TOP-K LINK RECOMMENDATION FOR DEVELOPMENT OF P2P SOCIAL NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Yusuf Aytaş

January, 2014

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Hakan Ferhatosmanoğlu(Co-Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Buğra Gedik

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Pınar Şenkul Karagöz

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

TOP-K LINK RECOMMENDATION FOR DEVELOPMENT OF P2P SOCIAL NETWORKS

Yusuf Aytaş
M.S. in Computer Engineering
Supervisor: Prof. Dr. Özgür Ulusoy
Co-Supervisor: Assoc. Prof. Dr. Hakan Ferhatosmanoğlu
January, 2014

The common approach for implementing social networks has been using centralized infrastructures, which inherently include problems of privacy, censorship, scalability, and fault-tolerance. Although decentralized systems offer a natural solution, significant research is needed to build an end-to-end peer-to-peer social network where data is stored among trusted users. The centralized algorithms need to be revisited for a P2P setting, where the nodes have connectivity to only neighbors, have no information of global topology, and may go offline and churn resulting in changes of the graph structure. The social graph algorithms should be designed as robust to node failures and network changes. We model P2P social networks as uncertain graphs where each node can go offline, and we introduce link recommendation algorithms that support the development of decentralized social networks. We propose methods to recommend top-k links to improve the underlying topology and efficiency of the overlay network, while preserving the locality of the social structure. Our approach aims to optimize the probabilistic reachability, improve the robustness of the local network and avoid loss from failures of the peers. We model the problem through discrete optimization and assign a score to each node to capture both the topological connectivity and the social centrality of the corresponding node. We evaluate the proposed methods with respect to performance and quality measures developed for P2P social networks.

Keywords: P2P Social Network, Link Recommendation.

ÖZET

P2P SOSYAL AĞLARI GELİŞTİRMEK İÇİN EN İYİ K BAĞLANTI ÖNERİSİ

Yusuf Aytaş
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Özgür Ulusoy
Ortak Tez Yöneticisi: Doc. Dr. Hakan Ferhatosmanoğlu
Ocak, 2014

Sosyal ağları hayata geçirmek için kullanılan merkezi altyapılar beraberinde gizlilik, sansür, ölçeklenebilirlik ve hataya dayanıklılık sorunlarını getirmektedir. Dağıtılmış sistemler sosyal ağlar için doğal bir çözüm sunsa da, bir uçtan uca sosyal bir ağ oluşturmak için ciddi bir araştırma gereklidir. Merkezi algoritmalar P2P altyapısı kullanıldığında yeniden ele alınmalıdır çünkü P2P altyapıda kişiler sadece komşularını bilmekte, tüm çizgeye ait bilgiden yoksun ve zaman zaman çevrimdışı olabilmektedirler. Sosyal ağ algoritmaları kullanıcıların çevrimdışı kaldığı ve ağın değiştiği durumlara karşı sağlam bir şekilde tasarlanmış olmalıdır. Biz sosyal ağı, kişilerin zaman zaman çevrim dışı olabildiği, belirsiz çizgeler olarak tanımlıyoruz ve bu ağların gelişmesini sağlamak için bağlantı öneri algoritmalarını sunuyoruz. Varolan sosyal ağı geliştirmek için en iyi k tane bağlantı önerisi yaparken sosyal ağın ve yerel yapıların korunması için çalışıyoruz. Hedefimiz olasılığa bağlı ulaşılabilirliği eniyileyerek yerel ağ sağlamlığını artırmak ve kayıplardan doğan hataları en aza indirmektir. Bu problemi her kişiye topolojik bağlılık ve sosyal ağdaki durumuna göre puanlama olarak modelliyoruz. Sunduğumuz yöntemleri geliştirdiğimiz performans ve nitelik ölçüleri ile değerlendiriyoruz.

Anahtar sözcükler: P2P Sosyal Ağ, Bağlantı önerisi.

Acknowledgement

I am grateful to my supervisors Prof. Dr. Özgür Ulusoy and Assoc. Prof. Dr. Hakan Ferhatosmanoğlu for their suggestions and criticisms about my study.

I am thankful to my close friend Levent Sezer for preparing delicious food for thesis committee and audience.

I would like to thank İzzeddin Gür for all of his efforts as we have worked closely to improve this thesis.

I am thankful to Assist. Prof. Dr. Buğra Gedik and Assoc. Prof. Dr. Pınar Şenkul Karagöz for kindly accepting to be in the committee and also for giving their precious time to read and review this thesis.

Last but not least, I would like to thank my small family. I dedicate this thesis to them.

Contents

List of Figures

Chapter 1

Introduction

Online social networks have drawn attention in the last decade with growing number of people using social platforms such as Facebook, Twitter, and LinkedIn. Social network providers offer a variety of services, which result in rich content and linkage data. The common approach of having a single owner administering the data is counter-productive with respect to both systems and practical perspectives. From a social perspective, users do not have the power to safeguard themselves from misuse of their data [1]. The owners of social networks can apply censorships and other exercises of central authority [2]. In decentralized social networks, the peers can maintain data collaboratively and each user can define their own level of privacy. Such a decentralized system is a natural alternative to the current "fat server/thin clients" model for social networks.

1.1 Problem Statement

Although a decentralized system has its clear advantages, it introduces significant challenges in terms of algorithms, topology, storage, updates, and locality [1]. In a P2P network, nodes do not have access to global addressing or routing information. The data flow only through neighbors. The resources available to peers are limited and the nodes may go offline or churn (i.e., join and leave). The

availability of data depends on the availability of the corresponding peers. Hence, the placement of data should consider the relevant and authorized peers as well as their availability. Traditional social network algorithms need to be revisited for P2P infrastructures because they assume a global deterministic graph, i.e., existence of the links and nodes as a priori deterministic.

Considering these challenges, we design a decentralized social network where the connectivity of the peers matches their social network relationship. As the nodes can go offline and churn time to time, we model the network as an uncertain graph where every node has a probability of being available. We introduce the P2P link recommendation problem to support development of a robust decentralized social network. Our focus is to maximize the reachability, i.e., ability to reach a node from others, while preserving the local topology. We model this problem using a discrete optimization framework and determine top-k links to recommend. Note that this problem differs from the traditional link prediction problem in social networks [3]. Here, the recommendation needs to improve both the P2P and social network aspects, and to be computed locally in a distributed fashion. The proposed solution utilizes a probabilistic model for graph reachability computing the availability of the paths between nodes. We introduce an approximate dual optimization that captures the complementary goals of improving the social structure, underlying P2P connections and reachability. A distributed Monte Carlo simulation based approach is used to estimate the reachability of nodes. We also investigate scalable reachability estimations for large-scale networks. Extensive experiments on real and synthetic data illustrate the accuracy and efficiency of the proposed approaches.

1.2 Contributions

In this thesis, we address the P2P link recommendation problem in P2P social networks. This problem addresses how the links should be recommended to the peers in a P2P setting where each peer has only local information about the network. We try to suggest new links to the peers that improve both connections and underlying infrastructure. We formally define reachability for P2P social

networks and present approximate methods for computing reachability in a P2P setting.
Contributions of this thesis can be summarized as follows.

- We study P2P social networks and address the problem of P2P link recommendation.
- We introduce exact and approximate P2P link recommendation algorithms.
- We present exact and approximate methods to calculate reachability.
- We experiment both accuracy and effectiveness of P2P link recommendation algorithms.
- We experiment effectiveness of reachability score.

1.3 Outline

The organization of this thesis is as follows. In Chapter 2, we provide background and related work. In Chapter 3, we discuss how a P2P social network can be implemented, present our graph model, and define reachability based on this model. In Chapter 4, we introduce the problem of link recommendation, our optimization framework to model this problem, and the proposed solutions for top-k link recommendation. In Chapter 5, we present our distributed algorithm for reachability estimation. In Chapter 6, we evaluate experimental results. In Chapter 7, we discuss some important issues about P2P social networks and conclude.

Chapter 2

Related Work

2.1 Social Networks and Link Prediction

Social networks have introduced a variety of research problems such as community detection, influence analysis, ranking, node classification, and link prediction [3]. Nowell and Kleinberg defined the link prediction problem as estimating new interactions between the nodes of a social network [4]. Methods for link prediction rely on content shared among the nodes and topology of the network. Topological methods are based on paths between nodes and neighborhoods [5]. These approaches use shortest path, ensemble of paths or their variants to handle the link prediction problem. Likewise, Bakstrom and Leskovec use the network structure and node/edge attributes to predict new interactions by the help of random walks [6].

Neighborhood approaches, such as Common Neighbors, are used in link prediction. Adamic and Adar use weighted neighborhood information to find relationship between individuals [7]. The intuition is that a node is more likely to interact with another node if the overlap of their neighbors is high. It is a simple heuristic that can often outperform complex heuristics [8].

2.2 P2P Infrastructures

P2P systems enable sharing data and resources between the peers. File sharing applications such as Gnutella and BitTorrent are best-known realization of P2P systems. A P2P framework can also be used to support social network applications. In a decentralized social network, peers collaboratively can serve the needs and requirements of the social network. One can design P2P social networks through super-peers that organize the rest of the network. By using super-peer based architecture, one can overcome problems like recovery and routing, which are more challenging in a fully decentralized system. Buchegger et al. discuss the feasibility of a P2P infrastructure for social networks including distributed storage of data, networking, security, and privacy [1]. In a P2P social network environment, providing a reliable and secure platform is an important challenge. This can be achieved by encryption of data and digestion of access authentication [9]. A potential solution is to use available metadata information, which has some potential side effects [10]. These challenges can be partially addressed by a friend-to-friend (F2F) network or a social overlay approach where the underlying network is formed by social connections. In a F2F system, real life social trust is exploited and data access confined to neighborhood [11].

In a P2P setting, traditional social network problems need to be revisited since a node has neither full information nor control over the network. The fact that each node has partial information about the network, which can evolve dynamically, should be taken into account while implementing algorithms for P2P social networks. In this thesis, we focus on link recommendation and develop a common neighbor based approach to locally gather and merge link strengths from neighbors. We consider this merging problem as a variant of top-k query processing [12] and propose a class of distributed top-k link recommendation algorithms.

2.3 Decentralized Methods

We formally define the problem of P2P link recommendation and propose solutions to improve reachability in P2P uncertain graphs. To the best of our knowledge, this is the first work on link recommendation on an uncertain graph in a P2P setting. However, there is extensive work on the link prediction in a global graph and recently some for local settings.

CNP (Common Neighbor Predictor) predicts future links in a P2P environment by using a distributed algorithm [13]. Although this paper discusses performance in general, they do not focus on P2P performance issues. First, NCNP (Neighbors Common Neighbor Predictor) is proposed that considers neighbors common neighbor when predicting a new link, when at least two neighbors of a node share the same node in common as a neighbor. Later, the algorithm is refined to be popularity aware which considers the weights of the possible links.

SoCS (Social Coordinate Systems) is proposed for link prediction in decentralized social networks [14]. SoCS uses force based graph embedding that depends on iterative forces that are attractions and repulsions. The algorithm calculates the distance between the node and its neighbors neighbors and returns the distances that are less than or equal to an acceptable range. SoCS does not consider a P2P environment.

Our work includes an adaptation of top-k query processing for middleware that filters conditions to get relevant objects [15]. Since the optimality of this algorithm is often achieved in the worst case, TA (Threshold Algorithm) is proposed which is instance optimal [16]. Top-k processing is also discussed in [17] for unstructured P2P networks, focusing on challenges of dynamic structure. Additionally, Theobald et al. present approximate top-k query processing with probabilistic guarantees [18].

We utilize the concept of reachability query within our methods. Yu et al. present a study on reachability queries for directed acyclic graphs [19]. They focus on both space and time consumption to search for a path between two nodes. They compare the algorithmic complexity of the algorithms using query time, index construction time, and index size. But these solutions are not designed for uncertain graphs and have to be reconsidered. To calculate the reachability of a node,

several algorithms are proposed. Zhu et al. give a Monte Carlo based approach to estimate probabilistic reachability queries [20]. Their approach uses a binary tree to estimate the reachability over uncertain graphs in a threshold fashion. It assumes that topological information is available and a binary tree can be generated over possible nodes. The method neither considers a P2P infrastructure nor is applicable to a large-scale social network.

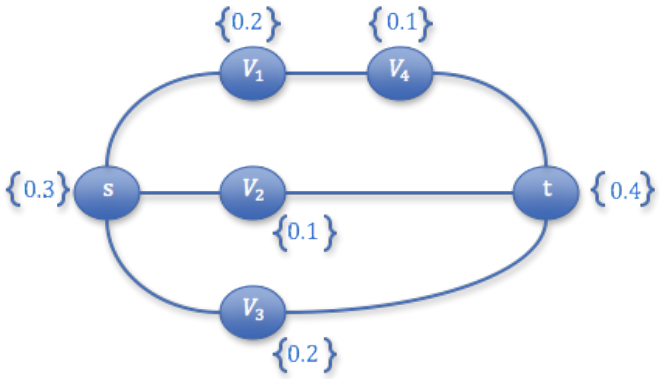

Figure 2.1: Graph with Probabilistic Availabilities

Chapter 3

P2P Social Networks

Implementation of online social networks has been traditionally based on a centralized approach where the server has control of data and waits for the clients to manipulate data. A decentralized approach has clear advantages over this current approach. The challenges on how to store and control the data in a decentralized system are now being discussed in various research communities. For example, a semi-structured architecture has been proposed where super-peers are used to organize the network [21], [22]. The overlay network can be organized according to social connections that provide easy dissemination of updates and address some of the security problems for data maintenance. For instance, Mega et al. focus on building decentralized network on a social overlay by using gossip protocols for efficiently dissemination of updates [23]. The common challenge in a decentralized social network is to maintain both data and connection properties of the peers. This problem does not arise if all the peers were always online, which is the assumption of the current social network algorithms. There is a high probability of peers to churn in P2P networks; hence the availability is an important property to include in any social network algorithm.

We model the P2P social network as an uncertain graph where the nodes become online and offline from time to time. The network needs to grow by introducing new links within local neighborhood that improve the overall robustness, i.e., probabilistic reachability, as we will formally define. We first provide the

definitions used throughout the thesis including the definition of probabilistic reachability in the context of P2P social networks.

