Information Flow and Search in Unstructured Keyword based Social Networks*

Ankush Garg   Prantik Bhattacharyya   Charles U. Martel   S. Felix Wu
Department of Computer Science
University of California, Davis
{garg, pbhattacharyya, cumartel, sfwu}@ucdavis.edu

Abstract—In online social networks (OSNs), user connections can be represented as a network. The network formed has distinct properties that distinguish it from other network topologies. In this work, we consider an unstructured keyword based social network topology where each edge has a trust value associated with it to represent the mutual relationship between the corresponding nodes. Users have keywords as their profile attributes that have policies associated with them to define abstractly the flow of keyword information and the accessibility to other users in the network. We also address privacy concerns as outlined in works on future OSN architectures.

This paper makes two key contributions. First, we develop an information flow model to disseminate keyword information when users add keywords as their profile attributes. Second, for keyword based queries, we design and develop a search algorithm to find users with the specified keywords in their profile attributes. It is based on a linear combination of topological distance and trust metrics. It is also dynamic in nature such that it adapts itself for each individual node during the search process. We observe an improvement in orders of magnitude when the search algorithm is compared to breadth first search.

I. INTRODUCTION

In online social networks (OSNs), user connections can be represented as a network. The network formed has distinct properties that distinguish it from other network topologies. E.g., it has high average node degree, high value of clustering and displays small world properties [1], [2]. For accurate results, these properties need to be considered during analysis of OSNs. For instance, when a user searches for other users (or user characteristics), the network's search scheme may have to search and sort through a large set of results even at small topological distances. Additionally, users consider search results relevant based on their position in the network rather than globally relevant results.

In this work, we consider an unstructured keyword based social network topology where a link between two nodes represents friendship between them. Each edge has a trust value associated with it to represent the mutual relationship between the corresponding nodes. Users have keywords (profile attributes) as their characteristics which have policies associated with them to define abstractly the flow of keyword information and the accessibility to other users in the network. Such keyword based social networks, which try to mimic human behavior, could potentially be used for internet routing [3], [4]. For instance, assume that Bob has an interest in

soccer. He would like to establish a contact with someone who is also interested in soccer. His natural tendency would be to check if any of his direct friends plays soccer. Even amongst them, he would prefer those friends with whom he shares a closer (more trustworthy) relationship. Then, if he wants to find more soccer playing people, he would go to his friends of friends, etc. Keyword based social networks try to incorporate this real life human social behavior in OSNs through the use of keywords by allowing communication only through the available trustworthy social paths.

This paper makes two key contributions. First, we develop an information flow model to disseminate keyword information when users add keywords as their profile attributes. To address privacy concerns as outlined in works on future OSN architectures [5], [3], [4], [6], we restrict the identity of users during information flow to only their direct friends. Beyond that, even though information about keywords is allowed to flow, the identity of users is not propagated. Second, we design and develop a search algorithm based on keyword information. The search problem is broadly defined as the scenario when a user queries with a set of keywords to contact other users (termed as *targets*) who have all those keywords as their profile attributes. In real life, as with Bob, people first seek information from their friends, then friends of friends and so on. Also, people inherently grade their friends because of the mutual relationship. Keeping this in mind, we aim at finding a subset of these targets which are topologically closer to the querying node and have high trust on the edges connecting them to the querying node. We believe that our contributions can be applied to other network applications such as peer to peer networks or mobile social networks that may show graph properties similar to an unstructured social network.

When knowledge about users and their keyword information is available to each individual node, aided by a centralized system (like Facebook, Orkut, etc.), then the search problem reduces to sorting through a list of targets who match the set of keywords in the query and construct the good result set. Here, we broaden the scope to consider future OSN architectures [5], [3] with a decentralized architecture where no node has access to complete information about all users. For these decentralized settings, searching for targets becomes a challenging problem. Further, the unstructured nature of the network increases the difficulty. Simplest would be to broadcast the network with queries till all possible targets are

found, but this is inefficient and unscalable. Other decentralized search techniques based on either breadth first search or random walks are also not good candidates as they don't utilize the level of information available, in the form of user keywords and their policies, in a keyword based social network. The search algorithm we present utilizes the information available in the keyword based social network as it looks for targets. It uses a linear combination of two primary metrics: a distance metric to find topologically closer targets and, a trust metric to find paths with high trust values between the querying node and the target. It is also dynamic in nature such that it adapts itself for each individual node during the search process.

We describe the unstructured keyword based social network model in section II. Section III presents the proposed model for information flow in the social network. Here, we also discuss how to maintain recent information and address the privacy concerns of users. Section IV describes the search algorithm and how it adapts itself for different nodes. Section V deals with the setup used for evaluation and results are presented in section VI. Section VII covers the related work in this field. Finally, we conclude in section VIII with possible extensions.

II. BACKGROUND

In this paper, we focus on unstructured keyword based social networks where the user topology is represented by a set of nodes and edges. Let the social network be an undirected graph $G = (V, E)$, where V represents the set of nodes in the social network and E is the set of edges between the vertices in V. A link exists between nodes i and j if both the nodes want to be friends with each other. Each edge has a trust value associated with it to represent the mutual relationship between the corresponding nodes. The concept of trust can be generalized to include different models, that characterize different friendship levels [7], such as the frequency or quality of interactions, privacy settings during information sharing, etc. The value of trust lies between 0.0 and 1.0 with a higher value representing more trust.

