 2013, pp. 225–228.

[100] W. H. Press, "Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research," *Proceedings of the National Academy of Sciences*, vol. 106, no. 52, pp. 22 387–22 392, 2009.

[101] B. Awerbuch and R. Kleinberg, "Online linear optimization and adaptive routing," *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 97–114, 2008.

[102] M. D. Hoffman, E. Brochu, and N. de Freitas, "Portfolio allocation for bayesian optimization." in *UAI*. Citeseer, 2011, pp. 327–336.

[103] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 661–670.

[104] F. Murai, D. Rennó, B. Ribeiro, G. L. Pappa, D. Towsley, and K. Gile, "Selective harvesting over networks," *Data Mining and Knowledge Discovery*, vol. 32, no. 1, pp. 187–217, 2018.

[105] K. Madhawa and T. Murata, "A multi-armed bandit approach for exploring partially observed networks," *Applied Network Science*, vol. 4, no. 1, pp. 1–18, 2019.

[106] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[107] M. Bowling and M. Veloso, "An analysis of stochastic game theory for multiagent reinforcement learning," Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, Tech. Rep., 2000.

[108] S. Still and D. Precup, "An information-theoretic approach to curiosity-driven reinforcement learning," *Theory in Biosciences*, vol. 131, no. 3, pp. 139–148, 2012.

[109] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[110] C. Buck, J. Bulian, M. Ciaramita, W. Gajewski, A. Gesmundo, N. Houlsby, and W. Wang, "Ask the right questions: Active question reformulation with reinforcement learning," *arXiv preprint arXiv:1705.07830*, 2017.

[111] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *arXiv preprint arXiv:2003.03600*, 2020.

[112] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[113] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 6348–6358.

[114] W. Zhang and T. G. Dietterich, "Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling," *Journal of Artificial Intelligence Reseach*, vol. 1, pp. 1–38, 2000.

[115] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[116] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[117] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, "Learning to fly," in *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 385–393.

[118] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020.

[119] E. Berger, H. B. Amor, D. Vogt, and B. Jung, "Towards a simulator for imitation learning with kinesthetic bootstrapping," in *Workshop Proceedings of Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, 2008, pp. 167–173.

[120] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[121] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[122] J. Han, Q. Chen, X. Jin, W. Xu, Y. Wanxian, S. Kumar, L. Zhao, H. Sundaram, and R. Kumar, "Fitnet: Identifying fashion influencers on twitter." in *Proceedings of ACM Human-Computer Interaction*, ser. CSCW '21. ACM, 2021.

[123] S. Kumar and H. Sundaram, "Attribute-guided network sampling mechanisms," in *ACM Transactions on Knowledge Discovery from Data*, 2020.

[124] D. J.-L. Lee, J. Han, D. Chambourova, and R. Kumar, "Identifying fashion accounts in social networks," in *KDD Workshop on ML Meets Fashion*, 2017.

[125] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[126] F. Perez-Cruz, "Kullback-leibler divergence estimation of continuous distributions," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1666–1670.

[127] Q. Wang, S. R. Kulkarni, and S. Verdu, "A nearest-neighbor approach to estimating divergence between continuous random vectors," in *Information Theory, 2006 IEEE International Symposium on*. IEEE, 2006, pp. 242–246.

[128] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*. Singapore, 1997, pp. 21–34.

[129] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 420–429.

[130] A. Kumar, "Bayesian attributed network sampling," Ph.D. dissertation, UIUC, 2020.

[131] E. Raisi and B. Huang, "Cyberbullying detection with weakly supervised machine learning," in *ASONAM*. ACM, 2017, pp. 409–416.

[132] H. Karimi, J. Tang, and Y. Li, "Toward end-to-end deception detection in videos," in *IEEE International Conference on Big Data*. IEEE, 2018, pp. 1278–1283.

[133] M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro, "Crawling the content hidden behind web forms," in *International Conference on Computational Science and Its Applications*. Springer, 2007, pp. 322–333.

[134] S. Kumar, H. Gao, C. Wang, K. C.-C. Chang, and H. Sundaram, "Hierarchical multi-armed bandits for discovering hidden populations," *ASONAM*, 2019.

[135] M. De Choudhury, M. Gamon, S. Counts, and E. Horvitz, "Predicting depression via social media," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 7, 2013.

[136] S. Tsugawa, Y. Kikuchi, F. Kishino, K. Nakajima, Y. Itoh, and H. Ohsaki, "Recognizing depression from twitter activity," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 3187–3196.

[137] D. Robinson, "Introduction to empirical bayes: Examples from baseball statistics," 2017.

[138] S. Agrawal and N. Goyal, "Further optimal regret bounds for thompson sampling," in *Artificial Intelligence and Statistics*, 2013, pp. 99–107.

[139] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[140] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[141] J. Komiyama, J. Honda, and H. Nakagawa, "Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays," *arXiv preprint arXiv:1506.00779*, 2015.

