**Enhanced Random Walk Sampling over Online Networks Leveraging Topology Knowledge**

by Naian Yin

B.S. in Software Engineering, June 2012, Tongji University, China
M.S. in Technology and Society, May 2015, Stony Brook University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial satisfaction of the requirements
for the degree of Doctor of Philosophy

January 10, 2020

Dissertation directed by

Nan Zhang
Professor of Information Technology and Analytics

The School of Engineering and Applied Science of The George Washington University certifies that Naian Yin has passed the Final Examination for the degree of Doctor of Philosophy as of May 17, 2019. This is the final and approved form of the dissertation.

**Enhanced Random Walk Sampling over Online Networks Leveraging Topology Knowledge**

Naian Yin

Dissertation Research Committee:

Nan Zhang, Professor of Information Technology and Analytics, Dissertation Director

Xiuzhen (Susan) Cheng, Professor of Computer Science, Committee Member

Hyeong-Ah Choi, Professor of Computer Science, Committee Member

Howie Huang, Professor of Computer Engineering and Computer Science, Committee Member

© Copyright 2019 by Naian Yin
All rights reserved

**Acknowledgments**

I would like to express my sincere gratitude to many people in my mind who helped me all the way along this dissertation and the whole graduate study. All of the names listed or not are of subtle importance, without whom I will not able to go through the years.

First, I'd like to give most of the credit to my academic advisor and dissertation director Professor Nan Zhang who offered me the opportunity to start the PhD study and direct me all the way through my researches. He inspired me of great topics and ideas to explore in cross disciplinary field of computer science and statistics, and gave incredibly many useful insights in all aspects of systematic academic researches and writings, which forms me into a more established researcher.

Next, I'd like to thank my academic advisor in the last 2 years of PhD study, Professor Susan Cheng, who generously put her time and effort into my research works and study progress when Professor Zhang has to work remotely. I won't be able to accomplish my research work smoothly without Professor Cheng's supervision and suggestions.

Besides my advisors, I'm also more than grateful to the rest of my dissertation committee (Professor Hyeong-Ah Choi as the committee chair, and Professor Howie Huang) for their being strongly supportive to my research process and sharing previous advice. I'd also like to thank them for investing time into my pro-longed defense and graduation application period.

Then I would like to thank my fellow researcher, Yachao Lu. We spend several months together for one of the research accomplished in the dissertation. We shared a large portion of research interest on sampling techniques over large online networks and had great collaboration as well as fruitful research outcome. I also want to extend my gratitude to my lab mates, our department of Computer Science and SEAS graduate services for being continuously supportive and very responsive for any questions I have.

I also have a special thanks to my one and only favorite music artist, Jane Zhang, whose remarkable works and charisma have been fueling my soul all along the hard times.

Last but not least, my deepest gratitude is to my family and friends. This dissertation is an accomplishment of their endless support and love.

## Abstract

### Enhanced Random Walk Sampling over Online Networks Leveraging Topology Knowledge

Online networks, as digital images reflecting every aspects of real world especially under the current context of information technology, have become the most massive and opulent data source these days, where vast amount of information keeps being produced and transmitted. Their rapid expansion in scale and data volume draws wide attention and interests from researchers, while challenges emerge along as well.

The inherent growing demands of data fetching required by analysis conflict greatly with ON providers' efforts to protect their digital assets and considerations of user privacy. Restrictions of the web and other application interfaces of ONs prevent third party researchers from gathering sufficient data from the networks and the global image of these networks are also hidden.

Under such circumstances, very few existing analytic tools but the random walk family techniques can be adopted to fulfill ON related data tasks. They are capable of sampling network data distributed over desired distributions without knowledge of the entire graphs. However, their possibly intolerable and hard-to-foresee long 'burn-in' (or mixing time) before convergences also bring cost dilemmas that need to be conquered.

Reviewing existing sampling techniques being used over ONs, most of them are more generalized approaches that can be used on arbitrary networks, which inspired my work to find the possibility of speeding up random walks by leveraging some well established ON topology features. This dissertation includes my discussions of several speeded random walk like approaches standing on common characteristics of ONs.

Firstly, online networks are believed to be scale-free and follow power law in their degree distributions. It leads to the presence of very high degree nodes or "hubs" that a relatively small number of them can access nearly the whole network. The first idea is to perform random walks in the smaller sub-networks consisting of hubs. This reduces the size of ON

related problems and gives the walks quicker access to larger portion of non-hub nodes through hubs, seeking for faster mixing and exploration over networks.

And the idea of walking over hub nodes also carries potential to solve basic graph problems in topology ambiguous online networks context, where I think of the single pair path problem. It is one of those most basic element in graph analysis used to answer or estimate some important graph properties, but with topology unavailable it's lacking efficient approach to meet the growing demand. I take advantage of the presence of hub nodes mentioned above to boost the path searching speed, either in pure greedy approach or optimal guaranteed A* manner,and thus try to avoid the expensive query cost using existing methods like breadth first search.

Then I considered the very common community structure showed up in ONs (also known as clusters or modules). These are groups of nodes that share partially identical properties or serve similar roles within the networks, where existing random walks can easily get stuck in place because of low inter community connectivity. My idea is to leverage community information that pushes (by raising the probability of crossing edges) random walks to go across the "bottle neck" between different communities.

All ideas aim to fast raise the probability of exploring nodes from entire network and avoid wandering locally for too long.

## Table of Contents

**Acknowledgments** . . . . . **iv**

**Abstract** . . . . . **vi**

**List of Figures** . . . . . **xi**

**List of Tables** . . . . . **xii**

**List of Abbreviations** . . . . . **xiii**

**Glossary of Terms** . . . . . **xiv**

**1 Introduction** . . . . . **1**

1.1 Problem motivation . . . . . 1
1.2 Existing techniques and challenges . . . . . 2
1.3 The hub sampling framework . . . . . 4
1.4 Hub first search for single pair path problem . . . . . 5
1.5 Speeding up random walk by leveraging community affiliation information . . 6
1.6 Contribution outline . . . . . 7

**2 Preliminaries** . . . . . **9**

2.1 Graph model of online networks . . . . . 9
2.2 Sampling via random walks . . . . . 10
2.2.1 Markov Chains . . . . . 10
2.2.2 Random walk . . . . . 11
2.3 Conductance . . . . . 13
2.4 Graph generative models . . . . . 14

**3 Hub nodes prioritized graph exploring** . . . . . **16**

3.1 Problem definition . . . . . 16
3.2 Hub nodes random walk and sampler . . . . . 16
3.2.1 Basic design . . . . . 16
3.2.2 Technical challenges . . . . . 17
3.2.3 Determination of hub threshold . . . . . 18
3.2.4 Computing sampling probability . . . . . 19
3.2.5 Dealing with nodes with zero hub neighbors . . . . . 20
3.3 Algorithm and estimators . . . . . 21
3.3.1 Estimating total count . . . . . 22
3.3.2 Estimating average degree . . . . . 22
3.4 Evaluation . . . . . 23
3.4.1 Theoretical evaluation of mixing time . . . . . 23
3.4.2 Monitoring convergence using diagnostics . . . . . 25
3.4.3 Total count and degree estimators . . . . . 27

3.5 More discussion on avoiding poor conduction . . . 29

**4 Hub first search for single pair path problem . . . 31**

4.1 Motivation . . . 31
4.1.1 Idea of hub first searching . . . 32
4.2 Problem Definition . . . 33
4.2.1 Challenges . . . 33
4.2.2 Existing Techniques . . . 34
4.3 Hub first searching . . . 35
4.3.1 The A* search . . . 35
4.3.2 Baseline Design . . . 36
4.3.3 Theoretical analysis . . . 38
4.4 The greedy alternative . . . 41
4.4.1 Limitation of A* . . . 41
4.4.2 Greedy design . . . 42
4.4.3 Theoretical analysis . . . 44
4.5 Evaluation . . . 44
4.5.1 Experiment setup . . . 45
4.5.2 Experiment result . . . 46
4.6 Extended discussions . . . 49
4.6.1 Different network interfaces . . . 49
4.6.2 Other leveraging information . . . 51
4.6.3 Dynamically adjusted heuristics . . . 51
4.6.4 Definition of hubs . . . 52

**5 Community affiliation incorporated random walks . . . 53**

5.1 Motivation . . . 54
5.1.1 Speed up random walk by leveraging community affiliation . . . 55
5.2 Terminologies . . . 58
5.2.1 "Community-Structured" graph and community affiliation information . . . 58
5.2.2 Measurements of performance . . . 59
5.2.3 Graph generative model . . . 60
5.3 Community affiliation based random walk . . . 60
5.3.1 Basic design . . . 61
5.3.2 Theoretical analysis . . . 62
5.3.3 Algorithm . . . 67
5.3.4 Experimental evaluation . . . 68
5.4 Dealing with community inference . . . 75
5.4.1 Community detection algorithms . . . 75
5.4.2 Naive community affiliation inference . . . 75
5.4.3 Community affiliation inference with attribute selection . . . 76
5.4.4 Experiments . . . 78
5.5 Discussions . . . 81

**6 Conclusion and Discussions . . . 83**

6.1 Summary of accomplishments . . . . . 83
6.2 Summary of possible discussions . . . . . 84

**Bibliography** . . . . . **86**

## List of Figures


3.1 Mixing time boundaries vs difference tolerance BA model . . . . . 24
3.2 Mixing time boundaries vs difference tolerance Facebook data set . . . . . 25

4.1 Theoretical estimation of query number . . . . . 40
4.2 Graph G result . . . . . 47
4.3 Twitter result . . . . . 48
4.4 Facebook result . . . . . 49
4.5 Extra path length bring by Greedy . . . . . 50

5.1 A highly clustered large network . . . . . 56
5.2 Community affiliation incorporated random walk (CRW)'s choice of next hop 62
5.3 Simple random walk (SRW) and ORW's paths in Lancichinetti–Fortunato–Radicchi benchmark graph (LFR) graph . . . . . 68
5.4 Test results of LFR benchmark graphs . . . . . 72
5.5 Test results of real world data sets . . . . . 73
5.6 The Amazon product data of books under naive attribute selection . . . . . 79
5.7 The Amazon product data of books under Minimum redundancy maximum relevance algorithm (mRMR) . . . . . 81

## List of Tables


4.1 Generated BA graphs . . . . . 39
4.2 Tested data sets information . . . . . 45

5.1 The LFR graphs tested . . . . . 69
5.2 The Real World data sets tested . . . . . 70
5.3 mRMR scores of 5 attributes selection . . . . . 80

## List of Abbreviations

**BA** Barabasi-Albert 14, 24, 39

**BFS** Breadth first search 5, 6, 32, 34, 35, 38, 41, 44, 46, 48–51, 84

**CRW** Community affiliation incorporated random walk xi, 15, 55–57, 60–64, 67, 68, 71, 73, 74, 78, 79, 81, 83

**ESS** Effective sample size 60, 71, 73, 74, 80

**HFS** Hub first search 35, 36, 39–42, 46, 48, 52, 83–85

**HNRW** Hub node random walk 18, 21–25, 27, 28, 83

**i.i.d.** independent and identically distributed random variables 59, 71

**LFR** Lancichinetti–Fortunato–Radicchi benchmark graph xi, xii, 15, 60, 68, 69

**MCMC** Markov Chain Monte Carlo 10, 11, 16, 71

**MH** Metroplis Hastings 12, 13, 19, 62

**mRMR** Minimum redundancy maximum relevance algorithm xi, xii, 76–82

**ON** Online network vi, vii, 1, 2, 4–7, 11, 14–18, 22, 29, 31–36, 39, 41, 45–51, 54, 55, 57, 59, 61, 68, 69, 71, 83

**RW** Random walk xv, 3, 7, 11

**SPP** single pair path problem 32–35, 41, 46

**SPSP** single pair shortest path problem 33, 35

**SRW** Simple random walk xi, 4, 7, 12, 13, 17–20, 22, 27–29, 54, 56–64, 68, 73, 74, 77, 79, 81

## Glossary of Terms

**A*** is a family of informed graph path search algorithms that extend "best" path minimizing the estimation of the total cost from source to destination. ix, 6, 32, 35, 83

**breadth first search** is a traversal method on graph data structure that explore all nodes at current depth before start visiting nodes in the next depth layer. vii, xiii, 5, 31, 32

**community** is a well knitted collection of entities in a network that is loosely connected to the rest of the system. plural vii, 4, 53, 58, 83

**conductance** is a measure describing how well-knitted is a graph, it's a direct bound on how fast random walks can converge on the graph (i.e. a hard bound of mixing time). 3, 17, 31, 83

**edge** is a connection between a pair of nodes, it can be directed with one of the connected node be the source and the other be the destination or un-directed representing indifferent mutual relationship. vii, xiv, 1, 9, 16, 83

**effective sample size** is the number of i.i.d. samples required to get the same information as the set of correlated samples of interest. xiii, 59, 60, 71

**feature selection** is the selecting process of most relevant subset of features to be included in mathematical models. 7, 57

**graph** is a data structure represented by a collection of nodes and edges xiv, xv, 1, 9, 16, 31, 83

**mark and recapture** a method used to estimate population size by marking a portion of the population and placing them back. Estimation are made by recapturing a number of new samples and calculating the proportion of marked individuals. 3, 22

**Markov Chain** is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. viii, xiv, xv, 7, 10, 56

**Markov Chain Monte Carlo** is the family of sampling methods that construct Markov Chains with desired stationary distribution, and use the chains as samples. random walks on graphs are typical MCMC applications. xiii, 10

**Metroplis Hastings** is an MCMC algorithm that approximate any distribution of interest as its stationary distribution when direct sampling is difficult. MH use the property of reversible to construct the chain's stationary distribution. xiii, 7, 12, 56, 63

**mixing time** the time/steps Markov Chain needs until it's within a threshold to its stationary distribution, also known as "burn-in" time. vi, xiv, 3, 10, 13, 17, 54, 83

**node** is an object in a graph representing an entity, also known as a vertice. vi, xiv, xv, 1, 9, 16, 32, 83

**online network** are online platforms which are used to build or represent real-world entities and their relationships, it can be modeled as graphs vi–viii, xiii, 1, 3, 5, 9, 31, 53, 54

**power law** is a function relationship in the form of: $f(x) = ax^{-k}$. vi, xv, 4, 16, 60

**random walk** is a stochastic process describing a path consisting of a succession of random steps on some mathematical space (nodes in the context of this dissertation). Some of RWs falls into MCMC category. vi, viii, xiii–xv, 2, 4, 7, 10, 50, 53, 54, 60, 83

**reversible** is used to describe the property of a Markov Chain if a probability distribution $\pi$ exists that given $\pi$, the transition probability between any pair of states in the chain at any time will be balanced. It's the key condition for Markov Chain to have a stationary distribution. xiv, 7, 10, 56

**scale-free** is a graph property that the degree distribution of its nodes follows power law. vi, 4, 16, 32, 83

**simple random walk** is a random walk method that choose the next step uniformly at random, widely used in graphs related studies. xi, xiii, 3, 12, 16, 29, 31, 48, 54

**stationary distribution** is the probability distribution of a Markov Chain's states that will be reached and unchanged when time is sufficiently long. xiv, xv, 3, 10, 54

**transition matrix** describe the transition probability between states in a Markov Chain within the context of finite state space, known as transition kernel in the general theory of Markov Chains. 7, 10, 23, 63

# Chapter 1: Introduction

Social networks have long been serving a vital role in data analysis over a broad range of topics either for research or commercial purposes. Now with the occupying information technologies, modern life is stepping from physical world to the balance point of a parallel between real and virtual. Internet functions for considerably many daily essentials especially communication and information exchanging, which is to some extent reflected by the tremendous amount of data being produced, transacted and stored in online version of social networks.

Online networks have never been a more representative reproduction of real-world information and notably with semi-structured data objects. Comparing to the complicated natural language context in real world, ONs offer a more analysis friendly arena. As their scales' growing up, more interests rise to dig into these rich data sources, inherently making efficient analysis techniques over large networks an important topic to explore.

## 1.1 Problem motivation

With these being stated, respectable number of researches have already been cast to learn ON based data, [59] adding impressive and fruitful outcomes to practical applications ranging from demographic studies, locating trends in public affairs to economics or marketing purposed inferences. While challenges also emerges along with the inflating of ON scale and data volume, restricting the feasibility of many known statistical tools to be appropriately applied.

Mostly online networks are regarded as directed or un-directed graphs with real-world objects as nodes or vertices, and the mutual relationship between pairs of objects as identically weighted edges. In order to efficiently analyze ONs' buried information, very commonly we'll have to deal frequently with graphs of ON characteristics, answering global and

conditional aggregates such as sum, mean, and count (e.g., the average age of all users, total users of a community or entire network) , or detecting topology properties over them.

One major barrier for these tasks is the absence of global information from most ONs. [9][64] With data now being valuable digital assets and a rising awareness of privacy and security after many famous disclosures like the Cambridge Analytica scandal [54], there are plenty of reasons for ON providers to keep details of their networks away from the public. Unless the providers are willing to release public data, it is not possible to observe the network graphs in a global scope.

Hence in the very majority of practical problems involving ONs, third party researchers only have topical vision of the graphs from where they landed through provided web or other application interfaces. Query to a node in the networks barely reveals little information on its immediate neighborhood. Also for similar concerns and to prevent any malicious users from sending harmfully many requests causing performance issues, large ONs' providers make great efforts to enhance their web crawlers defencing features, limiting the number of queries or transactions can be committed in a certain time window from a single source. [13]

Under such circumstances approaches based on the retrieval of entire graph, considering the huge sizes of ONs nowadays, are wiped out due to the prohibitively expensive third party query cost. More practical data interfaces are calling for effective localized alternatives to tackle the challenges.

### 1.2 Existing techniques and challenges

To conquer these challenges, ONs involved studies resort to sampling methods that collect samples from networks following a pre-determined distribution. Estimations of desired data aggregation can then be calculated via sampled data used in place of ground truths. Few existing statistical approaches but the random walk family techniques, [20] being capable of sampling under designed stationary distribution and running locally without knowing

global topology, can fulfill the requirements and fit into practical interface limitations. More specifically, simple random walk, with its adaptability to different scenarios as well as ease of implementation, has been widely adopted and considered a golden baseline of the sampling techniques, on which I focused to evaluate my ideas in my dissertation.

However, one dilemma all RWs have to conquer is possibly intolerable long waiting time known as the "burn-in" periods (or mixing time) when large numbers of queries or transactions are required before RWs' sampling distributions can converge to the desired stationary distribution and start to draw samples. [1] Current works [18][27][43] on online network sampling are mostly topology oblivious techniques that do not intentionally leverage any special property of real-world large networks. Instead, they are sampling techniques for general-purpose graphs that, roughly speaking, work well so long as the underlying graph features a large conductance (which is in graph theory known to be a key factor affecting convergence speed for random walks).

Unfortunately, recent studies found that the conductance of real-world social networks such as Facebook, Livejournal, etc. is substantially lower than expected, leading to prolonged burn-ins.[59][32] One reason is that real-world social networks' tendency to exhibit highly clustered topology generates bottle-necks that are hard for random walks to cross through. [44][43] Without sufficient knowledge of entire networks, it's also difficult to foresee the mixing costs or monitor actual convergence.

In addition, it is exceedingly expensive to use these RW samplers to efficiently get count or sum estimators. Almost all solutions in the literature use a variation of the mark and recapture technique, [55] which uses the number of duplicate samples (in a sampling-with-replacement context) to estimate the total number of nodes $|N|$ in the interested networks, and thereby count/sum estimates. However, doing so generally requires $O(\sqrt{|N|}$ sample nodes to be drawn, incurring a considerably high query cost in practice. Similarly some very basic graph topology problems like find path between a single pair of nodes are especially costly and challenging due to lacking of global topology knowledge.

And besides the highly clustered and community-like structures mentioned, real-world large networks are also known to follow other certain properties (e.g. scale-free and power law distributed node degrees), which is under explored and rarely incorporated to solve ON related problems. [37][49]

In my works I tried to develop ON sampling and exploring techniques that leverage these special properties of real-world large networks to enable more efficient sampling and estimation. The next subsections will briefly introduce several ideas discussed in the dissertation.