In a keyword based social network, users are characterized by keywords [4], [8]. A node v declares its *Profile Attributes* (K_v^{PAtt}) which is a set of keywords and each keyword has a policy associated with it [3]. Each node also has another set of attributes called the *Friendly Attributes*, K_v^{FAtt}, that is the set of keywords that v receives from its friends. The Profile Attributes represent the information that the user wants other users to know about itself whereas Friendly Attributes is the information that the user receives from other users. The policy definition in [3] uses a broader definition based on subjective parameters that is beyond the scope of this work. Here, we consider the policy for a (profile attribute) k of v as a tuple:

$$\forall k \in K_v^{PAtt}, \ \exists Policy(k) = [D, T]$$

where, D is the maximum distance (in hops) of nodes from whom v is willing to get contacted with k as the keyword and, T is the minimum trust that each link must have from the query-issuer to v on the social path.
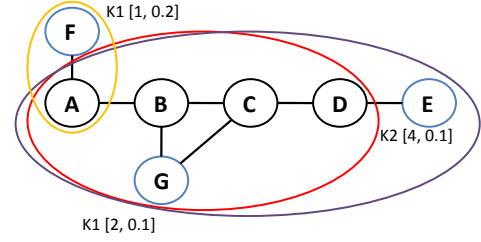

Fig. 1: Example of the keyword based social network

Figure 1 shows an example graph for the given framework. Nodes F and G, each sets keyword $K1$ and E sets $K2$ as their profile attribute with the policies shown. For simplicity, assume that all links have a trust value higher than 0.5. The ovals show the nodes to which the keywords will be propagated depending on their policy during the initial information flow phase. A user u can issue a query with keywords Q_k from K_u^{FAtt} which will enable him to establish contact with other users who have all the keywords in Q_k as their profile attributes. The set of potential targets for a querying node u is defined as: $Targets_{Potential} = \{v : v \in V$ where $\forall k \in Q_k, \ k \in K_v^{PAtt}$ && u satisfies $Policy(k)$ set by $v\}$. E.g., when node A issues a query $Q_k = \{K1\}$, $Targets_{Potential} = \{F, G\}$ and if node B issues the same query then $Targets_{Potential} = \{G\}$. Returning to Bob's example from the introduction, he can issue a query with $Q_k = \{soccer\}$ assuming $soccer \in K_{Bob}^{FAtt}$. Targets would be all users who have 'soccer' as their profile attribute such that Bob satisfies the policies set by them. Bob can reach the targets through those of his direct friends who have 'soccer' as their friendly attribute implying that there exists a social path through them. Bob can also issue more specific queries. E.g., $Q_k = \{soccer, Davis \ CA\}$ if he wants to find soccer playing users in the city of Davis in California.

To find all the potential targets a search algorithm may have to inspect a huge number of nodes, due to the high node degree and clustering coefficient, making the search process impracticable. Thus, it would suffice to find good targets from $Targets_{Potential}$ where a target is defined as good when the path connecting it to the querying node is small in length and has high trust on the edges. The definition of a good target has been kept at an abstract level since its tough to quantify goodness owing to its subjective nature from a querying node's point of view. Results and targets describe the same concept and have been used interchangeably in rest of the paper. In the next sections, we develop an information flow model and a search algorithm to find this subset of targets.

III. INFORMATION FLOW MODEL

Once a user joins the network and adds keywords as his Profile Attribute(s), information needs to be propagated in the network so that other users can search and contact him. The information flow model primarily needs to satisfy three conditions while spreading the profile attributes. First, it must propagate the information properly using minimal resources. Second, it must address the privacy concerns of users [6]. Finally, it must ensure that nodes maintain the most recent

information i.e. information about changes in friendship(s) or keyword policies must be propagated to nodes quickly.

A. Keyword Propagation Process

When a user v adds a keyword K in its Profile Attributes with policy $P(K) = [D, T]$, the keyword has to be propagated to other nodes. Other nodes receive this keyword and store it as a Friendly Attribute. Information is spread out in the form of propagation messages where the structure of a propagation message is as follows: $\langle PID, Keyword, Hops_{remaining}, Hops_{covered}, Trust\rangle$, where PID represents the propagation identity, $Keyword$ is the keyword that is being propagated, $Hops_{remaining}$ is the number of hops that still need to be covered, $Hops_{covered}$ is the number of hops that have been covered so far and $Trust$ is the trust that was set in the policy by the node who added the keyword. PID is a randomly generated large binary string and we assume that the number of bits in PID are sufficiently large so that the probability of two different nodes generating the same PID for the same keyword is negligible. The motivation to use PID is to avoid cycles in the graph which could be numerous owing to the high clustering coefficient. PID makes sure that a propagation message does not go around a cycle from both directions and thereby avoids unnecessary messages.

