Privacy-Preserving Item-Based Collaborative Filtering Using Semi-Distributed Belief Propagation

Jun Zou, Arash Einolghozati, and Faramarz Fekri
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332, USA
Email: {junzou, einolghozati, fekri}@ece.gatech.edu

***Abstract*—Recommender systems are increasingly employed by e-commerce websites to suggest items to users that meet their preferences. Collaborative Filtering (CF), as the most popular recommendation algorithm, exploits the collected historic user ratings to predict ratings on unseen items for users. However, traditional recommender systems are run by the commercial websites, and thus users have to disclose their personal rating data to the websites in order to receive recommendations. This raises the privacy issue, as user ratings can be used to reveal sensitive personal information. In this paper, we propose a privacy-preserving item-based CF recommender system using semi-distributed Belief Propagation (BP), where rating data are stored at the user side. Firstly, we formulate the item similarity computation as a probabilistic inference problem on the factor graph, which can be efficiently solved by applying the BP algorithm. To avoid disclosing user ratings to the server or other user peers, we then introduce a semi-distributed architecture for the BP algorithm, where only probabilistic messages on item similarity are exchanged between the server and users. Finally, an active user locally generates rating predictions by averaging his own ratings on items weighted by their similarities to unseen items. As such, the proposed recommender system preserves user privacy without relying on any privacy techniques, e.g., obfuscation and cryptography. Further, there is no compromise in recommendation performance compared to the centralized counterpart of the proposed algorithm. Through experiments on the MovieLens dataset, we show that the proposed algorithm achieves superior accuracy.**

I. Introduction

The thriving of the Internet and online services has overwhelmed users with an explosive amount of product information, which users never experienced in traditional real-entity consuming activities. It is too exhaustive for users to go through the complete list of thousands of items, e.g., books and movies, to find items interest them. Recommender systems have been widely used in e-commerce websites, such as Amazon.com and Netflix.com, to suggest to users items that they might like. Good recommendation services increase user satisfaction and boost business. Since users have different tastes, it necessitates personalized recommendation services that meet an individual's preferences.

Collaborative Filtering (CF) is so far the most popular recommendation algorithm [1], where users express opinions on items by rating them, and the CF algorithm exploits the collected historic user ratings to predict ratings on unseen items for an individual user, and recommends to the user the items with the highest predicted ratings. The CF recommendation can be divided into user-based and item-based methods. The user-based method recommends to an active user new items favorably rated by other users with similar tastes to the active user [2]. The item-based method on the other hand analyzes the similarity between items using the aggregated user ratings, and recommends to an active user new items that are similar to the items he liked in the past [3]. Closely related to the item-based CF is the content-based recommendation, where each item is characterized by a set of attributes, based on which item similarity is estimated [1]. However, content-based recommendation suffers from limited content-analysis, e.g., it is difficult to explicitly describe multimedia data using features, whereas the CF recommendation is immune to such problems, an important reason for the popularity of the CF recommendation.

In traditional recommender systems, the recommendation algorithm is run at the central server owned by the commercial website, and thus the users have to disclose their personal information, such as preferences, age and gender, to the websites in order to receive satisfactory recommendation services. This raises the privacy issue, as users simply have no control over how their personal data will be disseminated and used [4]. It is not uncommon that websites sell to other parties such data, which are valuable for targeted advertising. As users become more concerned about their online privacy, they are less willing to directly release their personal information. Most recommender systems instead rely on the user ratings, through the CF recommendation.

Although rating data do not directly tell personal details, it is still possible to infer user demographics, such as age and gender, from their ratings [5], and even uncover user identities and reveal sensitive personal information with access to other databases [6]. While users certainly do not want their privacy compromised, they still hope to enjoy the fun and convenience brought by recommender systems. Privacy-preserving recommender systems are thus in urgent need. The challenge stems from the conflict between accuracy and privacy. That is, to provide recommendations that better match the user's tastes, the system needs to know more about the user. Since completely withholding the user data from the recommender system is not favourable, it seems attractive to obfuscate user ratings with random noise, e.g., perturbation [7]

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1115199, and a gift from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation.

and differential privacy [8], or to disguise genuine user profiles by adding extra fake data [5], [9]. However, obfuscation trades accuracy for privacy, and users have to sacrifice more privacy for better recommendation.

An alternative approach is to preserve privacy through user-side storage of personal data [10]. If the central server still participates in generating recommendations, advanced cryptography techniques can be employed to hide user information from the recommender server, which only operates on the encrypted user data [11], [12], but the required key management is difficult to implement in practice. A more aggressive move is to switch to a fully distributed recommender system, where user ratings are only exchanged among users themselves, and recommendation is generated in a distributed fashion without relying on the central server [13]. The only problem is, users may not feel comfortable to share personal data with other users either, which discourages users from participation. Indeed, an attacker can easily mimic the behaviour of genuine users to acquire their personal data. A compromising solution is to apply obfuscation techniques [7] to distributed systems as well [14].

Most privacy-preserving techniques for recommender systems are developed without modifying the recommendation algorithm itself. Rather than design a separate privacy module for the recommendation algorithm, we hope to build a recommender system with intrinsic privacy-preserving properties. To achieve this goal, it is evident that any communications between the server and users or between user peers should avoid exposing user ratings. An interesting work in [15] utilized concordance measure for computing user similarity, a key step in the user-based CF algorithm, where computation is conducted between users without exposure of their ratings to each other, but unfortunately, to collaboratively generate recommendation, users need to reveal ratings to other users. We notice that user-based CF relies on direct collaboration among users, i.e., a user needs to know what other people like to find out what he might like, whereas item-based CF exploits the consistency in an individual user's taste, i.e., if a user likes an item, he might also like other items similar to it. We consider item-based CF a better choice for privacy-preserving recommender systems. The accuracy of the item-based CF system depends on measure of item similarity, which has to be estimated based on ratings on items, and further, subject to the privacy constraint.

