On the Feasibility of Common-Friend Measurements for the Distributed Online Social Networks

Yongquan Fu and Yijie Wang
National Key Laboratory for Parallel and Distributed Processing, College of Computer Science
National University of Defense Technology
Hunan province, China, 410073
Email: yongquanf@nudt.edu.cn, wangyijie@nudt.edu.cn

***Abstract*—Distributed social networks have emerged recently. Nevertheless, recommending friends in the distributed social networks has not been exploited fully. We propose FDist, a distributed common-friend estimation scheme that estimates the number of common-friends between any pair of users without disclosing the friends' information. FDist uses privacy-preserving common-friend measurements to collect a small number of common-friend samples, and uses low-dimensional coordinates to estimate the number of common friends to other users. Simulation results on real-world social networks confirm that FDist is both scalable and accurate.**

I. Introduction

Online social networks such as Facebook, LinkedIn have become quite popular these days. However, there are also hot debates over the privacy protection of these centralized social services. For example, the service providers can peek at users' profiles at will for targeted advertisements or even sell these sensitive information to third parties for profits. As a result, there has been strong urges to improve the privacy protection of social networks.

Fortunately, borrowing the success of the P2P systems, the distributed online social networks (DOSN for short) e.g., Safebook [1], LotusNet [2], Cuckoo [3], diaspora [4], Peerson [5], Vis-a-Vis [6] have been proposed to better protect users' privacy based on decentralized infrastructures. The key idea is to store personal data on decentralized nodes and enforce strict access polices on who can visit a user's profile.

To increase the popularity of distributed online social networks, an open question is how to find potential friends for users similar to the friend recommendation on Facebook, LinkedIn. The centralized online social networks are able to compute the possible friends using complete knowledge of users' profiles. One of the most popular recommendation metric [7] is based on the Number of Common Friends (NCF for short), which has been shown to be very effective to recommend new friends or find old friends.

Measuring NCF in the distributed online social networks is however, a difficult task. First, since disclosing personal information such as friend lists to non-friend users violates the goal of the privacy protection. To the contrary, the friend lists should be well protected against curious users. Second, since most end hosts have limited bandwidth capacity, the NCF computation should also scale well with increasing number of friend lists.

Talash [8] computes the common friends based on exchanging friend lists between two users, which not only leaks personal information, but also does not scale well. The PSI approach [9]–[12] represents a list of friends using coefficients of a polynomial, exchanges Homomorphic encrypted coefficients and computes the sum of coefficients to obtain common friends of two friend lists. The PSI approach improves the privacy protection, however at the expense of increasing transmission bandwidth cost and computation cost.

We propose a scalable and accurate distributed NCF estimation method called FDist (*F*riend *Dist*ance measurement). FDist provides NCF estimation between any user pair on a distributed online social networks, but avoids leaking friend lists of a user A to non-friend users.

Since each user on a distributed online social network is often assigned a random string that is created by cryptographic hash functions like SHA-1, estimating the random string of a user is computationally difficult. As a result, to keep the privacy of friend lists, FDist represents the friend lists with well-known Bloom filters that hides the friend identifiers with a bit array. Therefore, a curious user is unable to enumerate the friends efficiently, but a user can query the common friends using the other user's Bloom filter.

Unfortunately, exchanging the Bloom filters may need several-KBytes bandwidth cost in order to control the false positives of the Bloom filter. To further increase the scalability, we propose to estimate the NCF values with low-dimensional coordinates. As a result, *the transmission bandwidth cost is fixed to be the size of coordinates, which is independent of the size of friend lists*.

This work is supported by the National Grand Fundamental Research 973 Program of China under Grant No.2011CB302601; the National Natural Science Foundation of China under Grant No.60873215; the Natural Science Foundation for Distinguished Young Scholars of Hunan Province under Grant No. S2010J5050; Specialized Research Fund for the Doctoral Program of Higher Education under Grant No.200899980003; the National High Technology Research and Development Program of China (863 Program) under Grant No.2011AA01A202.

