Client-side Web Mining for Community Formation in Peer-to-Peer Environments

Kun Liu, Kanishka Bhaduri, Kamalika Das, Phuong Nguyen and Hillol Kargupta [*]
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
{kunliu1, kanishk1, kdas1, phuong3, hillol}@cs.umbc.edu

ABSTRACT

In this paper we present a framework for forming interests-based Peer-to-Peer communities using client-side web browsing history. At the heart of this framework is the use of an order statistics-based approach to build communities with hierarchical structure. We have also carefully considered privacy concerns of the peers and adopted cryptographic protocols to measure similarity between them without disclosing their personal profiles. We evaluated our framework on a distributed data mining platform we have developed. The experimental results show that our framework could effectively build interests-based communities.

Keywords

Peer-to-Peer community, Order Statistics, Privacy Preserving Data Mining

1. INTRODUCTION

According to Maslow's theory [17], social motive, which drives people to seek contact with others and to build satisfying relations with them, is one of the most basic needs of human beings. The tendency to have affiliations with others is visible even in virtual environments such as the World Wide Web. Many online communities like Google and Yahoo groups provide the user a place to share knowledge, and to request and offer services. These communities are usually implemented as forums or mailing lists and under certain central control. As the Web continues to grow in both contents and the number of connected devices, Peer-to-Peer (P2P) distributed computing is becoming increasingly popular. Applications like Napster, KaZaA, BitTorrent, and SETI have already demonstrated the power of such computation. Peer-to-Peer technologies harness the CPUs and storage devices over the network to produce huge data stores, processing engines and communications systems. Each peer in the P2P environment acts as an autonomous and independent agent that shares knowledge by submitting queries and by replying with relevant information. Dynamically aggregating peers with similar interests could greatly enhance the capability of each individual, facilitate knowledge sharing, and reduce the network load. For example, a peer community allows the establishment of an abstract region of specialization. When a peer needs some relevant resources, the query could be propagated to the community members first to avoid the flooding of the request, and to maximize the quality of search results.

In this paper we address the problem of forming interest-based communities in a Peer-to-Peer environment. We define a Peer-to-Peer community as a collection of nodes in the network that share common interests. Traditional web mining has spent lots of efforts on the web server side, *e.g.* to analyze the server log. Instead, in this paper, we propose the usage of client-side information, namely, the web browsing cache, to model a peer's personal interests and to build Peer-to-Peer communities. Compared with other related work, our framework has the following specific features:

- It makes use of an order statistics-based approach to build communities with hierarchical structures.
- It considers privacy concerns of each peer, and adopts cryptographic protocols to measure similarity between peers without disclosing their personal profiles.
- Any technique that creates and represents a peer's personal profile as a feature vector can be plugged into our framework.

The remainder of this paper is organized as follows. Section 2 offers an overview of the literature on Peer-to-Peer community formation, Peer-to-Peer data mining, and privacy issues in Peer-to-Peer network. Section 3 presents some basic features of our Peer-to-Peer community framework. Section 4 and 5 address the community formation process. Section 6 discusses the message complexity of some key steps of the formation process. Section 7 studies the performance of the proposed framework and provides the experimental results. Finally, Section 8 concludes this paper with several directions for future work.

2. RELATED WORK

This section presents a brief overview of the literature on the formation of Peer-to-Peer communities, Peer-to-Peer

[*] Hillol Kargupta is also affiliated with AGNIK, LLC, USA.

WEBKDD'06, August 20, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-444-8 ...$5.00.

data mining, and privacy in Peer-to-Peer network. Due to the large volume of the literature we do not attempt a comprehensive citation listing. Instead we provide a sampling from a group of major categories.

2.1 Peer-to-Peer Communities

Generally speaking, the research on self-formation of Peer-to-Peer communities can be grouped into four major categories: 1) ontology matching-based approach; 2) attribute similarity-based approach; 3) trust-based approach; and 4) link analysis-based approach. We introduce each of them as follows.

Castano and Montanelli addressed the problem of formation of semantic Peer-to-Peer communities [4]. Each peer is associated with an ontology which gives a semantically rich representation of the interests that the peer exposes to the network, in terms of concepts, properties and semantic relations. Each peer interacts with others by submitting discovery queries in order to identify the potential members of an interest-based community, and by replying to incoming queries whether it can join a community. A semantic matchmaker is employed to check whether two peer share the same interests. The matchmaker performs dynamic ontology matching by taking into account both linguistic and contextual features of the concepts to be compared. The advantage of this approach is that peers do not have to agree on the same predefined ontology, and therefore they have lots of flexibility of describing their interests. However, the gain of flexibility comes at the price of accuracy because of the uncertainty of concepts. We refer the reader to [19] for a brief survey of existing ontology matching approaches. The other drawback of this approach is that a peer's interests are inevitably revealed, even to the peers that do not belong to the community, therefore the privacy of the peer is compromised.

Khambatti *et al.* proposed a Peer-to-Peer community discovery approach where each peer is associated with a set of attributes that represent the interests of that peer [15]. These attributes are chosen from a controlled vocabulary that each peer agrees with, which gets rid of the uncertainty of the fuzzy ontology matching. Peers whose attributes have non-empty intersection can be grouped together. A very basic privacy policy is applied such that a peer does not disclose attributes corresponding to its private interests. This means that the smaller the number of claimed attributes, the smaller the number of communities or community members discovered by a peer. In this paper, we also assume each peer has a set of attributes, which we call as profile vector. The difference is that each interest in the profile vector can be given a weight to show its importance. Moreover, we do not simply check the intersection of attributes, instead, we quantitatively compute the similarity between profile vectors (using scalar product), and we use an order statistics-based algorithm that can tell how similar a pair of peers are to each other in the whole network. Our privacy management scheme enables each peer to measure the similarity with other peer without worrying about privacy breach.

Trust-based community formation is usually discussed in the scenario of file sharing and service providing. The notation "trust" is a measure used by a peer to evaluate other peer's capability of providing a good quality service or resource. This trust is based on information about the peer's past behavior. Once a peer finds trustworthy peers, it invites them to join its community. We refer the reader to [27, 1] as a starting point on this topic. In this paper, we are interested in forming a community based on peers' interests without considering the past interactions of peers.

