An Edge-based Clustering Algorithm to Detect Social Circles in Ego Networks

Yu Wang
School of Computer Science and Technology, Xidian University Xi'an,710071, China
School of Economics and Management, Xidian University Xi'an 710071, China
cheerwangyu@163.com

Lin Gao
School of Computer Science and Technology, Xidian University Xi'an,710071, China
lgao@mail.xidian.edu.cn;

***Abstract*—Organizing users' friends in personal social networks, i.e., ego networks, into circles is an important task for online social networks. Social networking sites allow users to manually categorize their friends into social circles. However, it is time consuming and does not update automatically as a user adds more friends. In this paper, we propose an edge-based clustering algorithm to detect social circles in ego networks automatically. Firstly, we reconstruct ego networks by predicting the missing links. Then, we define the similarity of adjacent edges and cluster edges by single-linkage hierarchical clustering algorithm. Finally, we label each circle by abstracting its common properties to explain why this circle forms. The experimental results demonstrate it is a better way to characterize social circles from the respect of edges. Our algorithm outperforms the link community algorithm and low-rank embedding algorithm in terms of accuracy, and is more efficient than the probabilistic model algorithm. Our proposed method is validated as an effective algorithm in identifying social circles.**

***Index Terms*—graph clustering, ego networks, social circles, link community**

I. INTRODUCTION

Understanding and modeling network structures have been a focus of attention in a number of diverse fields, including physics, biology, computer science, statistics, and social sciences. Social network analysis started in the 1930's and has become one of the most important topics in sociology [1][2]. A fundamental problem in the study of social networks is community detection and some algorithms are proposed to solve this problem[3][4][5][6]. Communities can have concrete applications. For instance, identifying clusters of customers with similar interests in the network of purchase relationships between customers and products of online retailers (like, e. g., www.amazon.com) enables to set up efficient recommendation systems [7], that better guide customers through the list of items of the retailer and enhance the business opportunities. In recent years, Social Networking Services (SNS) such as Facebook and Twitter have become a major part of people's daily life. They are serving as a communication and information-sharing platform for their users. Users in such online social networks usually have hundreds of their friends and acquaintances. So it is necessary to organize their friends into what we refer to as social circles. Detecting social circles for a specific user can be formulated as an ego network clustering problem. The ego network of a user is the network of friendships among his friends. An illustration of ego network is shown in Fig. 1. In this example, user's friends are organized into four circles. The circle of colleagues is overlapped with the circle of college friends.

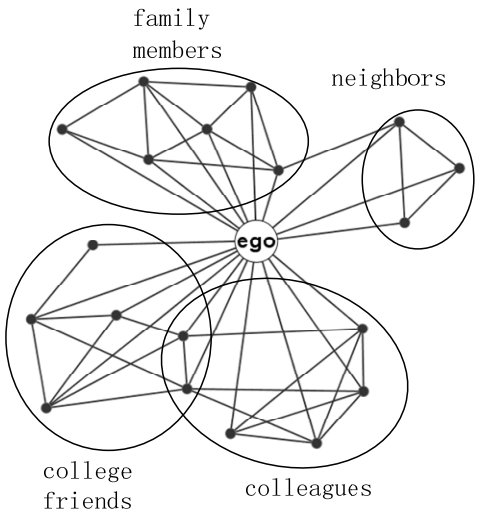


Fig. 1. An illustration of ego networks.

