

FASTER SAMPLING OVER THEORITICAL AND
ONLINE SOCIAL NETWORKS

by

RAMAKRISHNA ADURI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2015

Copyright © by Ramakrishna Aduri 2015

All Rights Reserved



Acknowledgements

I sincerely acknowledge the efforts of my supervising professor Dr. Gautam Das for guiding me throughout the thesis. I am grateful to Dr. Ramez Elmasri and David Levine for serving on the committee.

I would like to extend my acknowledgements to Saravanan Thirumurugunathan and Azade Nazi for their excellent advice and support.

I would like to thank my family and friends for providing me the moral support.

April 17, 2015

Abstract

FASTER SAMPLING OVER THEORITICAL AND
ONLINE SOCIAL NETWORKS

Ramakrishna Aduri, M.S.

The University of Texas at Arlington, 2015

Supervising Professor: Gautam Das

Online social networks have become very popular recently and are used by millions of users. Researchers increasingly want to leverage the rich variety of information available. However, social networks often feature a web interface that only allows local-neighborhood queries - i.e., given a user of the online social network as input, the system returns the immediate neighbors of the user. Additionally, they also have rate limits that restrict the number of queries issued over a given time period. These restrictions make third party analytics extremely challenging. The traditional approach of using random walks is not effective as they require significant burn-in period before their stationary distribution converges to target distribution. In this thesis, we build a prototype system SN-WALK-ESTIMATER that starts with a much shorter random walk and uses acceptance-rejection sampling to get samples according to a desired distribution. Using only minimal information about the graph such as diameter, SN-WALK-ESTIMATER produces high quality samples with a much lower query cost. We test the system over several theoretical graph families and real world social networks.

Table of Contents

Acknowledgements	iii
Abstract	iv
List of Illustrations	vi
Chapter 1 INTRODUCTION.....	1
Chapter 2 TECHNICAL BACKGROUND	4
Graph Sampling.....	4
Random Walks	5
Acceptance Rejection Sampling.....	9
Performance measures	10
Ideal-Walk.....	11
Walk-Estimate.....	13
Algorithm Walk	15
Unbiased Estimate	15
Variance Reduction Strategies.....	17
Algorithm Estimate	19
Chapter 3 ARCHITECTURE	20
Graph Walker.....	20
Graph Sampler	21
Graph Estimator	23
Chapter 4 USER INTERFACE DESCRIPTION	24
Chapter 5 EXPERIMENTS	27
Chapter 6 CONCLUSION	30
References.....	31
Biographical Information	32

List of Illustrations

Figure 3-1 System Architecture	20
Figure 4-1 Graph Selector.....	24
Figure 4-2 SN-WALK-ESTIMATER	25
Figure 4-3 Query Cost.....	26

Chapter 1

INTRODUCTION

The popularity of social networks has been growing ever since the communication has become online and led to creation of the interactive social networks such as Facebook, Twitter etc. The Facebook and Twitter ranks second and eighth in terms of online traffic according to Alexa, a traffic analytics website [7,8]. It can be observed that most of the users spend their time online not just for email but for social activity over internet via well-defined social networks. An online user can be described as someone who creates a profile on a social networking site such as Facebook and adds personal details which may not be required to be submitted. A user is connected to another user by means of having a similar interest or somebody whom the user might know outside the internet. The social network can be considered as either directed or undirected graph that has users as nodes and their connections as edges. A complex network will have weighted edges as can be seen in case of Facebook which allows the users to rank their connects as per their liking, such as close friend.

The enormous size of the social networks and the combined activity of all the users has created lot of interest among researchers who explore the behavioral traits of individuals. The whole social network can be treated like a breathing apparatus of emotions or expression of individuals reacting to worldwide events, irrespective of whether they are good or bad. It has become a norm until recently that the quickest way to communicate to wide audience is through the social networks.

The social networks have been used differently for different research communities. For example, the sociologists use them as a tool to collect data regarding the online human behavior. Advertising agencies use the data to take advantage of any better marketing strategies for their clients. The engineering researches can analyze how

to design better social networks by observing bottle necks within the existing framework. The social networking companies themselves can analyze the data to create even more user friendly options and cool techniques. They may even try to use the data to optimize the data storage facilities of their data centers. An internet service provider might analyze the data traffic generated by users online to understand the clogs within its delivery network [3].

As described above, the social networks offer the researchers to look for any data related behavioral traits and exploit those patterns to make their own systems better. Despite these networks offer such information clouded in their networks, all of such data is not available due to the fact that most of the social networking site organizations do not provide citing both technical and privacy related legal issues. On the other hand, researches cannot use techniques such as web scraping because of existing limitations on API query limits, login requirements. It is impossible to issue large number of local neighborhood queries because of these restrictions. This leaves only the option of performing sampling over the networks in order to understand the whole network. The sampling can be done over defined statistical methods. However, the employment of statistics requires a complete knowledge of the network to accurately assess the information, lack of it means that the sampling is going to be very difficult [3].

The sampling was carried by graph traversal techniques such as Breadth First Search (BFS). However, BFS induces bias towards high degree nodes and thus is not reliable [9]. The random walks offer the other alternative over the graph traversal techniques. The difference between the two methods is that in graph traversal a node is visited exactly once but in random walk the node can be visited more than once. Simple random walk consists of starting at a node and choose next node in uniformly random fashion.

The random walks such as Simple Random Walk are biased towards high degree nodes because of the inherent convergence of stationary distribution directly dependent on the degree of nodes. Another problem with random walks is their inability to converge to the target distribution quickly and this is called burn-in period. The random walks have an inherent property of selecting the nodes more than once and this can lead to high variance. All these lead to error in aggregate estimation of the social networks. To make matters difficult, if a graph's topology is not known, then it is not possible to calculate sampling distribution [4]. However, the random walks are still better than other techniques because they guarantee a convergence after long burn-in periods. This is because of the properties of Monte Carlo Markov Chain (MCMC) methods that they are based on. A clear example is in the case of Simple Random Walk (SRW), the stationary distribution is in directly proportionate to the degree of a node, eliminating the bottleneck of graph topology.