In this paper, we introduce a privacy-preserving item-based CF recommender system using Belief Propagation (BP). We formulate the item similarity computation as a probabilistic inference problem on a proper factor graph, which can then be efficiently solved using BP, a probabilistic message passing algorithm [16], [17]. Further, we develop a semi-distributed architecture for BP, where probabilistic messages on item similarity are exchanged between the server and users, without disclosing user ratings to the server or other peer users. There is no direct communication between users. The server decides when to quit BP by checking the convergence of the messages, and computes the item similarity using the observed messages upon convergence. The last piece that completes our privacy-preserving recommender system is user-side recommendation. The user locally generates rating predictions by averaging his own ratings on items weighted according to their similarities to other unseen items. Hence, the proposed item-based CF algorithm preserves user privacy in both item similarity computation and rating prediction, without employing any privacy-protection techniques, e.g., obfuscation and cryptography. In addition, there is no compromise in recommendation performance due to privacy preserving, compared to the centralized counterpart of the proposed algorithm. Through experiments on the MovieLens dataset, we show that our algorithm achieves superior accuracy, in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

The rest of the paper is organized as follows. In Sec. II, we introduce the background on CF recommendation algorithms. In Sec. III, we present the proposed privacy-preserving item-based recommender system in detail. In Sec. IV, we evaluate the recommendation performance of the proposed algorithm on the MovieLens dataset, and compare it against other well-known item-based CF algorithms without considering privacy. In Sec. V, we review the related works on privacy-preserving item-based recommender systems and application of BP to recommender systems. In Sec. VI, we conclude this paper.

II. Background on Collaborative Filtering

Assuming there are M users and N items in a recommender system, let $\mathbb{U} = \{1, \ldots, M\}$ represent the set of all users, and $\mathbb{I} = \{1, \ldots, N\}$ represent the set of all items. A user u expresses his opinion on item i in the form of rating r_{ui}. We arrange the collection of all ratings in an incomplete $M \times N$ matrix $\mathbb{R}$, with r_{ui} at the intersection of u-th row and i-th column. The entries of unknown ratings are unfilled. Let I_u denote the subset of items rated by user u, and U_i denote the subset of users who have rated item i. The task of a recommender system is to predict the ratings for an active user u on the subset of unseen items $\mathbb{I} \backslash I_u$. Throughout this paper, we focus on the item-based CF recommendation algorithm [3].

To predict the rating r_{ui} for user u on an unseen item i, the algorithm sorts the items in I_u according to their similarity to item i in descending order, and finds a subset of top K most similar items, denoted by $\mathcal{N}_{ui}$, where $|\mathcal{N}_{ui}| = K$. We refer to $\mathcal{N}_{ui}$ as the neighbourhood of item i from user u's perspective, and K as the neighborhood size. Then r_{ui} is predicted by

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_{ui}} s_{ij} \times r_{uj}}{\sum_{j \in \mathcal{N}_{ui}} |s_{ij}|}, \tag{1}$$

where s_{ij} is the similarity between items i and j. The accuracy of the algorithm depends on the measure of item similarity s_{ij}, which is computed based on the observed ratings on items. The user-based CF algorithm uses a similar formula to (1), but based on ratings from other users.

Several well-known methods for item similarity computation include Cosine Similarity (CS), Pearson Correlation Similarity (PCS), and Adjusted Cosine Similarity (ACS) [3]. Yet, all of those mentioned methods directly operate on ratings from different users, which requires users to disclose ratings to the server or other users. Next, we will introduce a semi-distributed BP algorithm for item similarity computation with intrinsic privacy-preserving property.

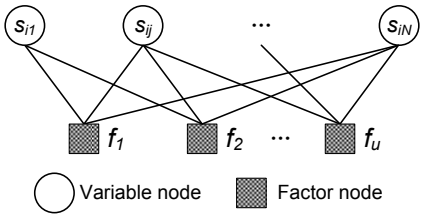


Fig. 1: The factor graph $\mathcal{G}_i$.

III. Proposed Privacy-Preserving Item-Based CF

A. Probabilistic Modelling of Item Similarity

We model the similarity s_{ij} between items i and j as a discrete random variable that takes values from a predefined set $\mathcal{S}$. The total number of possible values is $L = |\mathcal{S}|$. Let $\mathbb{S}_i = \{s_{ij} : 1 \leq j \leq N, j \neq i\}$ be the set of item similarities between item i and other items. We denote by $P(\mathbb{S}_i|\mathbb{R})$ the joint posterior probability distribution of $\mathbb{S}_i$. To obtain s_{ij}, we need the marginal posterior probability distribution of s_{ij}, which can be derived by

$$P(s_{ij}|\mathbb{R}) = \sum_{s_{i1}\in\mathcal{S}} \cdots \sum_{s_{i(j-1)}\in\mathcal{S}} \sum_{s_{i(j+1)}\in\mathcal{S}} \cdots \sum_{s_{iN}\in\mathcal{S}} P(\mathbb{S}_i|\mathbb{R}). \tag{2}$$

For notational convenience, we rewrite (2) for the sum over all variables in $\mathbb{S}_i$ except s_{ij} as

$$P(s_{ij}|\mathbb{R}) = \sum_{\mathbb{S}_i \setminus s_{ij}} P(\mathbb{S}_i|\mathbb{R}). \tag{3}$$

Similar notations are used for this type of sum throughout this paper. However, direct computation using (3) is not appealing, because it must be performed by a central server, and thus users are required to disclose their ratings to the server. Moreover, (3) incurs an exponential complexity of $\mathcal{O}(L^N)$. We instead resort to the factor graph to express the factorization of $P(\mathbb{S}_i|\mathbb{R})$, and apply the efficient sum-product BP algorithm to infer the marginal posterior probability distributions. More importantly, the BP algorithm can be carried out in a semi-distributed manner, so that the part of computation that requires knowledge of user ratings can be locally performed by the user, eliminating the need to disclosing user ratings.

A factor graph is a bipartite graph that expresses the factorization structure of a function, where variable nodes and factor nodes represent variables and local functions, respectively, and an edge connects a variable node to a factor node if and only if the variable is an argument of the local function represented by the factor node [17]. In our concerned problem, to solve for $\mathbb{S}_i$, we first find a proper factorization for $P(\mathbb{S}_i|\mathbb{R})$. For each user $u \in U_i$, we denote $\mathbb{S}_{ui} = \{s_{ij} : j \in I_u \setminus i\}$ as the set of item similarities between item i and other items user u has rated, and use a local function $f_u(\mathbb{S}_{ui})$ to model the dependencies among variables in $\mathbb{S}_{ui}$ from user u's perspective. Hence, $P(\mathbb{S}_i|\mathbb{R})$ factorizes into local functions as follows

$$P(\mathbb{S}_i|\mathbb{R}) = \frac{1}{Z} \prod_{u\in U_i} f_u(\mathbb{S}_{ui}), \tag{4}$$

where Z is a normalization constant. We construct a factor graph $\mathcal{G}_i$ for the factorization in (4) as illustrated in Fig. 1, where there are $|\mathbb{S}_i|$ variable nodes and $|U_i|$ factor nodes. Each $s_{ij} \in \mathbb{S}_i$ is represented by variable node j in $\mathcal{G}_i$, and each local function $f_u(\mathbb{S}_{ui})$ is represented by factor node u. The subset of variable nodes for $\mathbb{S}_{ui}$ are connected to factor node u via edges. Let U_{ij} denote the set of common users of items i and j, $U_{ij} = U_i \cap U_j$. Hence, variable node j is connected to $|U_{ij}|$ factor nodes in $\mathcal{G}_i$. Essentially, if a user u has rated item i and other items in $I_u \setminus i$, then this user has a belief on the similarity s_{ij}, $\forall j \in I_u \setminus i$, from his perspective. The factor graph allows users' beliefs be exchanged and aggregated following the principle of sum-product message passing.