Previous community detection methods have two main disadvantages in solving this problem. One flaw is they find dense subgraphs as communities in the "observed" network. However, there are "hidden" relations in ego networks. For example, some people are friends and form a circle but they have not built connections on website yet. If we only consider the density of subgraphs, we will lose these circles. Another limitation is they don't give explanations about why we organize these people as a circle: they are families, colleagues, living in the same towns or something else. To overcome these problems,

doi:10.4304/jcp.8.10.2575-2582

additional information such as user profiles is used. Tetsuya Yoshida [8] proposed low-rank embedding approach to find hidden communities based on user profiles. Nevertheless, this approach needs users to specify the number of circles in advance and cannot identify the overlapped communities. Stephan Günnemann et al. [9] developed the algorithm DB-CSC to find density-based subspace in graphs with feature vectors. This algorithm misses finding circles with small size. These two methods still cannot interpret what makes each circle emerge. Julian McAuley et al. [10] suggested a probabilistic model algorithm to discover circles. This approach can not only identify circles, but also explicate what dimensions of the profile caused the circle to form. However, this approach never identifies more than 10 circles per user. What's more, the efficiency of this approach is very low. It needs hours of time to execute when the size of the ego network is bigger than 1000.

Considering the fact that a circle forms because people in that circle have the same relationship, it is a natural thought to identify circles by finding similar edges, since edges represent relationships in graph model. In this paper, we detect circles in ego networks by an improved link community algorithm. Circles we identified are not only densely connected but its members also share common properties. We first reconstruct ego networks to recover the missing relations. Then we give a new similarity definition of the edges and cluster edges by link community algorithm. Finally, we label circles to explicate what features caused these circles' emergence. The experimental results show that compared with link community algorithm and low-rank embedding algorithm, our method achieves a higher accuracy between the predicted circles and the known circles. Our algorithm outperforms the probabilistic model algorithm in terms of running time. Our proposed method is validated as an effective algorithm in discovering social circles in ego networks.

II. METHODS

A. Link Community Algorithm

Ahn et al. [11] suggested an unorthodox approach which reinvented communities as groups of edges rather than vertices. They defined the inclusive neighbors of a vertex i as: $n_+(i) \equiv \{x | d(i,x) \le 1\}$, where $d(i,x)$ is the length of the shortest path between vertices i and x. The set simply contains the vertex itself and its neighbors. From this, they defined the similarity between each pair of adjacent edges. The similarity between edges e_{ik} and e_{jk} is

$$S(e_{ik}, e_{jk}) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|}.$$

Single-linkage hierarchical clustering is employed to build a dendrogram where each leaf is an edge from the original network and branches represent link communities. To solve the problem of cutting this dendrogram at which level, conceptions of link density and partition density are given. For a network with M edges and N vertices, $P = \{P_1, ..., P_C\}$ is a partition of the edges into C subsets. The number of edges in subset P_c is $m_c = |P_c|$. The number of induced vertices, all vertices that those edges touch, is $n_c = |\cup_{e_{ij} \in P_c} \{i, j\}|$. The link density D_c of community c is

$$D_c = \frac{m_c - (n_c - 1)}{n_c(n_c - 1)/2 - (n_c - 1)}.$$

The partition density D is $D = \frac{2}{M} \sum m_c \frac{m_c - (n_c - 1)}{(n_c - 2)(n_c - 1)}$, which is the average of D_c.

The dendrogram could be cut at the level with maximum partition density, but it doesn't mean it is the only level worth exploring. For example, students at university may organize at multiple scales, from schools, down to the departmental level and then further down to research groups and small-scale collaborations. The most modular structure may form at the departmental level, but the structures of both smaller research groups and larger school-wide organization are still relevant.

B. Network Reconstruction

Ego networks are dynamic, and new relationships build over time. A group of friends with relatively sparse relationships now could be found as a circle later when the network grows with more edges in future. To identify such circles, we predict those missing links and reconstruct ego networks. We predict there should be an edge between two vertices if these two vertices are similar in both topological structure and feature profile. In our datasets, the profile information of all people in an ego network can be represented as a tree. An example is shown in Fig. 2. Users' features are classified. A leaf of the tree represents a specific feature that at least one person in the ego network has. Note that user profile trees are defined per ego network: while many thousands of colleges (for example) exist among all social network users, only a small number appear among any particular user's friends. We use the profile vector to characterize a user's feature. If a person x whose hometown is Boston is working as a programmer and has studied in MIT and Stanford, then his profile vector, $profile(x)$, is (0,1,0,1,0,1,1,0).