Keyword	Propagation Data	Friends
K_1	$\{PID_1, Hr_1, Hc_1, T_1\}$	F_1, F_2, F_3
	$\{PID_2, Hr_2, Hc_2, T_2\}$	F_2, F_4
K_2	$\{PID_6, Hr_6, Hc_6, T_6\}$	F_1, F_3, F_4

TABLE I: Keyword Forwarding Table

Each node $w \in V$ maintains a Keyword Forwarding Table, FT_w, to store which keywords (along with their propagation data: Hr represents $Hops_{remaining}$, Hc represents $Hops_{covered}$ and T is the $Trust$) it has propagated to which of its direct friends so far. Table I shows an example of the keyword forwarding table. It shows that keyword K_1 was propagated to four direct friends by w and it was propagated to F_2 with two different policies (see below for explanation).

When a node gets the same keyword to propagate with different propagation data, it needs to decide if it should forward (and to which of its friends) the keyword with the newer policy because if the new data has already been covered (or included) in earlier propagation data then sending the new propagation message is unnecessary. We say that propagation data 1 is non-inclusive of propagation data 2 if

$$((\Delta Hr > 0) \ || \ (\Delta T < 0) \ || \ (\Delta Hc < 0))$$

where, $\Delta x = x_2 - x_1$. If the new propagation data says that more hops are remaining or the new trust is lower, then the keyword needs to be propagated again. If the new message has covered fewer hops then the keyword also needs to be propagated to store correct search information (section IV-A).

Consider the scenario when u receives a propagation message from v. It will process the message using Algorithm 1. The main step is to see that if the new propagation data has

---

Algorithm 1: Processing Propagation Message

Input: v sends prop. message $\langle pid, K, hr, hc, T\rangle$ to u

1 **if** $((K \in FT_u) \ \&\& \ (pid \in FT_u))$ **then**
2   **if** *new prop. data non-inclusive w.r.t stored prop. data corresponding to pid* **then**
3   update $\{pid, hr, hc, T\}$ in FT_u;
4   update search information;
5   **foreach** *(friend z ($\neq v$) of u)* **do**
6   **if** $((hr > 1) \ \&\& \ (Trust_{uz} > T))$ **then**
7   send $\langle pid, K, hr - 1, hc + 1, T\rangle$ to z;
8   add/update z (for pid) in FT_u;
9   **else** drop message;
10 **else if** $((K \in FT_u) \ \&\& \ (pid \notin FT_u))$ **then**
11   **foreach** *(friend z ($\neq v$) of u)* **do**
12   **if** *new prop. data non-inclusive w.r.t all stored prop. data corresponding to z* **then**
13   add $\{pid, hr, hc, T\}$ and z to FT_u;
14   update search information;
15   **if** $((hr > 1) \ \&\& \ (Trust_{uz} > T))$ **then**
16   send $\langle pid, K, hr - 1, hc + 1, T\rangle$ to z;
17   **else** drop message;
18 **else**
19   add K and $\{pid, hr, hc, T\}$ to FT_u;
20   update search information;
21   **foreach** *(friend z ($\neq v$) of u)* **do**
22   **if** $((hr > 1) \&\& (Trust_{uz} > T))$ **then**
23   send $\langle pid, K, hr - 1, hc + 1, T\rangle$ to z;
24   add z (for pid) in FT_u;

---

already been included for each friend or not and then forward the message accordingly. The search information that needs to be updated (steps 4, 14 and 20) will be discussed later (section IV-A). To initiate the propagation of keyword K with policy $P(K) (= [D, T])$, a node generates the propagation message $\langle pid, K, D, 0, T\rangle$ and processes it according to Algorithm 1.

B. Identity Privacy Issues

To respect privacy requirements, any social network model and corresponding applications must be designed so that user privacy is protected as information flows within the graph. Our design of message propagation removes the identity of users as their information flows deeper in the graph, i.e. at distances further away from their direct friends. The keywords, propagation data and PID's are stored but they are insufficient to reveal any relevant information about the people who are not direct friends of the user. As the identity of only the direct friends is stored, the network has the capability of supporting both anonymous and identified messages (depending on the application) without compromising the privacy of users making the message propagation model more general and suitable for wider use.

C. Information Maintenance

As friendships change, the topological structure of the graph changes. Policies of individual keywords may also be

changed by users. Such updates should also be reflected in the information that is stored by users. Thus, the information flow process must notify nodes about such updates so that each user has access to the latest information. To account for such scenarios, we introduce the concept of timely updates for propagated keywords that have flowed into the network. Each node will send beacons (which could be the ($keyword$, PID) pair) after a certain δt time to tell other nodes that the corresponding keyword still exists in the network. Thus, nodes need to store only that data which was received within the last δt time. If a node wants to change the policy it can generate a new PID and propagate the keyword again. The data stored for the earlier PID will be removed by other nodes as they will not get the beacon anymore. Nodes can use the forwarding table to forward beacons to other nodes. This process not only takes care of updated link topology but also makes sure that nodes have the most recent information about friendly attributes and their policies. The numerical value of δt will depend on the underlying physical network and the application and hence, is beyond the scope of this work.

IV. SEARCH ALGORITHM

In this section, we design and develop an algorithm to search in the keyword based social network. The search algorithm is composed of three key components. First, it uses a linear combination of a distance metric and a trust metric to define the 'value' of a direct friend. Second, a threshold function that helps in dynamically pruning the network. The last component is the query message processing algorithm.

A. Selecting Topologically Closer Nodes

In real life, people seek information first from their friends, then friends of friends and so on. We model this scenario by giving precedence to targets that are topologically closer to the querying node. In order to decide which of the direct friends of a node will be suitable to route queries in terms of distance, the node stores information about the distance traveled by received keywords to make an intelligent decision.