There exists another area of research that focuses on the link structure analysis of network to identify patterns of interaction. For example, Scott identified the various cliques, components and circles into which networks are formed [22]. Flake *et al.* described an approach to identify web communities [8]. Here a web community is a collection of web pages in which each member page has more hyperlinks within the community than outside it. Such communities help to create improved search engines, to perform content filtering, etc. The drawback of link analysis-based approach is that it depends on the stable link structure of the network, and therefore precludes a peer from being a member of more than one community simultaneously.

2.2 Peer-to-Peer Data Mining

Peer-to-Peer data mining is a relatively new field. It pays careful attention to the distributed resources of data, computing, communication, and human factors in order to use them in a near optimal fashion. Wolff *et al.* proposed algorithms for association rule mining [29] and local L2 norm monitoring [28] over P2P networks. Datta *et al.* proposed an algorithm for K-Means clustering over large, dynamic networks [5].

2.3 Privacy in Peer-to-Peer Network

The objective of large scale distributed network is to maximize the availability and utilization of information. This goal would be achieved if the free flow of information was ensured, and if the owners of different data resources were able to share the data with each other. However, this is frequently restricted by legal obligations or by commercial and personal privacy concerns. Privacy, or lack of it, is becoming an increasingly important issue in many distributed application scenarios including file sharing, cooperative computation, etc. Previous research on privacy in Peer-to-Peer network can be roughly classified into two categories: 1) user anonymity; and 2) data privacy.

User anonymity aims at offering the users privacy protection by letting them hide their identities from the communicating peers or from malicious eavesdroppers. There are many uses of anonymous P2P technology that help internet users surf the web anonymously and shield their online activities from corporate or government eyes. Anonymous communication system is also used by government for intelligence gathering and politically sensitive negotiations. Usually a special protocol for anonymous routing is applied in the network (see *e.g.* [2]). The anonymity comes from the idea that no one knows who requested the information as it is difficult – if not impossible – to determine whether a user requested the data for himself or simply requested the data on behalf o somebody else. The end result is that everybody on the network acts as a universal sender and universal receiver to maintain anonymity. There are many decentralized anonymous and censorship-resistant P2P frameworks in the market such as the Freenet [9] and the GNUnet [11], to name a few.

The objective of protecting data privacy is to hide the sensitive information owned by a peer from being disclosed in a cooperative computation environment, where the revelation of a peer's identity is unavoidable. For example, it may not be possible to hide the identity (*e.g.* IP, port number, URI) of a peer in a Peer-to-Peer community since without this information, peers may not be able to communicate with each other. To be more specific, the data privacy problem in a large scale cooperative computation environment can be defined as follows. Assume that n participants $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, each owning a private input x_i, wish to jointly compute the output $f(x_1, x_2, \ldots, x_n)$ of some common function f, without revealing anything but the output. Privacy preserving data mining (PPDM) [26] strives to provide a solution to this problem. It aims to allow useful data patterns to be extracted without compromising privacy. For example, Gilburd *et al.* presented a privacy model called k-TTP for large-scale distributed environment [10]. The intuition is that at any time each participant can only learn a combined statistics of a group of at least k participants, and therefore any specific participant's private input is hidden among at least $k-1$ other participants' input. In Section 5.3 we will revisit this problem and discuss how to compute the scalar product of two private vectors owned by two peers.

3. FEATURES OF PEER-TO-PEER COMMUNITY

In this section, we present some features that characterize the formation of our Peer-to-Peer communities.

3.1 Peer Profiles

A crucial issue in forming Peer-to-Peer communities is to create peer profiles that accurately reflects a peer's interests. These interests can be either explicitly claimed by a peer, or implicitly discovered from the peer's behaviors. A peer's profile is usually represented by a keyword/concept vector. Trajkova and Gauch proposed techniques to implicitly build ontology-based user profiles by automatically monitoring the user's browsing habits [25]. The system classifies each web page the user has visited into the most similar concept in a predefined hierarchy of ontology. Each element of the user profile vector corresponds to the weight or the number of pages associated with that concept in the ontology. The Open Directory Project concept hierarchy [1] was used as the reference ontology. Figure 1 shows a sample ontology for user profile. Other sources of information have also been used in the literature to create profiles, such as using bookmarks [24], using queries and search results [23], etc. We refer the reader to [25] for a brief overview on this topic.

We point out that any approach that represents a peer's profile in a feature vector can be used in our framework. In this paper, we use the frequencies of the web domains a peer has visited during a period of time to construct the peer's profile vector. Each web domain can be viewed as an interest or topic and hence the frequency represents the weight of the interest for that topic. Detailed explanation about data collection is given in Section 7. To avoid the uncertainty of ontology matching, we expect all peers to agree on the same ontology defined by a controlled vocabulary. In this paper, this means that all peers agree on a superset of web domain names.

3.2 Similarity Measurement

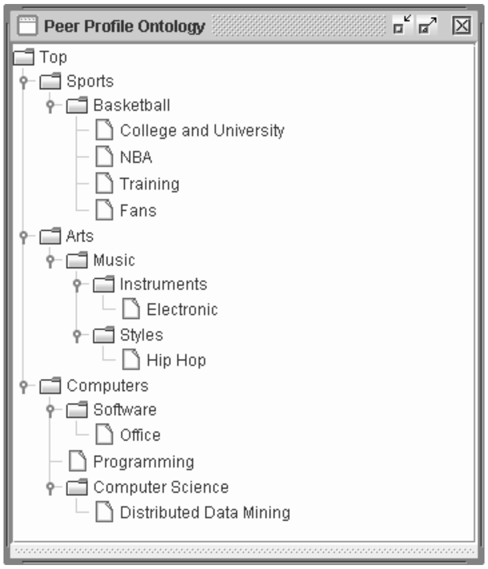


Figure 1: A sample ontology for user profile.

The goal of community formation is to find peers sharing similar interests. However, one of the important questions is how a peer can decide whether another peer is similar to him, to what extent? If we simply choose a similarity measurement Ω and setup a subjective threshold such that peers with similarities greater than this threshold can be grouped together, we can't provide any statistical guarantee. Furthermore, this approach is not able to reflect the essential characteristics of a social community, namely, *hierarchy.* In a social network, a person may have multi-level friends, where the first level might be family members and closest friends, the second level might be some other colleagues. A person could also have indirect friends from his/her friends' social network. A Peer-to-Peer community from one peer's perspective should also have such kind of hierarchical structure. That is, some peers share more interests with this peer, and some less, under some similarity measurement.