The key challenge is to have a quick sampling technique which does not compromise on the quality of the sampled data. This thesis will describe building SN-WALK-ESTIMATER which produces sampled distribution in faster convergence to target distribution. The following chapters describe the technical background of the problem and discuss how those problems have been tackled.

Chapter 2

TECHNICAL BACKGROUND

Graph Sampling

There has been explosive growth among online social networks in the last few years and more new users are joining these networks on daily basis. Facebook, with 890 million daily active users, is one example [1] and Twitter, with its 288 active million monthly users who are sending 500 million tweets per day, is another example of the explosive growth [2]. It is apparent that these huge numbers of users being active round the clock present the researchers who analyze the behavior characteristics of such online network of users, an opportunity to decode the common or trendy patterns, which could help them understand how these networks actually behave in various characteristic ways. However, it is not simple to sift through these huge graphs of social networks because of their sheer size besides the ever existing problem of network organizations not providing the data due to various issues ranging from technical to ethical. Even if they did provide the whole data it is very difficult to handle such huge amount of data without facing technical issues. The crawling of entire graph of network is not only practically infeasible but the network organizations also have restrictions on the number of queries that can be issues online. Therefore, it is of paramount importance that there must be a way to shrink the size of social networks so that the smaller graphs of networks can be better analyzed in an efficient manner and are understood to reveal the behavioral patterns of all the users.

Graph Sampling is an efficient method of producing smaller graphs which represent the huge social networks in a way that they can be analyzed at a micro level efficiently and provides the big picture of huge social networks. It is a challenge to create a small graph which efficiently represents the huge graph of social network and acts as a

sample to generate the desired aggregate estimations of sampled nodes which represent the original graph. The graph sampling algorithms have been researched upon over the years and there has been fair evaluation of such graph sampling algorithm candidates. Bread First Search (BFS) is one such early candidate which is biased towards high degree nodes [3]. Random walk algorithms based on Monte Carlo Markov Chain (MCMC) methods are an efficient for sampling nodes from a typical huge online social network. There are traditional random walk algorithms such as Simple Random Walk (SRW) and well known Metropolis-Hastings Random Walk (MHRW). These algorithms suffer from long execution times even though they are efficient. In order to do a faster sampling, a new algorithm called Walk-Estimate is proposed [4].

The model of the graph can be considered as followed. The online social networks are directed edges among nodes which can be deemed as nodes. In practice, the directed edges can be reduced to undirected edges only if the edges exist in both directions between a set of two users. Let V be the set of all vertices in the graph G and E be the set of edges, then $|E|$ can be used to denote the number of edges. For a given node $v \in V$, Let $N(v)$ be the neighbors of the node, and the degree of node becomes $d(v) = |N(v)|$. The goal of graph sampling is to generate a sample of V which is small enough to be analyzed and produce behavioral characteristics of the whole graph.

Random Walks

Markov Chains having a stationary distribution are used in drawing samples so that the drawn samples get support from the distribution. Markov chain exhibits a unique property of drawn samples, wherein, the next drawn sample is dependent only on the current drawn sample but not anything further than it. This rises to important notion of a distribution which does not depend on either the initial state of drawn sample or the

instance at which a sample is drawn. The Markov Chain converges to a stationary distribution [5]. A random walk can be described as MCMC method for a given graph. A random walk starts from a node, say $v_i \in V$, and transits to another node v_{i+1} in the neighborhood of v_i , according to a pre-determined distribution. This distribution over $N(v)$ is referred as transit design [4].

For the MCMC based random walk to converge to a stationary distribution, it has to satisfy three important properties. First is the irreducibility which means the random walk can start at any node in the graph and can produce a non-empty set of nodes with positive probabilities in finite number of iterations. This property is quite suitable for design of random walk for social networks because it eliminates the uncertainty surrounding the starting node. Second is the aperiodicity which will ensure the random walk does not have fixed period of oscillation. Third is the property of random walk being positively recurrent. This means that if an initial node is sampled from a distribution, all the subsequent samples are also sampled according to the same distribution [5].

In Random Walk over the social networks, the samples can be obtained by crawling. The process of crawling involves in selecting an initial node and repeats the process of selecting another node. In each iteration, the initially selected node's neighbors are visited based on how they are selected. Two popular types of random walk are Simple Random Walk (SRW) and Metropolis-Hastings Random Walk (MHRW) which are different from each other on how the neighbors are chosen from starting node.

The Simple Random Walk is biased towards high degree nodes. This is due to fact that the probability of crawling from a node to its neighbor always converges to a stationary distribution. This ensures the crawling happens towards high degree nodes. Since Simple Random Walk is biased towards high degree nodes, the samples obtained tend to be biased while arriving at aggregate estimation.

The SRW can be defined as follows:

"Given graph $G(V, E)$, and a current node $u \in V$, a random walk is called Simple Random Walk if it uniformly at random chooses a neighboring node v from u 's neighbors as the next step. The transition matrix T is

$$T(u, v) = \begin{cases} 1/|N(u)|, & \text{if } v \in N(u) \\ 0, & \text{otherwise.} \end{cases}$$

" [4].

The Metropolis-Hastings Random Walk offers the possibility of correcting the bias induced by Simple Random Walk by altering the transition probabilities. The MHRW modifies the transition probabilities such that the probability of crawling to a smaller degree node is more than crawling to a higher degree node. This is achieved by the following transition matrix:

$$T(u, v) = \begin{cases} \frac{1}{|N(u)|}, & \text{if } v \in N(u) \\ 1 - \sum_{w \in N(u)} T(u, w) & \text{if } u = v \\ 0 & \text{otherwise [4].} \end{cases}$$

The stationary distribution then reduces to uniform distribution, which gives an opportunity to the algorithm to reject some of higher degree nodes thus ensuring the impact of biased samples is reduced [3].

