Privacy-Preserving EM Algorithm for Clustering on Social Network

Bin Yang[1], Issei Sato[2], and Hiroshi Nakagawa[2]

[1] Graduate School of Information Science and Technology, The University of Tokyo
{yangbin,sato}@r.dl.itc.u-tokyo.ac.jp
[2] Information Technology Center, The University of Tokyo
nakagawa@dl.itc.u-tokyo.ac.jp

Abstract. We consider the clustering problem in a private social network, in which all vertices are independent and private, and each of them knows nothing about vertices other than itself and its neighbors. Many clustering methods for networks have recently been proposed. Some of these works have dealt with a mixed network of assortative and disassortative models. These methods have been based on the fact that the entire structure of the network is observable. However, entities in real social network may be private and thus cannot be observed. We propose a privacy-preserving EM algorithm for clustering on distributed networks that not only deals with the mixture of assortative and disassortative models but also protects the privacy of each vertex in the network. In our solution, each vertex is treated as an independent private party, and the problem becomes an n-party privacy-preserving clustering, where n is the number of vertices in the network. Our algorithm does not reveal any intermediate information through its execution. The total running time is only related to the number of clusters and the maximum degree of the network but this is nearly independent of the total vertex number.

1 Introduction

The analysis of social networks has attracted increasing amounts of attention in recent years since there have been progressively more social applications used in practice. Many clustering algorithms with respect to vertices in networks, on the other hand, such as the graph min-cut and label propagation, have been proposed. Most of these methods deal with a so-called assortative mixing model, in which vertices are divided into groups such that the members of each group are mostly connected to other members of the same group [1]. For example, in a communication network (Fig. 1(a)), each student belongs to either of two clubs. They communicate with other students in the same club more frequently than they do with students outside the club. Thus, the methods used for the assortative mixing model could be used to detect the structures in this kind of network. Inversely, in the disassortative mixing networks, vertices have most of their connections outside their group. For example, there are two groups of people in Fig. 1(b), producers and consumers. Most of their exchanges, denoted by the edges, will occur between these two classes. Even though both assortative and disassortative mixing models have theoretical and practical significance, their mixture is more meaningful in most practical applications. If, for example (Fig. 1(c)), researchers in the same field were treated as one group, there would generally be more connections inside each group. In addition, some cross-disciplinary researchers, such as researchers in computational linguistics, may regularly connect with researchers in other related fields, such as linguistics, psychology, and computer science, although they may frequently also connect with other members of the same group. Newman et al. [1] proposed a probabilistic mixture model that could deal with an assortative and disassortative mixture using an EM algorithm. Such a model is realistic for the clustering problem in social networks.

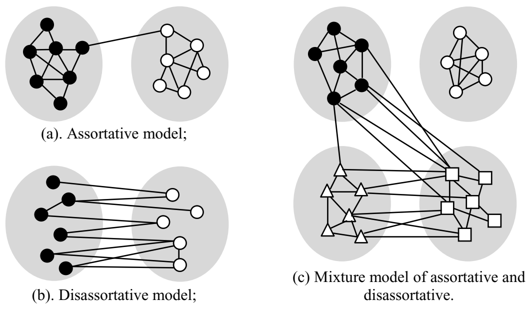
(a). Assortative model;
(b). Disassortative model;
(c) Mixture model of assortative and disassortative.

Fig. 1. *Assortative* model, *disassortative* model and mixture model

With increasing concerns about the issue of personal information and privacy protection, many privacy-preserving data mining algorithms have been proposed. In this paper, we consider the clustering problem on social networks, in which each member contacts the others via various means of communication, such as social applications (MSN, Yahoo Messenger, etc.), mobile phones of different service providers, etc. Their records are stored in different organizations, such as Microsoft, Yahoo, and mobile service providers. The collection of their data always contains large commercial value. However, it is impossible to make these competitors collaborate to perform data mining algorithms, such as clustering. This motivate us to develop a secure clustering algorithm that can be performed without any support of the organizations. Using this algorithm, not only vertices in the network are clustered, but the privacy of each vertex is also protected.