978-1-4673-2699-5/12/$31.00 © 2012 IEEE

To obtain accurate coordinates, we model the NCF value using a fully decentralized matrix factorization model that adapts the sparseness of common friends, since only two-hop users have non-zero NCF values. The matrix factorization is implemented as a decentralized iterative optimization process. Each user periodically updates his (or her) coordinate based on a distributed conjugate gradient optimization process that converges quickly. Finally, extensive simulation on real-world social network topologies show that FDist provides accurate estimation with better scalability than state-of-art methods.

The rest of the paper is organized as follows. Section II summarizes related work. Section III introduces the background information. Section IV presents how to probe the NCF value to a user. Section V presents how to estimate NCF values to unknown users. Section VI presents the simulation results. Section VII concludes the paper.

II. Related Work

Talash [8]calculates the common friends by exchanging two friend lists, which however, does not scale well with increasing friends; meanwhile, Talash also discloses sensitive personal information to unknown entities.

The Private Set Intersection (PSI) or private matching method [9]–[12] computes the intersection of two sets with privacy protection. Generally, each set is represented by a list of coefficients of a polynomial. Estimating the intersection of two sets is implemented as the arithmetic operations on the coefficients of two polynomials. To improve the privacy protection, the homomorphic encryption is enforced over the arithmetic operations in order to hide the coefficients to the other user. Unfortunately, the PSI methods need a large amount of bandwidth cost and computation cost, which renders them to be less practical for bandwidth-limited end hosts.

Song et al. [13] estimate diverse proximity metrics between users through a matrix factorization process that is computed in a centralized manner, which is impractical for distributed online social networks. Dong et al. [14] propose a secure and privacy-preserving dot product protocol for static coordinates computed by [13], in order to estimate the social proximity in mobile social networks.

III. Background

A. Distributed Online Social Network

Let a **user** be an online entity that participates in the DOSN. A **friend** B of a user A is a user B that establishes the **social link** or **friendship link** with user A on the DOSN. A social link means that users A and B can access the profiles and friend lists of each other. Users and social links altogether lead to a **social graph**. The number of **hops** between users A, B on the DOSN is the shortest path length between users A and B on the social graph. The **common friends** S_{AB} of two users A, B denote the subset of users that are friends to user A and B at the same time. Finally, each user A has a **friend list** S_A of a set of friends of user A and a **profile** that represents a set of personal records such as the personal status, or personal interests, etc.

B. Adversary Model

The goal of the privacy preservation is *to hide a user's friend list from non-friend users*. On the other hand, each user is able to request the friend list of his (or her) friend, since each user trusts his (or her) friends and allows friends to visit the friend list.

We assume that users are **semi-honest** [15]: users may be curious to learn the friend lists of users on other users, but do not claim fake friends about the friend lists of stored users. Particularly, each user follows the common-friend measurement protocol, but is able to eavesdrop on the logical communication links connecting that user.

C. Assumptions

Our key assumption is that the DOSN has already computes a system-wide unique key called a **identifier**, since the DOSN relies on a Peer-to-Peer substrate providing such identifiers to route messages among users. Particularly, the identifier is a randomized string (e.g., 160 bits by default) created by a cryptographic hash function like SHA-1.

The identifier can be regarded as a perfectly random variable and a curious user is unable to guess the identifier of a user by a brute-force enumeration. In other words, these identifiers decouple the DOSN from the real-world identity of a person, and more importantly, are robust against dictionary attacks by curious users. This is because the space of the identifiers is quite huge, for example, there are 2^{160} kind of possible strings for 160-bit identifiers, which prohibits the enumeration task due to the overwhelming computation overhead.

D. Bloom Filter

A Bloom filter is a probabilistic data structure for the set representation. Given a set S of n elements, a standard Bloom filter $BF(S)$ represents S with a bit array I of the length m. Each bit $I[i]$ is initialized to be zero for $i \in [1, m]$. When we **insert** an element y in to the set S, we use k independent hash functions $h_1, \ldots, h_k$ to map the element y into k random numbers within the interval $[1, m]$, such that each bit $I[h_i[y]]$ is set to 1 for $i \in [1, k]$. The Bloom filter supports the **query** "is $y \in S$?" by testing whether each bit $I[h_i[y]]$ is 1 for $i \in [1, k]$: if so, then y is assumed to be in the set, otherwise not. A Bloom filter may incur the **false positive** problem: if the k bits for the element z not in the set S have already been set by other elements in the set S, then the Bloom filter always returns that z is in the set S.