There is an inherent problem of these two random walk, that, they both depend on the convergence of stationary distribution. The convergence of distribution becomes even more significant problem considering a huge graph of social networks. The key question is how quickly the random walks converge or when to stop looking for sampled nodes. This is the key performance characteristics of random walk algorithms and often called by the name "burn-in period". The burn-in period is can be termed as the number of steps that a random walk takes to arrive at stationary distribution [4]. Since the

convergence ensures the elimination of dependency on the starting node, it comes at cost of performance of random walk algorithm.

The Relative Point-wise Distance, a term which describes the largest relative difference between the distribution at a given time and the stationary distribution, can be defined as follows:

$$\Delta(t) = \max_{u,v \in V, v \in N(u)} \left\{ \frac{|T_{uv}^t - \pi(v)|}{\pi(v)} \right\}$$

where T_{uv}^t - the element of T^t with indices u and v

π - the stationary distribution of random walk

In the view of estimating the performance of a random walk, it is essential to note that the above difference ($\Delta(k)$) must be converging to zero. The burn-in period can be understood as the minimum value of k , such that it remains below a threshold value which is a pre-determined value of the relative point-wise distance.

In order to monitor such values for relative point-wise distances, it is imperative to have monitors which may be called as convergence monitors. One such convergence monitor test is Geweke Diagnostic. The basic idea of applying Geweke Diagnostic is to assess two parts of the same random walk and observe the means of each part to determine whether they are different from each other or not. The test is to observe whether two parts have the same distribution so much as that they both are almost equal. The Geweke Diagnostic works on two windows, Window A and Window B, which are created out of “ k ” steps of random walk, with one being 10% of k and other being last 50% of k . For any attribute θ , the z value can be calculated as shown below:

$$Z = \left| \frac{\theta_A - \theta_B}{\sqrt{(S_A + S_B)}} \right|$$

where θ_A and θ_B are means of θ and S_A and S_B are corresponding variances.

[6].

As it can be observed that as z value tends to zero, the stationary distribution converges. Therefore, in practical scenario, since it is hard to achieve absolute zero value to have 100% confidence of achieving the convergence, it is useful to have a threshold of 0.1 to have enough confidence [6].

The property of a graph called spectral gap or eigenvalue helps in understanding the burn-in period of a random walk. Since each random walk has its own transition matrix, its eigenvalues can be computed and used to estimate the length of burn-in period. The spectral gap of a graph is actually the difference between its two largest eigenvalues. However, the largest eigenvalue is 1 corresponding to the stationary distribution. This is denoted by the symbol, λ and is calculated as “1-(second largest eigenvalue)” [4].

Acceptance Rejection Sampling

The unknown graph of online social networks present a problem of drawing samples from the target distribution. The sampling of such nodes becomes difficult to adhere to the target distribution because of its probability density function (pdf). The pdf is complex so that it is almost impossible to directly draw the samples from the target distribution. Therefore, a novel idea of a known distribution's probability distribution function multiplied with a constant, is used in determining whether to accept the sample or reject it. For a sample node of u is drawn with the probability of $p(u)$ while the target distributions probability is $q(u)$. It is multiplied by a constant which is generally a threshold value designed. This is the key challenge in determining the constant such that the number of rejected samples is low. Consequently, a bias is introduced with large threshold value. The target distribution can be computed as shown below:

$$\beta(u) = \frac{q(u)}{p(u)} * \min_{v \in V} \left(\frac{p(v)}{q(v)} \right)$$

However the challenge is to compute $\min_{v \in V} \left(\frac{p(v)}{q(v)} \right)$ because of unknown nature of graph beforehand. A threshold can be set for the purpose of handling $\min_{v \in V} \left(\frac{p(v)}{q(v)} \right)$ even though it can induce certain degree of bias.

Performance measures

The performance of a sampling algorithm is an interesting one to understand. One performance measure can be the query cost which is the total number of steps that the algorithm takes to arrive at the desired sample set of nodes in an online social network. The online social networks do not allow the queries to be executed beyond a certain limit and they usually filter such crawlers which issue the queries incessantly by IP addresses. Or even if the social networks do not have limit on the number of queries issued, it is computationally not possible to derive the sample sets for aggregation estimates. Another performance measure depends on the bias introduced by the algorithm itself. For example, in the case of Simple Random Walk, the drawn samples are biased towards high degree nodes and the actual probability distribution can deviate considerably from the target distribution. Generally, the target distribution can be the uniform distribution which is an unbiased distribution. It is ideal to have the actual distribution can converge to the uniform distribution. However, the drawn samples can have bias as the random walk algorithms are naturally inclined towards high degree nodes. One problem with the bias is that the calculating how different the two distributions, the actual distribution and the target distribution, are from each other and theoretically it can be calculated as a form of vector difference [4].

Ideal-Walk

The existing random walk algorithms have an issue with the burn-in period or the time it takes to converge to the target distribution so that the algorithm can stop exacting more samples. Both the Simple Random Walk and Metropolis-Hasting Random Walk suffer from the problem of converging quickly to the desired set of samples. These two random walks require the walk to be long enough so that the difference between the actual distribution or sample distribution and the target distribution or stationary distribution can be minimal or negligible. Consequently, this results in high query cost which is the number of steps that the algorithm takes to converge. The distance between the two probability distributions can be measure by vector difference and it forms a key performance measure as discussed in preceding section [4].