[142] B. H. Hall, A. B. Jaffe, and M. Trajtenberg, "The nber patent citation data file: Lessons, insights and methodological tools," National Bureau of Economic Research, Tech. Rep., 2001.

[143] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *Kdd*, vol. 96. Citeseer, 1996, pp. 202–207.

[144] A. M. Medina and M. Mohri, "Learning theory and algorithms for revenue optimization in second price auctions with reserve," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 262–270.

[145] "Imdb dataset," https://www.imdb.com/interfaces/, 2019, accessed: 11-12-2019.

[146] R. R. Larson, "Introduction to information retrieval," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 4, pp. 852–853, 2010.

[147] A. Nazi, S. Thirumuruganathan, V. Hristidis, N. Zhang, and G. Das, "Querying hidden attributes in an online community network," in *Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on*. IEEE, 2015, pp. 657–662.

[148] A. Dasgupta, G. Das, and H. Mannila, "A random walk approach to sampling hidden databases," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 629–640.

[149] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[150] Twitter Inc., "Annual report, http://bit.ly/2LhgDEc," 2018.

[151] L. Mullen, C. Blevins, and B. Schmidt, "Gender: predict gender from names using historical data," *R package version 0.5*, vol. 1, 2015.

[152] G. Inc, "Celebrating nine years of github with an anniversary sale," 2017.

[153] E. Even-Dar, S. Mannor, and Y. Mansour, "Pac bounds for multi-armed bandit and markov decision processes," in *International Conference on Computational Learning Theory*. Springer, 2002, pp. 255–270.

[154] M. Salehi, H. R. Rabiee, N. Nabavi, and S. Pooya, "Characterizing twitter with respondent-driven sampling," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, Dec 2011, pp. 1211–1217.

[155] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks." in *NIPS*, vol. 2012, 2012, pp. 548–56.

[156] J. Wang, M. Li, H. Wang, and Y. Pan, "Identification of essential proteins based on edge clustering coefficient," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1070–1080, 2011.

[157] A. D. Broido and A. Clauset, "Scale-free networks are rare," *Nature communications*, vol. 10, no. 1, p. 1017, 2019.

[158] P. Doreian, V. Batagelj, and A. Ferligoj, *Generalized blockmodeling*. Cambridge university press, 2005, vol. 25.

[159] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li, "Understanding graph sampling algorithms for social network analysis," in *ICDCS*. IEEE, 2011, pp. 123–128.

[160] V. Krishnamurthy, M. Faloutsos, M. Chrobak, L. Lao, J.-H. Cui, and A. G. Percus, "Reducing large internet topologies for faster simulations," in *International Conference on Research in Networking*. Springer, 2005, pp. 328–341.

[161] R. Pienta, Z. Lin, M. Kahng, J. Vreeken, P. P. Talukdar, J. Abello, G. Parameswaran, and D. H. P. Chau, "Adaptivenav: discovering locally interesting and surprising nodes in large graphs," in *IEEE VIS Conference (Poster)*. Citeseer, 2015.

[162] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015. [Online]. Available: http://networkrepository.com

[163] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, June 2014.

[164] S. Ivanov and E. Burnaev, "Anonymous walk embeddings," *arXiv preprint arXiv:1805.11921*, 2018.

[165] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[166] P. Erdos and A. Renyi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[167] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[168] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer networks*, vol. 33, no. 1-6, pp. 309–320, 2000.

[169] P. Bellec, C. Chu, F. Chouinard-Decorte, Y. Benhajali, D. S. Margulies, and R. C. Craddock, "The neuro bureau adhd-200 preprocessed repository," *Neuroimage*, vol. 144, pp. 275–286, 2017. [Online]. Available: https://nilearn.github.io/modules/generated/nilearn.datasets.fetch_adhd.html

[170] C. Yan, D. Liu, Y. He, Q. Zou, C. Zhu, X. Zuo, X. Long, and Y. Zang, "Spontaneous brain activity in the default mode network is sensitive to different resting-state conditions with limited cognitive load," *PloS one*, vol. 4, no. 5, p. e5743, 2009. [Online]. Available: http://fcon_1000.projects.nitrc.org/indi/CoRR/html/bnu_3.html

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

[172] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *ACM SIGCOMM*, vol. 29, no. 4, pp. 251–262, 1999.

[173] N. A. Christakis and J. H. Fowler, "Social network sensors for early detection of contagious outbreaks," *PloS one*, vol. 5, no. 9, p. e12948, 2010.

[174] J. Hu, K. Meng, X. Chen, C. Lin, and J. Huang, "Analysis of influence maximization in large-scale social networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 4, pp. 78–81, 2014.

[175] X. Tang and C. C. Yang, "Ranking user influence in healthcare social media," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 4, pp. 1–21, 2012.

[176] M. Newman, *Networks*. Oxford university press, 2018.

[177] Z. Li, X. Fang, and O. R. L. Sheng, "A survey of link recommendation for social networks: Methods, theoretical foundations, and future research directions," *ACM Transactions on Management Information Systems (TMIS)*, vol. 9, no. 1, pp. 1–26, 2017.