Definition (Graph): A graph G(V,E) is defined as a set of vertices $V = (V_1, V_2, ..., V_k)$ with labels $N = (N_1, N_2, ..., N_k)$ and a set of edges $E = (E_1, E_2, ..., E_k)$ between vertices. In our context, labels of the vertices are independent random variables showing the availability of the corresponding nodes. More formally,

$$N_i \sim \textit{Bernoulli}(0,1), i\text{=}1,2, \ \dots \ , k \tag{3.1}$$

and $P(N_i = 1)$ is the probability that the node i is available. If $P(N_i = 0)$ for a node i, then it is apparent that all the paths that pass through the node i will be unavailable. This case is equivalent to removal of the node from the graph. For convenience we assume that availabilities are non-zero, which is $P(N_i = 1) > 0$ for all nodes i.

Path. A path between two nodes s and t is defined as a sequence of edges connecting s to t, or equivalently a sequence of nodes from s to t. For example $L = (s, V_1, V_4, t)$ is a path between s and t in the graph in Figure 2.1. All the paths we consider are simple paths, lacking of any circles.

Availability of a Path. Having defined a path between two nodes; we need to define its availability. We define random variables *R(L)∼Bernouilli(0,1)* for all possible paths L and if *R(L)=1* then the path is available.The availability of a path $L = (N_1, N_2, ..., N_{z+1})$ is obtained using,

$$P(R(L) = 1) = P(N_1 = 1 \wedge N_2 = 1 \wedge ... \wedge N_{z+1} = 1) \tag{3.2}$$

$$= P(N_1 = 1)P(N_2 = 1)P(N_{z+1} = 1) \tag{3.3}$$

Consider the graph in Figure 2.1 $L = (s, V_1, V_4, t)$ a path P(R(L)=1)=0.3*0.2*0.1*0.4. Using commutability property of logical conjunctions, the random variable *R(L)* is equivalent for all the permutations of the nodes in the path. As a special case, let $L_{s\to t} = (s, L_2, ..., L_{z-1}, t)$ be a path from s to t, then the availability of this path is equal to the availability of the same path backwards $L_{t\to s} = (t, L_{z-1}, ..., L_2, s)$, from t to s.

In an uncertain graph, it is important for a node to reach another node to exchange information. The more nodes one can reach, the better it can propagate social updates to others. The reachability of a node, which is the ability to get

through from one vertex to any other, is an important indicator for connectivity of the node to the rest of the network. Consequently, reachability can be used as a measure of connectivity. A formal definition for probabilistic reachability is as follows.

Definition (Probabilistic Reachability): Let *G(V,E)* be a graph where $s,t \in V$, then reachability from s to t is defined as the probability of having at least one available path from s to t, and is denoted by *Re(s,t)*. More formally let $P_{s\to t} = (L_1, L_2, ..., L_x)$ be all the possible paths from s to t, then

$$Re(s,t) = P(\exists L \in P_{s\to t}, R(L) = 1)) \tag{3.4}$$

$$= P(R(L_1) = 1 \vee R(L_2) = 1 \vee ... \vee R(L_x) = 1) \tag{3.5}$$

If there is no path between two nodes, then the reachability is defined as 0. In an undirected graph, the reachability from s to t equals to the reachability from t to s.

Reachability of a Node. The reachability of a node is the probability of existence of at least one path to each of the nodes in the graph, thus

$$Re(s) = P\Big(\forall t \in V - \{s\}, (\exists L \in P_{s\to t}, R(L) = 1)\Big) \tag{3.6}$$

$$P = \left(\forall t \in V - \{s\}, (\bigvee_{L\in P_{s\to t}} R(L) = 1)\right) \tag{3.7}$$

$$P = \left(\bigwedge_{t\in V-\{s\}} \bigvee_{L\in P_{s\to t}} R(L) = 1\right) \tag{3.8}$$

While the definitions of reachability for a node and from one node to another are clear, their computations are not trivial. The computation of the "connectedness" of two nodes or one node to the rest is #P-hard which is as hard as NP-hard [24]. These connectedness measures overlap with our reachability definitions which makes our reachability computation also #P-hard. Thus the exact computations are infeasible on large-scale networks. This motivates us to develop efficient approximation algorithms for reachability estimations. We explain these approximations in detail in Chapter 5.

If the reachability value from s to t is greater than some given threshold, then t is called reachable from s. We formally define this notion of being reachable as

follows.

Definition (Reachable): Given a graph *G(V,E)*, and a threshold value ϵ, node $t \in G(V, E)$ is called reachable from $s \in G(V, E)$, if $Re(s, t) > \epsilon$.

We use *Q(s,t,ϵ)* to denote if t is reachable from s or not. If t is reachable from s using a threshold ϵ, then *Q(s,t,ϵ)=1*, otherwise *Q(s,t,ϵ)=0*. We use *Q(s,ϵ)* to denote the number of nodes that s can reach. For every node in the graph, *Q(s,ϵ)* can be evaluated using

$$Q(s, \epsilon) = \sum_{t \in G, t \neq s} Q(s, t, \epsilon) \tag{3.9}$$

As the peers maintain the data and metadata, the connectivity of the peers is essential for robustness of the network. While forming and extending the network, we aim to increase the reachability to improve the robustness of the local network and avoid loss from failures of the peers. The number of reachable peers needs to be high enough to avoid overloads. Following these observations, we introduce the link recommendation problem in the next chapter.

Chapter 4

P2P Link Recommendation

To develop a robust P2P network, it is essential to set up the right set of connections among the peers. Each new connection would influence the topology of the network and change the storage, search, and routing in the network. New connections need to improve both social and P2P aspects of the system, such as reachability, community structures, bandwidth, and balance of the network. We define "link recommendation" as suggesting a new link to a peer that improves the P2P aspects while preserving its local social structure. Constraining the recommendations to local structures is a key difference from a traditional P2P system as the connections between peers also have a social annotation for us. Accordingly, we aim to generate links that promote P2P aspects such as reachability; however, without damaging social structures like communities by limiting recommendations to be local.

Definition (Link Recommendation in a P2P Social Network). Given a social network *G(V,E)*, the *top-k* link recommendation problem in a P2P environment for a node $s \in V$ is to find a set of nodes $U \subseteq V$ such that

i. s can only ask its neighbors to recommend a node,

ii. each neighbor returns nodes and the reachability values associated with them, and

iii. every $u \in U$ is close to s in a predefined manner (e.g. number of hops)

We model the problem through a discrete optimization framework. Let's assume that $Re_G(s)$ is the reachability of s on graph G. Then our purpose is

$$\begin{aligned} &\underset{t}{maximize} \;\; Re_{G'}(s) \\ &subject\ to\ H(s,t) < \delta \end{aligned} \tag{4.1}$$

where *H(s,t)* is the locality between s and t, and $G' = G'(V, E')$ where $E' = E \cup (s,t)$. $H(s,t)$ can be the number of hops from s to t.

The maximization of (4.1) is cumbersome in a P2P environment as a result of the #P-hardness. Adding an abstract link between two nodes to generate G' affects all the reachability between any pair of nodes. Even if we use a threshold or approximation, the estimation is costly because of the dependence of estimations. To solve this problem, we define the following maximization problem

$$\begin{aligned} &\underset{t}{maximize} \;\; \frac{Re(t)A(t)}{Re(s,t)} \\ &subject\ to\ H(s,t) < \delta \end{aligned} \tag{4.2}$$

where *A(t)* is the availability of t. The approximation comes from our intuition that the recommended node t must have the utility to reach the network and with a low reachability to s. If t is reachable from s, then s can reach other nodes through t with a high reachability. Thus recommending t may not increase the reachability of s.

We also define the following maximization problem as an alternative to (4.2) using our reachable definition instead of reachability

$$\begin{aligned} &\underset{t}{maximize} \frac{Q(t,\epsilon)A(t)}{Q(s,t,\epsilon)} \\ &subject\ to\ H(s,t) < \delta \end{aligned} \tag{4.3}$$

where *Q(s,t,ϵ)* is 1 if s and t are reachable, otherwise a very small number to avoid division by zero, *Q(t,ϵ)* is the number of reachable nodes from t.

We develop a *top-k* link recommendation algorithm on uncertain graphs to solve the introduced problem. The naïve approach would be to examine all possible nodes and obtain *top-k* neighbors that increase the reachability most. This would

become infeasible as the network size grows or the degree of the corresponding node is high. To minimize the communication cost, we propose a variety of methods including adaptations of Fagins approach (FA and TA) for middleware [16] optimized for our problem setting.

In the original *top-k* search problem, a set of objects each with m attributes is assigned scores, each attribute i is sorted on scores and another list L_i is constructed. Each object is assigned an overall score using a fixed monotone aggregation function (i.e., min, average, sum). Using the sorted lists, the purpose is to determine the *top-k* objects having highest (or lowest) overall score.

In our P2P setting, every node corresponds to an object and can assign scores to each of its neighbors, as opposed to a static set of objects and attributes. The scores are essentially the estimated values of each node t in (4.1, 4.2, or 4.3). We develop P2P solutions: $SN{-}FA$ (Social Network analog for FA), $SN{-}TA$ (Social Network analog for TA), and their approximations $SN-TA_\theta$, $SN-TA_{sorted}$, and $SN-TA+$. FA and TA based algorithms use static and a priori available set while the result set is filled iteratively in $SN-FA$ and $SN-TA$. This has the advantage for communication cost if the algorithms stop early since the algorithms may not retrieve all the rows. In the original algorithms, all rows are a priori necessary while $SN-FA$ and $SN-TA$ algorithms can run with having empty rows. These empty rows can be iteratively filled up, or can be discarded if the algorithm stops.

4.1 SN-FA

In $SN-FA$ we use δ=2 and obtain the candidate nodes within *2-hop* distance. $SN-FA$ first initializes an empty score table. The attributes correspond to the neighbors since neighbors will assign scores, and values are the assigned scores of the candidates by each neighbor. There are two phases: First, k candidate nodes are obtained with partially filled scores; second, the unassigned scores for the candidates are filled.

In the first phase, s iteratively asks its neighbors to deliver their top-i^{th} recommendations. Each neighbor u asks each of its neighbors t to return the estimation

of *Re(t)*A(t)*. u estimates *(Re(t)*A(t))/Re(u,t)* for each candidate and returns the top-i^{th} node with the estimated value. s updates the corresponding values in the score table by *(Re(t)A(t))/(Re(u,t)Re(s,u))*. We approximate *Re(s,t)* by *Re(u,t)Re(s,u)*. If we obtain k candidates of which all the attributes are filled, *SN-FA* finishes the first phase, otherwise starts another iteration by asking new neighbors.

Since we may have candidates that have unassigned scores, *SN-FA* starts the second phase to fill the empty entries. *SN-FA* asks the neighbors to collect the corresponding scores for the candidates that are not assigned. If the neighbor does not have a link to a candidate, then its corresponding score is assigned zero. If the data set is all filled, then *SN-FA* terminates with *top-k* candidate nodes. *SN-FA* correctly finds the top-results and is optimal in the worst case if the aggregation function is strictly monotone [15].

The drawback of *SN-FA* is that obtaining all the scores for a candidate may result in delivering all the possible candidates. We handle this problem in *SN-TA*.

4.2 SN-TA

TA was originally proposed to lessen the optimality strictness of FA; it stops at least as early as FA and has instance optimality [15]. Consider the same set up where s holds a score table and fills it with the values retrieved from its neighbors. At each iteration, *SN-TA* calculates a threshold value using the scores of the last encountered candidate. If there are k candidates that have higher rate than the threshold value, the algorithm stops. *SN-TA* always holds the *top-k* result, and discards the others. *SN-TA* reduces the communication cost. As the algorithm stops early and may never require a second phase, the size of the data transmitted is lower than that with *SN-FA*.

Algorithm 1 SN-TA Algorithm

```
recommendations := {}
while true do
  for each neighbor in neighbors do
    recommendation := neighbor.requestRecommendation()
    recommendations ∪ recommendation
  end for
  calculate threshold using last recommendations
  remember top-k so far, discard the others
  if all recommendations are greater than threshold then
    break;
  end if
end while
```

4.3 SN-TA+

In *SN-TA* and *SN-FA*, we use nodes with *2-hop* distance as candidates. However our optimization framework allows *k-hop* distant candidates. *SN-TA+* is a generalization of *SN-TA* such that it recommends nodes within *k-hop* distance. The *k-hop* distant algorithm uses *SN-TA* as a sub procedure. For a given node s, *SN-TA+* iteratively runs *SN-TA* on the candidate nodes and dynamically extends the candidate set.

Let the obtained candidate set at iteration i be CS_i where $\forall u \in CS_i, H(s,t) = i$ and $CS_i \subseteq CS_{i+1}$ for $i = 1, 2, 3, ..., k-1$. Prior to the first iteration, the candidate set is empty and is filled by running *SN-TA* on s. In the second iteration, we run *SN-TA* on CS_1 and obtain CS_2. In the third iteration, we run *SN-TA* on $CS_2 - CS_1$ and CS_3. The algorithm proceeds similarly until we obtain CS_{k-1}. We return *top-k* candidates from CS_{k-1} according to the assigned scores.

We implement another variation of the *SN-TA+* algorithm. Instead of running (k-1) iterations, the algorithm evaluates stopping criteria at each node that it encounters. Given a threshold value p for the score of any candidate node t, *SN-TA+* stops if *Re(t)*A(t)<p*. If the score of the candidate is too small, regardless of the value of *Re(s,t)*, t will have a negligible improvement on the reachability of s. Algorithm 2 illustrates the algorithm.

Algorithm 2 SN-TA+

```
recommendations := {}
call SN-TA()
for each neighbor in neighbors do
  recommendation := neighbor.SN-TA+()
end for
merge all recommendations
get top-k recommendations
```

4.4 $SN - TA_\theta$

One can exploit an upper bound on the threshold to stop earlier with a suboptimal result in *SN-TA*. Given an upper bound θ and current estimation of the threshold τ in SN-TA, θ-approximation is obtained by comparing the last node in the *top-k* list with the $\frac{\tau}{\theta}$ instead of comparing it directly with τ. Although θ-approximation is suboptimal, experiments show that it is considerably faster with a comparable accuracy to SN-TA. We may also obtain a θ-approximation for *SN-TA+* by using $SN - TA_\theta$ in *SN-TA+* instead of *SN-TA*.