After we insert n elements into the set S, the false positive probability for an element not in S can be asymptotically computed by assuming that the hash functions are perfectly random [16]. Let p be the probability that a random bit in the Bloom filter is 0, then $p = (1 - 1/m)^{n \times k} \approx e^{-nk/m}$ [16]. Consequently, the false positive of the Bloom filter becomes:

$$FP = (1 - p)^k \approx (1 - e^{-nk/m})^k \quad (1)$$

TABLE I
NOTATIONS AND THEIR MEANINGS.

k	the number of hash functions in a Bloom filter
m	the length of a Bloom filter
N	the number of users on the DOSN
$\widehat{X}$	the coordinate distance matrix
Y	the pairwise NCF matrix between a set of users
L	the maximal NCF
d	the coordinate dimension
θ	the NCF mapping thresholds

IV. MEASURING NCF BASED ON BLOOM FILTERS

We show how to measure the common friends without disclosing the information of friends by Bloom filters. Table I shows key parameters used in this paper.

A. Bloom Filter based Friend Representation

We use the well-known Bloom filter [16], [17] to represent friends. Suppose that a curious user C obtains a Bloom filter of the other user B, the user C is unable to efficiently enumerate B's friends, since the identifiers are perfectly random strings and thus are resilient against dictionary attacks.

Unfortunately, a Bloom filter may incurs false positives, i.e., reporting a user not in the friend list to be in. However, we can turn this drawback into a way of protecting the privacy: *a user can deny that a user is his (or her) friend, since the Bloom filter could fail due to the false positives*. Decreasing the false positives may increase the accuracy of estimation, but still fail to distinguish whether a user is in the Bloom filter with 100% certainty.

B. Adapting the Bloom Filter for Dynamic Friends

Since users dynamically establish friendships with other users, we need to adjust the size of the Bloom filter. This is because for a static Bloom filter, its false positive rate keeps increasing with increasing friends, i.e., a Bloom filter may report a user that is not in the set to be in the set.

Each user independently maintains his (or her) Bloom filter to control the **false positives**. Suppose that two users A, B represents sets S_A, S_B with Bloom filters F_A, F_B independently. Let τ be the pre-specified false positive threshold. Let n be the number of friends of a user. Let k be the number of hash functions. The minimum size of the Bloom filter m needs to be at least [16], [17]:

$$m_{(n,k,\tau)} = \frac{-nk}{\ln\left(1-\tau^{\frac{1}{k}}\right)} \tag{2}$$

in order to limit the false positive probability to be within the threshold τ.

We therefore adaptively configure the size m of the Bloom filter:

- Let the default size of m be 1 KBytes.
- When the false positive FP of the Bloom filter exceeds τ, we increase the Bloom filter as $\left(m_{(n,k,\tau)} + 1024 * 8\right)$.
- We reconstruct the Bloom filter for the updated friend list with updated size m of the Bloom filter and the number k of hash functions.

We deliberately fix the number k (6 by default) of hash functions, since there is a wide range of the number of hash functions with similar false positive probability.

C. Estimating Common Friends with Bloom Filters

Computing the common friends is straightforward:

- Alice and Bob exchanges their Bloom filters.
- Alice queries Bob's Bloom filter with her friend list, and vice versa. The subset of friends represented in the Bloom filter is returned as the estimated common friends.

Since Bloom filter may introduce false positives, i.e., the estimation may not coincide with the ground-truth common friends. For around 99% of all cases, the Bloom filter is able to predict correct common friends.