Keyword	Max. Hops	(Friend, Min. Hops)
K_1	$Hmax^{K_1}$	$(F_9, Hmin_{F_9}^{K_1})$, $(F_5, Hmin_{F_5}^{K_1})$
K_2	$Hmax^{K_2}$	$(F_2, Hmin_{F_2}^{K_2})$, $(F_6, Hmin_{F_6}^{K_2})$

TABLE II: Keyword Received Table

Each node w stores a Keyword Received Table, RT_w, in which entries are indexed by keywords. For each keyword k, we define $Hmax^k$ as the farthest distance covered by k as it traveled from nodes (who added k as a profile attribute) to w. This value is updated whenever a new propagation message for keyword k comes by comparing the stored $Hmax^k$ with the $Hops_{covered}$ field of the message and updating $Hmax^k$ with the greater value. Also, the node stores which of its direct friends forwarded the keyword to it. For each such friend, say F, the hops of the closest target ($Hmin_F^k$) reachable through F with keyword k is also stored. $Hmin_F^k$ can be updated during the propagation process (steps 4, 14 and 20 in algorithm

1); if the $Hops_{covered}$ for the propagation message from F with keyword k is less than $Hmin_F^k$ then the value is updated. E.g., table II shows that w received two keywords (K_1 and K_2) from its direct friends during the propagation process and it stores the corresponding $Hmax^{K_1}$ and $Hmax^{K_2}$ for each keyword. It also stores $Hmin_F^K$ values for each keyword for all direct friends who propagated that keyword. As with the Forwarding Table only the information from the previous δt time is stored in the Keyword Received Table.

To differentiate between direct friends of a node w, we define the distance value as follows:

Definition 1. *The distance value for a direct friend u of w who forwarded all the keywords in a keyword set S_k to w is:*

$$DV(u, S_k) = \frac{\min_{k \in S_k} Hmax^k - \max_{k \in S_k} Hmin_u^k}{\min_{k \in S_k} Hmax^k} \quad (1)$$

For a set of keywords the maximum of $Hmin$'s needs to be taken because no node before that distance exists which propagated all the keywords in S_k. Similarly, minimum of $Hmax$ is taken for a set of keywords as no node after that distance can exist which propagated all the keywords. A higher distance value for a friend tells that there is a better chance of finding a target close by through this particular friend. If paths of different lengths exist to the same target then the distance value will be higher for those direct friends through whom shorter paths exist. The distance value has been normalized w.r.t to the farthest target and it lies between 0 and 1.

B. Selecting Trustworthy Nodes

The next step towards finding a set of good results is to probe edges with high trust value between the connecting nodes. To achieve this we define the scaled trust function.

Definition 2. *The trust value for a keyword set S_k for a direct friend u (which forwarded all the keywords in S_k) of w is:*

$$TV(u, S_k) = \frac{T_{wu}}{T_{max}^{S_k}} \quad (2)$$

where T_{wu} is the trust of the (w, u) link and $T_{max}^{S_k}$ is the maximum trust amongst all friends of w from whom it received all the keywords in the set S_k.

The scaled trust defined above orders all those direct friends which forwarded all the keywords in S_k to w. This naturally helps w to find targets through those friends with whom he/she shares higher values of trust. We have used scaled trust to normalize the value. Next, using the two parameters defined so far, we discuss how a node can query the network for targets.

C. Querying with Keywords

Now, we describe the querying process that a user uses to search for targets. Nodes consider two important factors as they route query messages through the network: a) Value of Friends b) Threshold Function. The first parameter helps a node to determine the value of direct friends while the second parameter dynamically sets threshold values so as to reduce the number of edges the algorithm needs to inspect.

Definition 3. *The value for a keyword set S_k for a direct friend u (which forwarded all the keywords in S_k) of w is:*

$$V(u, S_k) = \rho \times DV(u, S_k) + (1 - \rho) \times TV(u, S_k) \quad (3)$$

where ρ lies between 0 and 1.

The parameter ρ decides the relative weight that is given to scaled distance value and to scaled trust value. Thus, if some application requires those paths which are very close to the querying node then it can set ρ close to 1 and if more precedence is given to trustworthy paths then ρ can be set close to 0. The structure of a query message is as follows: $\langle QID, Q_k, T_{min}, Hops_{done}, Hops_{left} \rangle$ where QID is a randomly generated binary string, Q_k is the set of keywords used for querying, T_{min} is the minimum trust on the path that has been traversed by the query message so far, $Hops_{done}$ is the number of hops that the query message has traversed and $Hops_{left}$ is the number of hops left for the message to travel. QID is the identity of the query and helps in distinguishing between different queries. Each node u also stores the QID's (and the corresponding direct friend who forwarded that QID) that it has processed in a table (QID_u) to avoid processing the same query multiple times.