### 1.3 The hub sampling framework

The first idea centers on the classification of nodes in a large ON into two categories, hub and non-hub nodes. While the precision definition of such a classification is a subtle issue that has to be addressed at length in the paper, here one can roughly understand hubs as those high-degree nodes that are the "center" of their local communities.

The power-law nature of degree distribution in a real-world network [49] indicates that hub nodes "cover" most of the edges in the whole system. We leverage this fact by devising a two-layered sampling framework. It starts with random walk that concentrates on only the hub nodes, followed by a second level sampling of non-hub nodes by following edges leading from their corresponding hub nodes.

The rationale here is that, since hub nodes form only a small fraction of all nodes in the interested network, hub-only sampling often requires random walks on only a small sub-graph. Thus, the first step of sampling tends to have a much faster mixing time than traditional ways, even the SRW, over the entire graph. On the other hand, since hub nodes cover a majority of edges, we can have an efficient second step on the hub-induced sampling of non-hub nodes.

Furthermore, since the number of hub nodes, say $|N_H|$ , is much smaller than the total size $|N|$ of the interested network, estimating count of hub nodes incurs a significantly lower query cost (as $\sqrt{|N_H|} << \sqrt{|N|}$ ). After "extrapolating" this count estimation to that of $|N|$

in the second-level sampling step, we can then enable count and sum estimations either with or without selection conditions in a much more efficient manner than the existing techniques. Meanwhile, we'll also need to address some issues brought by introducing additional steps.

- The challenge with applying this idea over any online networks, however, is also two-fold. First, we cannot expect a complete knowledge of the graph topology. This introduces many challenges to the two-level sampling framework. For instance, the precise and optimal definitions of hub and non-hub nodes should be theoretically studied and how should we determine in practical settings with trade-off, so as to determine or estimate the stationary sampling probability for the new sample approach.

- Even though the scale-free property is well-established for large networks, there are still significant variations between the topological properties of different real-world systems, demanding a self-adaptive design of the sampling framework. For example, one may have to use different definitions of hub nodes for different networks in order to ensure not only that hub nodes have well-knit connections among themselves but also that they cover a vast majority of edges with non-hub nodes.

### 1.4 Hub first search for single pair path problem

As mentioned before, many basic graph problems that call for a rich global topology knowledge are very hard and costly on large ONs due to underlying limitations. Single pair path problem is one of these required in sampling the graph conductance, diameter and other interesting properties which has been extensively studied in the context of traditional graph theory.[50]

Some of current studies use a complimentary result from the single source shortest path algorithms to address the problem which is unacceptably expensive in practical use on ONs. [46] Others stick to variations of Breadth first search (BFS) or two directional BFS, which expects a complexity of $O(\overline{d}^{\overline{Depth}})$. [4][42] After exploring the potential of taking use of

hub nodes in previous study, I believe these common features that has already been observed on ONs could be taken advantage of.

As discussed, the scale-free feature of ONs might offer reasonable assumptions during searching paths. Since a very small portion of hub nodes are well connected to rest of the graph, if hub nodes could be given priority in node expansion queries the search will be in a form of an A* algorithm that have access to greater number of nodes in the graph with less query cost and thus raise the probability of finding a path in early steps. [53] And further, tasks like detecting connectivity don't require a guaranteed optimal (shortest) path given a pair of interest, which means expanding known nodes in the order of descending degree can be a greedy strategy in place of the BFS approach to quickly find a possible path.

In the dissertation I introduce both an A* search strategy by giving priority to hub nodes in queries and still guarantee to find single pair shortest path, and a pure greedy search that always expand known hub nodes in order to find any existing path fast.

### 1.5 Speeding up random walk by leveraging community affiliation information

Then we consider a novel problem of how to significantly increase the conductance of the real-world networks with highly clustered topology graph by leveraging community information. Existing random walk approaches easily get stuck in place within one or several communities due to their low probability of crossing bottle-necks under clustered graph topology. [43]

Though there's hardly universally adopted definition of communities in terms of graph theory, they are generally considered as groups of nodes that are well knitted together while loosely connected to the rest of the system. Social communities in the context of social networks can be any subgroups of people are all friends to each other or have very view degree of separation. [35] To take use of the property, we give a definition of community and community structured graph within the context of our study, and proposed a way to improve the known poor conductance in these type of graphs.

Intuitively, raising the probability of crossing community steps in RW will naturally increase the conductance at the current community where the RW is in. Thus, by our definition, bottlenecks in community structured graphs like ONs between different communities can be conquered. Our twisted RW will be forced to jump to a new community when choosing next hop and if neighbors from different communities present. Though changing the transition matrix of random walk results in intractable stationary distribution, we explained in detail how the approach is still able to be re-balanced to a reversible Markov Chain [56] by Metroplis Hastings [57] so that identical stationary distribution as our baseline SRW can be guaranteed.

Further challenges include different extent of ON information available. Sometimes "perfect community affiliation information" is given with meta information from queries and the knowledge can well define community affiliation values ($C(x)$) for each node in graph. Then a direct application of our twisted random walk will be applied.

In cases that community affiliation information is not immediately available, we need to find workarounds and make inferences. In the work we make affiliation inferences via betting relationship between key attributes and the community value with feature selection algorithms. Though more precise community detection algorithms adopted in current literature might be better in the context of finding communities in graph, they either conflict with our lacking of topology knowledge or come with unacceptable query costs ($O(n^2)$ or more), [5] which are not considered in our study.

### 1.6 Contribution outline

In my work, I focused on designing topology based approaches to speed up traditional graph sampling, exploring and estimation techniques to be used over large ONs, and discuss the application of them being used to address graph related problems.

Contributions of the dissertation also include comprehensive sets of experiments over both synthetic data and real-world ON. [31]

After clarifying the terminologies and notations that will be used as well as briefly reviewing related concepts and contents in preliminaries Chapter 2. I'll detailed expand the hub sampling framework in Chapter 3, hub first search for single pair path problem in Chapter 4 and community affiliation enhanced random walk in Chapter 5. All come with explanation of the algorithms, theoretical analyses and experimental evaluations. Chapter 6 summarized the outcomes and give discussions on possible extensions that could be explored.

# Chapter 2: Preliminaries

In this chapter I'll clarify some notations used in the context as well as frequently referenced terms. Concepts tied to individual studies will be discussed further in their own chapters.

## 2.1 Graph model of online networks

In the dissertation, I consider all networks of interest to be un-directed graphs $G = (N, E)$, with $|N|$ nodes and $|E|$ edges. Any $x \in N$ represents a node interface that contains the objects' meta information (e.g., user name, profile, product info), and an edge $e(x,y) \in E, x, y \in N$ means there's an interaction between the two nodes (e.g. two users are 'friends', co-occurence of 2 authors). I use $\mathrm{d}(x)$ to denote the degree (i.e. number of edges incident to the node) of node $x$, and likewise $\mathrm{n}(x)$ as the set of nodes in x's neighborhood (i.e. $\mathrm{n}(x) = \{y | e(x,y) \in E, y \in N\}$).

The model is capable of summarizing major ONs in arbitrary categories. For directed interaction like Twitter's followee/follower-ship (followee as in edges and follower as out edges), the transformation to un-directed graph is simply taking unions of node's in and out neighborhood $(e(x,y) \vee e_(y,x) \rightleftharpoons e(x,y))$.

Mostly web interface of ONs in real world only allows local neighborhood queries. Namely I'll only have knowledge of queried nodes set $X_0$ and the neighborhood $n(x_0)$ of node $x_0$ for all nodes in $X_0$[64]. We also assume the knowledge of known nodes include other metadata that can be used for community affiliation decision, either explicitly or indirectly[40].

I'll also be using the H subscription to denote any similar concepts that belong to the induced sub-graphs from hub nodes mentioned in Chapter 1. For instance the number of hub nodes will be denoted as $|N_H|$, and this will be further discussed in Chapter 3.

## 2.2 Sampling via random walks

Then I'll take a brief look at majorly used random walk techniques that is naturally fitting to these restrictive web interfaces due to their capability of running locally yet gaining unbiased samples to certain distribution from the entire network.

### 2.2.1 Markov Chains

Random walks usually stands on the general theory of Markov Chains, which is a stochastic process that satisfied the memory-less Markov property. The next state of a chain is conditional solely on its current state, which means knowing the whole process will not make any difference to the chain's future steps. A Markov Chain is said to be reversible when there exists a distribution $\pi$ that given $\pi$, a balance equation (see details in Equation (2.1)) is satisfied for any pair of states x,y in the chain's state space. $\pi$ is also the stationary distribution of the Markov chain that the sampling distribution will converge to when time progresses and then remain unchanged. Transition matrix or transition kernel of a Markov Chain is the square matrix that describe pair wise state transition probability on the state space of the Markov Chain. [56]

*Markov Chain Monte Carlo (MCMC)* is the set of sampling algorithms that construct Markov Chains whose stationaries are probability distributions of interest but hard to be directly sampled from. By taking use of converged Markov Chains as samples, inferences and estimations can be made accordingly. [16] It should be noted that the samples collecting from MCMC are correlated and may introduce errors, [11] and a mixing time largely depends on the underlying state space and states transitions is also required to be invested before useful samples can be drawn.

The state space of Markov Chains can be either continuous or discrete, while in the dissertation I'll focus on discrete space, or more specifically the node spaces of graphs. Random walks on graphs are mostly MCMC algorithms sampling from these node spaces.

### 2.2.2 Random walk

Given a graph G and a starting node $x_0$, RW select the next node $x_1$ from its neighborhood $n(x_0)$ according to a transition matrix $P(x,y), x,y \in N$ and then query the selected node to gain its neighborhood. The process would be repeated for a number of iterations until desired samples are drawn. This is a MCMC approach with $N$ as the state space. I will then review some important properties of random walks. [20].

#### 2.2.2.1 Stationary distribution

The sequence of nodes selected is a random walk on $G$ with finite state space $N$ and $P(x,y)$ being its irreducible and aperiodic transition matrix. Because these random walks are finite Markov chains that are time-reversible, as long as a random walk can reach all nodes in $N$ in finite time space (depending on how $P$ is chosen), the probability distribution for the walk to land on each node will converge to a *stationary distribution* $\pi$ after a number of 'burn-in' steps (or mixing time)[16], which is then used as the sampling distribution in many analysis works. By the condition of Markov chain convergence[33], the *edge measure* Q(x,y) which is the probability of moving from node $x$ to $y$ at stationary distribution, should be identical among all node pairs. Namely:

$$Q(x,y) = \pi(x) \cdot P(x,y)$$

$$Q(x,y) = Q(y,x), e(x,y) \in E \tag{2.1}$$

The above equation is also known as the local balance equation. [56] The theory of stationary distribution provides foundation for most existing analysis tools for ON sampling.

#### 2.2.2.2 Simple random walk

Among the popular random walk techniques, simple random walk is still the one considered as a golden baseline[18]. It selects the next-hop node $y$ uniformly at random among the neighbors $n(x)$ of the current node $x$. More specifically, for SRW whose transition matrix $P$ is simply:

$$P(x,y) = \begin{cases} \frac{1}{\mathrm{d}(x)}, & e(x,y) \in E \\ 0, & else \end{cases} \quad x,y \in N \tag{2.2}$$

Its stationary distribution is proportional to the node degrees: $\pi(x) \propto \mathrm{d}(x)$, and would be given by:

$$\pi_{\mathrm{SRW}}(x) = \frac{\mathrm{d}(x)}{2|E|} \tag{2.3}$$

#### 2.2.2.3 Metropolis Hastings based random walk

Metropolis Hastings based random walks are applications of the Metroplis Hastings algorithm [57] that is capable of correcting transition probabilities and achieve convergence to some specifically designed stationary distributions. In Metroplis Hastings (MH), a *proposal distribution* (an initial transition matrix without correction) $g(x,y)$ is chosen, and it come with an acceptance probability matrix $\alpha(x,y)$ calculated by the condition mentioned in Equation (2.1). Starting from node $x_0$, a candidate move is generated from the proposal $g(x,y)$ in $n(x_0)$. Let the candidate be node $y$, the move is then censored with the probability $1-\alpha(x_0,y)$. That is, with probability $\alpha(x_0,y)$, $y$ is "accepted" as the next state $x_1$, and otherwise $x_0$, with the remaining probability $1-\alpha(x_0,y)$, is adopted as $x_1$ and y is dumped. The detailed calculation of MH's transition matrix $P_{\mathrm{MH}}(x,y)$ would be given:

$$\alpha(x,y) = \begin{cases} \min[\frac{\pi(y)g(y,x)}{\pi(x)g(x,y)}, 1], & \text{if } \pi(x)g(x,y) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

$$P_{\mathrm{MH}}(x,y) = \alpha(x,y) \cdot g(x,y) \tag{2.5}$$

In this manner we'd be able to design both proposal and stationary distributions as desired that fits to our motivation. In fact, one notable application of MH in random walk techniques is to curve SRW's stationary to a uniform distribution on $N$. In the dissertation, I'll be comparing our experiment result using MH when the stationary distributions of my approaches are not identical to SRW or intractable for fair and strict comparisons.

#### 2.2.2.4 Mixing time

As stated, random walk requires a "burn-in" period before its convergence, also know as the mixing time, which is regarded as a crucial measurement of performance. It is the time required by a random walk for the distance to stationary to be sufficiently small (a pre-determined threshold $\varepsilon$). The mixing time is defined by:

$$t_{mix}(\varepsilon) = \min\{t : \max_{x,y \in V, y \in n(x)} |P^t(x,y) - \pi(y)| < \varepsilon\}, \tag{2.6}$$

where the threshold $\varepsilon$ is usually set to $\frac{1}{4}$. [33] Mixing time captures the number of steps a random walk needs to converge to a satisfying state from starting state. Less mixing time is an indication of faster sampling and less auto-correlation (i.e. higher quality samples).

## 2.3 Conductance

*Graph conductance* is also called bottleneck ratio, which is a graph property known to hard bound the mixing time described before. Definition of commonly used term conductance can be summarized as following. [6] For a graph $G$, a cut $(S,\overline{S})$ is a partition of $N$ into two disjoint subsets. We denote the conductance of cut $(S,\overline{S})$ in a graph $G$ as:

$$\varphi(S) = \frac{Q(S,\overline{S})}{\pi(S)} \tag{2.7}$$

The conductance of the whole graph is the minimum conductance over all the possible cuts:

$$\phi(G) = \min_{S \subset N, \pi(S) \leq \frac{1}{2}} \varphi(S) \tag{2.8}$$

The lower bound $\phi(G)$ places on mixing time $t_{mix}$ satisfies the inequality [33] of:

$$t_{mix} = t_{mix}(1/4) \geq \frac{1}{4\phi(G)} \tag{2.9}$$

Thus conductance can be a direct indicator for random walk performance. And it should be noted that conductance of different stochastic processes with regard to the same target graph $G$ might be different, and we'll be using notations with subscripts (e.g. $\phi_{\mathrm{SRW}}(G)$) to differentiate among those conductance measures.

## 2.4 Graph generative models

For latter use of theoretical analysis and easier experiments, simulation of ONs from graph generating models are required before step into larger real world data sets. Considering the scale-free feature of ONs, special cases like trees, circles or random graphs are not appropriate.

In traditional graph theory, a number of specific type of graph models have already been well studied and used to address different problems in real production. Some of these includes trees, circles, barbell graphs, random graphs and so on. In the dissertation I mainly use two types of models to capture the properties I need in ONs.

Preferential attachment models are the widest adopted graph generative methods when simulating ONs, sharing the idea of adding new nodes and generating edges using probability proportional to chosen node preference values. Though many variations have been proposed, I choose to implement the very basic Barabasi-Albert (BA) model. It adds a new node to graph with k edges attached to existing node set $X$ each time. The attraction of each existing node $x_i$ is proportional to its current degree $\mathrm{d}(x_i)$. Identical initial values are then added

with the attraction to ensure non-zero attaching probability to each node. An initial set of $X_0$ vertices are generated randomly before adding nodes. [2]

The LFR model [28] captures both power law degree distribution the community structure in ONs. It first generate an $|N|$ node graph having the degree sequence with power law exponential $\tau_1$ at a mean degree of $\overline{\mathrm{d}(x)}$. A mixing parameter $\mu$ is chosen to decide the fraction of edges for each node that is inner-community. Community sizes are then generated by power law exponential $\tau_2$, sum of which is equal to the node count $|N|$. The minimal and maximal sizes should satisfy: $min(|C_i|) > min(\mathrm{d}(x))$ *and* $max(|C_i|) > max(\mathrm{d}(x))$. Nodes of the graph are randomly assigned to some community if community size is not violated. The leftover nodes are then iteratively been assigned to random communities and kick random nodes out as new leftovers if target community is full until no nodes are alone without a community assignment. Final step of LFR is a rewiring of each nodes to meet the mixing factor $\mu$ of inner-community edge ratio without change the degrees. As the model has complimentary community affiliation information and capture major features of large ONs, it's naturally a good candidate for my idea of Community affiliation incorporated random walk (CRW) that will be discussed in Chapter 5.

# Chapter 3: Hub nodes prioritized graph exploring

## 3.1 Problem definition

As stated in the introduction, sampling through ONs in current literature are mostly accomplished by topology oblivious techniques like simple random walk. Without intentionally considering graph characteristics, these processes might not work as efficiently as expected in many complicated real world networks. It is not surprising that topological knowledge of the underlying graph can significantly speed up random walks. For example, prior work have shown that, if we know the graph topology to be a tree or an expander graph, then specially designed random walks can generate samples far more efficiently than general purpose random walks.[52]

In this section we aimed at leveraging the known scale-free feature in ONs and design an asymptotically unbiased sampling algorithm accordingly.

It should be noticed that biases cannot be reduced to 0 for MCMC processes in general due to the diagnostics we choose to monitor the chain's convergence. Even if starting from stationary distribution, the terminating distribution will be biasd. [11]

## 3.2 Hub nodes random walk and sampler

### 3.2.1 Basic design

We first consider the normal node and degree interface where our solution will be based, which means one query request to a single node will return the neighbor nodes list as well as their corresponding degrees.

As discussed in Chapter 1, the power law nature of degree distribution in scale-free networks like ONs indicates the presence of hub nodes which are a small number of nodes in graph with relatively high degrees. They can "cover" most of the edges in the underlying network,

and tend to be well knitted to each other.

Let's suppose the hub nodes of an ON form an induced sub-graph over G, say $G_H = (N_H, E_H)$, whose definition is set by a pre-determined threshold $\varepsilon$. Ideally if the threshold $\varepsilon$ is set properly, we shall expect $G_H$ a satisfying conductance, significantly reduced graph size comparing to G, as well as an entire coverage to non-hub nodes. Moreover if we think of some real-world ON instances like Twitter or Instagram, finding some arbitrary subsets of hub nodes is quite simple. Most of these networks place their popular users (i.e. the high degree nodes) in the recommendation lists or ranking pages.

Under the context, random walk over $G_H$ is heuristically reasonable substitute for traditional simple random walk not only for relieving expensive mixing time or potential access to nearly all nodes, but also it is not difficult to start with.

The basic process could be summarized as:

- Determine a definition of hub nodes either by simply a degree cutoff $\varepsilon$ or adding more criteria. Here we start from naively choosing a fixed threshold $\varepsilon$ by observing a pilot random walk on the graph. The aim is to get a rough estimation of degree distribution, and manually set a reasonable number.

- Arbitrarily pick a hub node from the sources mentioned before, start from the hub, and perform SRW over $G_H$ to sample a hub node $x$. This first layer generates hub samples with probability proportional to their hub degree $\mathrm{d}(x)$.

- Sample a non-hub node y from x's direct neighborhood.

### 3.2.2 Technical challenges

The solution enables us to sample nearly all nodes from G by walking on an only a small portion of it. However, flawless graph that perfectly fit our assumptions can hardly be the practical cases. We could encounter technical challenges when implementing the idea in real world ONs:

- Simple random walk provides all nodes with sampling probability proportional to the degrees, while in this approach our hub sampler cut the graph into two parts and sample them separately with two different distributions. Thus using the raw sample directly as usual might not be appropriate approach for statistical purposes.

- Only in ideal cases can we hope an entire coverage from $G_H$ to G, but it is more likely that, in practical implementation, more or less vertices of G will have no hubs as their immediate neighbor. This raise a bias for the sampler. And also we will want to set the threshold $\varepsilon$ high enough to get sufficiently small $G_H$, so that our approach sees significant domination over traditional SRW. However naturally higher cutoff leads to lower coverage.