We summarize the related works in section 2. Section 3 introduces some background knowledge and Section 4 formulates our problem. We develop two basic secure summation protocols in Section 5, and propose our main EM-algorithm for private clustering based on these protocols in Sections 6. In Section 7, we discuss our evaluation of the performance of our protocols. The results of experiments conducted to evaluate the performance of our protocols are explained in Section 8, and concluding remarks are given in the last section.

2 Related Work

Newman et al. [1] proposed a probabilistic model for the mixture of assortative and disassortative models, and provided a corresponding EM algorithm, by which all vertices are clustered so that vertices in the same cluster had the same probabilities as if they had a connection with each vertex in the network.

Many kinds of privacy-preserving methods have been proposed to carry out data mining while protecting privacy. In general, privacy-preserving K-means clustering problems can be classified into horizontally partitioned K-means [4], vertically partitioned K-means [5] and arbitrarily partitioned K-means [6]. Methods using privacy-preserving EM clustering have also been proposed [7]. All of these methods deal with distributed databases with large numbers of data.

Secure data analysis in networks has recently attracted increasingly more attention. Hay et al. [8] proposed an efficient algorithm to compute the distribution of degrees of social networks. Another method of computing users privacy scores in online social networks was provided by Liu et al. [9]. Sakuma et al. [10] used the power method to solve ranking problems such as PageRank and HITS where each vertex in a network was treated as one party and only knew about its neighbors. Similar work was done by Kempe et al. [11].

Even though all these works provided valuable studies, clustering in private peer-to-peer networks has not yet attracted adequate attention. We focus on the clustering problem based on these kinds of private networks. In addition, we also concentrate on the mixture of assortative and disassortative models.

3 Preliminaries

Let us consider an un-weighted directed network of n vertices, numbered $1, \cdots, n$. The adjacency matrix of the network is denoted by $\mathbf{A}$ with elements $A_{ij} = 1$, if there is an edge from i to j, and 0 otherwise. If there is an adjacency from i to j, we say that i is a parent of j, and j is a child of i. We denote $pa(i)$ as the set of all parents of vertex i and $ch(i)$ as the set of its children. The union of $pa(i)$ and $ch(i)$ is referred to as the neighbors of i. All vertices fall into C clusters, and g_i denotes the cluster to which vertex i belongs. These g_is are treated as unknown or hidden data, and the purpose of our model is to deduce g_is from the adjacency matrix $\mathbf{A}$. We use the notation $[C]$ to denote the collection, $\{1, 2, \cdots, C\}$.

3.1 Probabilistic Mixture Model

Let θ_{ri} denote the probability that a link from a vertex in cluster r is connected to vertex i, and π_r denote the fraction of vertices in cluster r. The normalization conditions ($\sum_{r=1}^{C} \pi_r = 1$, $\sum_{i=1}^{n} \theta_{ri} = 1$) are satisfied. Using the probabilistic mixture model [1], the structural features in large-scale network can be detected by dividing the vertices of a network into clusters, such that the members of each cluster had similar patterns of connections to other vertices. We illustrate this generative graphical model in Fig. 2.

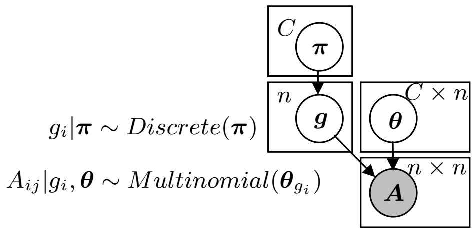
C
$\boldsymbol{\pi}$
n
$\boldsymbol{g}$
$C \times n$
$\boldsymbol{\theta}$
$n \times n$
$\boldsymbol{A}$
$g_i|\boldsymbol{\pi} \sim Discrete(\boldsymbol{\pi})$
$A_{ij}|g_i, \boldsymbol{\theta} \sim Multinomial(\boldsymbol{\theta}_{g_i})$