The local function $f_u(\mathbb{S}_{ui})$ determines how user u estimates item similarity based on his own ratings. It should be properly designed with regard to the eventual goal to predict ratings using (1). For a user u who has rated item i, we assume for now rating r_{ui} is unknown, and let $\hat{I}_u = I_u \setminus i$. Given a configuration of item similarities in $\mathbb{S}_{ui}$, user u predicts r_{ui} as

$$\hat{r}_{ui}(\mathbb{S}_{ui}) = \frac{\sum_{j\in\hat{I}_u} s_{ij} \times r_{uj}}{\sum_{j\in\hat{I}_u} |s_{ij}|}. \tag{5}$$

Note that (5) has a similar form to (1). Then user u checks $\hat{r}_{ui}(\mathbb{S}_{ui})$ against the actual rating r_{ui} using the following factor node function

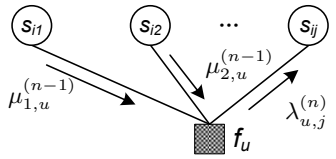
$$f_u(\mathbb{S}_{ui}) = \frac{1}{Z_u} \exp\left\{-\frac{1}{\sigma^2}(\hat{r}_{ui}(\mathbb{S}_{ui}) - r_{ui})^2\right\}, \tag{6}$$

where Z_u is a normalization constant, and σ is a designing parameter that controls the sensitivity of $f_u(\mathbb{S}_{ui})$ to the discrepancy between $\hat{r}_{ui}(\mathbb{S}_{ui})$ and r_{ui}. We note that $f_u(\mathbb{S}_{ui})$ decreases with increasing discrepancy. Now the specification of the factor graph $\mathcal{G}_i$ is completed.

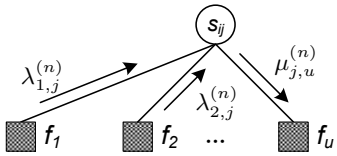
To solve for all item similarities $\{\mathbb{S}_i : 1 \leq i \leq N\}$, a total of N such factor graphs need to be constructed. Note that different from traditional similarity measures such as CS and ACS, our algorithm does not possess the symmetric property of $s_{ij} = s_{ji}, i \neq j$. Indeed, this could be one of the reasons that our algorithm actually achieves better accuracy as we will see in Sec. IV. According to (1), an item's influence on the rating prediction on item i depends on its weight relative to those of other items. In other words, to achieve accurate prediction on item i, the item that is more similar to it should be assigned with a larger weight, and thus it is the relative similarity that matters rather than the absolute similarity. Our algorithm can be considered as a collaborative process among users to find relative similarities. It is not surprising that, while item j can be one of the top items most similar to item i, item i might be perceived as mildly similar to item j compared with other items.

B. BP for Similarity Computation

We first describe the sum-product BP algorithm to infer the marginal posterior distribution $P(s_{ij}|\mathbb{R})$, $\forall s_{ij} \in \mathbb{S}_i$, on the factor graph $\mathcal{G}_i$, without worrying about privacy. Later in Sec. III-D, we will introduce the semi-distributed implementation of BP for privacy, yet with no impacts on the computed similarities. Since the constructed factor graph has loops, we apply the "loopy" BP algorithm that iteratively exchanges messages between factor nodes and variable nodes along the

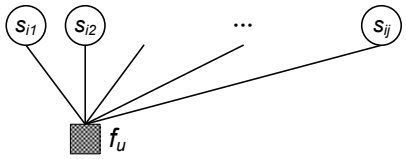


(a) The λ-message $\lambda_{u,j}^{(n)}(s_{ij})$.

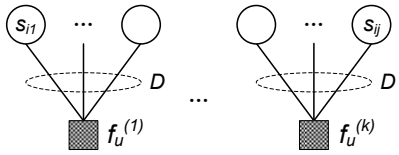


(b) The μ-message $\mu_{j,u}^{(n)}$.

Fig. 2: Illustration of message passing at iteration n.



(a) A high-degree factor node.



(b) Multiple D-degree factor nodes.

Fig. 3: Illustration of complexity reduction via grouping.

edges until convergence [17]. As in the sum-product principle, there are two types of messages passed on $\mathcal{G}_i$: (i) The λ-message $\lambda_{u,j}(s_{ij})$ sent from a factor node u to a variable node j, and (ii) the μ-message $\mu_{j,u}(s_{ij})$ sent from a variable node j to a factor node u.

We illustrate the message passing in Fig. 2. In each iteration, each node (factor node or variable node) in the factor graph generates and sends messages to the neighbor nodes connected to it, based on incoming messages. In iteration n, factor node u generates the λ-message $\lambda_{u,j}^{(n)}(s_{ij})$ sent to variable node j by computing the product of local factor function $f_u(\mathbf{s}_{ui})$ with all μ-messages received in the previous iteration from neighbor variable nodes of factor node u, excluding the message from the recipient variable node j, and sums out all variables except s_{ij} as follows

$$\lambda_{u,j}^{(n)}(s_{ij}) \propto \sum_{\mathbb{S}_{ui} \backslash s_{ij}} f_u(\mathbb{S}_{ui}) \prod_{h \in \hat{I}_u \backslash j} \mu_{h,u}^{(n-1)}(s_{ih}). \tag{7}$$

The λ-message $\lambda_{u,j}^{(n)}(s_{ij})$ is a list of the beliefs on the similarity $s_{ij} = s, \forall s \in \mathcal{S}$, perceived from user u's perspective, given the current collective knowledge of the similarity between other items in $\hat{I}_u \backslash j$ and item i.

Variable node j generates the μ-message $\mu_{j,u}^{(n)}(s_{ij})$ sent to factor node u as the product of all incoming λ-messages received in the current iteration from all factor nodes connected to variable node j, excluding the one from the recipient factor node u as follows

$$\mu_{j,u}^{(n)}(s_{ij}) \propto \prod_{f \in F_j \backslash u} \lambda_{f,j}^{(n)}(s_{ij}), \tag{8}$$

where F_j denotes the set of factor nodes connected to variable node j. Here in graph $\mathcal{G}_i$, $F_j = \{u : u \in U_{ij}\}$. The μ-message $\mu_{j,u}^{(n)}(s_{ij})$ is a list of beliefs on the similarity $s_{ij} = s, \forall s \in \mathcal{S}$, which is generated by aggregating beliefs from other users in $U_j \backslash u$ on the similarity s_{ij}.