- Another problem might be a violation to our conduction assumption. Random walk can incur failure due to the disconnection of $G_H$ caused by improper $\varepsilon$ setting or the topology of original graphs. And further with restrict view and limited beforehand knowledge of the graph, it's even hard to merely realize the problem.

How well we can address these issues determines the feasibility of Hub node random walk (HNRW) in specific tasks.

### 3.2.3 Determination of hub threshold

The setting of hub definition is a key factor of HNRW's performance. In this section we consider hubs as high-degree nodes in graph with a preset degree threshold $\varepsilon$. The induced sub-graph determined by the threshold should be as small as possible without losing conductance.

It is known that the induced sub-graph consisting of nodes whose degree is greater than $\sqrt{|N|}$ from a power law distributed graph with power k$\geq$ 3 (ONs fall into this criteria in general [49]) also follows power law with k'=k-1. [23] And study has shown that the conductance of power law networks has a distribution $\phi_G(x)\ x^{1-2k}$, [22] which means choosing $\varepsilon$ at $\sqrt{|N|}$

would be safe enough to get sufficiently small hub graph with better conductance.

However, since the total count of nodes are unknown this can be hardly implemented in practice. One may run a SRW and try to observe the first recaptured node and use it as an indicator of $\sqrt{|N|}$, [55] but this not only carries great variance, the cost of recapture is hard to foresee as well.

A simple alternative to tune value of $\varepsilon$ is to run a pilot SRW over the graph, and find the min value of highest degree neighbors for all nodes along the path. Namely set

$$\varepsilon = min_{x \in SRW}(max_{y \in \mathrm{n}(x)}(\mathrm{d}(y))) \tag{3.1}$$

We implement the heuristics in our experiment for its ease of use. It should be noted that the absolute min value might be very low due to outliers in the network not well connected to the system. There's no sense to use the absolute min in these cases as its no longer a good indicator, thus quantiles are used instead.

### 3.2.4 Computing sampling probability

Since for a known distribution, it is possible to curve it to any desired distribution so as to perform proper estimation by MH as discussed before. The only challenge raised by two-layer sample is actually computing the sampling probability for both of them. For the first layer concerning only hubs the answer is quite clear, so the way we sample non-hub nodes is the key point. Our sampler picks y uniformly at random in the sampled hub x's non-hub neighborhood.

Similar to traditional SRW, we shall note that the sampling probability of sampled hub x in first layer actually depends on its number of hub neighbors $\mathrm{d}_H(x)$. If we denote the total edges between hubs by $|E_H|$, this probability would be given by:

$$p(x) = \frac{\mathrm{d}_H(x)}{2|E_H|} \tag{3.2}$$

Then if we consider the non-hub nodes, a sampler y will be sampled only when one of its hub neighbors is chosen in last step. Thus, sampling probability for y depends on the total sampling probability of its hub neighbors $\mathrm{n}_H(y)$. Since y is picked uniformly at random from x's non-hub neighborhood, denoting the non-hub neighbor number of x by $\mathrm{d}_N H(x)$ we can than derive it as:

$$p(y) = \sum_{x \in \mathrm{n}_H(y)} \frac{\mathrm{d}_H(x)}{2|E_H|\mathrm{d}_N H(x)} \tag{3.3}$$

Remember that we suppose a neighbor-degree interface here, so to get all the information needed we will need to query each hub in $\mathrm{n}_H(y)$. This would only cause an extra cost of $\mathrm{d}_H(y)$.

### 3.2.5 Dealing with nodes with zero hub neighbors

Achieving asymptotic un-biasedness here is simple, by introduce a separate SRW instance we will ensure non-zero sampling probability for nodes with zero hub neighbor.

Actually we've run a pilot SRW for suitable $\varepsilon$ setting, which could be used further for our purpose here. Within this SRW instance, we will be able to estimate the percentage of nodes that are not hubs-doable $r_N HD$ by:

$$r_N HD = \frac{|N_{NHD}|}{|N|} = \frac{E(p_{NHD}(x)^{-1})}{E(p(x)^{-1})} = \frac{E(\mathrm{d}_{NHD}(x)^{-1})}{E(\mathrm{d}(x)^{-1})} \tag{3.4}$$

We can simply reject all hubs and hub-doable nodes in pilot run to get a pure sample with these nodes having zero hub neighbors. Again the distribution for this separate sample would be proportional to the node degree and can be curved and assembled to desired. E.g. The percentage we've calculated could be then set as the rejection rate for our sampler if the target is uniform. And estimations could also be based on the ratio aggregating this separate sample.

Possible downside with this simple solution is the potentially high rejection cost introduced

by the small sampling probability for nodes with zero hub neighbor, which means we might walk extremely long to actually sample one of these nodes.

### 3.3 Algorithm and estimators

Traditional simple random walk pick randomly at uniform from node x's neighborhood with equal probability $\frac{1}{\mathrm{d}(x)}$. Similarly, first layer of hub sampler's walking over $G_H$ takes $\frac{1}{\mathrm{d}_H(x)}$ probability to walk to a hub neighbor from the current hub x. And after enough steps when hub $x^*$ is sampled, its non-hub neighbor can be again pick at randomly with identical probability $\frac{1}{\mathrm{d}_N H(x)}$. Assuming a pre-determined $\sigma$ is set to be the boundary of reaching asymptotically convergence over $G_H$, under a fully observable graph topology the hub node random walking algorithm can be simply described as below.

**Input** : $x_0 \in N_H$

**Output :** paired sample set $(\mathscr{X}, \mathscr{Y}) = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)), n > 0\}$

Initialization;

Node $x = x_0$;

**while** $\frac{|P^t(x) - \pi(x)|}{\pi(x)} \geq \sigma$ **do**

    *run random walk on* $G_H$ $x = \mathrm{n}_H(x)[random(0, \mathrm{d}_H(x) - 1)]$;

**end**

**for** $i = 1 \rightarrow$ *Sample_size* **do**

    *Send a query to node x, GET attributes*;

    *Record node* $x \rightarrow x_i$;

    *Sample a non hub node y* y=$\mathrm{n}_{NH}(x)$[random(0,$\mathrm{d}_{NH}(x) - 1$)];

    *Send a query to node y, GET attributes and record required data*;

    *Record node* $y \rightarrow y_i$;

**end**

*Make estimation or inference based on Equation* (3.2)*, Equation* (3.3)

**Algorithm 1:** HNRW

### 3.3.1 Estimating total count

As mentioned in Chapter 1, it's especially hard for large ONs to get efficient count or sum estimations. A number of variations of mark and recapture approaches have been applied. [55] We incorporate the previously discussed HNRW into the popular collision count estimation [36] to give our estimator on the total node count $|N|$ Collision count estimator uses the duplicated observations in a serialized sample to estimate the total population size, which is good fit for our approach.It's given by:

$$|N| = \frac{\sum_1^{|S|} Wi * \sum_1^{|S|} \frac{1}{Wi}}{2} \tag{3.5}$$

Where |S| is the sample size. From Equation (3.5) we can easily derive accordingly the $|N_H|$ estimator of HNRW as:

$$|N_H| = \frac{\sum_1^{|S_H|} \mathrm{d}_H * \sum_1^{|S_H|} \frac{1}{\mathrm{d}_H}}{2} \tag{3.6}$$

It should be noted that, in total count estimation we can simply aggregate $|N_H|$ with the total to hub ratio that can also be estimated during the pilot SRW mentioned before, without sampling another non hub node. This ratio can be calculated as:

$$r_H = \frac{|N|}{|N_H|} = \frac{E\left(\frac{1}{p(x)}\right)}{E\left(\frac{1}{p(x)} I_{x \in N_H}(x)\right)} = \frac{|S_H| \sum \frac{2|E|}{\mathrm{d}}}{|S| \sum_{x \in N_H} \frac{2|E|}{\mathrm{d}}} = \frac{|S_H| \sum \frac{1}{\mathrm{d}}}{|S| \sum_{x \in N_H} \frac{1}{\mathrm{d}}} \tag{3.7}$$

Getting the product of Equation (3.6) and Equation (3.7), the estimation of total node count $|N|$ can be given by:

$$|N| = |N_H| \cdot r_H \tag{3.8}$$

### 3.3.2 Estimating average degree

We are also interested in making mean inferences which are extremely useful when ground truth is not available. SRW comes with established estimator as well as promising performance on topology in-variant variable such as mean degrees. The calculation can be given

by:

$$D = \frac{2|E|}{|N|} = \frac{2|E|}{E\left(\frac{1}{p(x)}\right)} = \frac{|S|}{\sum_1^{|S|} \frac{1}{\mathrm{d}(x_i)}} \tag{3.9}$$

Due to the different transition matrix, the estimation from HNRW should follow the sampling probability in Equation (3.2) and Equation (3.3) and can be derived as follows:

$$p(x) = \sum_1^{||} \frac{\mathrm{d}_H}{2|E_H| \times d_{NH}} D = \frac{2|E|}{N} = \frac{2\left(|E_H| + |E_{NH}|\right)}{|N_H| + |N_{MH}|} \tag{3.10}$$

where:

$$|N_{NH}| = E\left(\frac{1}{p_{NH}}\right) = 2|E_H|\left(1/\sum_1^{\mathrm{n}_H} \frac{\mathrm{d}_H}{\mathrm{d}_{NH}}\right) |E_{NH}| = E\left(\frac{d_{NH}}{p_{NH}}\right)/2 = |E_H|\left(d/\sum_1^{\mathrm{n}_H} \frac{\mathrm{d}_H}{\mathrm{d}_{NH}}\right) \tag{3.11}$$

Combining Equation (3.10) and Equation (3.11), we have the mean degree estimator.

## 3.4 Evaluation

### 3.4.1 Theoretical evaluation of mixing time

It is known that for a reversible, irreducible and aperiodic Markov Chain, at step t the boundaries when random walk starts from node s yields: [33]

$$|\frac{P_t(x,s)}{\pi(s)} - 1)| \leq \frac{(1-\gamma)^t}{\pi_{min}}$$

where $\gamma$ is the spectral gap of the graph given by $1 - \lambda_2$ (The greatest eigenvalue of the transition matrix P strictly smaller than 1). Let $T_{rel} = \frac{1}{\gamma}$, then for the random walk on G, the $\sigma$ mixing time would be bounded by:

$$t_{mix} \leq \log(\frac{1}{\sigma\pi(min)})T_{rel}$$

And also the lower bound is given by:

$$t_{mix} \geq (T_{rel} - 1)\log(\frac{1}{2\sigma})$$

Similarly for HNRW the upper bound of mixing time for the first layer is:

$$(T_{hrel} - 1)\log(\frac{1}{2\sigma}) \leq t_{hmix} \leq \log(\frac{1}{\sigma\pi(hmin)})T_{hrel} \tag{3.12}$$

Here we generate a scale-free random graph from the well-known BA model with $|N| = 2000, k = 5, m_0 = 10$. And compute numerically for mixing time to compare HNRW sampler to SRW. A small Facebook dataset with $|N| = 4039$ and $|E| = 88234$ is also used to see real-life performance. We can observe from Figure 3.1 that our Hub node random walk

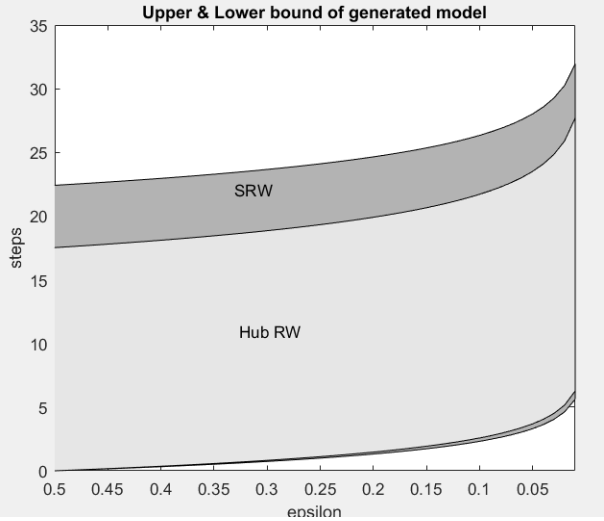



Figure 3.1: Mixing time boundaries vs difference tolerance BA model

(HNRW) expects quicker convergence theoretically on the BA model. Though dominance is not significant, but we shall notice that we'll have one hub sample together with one non-hub each time. Further, walk on the hub also quickly go through all hubs when the sample size grows. Thus the actual query cost will be bounded by the hub number. For this graph the $|N_H|$ is only 78 comparing to the total nodes 2000 by setting the threshold to be 20.

Figure 3.2 shows the same analysis under a subset of Facebook. Obviously the theoretical

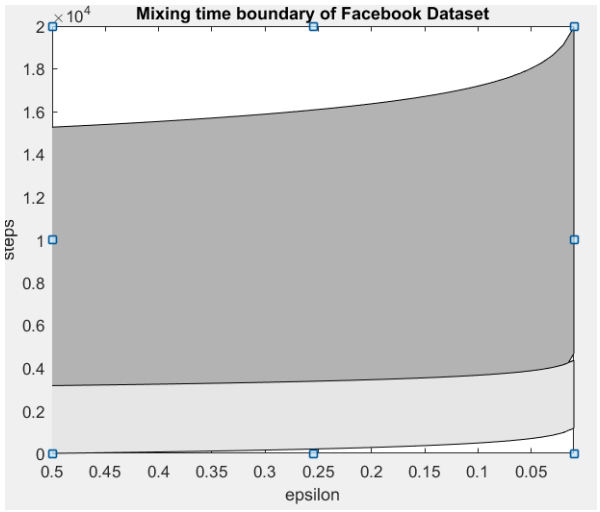



Figure 3.2: Mixing time boundaries vs difference tolerance Facebook data set

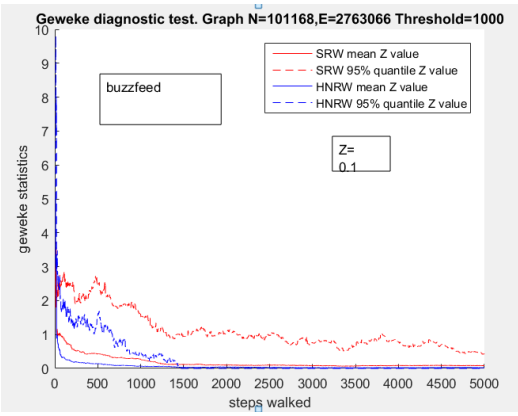
boundaries see significant reduction by doing HNRW this time. This could be a hint that in real world networks the power law is even more powerful so that the degree distribution expects sharper peak and is heavy-tailed, which benefits our algorithm.

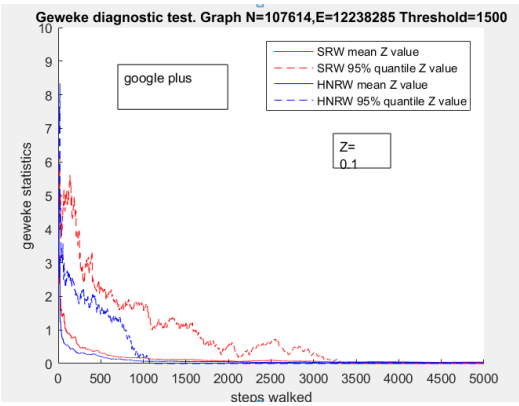
### 3.4.2 Monitoring convergence using diagnostics

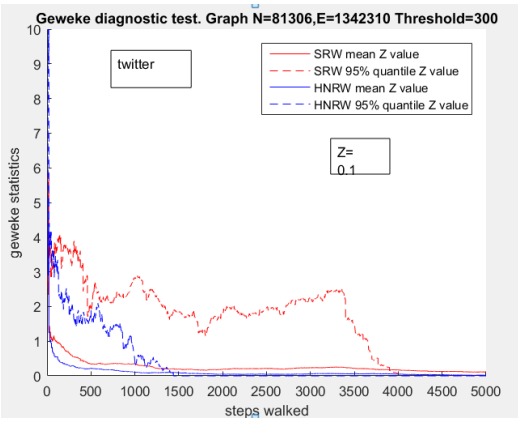
Though mentioned above that introducing convergence diagnostics will add bias to the result sample, for unknown topology heuristic approaches are still only candidates to monitor asymptotically stable. In our experiment over real-world network data sets, we choose Geweke diagnostics which follows: [10]

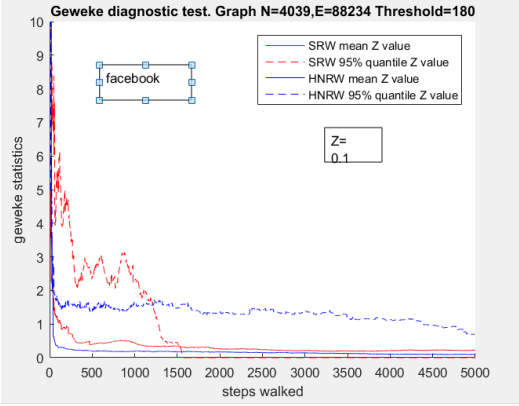
$$T = \frac{|\theta 1 - \theta 2|}{\sqrt{\frac{s1}{n} + \frac{s2}{m}}}$$

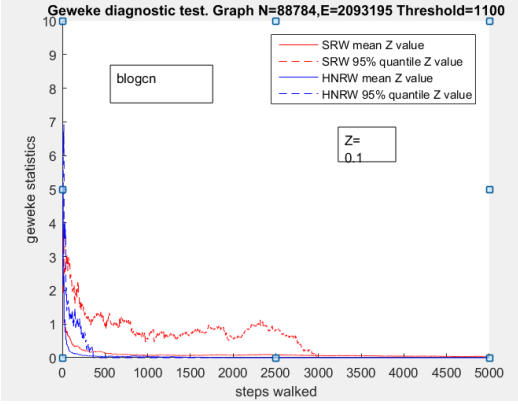
With $\theta$ being set to an desired small value to meet different accuracy requirements.