4.5 $SN - TA_{sorted}$

$SN - TA_{sorted}$, is another approximation for *SN-TA* based on predicting the total score of a candidate item. The algorithm prunes the candidates that cannot be possibly in *top-k*. In $SN - TA_{sorted}$, s iteratively obtains top-i^{th} candidates from its neighbors with scores, and estimates the minimum average score in the current candidate list. Upon receiving a recommendation, $SN - TA_{sorted}$ updates the corresponding score of the candidate. If the worst score of this candidate is higher than the minimum score, then it is added to the candidate set, and the candidate with minimum score is removed. Otherwise, the candidate is discarded. At the end of the iteration, if the threshold value is less than the minimum score,

then the algorithm terminates and returns the *top-k* set. Otherwise, it continues to collect the candidates.

Algorithm 3 $SN - TA_{sorted}$ Algorithm

```
top-k := {}
candidates := {}
while true do
  for each neighbor in neighbors do
    recommendation := neighbor.requestRecommendation()
    candidates ∪ recommendation
    calculate recommendation.bestScore
    calculate recommendation.worstScore
    if recommendation.worstScore>min-k then
      remove the worst recommendation in top-k
      top-k ∪ recommendation
      add worst recommendation to candidates
    end if
    if recommendation.bestScore<min-k then
      candidates - recommendation
    end if
    threshold := candidates' bestScore
    if threshold<min-k then
      break;
    end if
  end for
end while
```

Chapter 5

Distributed Computation Of Reachability

The reachability and locality values between two nodes need to be estimated in a distributed fashion considering the P2P network constraints. In this chapter, we present our estimation algorithms by starting with a Karp-Luby based Monte Carlo sampling. We then present our scalable reachability approach that exploits local maximum reachability paths between nodes. Finally, we explain our approximations to reachability formulas.

5.1 Computing Reachability by Karp-Luby Sampling

A Monte Carlo sampling approach where the global graph is available was proposed to calculate reachability [20]. This approach considers a setup where an edge is associated with a probability value indicating the confidence of its existence. In our framework, we define reachability based on node availability in a P2P setting. We formalize the problem and explain our Karp-Luby based P2P computation. We first explain the computations as if we have a global view, and

then focus on the P2P structure.

Definition (*k-neighborhood*): Given a graph *G(V,E)* and a node $s \in V$, the *k-neighborhood* of s is defined as the nodes that have a path length smaller or equal to k. More formally, let *h(u,v)* be the number of hops on shortest path between u and v, where $u, v \in V$, and $N_k(u)$ be the *k-neighborhood* of the node u, then

$$u \in N_k(u) \Leftrightarrow h(u, v) \leq k \tag{5.1}$$

If *k=1*, then *k-neighborhood* is simply the neighborhood, and for notational convenience we use $N_1(u) = N(u)$. We calculate the reachability of a node using all the nodes in its *k-neighborhood* and call this the exact calculation. We first build a BFS tree $BFS_G(s, k)$ on graph *G(V,E)* rooted at node s using all the nodes in its *k-neighborhood*. This tree will give us the number of possible paths

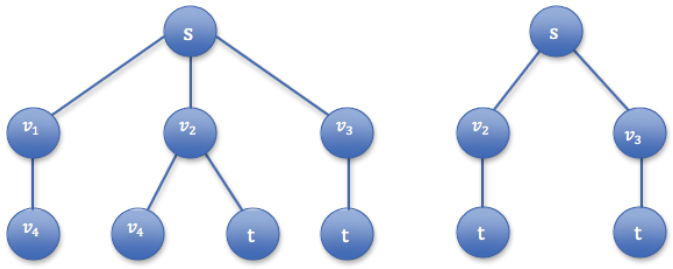

Figure 5.1: The BFS tree rooted on s of the graph in Figure 2.1 and the pruned-BFS tree

between s and any of the nodes in its *k-neighborhood*. The exact reachability of the node s and from s to another can easily be calculated in this BFS tree. We use the subtree that includes t on its leaves to estimate reachability from s to t. We denote this subtree by $BFS_G(s, t, k)$. For convenience, we refer to the former tree as BFS tree and the latter as pruned-BFS tree (Figure 5.1).

We give a possible world definition for an uncertain graph *G(V,E)* to estimate the reachability in a Monte Carlo sample. Then, the results for a BFS tree $BFS_G(s, k)$ are adapted from [25].

Definition (Possible World): Given a graph *G(V,E)*, a possible world is defined as $w = \{N_u | u \in V\}$.

This definition gives us a realization of the graph, where a node is available or not. If the node u is available, then $N_u = 1$, otherwise $N_u = 0$. The space of

all the possible worlds on a graph *G(V,E)* is denoted by *W*. The probability of a possible world can easily be obtained using

$$P_G(w) = \prod_{u \in V} \Big(P(N_u = 1) N_u + P(N_u = 0)(1 - N_u) \Big) \tag{5.2}$$

Next, we define the variable $R_w(s,t)$ in a given possible world w. If a node s can reach another node t in w then $R_w(s,t) = 1$, otherwise $R_w(s,t) = 0$. Also we use $R_w(s)$ as the number of nodes that s can reach in a given possible world.

Algorithm 4 Stopping Rule Algorithm

```
S := 0, λ := e-2, N := 0
γ := 4λln(2/δ)ε^2
γ1 := 1 + (1 + ε)γ;
Re^0(s) := 0
while S < γ1 do
   pick a random sample
   estimate Re^N(s)
   S := S + Re^N(s)
   N := N+1
end while
return γ1/N
```

An uncertain graph *G(V,E)* with a possible world w gives us a deterministic graph and is denoted by $G_w(V_w, E_w)$. The set of all possible deterministic graphs of G is denoted by $G_W(V,E)$. An equivalent form of our reachability between two nodes using a possible world can easily be obtained as follows

$$Re(s,t) = \sum_{w \in W} P_G(w) R_w(s,t) \tag{5.3}$$

The possible world and reachability definitions for our *k-neighborhood* approach can be obtained using the BFS and pruned-BFS trees instead of the graph itself in the original definitions.

The reachability from s to t can be estimated using $R_w(s,t)$ instead of $R_w(s)$ in the procedure we give in Algorithm 4.

We now give an example to illustrate our Karp Luby sampling. Consider the pruned-BFS tree in Figure 5.1. We have four distinct nodes, $\{s, v_2, v_3, t\}$. At

each iteration, we assign either 1 or 0 to each of the nodes randomly. Lets assume that at some iteration we have the sample possible world *w=(1,0,1,1)*. Then the probability of our possible world will become $p_s(1-p_2)p_3p_t$. Next we look if there is any path between s and t. In this sample there is a path over the node v_3, thus $R_w(s,t)=1$. The reachability between s and t for this sample will become $p_s(1-p_2)p_3p_tR_w(s,t)$. Then we iteratively generate another sample and normalize the sum.

KL Sampling in P2P Networks. In case where a node can not obtain the local topology, we have to use sampling in a distributed fashion. The idea is to implement a distributed BFS tree based approach. We obtain a possible world using a Gossip protocol, and estimate the probability of this possible world (5.2). We iteratively generate possible worlds and estimate reachability by (5.3) until the estimation is within a given bound. We initiate a sampling process to differentiate each sampling.

Random Sampling. We start the P2P sampling process in s by asking its neighbors to generate a sample from Bernoulli distribution representing the availability of the node. Then, the available neighbors ask their neighbors and the process continues until we hit all the nodes in *k-neighborhood* of s.

Calculation of $P_G(w)$. Simultaneous to the sampling process, a node also collects the availability of its neighbors. If a node u is exactly *k-hop* distant from s, it returns p_u if it is available, $1-p_u$ otherwise. All the intermediary nodes v returns the multiplication of returned values from its neighbors and p_v if available, $1-p_v$ otherwise. If a node is asked more than once, then the node returns 1 to all subsequent requests other than the first.

Estimation of $R_w(s,t)$. If there is an available path from s to t in a sample w, then $R_w(s,t)=1$, otherwise $R_w(s,t)=0$. We evaluate this simultaneous to the sampling process. If we hit, t then there is an available path from s to t thus $R_w(s,t)=1$, otherwise $R_w(s,t)=0$.

Estimation of $R_w(s)$. The number of nodes that s can reach is estimated similar to the estimation of $R_w(s,t)$. We count the number of distinct nodes that the process hits.

Estimation of $Re^{i+1}(s)$. At the end of the process, s updates its reachability

Algorithm 5 Karp Luby Reachability Algorithm

S := 0, λ := e-2
$\gamma := 4\lambda ln\big(\frac{2}{\delta}\big)\epsilon^2$
$\gamma_2 := 2(1+\epsilon)(1+\sqrt{\epsilon})\big(1+ln\frac{3}{2}/ln\frac{2}{\delta}\big)\gamma$
$\hat{Re} :=$ StoppingRuleRe($min\{\frac{1}{2}, \sqrt{\epsilon}\}, \frac{\delta}{3}$)
let N_0 be the number of steps in StoppingRule
N := $\gamma_2\epsilon/\hat{Re}$, n = min(N,$N_0$)
if $N < N_0$ **then**
  sample $N - N_0$ more
end if
estimate sample variance S^2 using $Re^0, Re^1, ..., Re^n$
$p_z := \max(S^2/n, \epsilon\hat{Re})$
N := $\gamma_2 p_z/\hat{Re}^2$, S := 0
for i=1, ... , N **do**
  S = S + $Re^{(i)}$
end for
return S/N

using (5.3).

We adapt the approach in [25] to build our Karp-Luby based sampling. Algorithm 4 gives the algorithm for Stopping Rule in a P2P setting. The algorithm takes two parameters and iteratively generates a sample using our Random Sampling steps. It returns an approximate reachability value and a set of samples to be used in our main Karp-Luby algorithm.

The procedures for Karp-Luby based reachability estimation are given in Algorithm 5. We first run Stopping Rule algorithm. We then estimate sample variance and generate more samples if needed. Finally we use all the samples we generated to approximate the reachability.

The above approach does not require any knowledge on the local topology of the network, or the values that each peer can hold other than its neighbors. But Monte Carlo sampling is costly for large networks. We provide efficient algorithms that can easily scale to large networks.

5.2 Estimation of Reachability using Maximum Reachable Path

Chen et al. propose an algorithm based on local topology of the network for the #P-hard influence estimation problem [26]. The approach uses shortest paths and assumes that the influence propagates through these paths. Our approach is similar by exploiting shortest paths for reachability estimation. We define Maximum Reachable Path (MRP) as follows.

Definition (MRP): Given a graph *G(V,E)*, lets assume that $P_{s \to t}$ be all the possible paths from s to t. Then the MRP from s to t is the path where the reachability is maximum. More formally,

$$MRP(s,t) = \underset{L}{argmax} \;\; P\Big(R(L) = 1 | L \in P_{s \to t}\Big) \tag{5.4}$$

Ties are broken so that suboptimality property is satisfied, i.e., any subpath from u to v in *MRP(s,t)* is also in *MRP(u,v)*.

MRP(s,t) can be estimated using shortest path algorithms. The availabilities of s is ineffective in the estimation of *MRP(s,t)* because they are always included. So lets adjust the edges so that the weights of the edges are equal to the negative of the log transformation of availability of the predecessor of the edge, i.e., if $(s,u) \in E$ then $w(s,u) = -log(P(N_u = 1))$. The shortest path from s to t will be the maximum reachability path having the maximum reachability value.

MRPs are the building blocks of our estimations. Instead of considering all the possible paths between two nodes, we use *MRPs* to estimate the reachability between two nodes. The reachability estimated on *MRP* structures is a lower bound on exact reachability. To estimate the reachability of a node s to the rest of the graph, we need all the *MRP(s,t)* for all $t \in V$. We propose to use Maximum Reachable Out Arborescence (MROA). We combine all the *MRPs* of a node s to obtain the MROA of s. This structure gives all the necessary information to approximate the reachability from s to any other node. We use a threshold ϵ to eliminate the paths that have a very small reachability.

Definition (MROA): Given a graph *G(V,E)*, and ϵ, the MROA of a node s is

$$MROA(s,\epsilon) = \bigcup_{t \in V, P\left(R(MRP(s,t))=1\right) > \epsilon} MRP(s,t) \tag{5.5}$$

Intuitively *MROA* represents the local region of nodes that a node can reach. Note that as we break ties based on suboptimality, a node can only appear once in an *MROA* and there are no cycles.

In our model, we assume that a node s can reach any other node only through its $MROA(s,\epsilon)$. Thus the reachability from s to t is

$$Re(s,t) = \begin{cases} P\big(R(MRP(s,t)) = 1\big) & \text{if } MRP(s,t) > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

And the reachability of s is

$$Re(s,\epsilon) = \sum_{L \in MROA(s,\epsilon)} P(R(L=1)) \tag{5.7}$$

Also $MROA(s,\epsilon)$ is sufficient to estimate $Q(s,t,\epsilon)$ and $Q(s,t)$ exactly. If $u \in MROA(s,\epsilon)$, then u is reachable from s, i.e., $Q(s,t,\epsilon) = 1$, and otherwise $Q(s,t,\epsilon) = 0$. Furthermore the number of nodes in $MROA(s,\epsilon)$ except s is the number of nodes that s can reach, i.e. $Q(s,\epsilon) = |\{u | u \in MROA(s,\epsilon), u \neq s\}|$.

5.3 Estimation of Reachability using Approximate Reachability Definition

Since exact computation of reachability is infeasible on large-scale networks, we give approximate definitions for reachability.

Approximate Reachability: We relax the dependency in the computation of reachability. We define $Re'(s,t)$ assuming that, all the paths between s and t are independent and then normalize this using the number of all the paths between s and t.

$$Re'(s,t) = \frac{1}{P_{s \to t}} \sum_{L \in P_{s \to t}} P(R(L) = 1) \tag{5.8}$$

It can be shown that $0 \leq Re'(s,t) \leq 1$. We define $Re'(s)$ of a node s as the average reachability of s over all the other nodes in the graph.

$$Re'(s) = \frac{1}{|V - \{s\}|} \sum_{t \in (V - \{s\})} Re'(s,t) \tag{5.9}$$

It can also be shown that $0 \leq Re'(s) \leq 1$. Following those approximations, we offer a heuristic to calculate reachability. Instead of using *Re(s)* directly, we simply multiply the availability values of all the nodes in a path and normalize the sum of these. For the BFS tree in Figure 5.1, the result would be

$$Re'(s) = 1/4 p_2(p_1 p_t + p_2(p_4 + p_t) + p_3 p_t) \tag{5.10}$$

Also for the pruned-BFS tree in Figure 5.1, the result would be

$$Re'(s,t) = 1/2 p_s(p_2 + p_3) p_t \tag{5.11}$$

The estimation is similar to our MC approaches. At each iteration we propagate an estimation-query to all the neighbors of s. If the query reaches a node that is *k-hop* distant from s or cant́ propagate the query (because of cycles) it returns its availability value. Otherwise, the node returns the multiplication of the results returned from its neighbors and its availability value. s estimates its reachability similarly.