To achieve this goal, we propose an order statistics-based approach (to be described later in Section 5.1) that enables a peer to know how similar the other peer is to himself. In other words, our statistical measurement guarantees that if the similarity between peer P_i and P_j is above a threshold, P_i can determine with confidence level q that P_j is among the top $(1-p)$ quantile most similar peers of P_i's. Here the quantile, denoted by ξ_p with $0 < p < 1$, of a continuous random variable $\mathbf{X}$ is defined by $Pr\{x \leq \xi_p\} = p$, *e.g.* $\xi_{0.5}$ is called the median of the distribution. We use the term "top $(1-p)$ quantile" to denote the area $[\xi_p, \xi_1)$, *e.g.* top (1-0.9) quantile means the largest 10% of data. As a running example, let us assume there are 5 peers $\{P_1, P_2, P_3, P_4, P_5\}$ in the network, and the similarity measures between P_1 and all other peers are $\{1, 3, 2, 4\}$, respectively, where the higher the value, the higher the similarity. If P_1 knows the similarity between him and P_5 is 4, our approach will enable P_1 to know, with high confidence, that P_5 is among the top 25% most similar peers of P_1's in the network, without computing all the similarity values.

Now we formally define a Peer-to-Peer community based

[1] Open Directory Project – http://dmoz.org/

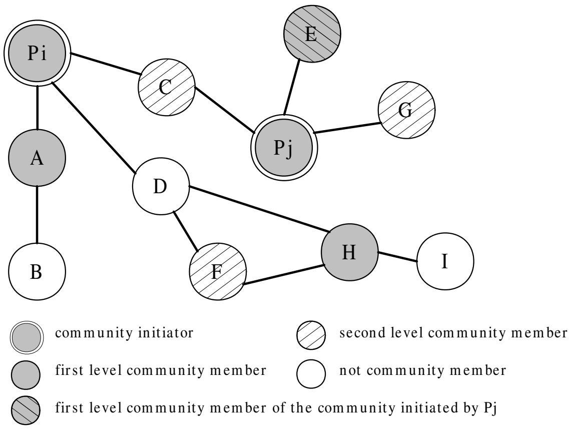

Pi
E
C
G
Pj
A
D
H
I
B
F
community initiator
second level community member
first level community member
not community member
first level community member of the community initiated by Pj

Figure 2: Example of Peer-to-Peer communities.

on our above discussion.

Definition 3.1. [(Ω, p, q)-P2P Community] *A (Ω, p, q)-P2P community from peer P_i's view is a collection of peers in the network, denoted by $\mathcal{C}$, such that the similarity measures Ω between P_i and all the members in $\mathcal{C}$ are among the top $(1-p)$ quantile of the population of similarity measures between P_i and all the peers in the network, with confidence level q.*

Definition 3.2. [Extended (Ω, p, q)-P2P Community] *An extended (Ω, p, q)-P2P community from peer P_i's view is the union of $\mathcal{C}$ (defined by Definition 3.1) and all the peers from the (Ω, p, q)-P2P community of each member in $\mathcal{C}$.*

These two definitions implicitly capture the hierarchical characteristics of the community. When a peer finds a similar buddy, he could compute the quantile value and determine which area this buddy belongs to. A peer could also specify a p value and only invite those belonging to top $(1-p)$ quantile area to be his community members. The community could be expanded to include members from members's community. For example, in Figure 2, Peer A, P_j, H are the first level members (with larger p) of community initiated by P_i. Peer C, F and G are the second level members (with smaller p) of community. Note that P_j is also a initiator of another community, and it has E as its first level community member. Peer A, P_j, H, E compose an extended P2P community initiated by P_i.

In this paper, we use the scalar product between two profile vectors to quantify the similarity between two peers. Other similarity metrics such as Euclidean distance can also be applied in our framework without any hurdle. Details on how to determine the threshold for quantiles using order statistics theory are given in Section 5.1.

3.3 Privacy Management

A major drawback of most existing community formation approaches is that none of them take serious consideration to protect a peer's privacy. For example, a peer may not want to reveal some of his interests, or the weights of his interests in his profile vector. Privacy becomes an extremely important issue especially when the profile is implicitly discovered from the peer's personal activities.

In our framework, we provide the peer with two-level privacy protection. The first level allows the peer to explicitly filter out extremely private sensitive interests by assigning zero weights to the corresponding concepts in the profile vector. The second level protection relies on the notion of cryptographic secure multi-party computation (SMC) [31]. Loosely speaking, SMC considers the problem of evaluating a function of the private inputs from two or more parties, such that no party learns anything beyond what can be implied from the party's own input and the designated output of the function. We adopt private protocols that are proved to be cryptographic secure such that any pair of peers can compute the similarity of their interests without knowing each other's actual profile vector. Details about the private computation are given in Section 5.3.

We need to note that no protocols can build a similar interest-based community without revealing the information that *these peers share the same interests.* A high similarity value between two peers tells them they have a lot in common, and their profile vectors are close. Nevertheless, SMC-based protocols can guarantee that neither party would know the other's actual input, namely, the actual profile vector. Moreover, if the similarity value is low, no significant information about the other peer's interest is disclosed.

4. COMMUNITY FORMATION PROCESS

In this section, we address the Peer-to-Peer community formation process under the assumptions that: 1) each peer can be a member of multiple virtual communities; 2) peers interact with each other by submitting or replying queries to determine the potential members of a given community; and 3) there is no superpeer as a centralized authority.

The Peer-to-Peer community emerges as a peer, P_i, called community initiator, invokes a community discovery process which consists of the following tasks: *sample size computation, quantiles estimation, member identification, member notification and acceptance,* and *community expansion.*

- **Sample Size Computation:** The initiator P_i first selects a confidence level q and the order of population quantile p it would tolerate. Based on the algorithm described in Section 5.1, the initiator calculates the number of samples required to compute the threshold such that any peers that have similarity values with P_i greater than this threshold are among the top $(1-p)$ quantile most similar peers of P_i's. Let us denote the sample size as N.

- **Quantiles Estimation:** Given the sample size N, the initiator invokes N random walks using the protocols described in Section 5.2 to choose independent sample peers in the network. Whenever a new peer P_j is chosen, it replies to P_i with its address and port number, and builds an end-to-end connection with P_i. Then P_i computes the scalar product of its profile vector and P_j's profile vector using the private scalar product protocol described in Section 5.3. This private protocol guarantees that neither P_i nor P_j could know the other party's profile. After P_i collects all the N scalar products, it finds the largest one as the threshold for quantiles of order p.