[178] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[179] 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*, 2011, pp. 281–292.

[180] S. Milgram, "The small world problem," *Psychology today*, vol. 2, no. 1, pp. 60–67, 1967.

[181] M. Draief and A. Ganesh, "Efficient routeing in poisson small-world networks," *Journal of Applied Probability*, vol. 43, no. 3, pp. 678–686, 2006.

[182] V. Senna, I. Vieira, R. Cheng, and S. Meacock, "Sexual transmission of hiv in a small world," *Operational research and its appications*, p. 503, 2003.

[183] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 867–876.

[184] A. Ukkonen, C. Castillo, D. Donato, and A. Gionis, "Searching the wikipedia with contextual information," in *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008, pp. 1351–1352.

[185] P. Singla and M. Richardson, "Yes, there is a correlation: -from social networks to personal behavior on the web," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 655–664.

[186] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: http://igraph.org

[187] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[188] J. P. Canning, E. E. Ingram, S. Nowak-Wolff, A. M. Ortiz, N. K. Ahmed, R. A. Rossi, K. R. Schmitt, and S. Soundarajan, "Predicting graph categories from structural properties," *arXiv preprint arXiv:1805.02682*, 2018.

[189] E. M. Airoldi, X. Bai, and K. M. Carley, "Network sampling and classification: An investigation of network model representations," *Decision support systems*, vol. 51, no. 3, pp. 506–518, 2011.

[190] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[191] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *The Journal of Machine Learning Research*, vol. 11, pp. 985–1042, 2010.

[192] K. Dadi, M. Rahim, A. Abraham, D. Chyzhyk, M. Milham, B. Thirion, G. Varoquaux, A. D. N. Initiative et al., "Benchmarking functional connectome-based predictive models for resting-state fmri," *Neuroimage*, vol. 192, pp. 115–134, 2019.

[193] J. M. Kleinberg, "An impossibility theorem for clustering," in *Advances in neural information processing systems*, 2003, pp. 463–470.

[194] C. Aine, H. J. Bockholt, J. R. Bustillo, J. M. Cañive, A. Caprihan, C. Gasparovic, F. M. Hanlon, J. M. Houck, R. E. Jung, J. Lauriello et al., "Multimodal neuroimaging in schizophrenia: description and dissemination," *Neuroinformatics*, vol. 15, no. 4, pp. 343–364, 2017. [Online]. Available: http://fcon_1000.projects.nitrc.org/indi/retro/cobre.html

[195] Z. Shehzad, A. C. Kelly, P. T. Reiss, D. G. Gee, K. Gotimer, L. Q. Uddin, S. H. Lee, D. S. Margulies, A. K. Roy, B. B. Biswal et al., "The resting brain: unconstrained yet reliable," *Cerebral cortex*, vol. 19, no. 10, pp. 2209–2229, 2009.

[196] C. Yan, D. Liu, Y. He, Q. Zou, C. Zhu, X. Zuo, X. Long, and Y. Zang, "Spontaneous brain activity in the default mode network is sensitive to different resting-state conditions with limited cognitive load," *PloS one*, vol. 4, no. 5, p. e5743, 2009.

[197] G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, and B. Thirion, "Multi-subject dictionary learning to segment an atlas of brain spontaneous activity," in *Biennial International Conference on information processing in medical imaging*. Springer, 2011, pp. 562–573.

[198] H. Richardson, G. Lisandrelli, A. Riobueno-Naylor, and R. Saxe, "Development of the social brain from age three to twelve years," *Nature communications*, vol. 9, no. 1, pp. 1–12, 2018.

[199] B. Thirion, G. Varoquaux, E. Dohmatob, and J.-B. Poline, "Which fmri clustering gives good brain parcellations?" *Frontiers in neuroscience*, vol. 8, p. 167, 2014.

[200] M. Gupta, R. Li, and K. C.-C. Chang, "Towards a social media analytics platform: event detection and user profiling for twitter," in *WWW*, 2014, pp. 193–194.

[201] A. Malhotra, L. Totti, W. Meira Jr, P. Kumaraguru, and V. Almeida, "Studying user footprints in different online social networks," in *ASONAM*. IEEE Computer Society, 2012, pp. 1065–1070.

[202] A. Birnbaum, "On the foundations of statistical inference," *Journal of the American Statistical Association*, vol. 57, no. 298, pp. 269–306, 1962.

[203] P. J. Carrington, J. Scott, and S. Wasserman, *Models and methods in social network analysis*. Cambridge university press, 2005, vol. 28.

[204] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 177–187.

[205] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[206] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *arXiv preprint arXiv:1911.06455*, 2019.

[207] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[208] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016118, 2009.

[209] 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, pp. 1–56, 2013.

[210] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal, Complex Systems*, vol. 1695, no. 5, pp. 1–9, 2006.

[211] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Summer School on Machine Learning*. Springer, 2003, pp. 169–207.