The number of paths can be obtained using the same query. At each iteration, if a node is *k-hop* distant from s or cant́ propagate the query, it returns 1. Otherwise, it returns the sum of the values returned by its neighbors. s estimates the number of paths similarly. Algorithm 6 illustrates the algorithm.

Algorithm 6 Approximate Reachability Algorithm

```
result := 1
if k ≠ 0 then
  for each neighbor in neighbors do
    result := result * neighbor.appRe(k-1)
  end for
  result := result * availability
end if
return result;
```

Distributed approximation algorithm may run simultaneously in all peers.

Chapter 6

Experimental Results

To evaluate the proposed algorithms, we designed a P2P social network setting using several real P2P data sets and random graph generators including power-law graphs, small-worlds and clustered graphs. As a baseline comparison, we design local recommendation (LR) algorithm. LR uses all the possible candidate sets that are within *2-hop* or *k-hop* distance in case of *SN-TA+* and chooses k candidates from the set using uniform sampling.
We first evaluate our results on communication cost and show that $SN - TA_\theta$ and $SN - TA_{sorted}$ are preferable. We then compare our approaches on different types of accuracy measures to show the accuracy of the proposed approximations. And finally we evaluate the effectiveness of our approaches on various reachability scores. In all the experiments, the ground truth result set is obtained by $\mathrm{S}N - FA$ and $SN - TA$.

6.1 Datasets

The experiments include three real datasets and several synthetic datasets. The real data sets are: Gnutella[27], Wikivote[28] and Friendster[29]. Gnutella data is one of the snapshots of Gnutella network in 2002. In this snapshot, there are 6301 nodes and 20,777 edges with an average clustering coefficient of 0.0150.

Wikipedia vote network data set includes a small part of the Wikipedia contributors voting each other to become an administrator. Wikipedia voting data is extracted from this election data and vote history having 7115 nodes and 103,689 edges with average clustering coefficient of 0.2089. We use the directed structure of these networks. Furthermore, we use Friendster data set, which is an online gaming network for big data experiments. Friendster was a social networking site where users can form friendship edge each other. Friendster data set consists of 65,608,366 nodes and 1,806,067,135 edges while it has a clustering coefficient of 0.1623. Friendster data set has 4,173,724,142 triangles where fraction of closed triangles is 0.005859.

We also generated synthetic networks using the small world model of Watts and Strogatz [30], the clustering model of Holme and Kim [31], power-law model, and uniform model. We assigned availabilities to the nodes using power-law and uniform distributions from the interval (0,1]. For power-law, we experimented using several values for cut-off and exponent parameters. We varied the density, number of nodes, and number of edges, to generate a variety of results. We generally give average results according to density, number of nodes and number of edges. Note that, all of these graphs are undirected.

6.2 Performance Measures

We first evaluate the efficiency of $SN - FA$, $SN - TA$, $SN - TA_\theta$ and $SN - TA_{sorted}$ algorithms using the communication cost (the number of messages) as the performance measure. We examine the relationship between the number of edges and communication cost on the Gnutella dataset. We removed edges randomly from the Gnutella dataset to have different edge sizes. We executed our algorithms on those graphs and retrieved *top-10* results. In Figure 6.1, we present the performance results. $SN - TA_\theta$ and $SN - TA_{sorted}$ have much lower communication cost compared to $SN - TA$ and $SN - FA$. We also executed algorithms on Wikivote dataset and results were almost the same with the Gnutella dataset.

We also executed our algorithms on all the synthetic datasets to retrieve *top-10*

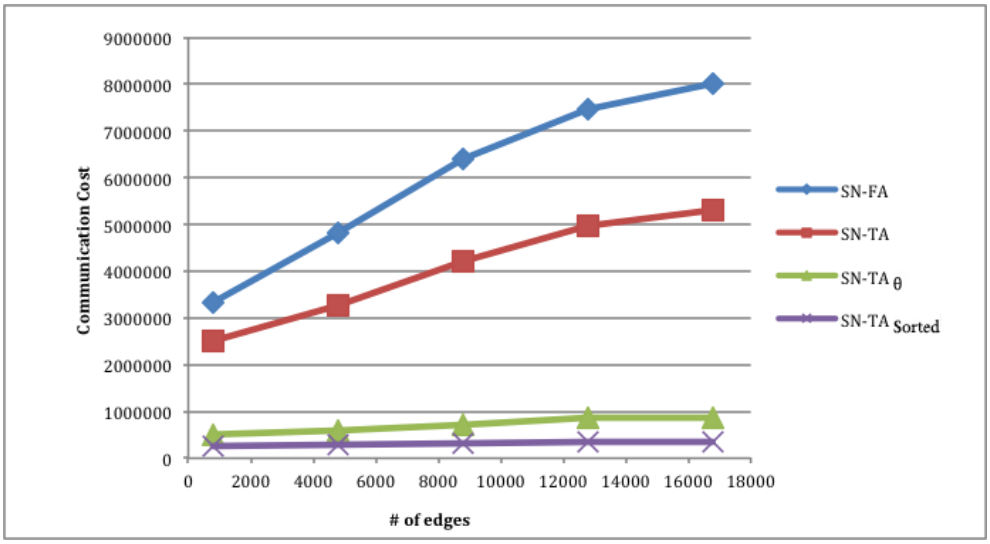

Figure 6.1: Communication Cost vs. Edges for Gnutella Dataset

results and examined the communication cost as a function of the edge size. We provide the average results over all generated networks. As illustrated in Figure 6.2, there is a linear relationship between the communication cost and the edge size in all of the algorithms. There is a large gap between $SN-TA$ and its approximations $SN-TA_\theta$ and $SN-TA_{sorted}$. $SN-FA$ and $SN-TA$ have almost the same communication cost. These results also support our findings on real datasets. $SN-TA_\theta$ and $SN-TA_{sorted}$ are more scalable than their counterparts.

In the next experiment, we use Friendster dataset to evaluate the performance of our algorithms on big data. In this experiment, we present the cost results on different vertices with varying number of edges, ranging from 23 to 1092. We have chosen the vertices with 23 edges as the starting point, and performed experiments with increasing number of edges. The average number of edges in Friendster is 28. Since our algorithms are local and do not need global network information, we have obtained similar results to the previous findings. As the number of edges increases, the communication cost for $SN-TA$ and $SN-FA$ grows exponentially. On the other hand, $SN-TA_\theta$ and $SN-TA_{sorted}$ seem to be stable regardless of the edges size. We visualize the result in Figure 6.3 where we have the similar patterns to the previous results. Consequently, big data does not cause problems since we do not need global network information.

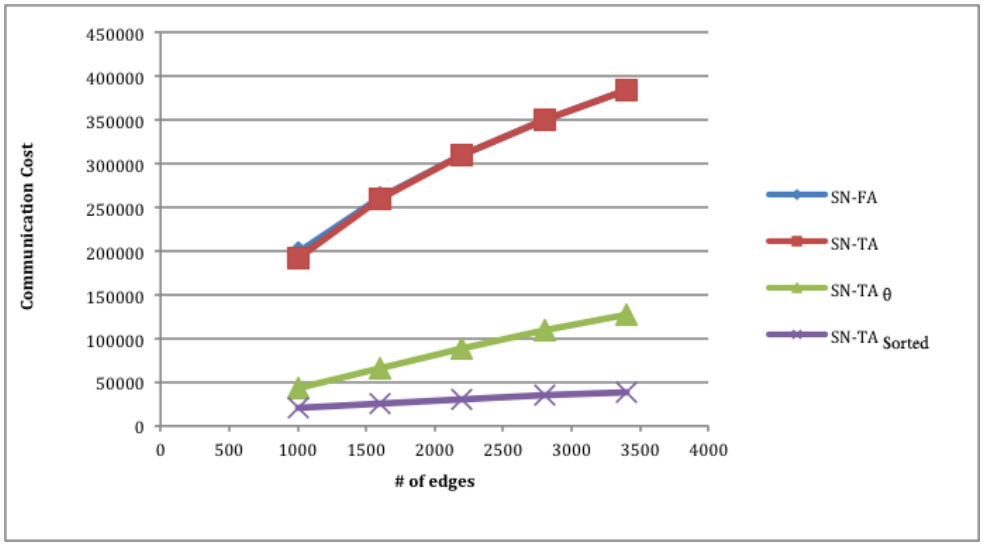

Figure 6.2: Communication Cost vs. Edges for Synthetic Dataset

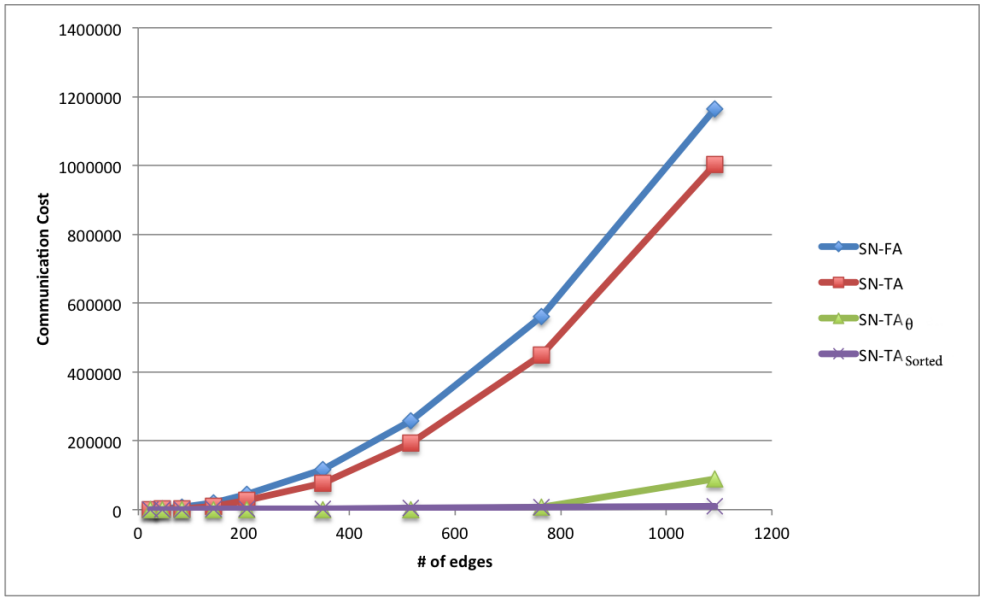

Figure 6.3: Communication Cost vs. Edges for Friendster Dataset

6.3 Accuracy Measures

While $SN - TA_\theta$ and $SN - TA_{sorted}$ are more communication friendly, we now examine their accuracy with varying densities. We evaluate our results on synthetic graphs. We execute the algorithms to retrieve the *top-10* results. We then evaluate those results according to instance-based, rank-based and finally weight-based approaches. Instance-based accuracy is the number of true-positives (TP) in the true result set. The rank based accuracy is the ratio between sum of the ranks of the retrieved result set and sum of the ranks of the correct result set. Ranks are assigned according to their ranks in correct result set, i.e. first having the highest rank and last having the lowest. The weight-based accuracy is the ratio between the sum of scores for result set and sum of the scores for correct result set.

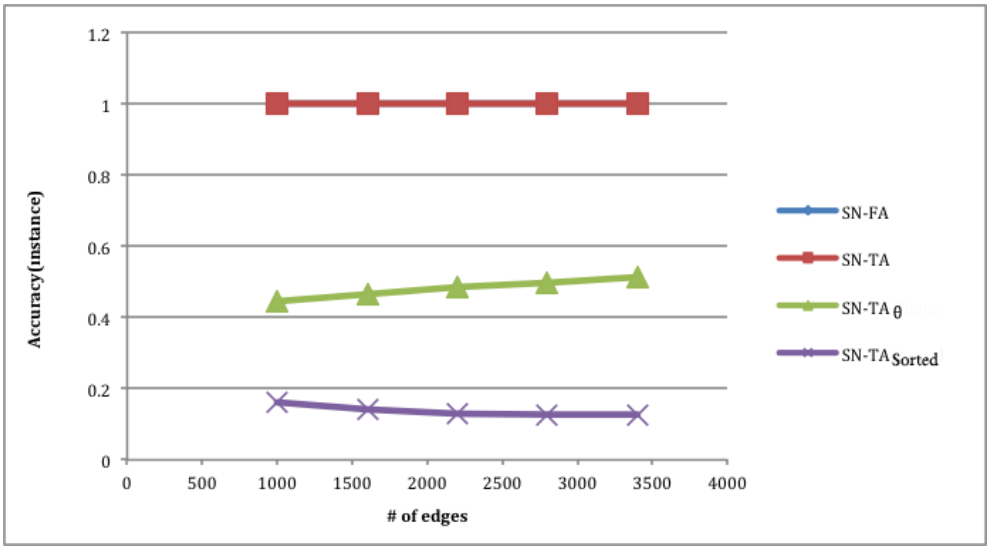

Figure 6.4: Algorithms in Instance Based Accuracy

We first used instance-based accuracy. Figure 6.4 illustrates that there is a significant difference between the approximations $SN - TA_\theta$, $SN - TA_{sorted}$ and

$SN - FA$ (or $SN - TA$). $SN - TA_\theta$ is clearly better than $SN - TA_{sorted}$. Although there are slight changes in accuracy, there is no significant difference in the results according to the density.