Fig. 2. Probabilistic generative model of mixture network

In Fig. 2, $\boldsymbol{\pi}$ expresses the vector $(\pi_1, \pi_2, \cdots, \pi_C)$; $\boldsymbol{\theta}_r$ expresses the vector $(\theta_{r1}, \theta_{r2}, \cdots, \theta_{rn})$ and $\boldsymbol{\theta}$ expresses the matrix $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_C)^T$. In this model, hidden variables g_is (the cluster labels of vertices) are generated from a discrete probability distribution with parameter $\boldsymbol{\pi}$, i.e., $\Pr(g_i = k) = \pi_k$, which means the probability that vertex i belongs to cluster k is π_k. After all g_is are determined, each vertex will choose some vertices with a multinomial distribution with corresponding parameter $\boldsymbol{\theta}_{g_i}$ (the g_i^{th} row in $\boldsymbol{\theta}$), and connect itself to each chosen vertex. Since the members of each cluster share the same parameter $\boldsymbol{\theta}_{g_i}$, they have similar patterns of connections to other vertices in the network.

Newman et al. [1] also proposed an EM-algorithm to infer the probabilities of these hidden variables. The E-step and M-step are derived as follows.

$$\text{E-step:} \quad q_{ir} = \frac{\pi_r \prod_{j=1}^{n} \theta_{rj}^{A_{ij}}}{\sum_{s=1}^{C} \pi_s \prod_{j=1}^{n} \theta_{sj}^{A_{ij}}}; \tag{1}$$

$$\text{M-step:} \quad \pi_r = \frac{1}{n} \sum_{i=1}^{n} q_{ir}; \quad \theta_{rj} = \frac{\sum_{i=1}^{n} A_{ij} q_{ir}}{\sum_{i=1}^{n} (\sum_{j=1}^{n} A_{ij}) q_{ir}}. \tag{2}$$

Here q_{ir} is defined as the probability that vertex i is a member of cluster r:

$$q_{ir} = \Pr(g_i = r | A, \pi, \theta) \ . \tag{3}$$

Since q_{ir}s denote the probability of g_i, inferring q_{ir}s is equivalent to inferring g_i. The adjacency matrix, $\mathbf{A}$, is treated as observed data. q_{ir}s are initialized to be arbitrary values in the beginning of the EM-algorithm, and converge to the final results after several rounds of E-steps and M-steps.

3.2 Utilities of Privacy-Preserving Data Mining

Homomorphic Encryption. In a public key encryption system, a public key pk, used to encrypt a given message, and a private key sk, used to decrypt the cryptograph, are generated by an asymmetric key algorithm. The private key is kept secret, while the public key may be widely distributed. Given a plaintext m, $c = E_{pk}(m, t)$ denotes a random encryption of m, and $d = D_{sk}(c)$ denotes the decryption of c, where t is randomly generated from Z_N, and N is a large positive integer. Paillier encryption [12] is a public key encryption system. It also satisfies additive homomorphism, i.e., there is an operation "$\cdot$", s.t. $\forall m_1, m_2 \in Z_N$,

$$E_{pk}(m_1 + m_2, t) \equiv E_{pk}(m_1, t_1) \cdot E_{pk}(m_2, t_2) \pmod{N^2} , \tag{4}$$

where t, t_1 and t_2 are random numbers. We will omit these random numbers when they were not necessary. Using this property, we can securely compute the cryptograph of the summation of two numbers, m_1 and m_2, only given their cryptographs. The following condition can be obtained from (4).

$$E_{pk}(m \cdot k) \equiv E_{pk}(m)^k \pmod{N^2} . \tag{5}$$

Secure Summation Protocols. Suppose each party has a private input. All the parties collaborate to compute the summation of all their inputs, without any party obtaining any information about other parties. Such a protocol is called a secure summation protocol. Many secure summation protocols, such as those by Kantarcoglu et al. [13], have been proposed. But these methods have been based on the assumption that any two parties are connected.