We use Jaccard coefficient to measure the topological similarity, since it consumes little time and performs relatively good among many local indices [12]. We use cosine which is commonly used to measure the similarity of two profile vectors. Algorithm 1 is used to reconstruct ego network. For each pair of directly linking vertices, we compute the topological similarity and profile similarity of them in line 2-5. We label the median of all the topological similarities as M_VTS and the median of all the profile similarities as M_VPS in line 6-7. We predict whether there should be a missing link in line

8-14. For each pair of vertices who has no direct connection, if their topological similarity is bigger than M_VTS and profile similarity is bigger than M_VPS, we link these two vertices.

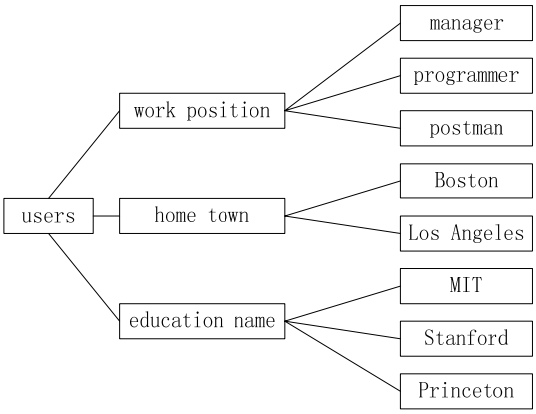


Fig. 2. An illustration of user profile tree.

Algorithm 1. Ego network reconstructing algorithm.
Input:
The adjacent matrix A of an ego network $G = (V, E)$;
Users' profile information.
Output:
The adjacent matrix RA of the reconstructed ego network $RG = (V, E_R)$.
Steps:
01: $RA = A$; $VTS = \Phi$; $VPS = \Phi$;
//Initialization
//Set VTS stores the topological similarities of vertices.
//Set VPS stores the profile similarities of vertices.
02: for each edge $e \in E$ do
03: calculate the topological similarity of the two vertices linked by edge e and insert it in set VTS;
04: calculate the profile similarity of the two vertices linked by edge e and insert it in set VPS;
05: endfor
06: compute the median of set VTS, M_VTS;
07: compute the median of set VPS, M_VTS;
08: for each vertex $i \in V$ do
09: for each vertex $j \in V$ do
10: if $vts(i, j) > M\_VTS$ and $vps(i, j) > M\_VPS$ do
// $vts(i, j)$ is the topological similarity of vertex i and // vertex j.
// $vps(i, j)$ is the profile similarity of vertex i and // vertex j.
11: set $RA(i, j) = 1$;
12: endif
13: endfor
14: endfor
15: return RA

C. Circle Detection

Our circle detection algorithm contains three steps. Firstly, we compute the similarity of each pair of edges who have common vertex. The similarity of edges consists of two parts: topological similarity and profile similarity. The definition of the topological similarity of two adjacent edges, $lts(e_{ik}, e_{jk})$, is the same as what in link community algorithm. The profile similarity of edges e_{ik} and e_{jk} is defined as $lps(e_{ik}, e_{jk}) = \cos(profile(i), profile(j))$. We use parameter α to balance the effect of topological structure and profile information. The final similarity of edges e_{ik} and e_{jk} is $S(e_{ik}, e_{jk}) = \alpha \times lts(e_{ik}, e_{jk}) + (1-\alpha) \times lps(e_{ik}, e_{jk})$. Then we employ single-linkage hierarchical clustering algorithm to partition these edges. The induced vertices of similar edges constitute a circle. To simplify the clustering process, we utilize similarity threshold parameter τ to decide the cluster granularity. Finally, we abstract common features from the edges in a circle to explain why this circle emerges. Algorithm 2 is applied to detect the circles.