After convergence, the marginal posterior distribution $P(s_{ij}|\mathbb{R})$ is computed at variable node j as product of all λ-messages received from neighbor factor nodes connected to variable node j as

$$P(s_{ij}|\mathbb{R}) = \frac{1}{Z_{ij}} \prod_{f \in F_j} \lambda_{f,j}^{(n)}(s_{ij}), \tag{9}$$

where Z_{ij} is a normalization constant. Finally, based on the marginal posterior distribution $P(s_{ij}|\mathbb{R})$, the item similarity s_{ij} can be estimated in various ways. We consider using the minimum mean squared error criterion, for which the optimal estimated s_{ij} is given by the expectation

$$\hat{s}_{ij} = \sum_{s \in \mathcal{S}} s \times P(s_{ij} = s|\mathbb{R}). \tag{10}$$

We summarize the BP algorithm for computing $\mathbb{S}_i$ on factor graph $\mathcal{G}_i$ in Alg. 1.

Algorithm 1 BP on $\mathcal{G}_i$ for computing item similarity $\mathbb{S}_i$.

- Initialize all messages as $\lambda_{u,j}^{(0)}(s_{ij} = s) = \frac{1}{L}$ and $\mu_{j,u}^{(0)}(s_{ij} = s) = \frac{1}{L}$, $\forall s \in \mathcal{S}$, and set iteration counter $n = 1$.
- Iterative message passing until convergence.
  - (a) Update all λ-messages using (7);
  - (b) Update all μ-messages using (8);
  - (c) $n = n + 1$. If not convergent, repeat (a) and (b).
- Compute marginal posterior probability distributions of $s_{ij} \in \mathbb{S}_i$ using (9).
- Compute item similarity $s_{ij} \in \mathbb{S}_i$ using (10).

C. Complexity Reduction

The computational complexity of the BP algorithm is determined by the computation of λ-messages and μ-messages. While the complexity of generating a μ-message using (8) is $\mathcal{O}(|I_u|)$, the complexity of generating a λ-message using (7) is $\mathcal{O}\left(|I_u|L^{|I_u|}\right)$, where $L = |\mathcal{S}|$, which is exponential in the degree of the factor node, i.e., the number of items user u has rated. Unfortunately, in recommender systems, a user can rate over hundred of items, rendering the BP algorithm almost practically infeasible. We thus propose a complexity reduction technique by controlling the degree of the factor node. It is based on the understanding that, user u first randomly divides the items in $\hat{I}_u$ into multiple groups of size D, where D is

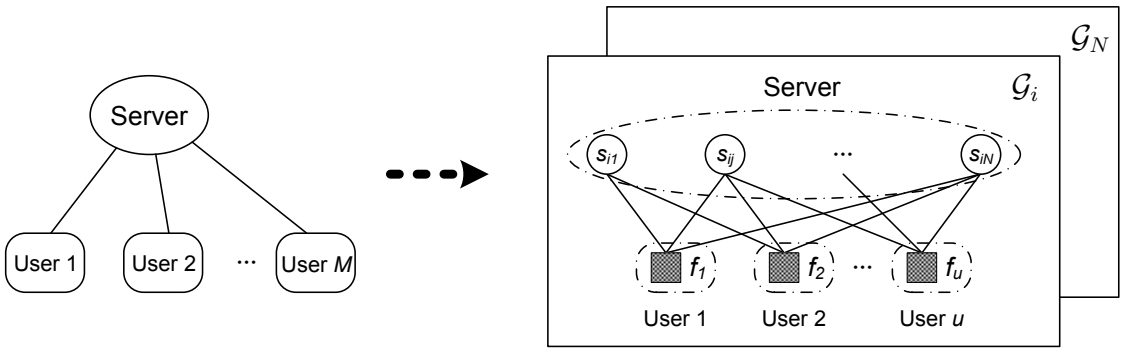


Fig. 4: Architecture of semi-distributed BP.

a small integer, and decides item similarity on the basis of groups, as if items in different groups were rated independently by user u.

Hence, we modify the factor graph $\mathcal{G}_i$ accordingly to obtain a new factor graph $\hat{\mathcal{G}}_i$ for complexity reduction. Since variable node j is associated with item j, each user u groups the variable nodes in $\hat{I}_u$ exactly as the grouping among items. There are $G_u = \lceil |\hat{I}_u|/D \rceil$ such groups from user u's perspective. Let $D_u^{(k)}$ denote the size of group k, $D_u^{(k)} = D$ for $1 \leq k \leq G_u - 1$ and $D_u^{(k)} = |\hat{I}_u| - D(G_u - 1)$ for $k = G_u$. Let $I_u^{(k)}$ denote the variable nodes in group k. We set an indicator $j_u^{(k)} = 1$ if $j \in I_u^{(k)}$ and $j_u^{(k)} = 0$ otherwise. Since each item $j \in \hat{I}_u$ only belongs to one group of user u, we have $\sum_{k=1}^{G_u} j_u^{(k)} = 1$. Instead of connecting all variable nodes in $\hat{I}_u$ to one factor node u, we connect a separate factor node $u^{(k)}$ to the group of variable nodes in $I_u^{(k)}$. We illustrate the complexity reduction at one factor node in Fig. 3.

The local function at factor node $u^{(k)}$ can be similarly derived as (6). Let $\mathbb{S}_{ui}^{(k)} = \{s_{ij} : j \in I_u^{(k)}\}$. Again, assuming r_{ui} on item i is unknown, then using ratings from items within group k, user u predicts r_{ui} as

$$\hat{r}_{ui}(\mathbb{S}_{ui}^{(k)}) = \frac{\sum_{j \in I_u^{(k)}} s_{ij} \times r_{uj}}{\sum_{j \in I_u^{(k)}} |s_{ij}|}. \tag{11}$$

We substitute $\hat{r}_{ui}(\mathbb{S}_{ui})$ with $\hat{r}_{ui}(\mathbb{S}_{ui}^{(k)})$ in (6) to obtain the new local function of factor node $u^{(k)}$

$$f_u^{(k)}\left(\mathbb{S}_{ui}^{(k)}\right) = \frac{1}{Z_u^{(k)}} \exp\left\{-\frac{1}{\sigma^2}\left(\hat{r}_{ui}(\mathbb{S}_{ui}^{(k)}) - r_{ui}\right)^2\right\}, \tag{12}$$

where $Z_u^{(k)}$ is a normalization constant.