The difference between the two probability vector changes as the length of random walk increases. This is evident in observing the probability at the starting node which is always 1 and the probability at all other nodes is zero. As a result, the distribution will be skewed at the start. As the random walk progresses, the probabilities at nodes it visits becomes more than zero, making the probability vector to contain positive values. The interesting factor is to observe the rate at which the values of probability vector become positive and consequently, how it affects the distance measure. The values of probability at all nodes become positive as the length of random walk exceeds the diameter of the graph. The maximum value in the set of all nodes experiences the sharpest decline after few initial steps of the walk and the distance between the actual distribution and target distribution also becomes smaller. However, the rate at which the decrease in the distance between the two probability distributions happens slowly after it experiences a drastic fall during the few initial steps of the walk. It is worth to recall that the convergence to absolute zero between the two distributions can

never occur and it takes very large number of steps to achieve quasi-zero difference between the two probability distributions, namely, the actual distribution of the samples and the target distribution. [Reference to Figure 1], As it can be observed, the maximum probability falls drastically after few initial steps of the walk and once the length of the walk exceeds the diameter of the graph, the drop in the value of maximum probability becomes stagnant. Also, it can be seen that the minimum probability sharply rises and then becomes stagnant as well. It can be inferred from the behavior of probability changes during the walk that few initial steps are crucial to major change in the values and once the values change significantly, no matter how many steps that the random walk takes after that, the change becomes negligible. This reveals an interesting point on how random walks behave over a number of steps during the walk that it relies heavily on the initial stages for greater change and later part for much lesser change. This results in high burn-in periods of traditional random walks [4].

On the contrary to the behavior of random walk's convergence to target distribution, the performance of acceptance and rejection sampling behaves in an opposite manner. If the rejection sampling is applied even before the length of the walk is at least as long as the diameter of the graph, it costs heavily and computationally infeasible. On the other hand, once the random walk's length becomes as much as the diameter, the rejection sampling becomes cost effective. This can be proven by way the acceptance rejection sampling depends on the minimum probability that it requires to assess the possibility of accepting a sample. Since at the beginning of the walk, the minimum probability is zero for most of the nodes, it is not possible to accept any sample and as the length of the walk increases, the minimum probability becomes positive allowing the samples to be accepted. Therefore, the time when it happens that the minimum probability reach the level of stability within its value, the acceptance and

rejection sampling can be applied to quicken the process of accepting the samples. Subsequently, the acceptance and rejection sampling also becomes cost effective [4].

There are two patterns emerge from above description of how the convergence can be achieved at different stages of the random walk. One is to simply wait for the random walk to converge to target distribution and the other is to use acceptance rejection sampling to arrive at the convergence fast enough. An observation can be made while deciding when to apply the acceptance rejection sampling that if it is applied when the sample distribution reaches a point where the random walk becomes longer than the diameter of the graph, the acceptance rejection sampling becomes much effective. If a threshold which informs the walk when to apply the acceptance rejection sampling can be estimated, the whole burn-in period of the walk itself can be controlled. The value of the threshold must not be large because it will not help determine when to apply rejection sampling. Therefore, the idea of an Ideal Random Walk could be the one that completes the task of drawing suitable samples quickly enough.

Walk-Estimate

As stated above sections, it is near impossibility that crawling over the entire online social network is possible and it requires many queries to achieve that. To avoid the issue of crawling entire network of nodes to arrive at desired aggregate estimations, the idea of sampling can be used. However, this idea is plagued with convergence to the desired target distribution and results in high query cost, a key performance indicator considering the number of nodes to be sampled. A new algorithm, called Walk-Estimate, is introduced and it is designed to eliminate the delay in reaching the target distribution by random walk [4]. The algorithm takes the advantage of the observations made during the traditional random walk by applying the acceptance rejection sampling after a short

random walk. If the exact point where the rate of change in the maximum value of probability value in the actual distribution becomes almost zero, which means the change in the actual value, becomes stagnant the walk can be stopped and the acceptance rejection sampling can be applied. The Walk-Estimate does not wait for the convergence to happen rather it goes ahead with acceptance rejection sampling after a short walk. The Walk-Estimate does the estimation of probability of sample node $p_t(u)$, say for node u , after certain steps of random walk, say t and use the acceptance rejection sampling to adjust the probability of the node u to the desired target distribution. It is proven that the additional cost of acceptance and rejection sampling is far less than the actual cost of running the traditional random walk in the case of it running more than the diameter of the graph [4]. Thus, the algorithm Walk-Estimate's input is the random walk based on MCMC sampler and outputs the desired sample set of nodes according to the desired target distribution, i.e., the stationary distribution.

The Walk-Estimate consists of two components. One is the Walk and other is the Estimate. The Walk part of the algorithm delivers the short walk and its length while the Estimate part provides the estimation of reaching a sampled node after such short walk. For example, if the sampled node is v and the estimate probability is $p(v)$, after a walk of say t -steps, which is estimated by Walk component, the Estimate is called by Walk to provide the value of $p(v)$. Based on $p(v)$, the Walk then performs the acceptance rejection sampling on the node v [4].

The input to the Walk-Estimate also consists of two parts, one being the design of MCMC sampling algorithm and two being the desired sample size. The aim of the Walk-Estimate is to reduce the wait cost of traditional random walks and produce a set of desired samples which conform to target distribution.

Algorithm Walk

The key challenge for the Walk to work to be aware of the graph topology before it calls the Estimate to estimate the probability at any given node. However, in practice, it is not possible to know the graph topology beforehand and it gives rise to question of how to determine number of steps that the Walk takes before it calls the Estimate. Since it is impossible to estimate the length of short walk or the number of steps that Walk takes, it can only be arrived at after observing the behavior of Walk over a number of graphs. The Walk-Estimate is performed over theoretical graphs, Cycle, Hypercube, Barbell, Tree and Barabasi-Albert [4]. The results showed that the length of the short walk is very small and it can be set to a lower value without compromising on the query cost. A default walk length which is twice the graph diameter can be used and the exact value is 10 for real social networks. However, setting up the length of Walk remains as key challenge as far as Walk is considered and it must be at least the diameter of the graph to allow the Walk to have conservative approach towards final desired sample set.