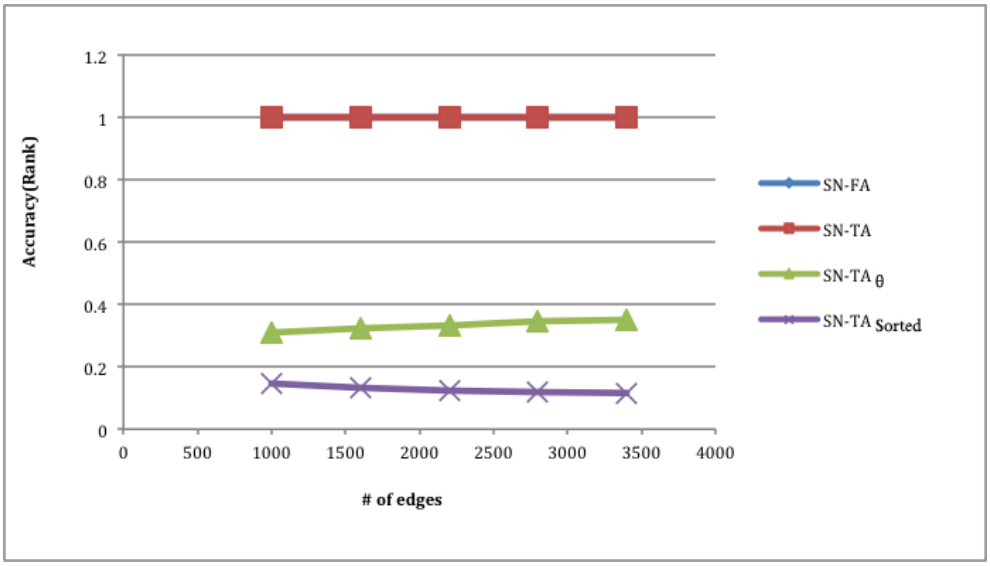

Figure 6.5: Algorithms in Rank Based Accuracy

Figure 6.5 presents the results on the rank-based accuracy, which have a similar pattern to the instance-based accuracy. The number of edges has a negligible effect on the rank-based accuracy. Although the gap between $SN - TA_\theta$, $SN - TA_{sorted}$ diminishes, the rank of the algorithms stands still. We present our results on weight-based evaluation in Figure 6.6. $SN - TA_\theta$ gets very close to the optimal result. Any result set that is returned misses only one or two correct results. This shows a difference from both instance-based and rank-based approaches.

6.4 Reachability Score Effectiveness

We present the results of two different reachability experiments: first evaluating boolean reachability value, and next using the probability estimation of reachability. On evaluating a reachability query, we use 0.1 as our threshold value in

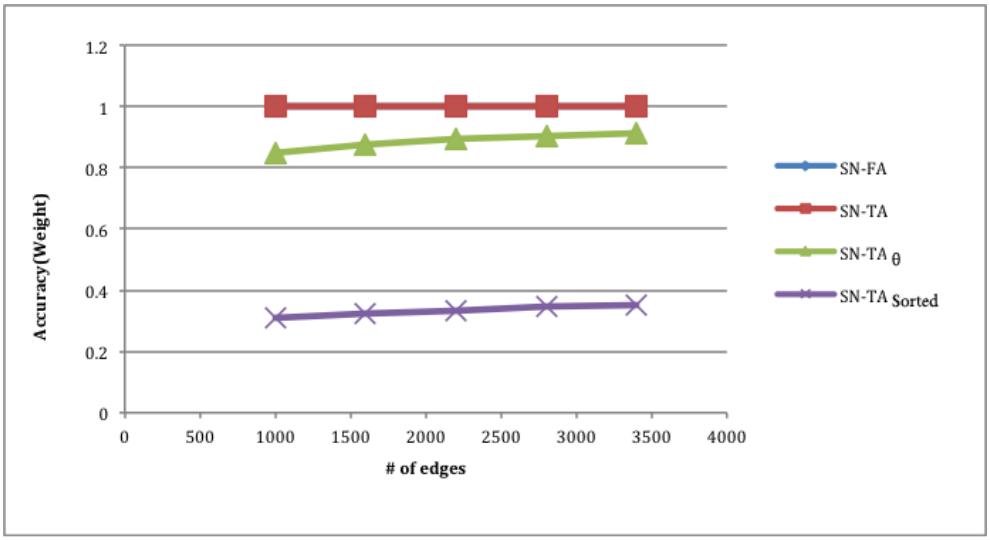

Figure 6.6: Algorithms in Weight Based Accuracy

our experiments. We first run our experiments on WikiVote dataset using randomly chosen 1000 nodes. We employ our MROA based algorithm to estimate the reachability values. The average number of reachable nodes and the clustering coefficient with each recommendation are presented in Figures 6.7 and 6.8, respectively we use clustering coefficient to show how the social structure of the sample changes.

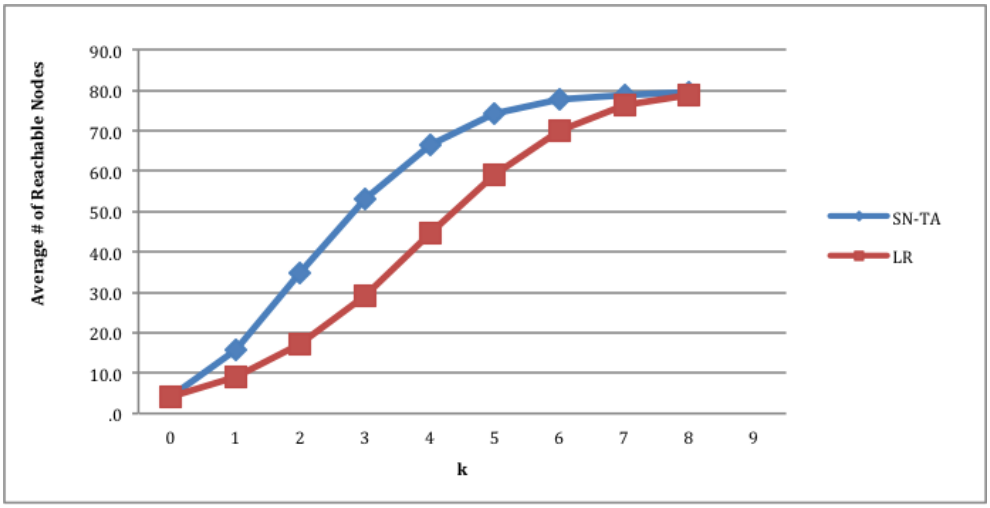

Figure 6.7: Average reachable nodes vs. number of recommendations for WikiVote Dataset

Figure 6.7 shows that $SN - TA$ is clearly better than local recommendation

(LR) in terms of the average number of reachable nodes. As we recommend more nodes, the difference between $SN - TA$ and LR decreases. $SN - TA$ reaches a saturation point where the graph is reachable as much as possible.

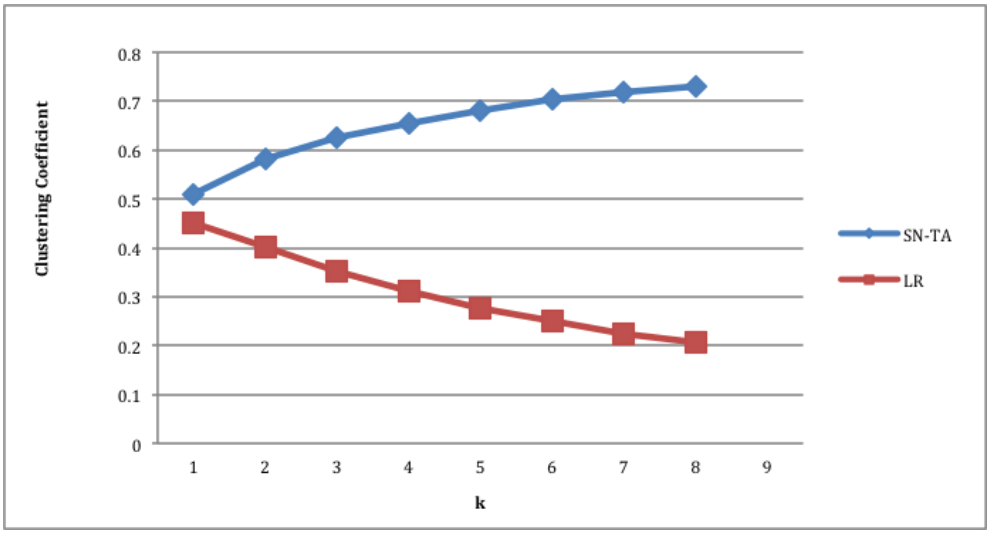

Figure 6.8: Clustering coefficient vs. number of recommendations for WikiVote Dataset

As $SN - TA$ recommends the best possible nodes, after a while the nodes will reach all the possible nodes that they can reach, and recommending another node will not make a significant difference. $SN - TA$ converges in a few iterations. Figure 6.8 shows the results on how the algorithms affect the clustering coefficients for WikiVote. As the nodes are recommended, $SN - TA$ has always better clustering coefficient than the original, preserving the social structure of the underlying network. It recommends local and more central nodes, thus improving the local structure of the P2P social network. However, LR reduces the clustering coefficient below the original after recommending 19 nodes. Recommending only one node (i.e., k=1) significantly increases the clustering coefficient. But as we proceed, the clustering coefficient drops. The reason is that, the first recommendation is taken from a very close circle of a node. Thus the number of cliques the nodes shares increases vastly. But in a large network, recommending the first node greatly extends the social circle of a node causing the drop in clustering coefficient as we recommend more nodes.

Figure 6.9 shows our findings on the average number of reachable nodes using

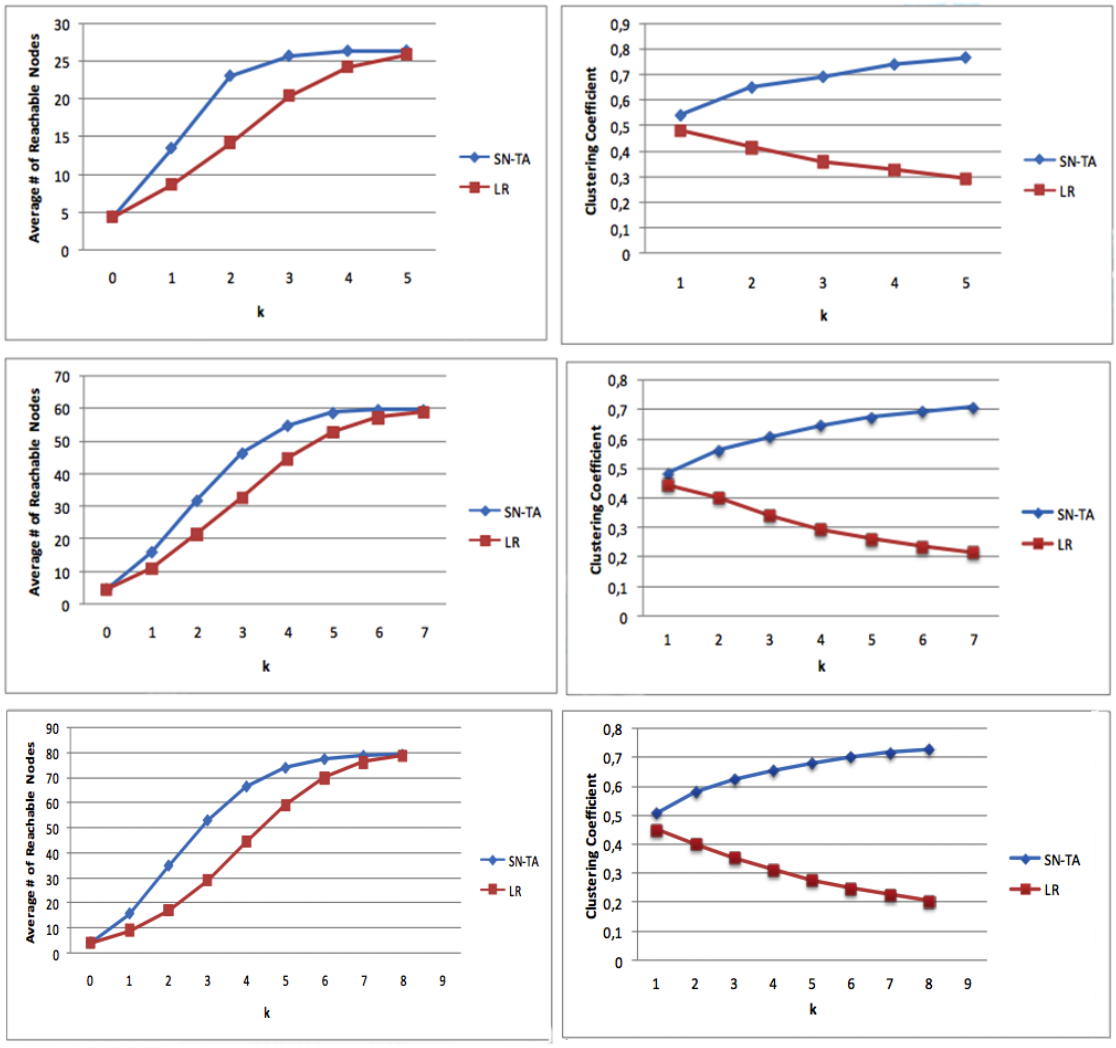

Figure 6.9: Average reachable nodes/Clustering Coefficients vs. number of recommendations for 200, 400 and 600 nodes

synthetic datasets. We present the results according to the density, which are similar to those obtained with the WikiVote dataset. $SN-TA$ outperforms the LR approach in all of the cases. There is a sharp increase in the average reachability for the first recommendations of $SN-TA$. Then we reach a saturation point where recommending any node will not make a significant difference.
Figure 6.9 presents the results on clustering coefficient using synthetic datasets. As we recommend nodes, $SN-TA$ always improves the clustering coefficient. In contrast, LR decreases the clustering coefficient of the underlying graph. Although there is an improvement in the reachability results in LR, the social network structure strongly degrades.