4 Problem Statement

We focus on the clustering problem in a social network described in Section 3. Furthermore, we also protect the privacy of each vertex in the network.

4.1 Assumptions

We treat each vertex in the network as one party. Thus, a network containing n vertices becomes an n-party system. We also assume that this network is a connected network, in which there is at least one path between any pair of vertices. There is no special vertex in the network, i.e., each vertex performs the same operations. These assumptions are of practical significance. For example, the relations of sending e-mail can be used to construct a network. Each vertex (an e-mail user) can be seen as one party. Hence, each e-mail user knows its neighbors, since it is connected to each neighbor using e-mail. Also, all e-mail users in this network are equivalent.

Since the vertices, such as e-mail users, never want to reveal private information about themselves in practice, we need to consider the privacy of each vertex in the network. We specifically assume that each vertex only knows about itself and its neighbors. First, it knows all information about itself. Second, it only knows about the connections with its neighbors. Third, it knows nothing about other vertices, not even whether they exist. Moreover, we assume all parties are semi-honest, which means that they all correctly follow the protocol with the exception that they keep a record of all their intermediate computations.

The knowledge range of any vertex is outlined in Fig. 3. We take the white vertex as the current vertex. It only knows about its neighbors (gray) since there is an edge between them. However, it does not know anything about the other (black) vertices, and even does not know whether they exist. In addition, it even does not know whether any pair of its neighbors is connected or not.

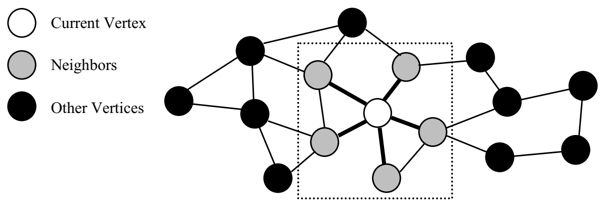

Fig. 3. Range of vision of a vertex in private network

Table 1. Protocol 1 - Local Secure Summation Protocol

```
Inputs: Party $i$ has an $x_i$, for $i = 0, 1, 2, \cdots, m$;
Outputs: Party 0 gets $x = \sum_{i=0}^{m} x_i$; other parties get nothing;
01 Party $m$ generates a set of keys $(pk, sk)$;
02   $pk$ is published to all parties; $sk$ is known to party $m$ only;
03 For $i = 1$ to $m - 1$
04   Party $i$ encrypts its input: $X_i = E_{pk}(x_i)$, and sends $X_i$ to Party 0;
05 Party 0 generates a random number $r_0$, and encrypts it: $R_0 = E_{pk}(r_0)$;
06 Party 0 computes $Z = R_0 \cdot \prod_{i=1}^{m-1} X_i$, and sends $Z$ to Party $m$;
07 Party $m$ decrypts $Z$: $z = D_{sk}(Z) = r_0 + \sum_{i=1}^{m-1} x_i$;
08 Party $m$ computes $z' = z + x_m$, and send $z'$ to Party 0;
09 Party 0 computes $x = z' - r_0 + x_0 = \sum_{i=0}^{m} x_i$.
```

4.2 Private Variables and Public Variables

In our private network, the variables A_{ij}, π_r, θ_{rj}, and q_{ir} are distributed into all parties. The A_{ij} denotes whether the pair of (i, j) is connected, so we treat it as private information of parties i and j. The π_r denotes the fraction of vertices in cluster r. As it contains nothing about individual parties, we publish it to all parties. The θ_{rj} expresses the relationship between cluster r and vertex j. We assume it is only known to party j. The q_{ir} is similarly only known to party i.

5 Secure Summation Protocols on Networks

We propose two secure summation protocols for the private network.