Now, when a node processes a query with a particular keyword set, it needs to decide to which of its direct friends it should send the query to. We propose that the query should be sent to only those friends whose 'value' is above a threshold value (Θ). The threshold value is crucial to the behavior of the search strategy as it helps decide how aggressively the searched network is pruned. As the degree of nodes could vary drastically, a system wide constant value would prove insufficient to meet the requirements of the query algorithm. Hence, we model the threshold value to be a function of degree rather than a constant value. Consider a peer u on the social network and let us assume that $N_u^{Q_k}$ friends of u forwarded the keywords in Q_k to u. A static value of threshold may be significantly higher or lower than the 'value' of majority of friends in $N_u^{Q_k}$. This may lead either to drastic pruning or generation of excessive query messages, ultimately degenerating the search process. The threshold value should dynamically adjust itself according to the 'value' of the friends of u in $N_u^{Q_k}$. This will make the algorithm adaptive where each node calculates the threshold depending on its degree and the characteristics of its friends. Thus, we propose the threshold to be dependent on $N_u^{Q_k}$ for the node u.

$$\Theta(u, Q_k) = \max_{w \in N_u^{Q_k}} V(w, Q_k) - f(N_u^{Q_k}) \times$$
$$(\max_{w \in N_u^{Q_k}} V(w, Q_k) - \min_{w \in N_u^{Q_k}} V(w, Q_k)) \quad (4)$$

where the function f should be such that $f(1) = 1$, $\lim_{N_u^{Q_k} \to \infty} f(N_u^{Q_k}) = 0$ and it should be a monotonically decreasing function. This ensures that as $N_u^{Q_k}$ increases the threshold approaches the maximum 'value' reducing the number of friends to which the query needs to be sent. The function f decides the extent to which the search tree is pruned and

hence, we call it the *Pruning Function*. The whole class of functions $g(x) = x^{-p}$ for $p \geq 0$ satisfy the conditions of the pruning function. When $p = 0$, $f(N_u^{Q_k})$ will always set the threshold value to the least possible 'value' giving the breadth first search (BFS) mechanism. As p increases the threshold value reaches the maximum 'value' at a faster pace, pruning the search tree more drastically. Thus, by varying p we get many pruning functions with varying slopes.

Algorithm 2: Processing Query Message

Input: v sends query message $\langle qid, Q_k, T_m, H_d, H_l \rangle$ to u

1   **if** $(qid \in QID_u)$ **then** drop message;
2   **else**
3   add qid to QID_u;
4   **if** $((Q_k \subseteq K_u^{PAtt})$ && $(\forall k \in Q_k, [H_d + 1, T_m]$ *satisfies* $Policy(k)_u))$ **then** send success message to v;   /* u is a target */
5   **if** $(H_l > 0)$ **then**
6   **foreach** *(friend z ($\neq v$) of u such that z sent the keywords in Q_k)* **do**
7   **if** $(Trust_{uz} < T_m)$ **then** $T_m \leftarrow Trust_{uz}$;
8   **if** $(V(z, Q_k) \geq \Theta(u, Q_k))$ **then**
9   send $\langle qid, Q_k, T_m, H_d + 1, H_l - 1 \rangle$ to z;

When a node u receives a query message from node v, it determines the 'value' of each direct friend, the threshold value and then processes the query message using Algorithm 2. Once a target is found, a success message is generated which goes back to the query-issuer (user QID table, QID_u, helps in deciphering the reverse path) and this completes the search process. To initiate a query, the querying node sets QID to a new qid, Q_k to the set of query keywords, T_{min} to 1 (maximum allowed value), $Hops_{done}$ to 0 and $Hops_{left}$ to H_l (maximum hops to which it wants to search) generating the query message $\langle qid, Q_k, 1, 0, H_l \rangle$. By checking the 'value' of a node with a threshold and sending the message to only those friends which are above the threshold, we achieve the following benefits:

• This reduces the number of messages sent in the network. Aggressively pruning the search tree and reducing the load on the network helps in making the unstructured social network more scalable.

• The 'value' function chooses those targets that are topologically closer and uses links which have high trust returning good results amongst all obtainable results.

Analytical methods to analyze such designs are difficult to develop due to the lack of structure in the social network and the presence of multiple parameters in the algorithm. Thus, we use simulations to evaluate our algorithm.

V. SIMULATION METHODOLOGY

The evaluation methodology consists of four steps: generation of graphs with properties of social networks (high average values of node degree and clustering) as well as small world properties (low diameter), distribution of trust among the

edges, assignment of keywords to user nodes with corresponding policies and their propagation and finally, issuing queries from a set of nodes to see how the algorithm performs. We assumed that the time taken by the graph topology to change or for a user to update keywords and/or its policies is much greater than the time taken to search for a query. Thus, for simulation we used a static graph environment.

A. Graph Generation

In this work, we used a graph of one thousand nodes using the model given in [9] with properties observed during the measurement of existing online social networks [1], [2]. The properties of the generated graph are: average node degree = 18.63, diameter = 5 and clustering coefficient = 0.399. The small value of the diameter, high average node degree and high clustering coefficient emulate the properties possessed by realistic OSN graphs. Due to time considerations, we evaluated a graph of one thousand nodes only as a preliminary evaluation step at this stage.