- **Member Identification:** The initiator P_i composes a discovery message containing its address and port number, as well as a time-to-live (TTL) parameter defining the maximum number of hops allowed for the discovery propagation. Then the discovery message is sent to all P_i neighbors. When a peer P_j receives this message, it replies to P_i with its address and port number. P_i then invokes a private scalar product computation (to be described in Section 5.3) to get the similarity value. Independently from the similarity computation and if TTL ≥ 0, P_j forwards the discovery message to all its neighbors, except for the peer from which the message has been received. Each peer discards duplicate copies of the same discovery message possibly received.

- **Member Invitation and Acceptance:** The initiator P_i evaluates the quality of the discovered peers by comparing the similarity values with its threshold. If the similarity is above the threshold, P_i sends an invitation message to that peer. If the similarity is below the threshold, P_i still could analyze, with the same confidence level, the order of quantile that the peer belongs to; but note that this order will be lower than the preset p. Given this information, P_i can decide whether to send an invitation to a peer with less similarity. For the sake of simplicity, in our experiments, P_i will not send invitations in this circumstance. Once a peer P_j receives an invitation message, it decides whether to accept it or not by replying an acceptance message. Receiving the acceptance message, P_i records P_j in its local cache.

- **Community Expansion:** When a peer P_j accepts the invitation, it replies to the initiator an acceptance message, as well as the member lists in its local cache. These members are from the P2P community or extended P2P community initiated by P_j. As a reward, the initiator sends the current member list in its local cache to P_j. In this way, each peer has an extended Peer-to-Peer community.

In our framework, peers interact with each other by sending discovery queries, or by answering queries. If a peer P_i polls another peer P_j but does not get a reply in a reasonable amount of time, P_i simply assumes that P_j has left the network. In this case, P_i can resend query to other peers to get necessary information.

5. BUILDING BLOCKS

This section elaborates on some building blocks that are necessary to complete the Peer-to-Peer community formation process.

5.1 Distribution-Free Confidence Interval for Quantiles

Given $\mathbf{x}$, a feature vector, and $\mathcal{Y}$, a set of other feature vectors, we want to find out how similar $\mathbf{x}$ and a $\mathbf{y} \in \mathcal{Y}$ are to each other in comparison with the similarities of $\mathbf{x}$ and other $\mathbf{y}$s in $\mathcal{Y}$. A trivial approach to this problem would be to collect the entire set of $\mathcal{Y}$ and compare all the scalar products of $\mathbf{x}$ and $\mathcal{Y}$. This simple approach, however, does not work in a large-scale distributed P2P environment because the network state is not stable with frequent nodes arrivals and departures, and the overhead of communication would be extremely high. Theories from order statistics, however, could relieve us from this burden by considering only a small set of samples from $\mathcal{Y}$ and producing a solution with probabilistic performance guarantees. The following part of this section discusses this possibility.

Let $\mathbf{X}$ be a continuous random variable with a strictly increasing cumulative density function (CDF) $F_{\mathbf{X}}(x)$. Let ξ_p be the population quantile of order p, *i.e.* $F_{\mathbf{X}}(\xi_p) = Pr\{x \leq \xi_p\} = p$. Suppose we take N independent samples from the given population $\mathbf{X}$ and write the ordered samples as $x_1 < x_2 < \cdots < x_N$. We are interested in computing the value of N that guarantees

$$Pr\{x_N > \xi_p\} > q.$$

Since

$$\begin{aligned} Pr\{x_N > \xi_p\} &= 1 - Pr\{x_N \leq \xi_p\} \\ &= 1 - Pr\{\text{all the N samples} \leq \xi_p\} \\ &= 1 - p^N, \end{aligned}$$

we have

$$1 - p^N > q \Rightarrow N \geq \left\lceil \frac{\log(1-q)}{\log(p)} \right\rceil. \quad (1)$$

For example, for $q = 0.95$ and $p = 0.80$, the value of N obtained from the above expression is 14. That is, if we took 14 independent samples from any distribution, we can be 95% confident that 80% of the population would below the largest order statistic x_{14}. In other words, any sample with value greater or equal to x_{14} would be in the top 20 quantile of the population with 95% confidence. The smaller the p is, the smaller the N, *e.g.* when $p = 0.70$, $N = 9$. Therefore, given 14 samples, we can also determine the threshold for any quantile of order less than 0.80. Recall in the community formation process, if the initiator P_i finds a peer P_j with similarity value less the threshold, the initiator cannot say P_j is among the top $1-p$ quantile most similar peers, but the initiator can still find out a smaller $p' < p$ and determine with the same confidence level that P_j is among the top $1-p'$ quantile most similar peers. For detailed treatment of order statistics, we refer the reader to David's book [6].

When $\mathbf{X}$ is discrete, the equation $F_{\mathbf{X}}(x) = p$ does not have a unique solution. However, ξ_p can still be defined by $Pr\{x < \xi_p\} \leq p \leq Pr\{x \leq \xi_p\}$. This gives ξ_p uniquely unless $F_{\mathbf{X}}(\xi_p)$ equals p, in which case ξ_p again lies in an interval. It can be shown that in this case, $Pr\{x_N < \xi_p\} \leq I_p(N,1) = p^N$, where $I_p(N,1)$ is the incomplete beta function. Therefore, in the discrete scenario, we have

$$\begin{aligned} Pr\{x_N \geq \xi_p\} &= 1 - Pr\{x_N < \xi_p\} \\ &\geq 1 - p^N > q. \end{aligned}$$

This does not change the conclusion from Eq. 1.

5.2 Random Sampling

Random sampling in the networks is a prerequisite to the estimation of population quantile. It can be performed by modeling the network as an undirected graph with transition probability on each edge, and defining a corresponding Markov chain. Random walks of prescribed length on this graph produce a stationary state probability vector and the corresponding random sample. The simplest random walk algorithm chooses an outgoing edge at every node with equal probability, *e.g.* if a node has degree five, each of the edges is traversed with a probability 0.2. However, it can be shown that this approach does not yield a uniform sample of the network unless the degrees of all nodes are equal [16]. Since typical large-scale Peer-to-Peer network tends to have non-uniform degree distribution, this approach will generate a biased sample in most practical scenarios.