5.1 Local Secure Summation Protocol

Suppose a party, numbered 0, has m children, numbered $1, 2, \cdots, m$, in turn. Each of these parties has a private input x_i $(i = 0, 1, \cdots, m)$. After this protocol is executed, party 0 securely obtains the summation: $x = \sum_{i=0}^{m} x_i$, while other parties obtain nothing. We assume that party 0 only communicates with its children, and these children do not communicate with each other.

The detail of this protocol is shown in Table 1. Since only Party m can decrypt messages, Party 0 can obtain nothing about x_i from X_i $(i \in \{1, 2, \cdots, m-1\})$. From the homomorphism of encryption, $Z = E_{pk}(r_0 + \sum_{i=1}^{m-1} x_i)$, Party m can then compute $z = r_0 + \sum_{i=1}^{m-1} x_i$. As Party m does not know r_0, it can obtain nothing about $\sum_{i=1}^{m-1} x_i$ from the values of z. In summary, nothing can be inferred from the intermediate information other than the final result, x.

5.2 Global Secure Summation Protocol

The goal of this protocol is to securely sum up the inputs of all parties in our distributed network without revealing the privacy of any party. The final result is published to all parties throughout the network. Under the assumption in Section 4, each party can only communicate with its neighbors. Nevertheless, we can arbitrarily choose one party as a root and construct a spanning tree $\mathbf{T}$ from the network, since the network is connected. We use $\mathbf{T}$ to accumulate and distribute data.

Table 2. Protocol 2 - Global Secure Summation Protocol

```
Inputs: Party $i$ has an $x_i$, where $i = 1, 2, \cdots, n$, (all vertices in the network)
        where 1 is the root, and $2, 3, \cdots, m$ are all children of root in $\mathbf{T}$;
Outputs: All parties get the summation $x = \sum_{i=1}^{n} x_i$;
01  Choose a leaf party, called $n$, who has no child in $\mathbf{T}$;
02  Party $n$ generates a set of keys $(pk, sk)$;
03    $pk$ is published to all parties via the paths in $\mathbf{T}$; $sk$ is known to party $n$ only;
04  Each party encrypts its input: $X_i = E_{pk}(x_i)$;
05  Each party $i$ computes $Y_i$ as the following
06    $Y_i := X_i \cdot \prod_j Y_j$ ($j$ is the child of $i$ in $\mathbf{T}$);
07  Party 1 sends $Y_1$ to Party $n$ via the paths in $\mathbf{T}$;
08  Party $n$ decrypts $Y_1$: $x = D_{sk}(Y_1) = \sum_{i=1}^{n} x_i$;
09  Party $n$ publishes $x$ to all parties via the paths in $\mathbf{T}$.
```

The detail of this protocol is shown in Table 2. The equation in line 06 implies that each vertex accumulates the cryptographs of the summation of the sub-tree of itself, and sends the result, Y_i, to its parent in $\mathbf{T}$. Hence Y_1 in the root is the cryptograph of the summation of all vertices. Moreover, since only Party n can decrypt messages, nothing is revealed through the execution of the protocol other than the final result.

6 Private Clustering on Networks

The main procedure in our private clustering is the same as the original method, in which E-step and M-step were performed repeatedly until convergence. The only difference is that we need to protect the privacy of each vertex in each step.

6.1 Private E-step

In the private E-step, each Party i computes its private q_{ir}s in (1) without revealing θ_{rj}s. We now simplify the E-step (1) by introducing a new variable:

$$\alpha_{ir} = \pi_r \prod_{j=1}^{n} \theta_{rj}^{A_{ij}} \ . \tag{6}$$

Hence, the q_{ir} of party i can be rewritten as follows, for $r \in [C]$.