On the new factor graph $\hat{\mathcal{G}}_i$, we apply the BP algorithm described in Sec. III-B. The λ-messages and μ-messages are exchanged between the new factor nodes and variable nodes. The λ-message $\lambda_{u^{(k)},j}^{(n)}(s_{ij})$ sent from factor node $u^{(k)}$ to variable node j is given by

$$\lambda_{u^{(k)},j}^{(n)}(s_{ij}) \propto \sum_{\mathbb{S}_{ui}^{(k)} \setminus s_{ij}} f_u(\mathbb{S}_{ui}^{(k)}) \prod_{h \in I_u^{(k)} \setminus j} \mu_{h,u^{(k)}}^{(n-1)}(s_{ih}). \tag{13}$$

And the μ-message $\mu_{j,u^{(k)}}^{(n)}(s_{ij})$ sent from variable node j to factor node $u^{(k)}$ is given by

$$\mu_{j,u^{(k)}}^{(n)}(s_{ij}) \propto \prod_{f \in \hat{F}_j \setminus u^{(k)}} \lambda_{f,j}^{(n)}(s_{ij}), \tag{14}$$

where $\hat{F}_j = \{v^{(k)} : v \in U_{ij},\, j_v^{(k)} = 1,\, 1 \leq k \leq G_v\}$.

The complexity of updating a λ-message is effectively reduced to $\mathcal{O}\left(DL^D\right)$ from $\mathcal{O}\left(|I_u|L^{|I_u|}\right)$ by using (13). Meanwhile, the total number of λ-messages needs to be generated from user u's perspective in each iteration remains $|\hat{I}_u|$, which is the same as in $\mathcal{G}_i$. There is no change in complexity regarding the μ-messages. The overall complexity of the BP algorithm on $\hat{\mathcal{G}}_i$ with complexity reduction in each iteration is $\mathcal{O}\left(\bar{M}\bar{N}DL^D + N\bar{M}^2\right)$, where $\bar{N}$ is the average number of items rated by one user, and $\bar{M}$ is the average number of users of one item. Since the number of items a user can consume is limited by his time and money, we can assume $\bar{N}$ is much smaller than N. As for $\bar{M}$, we assume $\bar{M}$ grows in the order of $M^{1-\epsilon}$, where $0 < \epsilon < 1$. Then we can rewrite the computation complexity on $\hat{\mathcal{G}}_i$ as $\mathcal{O}\left(M^{1-\epsilon}\bar{N}DL^D + NM^{2(1-\epsilon)}\right)$, and when N and M is large, it is dominated by the second term, so we have $\mathcal{O}\left(NM^{2(1-\epsilon)}\right)$.

D. Semi-Distributed Implementation of BP

Now we are ready to introduce the semi-distributed implementation of the proposed BP algorithm for item similarity computation. Our goal is to build a privacy-preserving recommender system. From (7) and (8), we can see that the user rating data are only used in computing $f_u(\mathbb{S}_{ui})$ to generate λ-messages. Hence, the key to preserve user privacy is to compute λ-messages at the user side. The μ-messages can be generated at a central server by performing multiplication operations on received λ-messages. This leads to the semi-distributed implementation of the BP algorithm. As we will see next, the computation follows the BP algorithm described in Sec. III-B and III-C, and thus the semi-distributed BP does not impact the computed results.

The architecture of the semi-distributed BP is shown in Fig. 4. The message passing on graph $\mathcal{G}_i$ is carried out by exchanging messages between the server and the users. In addition, users also do the grouping for complexity reduction as explained in Sec. III-C, but the grouping information is not needed at the server side. Users locally store their personal ratings, and generate λ-messages according to (13) without

TABLE I: The format of λ-messages.

Graph i	User u	Item j	Iteration n	L-entry vector $\overrightarrow{\lambda}$

TABLE II: The format of μ-messages.

Graph i	User u	Item j	Iteration n	L-entry vector $\overrightarrow{\mu}$

disclosing their ratings to the server or other users. User u sends the λ-message $\lambda_{u,j}^{(n)}(s_{ij})$ to the server in the format shown in Table I, where the L-entry vector $\overrightarrow{\lambda}$ stores the values of $\lambda_{u,j}^{(n)}(s_{ij}=s)$, $\forall s \in \mathcal{S}$. Note that the group information "k" is not included in the λ-messages, since only one $\lambda_{u,j}^{(n)}(s_{ij})$ message is generated by user u, regardless of within which group of user u the message is generated. In other words, the grouping step is transparent to the server, and the complexity reduction is for reducing the computational burden of the users. Specifically, the computational complexity on graph $\mathcal{G}_i$ at each user is only $\mathcal{O}\left(\bar{N}DL^D\right)$, regardless of N and M.

The server is responsible for generating μ-messages. To compute the μ-message $\mu_{j,u}^{(n)}(s_{ij})$, the server checks if all λ-messages $\lambda_{u,j}^{(n-1)}(s_{ij})$ from users in U_j have been received. The server then performs multiplication operations on the λ-messages to generate μ-messages according to (8), and sends the μ-message $\mu_{j,u}^{(n)}(s_{ij})$ to user u in the format shown in Table II, where the L-entry vector $\overrightarrow{\mu}$ stores the values of $\mu_{j,u}^{(n)}(s_{ij}=s)$, $\forall s \in \mathcal{S}$. At the user side, user u can easily recover the group information "k" from $\mu_{j,u}^{(n)}(s_{ij})$ by looking up for k with $j_u^{(k)}=1$, and obtain $\mu_{j,u^{(k)}}^{(n)}(s_{ij})$, as if it was computed using (14). The server checks the convergence of messages, and obtains the item similarity after convergence using (10).

The computation of all item similarities in $\{\mathbb{S}_i : 1 \leq i \leq N\}$ requires N factor graphs, with $\mathbb{S}_i$ computed using factor graph $\mathcal{G}_i$. We introduce three protocols for coordinating the message passing on different graphs: the serial protocol, the parallel protocol, and the pipeline protocol. In the serial protocol, the message passing on factor graphs is performed in a serial manner, that is at any time, all generated messages belong to one factor graph, and unless inference on that graph is finished, no messages on other factor graphs are generated. Alternatively, we can adopt the parallel protocol. If the bandwidth of the communication channel between users and the server is sufficient, and if the user's computational capability allows, message passing on multiple factor graphs can be performed in parallel to accelerate the inference process. The pipeline protocol, which is a combination of the serial and parallel protocols, is favorable when the delay in the network is large. While waiting for the messages on one factor graph to arrive, users can compute messages on other factor graphs and continue to send them to the server, so as to increase throughput and make more efficient use of computational resources as well, but the total amount of messages transmitted on the network needs to be carefully controlled to avoid paralyzing the network.