Unbiased Estimate

The Walk described in the section above calls the Estimate to get probability of the sampled node after a short walk, so that it can apply the acceptance and rejection sampling to determine whether to accept the sample or not. The Estimate produces an unbiased estimate of the sample's probability with negligible query cost. However, the estimation of the simple method produces considerable error because of its high value of variance estimation.

The Unbiased Estimate for a given sample node u , can be computed as below:

$$p_t(u) = \sum_{v \in N(u)} \frac{p_{t-1}(v)}{|N(v)|}$$

Where, $p_t(u)$ is the probability of estimation for the node u ,

$p_{t-1}(v)$ is the probability of node v , which is a neighbor of u ,

$N(v)$ is the degree of the node v .

The probability estimation of node u is dependent on its neighboring node v which in turn depends on its neighboring node and so on. This is a clear recursive process of finding the probability of a given node at a certain random walk length. Since the starting node will have a probability of 1, the limits can be set as $p(\text{starting_node})$ as 1 and for rest of others it could be zero. As the calls return the successive value for corresponding nodes, the probability values at the visited nodes become non-zero and positive values.

It can be proven that the estimate is an unbiased estimation as shown below:

The unbiased estimation of node u is simply,

$$q_t(u) = \sum_{v \in N(u)} \frac{|N(u)| * p_{t-1}(v)}{|N(v)|}$$

where $q_t(u)$ is the unbiased estimate.

The expected value of the $q_t(u)$ can be derived to get the $p_t(u)$, which is the estimated value.

The simple derivation is [4],

$$\begin{aligned} E(q_t(u)) &= \sum_{v \in N(u)} \frac{1}{|N(u)|} \frac{|N(u)|}{|N(v)|} E(q_{t-1}(v)) \\ &= \sum_{v \in N(u)} \frac{1}{|N(v)|} E(q_{t-1}(v)) \\ &= \sum_{v \in N(u)} \frac{1}{|N(v)|} p_{t-1}(v) \\ &= p_t(u) \end{aligned}$$

The Unbiased Estimate algorithm can be stated as:

For Node u , Start node w and Walk Length t ,

Unbiased-Estimate (u,w,t) =

If t == 0 and u == w then return 1

If t == 0 and u != w then return 0

Else return $\frac{p_{uv}}{q_{uv}} * \text{Unbiased-Estimate} (v, w, t-1)$ [4].

However, the error of estimation will be high because of high value of variance. The section below details how to tackle the problem of high variance and presents the method to reduce it.

Variance Reduction Strategies

The sampled nodes should exhibit less variance so that the confidence of the drawn sampled nodes is high and to keep the standard error low. Since the estimation of the nodes is high, it causes high variance and hence the high estimation error. The error should be reduced and it can be proven that the probability of sampled nodes is low aiding the error to be on higher side. This is conspicuous by the error calculation by $\sqrt{(1 - p_t(u))/p_t(u)}$, where the $p_t(u)$ is the probability of the sampled node u . The reduction of the error can be achieved by two stages, one the initial crawling and the other weighted sampling [4].

The initial crawling performs the crawl for the for certain number of hops in the neighborhood of the starting node. For example, for a starting node s , the probability of all its neighbors v , $p(v)$ can be computed as, $1/|N(s)|$. In order to keep the cost of the initial crawling low, the number of hops can be anywhere between 2 to 3 [4]. The cost will be likely to be low because the nodes might have been already visited by the Walk.

The weighted sampling takes advantage of the knowledge of the random walks and back ward estimations performed already and picks the node with the computed

probability. The sample probability of a node u , $p(u)$ can be calculated based on the values of $p_{t-1}(u')$, instead of relying only on the uniform random basis. This is justified because of the varying values of probabilities of neighboring nodes. This can only be performed after the initial crawling because of the initial conditions of probability values will not be different from each other (zero values). In this way, the sampling process can be shifted from being random to being based on weighted sampling of nodes. To retain the unbiased estimate of the probability, the minimum probability of ϵ is assigned to all the nodes and remaining $1-\epsilon$ is assigned in direct proportion to the number of times the node u' is visited during the Walk for $t - 1$ number of steps [4].

All the random walks start from the same node and let us assume the number of such walk are n_{hw} and the number of times the node u' is reached after $(t - 1)$ steps is $n_{u',(t-1)}$. Then the ratio of $\frac{n_{u',(t-1)}}{n_{hw}}$, will have impact on how the node u' is picked.

The algorithm can be summarized as,

WS-BW (, w , t) =

Initiate ϵ

If $t = 0$ and $u = w$, then return 1

If $t = 0$ and $u \neq w$, then return 0

else

$$\pi_{u'} = \frac{\epsilon}{|N(u)|}$$

$$\pi_{u'} = \pi_{u'} + (1-\epsilon) * \frac{n_{u',(t-1)}}{n_{hw}}$$

$$\text{return } \frac{1}{|N(u)| * \pi_v} * \text{WS-BW} (u, w, t - 1)$$

Algorithm Estimate

The algorithm Estimate can be produced by using the two concepts of initial crawling and weighted sampling. To reduce the standard error even further, the algorithm Estimate can be run multiple times and the average of all the runs can be taken as final value. The conspicuous question of how many such executions are possible depends on how efficient the query system is and how much the system can tolerate a given query cost. The number of executions for each node can be evaluated as a proportionate value to that of the estimated variance.