Fortunately, the elegant Metropolis-Hastings algorithm [18, 13] implies a simple way to modify the transition probability so that it leads to a uniform stationary state distribution, and therefore results in uniform sample. In this paper, we implement an adaptation of this classical algorithm. The work in [3] proposed a more efficient random walk algorithm, the Random Weight Distribution (RWD) algorithm, that allows uniform sampling while minimizing the length of the walk. We will experiment with that algorithm in our future work. Next we briefly introduce the Metropolis-Hastings algorithm for random walk.

Let $G(V, E)$ be a connected undirected graph with $|V| = n$ nodes and $|E| = m$ edges. Let d_i denote the degree of a node i, $1 \leq i \leq n$. The set of neighbors of node i is given by $\Psi(i)$ where $\forall j \in \Psi(i)$, edge $(i, j) \in E$. Let $P = \{p_{ij}\}$ represent the $n \times n$ transition probability matrix, where p_{ij} is the probability of walking from node i to node j in one message hop. $0 \leq p_{ij} \leq 1$ and $\sum_j p_{ij} = 1$. Protocol 5.2.1 gives the basic algorithm.

Protocol 5.2.1 Metropolis-Hastings Random Walk

```
FOR each node i, $1 \leq i \leq n$
  IF receives a query $q$
    Replies with $d_i$
  IF receives a random walk message
    IF TTL == 0
      Terminates the walk
    ELSE
      TTL = TTL - 1
      Sends out a query $q$ to its neighbors $\Psi(i)$
      IF receives all the replies from its $\Psi(i)$
        Modifies transition probability $p_{ij}$ as follows:
        $p_{ij} = \begin{cases} 1/\max(d_i, d_j) & \text{if } i \neq j \text{ and } j \in \Psi(i) \\ 1 - \sum_{k \in \Psi(i)} p_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
        Walk to next node with probability $p_{ij}$.
```

This algorithm generates a symmetric transition probability matrix and is proved to produce uniform sampling via random walk [3]. As stated in [16], the length of random walk necessary to reach to stationary state has order $O(\log n)$. Empirical results show that when the length of walk is $10 \times \log n$, this algorithm approaches uniform distribution reasonably well. The network size n could be estimated using the localized estimation scheme proposed in [14]. Figure 3 shows the probability of selection using the Metropolis-Hastings algorithm over a simulated network with 5000 nodes. As can be easily seen, the probability of selection is almost uniform even for varying degree distribution. The longer the simulation runs, the clearer the curve shows a uniform distribution.

To implement a random walk, the initiator sends out a message contains a time-to-live (TTL) parameter that indicates the length of the walk. Whenever a node receives the message, it checks the TTL parameter. If TTL is 0, then the walk terminates; otherwise, this node decreases TTL by 1 and forwards the message to the next node based on the transition probability matrix.

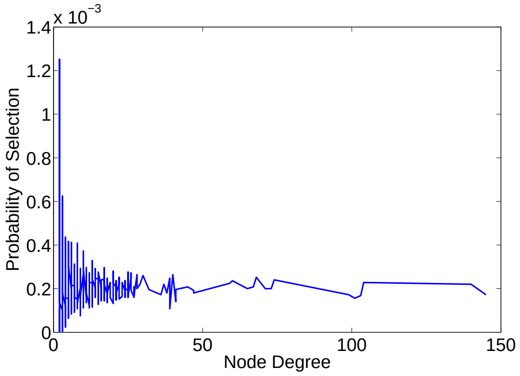


Figure 3: Selection probability in a simulated network with 5000 nodes (nodes arranged by their degrees).

5.3 Private Scalar Product Computation

Private scalar product computation serves as an important building block for privacy preserving data mining [26]. It considers the problem of computing the scalar product of two vectors owned by two different parties, respectively, so that neither party should learn anything beyond what is implied by the party's own vector and the output of the computation. Here the output for a party is either the scalar product or nothing, depending on what the party is supposed to learn. Many private scalar product protocols have been proposed in the literature. Generally speaking, these protocols can be classified into two categories: 1) cryptosystem-based approaches, which offer strong privacy protection, but incur high communication and computational cost (*e.g.* [30]); and 2) data perturbation-based approaches, which provide weaker privacy protection but allow more efficient solutions for more complicated data mining tasks (*e.g.* [7]). We refer the reader to [12] for an overview on this topic.

In this paper, we adopt the protocol proposed in [12] for the private similarity computation. This protocol is proved to be private in a strong cryptographic sense. To be more specific, no probabilistic polynomial time algorithm substituting one party can obtain a non-negligible amount of information about the other party's private input, except what can be deduced from the input and output of this party. Protocol 5.3.1 describes the basic procedures.

Protocol 5.3.1 Private Scalar Product

Private Input of Alice: Vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{Z}_\mu^d$
Private Input of Bob: Vector $\mathbf{y} = (y_1, \ldots, y_d) \in \mathbb{Z}_\mu^d$
Output of Alice: $\mathbf{x} \cdot \mathbf{y} \bmod \mu$

1: Alice generates a private and public key pair (sk, pk), and sends pk to Bob.
2: For each $i, i = 1, \ldots d$, Alice generates a random number $r_i \in Z_\mu$, and sends $c_i = E_{pk}(x_i, r_i)$ to Bob.
3: Bob computes $w = \prod_{i=1}^{d} c_i^{y_i} \bmod \mu^2$ and sends w back to Alice.
4: Alice computes $\mathbf{x} \cdot \mathbf{y} \bmod \mu = D_{sk}(w)$.

To understand this protocol, let us first take a brief review of public-key cryptosystem. A public-key cryptosystem $\mathcal{P}(G, E, D)$ is a collection of probabilistic polynomial time algorithms for key generation, encryption and decryption.