V. ESTIMATING NCF WITH DECENTRALIZED COORDINATES

Exchanging Bloom filters may incur several KBytes due to the increasing number of friends. We show how to estimate the NCF between any pair of users using decentralized coordinates for scalability. Each user is assigned a low-dimensional coordinate to each user. The NCF is calculated as the coordinate distance between corresponding users. Estimating NCF values with decentralized coordinates has at least two desirable advantages:

- Scalability. Transmitting the coordinates incurs a fixed bandwidth cost, which is independent of the size of friend lists.
- Privacy-protection. Exchanging coordinates only reveals the NCF value between two users without the information of the friend lists. Moreover, since the decentralized coordinates keep evolving, it is difficult for curious users to track other users' coordinates.

A. Coordinate Error Function

We first define an **error function** that measures the difference between coordinate distances and the ground-truth NCF values. The optimal solution of the error function provides the accurate classification of coordinate coordinates for NCF mappings.

The challenge is that since the ground-truth NCF value Y_{ij} are ordinal integers, i.e., $Y_{ij} \in \{0, 1, 2, \ldots, L\}$, we have to accurately map the coordinate distance $\widehat{X}_{ij}$ into ordinal numbers.

Fortunately, we can use thresholds to represent the ordinal numbers using continuous-range coordinate distances. For example, if we define two thresholds values $-0.2, 0.3$, and the input coordinate distance $\widehat{X}_{ij} = 0.2$, then we map $\widehat{X}_{ij}$ to the NCF value 2, because 0.2 is only larger than the first threshold value -0.2.

Suppose that the maximum number of common friends between any user pair is L. We introduce L real-valued

threshold variables $\theta_i = \left(\theta_{i1}, \ldots, \theta_{i(L)}\right)$ for each user i as the thresholds of mapping coordinate distances to NCF values.

To ensure an accurate NCF classification, we use the soft-margin constraint. First, we introduce a relaxed range $\theta_{i(Y_{ij}-1)} + 1 \leq \widehat{X}_{ij} \leq \theta_{iY_{ij}} - 1$ that generalizes the hard-margin constraints for classifying binary labels (it also follows that $\theta_{i(Y_{ij}-1)} < \widehat{X}_{ij} < \theta_{iY_{ij}}$), where $\theta_{i0} = -\infty, \theta_{iL} = +\infty$. Second, we use the soft-margin loss function $h(z) = \max(0, 1-z)$ that is popular in SVMs to represent the above constraints as $h\left(\widehat{X}_{ij} - \theta_{i(Y_{ij}-1)}\right) + h\left(\theta_{iY_{ij}} - \widehat{X}_{ij}\right)$.

Furthermore, by penalizing all violations of threshold constraints, i.e.,

- $\widehat{X}_{ij} \geq \theta_{ir} + 1, r < Y_{ij}$;
- $\widehat{X}_{ij} \leq \theta_{ir} - 1, r \geq Y_{ij}$

for robustness, the coordinate error function is

$$f\left(Y_{ij}, \hat{X}_{ij}\right) = \sum_{r=1}^{Y_{ij}-1} h\left(\hat{X}_{ij} - \theta_{ir}\right) + \sum_{r=Y_{ij}}^{L} h\left(\theta_{ir} - \hat{X}_{ij}\right)$$
$$= \sum_{r=1}^{L} h\left(T_{ij}^{r}\left[r, Y_{ij}\right] \cdot \left(\theta_{ir} - \hat{X}_{ij}\right)\right) \quad (3)$$

where $T_{ij}^{r}\left[r, Y_{ij}\right]$ unifies the sums of losses that penalize all violations of threshold constraints:

$$T_{ij}^{r}\left[r, Y_{ij}\right] = \begin{cases} +1 & r \geq Y_{ij} \\ -1 & r < Y_{ij} \end{cases}$$

B. Coordinate Structure

We next introduce the structure of the coordinates. For generalization purpose, we use the popular matrix factorization approach to represent the coordinate matrix $\widehat{X}$.

The matrix factorization uses the inner product of two low-dimensional vectors to denote each coordinate distance. Specifically, the coordinate distance matrix $\widehat{X}$ is represented by a linear combination of two low-rank matrices: $\widehat{X} = U \times V$, where U is a $N \times d$ matrix, V is a $d \times N$ matrix, and $d \ll N$.