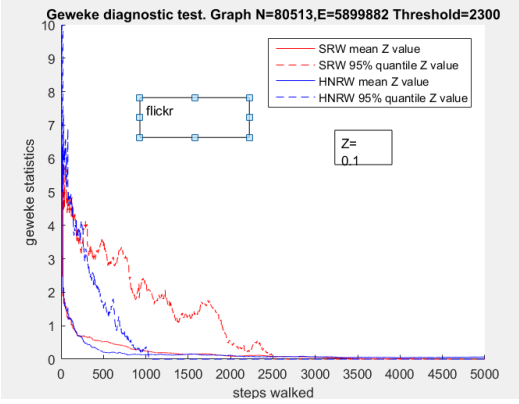


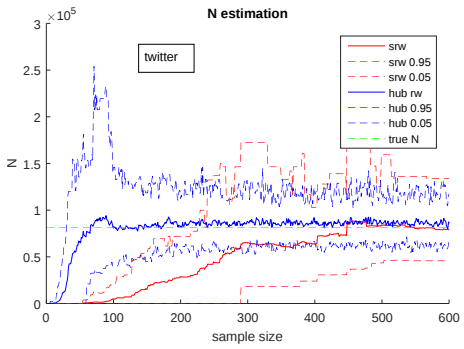



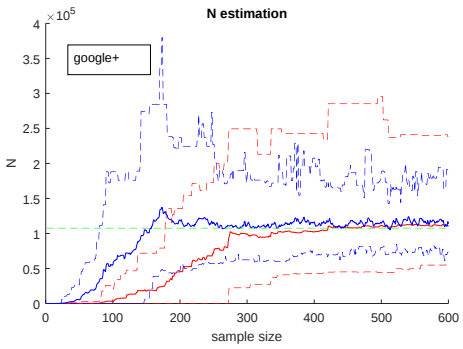

We monitor the actual convergence speed over different real world data sets from SNAP

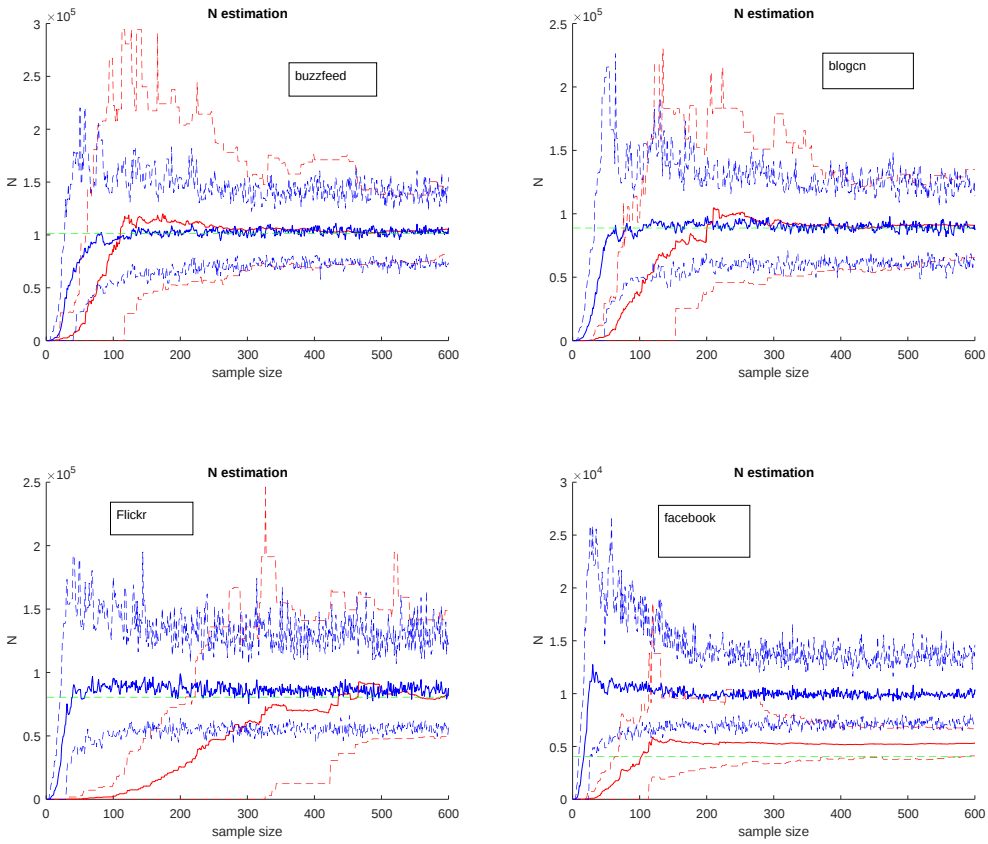
repositories, [31] observing much quicker convergence on data sets from Buzzfeed, Twitter, Blogcn and Google Plus as we expected. While there's not significant dominance in mixing time on Facebook and Flickr even if tuning $\varepsilon$ to a rather high value. We plotted the degree distribution and observe their corresponding degree distribution and conclude this might be a result of less skewed distribution in data sets, so that either the presence of hubs is weaker or the diagnostics itself is no longer sensitive. (It should also be noted that it can be a false negative due to the data collection processes. All the data sets we used in experiments are only a subset of the real networks, which means bias can be introduced.)

### 3.4.3 Total count and degree estimators

Since we have the ground truth of each data set on hand, the performance of estimators can be evaluated simply by their cost versus the error to true value. We use the estimators derived in previous sections and run HNRW for 1000 replications over each data set. We calculate the actual query cost for each HNRW and run cost equivalent SRWs to make estimations. The comparison results are plotted with mean value, 5% and 95% quantiles from the 1000 trials on different data sets.






The results of total count estimators is very promising that we observe good convergence to ground truth which confirms the approximate un-biasedness. (Facebook as an exception due to the topology issues mentioned before) And the HNRW converges much quicker than SRW so that we would be able to get reasonable estimation with little cost.

Dominance of HNRW in degree estimation is not that significant though the estimators are able to be asymptotically unbiased. The reason is not hard to guess. SRW has its proven efficiency on degree estimation. [18][27] And further the Geweke diagnostics we used is also with regard to the node degrees, which means the degree distribution will have complimentary convergence at the terminating point while other attributes, if available, might not. We might use data sets with other attributes for more tests, while the result from the set of expriments is powerful enough to prove the efficiency of our proposed HNRW

over ONs. We'll discuss some extension on the conduction issue in the next section.

### 3.5 More discussion on avoiding poor conduction

In previous study we intuitively fix the $\varepsilon$ by observing a pilot simple random walk sample. Since the main causes of poor conduction comes from possible improper setting or graph topology, we will want our solution to be more tolerant towards cases that $\varepsilon$ has to be changed or can hardly be pre-determined.

And such problems should be located as quick as possible at the beginning. Otherwise the effort and cost placed upon improper hub definition will be totally wasted, if every time $\varepsilon$ changes new experiments have to be done from the start.

Challenges here is mainly finding accurate $\varepsilon$ and fast detection of disconnection when it occurs. A good way would be roughly verify the level of $G_H$ conduction under current hub definition before experiments actually start and adjust accordingly, namely dynamically adjusting the hub definition.

To be more specific, we could perform the dynamic adjustment by monitoring not only the sampling probability, but also connectivity of hub induced sub-graph in the pilot SRW instance mentioned before. We can pick set of nodes that satisfied our current hub definition from all nodes walk through by this long run, and each starts a short walk on the current $G_H$ until hit another node in this set. By doing so we can group the hubs to see if the conduction is satisfying and keep or update the $\varepsilon$ .

Note that we can adjust not only the degree cutoff $\varepsilon$ on hub definition, but even change the degree-based definition. E.g., by including as hub nodes those with a lower degree

but a large number of hub neighbors. Once we found even lower $\varepsilon$ cannot provide good conduction, introducing high hub degree nodes will possibly help to connect hubs together.

# Chapter 4: Hub first search for single pair path problem

After exploring the possibility of taking use of specific graph topology of ONs, I think of several basic graph problems that might be benefits from similar strategy, from which the single pair shortest path is one of those ideal candidates. [50]

## 4.1 Motivation

As introduced, online networks are often regarded as directed or un-directed graphs with users or user groups as nodes, and the following/followeeship as identically weighted edges. Thus in analysis process very commonly we'll interested in graph topology features like the conductance and diameter (longest shortest path between nodes). Unfortunately data sets of whole network graphs can hardly be accessed for either commercial or ethic reasons like privacy and security. Unless the ON vendors release their statistics, there's actually no direct way researchers could observe the graph in a global scope. Thus sampling and estimation skills get extremely important to save the expensive query cost.

In my work the inspiration is that, for many existing sampling, searching algorithms being placed on online social networks, very few of them have taken a closer look to the topology features ONs uniquely present. For instance, sampling technique like simple random walk can walk on arbitrary graph, and breadth first search for finding path is also a very generally used approach. [37][42] But some types of special graph structures like trees or circles have topology based techniques, which have largely promoted the algorithm performance. The common features that has already been observed on ONs [37] could be taken advantage of,

and there's indeed urgent requirement to minimize query costs due to the growing magnitude of data.

### 4.1.1 Idea of hub first searching

Single pair path problem (SPP) is one of those very basic tasks needs for sampling the conductance, diameter and also for finding some interesting user relationship patterns (e.g. the very famous six person rule)[51]. This is the reason this chapter focus on the specific topic (Quite a lot of researches focus on the single source shortest path which will not be discussed in the paper[46]). Searching for a path between two nodes without any beforehand knowledge are often done by breadth first search, or a revised two-direction BFS.[4][42] But for both approach the cost could be extremely high as the path length grows. (Expecting a complexity of $O(\overline{\mathrm{d}}^{\overline{Depth}})$)

However, although we couldn't be sure with correct next steps, the scale-free feature of ONs might offer reasonable assumptions. It is widely observed that ON nodes degree distribution follows the power law, which means a very small portion of high degree nodes are well connected to rest of the graph. Hence if these 'hub nodes' could have priority in nodes queries, access to larger part of the graph could be done within minimal expectation of node expansion times, and thus give higher probability to discover the target nodes in early queries. [37][49]

The chapter introduces ON specific A* search [45] to solve SPP for optimal solutions (i.e. the shortest path), and also offers a greedy alternative aiming at find path as soon as possible. Detailed definition of the problem and a brief review of related knowledge and work will be addressed in the next section. Followed by the design and theoretical analysis of my

algorithms.

Set of comprehensive experiments based on above are performed over both generated and real world data sets, and the results are used to do evaluations. I also extends discussion on dealing with different network interfaces, analyze other leveraging information as well as possible extensions.

## 4.2 Problem Definition

This study will focus on the single pair path problem (SPP) and its optimal version the single pair shortest path problem (SPSP) over ONs. Graph models are as described in Chapter 2. All G discussed in the paper are treated as un-directed graphs. For directed ONs like Twitter (followee as in edges and follower as out edges), the transformation to un-directed version are done by taking unions of in/out neighborhood in order to achieve better conduction. Single pair path problem, in general, is to find a route between two different objects. On graph of ON, the problem is to find the path between two arbitrary chosen nodes u and v, that consists of edges in G with u,v as the two ends. If there's a path $P_{uv} = \mathbf{e}$ found, the two nodes are defined as connected and further used on estimation of graph conductance or diameter. Since all e in G carries identical cost, the length of $P_{uv}$ is then number of edges in the path: $l = |P_{uv}|$. When the requirements call for optimal solution, the problem turns into the SPSP as finding $P_{uv}$ that minimize l. [50]

### 4.2.1 Challenges

Most ONs only allow a query to v returning a local view of itself, namely the immediate neighborhood summary of v. In many cases only neighbor list and corresponding degrees

are gained. And meantime, query number allocated for each user or ip address is very limited to ensure the vendor's server performance, where approaches requires huge amount of requests (like Dijkstra needs to crawl all nodes) are infeasible.

As a virtual graph, the edges between all vertices in G don't have a physical distance as a meaningful cost and similarly no information like coordinates could be used to heuristically pre-estimate l from current position to the destination node. The chaos and dynamic online relationship in ONs also prevent researchers from finding theoretically constant methods to fit all kinds of graphs. For example, Twitter might have a much dense graph than Facebook since the previous network includes many loose relationship but Facebook friends often represents real world connections.

### 4.2.2 Existing Techniques

As mentioned in previous subsection, the features of ONs place obstacles on finding faster searching algorithms involves heuristics and cost issues like A* or uniform cost search for SPP.[19][45] So many applications use the very basic breadth first search.[4][42] To find $P_{uv}$, BFS start from u to check if v is in the neighborhood, and if v doesn't show up, it then query each neighbor of u and search deeper layer after the previous layer has all been expanded. Thus for length l path $P_{uv}$, the expected cost would be:

$$E(c) = \sum_{i=0}^{l-2}(\frac{|2E|}{|N|})^i) + 0.5 \times (\frac{|2E|}{|N|})^{l-1} + 1) \tag{4.1}$$

Although the diameter of ONs are not long (usually less than 20), the exponentially growing cost quickly becomes a large problem especially when average degree is high.

Average degree of tens or even a hundred are quite common in ONs, which makes BFS not affordable even when l reaches 5. Thus there's also a 2-dimensional BFS that start both from u and v. By searching both neighborhood by layer and keeping backtracking, $P_{uv}$ would be found when 2 searches get overlap on some nodes. Obviously the cut down the cost to around square route of that by the original BFS since only $\frac{l}{2}$ layers are required to be expanded. Considering the famous 6 person rule, the 2-direction BFS is currently the most feasbile solution for SPP over ONs.

One advantage of both approach above is that, with thoroughly searching each layer of u's neighborhood, once a solution is found it is complimentary an optimal one that also solve the SPSP.

## 4.3 Hub first searching

### 4.3.1 The A* search

As mentioned before, the lacking of physical path cost and distance has prevented heuristics or cost involved searching for ON SPP problems. For A* search that needs to assign a h(i) for each candidate node, the ONs only ensure there is at least one step to go as long as the candidate is not the destination itself. And to guarantee the optimality, A* also require h to be admissible that never over estimate the length, the minimal possible h(i) assigned to each node would be identical 1s if only distance are considered. This is also what enforce the search finally become a BFS. [21][63]

Hence I come up with the Hub first search (HFS) idea that goes from another dimension. Instead of estimating the path length which can hardly be done, thinking of the probability

of containing target node in next query might be more doable. Obviously the probability would be approximately proportional to the number of nodes captured, and further quite much ONs return the neighbor degrees (but no further details) together with the neighbor list within one query. This shows the potential of saving query times by maximizing nodes reached with minimal nodes expanded.

### 4.3.2 Baseline Design

HFS is the A* search that gives high degree nodes higher priority to be expanded, in order to raised the probability of hit the target in earlier steps. The current cost value f(i) is simply the depth $dp_i$ of the candidate node i, and the heuristics score h(i) is assigned according to $d_i$ with a negative correlation. It should be noticed that for guaranteed optimal, h(i) should be admissible which means $h(i) \leq 1$ must be true for any node i in G. Without any knowledge on hand, there's a probability for any nodes except the destination itself to be just 1 step away from the solution. Thus a choice of h(i) could be then given by:

$$h(i) = 1 - d_i/M \tag{4.2}$$

where M is an arbitrarily chosen constant larger than any current node degree. And the final goodness score of node i g(i) is then:

$$g(i) = dp_i + 1 - d_i/M \tag{4.3}$$

The heuristics will force the algorithm to search each layer in degree descending order until find target. Here I give a piece of my algorithm code:

```
function hubFirstSearch(G, origin, dest)
current=origin;

fr=[current,0,0];%set up the frontier
pointer=1;%point to current node
M=99999;%set constant

%loop until dest found
while ~ismember(dest, fr)
    neighborList=neighbors(G, current);%expand the next node

    %remove duplicates, assign scores, and append to frontier
    neighborList=neighborList(~ismember(neighborList, fr(:,1)));
    degreeList=degree(G, current);
    depth=fr(pointer,1)+1;
    n=numel(neighborList);
    score=depth+1-degreeList./M;

    fr=vertcat(fr,[neighborList, ones(n,1)*depth, score]);
```

```
        %find the node with min cost
        next=fr(fr(:,3)==min(fr(:,3)),:);
        next=next(1,:);

        %swap the min cost node with current next
        pointer=pointer+1;
        temp=fr(pointer,:);
        fr(fr(:,1)==next(1),:)=temp;
        fr(pointer,:)=next;

        current=fr(pointer,1);
    end
    end
```

### 4.3.3 Theoretical analysis

In BFS, when search reach the (l-1)th layer, there will be around $N_{l-1} = \frac{|2E|}{|N|}^{l-1}$ candidate nodes. Given the length l and the fact that each node will have the same chance to be expanded, the expected query times until the destination is found would be:

$$E(t) = \sum_{x=1}^{N_{l-1}} \frac{x}{N_{l-1}} = \frac{1+N_{l-1}}{2} \tag{4.4}$$

which means half of nodes at layer l-1 will be queried on average.

Then if switch to the HFS, since the nodes will be expanded in a fixed order according to their degrees, the probability to have the target in $i_{th}$ node is actually about $\frac{d_i}{N_l}$. Then similarly the E'(t) would be given by:

$$E'(t) = \sum_{x=1}^{N_{l-1}} \frac{xd_x}{N_l} = \sum_{x=1}^{N_{l-1}} \frac{xd_x}{N_{l-1} \times N_1} \tag{4.5}$$

Obviously the higher kurtosis (i.e. heavier right tail) of the degree distribution, the more weight first several queries will have and leads to a smaller expectation value.

For further analysis the algorithms' theoretical performance on ONs, I generate from the BA approach simulated scale-free graphs with different starting parameter (See Table 4.1), and numerically calculate the expected queries needed in the last step to compare accordingly. It should be noticed that, if the candidate nodes are seen as a representation of the whole graph (reasonable especially when l grows), the E'(t) is actually asymptotically to:

$$\frac{1}{N_1} \sum_{i=1}^{|N|} \left\lfloor \frac{iN_{l-1}}{|N|} \right\rfloor \frac{d_i}{|N|} \tag{4.6}$$

And given the full knowledge of generated graph the calculation is then feasible.

| Graph Identifier | F | G | H |
|---|---|---|---|
| $M_0$ | 8 | 20 | 2 |
| k | 6 | 10 | 2 |
| $\|N\|$ | 1000 | 3000 | 5000 |
| $\|E\|$ | 5848 | 29620 | 9974 |
| mean(d) | 11.7 | 19.75 | 3.99 |
| kurtosis | 39 | 63 | 536 |

Table 4.1: Generated BA graphs

The results are showed in the Figure 4.1. Above all three graphs the HFS shows domination in expected queries needed theoretically. Obviously the query number depends largely on the average degree of the graph, and not surprisingly, graph with a higher kurtosis receives better promotion in percentage. Further, the generated graphs are actually more balanced in degree distribution, which means we may assume more significant outperforming by HFS in more extreme real world graphs (See experiments in section 4). The calculation is valid for both one-direction or two-direction implementation, by changing the l to $\frac{l}{2}$ for two-direction search.

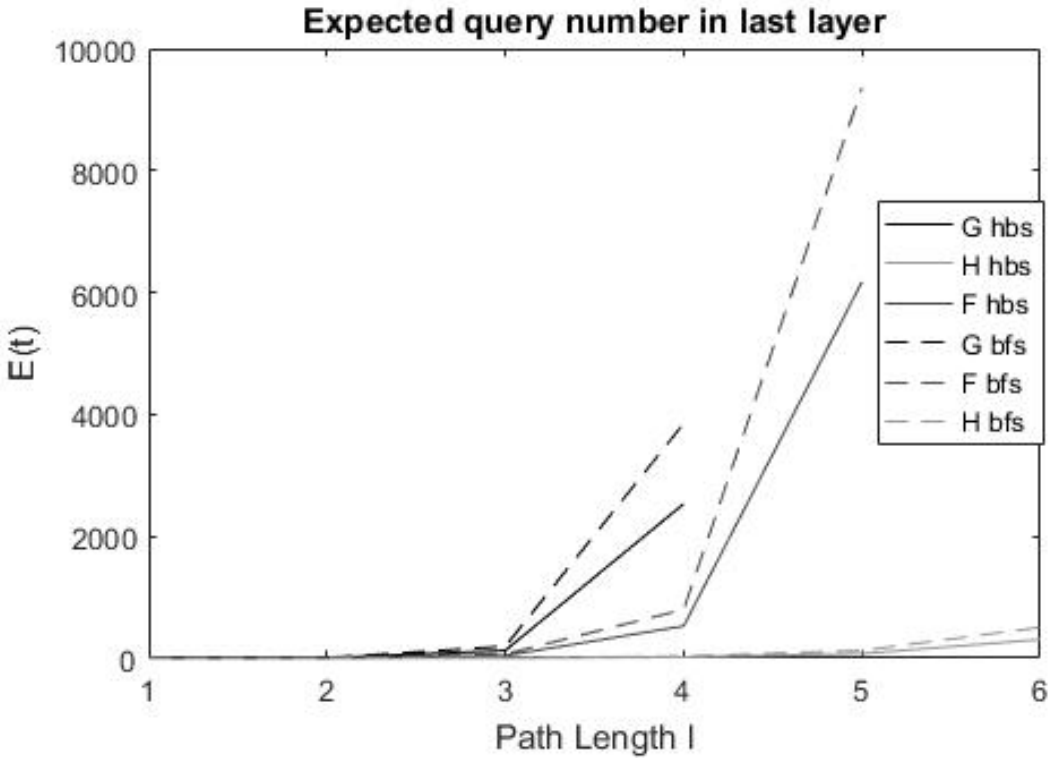



Figure 4.1: Theoretical estimation of query number

## 4.4 The greedy alternative

### 4.4.1 Limitation of A*

Due to the complimentary optimal solution by BFS when solving SPP, the paper first introduce A* HFS keeping the optimality for a fair comparison. However, since each node are in a certain depth as its cost f(i), and the h(i) value has an upper limit of 1, nodes in the $l_{th}$ layer will never have a higher cost than any of those in the $(l+1)_{th}$ layer. This means A* is still enforced to search by layers and thus the performance is bounded by node count in previous layers.

In real cases, the requirement for shortest path is not always true especially on ONs where distance doesn't carry a physical meaning. The critical characteristic that separate the ON based SPP from other path searching problem is that the knowledge fetching on ONs is rather expensive but the walk itself actually cost nothing (and often not a big concern as well). Arbitrarily path between u and v is enough to define connectivity, and there's not an explicit preference among relationships with different l values.