The key generation algorithm G produces a private key sk and public key pk with specified key size. Anybody can encrypt a message with the public key, but only the holder of a private key can actually decrypt the message and read it. The encryption algorithm E take as an input a plaintext m, a random value r and a public key pk and outputs the corresponding ciphertext $E_{pk}(m, r)$. The decryption algorithm D takes as an input a ciphertext c and a private key sk (corresponding to the public key pk) and outputs a plaintext $D_{sk}(c)$. It is required that $D_{sk}(E_{pk}(m, r)) = m$. The plaintext is usually assumed to be from $\mathbb{Z}_\mu$, [2] where μ is the product of two large primes. A public-key cryptosystem is homomorphic when

$$\begin{aligned} &\forall m_1, m_2, r_1, r_2 \in \mathbb{Z}_\mu, \\ D_{sk}(E_{pk}(m_1, r_1)E_{sk}(m_2, r_2) \bmod \mu^2) &= m_1 + m_2 \bmod \mu; \\ D_{sk}(E_{pk}(m_1, r_1)^{m_2} \bmod \mu^2) &= m_1 m_2 \bmod \mu; \\ D_{sk}(E_{pk}(m_2, r_2)^{m_1} \bmod \mu^2) &= m_1 m_2 \bmod \mu. \end{aligned}$$

This feature allows a party to add or multiply plaintexts by doing simple computations with ciphertexts, without having the secret key. That is why Bob could compute an encrypted scalar product without knowing Alice's private inputs in Step 3 of the above protocols.

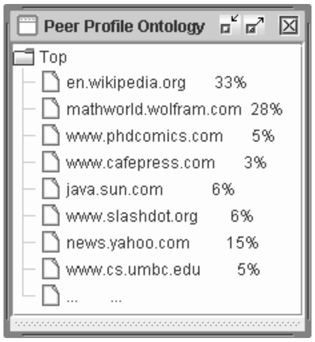
In our implementation, we use the Paillier cryptosystem [20] for public-key cryptography. Both encryption and decryption require modular exponentiations and modular multiplications of large numbers. The bit complexity of these basic operations in $\mathbb{Z}_\mu$ is $O(|\mu|^3)$, where $|\mu|$ is the size of the public key in bits. In Protocol 5.3.1, Alice performs k encryptions and one decryption; and Bob does not perform any cryptographic operations. Therefore a great amount of computation is attributable to Alice, which becomes the bottleneck of the whole process. However, in our scenario, this protocol can be optimized since each feature vector contains lots of 0's because the peer has never visited the corresponding web sites. Thus, each peer can pre-compute a large table of random encryptions of 0's (each encryption uses different r and therefore produces different ciphertext for 0). Then every encryption of 0 simply corresponds to fetching an element from the table, which can be very fast. The communication overhead of this protocol is $2|\mu|/|m|$, where $|m|$ is the size of the original plaintext in bits.

6. MESSAGE COMPLEXITY

This section discusses the message complexity of some key steps of our community formation process.

Random walk to fetch samples: Let the size of the network be n, the number of samples necessary be N (refer to Equation 1) and the length of a single random walk be λ. The community initiator needs to launch N parallel random walks. Each random walk message needs to carry a TTL token (which is initially set to the length of the walk), the IP address and port number of the initiator. If each of them are represented by a 32-bit integer, the total message complexity is $32 \times 3 \times N \times \lambda$ bits.

Reply to initiator: Once a peer receives the TTL with value 0, it needs to send its IP address and port number back to the initiator node for doing the private scalar product computation. Since there are N samples to be collected, there are N such peers. The message complexity in this step will be $32 \times 2 \times N$ bits.



Peer Profile Ontology

Top

en.wikipedia.org	33%
mathworld.wolfram.com	28%
www.phdcomics.com	5%
www.cafepress.com	3%
java.sun.com	6%
www.slashdot.org	6%
news.yahoo.com	15%
www.cs.umbc.edu	5%
...	...

Figure 4: Snapshot of a peer's profile. There are 722 such domains for each peer.

Private scalar product computation: Alice encrypts each element of her vector with a public key of size $|\mu|$ bits. The size of each ciphertext is $2 \times |\mu|$ bits. Alice then sends the entire encrypted vector to the Bob for computation. The total message payload for a vector of dimension d is therefore $2 \times |\mu| \times d$. After that Bob sends the encrypted scalar product ($2 \times |\mu|$ bits) back to Alice. Hence the total message complexity for N samples is $2 \times |\mu| \times (d + 1) \times N$ bits.

Message complexity for member identification and invitation can be derived in a similar way using the above results. We omit the analysis due to the space constraints.

7. EXPERIMENTS

In this section, we study the effectiveness of the proposed framework for Peer-to-Peer community formation.

7.1 Data Preparation

We use the web domains a peer has browsed to create the profile vector. Each element of the vector corresponds to the frequency that the domain has been visited by the peer during a period of time. The data was collected from the Internet Explorer (IE) history files of 5 volunteers from the DIADIC Research Lab at UMBC and 10 volunteers from the DSP Lab at Johns Hopkins University. There are in total 1387 KB of data accounting for 97050 browsing history records in our data set, and 722 unique web domains. These records are randomly split and distributed to peers in our network simulator so that each peer can compute its own profile vector. As we have stated previously, we assume all the peers agree on the same profile ontology, *i.e.* the same set of domain names, and therefore, all the profile vectors have the same size - 722. Figure 4 shows a snapshot of a peer's profile.

7.2 Network Topology and Simulator

Our network topology was generated by using *BarabasiAlbert* Model from BRITE [3], a universal topology generator. *BarabasiAlbert* model interconnects the nodes according to a incremental growth approach, *i.e.,* when a node i joins the network, the probability that it connects to a node j already

[2] The integers modulo μ, denoted $\mathbb{Z}_\mu$, is the set of (equivalence classes of) integers $\{0, 1, \ldots, \mu - 1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_\mu$ are performed modulo μ.

[3] BRITE - http://www.cs.bu.edu/brite/

belonging to the network is given by:

$$Pr(i,j) \quad = \quad \frac{d_j}{\sum_{k \in V} d_k},$$

where d_j is the degree of the target node, V is the set of nodes that have joined the network and $\sum_{k \in V} d_k$ is the sum of out degrees of all nodes that previously joined the network. We choose this model because it does a fairly good job in reproducing the out degree properties of Internet topologies.

We use the Distributed Data Mining Toolkit (DDMT) [4] developed by the DIADIC research lab at UMBC to simulate the distributed computing environment. This toolkit is build upon JADE (Java Agent DEvelopment Framework) [5], a multi-agent systems platform. We implemented all of our algorithms in Java JDK 1.5, and performed the experiments on a dual-processor workstation running Windows XP with 3.00GHz and 2.99GHz Xeon CPUs and 3.00GB RAM.

7.3 Effectiveness

Having discussed about the data and the simulator setup we are in a position to report the experimental results.