The algorithm Estimate can be stated as follows:

Estimate (w, t, h, F) :

- Start with node w and crawl the neighborhood of w for h hops
- Let V_f be the set of nodes that random walk F hits
- for $u \in V_f$ execute the following:

$$p_t(u) = \text{WS-BW} (u, w, t)$$

Compute estimation variance of $p_t(u)$

end for

Based on the query cost limitations, use WS-BW algorithm to reduce variance

Pick the nodes at random in proportion to their variance [4].

Chapter 3

ARCHITECTURE

The Architecture of the SN-WALKER-ESTIMATE as shown in Figure 4-1, consists of frontend UI which picks up the *json* files from a backend that has two major components, Graph Walker and Graph Sampler, which in turn contains a sub component of Graph Estimator.

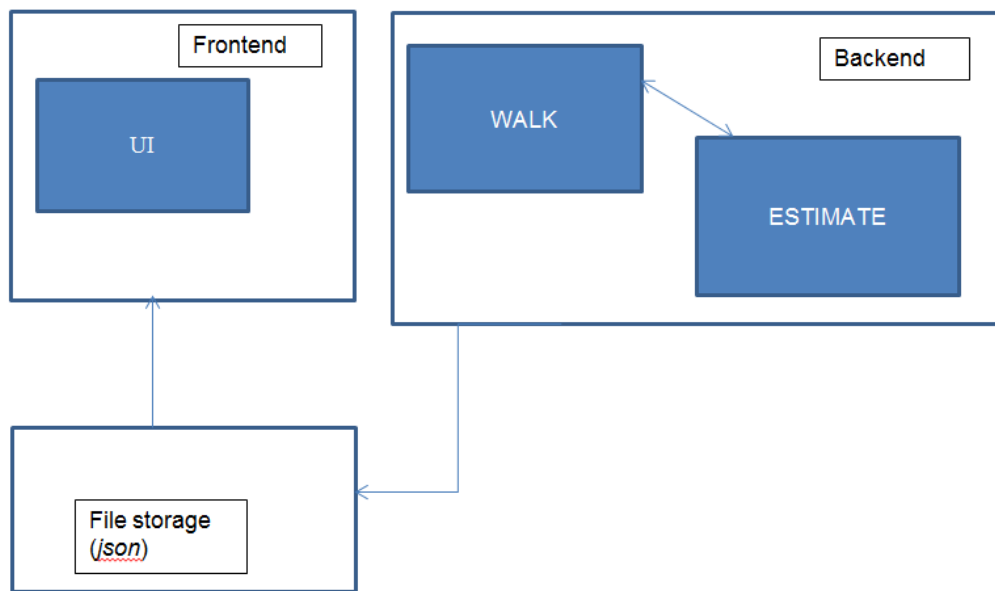


Figure 3-1 System Architecture

Graph Walker

The Graph Walker determines the probabilities for the nodes to be crawled. For the traditional random walk, the nodes are chosen based on either the Metropolis-Hastings Random Walk algorithm or Simple Random Walk algorithm. An initial check window controls the number of nodes to be crawled in the traditional random walk and

using the Geweke diagnostic the crawling can be stopped. The Geweke diagnostic is checked either for the cutoff or an absolute zero difference between the two means. The nodes that are visited during the random walk are set to a cutoff by Geweke diagnostic.

For the Walk-Estimate, the nodes are again chosen according to chosen algorithm of either Simple Random Walk or Metropolis-Hastings Random Walk. However, the probabilities are simultaneously calculated for estimation.

The Graph Walker is implemented by the Walk component of Walk-Estimate algorithm, which produces the short walk after an initial crawling of h-hop neighborhood which is a cost effective method of estimating the probabilities. The Graph Walker after performing a short walk conducts the acceptability of the sample using the acceptance and rejection sampling method. The probability of acceptance and rejection sampling is a fraction of minimum probability at the node which is considered to be the sample and the target distribution at the same node. Once the probability is computed, it is compared to a randomly generated floating number between 0 and 1, to decide whether to accept or reject the sample.

The Graph Walker runs in an iteration of a certain number of random walks so that it reduces the error in the sampling distribution and after it performs a short walk, it calls the component Graph Sampler which operates on the Estimate component of Walk-Estimate Algorithm.

Graph Sampler

In the traditional random walk algorithms, samples are simply obtained by chosen algorithm whether it is MHRW or SRW. The probabilities of the nodes are estimated during the forward walk of the random walk algorithm for certain number of walks which has been set to 100. The visited samples are then calculated for the

probabilities and the last node of the walk is chosen as candidate node after performing the acceptance rejection sampling on it.

The Graph Sampler uses stratified sampling as an option where the probabilities of neighboring nodes are computed to assess the samples. The Walk component of the Walk-Estimate calls the Estimate component to assess the sampling distribution and to perform the acceptance and rejection sampling. The Graph Sampler provides the estimation probabilities in two walks, one the forward walk and other the backward walk. In the forward walk, the Graph Sampler runs the Estimate component of the Walk-Estimate for certain number of walks and collects all the samples with the estimated probabilities. The estimated probabilities are calculated within a loop controlled by the length of the random walk, which is given by the Graph Walker, as a fraction of a constant less than 1 and degree of a particular node.

The Graph Sampler induces high variance because the low probability nodes are not considered and also due to the fact the estimate biased towards high degree nodes. To correct this error, the Graph Sampler uses the unbiased estimate to calculate the probabilities as a backward walk. The backward walk starts at the node which has low probability and tries to find the seed node or the start node from where it started the walk from originally. The backward walk happens exactly to the length of the forward random walk length. However, this is not a trivial case in case of a large graph and it. This is actually a limitation in the SN-WALK-ESTIMATER because of its lower probability of reaching the start node in its backward walk. The Graph Sampler handles such situation with a zero probability and this means that the Graph Sampler looks to conduct more backward random walks in search of finding a positive probability for such nodes.