B. Trust Distribution

The next step was to model the trust distribution among the edges. The classification of trust values and its distribution have been taken from [7]. The trust values are broadly classified into five categories: 'Blind Trust', 'High Trust', 'Medium Trust', 'Low Trust' and 'Don't Know'. We assigned a value of 0.9, 0.7, 0.5, 0.3 and 0.1 to each category respectively. The highest value of trust ('Blind Trust' = 0.9) signifies high trust among the nodes while the lowest value ('Don't Know' = 0.1) implies the presence of negligible trust. The edges were picked randomly to assign the trust values. The number of edges assigned to each category was based on the survey results presented in [7] where the final values of distribution are given as 19.28%, 12.27%, 34.35%, 25.95% and 8.15% respectively.

C. Information Propagation: Keywords and Policies

We set our information propagation process into two environments by varying the depth in the policy to see how the search algorithm performs in each of the environments.
- Restrictive Policy: In this case, all users uniformly set the policy associated with each keyword. When a keyword is added by a user, the maximum depth value (D) is set to 2, i.e. a keyword can travel from its originator to a maximum of two hops. The choice of two hops for restrictive policy is significant as it represents the 'friend of a friend' radius. This policy represents the case when users are very restrictive about their information being propagated in the network.
- Liberal Policy: When users are allowed to set any policy for keywords. Here, when a keyword is added by a user, a random integer between zero and diameter of the graph is set as the depth value in the policy. This helps us to observe the performance of the algorithm in a more realistic situation where some people are private in nature and concerned about who gets to access any information about them (and thus, set the depth value to 1 where only direct friends can access

the information) whereas other people let their information be accessible from many hops away in the network.

We next assigned to each keyword the weighted average of the trust values (that came out to be 0.4828) of the edges, as its policy, so that it uniformly propagates throughout the network. We randomly selected 100 nodes from the graph and initiated the process of keyword addition to these nodes and then propagated those keywords using Algorithm 1.

D. Query Distribution

Since, the search algorithm dynamically sets the value of the threshold by looking at the node degree and by applying the pruning function, it becomes necessary that query nodes be selected in such a way that they are representative of the various node degrees in the network. Thus, we sorted the nodes according to their node degree and picked 100 nodes having varying degrees with their degrees ranging from the minimum node degree (14) to the maximum node degree (26) of the generated graph. For each of the query nodes, we started the search algorithm and analyzed its behavior.

VI. RESULTS

We analyzed the performance of the algorithm by comparing the number of query messages generated and the number of targets found for three different pruning functions at different values of ρ and hops with the corresponding values for BFS. The pruning functions were obtained by setting p to 0.5, 1 and 2 such that $f(N_u^{Q_k}) = 1/\sqrt{N_u^{Q_k}}$, $1/N_u^{Q_k}$ and $1/(N_u^{Q_k})^2$ respectively. Different values of p were used to achieve different levels of pruning with $p = 0.5$ giving minimum pruning and $p = 2$ giving maximum pruning. We modeled BFS by setting $p = 0$ in the pruning function. Results have been normalized and averaged over 100 queries.

A. Performance at various ρ values

We varied ρ between 0.00 and 1.00 to obtain different linear combinations of distance and trust metrics. Figures 2 and 3 show the results for the networks with restrictive policy and liberal policy respectively. Figure 2a shows that the number of query messages sent during BFS is $\sim$121 while in the worst case evaluation of all pruning functions, the number of query messages is $\sim$12, showing a significant reduction in the number of query messages generated. Similar observations can also be made from figure 3a as the number of messages sent per query drops by orders of magnitude as compared to BFS. Here, the number of query messages is relatively larger as compared to figure 2a, because due to liberal policy, keyword(s) travel longer distances in the network.

Figures 2b and 3b show the targets found per query message for both the networks. The peak values observed for pruning functions are 0.5 and 0.135, whereas the corresponding values for BFS are 0.1 and 0.005 respectively. This means that our algorithm is much better at converting a generated query message to a successful probing message.

We also observe that for extreme values of ρ i.e. when distance or trust are considered separately, there is a sharp
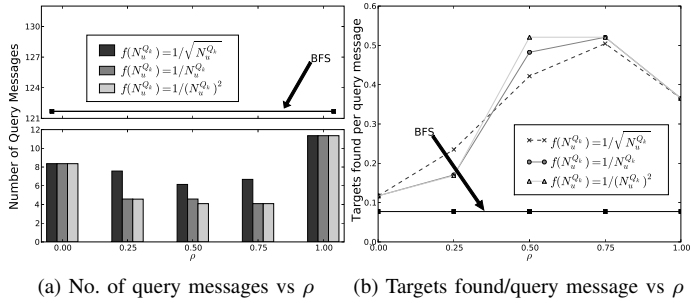
(a) No. of query messages vs ρ (b) Targets found/query message vs ρ

Fig. 2: Network with restrictive propagation policy

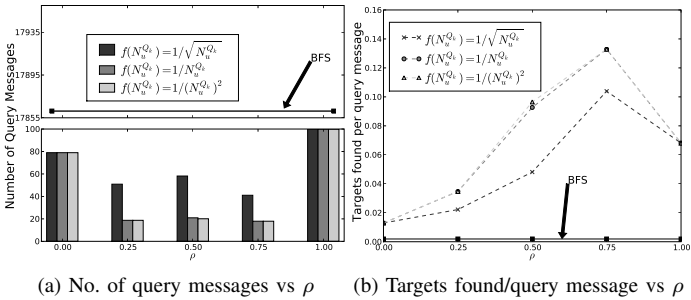(a) No. of query messages vs ρ (b) Targets found/query message vs ρ

Fig. 3: Network with liberal propagation policy

dip in the performance. For values of ρ between 0.50 and 0.75, the performance attains the peak level showing that a balanced combination of distance and trust metrics is best. We skip mentioning the absolute number of targets found as that can be determined from sub-figures of figures 2 and 3.