For each edge in the reconstructed network $RG = (V, E_R)$, if it isn't in a known circle, we insert it into an empty set named by $edges\_circle$ in line 5-9. Then we find the edges similar to any of the edge in set $edges\_circle$, and insert these edges into set $edges\_circle$ in line 10-12 (When we talk about two edges are similar, we mean the similarity between them is bigger than the parameter τ). This process will stop until we couldn't insert any edge into set $edges\_circle$. The induced vertices of the $edges\_circle$ form a circle and we insert this circle into set CS in line 17. Finally, we obtain the common features of the edges in set $edges\_circle$ by the circle feature abstract procedure in line 18.

Algorithm 2. Circle detecting algorithm
Input:
The adjacent matrix RA of the reconstructed ego network $RG = (V, E_R)$;
Users' profile information;
Similarity threshold parameter τ.
Output:
The set of social circles, SC;
The set of circle feature vectors, CFV.
// Each circle feature vector is corresponding to a circle
// to interpret the reason this circle exists.
Steps:
01: $SC = \Phi$;
02: $CFV = \Phi$;
03: $Searched\_Edges = \Phi$;
// Set $Searched\_Edges$ stores the edges in known
// communities

04: compute the similarity of each pair of edges in graph $RG = (V, E_R)$;
05: for each edge $i \in E_R$ do
06: if $i \notin Searched\_Edges$ do
07: insert i into $Searched\_Edges$;
08: $edges\_circle = \Phi$;
09: insert i into $edges\_circle$;
10: for each edge $j \in edges\_circle$ do
11: if $\exists$ edge $k \in E_R$ and $k \notin Searched\_Edges$ and $S(k, j) > \tau$ do
12: insert k into $edges\_circle$;
13: insert k into $Searched\_Edges$;
14: endif
15: endfor
16: endif
17: insert the induced vertices of $edges\_circle$ into SC;
18: $circle\_feature\_vector$ =circle feature abstract($edges\_circle$);
19: insert $circle\_feature\_vector$ into CFV;
20: endfor
21: return SC, CFV

A circle forms because people in this circle have the same relationship. We can explain the meaning of a circle by abstracting the common features of the edges in the circle. The feature vector of an edge $e_{i,j}$ is the common features of vertex i and vertex j, which is defended as $edge\_feature\_vector(e_{i,j}) = profile(i) \cap profile(j)$. In a circle, if there are some features appearing in almost every edge feature vector, and then it is reasonable to say these features cause the existence of the circle. Algorithm 3 is a subprogram of algorithm 2. It computes which dimensions of users' profile information lead to densely linked circles. Algorithm 3 returns a vector, called circle feature vector. The value of element i in the vector is a weight, and it represents the contribution of feature i to forming the circle. If it is big enough, say, 0.5, then we can consider that feature i should be responsible for the emergence of the circle. The detail of algorithm 3 is as follows:

Algorithm 3. Circle feature abstract (ES **)**
Input:
Edge set ES of circle C.
Output:
The circle feature vector of circle C.
Steps:
01: $sum\_vector = 0$; //initialization
02: for each edge $e \in ES$, do
03: $sum\_vector = sum\_vector + edge\_feature\_vector(e)$;
04: endfor
05: $circle\_feature\_vector = \frac{sum\_vector}{|ES|}$;
06: return $circle\_feature\_vector$

D. Our Algorithm

Individuals can participate in multiple circles, but relationships often exist for one dominant reason. It is appropriate to consider circles as sets of similar connection. Based on the analysis above, our method identifies circles by partitioning edges.We give the overall framework of our method in algorithm 4.

Algorithm 4. Our algorithm
Input:
The adjacent matrix A of an ego network $G = (V, E)$;
Users' profile information;
Similarity threshold parameter τ.
Output:
The set of social circles;
The set of circle feature vectors.
Steps:
1. Run algorithm 1 to get a reconstructed ego network;
2. Run algorithm 2 to detect social circles and the meaning of each circle.
3. Return the set of social circles and the set of circle feature vectors.