TABLE III: Summarization of entity functions.

Entity	Function
Server	Coordinate message passing; Generate μ-messages; Compute and store item similarity.
User	Store personal rating data; Generate λ-messages; Generate recommendations.

Aside from the semi-distributed implementation, a fully distributed implementation is also possible. For example, [18] proposed a distributed BP-based trust management algorithm for P2P networks. Users directly send λ-messages to other users instead of to the server, and also generate μ-messages locally using received λ-messages, without relying on the server. However, there is a significant increase in communication overhead, considering that a λ-message $\lambda_{u,j}(s_{ij})$ needs to be sent to each of the users in U_j, as they require this λ-message for updating μ-messages. Moreover, for each item a user has rated, he needs to find out other users who also have rated that item, i.e., the graph must be known by the users. Further, since the item similarity is locally computed by users, and user $u \in U_i$ only obtains $\{s_{ij} : j \in \hat{I}_u\}$, the users need to share with each other the item similarities. Thus, a more sophisticated protocol needs to be developed for a fully distributed system.

E. User-Side Recommendation

Thus far, we have focused on the item similarity computation using the semi-distributed BP algorithm. To complete the privacy-preserving item-based CF recommender system, it remains to specify the recommendation generation. As introduced in Sec. II, item-based CF computes rating prediction for user u on item i using (1), which takes as input the past ratings of user u and item similarities. To avoid revealing user ratings, users would then locally generate rating predictions. Since the item similarities are obtained at the server side in the semi-distributed BP algorithm, the server should send to users the required item similarities. To predict ratings on all unseen items in $\mathbb{I}\backslash I_u$, user u only needs item similarities in $\{s_{ij} : i \in \mathbb{I}\backslash I_u, j \in I_u\}$. After computing rating predictions, users can locally store the item similarities received from the server for future uses, and only update them periodically. We summarize the functions of the server and users in Table III.

It is worth noting that in addition to preserving privacy, user-side recommendation also enhances user trust in e-commerce. Traditionally, centralized recommender systems owned by the commercial websites can manipulate the recommendations in various ways for revenue. A website might place the items with the highest profits on top of the recommendation list, or even employ recommendations as tools for advertisement of new products. This trust issue is well addressed by user-side recommendation, where users locally generate recommendations on their personal computers.

F. Trade-offs

Before discussing the trade-offs, we like to emphasize that there is no trade-off between accuracy and privacy in the

TABLE IV: MAE performance comparison of various item-based CF algorithms under different neighborhood size K.

Algorithms	$N_c = 3$ ($P_{\text{pred}} = 74\%$)					$N_c = 8$ ($P_{\text{pred}} = 67\%$)				
	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
PCS	0.8839	0.8486	0.8370	0.8326	0.8418	0.8330	0.8073	0.8009	0.7989	0.8127
ACS	0.7812	0.7585	0.7609	0.8029	0.9278	0.7390	0.7264	0.7389	0.8044	0.9765
CS	0.7567	0.7601	0.7676	0.7751	0.7812	0.7418	0.7428	0.7494	0.7566	0.7632
Proposed	0.7512	0.7437	0.7486	0.7557	0.7632	0.7283	0.7255	0.7293	0.7359	0.7450

TABLE V: RMSE performance comparison of various item-based CF algorithms under different neighborhood size K.

Algorithms	$N_c = 3$ ($P_{\text{pred}} = 74\%$)					$N_c = 8$ ($P_{\text{pred}} = 67\%$)				
	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$	$K = 10$	$K = 20$	$K = 30$	$K = 40$	$K = 50$
PCS	1.0993	1.0562	1.0435	1.0392	1.0560	1.0403	1.0096	1.0031	1.0013	1.0208
ACS	0.9896	0.9622	0.9671	1.0473	1.2712	0.9433	0.9264	0.9475	1.0755	1.3740
CS	0.9754	0.9791	0.9876	0.9942	0.9992	0.9577	0.9583	0.9651	0.9716	0.9768
Proposed	0.9680	0.9543	0.9580	0.9637	0.9703	0.9397	0.9322	0.9349	0.9400	0.9485

proposed algorithm. Basically, all computations involved in the semi-distributed BP in Sec. III-D and user-side recommendation in Sec. III-E are carried out exactly as in a centralized approach, where the server performs BP for item similarity computation and generates rating predictions using (1). Thus, the accuracy performance shown in Sec. IV also represents the performance of the centralized counterpart of the proposed algorithm. However, the proposed algorithm does incur communication overhead, trading off efficiency for privacy. In the semi-distributed BP on graph $\mathcal{G}_i$, for each user $u \in U_i$, there are $|I_u|$ λ-messages and $|I_u|$ μ-messages exchanged between the server and use u in each iteration. And the server needs to send a total of $|I_u|(N - |I_u|)$ item similarities to user u for user-side recommendation. Whereas in the centralized approach, only the generated recommendations need to be sent to the user.

IV. Experimental Evaluation

We evaluate the accuracy of the proposed privacy-preserving item-based CF algorithm on the 100K MovieLens dataset[1], which consists of $100,000$ ratings on 1682 items (movies) by 943 users. Each rating is an integer between 1 and 5. We randomly divide the dataset into two disjoint sets: a training set containing 80% of the ratings, and a test set containing the rest 20% of the ratings. The ratings in the training set are used as memory for the item-based CF algorithm to compute item similarities and predict unknown ratings. We compare the predicted ratings with the actual ratings in the test set to evaluate the accuracy of the recommendation algorithms in terms of MAE and RMSE, which are computed as follows

$$\text{MAE} = \frac{1}{|\mathbb{T}|} \sum_{r_{ui} \in \mathbb{T}} |r_{ui} - \hat{r}_{ui}|,$$

$$\text{RMSE} = \sqrt{\frac{1}{|\mathbb{T}|} \sum_{r_{ui} \in \mathbb{T}} (r_{ui} - \hat{r}_{ui})^2},$$

where $\hat{r}_{ui}$ is the predicted rating, and r_{ui} is the actual rating in the test set denoted by $\mathbb{T}$. The smaller the MAE and RMSE, the better accuracy. Note that RMSE is more sensitive to large errors than MAE.

We compare our proposed privacy-preserving algorithm with other item-based CF algorithms using the well-known similarity measures, including the CS, PCS and ACS methods as introduced in Sec. II. In all cases, rating predictions are generated by (1). We assume no privacy requirement is imposed on other algorithms, so the CS, PCS and ACS measures are directly applied to original rating data, and thus their results are not compromised by privacy techniques such as obfuscation. In particular, the presented results of the CS measure represent the best achievable performance of the distributed personal recommender system proposed in [13], as the item similarity between two items is computed based on the complete rating vectors associated with them in $\mathbb{R}$, rather than computed in an incremental manner as in [13].