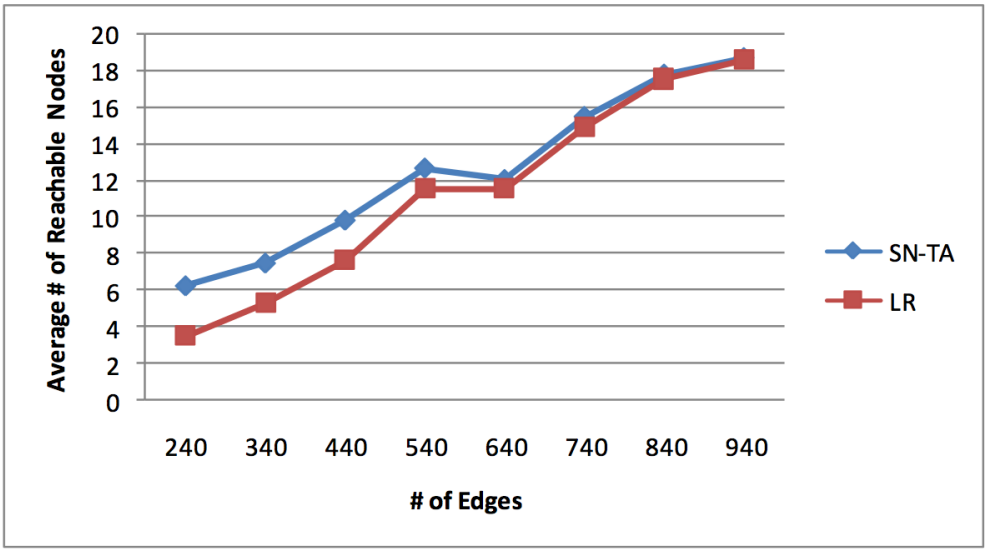

Figure 6.10: Average reachable nodes vs. increasing number of edges

Figure 6.10 illustrates the changes in the number of reachable nodes as we increase the number of edges when the number of nodes stays the same for only one recommendation. As the graph gets denser the $SN-TA$ algorithm and LR converge to a point. The graph becomes so connected that recommending another node does not cause any increase in the average reachability of the graph. For any given number of nodes, $SN-TA$ and LR converge as the density increases. As most of the social graphs in real life have a high number of nodes and a high density, we also show how increasing the number of nodes affects the convergence time of $SN-TA$ and LR in terms of the number of recommendations

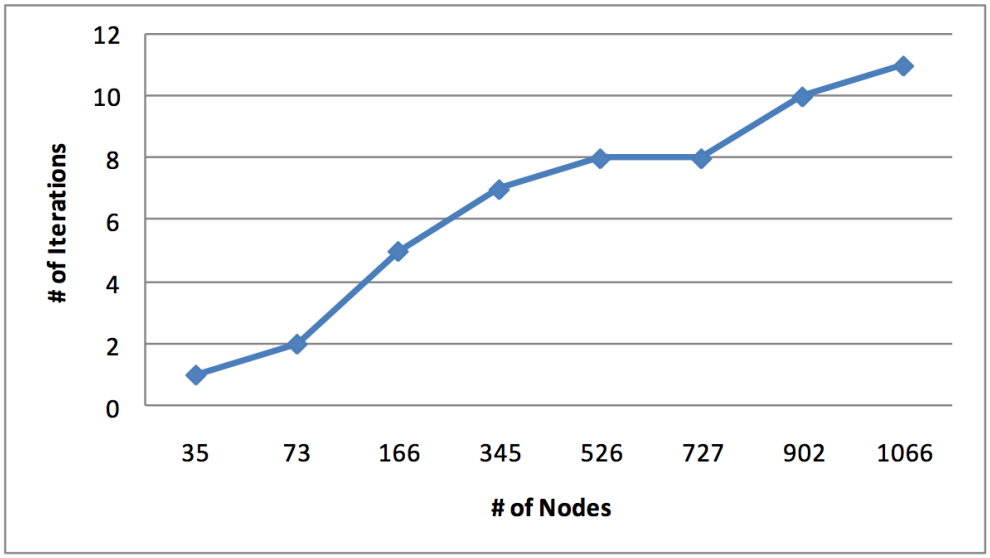

Figure 6.11: Number of iterations and increasing number of nodes

(Figure 6.11). As the number of nodes increases, the convergence of $SN - TA$ and LR gets much slower.

6.5 SN-TA vs SN-TA+

As we described before, $SN-TA+$ is a generalization of $SN-TA$ in which we recommend nodes within *k-hop* distance. We compared the performance of $SN-TA$ and $SN-TA+$ algorithms on different graphs we generated by increasing edge sizes. In Figure 6.12, we illustrate the results in terms of the weight-based accuracy we described above. In all cases, $SN-TA+$ produces better results compared to SN-TA. This is expected since SN-TA+ algorithm reaches more nodes than $SN-TA$. On the other hand, $SN-TA+$ involves more communication cost. Furthermore, as k gets bigger, the clustering coefficient gets smaller. So there is a trade-off between the value k and the clustering coefficient.

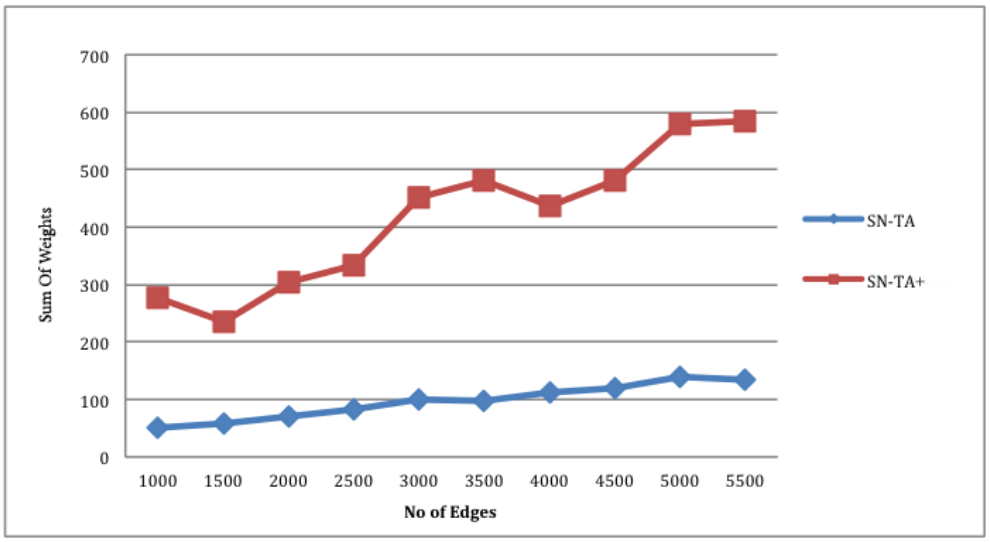

Figure 6.12: SN-TA vs. SN-TA+

6.6 Load Preserving Reachability Score vs. Reachability Score

Load Preserving Maximization Problem. The nodes can naturally have a skewed distribution in terms of links (neighbors) vs. capacity. This situation results in an imbalanced network where some nodes require more resources than

the others. The network would be significantly affected when those heavy loaded nodes are offline. If the recommendation focuses only on the reachability, it can cause overload of the nodes with high reachability scores. One needs to design a score that increases reachability while preserving the load of the network. In a balanced network, the nodes should have similar utilizations in terms number of links they have vs. their link capacity. Utilization u can be defined as link load l over capacity c. Overall utilization of the network can be defined as follows.

$$u_{avg} = \frac{\sum_{i=1}^{n} l_i}{\sum_{i=1}^{n} c_i} \tag{6.1}$$

Furthermore, we also define balance quality to determine how much balanced our P2P social network is.

$$b_{quality} = \frac{\sum_{i=1}^{n} |u_{avg} - u_i|}{u_{avg}} \tag{6.2}$$

A simple heuristic is to recommend nodes with high utilization to the nodes with low utilization. By using this simple heuristic, we define load-preserving maximization problem as follows.

$$\begin{aligned} \underset{t}{maximize} \quad & \frac{Re(t)A(t)^{(u_{avg}-u)\epsilon+1-\epsilon}}{Re(s,t)} \\ & subject\ to\ H(s,t) < \delta \end{aligned} \tag{6.3}$$

In above formula, u_{avg} is average utilization of the social network and ϵ is the constant that we use to adjust importance of reachability vs. load-preservation. We compare reachability scores recommendation vs. load preserving reachability scores recommendation on their performance for reachability. We again generated a small-world graph using power-law distribution for availabilities. As we can see from the Figure 6.13, load preserving reachability score is slightly worse than the reachability score although it is better than random recommendation. Therefore, we can infer that load preserving reachability score will be good enough to recommend nodes while we provide a balance factor.

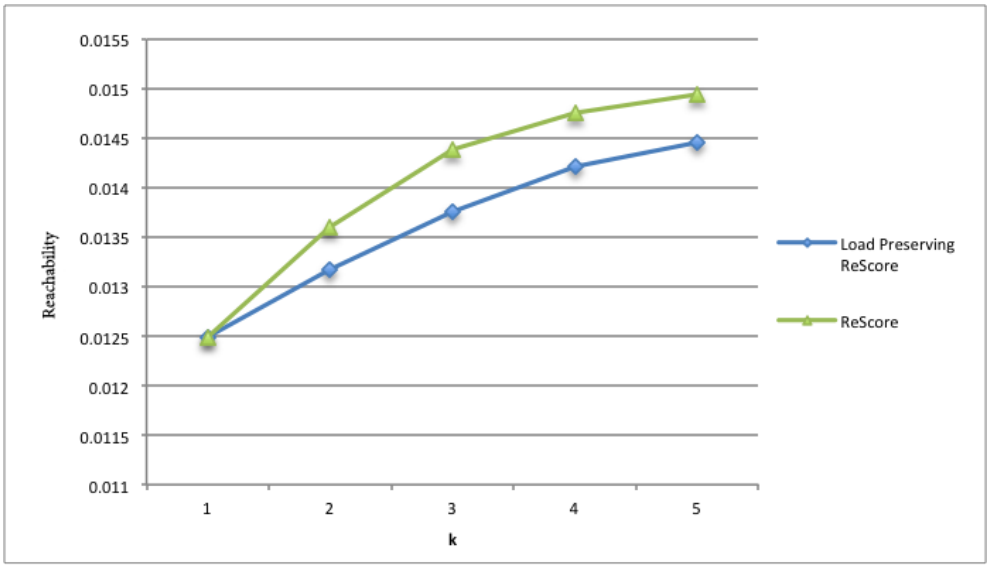

Figure 6.13: Reachability Score vs. Load Preserving Reachability Score

Chapter 7

Discussion

For an end-to-end P2P social network, there are several issues to overcome varying from encryption to maintaining user data. One can develop a P2P social network using different types of architectures. Naïve approach would be sharing the data randomly among the peers, which would not be appropriate in terms of service availability and data maintenance. Likewise, it would be difficult to recover from a failure or even to find out which nodes failed or went offline. Instead, a hierarchical architecture allowing the existence of super-peers would serve in handling such problems as described in the following.

7.1 Design Alternatives

A DNS like hierarchical architecture can be implemented for a super-peer based approach. Each super-peer can have a higher-level super-peer to which it is connected. At the highest level, there will be one or more super-peers, which would be available all the time. Any failing request would go through the top-level super-peers and would be routed through the appropriate super-peer. As the number of users increases, a need will arise for new super-peers which can be achieved by using super-peer selection algorithms partially based on their

availability. On a failure scenario or load problems for super-peers, a new super-peer can be selected from the peers and peers without a super-peer can be pointed to this new super-peer.

A key problem in P2P social networks is how to identify online peers and their properties. This problem can be handled by having lookup services at super-peers that will return the connection properties and status of the peers. If a requested lookup does not exist in a super-peer, it can route the request to a hierarchy of super-peers. Once the data is received from the super-peer, it can be returned to the peer itself. To have such lookup services, one needs identification for each peer existing in the P2P social networt. This can be solved with GUIDs [21], the globally unique ids that can be generated when a user creates an account. If a login request from a super-peer is valid, the connection properties and status of the peer can be updated. By doing so, friends can reach the latest connection properties and the status. After the login process, users can request lookup for their friends.

Data maintenance will be another problem for P2P social networks. Different from a traditional P2P system, people share data with their friends, not with everybody. One needs to distribute the data to the friends. Even if a peer were offline, parts of its data would be reachable from its friends. One can store the most recent data of a peer in its friends, and the old data in the peer itself because people have tendency to check out what is new. A secure transfer of data between peers is also needed using the encryption [32] methods such as using public key infrastructure which not only supports encryption both also authentication.

We are currently building a P2P social network application following a hybrid P2P infrastructure [33]. To provide peer addresses, we utilize super-peers that have a DNS like protocol in which each super-peer delegates address inquiry message to parent super-peer if peer address is not found in local repository. The super-peer has permanent addresses for system start-up and keeps track of the addresses.

7.2 Scoring for Link Recommendation

In the thesis, we developed a new node scoring method that can be used for robust development of P2P social networks. One can mark a node as important if the removal of the node degrades the reachability of the network significantly. This definition handles both the topological connectivity of the network, and the social centrality of the node. If the removal of the node causes a high decrease in the reachability of the network, then this node will have a high impact on the connectivity of the network. Also reachability of a path degrades as the distance between two nodes increases. If a node has a high centrality value, then a lot of shortest paths pass through the node. Thus the removal of the node causes a high decrease in the reachability of the network if the node has a high centrality. One can also come up with node measures by combining the traditional node scoring of social networks and P2P systems. One such alternative can be "trusted centrality" that combines P2P trust and graph centrality measures. Trust is a challenging factor in P2P systems since a node can appear and disappear instantly. Trust and reputation models are based on the values that are assigned between nodes such that node i assigns a trust value to node j, and vice versa. There is a significant set of trust models, including Cuboid [34] Trust, EigenTrust [35], BNBTM [36], GroupRep [37], etc. Another score can be available authority that combines availability in P2P systems, and the authority score from the network topology. The lifetime of a peer determines its availability. The simplest way of implementing availability is waiting up for a given time and marking the node as online or offline.

7.3 Conclusion

We presented a new problem and solutions of top-k link recommendation in P2P social networks. We followed exact and approximate versions of reachability based models on uncertain graphs. We developed a new node scoring using both the

reachability definition and locality of the nodes. Based on these, we proposed distributed top-k link recommendation algorithms. We used a Monte Carlo based sampling approach for exact reachability estimations and a computationally appropriate algorithm. Experimental results include the analysis of performance of the algorithms and the reachability score for link recommendation. The proposed node score improves the reachability more than a local random recommendation approach. It also increases the clustering coefficient of the graph, while the random recommendation degrades clustering coefficient. Our approximations are almost accurate as their exact counterparts and have much less computational cost.

Bibliography

[1] S. Buchegger and A. Datta, "A Case for P2P Infrastructure for Social Networks - Opportunities and Challenges," *Wireless on-demand Network Systems and Services*, vol. 28, pp. 161–168, 2009.

[2] P. E. Agre, "P2P and the Promise of Internet Equality," *Communications of The ACM*, vol. 46, no. 2, pp. 39–42, 2003.

[3] C. C. Aggarwal, *Social Network Data Analytics*. Springer Publishing Company, 2011.

[4] D. Liben-Nowell and J. Kleinberg, "The Link Prediction Problem for Social Networks," in *International Conference on Information and Knowledge Management*, pp. 1019–1031, 2003.

[5] L. A. Adamic and E. Adar, *Social Network Data Analytics*. Springer Publishing Company, 2011.

[6] L. Backstrom and J. Leskovec, "Supervised Random Walks: Predicting and Recommending Links in Social Networks," in *Acm International Conference On Web Search and Data Mining*, pp. 635–644, 2011.

[7] L. A. Adamic and E. Adar, "Friends and Neighbors on the web," *Social Networks*, vol. 25, pp. 211 – 230, 2001.