Graph Estimator

The Graph Estimator takes the samples and computes the metrics such as Average Degree of the samples, number of steps taken to converge and query cost. The Graph Estimator provides the samples over which the aggregation estimation can be performed. In case of the social networks such as Twitter, the in-degree and out-degree of the nodes can be treated as an important parameter when computing the aggregate estimations. The Graph Estimator provides the average degree of confirmed samples from the Walk and Estimate components.

The Graph Estimator also plays the role of file collector and dumps all the relevant files required for the frontend to produce visualizations. However, it is hard to produce all the files for large graphs because they can overload the frontend to cause performance issues.

Chapter 4

USER INTERFACE DESCRIPTION

The User Interface is developed using d3 java script library as show in Figure 4-1. It consists of front-end to display the content and animation of the graph supported by the backend which is a flat file system of *json* files generated by the SN-WALK-ESTIMATOR. The *json* format is designed to capture all the events happening during the random walk. It captures the data for traditional random walk and also generates the data for the SN-WALK-ESTIMATOR. The files are generated to be picked by the front-end as shown in Figure 4-2 which loads these *json* files and populates the nodes, edges of the graph. The *json* files are two types, one being the data containing the graph and the other containing the event data. The front-end initially loads the graph with the nodes and edges being displayed with a certain color. Depending on the event, the animation shows the slow movement of progress of the corresponding event. The events are all clickable links and the data is picked up separately for each clickable event.

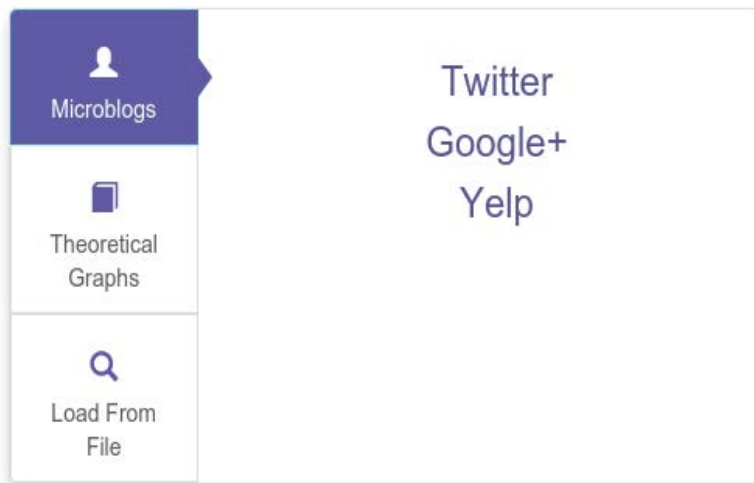
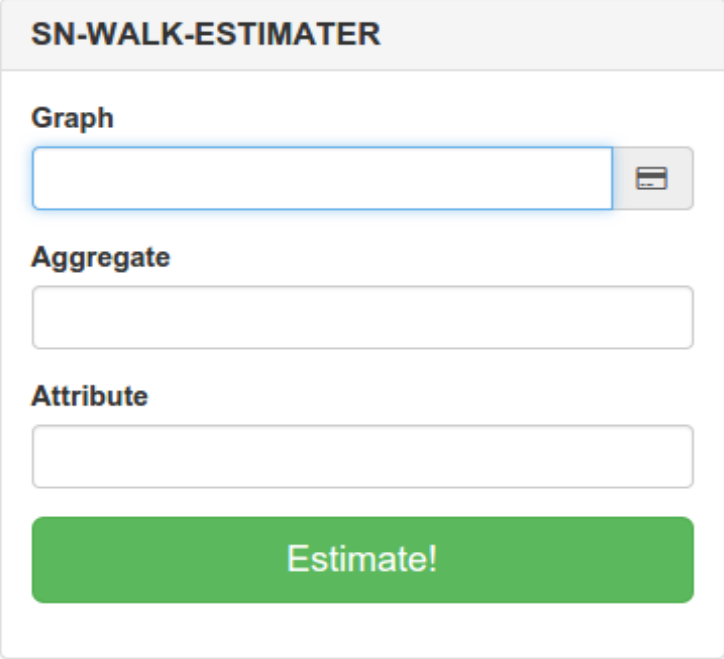


Figure 4-1 Graph Selector

For example, the Fixed Nodes and Potential Samples for the estimation are generated by the algorithm are read from the *json* file and the animation picks up the each path traversed by the algorithm coloring them red and the potential sample to green. Once the potential sample is colored green, it remains so for the entire execution of the event. The events are shown as explained above. The aggregate estimation are shown at the end. These include average degree for the node, the query cost for the entire walk and the accepted samples.



The image shows a web interface titled "SN-WALK-ESTIMATOR". It contains three input fields: "Graph", "Aggregate", and "Attribute". The "Graph" field has a file upload icon on its right side. Below these fields is a large green button labeled "Estimate!".

Figure 4-2 SN-WALK-ESTIMATER

The query cost of random walks is displayed as shown in Figure 4-3. The only drawback of the UI is its inability to display the entire information for each node on a mouse hover event. It just shows only the ID of a particular node which is just an integer. One more drawback of UI is that the performance matching algorithm while creating sub-selection of entire set of nodes is linear, as the search within d3 java script library is of linear nature. For a small graph, the visualization can be observed clearly and as the size of the graph grows, the UI is helpful in only realizing how the random walk performs over different set of nodes.



Figure 4-3 Query Cost

Chapter 5

EXPERIMENTS

The experiments are carried out on both theoretical graphs and real world social networks such as yelp users, google plus and twitter users. The real world datasets are provided by corresponding vendors and contain unique user defined fields with which all the other details can be extracted from other datasets. This is fair as far as the sampling is concerned as the entire properties of each node need not be considered, as it carries no value to the output of the algorithm, which is the sampled nodes converged to target distribution.