Since each row vector of $\widehat{X}$ is represented by the inner product of vectors from U and V, i.e., $\widehat{X}_{ij} = \sum_{m=1}^{d} u_{im} v_{mj}$, we let the coordinate for user i be $\left(\vec{u_i}, \vec{v_i}, \vec{\theta_i}\right)$, where $\vec{u_i}$ denotes i-th row vector of U, $\vec{v_i}$ denotes the i-th column vector of V, $\vec{\theta_i}$ represents the threshold vector for NCF mappings.

The dimensions of $\vec{u_i}$ and $\vec{v_i}$ are both d, and the dimension of $\vec{\theta_i}$ is L. Then the coordinate distance $\widehat{X}_{ij}$ is determined by $\vec{u_i}$ and $\vec{v_j}$.

C. Optimization Objective Function

We could direct optimize the error function (3) to obtain a set of coordinates. Unfortunately, it is impossible to find optimal coordinates that can match all observed NCF values, which called an **overfitting**. This is because the NCF metric is not completely low-rank. Therefore, it is important to prevent an overfitting for the error function for robustness.

To do that, we add a regularization item into the error function (3) that is the sum of the **Frobenius norm** of coordinates. Now, the updated **coordinate optimization objective** becomes:

$$J(u, v, \theta) = \sum_{r=1}^{L} \sum_{(i,j)\in\Omega} h\left(T_{ij}^{r}\left[r, Y_{ij}\right] \cdot \left(\theta_{ir} - \vec{u_i}\vec{v_j}\right)\right) + \frac{\lambda}{2}\left(\sum_{i=1}^{N}\left(\left\|\vec{u_i}\right\|_F^2 + \left\|\vec{v_i}\right\|_F^2\right)\right) \quad (4)$$

where Ω represents the set of observed NCFs, $\|x\|_F^2 = \sum_{i=1}^{m} x_i^2$ denotes the Frobenius norm and λ is a regularized constant.

The objective function (4) needs complete NCF values, which requires a centralized computation process that is infeasible for the distributed environments. Furthermore, each user i can only measure NCFs to a small number of users (denoted as S_i) due to bandwidth cost.

Therefore, we need to reformulate (4) as a distributed optimization objective. Specifically, we decompose (4) into N sub-objective consisting of each user i and its neighbors S_i:

$$J_D\left(\vec{u_i}, \vec{v_i}, \vec{\theta_i}\right) = \sum_{r=1}^{L} \sum_{j\in S_i} h\left(T_{ij}^{r}\left[r, Y_{ij}\right] \cdot \left(\theta_{ir} - \vec{u_i}\vec{v_j}\right)\right) + \frac{\lambda}{2}\left(\left\|\vec{u_i}\right\|_F^2 + \left\|\vec{v_i}\right\|_F^2\right) \quad (5)$$

Now, each user i can optimize (5) to find its coordinate using NCF measurements to its neighbors S_i and the coordinates of S_i.

D. Distributed Algorithm

We propose a scalable distributed algorithm to adjust the coordinate in an incremental manner. Each node maintains a low-dimensional coordinate by minimizing the objective (5) through a nonlinear conjugate gradient optimization, which finds the local minimum of the objective that is known to be robust against local minima [18].

Each user i maintains its own coordinate in rounds independently, starting at randomized positions. Algorithm 1 shows the distributed coordinate update procedure.

The procedure constructs the vector x_i as the concatenation of user i's coordinate, then calculates the steepest direction Δ of x_i based on measurements to neighbors. Next we calculate the new conjugate direction Λ and step length α_i. Then we update the new steepest direction Δx_i and the conjugate direction Λx_i. Finally, we move the vector x_i at a new step and we reconstruct user i's coordinate from vector x_i.

VI. Evaluation

Our evaluation tries to ask whether FDist can estimate the number of common friends accurately and scalably: (1) How does the Bloom filter based NCF estimation method scales for real-world social networks? (2) Is the decentralized coordinate more accurate than existing NCF estimation methods? (3) How does FDist scales for real-world social networks.