Instead of focusing on minimizing the path length due to a real cost [63][19] but with much more knowledge can be arbitrarily used like railroad maps, main concentration of ON based path finding is to minimize the knowledge needed for a solution (i.e. number of queries). This makes a greedy alternative that doesn't necessarily guarantee optimality, but aims at finding a path quicker plausible.

### 4.4.2 Greedy design

Naturally one way to choose the greedy candidate would be deciding merely by degrees of nodes in current queue, and algorithm always expands the one with the highest degree in the next step. This helps to reach as many nodes as possible in each query, hence promote significantly the coverage of the graph in few steps.

Considering that v might not have hubs as immediate neighbors, it's more reasonable to apply two directional search expecting two walks from both nodes hit at some time. Since the current depth is omitted, greedy skips nodes carrying fewer neighbors, and push the walks into the hubs' network where there is usually a supposed tighter inner-connection. Thus the algorithm not only captures more nodes in few queries, but also brings the nodes to where it is more likely to get overlap.

As only degrees are concerned, it also gives the upside of easier implementation and calculation. The variation of greedy HFS is given by:

```
function greedy(G, origin, dest)
%search from 2 direction
fr1=[origin, degree(G, origin), 0];
fr2=[dest, degree(G, dest), 0];
pointer=0;

%loop for no overlap
while(sum(ismember(fr1(:,1), fr2(:,1)))==0)
    pointer=pointer+1;
```

```
    %find node in frontier with highest degree
    maxd1=max(fr1(pointer:,2));
    cur1=fr1((fr1(pointer:,2)==maxd1),:);
    cur1=cur1(1,:);
    %swap to current index
    temp=fr1(pointer,:);
    fr1(fr1(:,1)==cur1(1),:)=temp;
    fr1(pointer,:)=cur1;

    %expand node, remove duplicates,append to frontier
    nbl1=neighbors(G,cur1(1));
    nbl1=nbl1(~ismember(nbl1,fr1(:,1)));

    dgl1=degree(G,nbl1);
    parent1=ones(numel(nbl1),1).*pointer;
    fr1=vertcat(fr1,[nbl1,dgl1,parent1]);

    %repeat for frontier 2
    ...
    ...
end
```

**end**

### 4.4.3 Theoretical analysis

In greedy, the length l of $P_{uv}$ is not fixed, so the expectation can't be achieved using same approach as A* and BFS. However, since greedy doesn't expand nodes in a layer manner, it actually reduce the correlation among nodes being reached by skipping most nodes that locate near u and v. Then, the two set of nodes reached by greedy search can be asymptotically regard as independent random subsets of graph G.

By the mark and recapture idea in estimating population size, for graph G with $|N|$ vertices the expected sample size (suppose 2 sets sizes are similar) to get a first recapture is $\sqrt{|N|}$ (bias introduced). [36] Thus the total queries needed for capture an overlap node could be given by:

$$E^g(q) = \frac{2\sqrt{|N|}}{d^*} \tag{4.7}$$

Where $d^*$ is the average degree of expanded nodes.

Obviously, the greedy that choose the largest degree node each time gives the highest $d^*$, which also minimize the $E^g(q)$. The exact values can hardly be numerically calculated with different degree distributions and starting nodes, and experiments will be casted in the next section.

## 4.5 Evaluation

Based on the previous contents, the section will show a set of experiments based on both simulated graphs and real world data sets to verify the theoretical discussions.

### 4.5.1 Experiment setup

For convenient comparison, all three algorithms are implemented as two-direction search (i.e. start from both origin and destination). The implementation is accomplished by Matlab. Involving data sets includes the 3 generated graph discussed above, as well as different real ON data sets from Standford Network Analysis Project. [31]

Consider the large quantity, among these data sets I chose Facebook and Twitter as a representation to be showed in results as they might be the most concerned data sets because of popularity, while similar results are also observed over other real world ON data sets. One of the generated graph G is also included to see the difference. The basic statistics of graphs will be used in result analysis are showed in Table 4.2:

| Data set | G | Twitter | Facebook |
|---|---|---|---|
| $\|N\|$ | 3000 | 81306 | 4039 |
| $\|E\|$ | 29620 | 1342310 | 88234 |
| mean(d) | 19.75 | 33.02 | 43.69 |
| kurtosis | 63 | 390 | 57 |

Table 4.2: Tested data sets information

In each single trial of the experiments, the origin and the destination are randomly picked by uniform and the three algorithms search with the same pair. Costs are recorded as the times of query to the original graph, and the found paths are returned as well.

Each data set has been tested for 1000 replications through the above process, so that relative data for different path length could be sufficiently observed and collected. The 1000 data points are finally grouped by the shortest path length, and the mean cost are thus plotted for

comparison.

### 4.5.2 Experiment result

Plotted result of the experiments are showed in Figures 4.2,4.3 and 4.4. In general the A* HFS save about half of the query cost when the path length l doesn't go too far (specifically, an l<=5 keeps the dominance significant). When l grows however, the outperforming becomes weaker which violates a bit to the theoretical analysis. The reason is also not tough to figure out since in theoretical analysis no boundaries are placed upon the queries and the if so the query number could increase infinitely. It is natural that when l grows larger, the nodes left without being queried will decrease to a very small amount, leading to indistinguishable cost for A* search and BFS.

On the other hand, there is a very promising result showing Greedy will find a path significantly quicker than any of the two algorithms above. Besides, the number of query is not increasing exponentially. Instead the plotted lines present to be linearly correlated with l with very small slope, or even stay around some constant in some of the data sets. And this result also shows consistency with different type of ON graphs, which could conclude that the Greedy approach is an ideal alternative for solving SPP that doesn't call for optimality.

Some assumptions on graph topologies' effects on algorithm performance are also checked by the experiments. The final query cost of BFS and A* depends mostly by the path length. While With fixed l it can be observed an positive correlation between degree and cost.

Kurtosis of a graph also show influence in performance of both two HFS method. On Facebook where the value is far lower than that of Twitter, A* sometimes fails to beat BFS and the cost of greedy also increases much quicker. It is also interesting to observe that, on

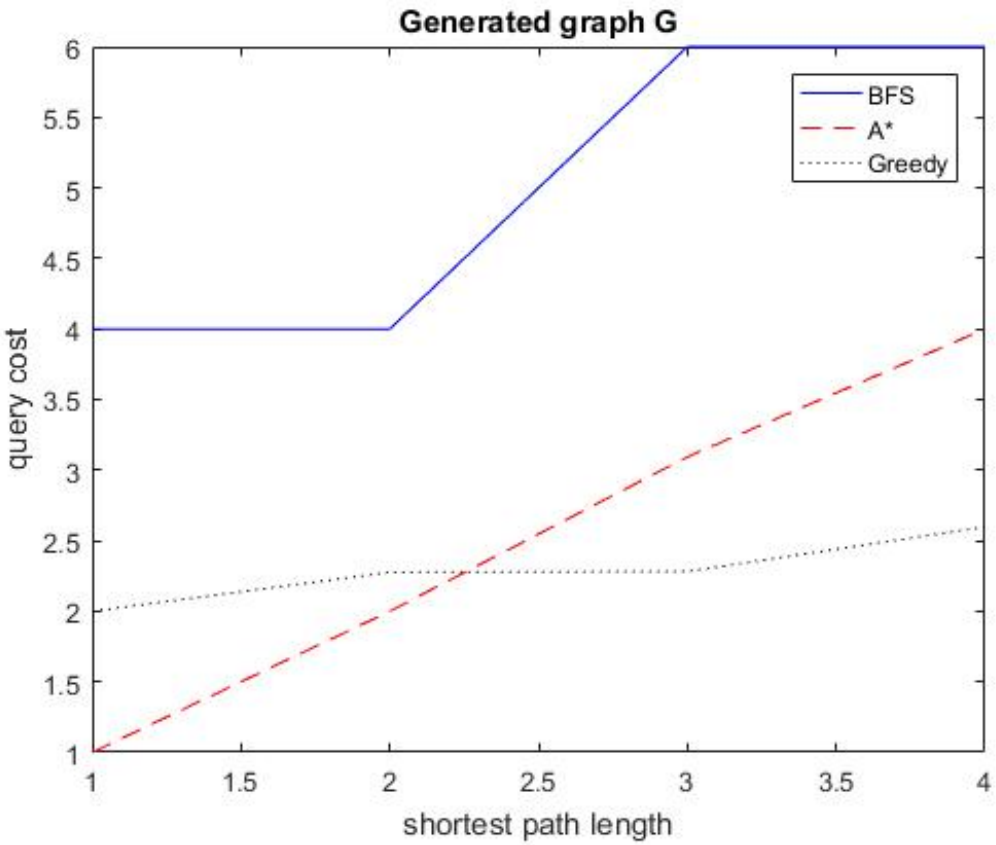



Figure 4.2: Graph G result

the generated graph with $|N|$ of several thousand the diameter(longest shortest path) is only about 4 and grows when the generated graph is larger. However, on real world ONs, the diameter seems to be around 7 regardless of the data set size which might be an evidence for the 6 person rule (The Facebook data set is much smaller than Twitter, but shows similar diameter and average path length).

For Greedy which is promising in cost, I also plot out the relative error of path length in Figure 4.5, namely the extra path length bring by the non-optimality, with regard to the true shortest path length l. The error does increase with the path length, but is not exceeding 1 even l is as long as 8. This indicates an at most 1 or 2 extra steps will be in the path Greedy found compare to the shortest, which in my point of view acceptable. To conclude, the

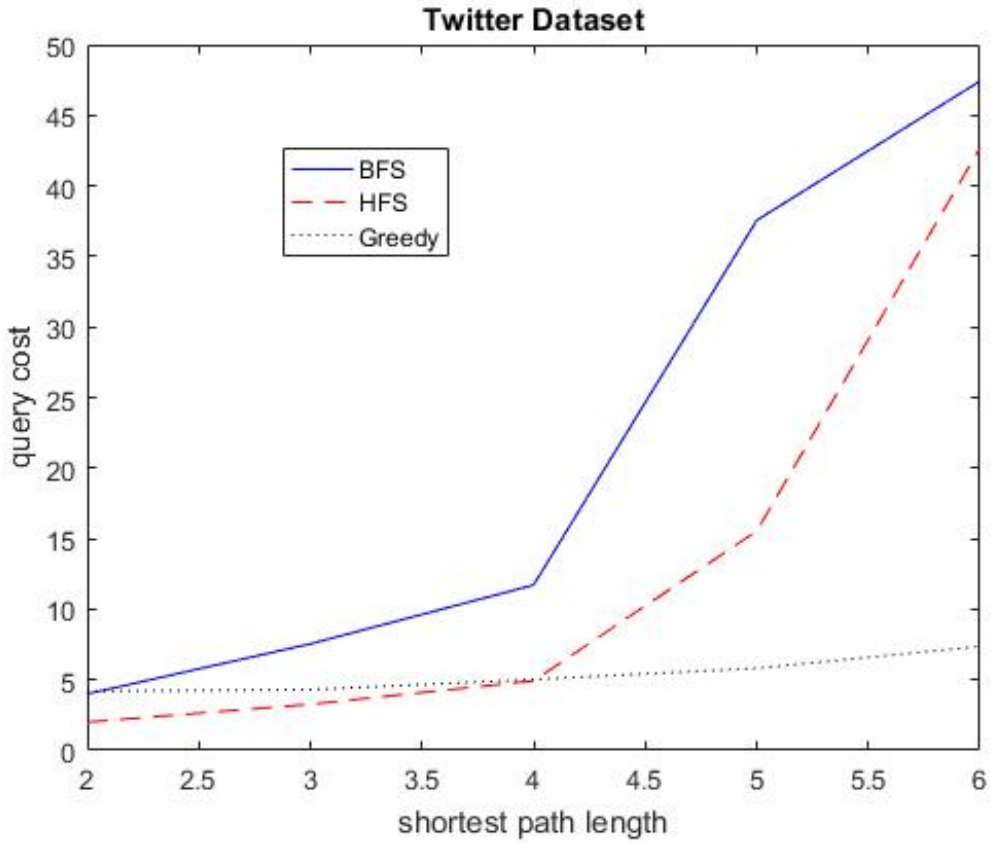



Figure 4.3: Twitter result

HFS algorithm outperform basic BFS class method over graphs with ON like topology. A* search will ideal when path length does not exceed 5, with significant faster search and not loosing optimal. Greedy is extremely fast when no optimal requirements exist. Choosing the appropriate approach should be based on specific problem needs as well as a rough look at the graph topology. (e.g. a short pilot simple random walk to see rough degree distribution, density,etc.).

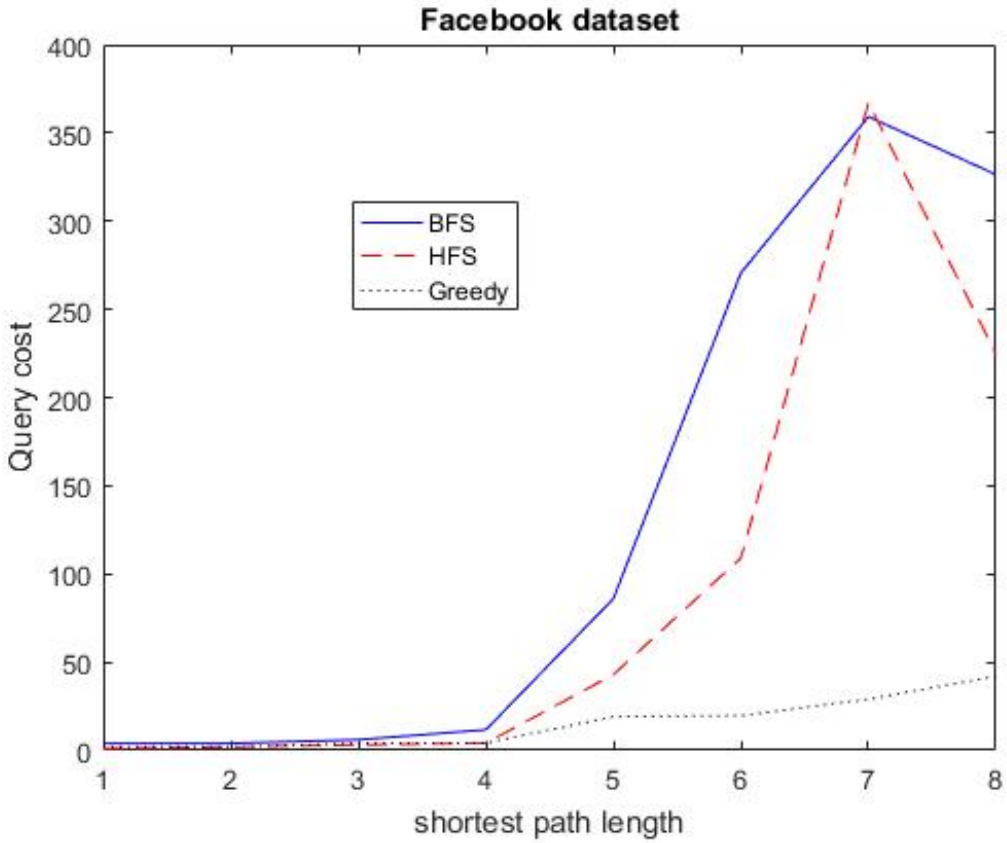



Figure 4.4: Facebook result

## 4.6 Extended discussions

### 4.6.1 Different network interfaces

In previous sections of this chapter, a neighbor-degree interface is supposed, namely query to one node returns all its neighbors and the corresponding degrees. The assumption holds for many of famous ONs like Twitter, but the interface could vary site by site.

Some ONs returns the neighbors in paginated format with k tuples at a time. In the case nodes are returned independently of degree (e.g. in time order), we would expect $\frac{2|E|}{|N|k}$ times more queries for each node in BFS, while query hubs no longer gives more information than non-hubs and the ultimate performance would be similar. However, if the nodes are

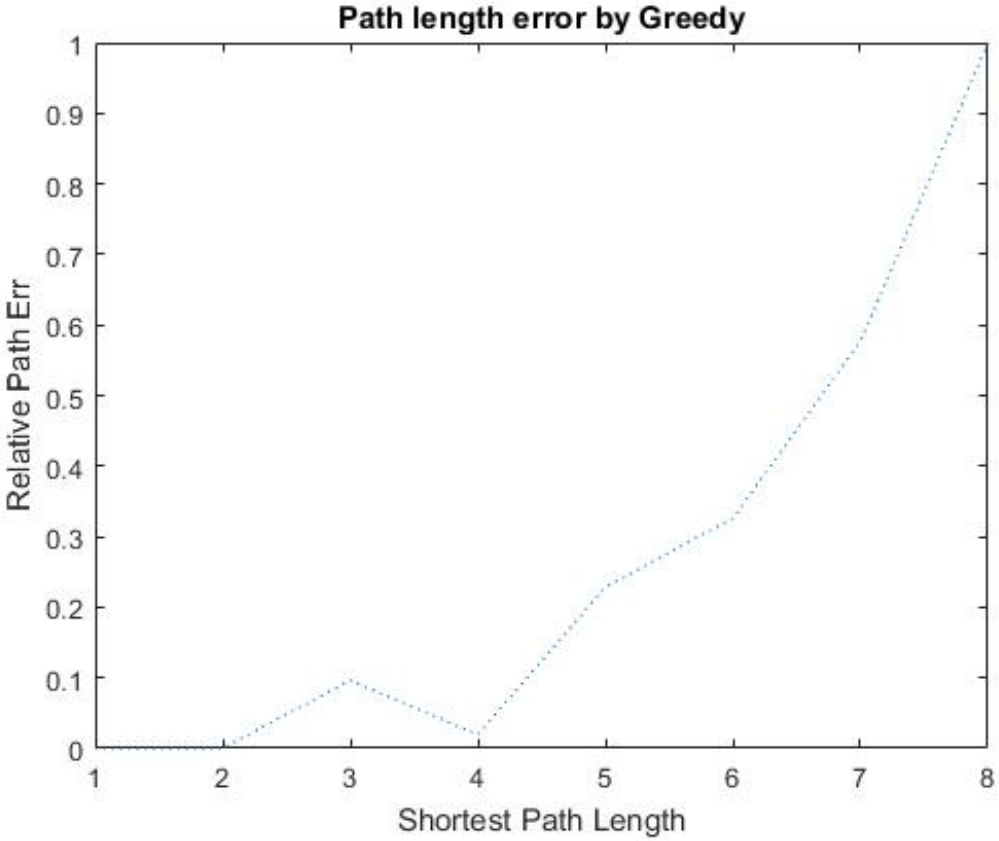



Figure 4.5: Extra path length bring by Greedy

returned in degree descending order (some ONs provide such option when query), the greedy still forces the search going to hubs' network and keep getting knowledge of nodes with relatively highest degree. This will, naturally, show dominance to BFS and A*.

Another frequently seen interface is the neighbor only ones, that query returns only the neighbor list without any extra information. This results in in-feasibility of A* and Greedy based on degrees, which means a substitute should be found in similar manner. Since random walk on network graphs usually gives higher sampling probability to higher degree nodes, the assumption that hubs will appear more frequently either in the walk or in neighborhood along the random walk is plausible. Thus, the degree based heuristics can then be similarly substituted by the frequency of each captured node f. $h^f(i)$ can be assigned accordingly, and

for greedy simply expand the node shows up most frequently.

### 4.6.2 Other leveraging information

The above discussion are in worse cases than the assumption in the paper, while sometimes ONs provide extra labels for each node. Some labels can be used for community detection, thus help to adjust the heuristics for better fit. I shall discuss an application for this in the next study in Chapter 5. For instance, some ONs gives tags users add to state their interests together with a query, then heuristics might leverage the overlap in tags of users as a parameter and assign the h(i) as a combination of degree and similarity in tags. Intuitively, the h(i) can be given:

$$h(i) = \pi(1 - d_i/M) + (1 - \pi)[2o/(tg_u + tg_i)] \tag{4.8}$$

where o is the number of overlap tags of i with regard to u, $tg_i$ is the number of tags user i has, and $\pi$ is chosen between [0,1] to state the importance of degree in determination of h(i). In more general form, if I consider more than 2 attributes (e.g. location might be available, thus the distance is also important information [60]), the actual h(i) should be in the form of:

$$h(i) = \sum_{a=1}^{k} w_a v_a \qquad , \sum_{a=1}^{k} w_a = 1 \tag{4.9}$$

### 4.6.3 Dynamically adjusted heuristics

As BFS makes no evaluations on candidate nodes, it takes no use of information already known which would be quite a waste. Even when encountering with nodes already queried,

the algorithm still keeps a uniform preference to each candidate, leaving the queried knowledge base unused.