$$q_{ir} = \frac{\alpha_{ir}}{\sum_{s=1}^{C} \alpha_{is}} \ . \tag{7}$$

If party i obtains the values of α_{ir}s ($r \in [C]$), the q_{ir}s can be directly computed. Consequently, we focus on securely computing of α_{ir}s (6). Although (6) is a product of n items, from the definitions of A_{ij}, we could eliminate the term θ_{rj} if party j is not a child of party i. In other words, the value of α_{ir} becomes the product of π_r and the θ_{rj}s of all children of party i, i.e.,

$$\alpha_{ir} = \pi_r \cdot \prod_{A_{ij}=1} \theta_{rj} = \pi_r \cdot \prod_{j \in ch(i)} \theta_{rj} \ . \tag{8}$$

Hence, we have

$$\log \alpha_{ir} = \log \pi_r + \sum_{j \in ch(i)} \log \theta_{rj} \ . \tag{9}$$

Here, each $\log \theta_{rj}$ can be seen as a private input of party j. Then, our goal becomes to securely compute the summation of these $\log \theta_{rj}$s ($j \in ch(i)$). To do this, we only need to perform Protocol 1 by treating these $\log \theta_{rj}$s ($j \in ch(i)$) as the parameters of this protocol (private inputs of children of party i). Throughout this execution, the value of each θ_{rj} is kept secret with each party.

6.2 Private M-step

In the private M-step, each Party j computes θ_{rj}s and all parties obtain the π_{rs} in (2) without revealing any q_{ir}s. We now introduce a new variable:

$$\beta_{rj} = \sum_{i=1}^{n} A_{ij} q_{ir}, \quad \beta_r = \sum_{j=1}^{n} \beta_{rj} \ . \tag{10}$$

Similarly to (8), β_{rj} can be rewritten as:

$$\beta_{rj} = \sum_{A_{ij}=1} q_{ir} = \sum_{i \in pa(j)} q_{ir} \ . \tag{11}$$

Similarly, treating q_{ir}s as the private inputs of its parents, party j can securely compute the value of β_{rj} with Protocol 1 without revealing any information about q_{ir}s. In addition, substituting the definition of β_{rj} into (2), θ_{rj} becomes

$$\theta_{rj} = \frac{\beta_{rj}}{\sum_{k=1}^{n} \beta_{rk}} = \frac{\beta_{rj}}{\beta_r} \ . \tag{12}$$

Since Party j does not know the values of β_{rk}s for $k \neq j$, it cannot compute the $\beta_r = \sum_{k=1}^{n} \beta_{rk}$. As β_r does not include any private information, we publish β_r ($r \in [C]$) to all parties. The problem of computing β_r becomes that of securely computing the summation of the private inputs of all parties in the network and publishing the final result to all parties. Hence, it can be solved with Protocol 2. Given the values of β_rs ($r \in [C]$), party j can compute θ_{rj}s using (12) by itself.

Using the definition of π_r and treating q_{ir} as the private input of each party in the network, securely computing π_r is equivalent to securely summing all parties in the entire network and it can thus be solved using Protocol 2.

7 Performance

We discuss the efficiency of our method here. Both computation and communication can be carried out in parallel in the execution of our protocol. As each party performs operators with only one neighbor at the same time, evaluating the total running time is equivalent to the edge coloring problem in graph theory. The edge coloring of a graph is generally the assignment of "*colors*" to its edges so that no two adjacent edges have the same color. Vizing [15] has shown that the color index of a graph with maximum vertex-degree K is either K or $K + 1$.

We now discuss the running time for one round of computation, which includes one E-step and one M-step. In the E-step, each party i performs Protocol 1 with all its children for C times. From Vizings conclusion [15], the total running time for this stage is $O(CK)$. In the M-step, the β_rs and π_rs are all accumulated with Protocol 2. Because the running time for one duration of Protocol 2 is $O(\log_K n)$ and the β_rs and π_rs include $2C$ values, the running time for these accumulations is $O(C \log_K n)$. The secure computation of β_{rj} involves C times of executions of Protocol 1 with all its parents. From Vizings conclusion [15], the total running time for this computation is at most $O(CK)$.