[8] P. Sarkar, D. Chakrabarti, and A. W. Moore, "Theoretical Justification of Popular Link Prediction Heuristics," in *In International Conference On Learning Theory*, pp. 295 – 307, 2011.

[9] B. Greschbach, G. Kreitz, and S. Buchegger, "New Privacy Challenges in Decentralized Online Social Networks," in *International Workshop on Security and Social Networking*, pp. 333 – 339, 2012.

[10] O. Bodriagov and S. Buchegger, "Encryption for Peer-to-Peer Social Networks," in *IEEE International Conference on Social Computing*, pp. 1302 – 1309, 2011.

[11] R. Sharma, A. Datta, M. Dell'Amico, and P. Michiardi, "An Empirical Study of Availability in Friend-to-Friend Storage Systems," in *IEEE International Conference on Peer-to-Peer Computing*, pp. 348 – 351, 2011.

[12] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A Survey of Top-k Query Processing Techniques in Relational Database Systems," in *ACM Computing Surveys*, pp. 1 – 58, 2008.

[13] Y.Zhang, G.Shen, and Y.Yu, "LiPS : Efficient P2P Search Scheme with Novel Link Prediction Techniques," in *IEEE International Conference on Communications*, pp. 24–28, 2007.

[14] A.-M. Kermarrec, V. Leroy, and G. Tredan, "Encryption for Peer-to-peer Social Networks," in *ACM International Conference on Information and Knowledge Management*, pp. 1209–1214, 2011.

[15] R. Fagin, "Combining Fuzzy Information: an Overview," *SIGMOD Record*, vol. 31, pp. 109–118, 2002.

[16] R. Fagin, A. L. Y, and M. N. Z, "Optimal Aggregation Algorithms for Middleware," in *ACM Symposium on Principles of Database Systems*, pp. 102 – 113, 2001.

[17] Z. Guan, G. Yan, and H. Huang, "A Novel Top-k Query Scheme in Unstructured P2P Networks," in *IEEE International Conference on Computer and Information Technology*, pp. 16 – 21, 2009.

[18] M. Theobald, G. Weikum, and R. Schenkel, "Top-k Query Evaluation with Probabilistic Guarantees," in *VLDB*, pp. 648 – 659, 2004.

[19] J. X. Yu and F. Cheng., "Graph Reachability Queries: A Survey," in *Managing and Mining Graph Data Advances in Database Systems*, pp. 181–215, 2010.

[20] K. Zhu, W. Zhang, G. Zhu, Y. Zhang, and X. Lin, "BMC: An Efficient Method To Evaluate Probabilistic Reachability Queries," in *International Conference on Database Systems for Advanced Applications*, pp. 434 – 449, 2011.

[21] S. Buchegger, D. Schiooberg, and L. H. Vu, "PeerSoN: P2P Social Networking: Early Experiences and Insights," in *ACM EuroSys Workshop on Social Network Systems*, pp. 46 – 52, 2009.

[22] R. Sharma and A. Datta, "Supernova: Super-peers Based Architecture for Decentralized Online Social Networks," in *International Conference on Communication Systems and Networks*, pp. 1 – 10, 2012.

[23] G. Mega, A. Montresor, and G. P. Picco, "Efficient Dissemination in Decentralized Social Networks," in *IEEE International Conference on Peer-to-Peer Computing*, pp. 338 – 347, 2011.

[24] J. S. Provan and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Graph is Connected," *SIAM Journal on Computing*, pp. 777–788, 1983.

[25] P. Dagum, K. R, M. Luby, and S. Ross., "An Optimal Algorithm for Monte Carlo Estimation," *SIAM Journal on Computing*, vol. 12, pp. 1484– 1496, 2000.

[26] W. Chen, C. Wang, and Y. Wang, "Scalable Influence Maximization for Relevant Viral Marketing in Large-Scale Social Networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1030 – 1038, 2010.

[27] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Gnutella Peer-to-Peer Network." http://snap.stanford.edu/data/p2p-Gnutella04.html, 2002.

[28] J. K. J. Leskovec, D. Huttenlocher, "Wikipedia Vote Network." http://snap.stanford.edu/data/wiki-Vote.html, 2009.

[29] J. Yang and J. Leskovec, "Friendster Social Network and ground-truth communities." http://snap.stanford.edu/data/com-Friendster.html, 2012.

[30] D. J. Watts and S. H. Strogatz, "Collective Dynamics of 'Small- World' Networks," *Nature*, vol. 393, pp. 409 – 410, 1998.

[31] P. Holme and B. J. Kim, "Growing Scale-Free Networks with Tunable Clustering," *Disordered Systems and Neural Networks*, vol. 65, p. 02610, 2002.

[32] O. Bodriagov and S. Buchegger, "Encryption for Peer-to-Peer Social Networks," in *IEEE Security and Privacy in Social Networks*, pp. 47 – 65, 2011.

[33] Y. Aytas, "SOWHOO: A P2P Social Networking Application." https://github.com/yusufaytas/sowhoo, 2012.

[34] X. C. R. Chen, L. Tang, and Z. C. J. Hu, "Cuboidtrust: A Global Reputation-Based Trust Model in Peer-to-Peer Networks," in *Autonomic and Trusted Computing*, pp. 203 – 215, 2007.

[35] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks," in *International Conference on World Wide Web*, pp. 640–651, 2003.

[36] Y. Wang, V. Cahill, E. Gray, C. Harris, and L. Liao, "Bayesian Network Based Trust Management," in *Autonomic and Trusted Computing*, pp. 372 – 378, 2006.

[37] H. Tian, S. Zou, W. Wang, and S. Cheng, "A Group Based Reputation System for P2P Networks," in *Autonomic and Trusted Computing*, pp. 342 – 351, 2006.

Appendix A

Reachability Theorems

Theorem 1. Given a connected graph *G(V,E)* where $u, v \in V$, then the reachability *Re(u,v)* is a semi-metric on V.

Proof. To show that *Re(u,v)* is a semi-metric on V, we will show that the above properties are satisfied.

i. Using the first axiom of probability theory, any event will have a non-negative probability. Thus the reachability definition will ensure that $Re(u, v) \geq 0$ for all nodes *u,v*.

ii. If *Re(u,v)=0*, then using reachability definition we can equivalently say that all the paths between u and v are unavailable. As we showed, for a path to be unavailable, at least one node should have a zero availability score, which is a contradiction with our non-zero property of availability.

iii. Let $L_{u\to v}$ be a path from u to v, and $L_{v\to u}$ be the backwards counterpart from v to u then

$$\begin{aligned} Re(u, v) &= P\left(\bigvee_{L_{u\to v}\in P_{u\to v}} R(L_{u\to v}) = 1 \right) \\ &= P\left(\bigvee_{L_{v\to u}\in P_{v\to u}} R(L_{v\to u}) = 1 \right) \end{aligned} \tag{A.1}$$

as $|P_{u\to v}| = |P_{v\to u}|$, and all the paths in $P_{u\to v}$ have their backward paths in $P_{v\to u}$, with $R(L_{u\to v}) = 1 \Leftrightarrow R(L_{v\to u}) = 1$).

This concludes our proof on the semi-metric of reachability on V.
In social networks it is quite frequently the case that there are bridges between different communities that connects them. We show that if the communication between two nodes can only be achieved over another middle node, then the reachability solely depends on the reachability between the end nodes and the middle node, and the availability of the middle node.

Theorem 2. Given a connected graph *G(V,E)* where $u, v, y \in V$, and the reachability *Re(u,y)* on *VxV*, if all the paths from u to y pass through the node v then

$$Re(u, y) = \frac{R(u, v)R(v, y)}{P(N_v = 1)} \tag{A.2}$$

Proof. Let $P_{u\to y}$ be all the paths from u to y. All the paths $L = (u, ..., v, ..., y) \in P_{u\to y}$ can be identified as a path $L_{u\to v} = (u, L_1, ..., v)$ from u to v and a path $L_{v\to y} = (v, L_{w+1}, ..., y)$ from v to y. Then

$$\begin{aligned} R(L) = 1 &\Leftrightarrow N_u = 1 \wedge ... \wedge N_v = 1 \wedge N_{L_{w+1}} \wedge ... \wedge N_y \\ &\Leftrightarrow N_u = 1 \wedge ... \wedge N_v = 1) \wedge N_v \wedge N_{L_{w+1}}) \wedge ... \wedge N_y) \\ &\Leftrightarrow R(L_{u\to v}) = 1 R(L_{v\to y}) = 1 \end{aligned} \tag{A.3}$$

Actually as $P_{u\to y}$ is the set of all the paths from u to y, then $P_{u\to y}$ contains all the paths in the Cartesian product of the paths from u to v and from v to y combined in order. For any path $L_{u\to v}$, every path from v to y can be used to reach to y. Put another way, for u to reach y, it has to first reach v and then

from v to y. Thus

$$\begin{aligned} Re(u,y) &= P\left(\bigvee_{L_{u\to y}\in P_{u\to y}} R(L_{u\to y})=1\right) \\ &= P\left(\left(\bigvee_{L_{u\to v}\in P_{u\to v}} R(L_{u\to v})=1\right)\wedge\left(\bigvee_{L_{v\to y}\in P_{v\to y}} R(L_{v\to y})=1\right)\right) \\ &= P\left(\left(\bigvee_{L_{u\to v}\in P_{u\to v}} R(L_{u\to v})=1\right)\wedge\left(\bigvee_{L_{v\to y}\in P_{v\to y}} R(L_{v\to y})=1\right)\middle| N_v\right)P(N_v=1) \end{aligned} \tag{A.4}$$

Given v, the reachability from u to v and from v to y are independent, thus we have

$$\begin{aligned} Re(u,y) &= \frac{R(u,v)R(v,y)}{P(N_v=1)P(N_v=1)}P(N_v=1) \\ Re(u,y) &= \frac{R(u,v)R(v,y)}{P(N_v=1)} \end{aligned} \tag{A.5}$$

This theorem indicates that, without loosening the community structures in a social network, the reachability between two communities can be increased by increasing the reachability between the nodes and the bridges.

Lemma 1. Given a connected graph *G(V,E)* and subgraphs $G(V_1,E_1), G(V_2,E_2)$ $,...,G(V_N,E_N)$ of the graph *G(V,E)* where $V_i \cap V_{i+1} = v_i$, and $V_i \cap V_j = \emptyset$ for $i \neq j$, then for $u = v_0 \in V_1, y = v_N \in V_N$,

$$Re(u,y) = \frac{\prod_{i=0}^{N-1} Re(v_i,v_{i+1})}{\prod_{i=0}^{N-1} P(N_{v_{i+1}})} \tag{A.6}$$

Proof (Proof by Induction): The reachability between u and y over v_i,

$$Re(u,y) = \frac{R(u,v_i)R(v_i,y)}{P(N_{v_i}=1)} \tag{A.7}$$

the reachability between u and v_i over $v_{i-j}, 0<j<i$,

$$Re(u,v_i) = \frac{R(u,v_{i-j})R(v_{i-j},y)}{P(N_{v_{i-j}}=1)} \tag{A.8}$$

and the reachability between v_i and y over $v_{i+k}, 0<k<N-i$,

$$Re(v_i,y) = \frac{R(v_i,v_{i+k})R(v_{i+k},y)}{P(N_{v_{i+k}}=1)} \tag{A.9}$$

Thus the reachability between u and y over v_{i-j}, v_i, v_{i+k},

$$Re(u,y) = \frac{Re(u,v_{i=j})Re(v_{i-j},v_i)Re(v_i,v_{i+k})Re(v_{i+k},y)}{P(N_{v_{i-j}}=1)P(N_{v_i}=1)P(N_{v_{i+k}}=1)} \tag{A.10}$$

As a result, we conclude that

$$Re(u,y) = \frac{\prod_{i=0}^{N-1} Re(v_i,v_{i+1})}{\prod_{i=0}^{N-1} P(N_{v_{i+1}})} \tag{A.11}$$

Hence, in a social network environment, if the communities are connected through a chain, the reachability of two nodes from different communities are dependent on the reachabilities between the bridges of communities and these two nodes.

Appendix B

SOWHOO : A P2P Social Network Application

SOWHOO is a peer to peer(P2P) social networking application. SOWHOO is built upon P2P infrastructure where each computer in the network can act as a client or server for the other computers in the network. As a social networking application, each computer in the network can exchange messages between neighbors.

Initial Design.

SOWHOO is a hybrid P2P infrastructure, where there are simple peers and super-peers. Each peer in the system has a super-peer to provide other peer addresses such as neighbor peers. In order to provide peer addresses, super-peers are designed to have a DNS like protocol in which each super-peer delegates address inquiry message to parent super-peer if peer-address is not found in local repository. As a result, there is at least one super-peer, which has permanent address for system start-up. This super-peer keeps track of super-addresses and if there is no other super-peer, it would also keep track of simple-peers.

We consider a partitioning algorithm for distribution of peers to super-peers. Each super-peer would have a self-balancing mechanism to hold similar number of addresses to have more uniform network. This would prevent the system from depending so much in particular super-peers, which may result in overload for

that particular super-peer.

Portability.

SOWHOO is a multi-platform application, which can run on different devices. In order to support portability, we have chosen Java as programming language. However, all devices must have JRE to run SOWHOO. Although main application will be same for all of the devices that SOWHOO will run on, user interfaces might change due to the different display features of the target platform. In Figure B1, devices that SOWHOO can run are shown.