Hardware:

The Hardware is a dual-core 2.13 GHz Intel running on Ubuntu 12.04 LTS with 3.9 GiB RAM. The algorithms were developed in python 2.7.

Datasets:

Google plus is one of the largest social networks with its 540 million monthly active Google+ users and 300 million monthly active 'in-stream' users who are just visitors [7]. For the experimental analysis, the dataset is modeled as an undirected graph with users being nodes and circles being edges. The dataset size is about 54 MB containing 16 thousand nodes and 4.5 million edges. The average degree of the graph turned out to be 560 [4]. Yelp, an organization helps connect the people with local businesses like dentists, hair stylists and mechanics, has about 135 million unique visitors and it boasts of users having written over 71 million local reviews [8]. Its dataset can be interpreted for experimental analysis as users being the nodes and edges being the reviews written for same business. The number of nodes in the graph is about 120

thousand with 954 thousand edges. Twitter user dataset has about 81 thousand users with 1.7 million edges. All the datasets above are treated like undirected graphs and the in and out degrees of a node are considered to be its attributes.

The traditional random walk algorithms, the Simple Random Walk (SRW) and Metropolis-Hastings Random Walk (MHRW) are evaluated against the Walk-Estimate algorithm (WE). The Walk-Estimate algorithm can have three versions depending on the variance reduction strategies. The variance reduction strategies are initial crawling and weighted sampling. Therefore the Walk-Estimate algorithm can have three variations as follows: WE-None, WE-Crawl and WE-Weighted [4].

Setting of parameters:

The parameters are set as follows. The Geweke diagnostic convergence (Z) has been used for only traditional random walks and is set to 0.1 ($Z \leq 0.1$). The WE algorithm has the walk length set to $2 * (graph_{diameter}) + 1$. For WE-Crawl which uses only initial crawling for variance reduction, the initial hop values are set to 1 and 2 for Google Plus and Yelp, Twitter respectively. For the weighted sampling the weight indicator ϵ is set to 0.1. The results are averaged for 100 runs and in the Walk-Estimate algorithm, the number of random walks are varied between 100 and 2000 [4].

The performance measures vary for different social networks. For Google Plus, the Average degree and Average number of words in the self-description of a user are considered. For Yelp and Twitter, the concerned graph properties such as the degree, shortest path are considered and in case of Twitter, the average number of followers can be computed from the in and out degree. However, the big challenge is to reduce the bias of the samples in determining these measures. Since it is of almost impossible task

of determining such bias, an error can be computed based on the difference between the actual and estimated aggregated values [4].

Results:

The results obtained showed a marked difference between how the Walk-Estimate performed against the traditional random walks, MHRW and SRW on the grounds of sample bias and query cost. From the figure (Google Plus), the WE algorithm fares better than both traditional random walks, MHRW and SRW. The relative error is less in case of WE and high in both MHRW and SRW. For Yelp, the same performance characteristic of WE is repeated where its relative error is less than that of both MHRW and SRW and the same has been repeated in case of Twitter also. However, since the WE algorithm statistics cannot be taken as absolute values to be compared with, a better way to analyze is to compare within its variants, WE-None, WE-Crawl, WE-weighted and WE. From the tests over Google Plus, WE performed better than all other single variants as expected. However, to test whether the samples obtained by WE are indeed of high quality, the distribution of samples is compared to the target distribution. This is also done for SRW so that both SRW and WE can be compared. As expected, the WE fared better than SRW in terms of the distance measures such as l_∞ and K-L divergence as stated in the table.

Chapter 6

CONCLUSION

In this thesis, the system built was the SN-WALK-ESTIMATER, whose performance over the traditional random walks, Simple Random Walk (SRW) and Metropolis-Hastings Random Walk (MHRW), is better in terms of its faster convergence to the sampling distribution without compromising on the quality of the samples. The SN-WALK-ESTIMATER performs a much shorter random walk and applies acceptance rejection sampling to get the sample set of nodes. It also employs techniques to reduce variance induced by random walks. Its two components one the WALK conducts the short walk and other ESTIMATE provides the necessary estimation of to be sampled nodes. The SN-WALK-ESTIMATER clearly demonstrates that the traditional random walks can be calibrated to improve their performance characteristics by overcoming the inherent problems built into them by their own nature.

References

- [1] <http://newsroom.fb.com/company-info/>
- [2] <https://about.twitter.com/company>
- [3] M. Gjoka, M. Kurant, C. Butts and A. Markopoulou. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *IEEE INFOCOM 2010*.
- [4] A. Nazi, Z. Zhou, S. Thirumuruganathan, N. Zhang and G. Das. Walk, Not Wait: Faster Sampling Over Online Social Networks. *PVLDB 8(6): 678-689*.
- [5] W. R. Gilks. Markov Chain Monte Carlo In Practice. Chapman and Hall/CRC, 1999.
- [6] Z. Zhou, N. Zhang, Z. Gong, and G. Das. Faster random walks by rewiring online social networks on-the-fly. *ICDE, 2013*.
- [7] <http://www.alexa.com/siteinfo/facebook.com>
- [8] <http://www.alexa.com/siteinfo/twitter.com>
- [9] M. Kurant, A. Markopoulou, and P. Thiran, "On the bias of BFS (Breadth First Search)," in Proc. 22nd Int. Teletraffic Congr.

Biographical Information

Ramakrishna Aduri, hails from India, and did his undergraduate studies at Andhra University College of Engineering, Visakhapatnam in the field of Mechanical Engineering and developed an interest in Computer Science. Later, he joined Cognizant Technology Solutions and had worked for six plus years. He intends to learn more and wants to become an entrepreneur in the future. His interests include data analytics and data visualizations.