The PCS and ACS methods compute the item similarity s_{ij} between two items i and j using the ratings from the set of their common users $U_{ij} = U_i \cap U_j$. We denote by N_c the minimum number of common users required for computing the item similarity s_{ij} using PCS and ACS. If $|U_{ij}| < N_c$, then neither item i nor item j will be used to predict each other's ratings. Let $\mathbb{N}_i$ be the set of all valid items for item i under N_c. To predict ratings on item i for user u, the neighborhood $\mathcal{N}_{ui}$ used in (1) is formed from the set of items in $\mathbb{N}_{ui} = \mathbb{N}_i \cap I_u$. In addition, given a required neighborhood size K, if $|\mathbb{N}_{ui}| < K$, we simply say the unknown rating r_{ui} in the test set $\mathbb{T}$ is unpredictable by the PCS and ACS. We denote $\mathbb{T}_K$ as the subset of all predictable ratings in $\mathbb{T}$ at neighborhood size K. Note that $\mathbb{T}_K$ changes with N_c, since N_c impacts $\mathbb{N}_i$. For fair comparison of different algorithms, we apply the same N_c to all evaluated algorithms.

In Tables IV and V, we examine the MAE and RMSE performances of various algorithms. The parameters of the proposed algorithm are set as $D = 4$, $\mathcal{S} = \{1, 2\}$, and $\sigma = 0.5$. We show the results for $N_c = 3$ and 8 with K varying from 10 to 50 in steps of 10. Under each N_c, to fairly compare performances of different K's, all results are obtained on the same subset of test ratings $\mathbb{T}_{50}$, and the percentage of ratings

[1] Available at: http://www.grouplens.org/node/73.

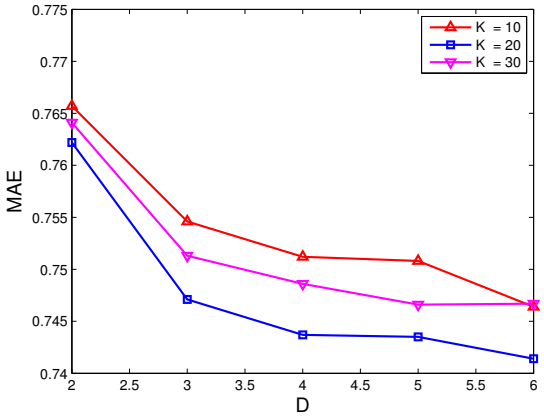


Fig. 5: Impact of D on MAE of the proposed algorithm.

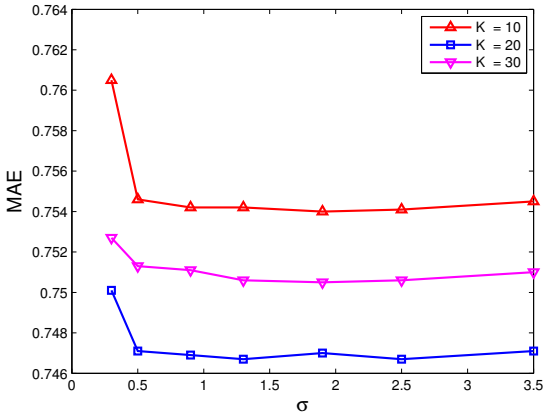


Fig. 6: Impact of σ on MAE of the proposed algorithm.

in $\mathbb{T}$ used for evaluation is

$$P_{\text{pred}} = |\mathbb{T}_{50}|/|\mathbb{T}| \times 100\%.$$

Our proposed algorithm achieves superior performance compared to other algorithms in terms of both MAE and RMSE. As K increases from 10, the performance of the proposed algorithm, as well as PCS and ACS algorithms, first improves but then degrades when K becomes too large, because ratings from neighbor items with smaller similarity to the active item, on which the rating is predicted, corrupt the prediction accuracy. Thus, to achieve the best performance, a proper neighborhood size K should be chosen. The computational complexity cost of our algorithm to solve for the similarities between item i and other items on graph $\mathcal{G}_i$ is $\mathcal{O}\left(NM^{2(1-\epsilon)}\right)$ for large N and M as discussed in Sec. III-C, whereas for the CS method the complexity is $\mathcal{O}\left(NM\right)$, and for PCS and ACS, the complexity is $\mathcal{O}\left(NM^{(1-\epsilon)}\right)$. Finally, as we will see next, the performance of our algorithm can be further improved if a higher degree D is used, but at the cost of increased computational complexity at users.

In the following experiments, we investigate the impact of the parameters D, σ, and $\mathcal{S}$ on the performance of our proposed algorithm. We always set $N_c = 3$ and evaluate the proposed algorithm on $\mathbb{T}_{50}$. In Fig. 5, we investigate the influence of group size D on the accuracy of the proposed algorithm, where we fix other parameters as $\mathcal{S} = \{1, 2\}$, and $\sigma = 0.5$. It can be seen that increasing D slightly improves accuracy. However, since the computational complexity at user side is $\mathcal{O}\left(\bar{N}DL^D\right)$, which is exponential in D, users need to wisely choose D according to their computational capability. In Fig. 6, we show the results for different σ's, where $D = 3$ and $\mathcal{S} = \{1, 2\}$. The performance slightly degrades if σ is too small or too large, since for a small σ, the curve of the factor function (6) quickly flattens out with respect to $|\hat{r}_{ui} - r_{ui}|$, while for a large σ, the curve becomes too flat. But overall, the algorithm is not sensitive for large dynamic ranges of σ.

In Table VI, we show the results of the proposed algorithm for different $\mathcal{S}$'s, where $\mathcal{S}_l = \{1, 2, \ldots, l\}, \forall l \in \{2, 5\}$, $D = 3$, and $\sigma = 0.5$. We observe that $\mathcal{S}_2$ and $\mathcal{S}_5$ achieve their best performance when $K = 20$, but $\mathcal{S}_2$ actually provides better accuracy than $\mathcal{S}_5$. This is because large l could cause overfitting, that is the obtained item similarity is strongly biased towards the memory data, and does not generalize well when used for prediction. Meanwhile, the computational complexity $\mathcal{O}\left(\bar{N}DL^D\right)$ at user side significantly increases with $L = |\mathcal{S}_l|$, depending on D.

TABLE VI: Impact of $\mathcal{S}$ on MAE of the proposed algorithm.