7.3.1 *Random Sampling and Quantile Estimation*

This experiment evaluates the accuracy of random sampling and quantile estimation. We chose three different p values - $80\%, 85\%$ and 90%. In all the three cases, the confidence level q was set to 95%, and the size of the network was fixed at 100 nodes. According to Equation 1, the number of samples, denoted by N, necessary to guarantee that the highest order statistic is within the top $(1-p)$ percentile of population is given by $14, 19$ and 29, respectively. Let P_i be the community initiator. The population can be defined as the set of all pairwise scalar products between P_i and all the other peers. Now, if P_i wants to find similar peers who are in the top $(1-p)$ quantile of the population, it launches N random walks. The terminal peer for each random walk refers to a sample and P_i computes the private scalar product between its own vector and the vector owned by the sample. P_i sorts all the N scalar products and finds the largest one as the threshold of quantile of order p. Figure 5 shows estimated threshold in the distributed experiment. To compare the results with centralized sampling, P_i first collects the pairwise scalar products between itself and all the peers in the network. P_i then performs a random sampling of size N and finds the largest scalar product. The threshold found by this approach is illustrated by the stars in Figure 5. Figure 5 also shows the actual population quantile of order p. As is evident from these results, the threshold found through random sampling and order statistics theory is above the actual population quantile. Therefore any scalar product greater than this threshold can be recognized as among the top $(1-p)$ quantile population with high confidence, which validates our claim in Section 5.1.

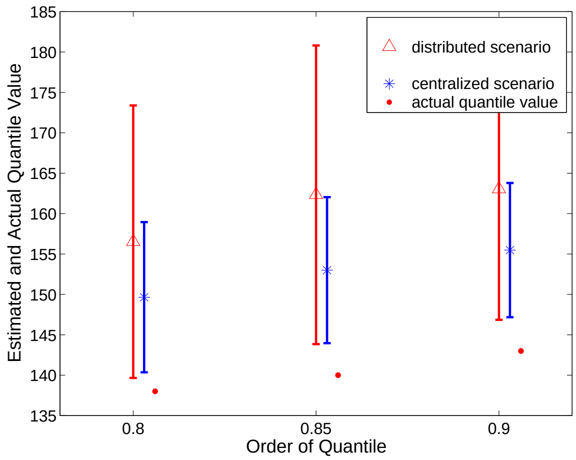


Figure 5: Estimated and actual quantile value w.r.t. the order of quantile. The results are an average of 100 **independent runs.**

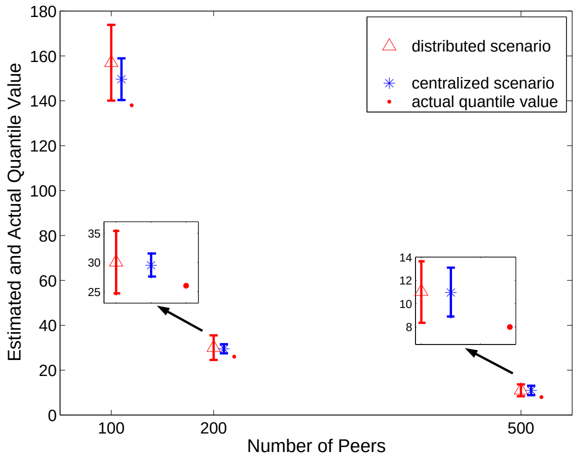


Figure 6: Estimated and actual quantile value w.r.t. the number of peers for fixed $p = 0.8, q = 0.95$**. The results are an average of** 100 **independent runs.**

The next experiment evaluates the accuracy of the random sampling and quantile estimation algorithm with respect to the number of peers - $100, 200$ and 500. In each of these cases, the quantile of the population to monitor was set at 80%, and the confidence level was fixed to 95%. Figure 6 shows similar results that in all the three cases the average ordinal thresholds are greater than the actual quantiles of the population. Note that as we increase the size of the network the scalar product between any two peers decreases because the original data set is now divided into more partitions and hence each profile vector becomes more sparse.

7.3.2 *Private Scalar Product Computation*

This experiments measures the complexity of private scalar product protocol in terms of running time. We generated vectors with size from 500 to 2000. 85% of the vector entries were set to zeros, and the remaining ones took values uniformly distributed in $[0, 100]$. We did this because most peers only visited a small subset of the domains, which made the profile vectors very sparse. Moreover, the complexity to encrypt zero is the same as to encrypt any other integers. As mentioned before, we implemented Paillier's cryptosystem in Java JDK 1.5, and conducted the experiments on a dual-processor workstation running Windows XP with 3.00GHz

[4] DDMT - http://www.umbc.edu/ddm/wiki/software/DDMT/

[5] JADE - http://jade.tilab.com/

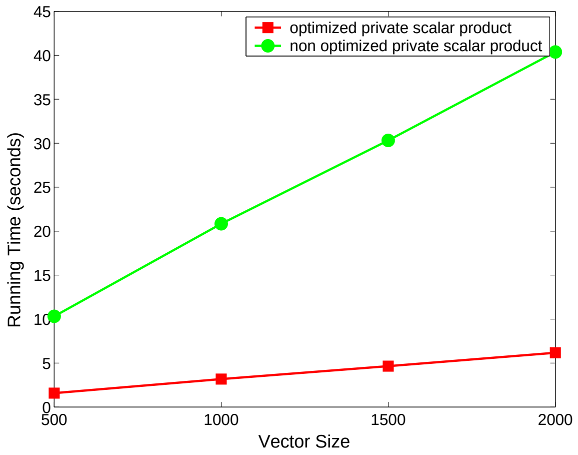


Figure 7: Time required to compute private scalar product with varying dimension of the vector. The results are an average of 10 independent runs.

and 2.99GHz Xeon CPUs and 3.00GB RAM. We tested the baseline protocol as well as its optimized version, *i.e.* precomputing the encryption of 0. Figure 7 shows the running time to compute a single scalar product of vectors of varying dimension both with and without the optimization. It can be seen that there is a great reduction in the running time with optimization.

Using the private scalar product protocol and order statistics theory, we can compute the threshold for quantile estimation. Table 1 presents the running time of threshold computation with fixed confidence level 95% and varying p.

Quantile p	Time (in secs)
0.80	35.04
0.85	50.40
0.90	75.00

Table 1: Time required to compute the threshold w.r.t. different population quantile.