In summary, the running time for one round of E-step and M-step is $O(CK + C \log_K n)$. Nevertheless, this is just an atomic operation of the entire EM-algorithm. If we need to perform R rounds of E-step and M-step until they converge, the entire running time will become $O(RC(K + \log_K n))$.

8 Experiments

We implemented the protocols in C++ using the OpenSSL library, which is an implementation with large numbers. Our machines were standard personal computers with Intel Pentium Core2 Duo CPUs, with a frequency of 2.67 GHz, and 2.00 GB of RAM. A homomorphic encryption system, the Paillier cryptosystem [12], was used to implement the protocols. The network environment in our experiments was a wireless LAN based on IEEE802.11g/IEEE802.11b.

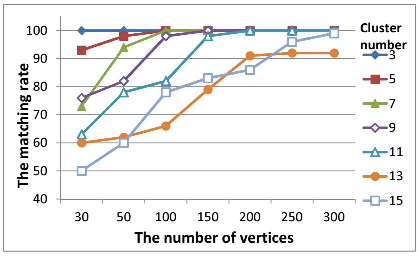

Fig. 4. Accuracy of matching

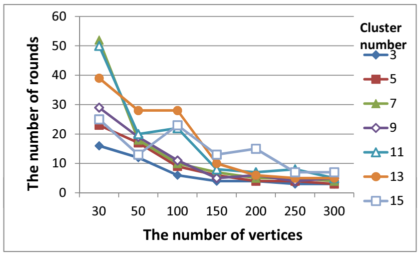

Fig. 5. Necessary number of rounds

We used artificial and real data to evaluate the accuracy and efficiency of our protocol. The artificial data were generated from generative models with different parameters. We evaluated them by comparing the inferred results with the corresponding parameters. Moreover, we selected a network of books about US politics compiled by Valdis Krebs [16] as the real data, in which nodes represented books about US politics sold by Amazon.com and edges represented the co-purchasing of books by the same buyers, as indicated by the *customers who bought this book also bought these other books* feature on Amazon. Nodes were given three labels to indicate whether they were *liberal*, *neutral*, or *conservative*. We compared our inferred results with them.

8.1 Accuracy

We executed our protocol and counted the number of results that matched the true values. We used matching rate, the percentage of matched data, to evaluate accuracy. In Fig. 4, each line expresses the relation between the number of vertices and the accuracy with respect to a special number of clusters. We found that the results could be correctly inferred with our protocol for three clusters. However, increasing the number of cluster will lead to a decrease in accuracy. Fortunately, we could increase accuracy by increasing the number of vertices. This can be verified from Fig. 4, in which each line is increasing. We also evaluated the speed of convergence by counting the number of necessary rounds of computation until convergence occurred (Fig. 5). We found that convergence became faster when there were far more vertices than numbers of clusters. We then evaluated the data set of books about US politics [16]. Although this network contains only 105 vertices and 441 pairs of edges, the accuracy was about 86%. The intuitive image of these experimental results of real data are shown in Fig. 6. We found they are quite close to the original data.

8.2 Efficiency

We used two computers in this experiment to simulate distributed computation. We executed the operators for each pair one-by-one by treating these two computers as two adjacent parties and recording the running time for each step.