$\mathcal{S}$	MAE		
	$K = 10$	$K = 20$	$K = 30$
$\mathcal{S}_2$	0.7546	0.7471	0.7513
$\mathcal{S}_5$	0.7504	0.7500	0.7545

V. Related Works

Several existing works have addressed privacy issues in item-based recommender systems. The work in [13] presented a personal CF recommender running on the user side. Each user stores his data locally, and constructs an item-item similarity model using the cosine similarity measure by incrementally incorporating ratings from other neighbour users in a peer-to-peer (P2P) environment. The quality of the locally constructed similarity model depends on the set of neighbours a user can find and contact. The best is to use ratings from all common users of two items to evaluate the item similarity, which is much harder to implement in P2P architectures than in a centralized CF system. Moreover, the user privacy is not guaranteed as users need to share their ratings with each other.

Another work in [19] applied homomorphic encryption to obscure user ratings from the central server. The server maintains an item-item similarity model, and generates rating predictions blindly by performing homomorphic operations on the encrypted ratings. However, the authors assumed the similarity model is known *a priori*, and did not provide any privacy-preserving solution for that. Besides, key management required by cryptography tools could be demanding in practice. The general privacy solutions presented in other works, including perturbation [7] and differential privacy [8], can also be readily extended to item-based CF systems, but they only alleviate the degree of privacy leakage, yet at the cost of accuracy loss.

Previously, BP has been applied to recommender systems without considering privacy in [20]–[22]. In [20], the proposed algorithm therein follows the philosophy of the user-based CF algorithm. To receive recommendations, an active user discloses his ratings to other users, who then compare their own ratings with the active user's ratings on common items in order to update their "confidence", which can be understood as similarity to the active user. The central server collects "confidences" as well as ratings of all users, and sends back to users probabilistic messages regarding predicted ratings on items. It is much the same in [21], except that each user combines his "confidence" and ratings to form probabilistic messages on ratings, and sends them to the server. The work in [22] proposed to predict ratings for an active user on a Pairwise Markov Random Field (PMRF), where the local evidence for each unknown rating is the aggregated ratings from other users similar to the active user, and probabilistic messages on predicted ratings are exchanged between similar items. Whereas in this work, we are concerned with preserving privacy in item-based CF recommendation. Instead of directly using BP for rating prediction as in [20]–[22], we employ BP for item similarity computation in a semi-distributed fashion, where messages are exchanged between the server and users, without disclosing user ratings. The rating prediction for an active user is then locally computed at the user side by combing his own ratings and item similarity.

VI. Conclusion

In this work, we proposed a item-based CF recommender system that preservers user privacy in both item similarity computation and rating prediction, without relying on any privacy techniques, such as obfuscation and cryptography. Firstly, we formulated the item similarity computation as a probabilistic inference problem on the factor graph, which can be efficiently solved via BP, a probabilistic message passing algorithm. To avoid disclosing user ratings to the server or other users, we then introduced the semi-distributed architecture of BP, where probabilistic messages on item similarity are exchanged between the server and users. In addition, we proposed a complexity reduction technique for efficient inference at the user side. Finally, an active user locally generates rating predictions by averaging his own ratings on items weighted by their similarities to unseen items. Overall, the computation is carried out exactly as in a centralized approach, and thus there is no compromise in recommendation performance compared to the centralized counterpart of the proposed algorithm. The experimental results on the MovieLens dataset validated the superior accuracy of the proposed algorithm in terms of both MAE and RMSE.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 734–749, Jun. 2005.

[2] P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 Computer Supported Cooperative Work Conference*, 1994, pp. 175–186.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, 2001, pp. 285–295.

[4] M. S. Ackerman, L. F. Cranor, and J. Reagle, "Privacy in e-commerce: Examining user scenarios and privacy preferences," in *Proceedings of the 1st ACM Conference on Electronic Commerce*, Denver, CO, USA, 1999, pp. 1–8.

[5] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft, "BlurMe: Inferring and obfuscating user gender based on ratings," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, Dublin, Ireland, 2012, pp. 195–202.

[6] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, 2008, pp. 111–125.

[7] H. Polat and W. Du, "Privacy-preserving collaborative filtering using randomized perturbation techniques," in *Proceedings of the Third IEEE International Conference on Data Mining*, 2003, pp. 625–628.

[8] F. McSherry and I. Mironov, "Differentially private recommender systems: Building privacy into the net," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 627–636.

[9] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux, "Preserving privacy in collaborative filtering through distributed aggregation of offline profiles," in *Proceedings of the Third ACM Conference on Recommender Systems*, 2009, pp. 157–164.

[10] D. Mulligan and A. Schwartz, "Your place or mine?: Privacy concerns and solutions for server and client-side storage of personal information," in *Proceedings of the Tenth Conference on Computers, Freedom and Privacy: Challenging the Assumptions*. ACM, 2000, pp. 81–84.

[11] J. F. Canny, "Collaborative filtering with privacy," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002, pp. 45–57.

[12] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1053–1066, Jun. 2012.

[13] B. N. Miller, J. A. Konstan, and J. Riedl, "Pocketlens: Toward a personal recommender system," *ACM Transactions on Information Systems*, vol. 22, no. 3, pp. 437–476, Jul. 2004.

[14] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci, "Enhancing privacy and preserving accuracy of a distributed collaborative filtering," in *Proceedings of the 2007 ACM Conference on Recommender Systems*, 2007, pp. 9–16.

[15] N. Lathia, S. Hailes, and L. Capra, "Private distributed collaborative filtering using estimated concordance measures," in *Proceedings of the 2007 ACM Conference on Recommender Systems*, 2007, pp. 1–8.

[16] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[18] E. Ayday and F. Fekri, "BP-P2P: Belief propagation-based trust and reputation management for P2P networks," in *Proceedings of the 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2012, pp. 578–586.

[19] Z. Erkin, M. Beye, T. Veugen, and R. L. Lagendijk, "Privacy-preserving content-based recommender system," in *Proceedings of the 14th ACM Workshop on Multimedia and Security*, 2012, pp. 77–84.

[20] E. Ayday and F. Fekri, "A belief propagation based recommender system for online services," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, Barcelona, Spain, 2010, pp. 217–220.

[21] E. Ayday, A. Einolghozati, and F. Fekri, "BPRS: Belief propagation based iterative recommender system," in *Proceedings of the 2012 IEEE International Symposium on Information Theory*, Cambridge, MA, 2012, pp. 1992–1996.

[22] E. Ayday, J. Zou, A. Einolghozati, and F. Fekri, "A recommender system based on belief propagation over pairwise Markov random fields," in *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, USA, 2012.