Our method can detect overlapping and hierarchical circles simultaneously. The time complexity is $O(m^2)$, which is relatively high. Here m is the number of edges in the ego network. Fortunately, since the scale of an ego network is usually not so big, the running time of our algorithm is acceptable.

III. Experiments and Results

A. Experimental Data

We obtained ego networks and ground-truth from Facebook. The ground-truth data work as benchmark. We obtained profile and network data from 10 ego networks, consisting of 193 circles and 4,039 users. On average, users identified 19 circles in their ego networks, with an average circle size of 22 friends. Examples of such circles include students of common universities, sports teams, relatives, etc. The same dataset are used in paper 6.

B. Evaluation Metrics

We validate the accuracy of our algorithm by comparing the set of predicted circles P with the set of benchmark circles B. Two evaluation metrics are employed, 1-BER and F-measure.

1) 1-BER

To measure the alignment between a predicted circle p and a benchmark circle b, we compute the Balanced Error Rate (BER) between these two circles [13],

$$BER(b, p) = \frac{1}{2}\left(\frac{|p \setminus b|}{|p|} + \frac{|b \setminus p|}{|b|}\right).$$

This measure assigns equal importance to false

positives and false negatives, so that trivial or random predictions incur an error of 0.5 on average. Since we do not know the correspondence between circles in B and P, we compute the optimal match via linear assignment by maximizing: $\max_{f:P\to B} \frac{1}{|f|} \sum_{p\in dom(f)} (1-BER(p, f(p)))$, where f is a (partial) correspondence between P and B. That is, if the number of predicted circles $|P|$ is less than the number of benchmark circles $|B|$, then every circle $p \in P$ must have a match $b \in B$, but if $|P| > |B|$, we do not incur a penalty for additional predictions that could have been circles but were not included in the ground-truth.

2) F-measure

To measure whether a predicted circle p and a ground-truth circle b are matched or not, we compute neighborhood affinity score (NA) between the two circles, $NA(p,b) = \frac{|b \cap p|^2}{|b| \times |p|}$. If $NA(p,b) \geq \omega$, then they are considered to be matching, and ω is usually set as 0.2. We use N_{cp} to represent the number of predicted circles which match at least a real circle. The mathematic expression is $N_{cp} = \left|\{ p \mid p \in P, \exists b \in B, NA(p,b) \geq \omega \}\right|$. N_{cb} is the number of real circles which match at least a predicted one, the mathematic expression is $N_{cb} = \left|\{ b \mid b \in B, \exists p \in P, NA(p,b) \geq \omega \}\right|$. Precision and recall are defined as [14]:

$$Precision = \frac{N_{cp}}{|P|} \qquad Recall = \frac{N_{cb}}{|B|}$$

F-measure which is the harmonic mean of precision and recall is defined as:

$$F-measure = 2 \times Precision \times Recall / (Precision + Recall).$$

It is used to evaluate the overall performance of the predict results.

C. Our Results

1) Parameter analysis

Our algorithm employs two parameters, including similarity threshold parameter τ used to control clustering granularity and balance parameter α used to balance the effect of topological structure and profile information. Since the preference τ is inherited from link community algorithm, we didn't analyze it in this paper and just set it as the average similarity of edge pairs. To study how the parameter α affects the results, we use 0. 1 as the step length and increase α from 0.1 to 0.9 step by step.