B. Performance at various hop values

All results in this section are given for targets found within different hops. We start by explaining the expected behavior of the algorithm. Let's assume that there are t potential targets when we probe the network till first hop. BFS should report each of these as it does not differentiate between their 'values' (definition 3). Since, the search algorithm differentiates between the values and adjusts the threshold to prune, it should return only a subset of these targets. This differentiation is based on trust of the edges as distance metric is the same for all of them and contributes equally towards the 'value'. Thus, the final subset should contain a fraction of t. When the search process probes up to two hops, such differentiation should continue and targets who have high trust on both the edges on the path connecting them to the querying node should be returned whereas BFS will again return a much larger set. The number of targets found at the second hop should increase when compared to the number of targets found till the first hop. But, the number of targets found as fraction of the number of targets found by BFS should decrease at second hop. This trend should continue while searching up to higher hops.

The expected behavior can be observed from table III and figure 4a. For pruning function with $p = 0.5$, the number of targets found using our algorithm are 0.73 and 1.68 at first and second hops while BFS finds 1.39 and 7.27 targets respectively (see table III for other hop values). The corresponding %age values when considered as fractions of BFS are ~52.5% and

(a) %age of targets found vs hops   (b) Targets found/query message vs hops
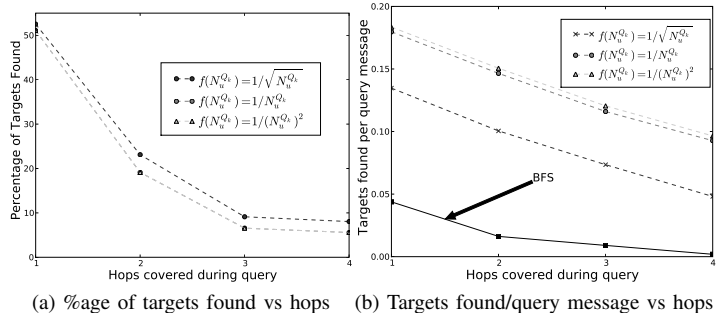
Fig. 4: Analysis of network (liberal policy) for varying hops

~23.1% (figure 4a). The number of targets found for pruning functions with $p = 1$ and $p = 2$ are the same (table III). An important feature of our algorithm is that it finds these targets with significantly fewer query messages than BFS (see figure 4b). For instance, at first hop, the number of targets found per query message vary in the range of 0.13 to 0.17 for different pruning functions which is much higher than the value of 0.004 observed for BFS. The same trend continues at higher hops. The trend line falls for all cases at higher hops because of the exponential growth in the number of probed links (table III).

In all graphs, it can be seen that as the denominator of the pruning function increases, the absolute number of results obtained decreases while the targets found per query message increases. This is because the search tree is pruned more effectively leading to fewer query messages. Thus, the pruning function decides how much of the network has to be searched. Depending on the steepness of the pruning function, the curve of these graphs may change but relative differences between values observed for BFS and pruning functions will remain, demonstrating the efficiency of the algorithm. We refrain from comparing the performance of our algorithm with other related works (section VII) as they either use structured networks or lack the level of information, in the form of keywords and their policies, that we used to design the algorithm.

VII. RELATED WORK

Search in social networks has been studied under different contexts. Structural knowledge about the network has been used to model search algorithms. In [10], the structure of the social network is based on an organizational hierarchy. Structural knowledge based on geographical distance, interest, etc. was used in [11], [12], [13]. These works base their search on greedy strategy as the linking probability between nodes was modeled to be proportional to the structural knowledge. But, in unstructured social networks these strategies find limited success as searching in an unstructured social network is a tougher problem than searching in a structured network [10]. We also studied works in peer to peer networks to gain insights on unstructured and decentralized networks.

Breadth First Search (BFS) has been used to search in decentralized and unstructured networks [14]. In random BFS [15], a node chooses some of its direct friends randomly

Hop Values → Pruning Function ↓	1		2		3		4	
	QM	TF	QM	TF	QM	TF	QM	TF
$f(N_u^{Q_k}) = 1/\sqrt{N_u^{Q_k}}$	5.42	0.73	16.71	1.68	34.83	2.56	58.15	2.80
$f(N_u^{Q_k}) = 1/N_u^{Q_k}$	3.95	0.71	9.49	1.39	15.86	1.84	20.93	1.94
$f(N_u^{Q_k}) = 1/(N_u^{Q_k})^2$	3.87	0.71	9.23	1.39	15.27	1.84	20.10	1.94
BFS $(f(N_u^{Q_k}) = 1)$	31.77	1.39	448.88	7.27	3116.83	28.06	17861.28	34.80

TABLE III: Query <u>M</u>essages and <u>T</u>argets <u>F</u>ound per query at different hops for the network with liberal policy and $\rho = 0.5$

to send the query message. Intelligent BFS [15] was also proposed where a node ranks peers for each query (using the cosine similarity) based on their past replies. In directed BFS [16], the query initiator chooses those friends who it thinks can give good results and after the first hop the search follows the flooding process. In iterative deepening [14], [16], search is continued with successively increasing TTL (hops to live for the message) till a successful result is found. Methods based on indexing or storing information of all nodes within some radius from a node [16], [17] have also been proposed.