In the case of A*, the h function can be updated according to previous searches. Nodes that are already detected connecting to the destination should be assigned a h value 0 telling algorithm to expand exactly that way. And similarly, heuristics value of nodes don't contain destination in neighborhood can now be over 1.

With the establishment of the knowledge base and growing times of search, there will be a better view of the graph. The overall proportion might still be low, but plenty knowledge of hubs is available which would speed up the correct route choosing especially on hubs. This is also the key in HFS performance.

### 4.6.4 Definition of hubs

Idea of HFS is to get large coverage over the graph within small amount of queries via querying nodes with good connection to the rest of the graph. In this dissertation, my definition of 'hubs' merely depends on the degree values, while some nodes with relatively lower degrees and meantime containing many hubs in their neighborhood can also be regarded as 'well-connected' in another manner. (Imaging an extreme case that a graph center connects only to community centers without connection to other nodes, this center may not present very high degree but can be regarded as the most important node to connect all nodes.)

Hence in future experiments I may also give priority to nodes that high hub degree $d_h$ observed. Of course this requires the knowledge base mentioned above being developed to some extent. [45] [36]

## Chapter 5: Community affiliation incorporated random walks

[62]

With online networks being subtle data sources these days, growing demands of analyses and restrictions on web interfaces resort to techniques like random walk approaches that can run under local neighborhood access. Meanwhile, the presence of highly clustered community like structure in large networks leads to random walk's poor conductance, causing intolerable and hard-to-foresee long mixing time before useful samples can be collected.

With lack of techniques incorporate online network topology features being the context, in this chapter we focus on taking use of community affiliation information that possibly comes with metadata when querying objects in online networks, and proposed a speeded version of random walk by raising the probability of inter-community edges being selected. Assuming the community structure is well established as promised, the community speeded random walk expects better conductance and faster convergence. Our method forces the sampler to travel rapidly among different communities that conquers the bottlenecks and thus the samples being collected are of higher quality. We also consider the scenario when community affiliation is not directly available, where we apply feature selection algorithms to select features as community.

## 5.1 Motivation

Online networks (ON) have long been serving a vital role in data tasks over a broad range of topics either for research or commercial purposes for the considerably tremendous amount of information being created, exchanged and archived every second. Due the challenges stated in previous chapters, ON involved studies often use sampling methods that collect samples from networks following a pre-determined distribution in place of retrieving the whole networks. Then estimations of desired data aggregation can then be calculated via sampled data. Among all these approaches random walks , [20] being capable of sampling under designed stationary distribution after a period of "burn-in" and running locally without knowing global topology, are popular and powerful tools which can fulfill the requirements and fit into practical interface limitations. More specifically, simple random walk (SRW), with its adaptability to different scenarios as well as ease of implementation is considered a golden baseline of the sampling techniques.

However, these techniques used by majority of current works [18][27][43] involving ON sampling are mostly topology oblivious work well only when the underlying graph features a large conductance. [1]. Unfortunately, the conductance of real world social networks is often substantially lower than expected. [32] This creates the dilemma for all random walk techniques of possibly intolerable long waiting time known as the “burn-in” periods (or mixing time), when large numbers of queries or transactions are required before the random walks’ sampling distributions can converge to desired stationaries and start to draw samples. The prolonged burn-ins are a result of real-world networks’ tendency to exhibit highly clustered topology [48] which generates bottle-necks hard for random walks to cross

through. [43][44] Without sufficient knowledge of entire networks, it's also difficult to foresee the mixing costs or monitor actual convergence.

### 5.1.1 Speed up random walk by leveraging community affiliation

Assuming the existing random walk approaches' getting trapped in one or several communities is due to the low inter-community travel probabilities, we then consider a novel problem of how to significantly increase the conductance of large ONs with highly clustered topology by leveraging community information. [40]

Here first we shall state the concept of "community" in our paper and its relation to graph conductance before propose our idea. Though universal definition of community is pretty ambiguous, from intuition (see example in Figure 5.1) communities are well knitted clusters of node collections in a graph with a few ties to the rest of the system, inside which more edges are often observed than crossing among them. Social communities in the context of social networks can be any subgroups of people are all friends to each other or have very few degree of separation, [35] and online communities can then be extended to any meaningful collection of objects. According to the definition, graph conductance of a certain set of nodes S is given by $\frac{P(S,\overline{S})}{\min(\pi(S),\pi(\overline{S}))}$ [6] with $P$ being the probability to leave S and $\pi$ being the sampling probability. This could be relatively low for the nature of community to be less likely connected to other part of graph.

Intuitively, increasing the weight of crossing community edges will significantly increase the conductance and conquer these bottlenecks. Our proposed Community affiliation incorporated random walk (CRW) (CRW) shows a way to address the task by choosing next

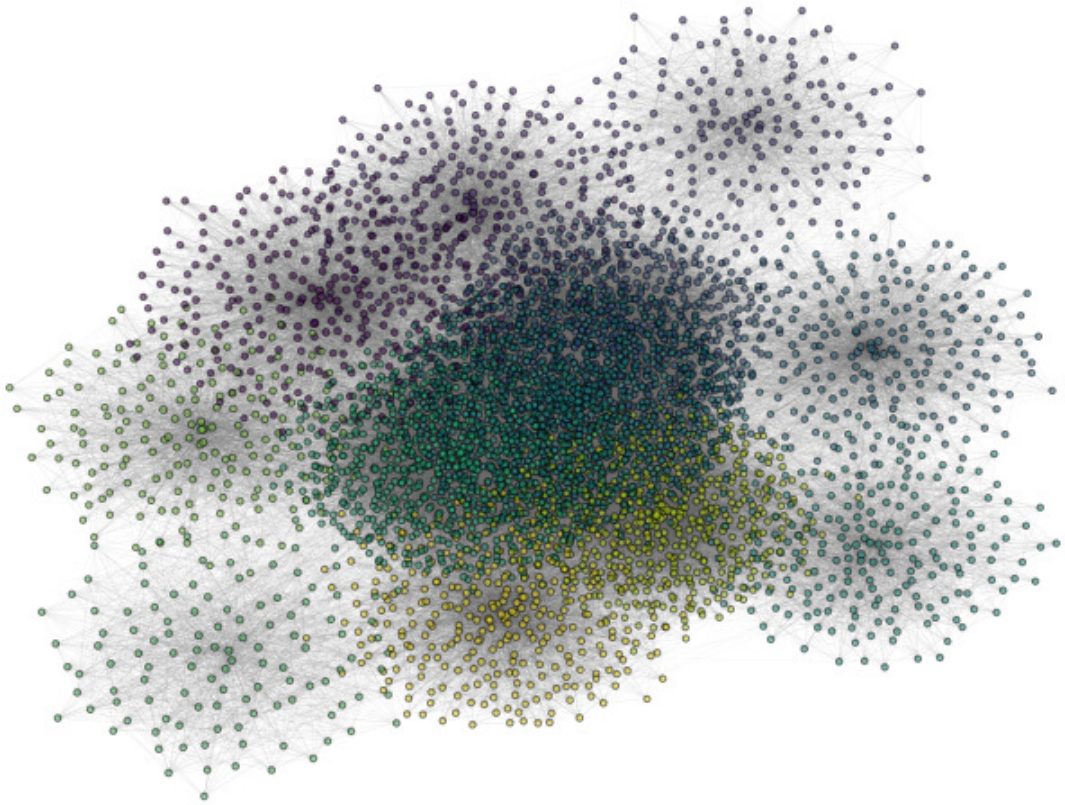


Figure 5.1: A highly clustered large network

step uniformly among communities instead of neighbors. CRW forces the sampler to jump through communities and leads to rapid exploration of the entire graph. On the other hand, we are still able to re-balance the random walk to a reversible Markov Chain by introducing Metroplis Hastings algorithm [57] so that an identical stationary distribution as current random walk approaches (e.g. SRW) can be guaranteed.

In the process we take use of the community affiliation information (i.e. the community value a node has a membership with: $C(x)$), that is any meaningful community definition per information available on hand (e.g. organizations could be a reasonable choice). And we could encounter different interfaces with ideal ones sometimes where a "perfect community affiliation information" is given based on knowledge (explicit and well-defined $C(x)$).

Then a direct application of CRW will be applied. While in cases $C(x)$ is not immediately available, workarounds are required to give inference on these values. In this paper we focus on selecting candidate metadata attributes to make affiliation inferences. Though there have been a wide selection of more precise community detection algorithms adopted in current literature, they either conflict with our lacking of topology knowledge or come with unacceptable query costs ($O(n^2)$ or so), [5] which have been dumped here.

In this chapter, we show our design of a topology oriented, or more specifically, community affiliation based approach CRW to speed up random walk based sampling techniques used over ONs, and discuss the application of them being used facing different data interface.

We perform experiments over both synthetic data and real-world online data sets[31], using the original SRW as a baseline comparison, to verify the correctness, effectiveness and efficiency of CRW.

We'll clarify some extra terminologies that we will be using as well as performance measures in the next section, and then detailedly expand original CRW design in Section 5.3 with assumption of perfect community affiliation. Theoretical analysis is given to prove the mathematical fundamental behind CRW, attached with the experiment results. In Section 5.4 we discuss the scenario of implicit community information when inferences are required. Among several solutions we choose to adopt feature selection algorithm to locate attributes as ground truth communities. We also place experiments of real-world data sets for this workaround to evaluate. Then we summarize our outcomes and think about possible improvements at last.

## 5.2 Terminologies

### 5.2.1 "Community-Structured" graph and community affiliation information

According to current literature, [32][48][17] large real-world networks tend to exhibit a highly-clustered topology, or so called "community-structured" topology. While in graph theory the definition of "community" is ambiguous, there exists greatly many popular definitions and a very rich context of techniques have been proposed to detect, measure and define the term. Intuitively, communities are groups of well knitted nodes in the graph, whose real-world reflections could probably involve interactive relationships (e.g. friends, family), shared properties (e.g. interests) or similar roles (e.g. key opinion leaders) within the graph. [25][24] For instance, if we consider students from same institution on Facebook, the possibility of connections within would be much higher than those to outside of the institution, which makes institutions as naturally ground truth communities. On the other hand, because of the absence of universal adopted definition, different divisions of a same graph can all be appropriate community selections and there might be overlapping in the communities.[61][34]

In this chapter, we first combine the definition of conductance and give our definition for *"community-structured" graph*:

**Definition 5.2.1.** We define a graph $G = (N, E)$ as a community-structured graph if the conductance of SRW on this graph satisfies Equation (5.1):

$$\phi_{\mathrm{SRW}}(G) = \min_{\mathscr{C}' \in \mathscr{C}, \pi(\mathscr{C}') \leq \frac{1}{2}} \left(\frac{Q(\mathscr{C}', \overline{\mathscr{C}'})}{\pi(\mathscr{C}')}\right) = \varphi_{\mathrm{SRW}}(\mathscr{C}') \tag{5.1}$$

Where $\mathscr{C}$ is a partition of graph G $\mathscr{C} = \{C_1, C_2, C_3..., C_n\}$, with its union $\cup_{C \in \mathscr{C}} C = N$. Also, we define $\mathscr{C}' \subset \mathscr{C}$ where $\mathscr{C}' \cup \overline{\mathscr{C}'} = \mathscr{C}$. $\mathscr{C}'$ might consists of one or more communities in set $\mathscr{C}$, i.e., $\mathscr{C}' = \{C_a, a \subseteq \{1, 2, ...n\}\}$.

We assume a community-structured graph's conductance at $\mathscr{C}'$ takes the minimal value over the whole graph, i.e., $\varphi_{\text{SRW}}(\mathscr{C}') = \phi_{\text{SRW}}(G)$. Thus, the reason random walk get stuck in place and encounter slow convergence is due to low conductance inter-communities. On the other hand SRW, as most popular random walk algorithm, is an ideal choice to place our comparison over. We further define *"Community affiliation information"* as node $x \in N$'s membership of communities:

$$C(x) = C_i, \iff x \in C_i, i > 0 \tag{5.2}$$

### 5.2.2 Measurements of performance

For sampling algorithms, it's always the case we compare sample quality versus the cost. Costs could be in a manner of the most precious resource involved in sampling, and with regard to ONs will be the query times. Sample qualities are ideally compared by the distance of sampled distribution to ground truth, while in real-world works it's usually intractable as unacceptably large number of samples are required especially when sampling space is massive. Considering the interface restriction before, we'll have to choose alternative measures.

*Effective sample size* as an measure of information quantity a sampler carries, can be considered as the sample's i.i.d. sample equivalence. It tells the auto-correlation of sample

and thus is a good indicator of sample quality. We use Stan's method [8] to estimate Effective sample size (ESS) in experiments. The next measure involved is the empirical *total variance* or alternatively the *L1 norm distance* [58] of samples and target distribution. Although the analytical sampling distribution of random walks can hardly be calculated, we can still use frequencies of data points to approximately estimate the values. And We also compare our CRW's community coverage speed over SRW by defining a measurement, *hitting time* (query cost versus number of communities reached), to verify our claim that CRW travels much faster inter communities. Random walks are know to have "fake convergence" on highly clustered graphs when the walks are trapped in several communities and the diagnostics are stable. [7] In such cases the samples collected might not be a good representation of entire graph. Hitting time could reveal the proportion of graph being explored.

### 5.2.3 Graph generative model

For latter use of theoretical analysis and easier experiments, We reviewed popular graph generative models and choose to use the Lancichinetti–Fortunato–Radicchi benchmark graph (LFR). [28]

LFR model captures both the power law degree distribution of large networks and the community structure. It also come with complimentary community affiliation information, which we consider it an ideal fit for our experiment.

## 5.3 Community affiliation based random walk

In this section, we proposed a twisted random walk (CRW) giving priority to edges crossing communities to be selected. We firstly discuss the main design of the algorithm, followed

by a theoretical analysis showing a conductance surpassing using CRW over SRW on same community-structured graph. We assume a perfect knowledge on community information in this section and extend our discussion to tackle less informative scenarios later in Section 5.4.

### 5.3.1 Basic design

At the ease of implementation and flexibility of being adapted to arbitrary interface, SRW is considered as a "golden standard" for random walk like sampling techniques, and is adopted by very majority data tasks involving ON sampling. [18][27] However, SRW's performance on community-structured graphs could be significantly slowed due to the low probability of exiting communities. The transition matrix of SRW that uniformly picks a neighbor gives all the edges identical margin probability of being chosen. While community-structured graphs are much densely connected intra-communities than inter-communities, preventing SRW from travelling smoothly through the entire graph. For instance, if we consider a barbell graph with corresponding two communities, the only one edge connecting two different communities in the graph carries an incredibly low probability of being selected (i.e. $\frac{1}{2|E|}$). Even if SRW is able to reach one of the two connected nodes, the chance of pick the cross-community edge is still relatively low as the nodes have much stronger connectivity to their own communities.

Intuitively one will want to raise the weight of crossing community edge to encourage random walk sampler to travel through it. [64] We came up with a rough idea of CRW that take use of community affiliation information of nodes (see Figure 5.2). After CRW reaches at node $x$ that is connected to at least one other community, it step into the decision of which neighbor to be the next hop. Instead of picking uniformly at random among $\mathrm{n}(x)$ as SRW,

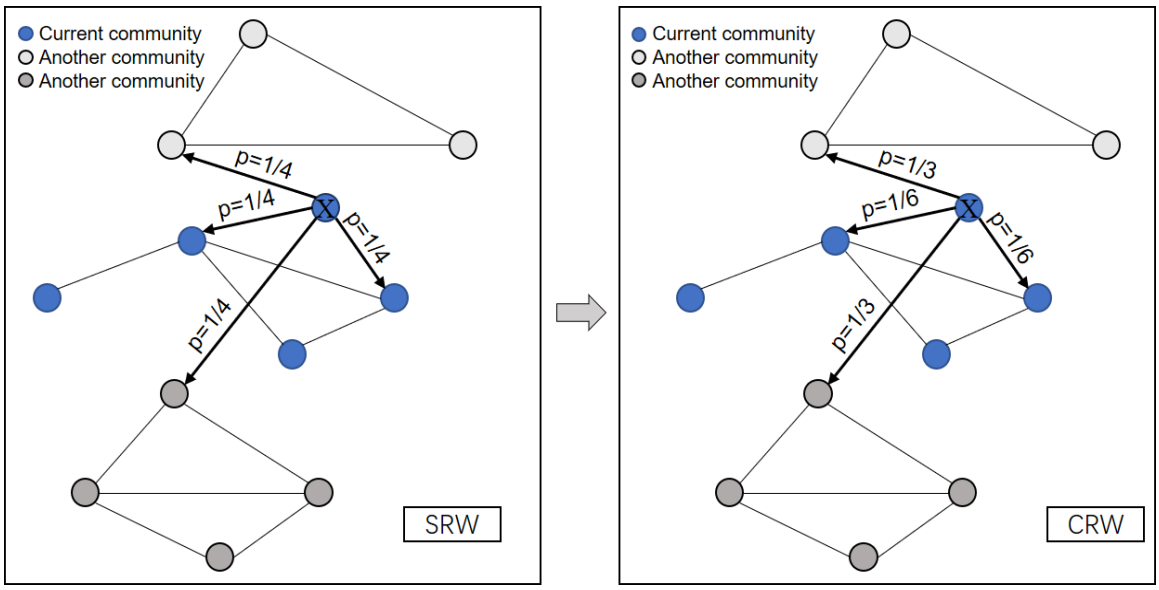



Figure 5.2: CRW's choice of next hop

When random walk reaches node $x$, CRW choose a community first, thus probability of exit current community at the node is raised to $\frac{2}{3}$ while SRW will give $\frac{1}{2}$ with treating edges identically.

CRW picks a community $c$ uniformly at random firstly from all different communities in $x$'s neighborhood, then pick one among all x's neighbors that is a member of $c$ at random to be the candidate. And the sampling probability, although intractable at this point, can be later curved by MH algorithm to desired.

### 5.3.2 Theoretical analysis

In this section, we're going to discuss the theoretical fundamental behind CRW.

#### 5.3.2.1 Transition matrix and stationary distribution

By definition of conductance $\varphi(C)$ at cut $(C,\overline{C})$ in Equation (2.7), the weight of cross-community edges can be assigned arbitrarily large to achieve better value. CRW choose communities uniformly at random each step in order to place identical transition probability to each of them. Hence a naive transition function of any node pair x and y would be given by:

$$g(x,y) = \frac{1}{|I(x)|}\frac{1}{|O(x,y)|} \tag{5.3}$$

In Equation (5.3), $I(x)$ denotes the set of different communities in neighborhood of $x$, i.e. $I(x) = \{C(y)|e(x,y) \in E\}$. We also denote $O(x,y)$ as set of the neighbors of $x$ that belongs to same community as node $y$, i.e., $O(x,y) = \{z|e(x,z) \in E, z \in C(y)\}$.

Obviously the transition function result in CRW's different stationary distribution from SRW. For the purpose of fair and strict comparison, we use Metroplis Hastings algorithm described in Chapter 2 to curve the stationary distribution to the same as that of SRW, namely our desired $\pi(x) = \frac{\mathrm{d}(x)}{2|E|}$. With $g(x,y)$ as the proposal and acceptance matrix $\alpha(x,y)$ calculated using Equation (2.4), the transition matrix of CRW is given by:

$$P_{\mathrm{CRW}}(x,y) = \frac{1}{|I(x)|}\frac{1}{|O(x,y)|} \cdot \alpha(x,y), \tag{5.4}$$

which ensures CRW keeps a same stationary distribution as SRW.

#### 5.3.2.2 Conductance analysis

Next we prove the conductance on "community-structured" graph using CRW is greater than or equal to the conductance of same graph using SRW, and so CRW will expect a quicker convergence according to Equation (2.9).