To explore the relationship between α and the number of predicted circles, we randomly choose 5 ego networks to observe the changing trends. The experimental results displayed in Fig. 3 illustrate that with the increment of α, the number of predicted circles increases too. This is because of the way we define the topological similarity and profile similarity. We sort topological similarities and profile similarities of all the adjacent edges in an ego network, and find that profile similarity increases slowly. An example is shown in Fig. 4. We utilize the average topological similarity and the average profile similarity of edges as the thresholds to cut topological similarity curve and profile similarity curve respectively. Much more edge pairs are considered dissimilar in the topological similarity curve, and this may cause relatively small circles. Parameter α is the weight of topological similarity, so when we set similarity threshold parameter τ as the average similarity of edges, the bigger α is, the more dissimilar edge pairs are. An illustration is displayed in Fig. 5. Then we study how the α affects 1-BER and F-measure. The experimental results are displayed in Fig. 6 and Fig. 7. In Fig. 6, changing regulations between α and 1-BER are not very clear. The same situation is observed in Fig. 7. This is because the profile and topological information are not the only effective factors for users to organize their circles. Many circles are created using information that users do not explicitly make available on social networks, such as a circle of 'best friends'. So we can't give a precise relationship between α and the accuracy evaluation metrics. However, we notice that in our data set, when α is small, say 0.2 or 0.3, we get the higher 1-BER and F-measure. So we set $\alpha = 0.3$ in our experiments.

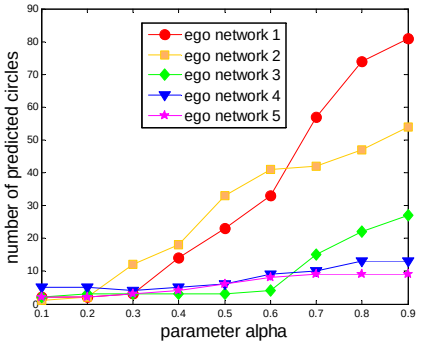


Fig. 3. The relationship between parameter α and the number of predicted circles. 5 ego networks are employed to illustrate the changing trend.

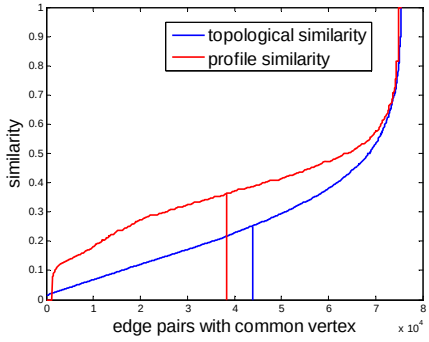


Fig. 4. The increment curves of topological similarity and profile similarity in an ego network. Vertical lines represent cutting these two curves with their average similarity. The average similarity of topological similarity curve is 0.25, and the similarities of 44000 edge

pairs are lower than 0.25. The average similarity of profile similarity curve is 0.36, and the similarities of 38300 edge pairs are lower than 0.36.

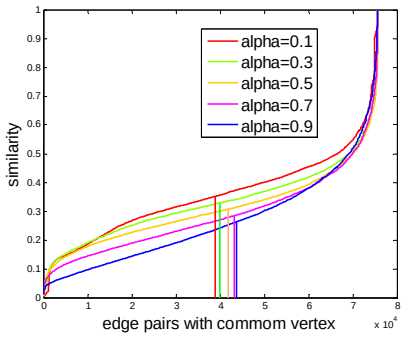


Fig. 5. The increment curves of similarity with different parameter α in an ego network. Vertical lines represent cutting these curves with their average similarity respectively. With the increment of α, there are more and more edge pairs whose similarity is lower than average similarity appear.

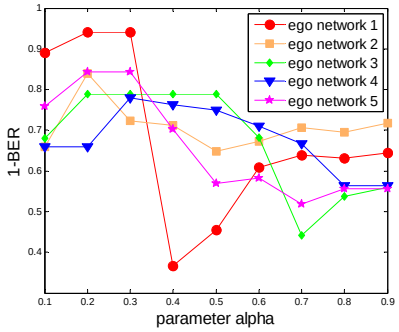


Fig. 6. The relationship between parameter α and 1-BER.