Other search algorithms use random walks [18]. The k random walker algorithm [14] starts k random walks simultaneously instead of a single random walker. In the adaptive version [19], [20], the parameters k and TTL are adaptively updated. In two level random walk algorithm [21], the query initiator sends k_1 random walkers with some TTL and when the TTL expires, the last nodes send another k_2 random walkers with a new TTL. In [22] the high degree nodes are utilized while performing the search through random walks. Hybrid search schemes [23] involving a mixture of random walks and flooding have also been proposed.

VIII. CONCLUDING REMARKS

This paper modeled the flow of information in keyword based social networks. We developed a search algorithm for the given information flow settings that showed improvement in orders of magnitude when compared to BFS. The algorithm concentrated on finding a subset of results that have good characteristics. We do so with special focus on decentralization and privacy as proposed in future social network architectures. We believe that the algorithms presented can be adapted for network applications that may show graph properties similar to an unstructured social network.

As future work, we would like to explore this paper in multiple directions. First, we want to evaluate the search strategy for real social network graphs. As this is an ongoing work, we presented a limited preliminary evaluation on a synthetic graph here. Second, we are interested in further modeling the threshold function by incorporating the distribution of 'value' of friends between the minimum and maximum values. Third, we would like to extend the definition of trust to bidirectional trust for an edge and model accordingly. Finally, a broader expansion of the search algorithm using semantics of query keywords will form the next stages of this paper.

REFERENCES

[1] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *IMC '07: Proc. of the 7th ACM SIGCOMM Conf. on Internet measurement*, 2007.

[2] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in *WWW '07: Proc. of the 16th International Conf. on World Wide Web*, 2007.

[3] L. Banks, P. Bhattacharyya, and S. F. Wu, "Davis social links: Leveraging social networks for future internet communication," To appear in FIST '09: Workshop on Trust and Security in the Future Internet, 2009.

[4] L. Banks, S. Ye, Y. Huang, and S. F. Wu, "Davis social links: Integrating social networks with internet routing," in *LSAD '07: Proc. of the 2007 Workshop on Large Scale Attack Defense*, 2007.

[5] M. Chisari, "The future of social networking," in *W3C Workshop on the Future of Social Networking*, 2009.

[6] L. A. Cutillo, R. Molva, and T. Strufe, "Privacy preserving social networking through decentralization," in *WONS '09: Proc. of the 6th International Conf. on Wireless On-demand Network Systems and Services*, 2009.

[7] K. Sarda, P. Gupta, D. Mukherjee, S. Padhy, and H. Saran, "A distributed trust-based recommendation system on social networks," in *HotWeb '08: Proc. of the 2nd IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2008.

[8] P. Bhattacharyya, A. Garg, and S. F. Wu, "Social network model based on keyword categorization," To appear in ASONAM '09: Proc. of the International Conf. on Advances in Social Networks Analysis and Mining, 2009.

[9] M. E. J. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, 1999.

[10] L. Adamic and E. Adar, "How to search a social network," *Social Networks*, vol. 27, no. 3, pp. 187 – 203, 2005.

[11] J. Kleinberg, "The small-world phenomenon: An algorithm perspective," in *STOC '00: Proc. of the 32nd annual ACM symposium on Theory of computing*, 2000.

[12] O. Sandberg, "The structure and dynamics of navigable networks," Ph.D. dissertation, Chalmers University, 2007.

[13] J. Kleinberg, "Small-world phenomena and the dynamics of information," in *Advances in Neural Information Processing Systems*, 2001.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *ICS '02: Proc. of the 16th International Conf. on Supercomputing*, 2002.

[15] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," in *CIKM '02: Proc. of the 11th International Conf. on Information and knowledge management*, 2002.

[16] B. Yang and H. Garcia-Molina, "Improving search in peer-to-peer networks," in *ICDCS '02: Proc. of the 22 nd International Conf. on Distributed Computing Systems (ICDCS'02)*, 2002.

[17] A. Crespo and H. Garcia-Molina, "Routing indices for peer-to-peer systems," in *ICDCS '02: Proc. of the 22 nd International Conf. on Distributed Computing Systems (ICDCS'02)*, 2002.

[18] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks: algorithms and evaluation," *Performance Evaluation*, vol. 63, no. 3, pp. 241–263, 2006.

[19] N. Bisnik and A. Abouzeid, "Modeling and analysis of random walk search algorithms in p2p networks," in *HOT-P2P '05: Proc. of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems*, 2005.

[20] N. Bisnik and A. A. Abouzeid, "Optimizing random walk search algorithms in p2p networks," *Computer Networks*, vol. 51, no. 6, pp. 1499–1514, 2007.

[21] I. Jawhar and J. Wu, "A two-level random walk search protocol for peer-to-peer networks," in *SCI '04: Proc. of The 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2004.

[22] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman, "Search in power-law networks," *CoRR*, vol. cs.NI/0103016, 2001.

[23] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes in unstructured peer-to-peer networks," in *Proc. of IEEE INFOCOM*, 2005.