As defined in Section 5.2, node set $N$ of graph $G$ is partitioned into disjoint subset of nodes, namely communities, $\mathscr{C} = \{C_1, C_2, C_3 ..., C_n\}$. And we define $\mathscr{C}' \subset \mathscr{C}$ as a set consisting of one or more communities in $\mathscr{C}$: $\mathscr{C}' = \{C_a, a \subseteq \{1,2,...n\}\}$. And we also claim that community-structured graph satisfy: $\phi_{\text{SRW}}(G) = \varphi_{\text{SRW}}(\mathscr{C}')$. Here introduced Theorem 1:

**Theorem 1.** *For a given graph $G = (N, E)$, if it satisfies the community-structured given in Definition 5.2.1, the conductance of CRW $\phi_{\text{CRW}}(G)$ over G is greater than or equal to the conductance of SRW over G $\phi_{\text{SRW}}(G)$.*

Proof:

As stated earlier in this section, we set CRW's stationary same as SRW for a straightforward comparison versus baseline, hence we only need to compare $Q(\mathscr{C}', \overline{\mathscr{C}'})$ using different approaches. (Recap that $Q(x,y)$ is the probability of moving from $x$ to $y$ at stationary distribution, and $Q(\mathscr{C}', \overline{\mathscr{C}'}) = \sum_{x \in \mathscr{C}', y \in \overline{\mathscr{C}'}} Q(x,y)$, see Section 5.2). Given SRW's transition matrix $P_{\text{SRW}}(x,y) = \frac{1}{d(x)}$ and that of CRW derived in Equation (5.4), we'll have:

$$Q_{\text{SRW}}(\mathscr{C}', \overline{\mathscr{C}'}) = \sum_{x \in \mathscr{C}', y \in \overline{\mathscr{C}'}} \frac{\mathrm{d}(x)}{2|E| \text{ and } e(x,y) \in E} \frac{1}{\mathrm{d}(x)} = \sum_{x \in \mathscr{C}', y \in \overline{\mathscr{C}'}} \frac{1}{2|E|} \tag{5.5}$$

and:

$$
\begin{aligned}
Q_{\mathrm{CRW}}(\mathscr{C}', \overline{\mathscr{C}'}) = & \sum_{x \in \mathscr{C}'} \sum_{y \in \overline{\mathscr{C}'}, e(x,y) \in E \text{ and } \alpha(x,y) < 1} \frac{\mathrm{d}(y)}{2|E|} \frac{1}{|I(y)|} \frac{1}{|O(y,x)|} \\
& + \sum_{x \in \mathscr{C}'} \sum_{y \in \overline{\mathscr{C}'}, e(x,y) \in E \text{ and } \alpha(x,y) >= 1} \frac{\mathrm{d}(x)}{2|E|} \frac{1}{|I(x)|} \frac{1}{|O(x,y)|}
\end{aligned} \tag{5.6}
$$

To further expand Equation (5.6), we define $f(x,y)$: $f(x,y) = |I(x)| \cdot |O(x,y)|$, where

$$
E[f(x,y)|x] = \sum_{y \in n(x)} \frac{|I(x)||O(x,y)|}{|I(x)||O(x,y)|} = \mathrm{d}(x). \tag{5.7}
$$

Since $\frac{1}{f(x,y)}$ is convex with regard to f(x,y), from Jenson's inequality [26] we have:

$$
\begin{aligned}
E[\frac{\mathrm{d}(x)}{f(x,y)}] &= E[\mathrm{d}(x) \cdot E(\frac{1}{f(x,y)}|x)] \\
&\geq E[\mathrm{d}(x) \cdot \frac{1}{E(f(x,y|x)}] = 1
\end{aligned} \quad , \tag{5.8}
$$

and thus our estimation of $Q_{\mathrm{CRW}}(\mathscr{C}', \overline{\mathscr{C}'})$ is:

$$
\begin{aligned}
E[Q_{\mathrm{CRW}}(\mathscr{C}', \overline{\mathscr{C}'})] = & \frac{1}{2|E|} \cdot ( \sum_{x \in \mathscr{C}'} \sum_{y \in \overline{\mathscr{C}'}, e(x,y) \in E \text{ and } \alpha(x,y) < 1} E[\frac{\mathrm{d}(y)}{f(y,x)}] \\
& + \sum_{x \in \mathscr{C}'} \sum_{y \in \overline{\mathscr{C}'}, e(x,y) \in E \text{ and } \alpha(x,y) >= 1} E[\frac{\mathrm{d}(x)}{f(x,y)}]) \\
& \geq \sum_{x \in \mathscr{C}', y \in \overline{\mathscr{C}'} \text{ and } e(x,y) \in E} \frac{1}{2|E|}
\end{aligned} \tag{5.9}
$$

Finally, combining Equation (2.7), Equation (5.5), Equation (5.6), Equation (5.9) we'll have the conductance over $G$ using CRW:

$$E[\varphi(\mathscr{C}')_{CRW}] = \frac{E[Q_{\mathrm{CRW}}(\mathscr{C}', \overline{\mathscr{C}'})]}{\pi(\mathscr{C}')} \geq \frac{\sum_{x\in\mathscr{C}', y\in\overline{\mathscr{C}'} \text{ and } e(x,y)\in E} \frac{1}{2|E|}}{\pi(\mathscr{C}')} = \varphi(\mathscr{C}')_{SRW} \qquad (5.10)$$

### 5.3.3 Algorithm

Summing up the discussion above, here's a brief look of CRW:

**Input** : $x_0 \in N, C(x_0) = C_0$

**Output** : sample set $\mathscr{X} = (X_1, X_2, X_3, ...X_n), n > 0$

Initialization;

Node $x = x_0$;

**for** $i = 1 \rightarrow$ *Sample_size* **do**

*Send a query to node x, GET attributes*;

*Record node* $x \rightarrow x_i$;

*Get the community-affiliation mapping* $\rightarrow C(y), y \in n(x)$;

*Calculate the number of* $C(y), y \in n(x) \rightarrow k$;

*Uniformly pick a number at random* $j \in (1,k)$;

*Uniformly pick the node* $y \in n(x)$ that $C(y) = C_j$;

*Draw u from* $U(0,1)$;

**if** $u \leq a(x,y)$ **then**

$y \rightarrow x$;

**else**

$x \rightarrow x$;

**end**

**end**

**Algorithm 2:** CRW

### 5.3.4 Experimental evaluation

#### 5.3.4.1 Synthetic data Using the LFR model

As mentioned in Chapter 2, we choose the LFR model that captures both degree distribution features as well as the community structure of large ONs to verify our theoretical claim. Starting from small graph at $|N| = 500$ for the purpose of convenient visualizing, we tested and draw the walk paths of CRW over SRW (see Figure 5.3). The graph is generated at a mixing factor $\mu = 0.1$, which gives $P(e(x,y) \in E \ \ \& \ \ C(x) \neq C(y)) = \mu$. (i.e. the approximate conductance at cuts of communities would be around 0.1). Due to the low cross community probability, SRW spend quite a long period wandering in a small neighborhood before it's able to leave while CRW has already reached most of the communities.

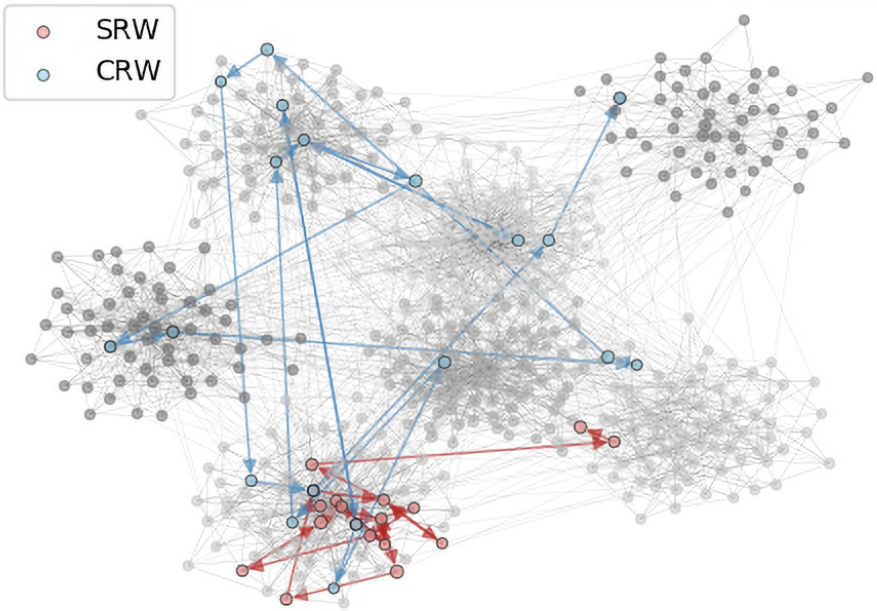



Figure 5.3: SRW and ORW's paths in LFR graph

With the promising performance on small graph, we set up parameter to gain graph of a larger size, and summarized the parameter settings and basic statistics for both graphs we tested in Table 5.1.

Table 5.1: The LFR graphs tested

| Data set | LFR_1 | LFR_2 |
|---|---|---|
| $\|N\|$ | 500 | 5000 |
| $\|E\|$ | 2811 | 71801 |
| $\tau_1$ | 3 | 3 |
| $\tau_2$ | 1.5 | 1.5 |
| $\mu$ | 0.1 | 0.1 |
| $\overline{d(x)}$ | 10 | 25 |
| $\min(\|\mathscr{C}_i\|)$ | 50 | 100 |
| $\|\mathscr{C}\|$ | 8 | 12 |

* $\min(|\mathscr{C}_i|)$ is minimum community size, and $\mathscr{C}$ is total number of communities.

#### 5.3.4.2 Real world data sets

Besides tests on synthetic graphs, we also set up experiments on real world large data sets from SNAP[31] repositories‘ section of networks with ground truth communities. Yang and Leskovec[61] provided a series of networks with community affiliation being set to comprehensive real-world concepts, among which we select three covering different ON topics.

*Youtube* data set[38] contains user created groups where other users can join. Edges represents user's mutual subscription-ships, and users in a group are considered a community.

*DBLP* data set[61] is researcher's co-authorship network that edges are given to those pairs who at least co-publish 1 article. The ground truth community of the data is chosen to be the journals and conferences, where authors who published at least once are considered

a member. Finally, *EU email network*[30] is constructed from members of an European research institution that department affiliation information are used directly as communities. The edges are those who have email interaction for at least once.

**Cleaning of data** As mentioned in Chapter 2, we consider all graph in the context undirected so the data sets are transformed accordingly, and isolated nodes are removed. In fact all random walk related experiments on graphs are tested on the largest connected component on graphs. The used data sets are collected in a manner of measuring best real-world communities on original graphs, thus over-lapping is allowed for different community definitions and not all nodes are assigned a community value. In our experiment we remove all overlapping by assign nodes to the first community we read, and take the induced graph of those who have a community value. The statistics of these data sets are summarized in Table 5.2, and we also noticed that real world network tends to form more communities than synthetic data.

Table 5.2: The Real World data sets tested

| Data set | Youtube | DBLP | EU |
|---|---|---|---|
| $\|N\|$ | 52675 | 260998 | 1005 |
| $\|E\|$ | 318432 | 950059 | 16706 |
| $\overline{d(x)}$ | 12.0904 | 7.28 | 33.25 |
| $\|\mathscr{C}\|$ | 10426 | 12390 | 42 |

We then discuss our experiment measurements followed by our test results.

#### 5.3.4.3 Performance Measures

**Effective sample size** Random walks are MCMC processes known to be collecting samples with high auto-correlations, for which we want to know the i.i.d. equivalence of the chains. ESS is widely used to measure the number of individual draws required to achieve the same expected precision from samples of interest over the same distribution. It's a direct indicator of the sample quality and how well Markov chains have mixed. We run Stan's[8] ESS estimation on our collected samples to compare two methods, and a detailed ESS estimation strategy was illustrated by Geyer. [16]

**Hitting Time** And yet we'll need to verify our claiming that CRW explore entire graph much faster, the regular diagnostics used to monitor convergence (and thus conclude the chain's exploration of sample space) might be less powerful in community structured graphs as it's known that diagnostics could be inconsistent if the chains don't hold over-dispersion. [7] On large ONs with massive sample space ($|N|$), being trapped in small communities is a cause for these false positive, and hence we define the *hitting time* as the community coverage (number of communities sample $S$ reached) at query cost $t$.

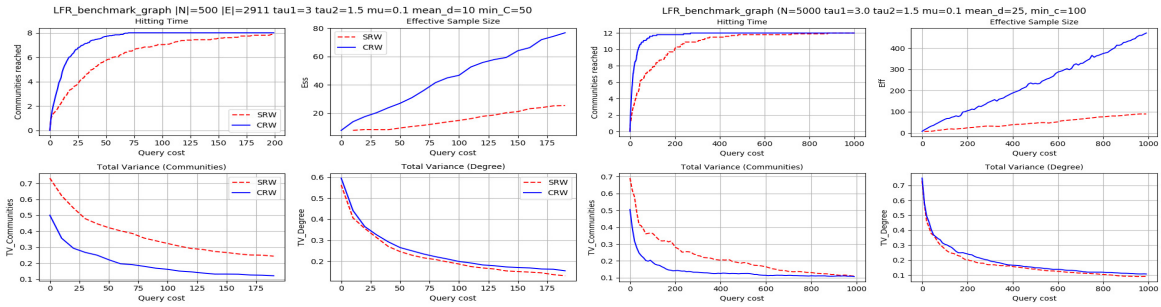
$$H(S,t) = |\mathscr{C}_h(S[:t])|, where\ \mathscr{C}_h(S) = \{\mathscr{C}_i | \exists s \in S, C(s) = \mathscr{C}_i\} \tag{5.11}$$

Hitting time is very explicit sign of whether the random walks are able to explore graph thoroughly.

**Total variance** As described in Section 5.2, random walks are monitored by their distance to the stationary distribution from its sampling distribution. We might use any measurement that detects the distance, and we choose to implement the total variance (or interchangeably L1 norm). [58] In practice the sampling distribution calculation are often intractable, where we applied the empirical distribution over our observed samples. We use both the community irrelevant degree distribution together with community distribution to compare two algorithms.

#### 5.3.4.4 Experimental Results

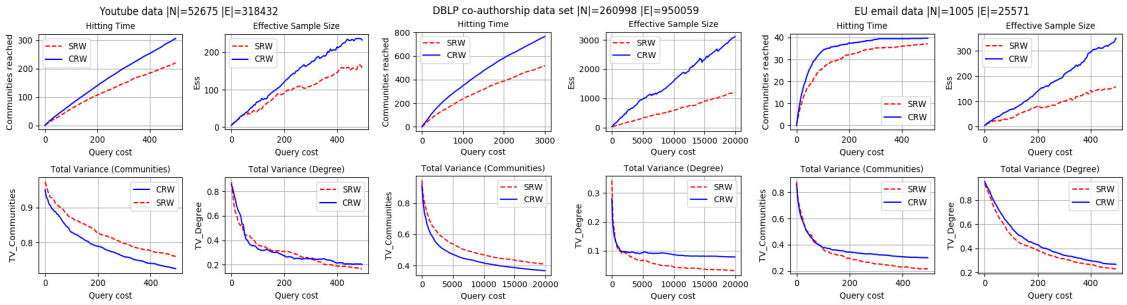
In each single trial of the experiments, starting point of both algorithms are randomly picked at uniform with samples being collected along the paths and the algorithms walk for same length. The random walk length depends on the graph size $|N|$ until we observe approximate convergence. Each data set has been tested for 30 replications through the above process and the mean measurements are thus calculated and plotted.




(a) LFR Benchmark 500 (b) LFR Benchmark 5000

Figure 5.4: Test results of LFR benchmark graphs

Plotted result of the experiments are showed in Figure 5.4 and Figure 5.5. Firstly we see a




(a) Youtube (b) DBLP (c) EU Email

Figure 5.5: Test results of real world data sets

very promising result that CRW overcomes SRW in the manner of ESS on tests over all data sets, which indicates a lower correlation of samples collected by our method. With these being observed, the expected precision of estimators driven by the samples will naturally be higher by definition of ESS. As ESS grows linearly with cost, CRW is in fact collecting usable samples at a higher rate.

Not surprisingly, CRW's dominance in hitting time is also significant. The algorithm travels very fast inter communities at the beginning state that ensure the sample will spread the entire graph. For graphs with large amount of smaller communities, CRW will need a longer time to explore but maintains its advantage over SRW. It then can avoid the false positive in convergence due to an under exploration of the graph.

We compare both methods on total distribution variance with regard to both degree and community distribution. For community irrelevant parameters like degree, the two methods are indifferent in performance since samples from a small portion of graph is representative if degree distribution isn't correlated with position in graph. But if we check for community distribution, we observed CRW's outperforming in general. The result also proved our claiming that SRW might report convergence due to some diagnostics when it's still inconsistent

at other dimension. As a result, for community-structured graphs SRW's estimators might be less precise than expected and CRW can be considered instead.

We noted that our algorithm isn't beating SRW on the EU graph despite its perfectly structured community affiliation (each node belongs to exactly one department with no nodes left). We were confident that department fits the community definition well and shall expect a good performance, so we tested for the conductance of the cuts for all department cuts. The average conductance is as high as around 0.9 while other graphs are much lower. Considering its high average degree for a $|N|$ of 1005, it's not hard to understand that the community structure of the graph is very weak in this data set. This is a sign that well-defined ground truth communities might not agree with a topology wise good community, of which the topic is also explored by some recent literature . [39][24] However, though indifferent in total variance, CRW is still capable of higher ESS rate and faster community reaching.

To conclude, CRW explores graphs with community-structured topology rapidly by crossing through different communities, leading to less correlation of the samples collected than SRW and shall expect a better performance on topology sensitive graph data. As long as a graph's community structure is well established and captured by the chosen affiliation information, CRW is better in all three dimension of measurements we choose, while we'll have to pay attention to the possible disagreement between ground truth community and topology communities.

## 5.4 Dealing with community inference

### 5.4.1 Community detection algorithms

In this section we discuss the very realistic scenario when no perfect community affiliation information is explicitly reachable. We firstly think of using established community detection algorithms which is already very well studied in many works. [5][14]

We first dump the global wise detection algorithms calls for a thorough knowledge of the graph, and found that major methods to infer unknown community affiliation information running locally lies on an exploration starting from the seed node/nodes [47] and try to minimize modularity or maximize node similarity. [39] Either way encounters an local exploration that leads to multiplication of queries we need to know each node. Hence we think of developing our cost-effective algorithms to infer $C(x)$ leveraging the node features as queries to web applications usually comes with metadata other than the graph topology.

### 5.4.2 Naive community affiliation inference

One intuitive idea is to directly choose community affiliation using extra knowledge on feature similarities among nodes in different communities. For instance a similar taste in music genre for music forums, close check-in geo-locations for those applications involving GPS or products with similar key words. [17] These non-rigorous criteria, although not explicitly show strong affiliation, give us an in-expensive alternative to bet on $C(x)$ without any extra cost. We simply pick an (or several if necessary) attribute A, and assign $C(x) = A(x)$.

Pros of the method not only come from the easy implementation and flexibility over all

networks, but taking use of researchers' prior knowledge as well. The con is obviously strong assumption on relevance between community structure and arbitrary attribute we pick, which lies heavily on individual knowledge and preferences. The result might not be consistent per the choice of different community definition.

### 5.4.3 Community affiliation inference with attribute selection

Then we come up with the attribute (or feature) selection techniques that are widely used in statistical learning to select a subset of features best fit to the model of calculating a classification variable. [15] Feature selection detects attributes with the strongest correlation with the output variable through training sufficient inputs and outputs, and then forms a classifier by calculating empirical probabilities over the whole result space given a set of input attributes it selected. We choose to use popular mRMR [41] algorithm in later experiment.

As we mention above, random walks tend to get stuck in densely connected parts corresponding to communities. [43] Therefore it's reasonable to suppose that most samples we gain from very short random walks belong to a same community. I.e., we assume $C(s_0) = C(s_t)$ when t is relatively small. In real world networks, its pretty convenient for us to gain a set of starting nodes who are nearly impossible to be in the same community (e.g. products in totally different languages, topics and categories, people from extremely different geological places and have no explicit common friends). If we are able to perform many short runs starting from the set of seeds, we can get the input and output data points for attribute selection. If we write all short random walks as **S**, the attributes of node x being $A(x)$ with coefficient $\chi$, we'll optimize: $\mathbf{A(S)} \cdot \chi = C(S)$.