7.3.3 Community Formation

Once the threshold is detected, the next step is to form the communities. We experimented with two community formation schemes. One is without community expansion and one is with expansion. The size of the network was fixed to be 100. Table 2 shows the average number of members found by a community initiator and the (simulated) running time with respect to different TTL values. Table 3 presents the results using the community expansion scheme. The interesting thing to note here is that peers can find more “similar” peers without a significant increase in running time. Note that in these simulations, we didn’t consider the network latency. Since the private scalar product protocol is the most expensive computation, the increase in running time due to network delay in the actual scenario would be insignificant. As a future work, we would like to conduct the experiment in a real distributed environment, and evaluate the performance of our framework.

8. CONCLUSIONS

TTL	Ave Num of Community Members	Time (in secs)
3	3	55.00
4	8	77.50
8	13	173.00

Table 2: Average number of community members found by the initiator without community expansion.

TTL	Ave Num of Community Members	Time (in secs)
3	7	59.00
4	12	82.50
8	17	179.00

Table 3: Average number of community members found by the initiator with community expansion.

In this paper we have presented a framework for forming interest-based communities in a Peer-to-Peer environment. We proposed an order statistics-based approach to quantify similarities between peers, and to build hierarchical communities. We also adopted cryptographic protocols to measure similarity between peers without compromising the privacy of their personal profiles. We have conducted simulations on our distributed data mining platform, and the experimental results show that our algorithm can effectively build similar interests-based communities.

As a future work, we are trying to develop a framework whereby the need for pre-specification of the global ontology is not required. For example, each peer can claim a set of interests and we can compute the set intersection size as a similarity index. In this case, no global set of ontology is required. Further we can explore a secure set intersection protocol to protect privacy. We are actively working to extend this work where not only the web domain names are used for peer’s profile but also the contents of the web pages. We also plan to run experiments on a real-life peer-to-peer environment where we can simulate with tens of thousands of peers using a large-scale data and measure the actual network latency and message overhead.

Acknowledgment

H. Kargupta acknowledges the support from the United States National Science Foundation CAREER award IIS-0093353.

9. REFERENCES

[1] A. Agostini and G. Moro. Identification of communities of peers by trust and reputation. In *Proceedings of the 11th International Conference on Artifical Intelligence: Methogology, Systems and Applications (AIMSA’04)*, volume 3192 of *Lecture Notes in Computer Science*, pages 85–95, Varna, Bulgaria, September 2004. Springer Berlin / Heidelberg.

[2] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, and S. Yi. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *Proceedings of the 22 nd International Conference on Distributed Computing*

Systems (ICDCS'02), pages 74–83, Vienna, Austria, June July.

[3] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama. Distributed uniform sampling in unstructured peer-to-peer networks. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 9, page 223c, Kauai, Hawaii, January 2006.

[4] S. Castano and S. Montanelli. Semantic self-formation of communities of peers. In *Proceedings of the ESWC Workshop on Ontologies in Peer-to-Peer Communities*, Heraklion, Greece, May 2005.

[5] S. Datta, C. Giannella, and H. Kargupta. K-Means Clustering over a Large, Dynamic Network. In *Proceedings of 2006 SIAM Conference on Data Mining*, Bethesda, MD, April 2006.

[6] H. A. David. *Order Statistics*. Wiley-Interscience, 2 edition, 1981.

[7] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of 2004 SIAM International Conference on Data Mining (SDM'04)*, Lake Buena Vista, FL, April 2004.

[8] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self organization and identification of web communities. *IEEE Computer*, 35(3):66–71, March 2002.

[9] Freenet. http://freenetproject.org/.

[10] B. Gilburd, A. Schuster, and R. Wolff. k-ttp: a new privacy model for large-scale distributed environments. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 563–568, Seattle, WA, 2004.

[11] GNUnet. http://gnunet.org/.

[12] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the The 7th Annual International Conference in Information Security and Cryptology (ICISC 2004)*, volume 3506 of *Lecture Notes in Computer Science, Springer-Verlag*, pages 104–120, Seoul, Korea, December 2004.

[13] W. K. Hastings. Monte carlo sampling methods using markov chains and their application. *Biometrika*, 57:97–109, 1970.

[14] K. Horowitz and D. Malkhi. Estimating network size from local information. *The Information Processing Letters Journal*, 88(5):237–243, December 2003.

[15] M. Khambatti, K. D. Ryu, and P. Dasgupta. Efficient discovery of implicitly formed peer-to-peer communities. *International Journal of Parallel and Distributed Systems and Networks*, 5(4):155–164, 2002.

[16] L. Lovász. Random walks on graphs: A survey. *Combinatorics*, Paul Erdös is Eighty, 2:1–46, 1993.

[17] A. H. Maslow. *Motivation and Personality*. HarperCollins Publishers, 3rd edition, January 1987.

[18] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[19] N. Noy. Semantic integration: A survey of ontology-based approaches. *ACM SIGMOD Record*, 33(4):65–70, 2004.

[20] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology - RUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, 1999.

[21] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN'02)*, San Jose, cA, January 2002.

[22] J. P. Scott. *Social Network Analysis: A Handbook*. Sage Publications Ltd., 2nd edition, March 2000.

[23] F. Tanudjaja and L. Mui. Persona: A contextualized and personalized web search. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, volume 3, page 53, Big Island, Hawaii, 2002.

[24] C. Thomas and G. Fischer. Using agents to personalize the web. In *Proceedings of the 2nd International Conference on Intelligent User Interfaces*, pages 53–60, Orlando, FL, 1997.

[25] J. Trajkova and S. Gauch. Improving ontology-based user profiles. In *Proceedings of RIAO*, pages 380–389, Vaucluse, France, April 2004.

[26] J. Vaidya, C. Clifton, and M. Zhu. *Privacy Preserving Data Mining*, volume 19 of *Series: Advances in Information Security*. Springer, 2006.

[27] Y. Wang and J. Vassileva. Trust-based community formation in peer-to-peer file sharing networks. In *Proceedings IEEE International Conference on Web Intelligence (WI'04)*, pages 341–338, Beijing, China, October 2004.

[28] R. Wolff, K. Bhaduri, and H. Kargupta. Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems. In *Proceedings of 2006 SIAM Conference on Data Mining*, Bethesda, MD, April 2006.

[29] R. Wolff and A. Schuster. Association Rule Mining in Peer-to-Peer Systems . In *Proceedings of the third IEEE International Conference on Data Mining*, Melbourne, FL, November 2003.

[30] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 713–718. ACM Press, 2004.

[31] A. C. Yao. How to generate and exchange secrets. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.