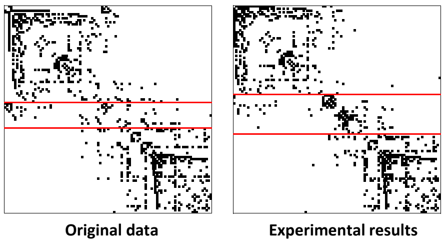

Fig. 6. Clustering result of real data

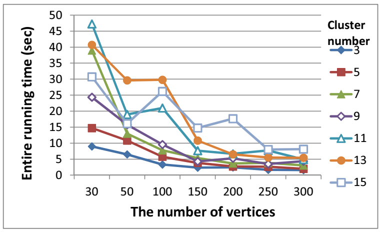

Fig. 7. Number of vertices vs. entire running time

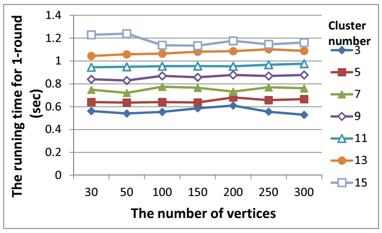

Fig. 8. Number of vertices vs. one-round of running time

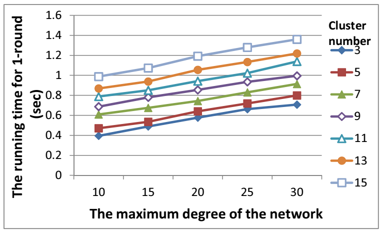

Fig. 9. Maximum degree vs. one-round of running time

We designed a parallel solution using Vizing's solution [15], and calculated the entire computational time in this parallel environment. All of our experimental results also contained the communication time. Fig. 7 plots the relation between the number of vertices and the total running time with respect to the different number of cluster. Combined with Fig. 5, we also obtained the results in Fig. 8, which illustrates one-round of running time with respect to different numbers of clusters and vertices. An interesting phenomenon is that increasing the number of vertices can decrease the entire running time (Fig. 7), although one-round running time (Fig. 8) is nearly independent of the number of vertices. This implies our privacy-preserving schema for clustering can be applied to very large-scale networks such as social networks. Fig. 9 also compares the running time with the maximum degree. The one-round of running time is increased with the increase in the maximum degree. We also found that the results in Fig. 9 agree with our description in Section 7. We also evaluated the real data using the protocol with encryption. It only needed 12 rounds of computations until convergence occurred. The entire running time was about 11 sec. That implies the average running time for one-round of computation is only about 1 sec.

9 Conclusion

We proposed a secure EM-algorithm to cluster vertices in a private network in this paper. This method deals with the mixture of assortative and disassortative mixing models. Assuming that each vertex is independent, private, and semi-honest, our algorithm was sufficiently secure to preserve the privacy of every vertex. The running time for our algorithm only depended on the number of clusters and the maximum degree. Since our algorithm does not become inefficient with larger amounts of data, it can be applied to very large-scale networks.

References

1. Newman, M.E.J., Leicht, E.A.: Grid Mixture models and exploratory analysis in networks. Proc. Natl. Acad. Sci. USA 104, 9564–9569 (2007)
2. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: The 14th ACM Conference on Computer and Communications Security (2007)
3. Koller, D., Pfeffer, A.: Probabilistic frame-based systems. In: The 15th National Conference on Artificial Intelligence (1998)
4. Jha, S., Kruger, L., McDamiel, P.: Privacy preserving clustering. In: The 10th European Symposium on Research in Computer Security (2005)
5. Vaidya, J., Clifton, C.: Privacy-Preserving k-means clustering over vertically partitioned data. In: The 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2003)
6. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: The 11th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2003)
7. Lin, X., Clifton, C., Zhu, M.: Privacy-preserving clustering with distributed EM mixture. Knowledge and Information Systems, 68–81 (2004)
8. Hay, M., Li, C., Miklau, G., Jensen, D.: Accurate estimation of the degree distribution of private networks. In: The 9th IEEE International Conference on Data Mining (2009)
9. Liu, K., Terzi, E.: A framework for computing the privacy scores of users in online social networks. In: The 9th IEEE International Conference on Data Mining (2009)
10. Sakuma, J., Kobayashi, S.: Link analysis for private weighted graphs. In: The 32nd ACM SIGIR Conference (2009)
11. Kempe, D., McSherry, F.: A decentralized algorithm for spectral analysis. Journal of Computer and System Sciences 74(1), 70–83 (2008)
12. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
13. Kantarcoglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. In: The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, DMKD 2002 (2002)
14. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
15. Vizing, V.G.: On an estimate of the chromatic class of a p-graph. Diskret. Analiz 3, 25–30 (1964)
16. Krebs, V.: `http://www.orgnet.com/`