It should be pointed out that the sample space by the process above is incomplete, and further our scenario call for a complete partition of the graph, thus we'll be only using mRMR for selection and use the subset being selected directly as an classifier. Calculating for probability is inappropriate for the above reason, and hence we need to limit the number of attributes in subset to avoid over partition. We summarize our idea as follows:

- **Collect Samples using multiple short random walks** We pick $M$ starting nodes and assume $C(x)$ is known by simply assign numerical identifier as the value. Then short SRW with $T$ steps are performed from these seeds. We could repeat the process for $R$ times as well for each starting point and have $T * R * M$ samples (with possible duplicates).

- **Discard irrelevant attributes manually (optional)** The step is optional. Although attribute selections algorithms themselves will detect irrelevant attributes while sometimes large cost from the requirement on sample size are assigned to ensure the accuracy of the result. We would take any effort to avoid query cost, so if some attributes are explicitly irrelevant we would dump it before start running mRMR. It should be noted that this step is not the same as naive inference in last subsection, as we only ignore those that are almost absolutely irrelevant (e.g. communities are very not likely to be based on time of registration especially on those content based networks).

- **Adapt attribute selection algorithms** For all the samples $s$ we collected from the same starting node $s_0$ we assign $C(s) = C(s_0)$, and load the attributes from metadata

of corresponding nodes to mRMR. We limit the number of attributes in our selected subsets to relatively small if the mutual information isn't significantly improved with later variables. Finally we use the selected attributes $\mathbf{A}'$ as a direct classifier, namely $C(x) = C(y) \iff \mathbf{A}'(x) = \mathbf{A}'(y)$.

### 5.4.4 Experiments

We also tested CRW with inference on real world data set. On SNAP we are able to find the amazon product metadata which collected earlier. [29] The data comes with a number of raw attributes including but not limiting to identifier, category hierarchy, sales, etc, in which we extract similar products as edges. After an initial cleaning we found the majority of products, at that moment, is under the large "Books" category, so we take the induced graph of all books nodes as our data source to make the community structure more ambiguous. We also eliminate the identifier as it seem to be randomly distributed and is irrelevant, and remove all nodes appear in similar products but have no meta information. The cleared data set's basic statistics are: $|N| = 270347, |E| = 741124, d(x) = 5.48$.

#### 5.4.4.1 CRW With Naive Selection

It seems to be not hard to decide a naive choice of community affiliation as categories are pretty plausible candidates, while the data set come with a category of up to 7 layers of category hierarchy which makes our choice pretty arbitrary. We decide to be very conservatively using the subcategory of book subjects, and run the same experiment as shown in Section 5.3 (see Figure 5.6).

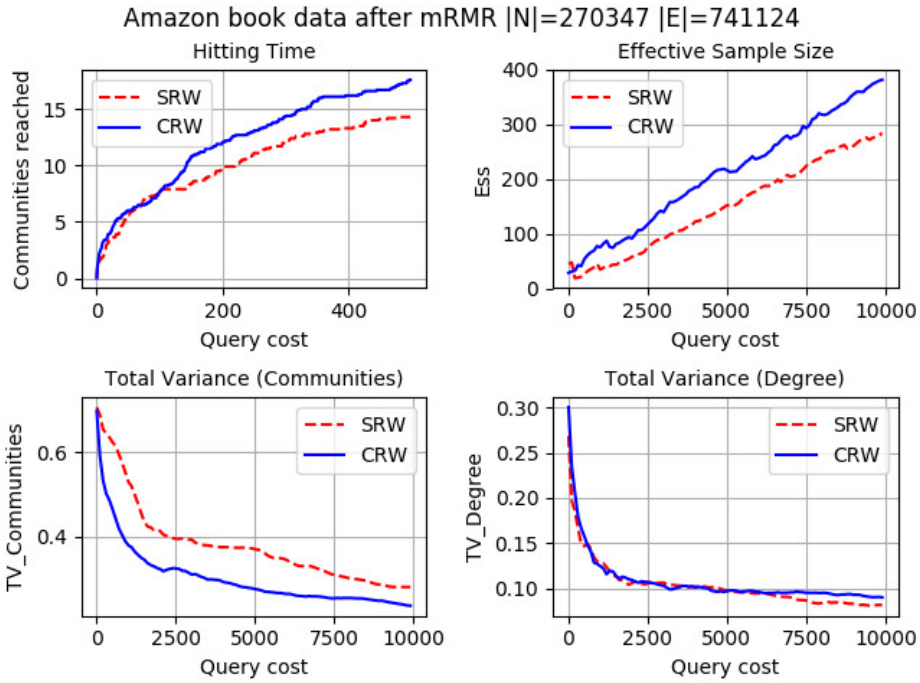



Figure 5.6: The Amazon product data of books under naive attribute selection

The assumption has been strong when the result isn't disappointing though. At no cost CRW is still able to maintain its leading position over SRW, which means the naive assumption did help the diffusion of CRW's exploring the network. At the same time, the shrinkage of our dominance is also the indicator of weaker community structure of our current partition at subcategory.

#### 5.4.4.2 CRW With mRMR

Finally we implement our idea of mRMR powered community inference by selecting the starting seed books from 10 different subjects with widely spread sales ranking and ensure there's no immediate common neighbor. In real world network this can be feasible at acceptable cost, by randomly browsing users/products on different tags/categories and make one query to each of then to get their neighborhood.

We then collect 5 replications of a 5 step short run on each seed, labeling each group numerically as the collecting order. These values are regarded as the output $C(x)$ for mRMR, and all raw metadata from original data are extracted as attributes being inputs. We again labeled categorical data by numbers according to different values, with all missing inputs be filled as -1 (a common strategy for mRMR). We show our analysis results of choosing a size 5 subset in Table 5.3, while other size choices are not giving significant difference.

Table 5.3: mRMR scores of 5 attributes selection

| Order | Variable | Score |
|---|---|---|
| 1 | 3rd level category | 3.201 |
| 2 | subcategory | 0.787 |
| 3 | 4th level category | 0.840 |
| 4 | 5th level category | 0.674 |
| 5 | sales rank | 0.731 |

By our design to avoid over partition of graph, we shall neglect any variables that wont help the model too much, which in this case is dumping all but the 3rd level category as it contributes to the majority of mutual information (mRMR score is 3.201) in the model. We then use the result as a classifier and rerun the experiment.

The result in Figure 5.7 shows a slightly better performance especially in ESS, and yet maintain performance in all other dimensions. In addition, it should be noted that the exploration of communities and the total variations can be very different when community number grows so direct comparison of 2 set of experiment is not perfectly fair, and meantime the subcategory itself is also a fairly good candidate in mRMR. The experiments prove that either method could be effective and chosen accordingly, while mRMR inference backed by

stronger statistical fundamental might be considered first as the extra cost is negligible.

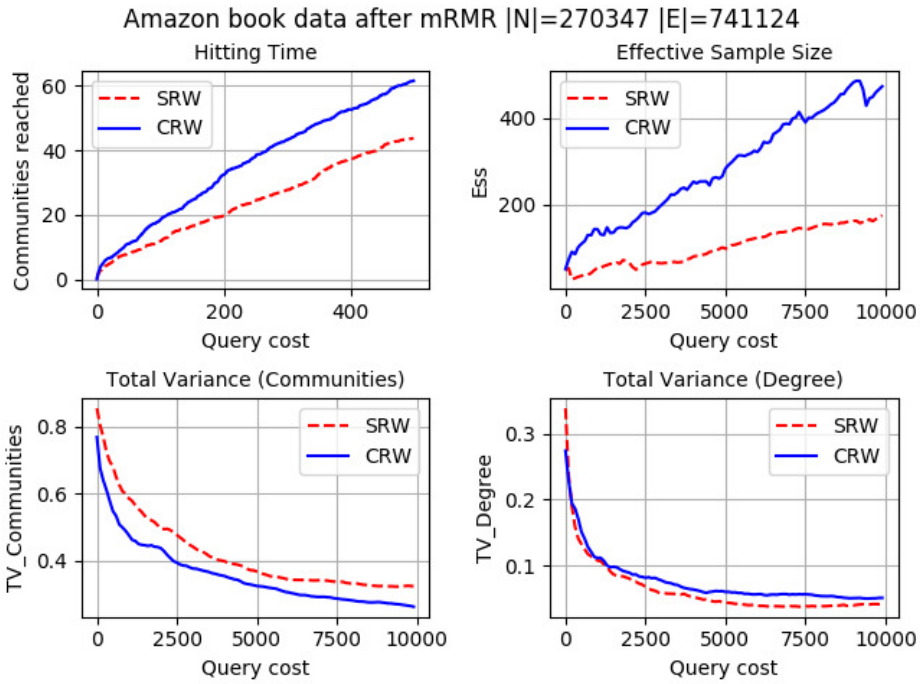



Figure 5.7: The Amazon product data of books under mRMR

## 5.5 Discussions

In this chapter we propose a community information leveraging random walk, CRW, to overcome poor graph conductance caused by the very commonly presence of communities like structure in large online networks. We showed theoretically that the proposed method has improved conductance and thus will converge faster. For graphs without explicit community affiliations we showed how feature selection algorithms can incorporate metadata available and help to select features as ground truth community. Our experiments on both synthetic and real world large networks demonstrate that CRW promisingly explores the network better and gains higher quality samples using same number of queries with SRW being a baseline.

We've thought of some extension discussion options. Firstly the choice of our feature selection algorithm is by our knowledge being mRMR, where there might be newer and more powerful analytical tools [12] as our task involves plenty of categorical data. And our transformation of categorical data right now would simply be mapping them to arbitrarily numerical data, and it might be a source of false correlation. We might discuss the possibly better processing [3] of these variables in future works.

# Chapter 6: Conclusion and Discussions

## 6.1 Summary of accomplishments

In the dissertation I was able to clarify the background and motivation of my topic. The work aims at designing faster sampling and exploring techniques for ONs by leveraging their established topology features into existing framework of random walk, which is widely adopted but too generalized for specific tasks. I proposed several ideas of leveraging different topology characteristics, namely the presence of "hubs" due to the nature of scale-free networks and a commonly observed community structure. The first method(HNRW) extract an induced sub-graph from the original system that consists of small number of high degree node. Random walks being performed on this small sub-graph have shorter mixing time and wider access to most non hub nodes. Then I consider using the hub information to address a classic graph problem, single pair path problem. The HFS is designed to speed up path finding process for any single pair of nodes in a graph, either in A* manner or greedy. The next idea (CRW) takes use of community affiliation information of nodes to force regular random walk to jump to a different community when the current node contains crossing community edges. CRW travels very quickly through communities so as to tackle the challenge of poor conductance occurs in community structured networks like ONs. The dissertation included my basic designs, theoretical analyses as well as experimental evaluations on all proposed approaches to confirm my claiming.

## 6.2 Summary of possible discussions

Though I was able to address the technical challenges I initially defined, during the research and experiments I've been thinking of possible extension of the studies. And the followings are some of possible discussions mentioned before:

- The hub definition in the dissertation is limited to a degree threshold. As discussed in dissertation, other possible solutions than degree cutoff $\varepsilon$ might make good sense as well. (e.g. shall we include those nodes with lower degree but a very high hub degree? Namely those "hub of hubs".)

- Compounded with the topological diversity of different graphs comes the diversity of access interfaces offered by real-world online networks. As all approaches requires certain knowledge from neighboring nodes to determine next step. The interface assumption discussed in the dissertation will give some information of both the current node and its neighbors. Though its true in many cases, different interface designs might be a key factor to whether the method is efficient enough. Queries return neighboring nodes with degrees will apparently affect the results versus those only return nodes.

- The design of HFS enables adjusting heuristics on the fly to boost performance of itself, while BFS makes no evaluations based on information already known which would be quite a waste. With HFS, nodes visited and are known to be connecting to the destination could have an adjusted h value of 0 forcing algorithm to go that way immediately, and for those who aren't in destination's neighborhood the value can be

over 1. As time progress when HFS has a better knowledge of hubs, the search can be even faster.

- Single pair path problem is the first problem I can think of that would take good use of hub information. While other basic graph problems in a topology ambiguous context might also worth to explore using similar strategy in the dissertation. Especially when fast solution detection is more important than finding an optimal is the main requirement.

## Bibliography


[1] Chen Avin, Michal Koucký, and Zvi Lotker. Cover time and mixing time of random walks on dynamic graphs. *Random Structures & Algorithms*, 52(4):576–596, 2018.

[2] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[3] Mohammad Hossein Bateni, Lin Chen, Hossein Esfandiari, Thomas Fu, Vahab S. Mirrokni, and Afshin Rostamizadeh. Categorical feature compression via submodular optimization. *Computing Research Repository*, abs/1904.13389, 2019.

[4] Scott Beamer, Krste Asanović, and David Patterson. Direction-optimizing breadth-first search. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 12:1–12:10, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[5] Punam Bedi and Chhavi Sharma. Community detection in social networks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 6(3):115–135, 2016.

[6] Béla Bollobás. *Modern Graph Theory*. Springer, 1998.

[7] Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455, 1998.

[8] Bob Carpenter, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1):1–32, 2017.

[9] Munmun De Choudhury, Yu-Ru Lin, Hari Sundaram, K. Selçuk Candan, Lexing Xie, and Aisling Kelliher. How does the data sampling strategy impact the discovery of information diffusion in social media? In *ICWSM 2010 - Proceedings of the 4th International AAAI Conference on Weblogs and Social Media*, pages 34–41, 2010.

[10] Mary Kathryn Cowles and Bradley P. Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[11] Mary Kathryn. Cowles, Gareth O. Roberts, and Jeffrey S. Rosenthal. Possible biases induced by mcmc convergence diagnostics. *Journal of Statistical Computation and Simulation*, 64(1):87–104, 1999.

[12] Jianhua Dai, Qinghua Hu, Jinghong Zhang, Hu Hu, and Nenggan Zheng. Attribute selection for partially labeled categorical data by rough set approach. *IEEE Transactions on Cybernetics*, 47(9):2460–2471, Sep 2017.

[13] Hariton Efstathiades, Demetris Antoniades, George Pallis, and Marios Dikaiakos. Distributed large-scale data collection in online social networks. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, pages 373–380, Nov 2016.

[14] Mark E.J. Newman and Michelle Girvan. Finding and Evaluating Community Structure in Networks. *Physical Review E*, 69(2):026113, Feb 2004.

[15] Daniela Witten Trevor Hastie Robert Tibshirani Gareth James. *An Introduction to Statistical Learning : with Applications in R*. Springer, 2013.

[16] Charles J. Geyer. Introduction to markov chain monte carlo. In *Handbook of Markov Chain Monte Carlo*, pages 29–74. Chapman and Hall/CRC, 2011.

[17] Michelle Girvan and Mark E.J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[18] Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, Apr 2010.

[19] Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. Reach for a*: Efficient point-to-point shortest path algorithms. In *2006 Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 129–143, 2005.

[20] Geoffrey Grimmett. Random walks on graphs. In *Probability on Graphs: Random Processes on Graphs and Lattices*, page 1–20. Cambridge University Press, 2010.

[21] Nilsson N. J. Hart, P. E.; and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Transactions on Systems Science and Cybernetics SSC4*, 4(2):100–107, 1968.

[22] Shlomo Havlin, Eduardo Luiggi Lopez, Sergey V. Buldyrev, and Harry Eugene Stanley. Anomalous conductance and diffusion in complex networks. 2016.

[23] Hasan Heydari, S. Mahmoud Taheri, and Kaveh Kavousi. Distributed maximal independent set on scale-free networks. *CoRR*, abs/1804.02513, 2018.

[24] Darko Hric, Richard K. Darst, and Santo Fortunato. Community detection in networks: Structural communities versus ground truth. *Physics Review E*, 90:062805, Dec 2014.

[25] Malek Jebabli, Hocine Cherifi, Chantal Cherifi, and Atef Hamouda. Community detection algorithm evaluation with ground-truth data. *Physica A: Statistical Mechanics and its Applications*, 492:651 – 706, 2018.

[26] Jens Wiik Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30:175–193, 1906.

[27] Liran Katzir and Stephen J. Hardiman. Estimating clustering coefficients and size of social networks via random walk. *ACM Transactions on the Web*, 9(4):19:1–19:20, Sep 2015.

[28] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark Graphs for Testing Community Detection Algorithms. *Physical Review E*, 78(4):046110, Oct 2008.

[29] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1), May 2007.

[30] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph Evolution: Densification and Shrinking Diameters. *ArXiv E-prints*, page physics/0603229, Mar 2006.

[31] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, Jun 2014.

[32] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[33] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2006.

[34] Weimin Li, Jun Xie, Mingjun Xin, and Jun Mo. An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk. *Expert Systems with Applications*, 91:277–285, 2018.

[35] R. Duncan Luce and Albert D. Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[36] Carter T. Butts Maciej Kurant and Athina Markopoulou. Graph size estimation. *arXiv*, 1210.0460(cs.SI), Oct 2012.

[37] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[38] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07)*, Oct 2007.

[39] Mark E.J. Newman and Aaron Clauset. Structure and inference in annotated networks. *Nature Communications*, 7:11863, Jun 2016.

[40] Manos Papagelis, Gautam Das, and Nick Koudas. Sampling online social networks. *IEEE Transactions on Knowledge and Data Engineering*, 25(3):662–676, March 2013.

[41] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, Aug 2005.

[42] Leo Polovets. What are some good algorithms to compute the shortest path between any two people in a social network, 2012.

[43] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Proceedings of the 20th International Conference on Computer and Information Sciences*, pages 284–293. Springer-Verlag, 2005.

[44] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67:026112, Feb 2003.

[45] Stuart J. Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Pearson, 3rd edition, 2009.

[46] Christian Sommer. *Approximate Shortest Path and Distance Queries in Networks*. PhD thesis, The University of Tokyo, 2010.

[47] Xianchao Tang, Tao Xu, Xia Feng, Guoqing Yang, Jing Wang, Qiannan Li, Yanbei Liu, and Xiao Wang. Learning community structures: Global and local perspectives. *Neurocomputing*, 239:249–256, 2017.

[48] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, Aug 2012.

[49] Wikipedia. Scale-free network, 2016. [Online; accessed 4-May-2016].

[50] Wikipedia. Shortest path problem, 2016. [Online; accessed 4-May-2016].

[51] Wikipedia. Six degrees of separation, 2016. [Online; accessed 4-May-2016].

[52] Wikipedia contributors. Expander walk sampling — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Expander_walk_sampling&oldid=720888295`, 2016. [Online; accessed 13-December-2019].

[53] Wikipedia contributors. A* search algorithm — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=A*_search_algorithm&oldid=925009518`, 2019. [Online; accessed 13-December-2019].

[54] Wikipedia Contributors. Cambridge analytica — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Cambridge_Analytica&oldid=896469913`, 2019.

[55] Wikipedia contributors. Mark and recapture — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Mark_and_recapture&oldid=923221236, 2019. [Online; accessed 13-December-2019].

[56] Wikipedia contributors. Markov chain — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Markov_chain&oldid=929413371, 2019. [Online; accessed 12-December-2019].

[57] Wikipedia Contributors. Metropolis–hastings algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Metropolis%E2%80%93Hastings_algorithm&oldid=896920817, 2019.

[58] Wikipedia Contributors. Norm (mathematics) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Norm_(mathematics)&oldid=895918140, 2019.

[59] Robert E. Wilson, Samuel D. Gosling, and Lindsay T. Graham. A review of facebook research in the social sciences. *Perspectives on Psychological Science*, 7(3):203–220, 2012.

[60] Haitao Zheng Ben Y. Zhao Xiaohan Zhao, Alessandra Sala. Efficient shortest paths on massive social graphs. *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 77–86, 2011.

[61] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, Jan 2015.

[62] Naian Yin, Yachao Lu, and Nan Zhang. Speed up random walk by leveraging community affiliation information. *CCF Transactions on Pervasive Computing and Interaction*, Nov 2019.

[63] W. Zeng and R. L. Church. Finding shortest paths on real road networks: the case for a*. *International Journal of Geographical Information Science*, 23(4):531–543, 2009.

[64] Zhuojie Zhou, Nan Zhang, Zhiguo Gong, and Gautam Das. Faster random walks by rewiring online social networks on-the-fly. *ACM Transactions on Database Systems*, 40(4):26:1–26:36, Jan 2016.