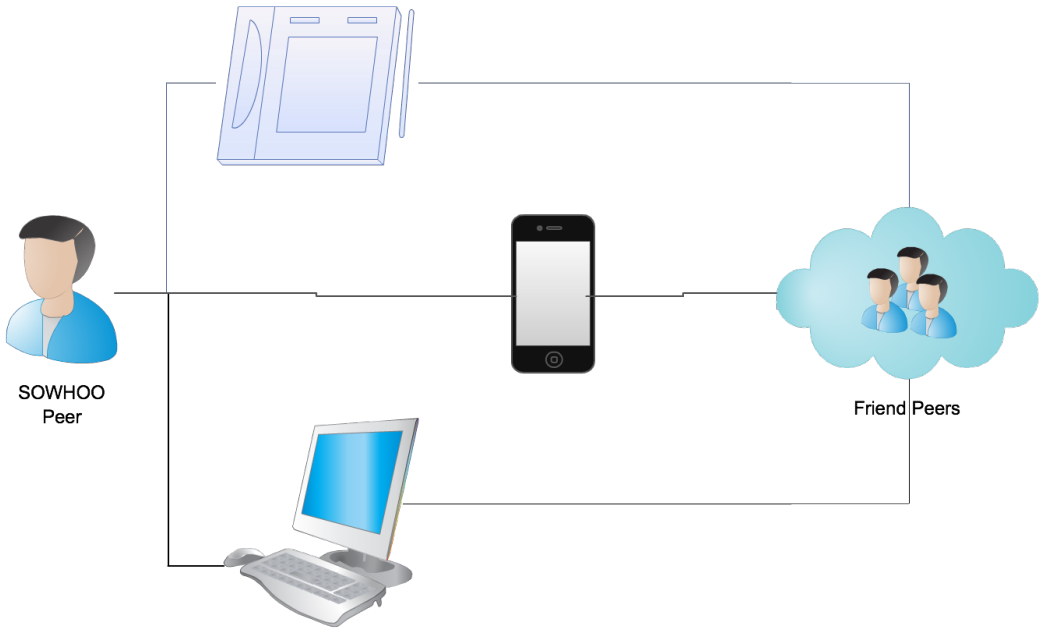

Figure B.1: Platforms that SOWHOO can run

Data Storage.

SOWHOO keeps user information and messages in the local devices and sends those messages to the friends devices. If user logins to the application from another device, SOWHOO would retrieve user messages from its friends and store those messages on the new device. In an extreme case, where no friend is online, it would not retrieve the data. However, we assume that one or more friends will be online since mobile device usage is extremely popular.

Furthermore, friends can see messages of their friends by requesting directly to them if they are online; otherwise, they will request messages from the common friends. Once messages are retrieved, they will be stored in the device. Our

current implementation does not support storing of messages partially which is better in terms of efficiency since keeping all of the friends messages would be redundant.

We also provide caching of the messages for the friends because people want to see the new messages instead of old messages. If a friend requests messages, it will be first retrieved from the cache. If it does not exist in the cache, SOWHOO will retrieve the messages from the persistent storage.

Architecture.

SOWHOO has mainly four layers. The first layer is the user interface, which interacts with the user. The second layer is the application logic layer, which gets the user requests and returns the corresponding responses. The third layer is the dispatcher, which sends and receives updates for the user. The last layer is network layer, which provides load balancing for super-peers, peer suggestions and score calculation. In Figure B.2, architecture of SOWHOO can be seen.

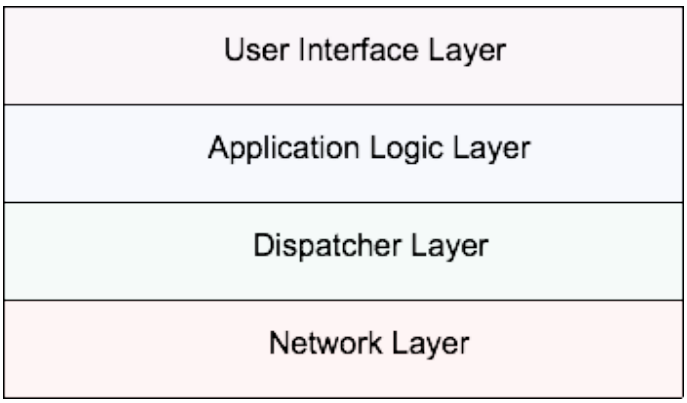

Figure B.2: Architecture of SOWHOO

Packaging.

SOWHOO has basically three packages, which are "common", "peer" and "super-peer"(Figure B.3). As its name implies "common" is used by both peer package and super-peer package. This package provides message types, messages and serialization. Using this "common" package, peer package gains the ability to send and retrieve messages between peers. These messages are in general text messages for the sake of simplicity. Moreover, each peer has a "super-peer" package; however, it is not used until a peer becomes a super-peer. After becoming a super-peer, this package handles requests coming from the peers.

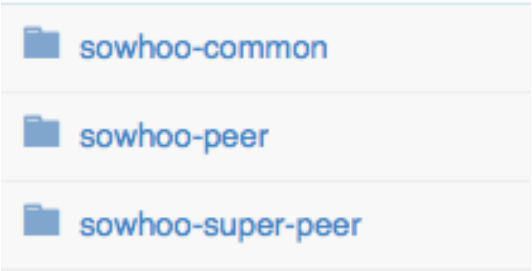

Figure B.3: Packaging of SOWHOO

Messaging Structure.

SOWHOO can support any type of messages by default. Nevertheless, we did not implement complex types of messages. We instead implemented text messaging which is shared between the peers. In Figure B.4, one can see initial messaging structure of SOWHOO. Each message has a header which keeps track of the information about message delivery details and a body which usually contains the data associated with the message. Moreover, message body can contain related attachments like photos, links, files, etc.

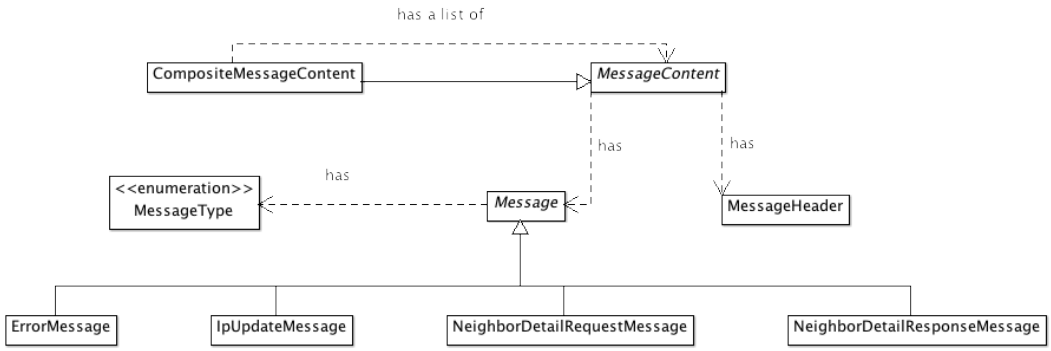

Figure B.4: Messaging Structure of SOWHOO

Screen Shots.

We developed an android user interface for SOWHOO. In Figure B.5, login screen of the SOWHOO is presented. In Figure B.6, message screen of SOWHOO is presented. Lastly, main screen of SOWHOO is provided in Figure B.7.

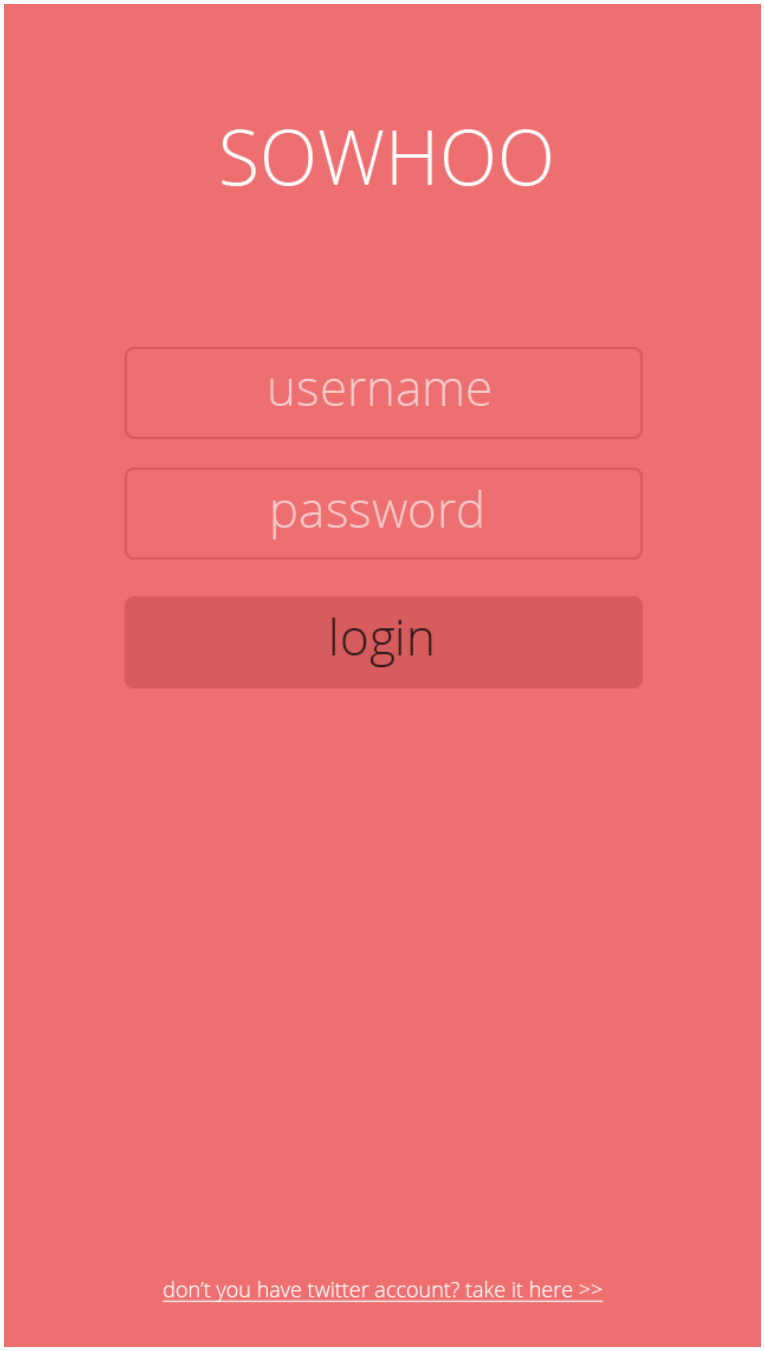

Figure B.5: Login Screen of SOWHOO

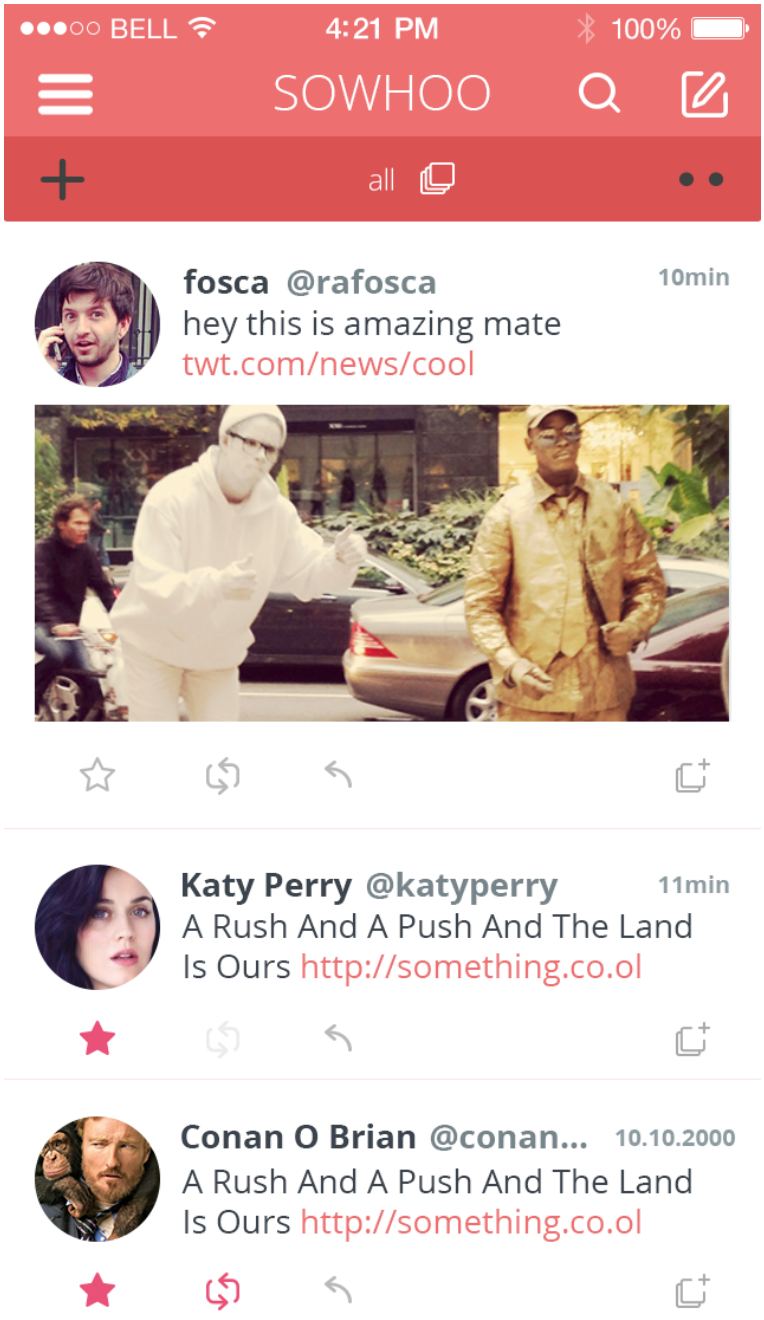

Figure B.6: Messages Screen of SOWHOO

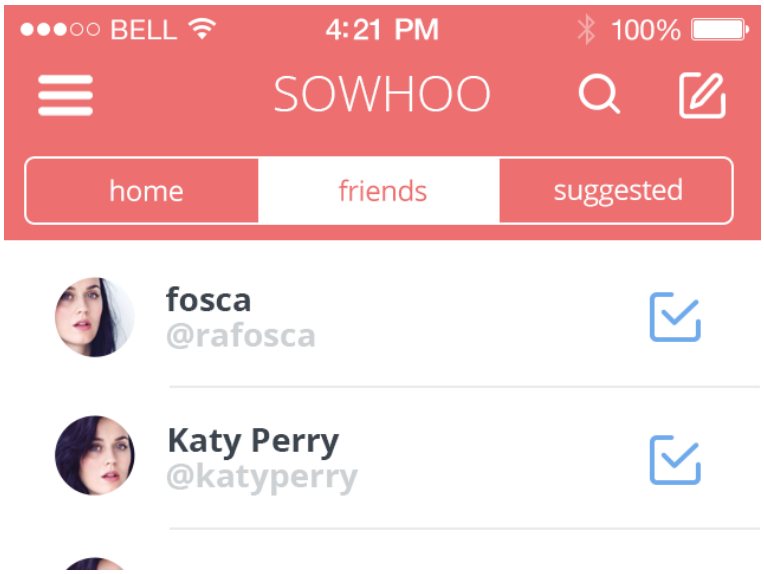

Figure B.7: Main Screen of SOWHOO