We represent each user with a 160-bit randomized string generated with a SHA-1 cryptographic hash function. All experiments are performed on a laptop with 2.13 GHz CPU

Algorithm 1 Update the coordinate in rounds.

▷ Input: user i's current coordinate $\vec{u_i}$, $\vec{v_i}$, $\vec{\theta_i}$, user i's steepest direction Δx_i, user i's conjugate direction Λx_i, the set of neighbors S_i, the NCF values from user i to its neighbors in S_i, the coordinates of neighbors in S_i.

▷ Output: user i's updated coordinate $\vec{u_i}$, $\vec{v_i}$, $\vec{\theta_i}$, user i's updated steepest direction Δx_i, user i's updated conjugate direction Λx_i

$x_i \leftarrow \left[\vec{u_i}; \vec{v_i}; \vec{\theta_i}\right]$; ▷ sequent concatenation of the coordinate

Compute the partial gradient

$$\frac{\partial J_D}{\partial u_{ih}} \leftarrow \lambda u_{ih} - \sum_{j \in S_i} \sum_{r=1}^{L} T_{ij}^r [r, Y_{ij}] \cdot h' \left(T_{ij}^r [r, Y_{ij}] \cdot (\theta_{ir} - \vec{u_i} \vec{v_j}) \right) v_{jh}$$

Compute the partial gradient

$$\frac{\partial J_D}{\partial v_{ih}} \leftarrow \lambda v_{ih} - \sum_{j \in S_i} \sum_{r=1}^{L} T_{ji}^r [r, Y_{ji}] \cdot h' \left(T_{ji}^r [r, Y_{ji}] \cdot (\theta_{jr} - \vec{u_j} \vec{v_i}) \right) u_{jh}$$

Compute the partial gradient $\frac{\partial J_D}{\partial \theta_{ir}} \leftarrow \sum_{j \in S_i} T_{ij}^r [r, Y_{ij}] \cdot h' \left(T_{ij}^r [r, Y_{ij}] \cdot (\theta_{ir} - \vec{u_i} \vec{v_j}) \right)$

$\nabla_x J_D(x_i) \leftarrow \left[\frac{\partial J_D}{\partial u}; \frac{\partial J_D}{\partial v}; \frac{\partial J_D}{\partial \theta}\right]$; ▷ gradient vector

$\Delta \leftarrow -\nabla_x J_D(x_i)$; ▷ steepest direction

$\beta \leftarrow \frac{\Delta^T (\Delta - \Delta x_i)}{\Delta x_i^T \Delta x_i}$;

$\Lambda \leftarrow \Delta + \beta \Lambda x_i$; ▷ conjugate direction

$\alpha_i \leftarrow \arg\min_{\alpha_i} J_D(x_i + \alpha_i \Lambda)$; ▷ optimal movement [18]

$x_i \leftarrow x_i + \alpha_i \Lambda$; ▷ move coordinate at a small step

$\Delta x_i \leftarrow \Delta$; ▷ update the steepest direction

$\Lambda x_i \leftarrow \Lambda$; ▷ update the conjugate direction

$\vec{u_i} \leftarrow x_i[1:d]$; ▷ reconstruct the coordinate

$\vec{v_i} \leftarrow x_i[(d+1):2d]$;

$\vec{\theta_i} \leftarrow x_i[(2d+1):(2d+L)]$;

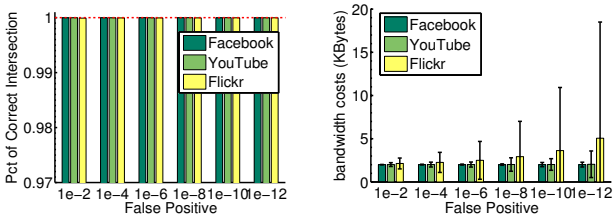
and a 2GB RAM. We implemented all related methods in Matlab 7.0. The experiments are repeated in ten times and we report the average results and the corresponding standard deviations.

We choose three representative social graphs that are all collected by the online social networks research group in the Max Planck Institute for Software Systems [19]–[21].

A. Bloom Filter Based Measurements

We compute the **Percentage of correct intersection** as the percentage of correct NCF results by the Bloom filter based NCF probing process. We compute the **bandwidth cost** of transmitting the Bloom filter from one node to the other node. The number k of hash functions is set to 6 by default. Due to space limitation, we do not report results for varying k.

We test whether there exists a suitable false positive threshold τ for the Bloom filter, such that the NCF probing results are accurate and the probing bandwidth is also modest. Fig 1 show the dynamics of the accuracy and bandwidth cost by tuning the false positive threshold τ of the Bloom filters. We can see that most NCF results are accurate as the threshold τ decreases. For example, when τ is no larger than 10^{-6}, more than 99% of all probing results are accurate, and further accuracy improvement is negligible. On the other hand, decreasing



(a) Accuracy (b) Bandwidth

Fig. 1. Accuracy and bandwidth cost for the NCF estimation by Bloom filters.

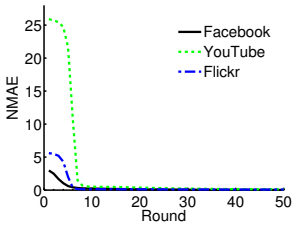


Fig. 2. Convergence of FDist on three data sets.

τ does not significantly increase the mean bandwidth cost, but increases the differences of probing bandwidth for different users, which causes imbalance probing bandwidth cost. As a result, we choose the false positive threshold τ to be 10^{-6} to trade off the bandwidth cost and the probing accuracy.

B. Estimation Accuracy

To quantify the accuracy of estimations, we use the popular metric **Normalized Mean Average Error** (NMAE), which is defined as

$$\frac{\sum_{(i,j):Y_{ij}>0} \left| Y_{ij} - \widehat{Y}_{ij} \right|}{\sum_{(i,j):Y_{ij}>0} Y_{ij}} \tag{6}$$

where Y_{ij} is the ground-truth value, $\widehat{Y}_{ij}$ is the estimated value. Smaller NMAE values correspond to higher prediction accuracy. We report the average results that are based on ten repeated simulations.

FDist is configured as follows: The regularized parameter λ in (5) is 0.3. The coordinate dimension is set to 6. The maximum number of logical neighbors is set to 32.

(1) Convergence. We first test the convergence of the NCF estimations. Fig 2 shows the dynamics of estimation accuracy with increasing rounds of coordinate updates. The estimation accuracy is quickly improved as the coordinate update rounds increase. The coordinates have converged to stable positions after ten rounds of coordinate updates. The computation time is less than 0.1 seconds, which indicates that the coordinate update process is quite efficient.

(2) Accuracy: We compare FDist with LandmarkMDS [22] that estimates discrete routing hops and the proximity estimation method [13] (denoted as MatrixFac) based on the matrix factorization.

From Fig 3 we can see that FDist outperforms LandmarkMDS and MatrixFac in several times. FDist is also consistently

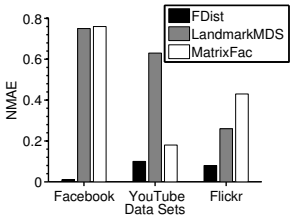


Fig. 3. Comparing the estimation accuracy of different methods on three data sets.

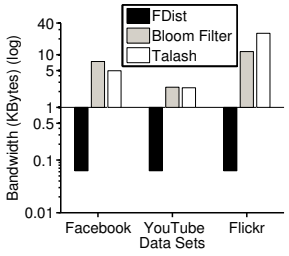


Fig. 4. Comparing the transmission size of different methods on three data sets.

accurate on different social networks. However, we also see that the performance on different data sets varies. This is because different social networks have varying distributions of common friends.

(3) Scalability: We next compare the scalability of different methods in terms of the transmission size. We limit our focus on comparing FDist and Talash. On the other hand, PSI's communication cost is much larger than Talash, since the PSI methods encode each item in the friend list with at least 2048 bits due to the homomorphic encryption. We also compute the transmission cost of exchanging Bloom filters between at most 32 logical neighbors of FDist.

From Fig 4 we can see that FDist incurs very low transmission size, since FDist only needs to exchange the coordinate vectors. However, Talash incurs increasingly large bandwidth from 2KBytes to more than 30 KBytes, which is much less scalable than FDist. On the other hand, exchanging the Bloom filters requires modest bandwidth from 2 KBytes to around 15 KBytes on different social networks. Fortunately, since FDist converges quickly, the bandwidth cost of exchanging Bloom filters can be controlled by increasing the period of updating coordinates.

VII. Conclusion

To estimate common friends in distributed social networks scalably with privacy protection, we propose a distributed common-friends prediction method that combines the privacy-preserving common-friend measurement using Bloom filters and the distributed common-friend estimation with decentralized coordinates. Simulation results show our coordinate based method estimates the NCF values accurately with low transmission cost. We plan to implement our method as a plugin for the social networks such as Facebook, LinkedIn.

References

[1] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: Feasibility of Transitive Cooperation for Privacy on a Decentralized Social Network," in *Proc. of IEEE WOWMOM'09*, pp. 1–6.

[2] L. M. Aiello and G. Ruffo, "LotusNet: Tunable Privacy for Distributed Online Social Network Services," *Comput. Commun.*, vol. 35, pp. 75–88, 2012.

[3] T. Xu, Y. Chen, J. Zhao, and X. Fu, "Cuckoo: Towards Decentralized, Socio-aware Online Microblogging Services and Data Measurements," in *Proc. of HotPlanet '10*, pp. 4:1–4:6.

[4] Diaspora, "Diaspora Alpha," https://joindiaspora.com/, September 2011.

[5] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, "PeerSoN: P2P Social Networking: Early Experiences and Insights," in *Proc. of SNS '09*, 2009, pp. 46–52.

[6] A. Shakimov, H. Lim, R. Cáceres, L. P. Cox, K. A. Li, D. Liu, and A. Varshavsky, "Vis-à-Vis: Privacy-preserving Online Social Networking via Virtual Individual Servers," in *Proc. of COMSNETS'11*, 2011, pp. 1–10.

[7] D. Liben-Nowell and J. Kleinberg, "The Link Prediction Problem for Social Networks," in *Proc. of CIKM '03*, pp. 556–559.

[8] R. Dhekane and B. Vibber, "Talash: Friend Finding In Federated Social Networks," in *Proc. of LDOW2011*.

[9] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient Private Matching and Set Intersection," in *Proc. of EUROCRYPT 2004*, pp. 1–19.

[10] E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of set intersection cardinality," Cryptology ePrint Archive, Report 2011/141, 2011, http://eprint.iacr.org/.

[11] L. Kissner and D. Song, "Privacy-Preserving Set Operations," in *Proc. of CRYPTO 2005*, pp. 241–257.

[12] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient Robust Private Set Intersection," in *Applied Cryptography and Network Security*, 2009, vol. 5536, pp. 125–142.

[13] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu, "Scalable Proximity Estimation and Link Prediction in Online Social Networks," in *Proc. of IMC '09*, pp. 322–335.

[14] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure Friend Discovery in Mobile Social Networks," in *Proc. of IEEE INFOCOM'11*.

[15] O. Goldreich, *Foundations of Cryptography*. Cambridge University Press, 2004, vol. 2: Basic Applications.

[16] A. Z. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," *Internet Mathematics*, vol. 1, no. 4, 2003.

[17] D. Guo, J. Wu, H. Chen, Y. Yuan, and X. Luo, "The Dynamic Bloom Filters," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 1, pp. 120–133, 2010.

[18] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Soc for Industrial & Applied Math, 1996.

[19] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the Evolution of User Interaction in Facebook," in *Proc. of WOSN '09*, pp. 37–42.

[20] M. Cha, A. Mislove, and K. P. Gummadi, "A Measurement-Driven Analysis of Information Propagation in the Flickr Social Network," in *Proc. of WWW '09*, pp. 721–730.

[21] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *Proc. of IMC'07*.

[22] B. Eriksson, P. Barford, and R. D. Nowak, "Estimating Hop Distance Between Arbitrary Host Pairs," in *Proc. of IEEE INFOCOM'09*, pp. 801–809.