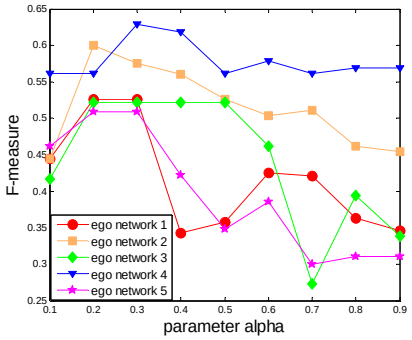
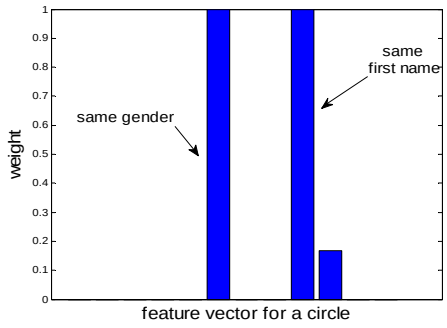


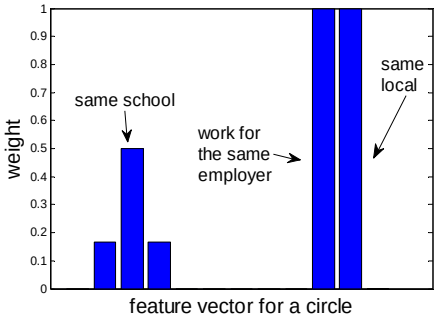
Fig. 7. The relationship between parameter α and F-measure.

2) Circles detected by our algorithm

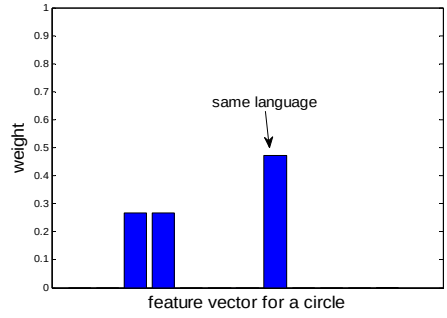
On average, our algorithm identified 21 circles in 10 ego networks, with an average circle size of 23. It is very close to the real situation. Our 1-BER score is 0.76 and f-measure score is 0.52. Fig. 8 shows the circle feature vectors of 4 circles we detected. People in Fig. 8 (A) have the same gender and they are relatives. People in Fig. 8(B) are colleagues and they live in the same town. People in Fig. 8(C) are friends who speak the same language. People in Fig. 8(D) are schoolmate. They are concentrating on the same specialty, learning in the same school and having the same level of education and living together.



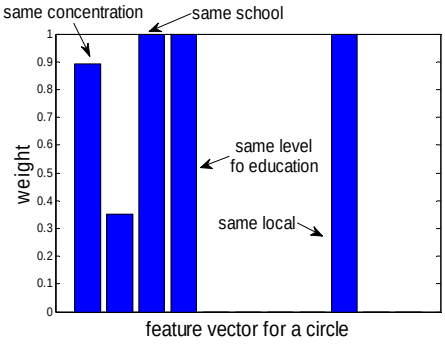
(A)



(B)



(C)



(D)

Fig. 8. Circle feature vectors of four circles.

D. Experimental Results Comparative Evaluation

In this section, we compare the performance, including accuracy and running time, of our algorithm with link community, low-rank embedding and probabilistic model algorithm.

Fig. 9 shows the results of the comparison of our

algorithm and three competitive approaches on 1-BER and F-measure. The 1-BER score of our algorithm is 0.76, which is 26.7% and 20.6% higher than link community and low-rank embedding. The F-measure of our algorithm is 0.52, which is 205.9%, 188.9% higher than link community and low-rank embedding. In the respect of accuracy, the probabilistic model is better than us. Our 1-BER score and F-measure are 9.5% and 11.9% lower than that of probabilistic model.

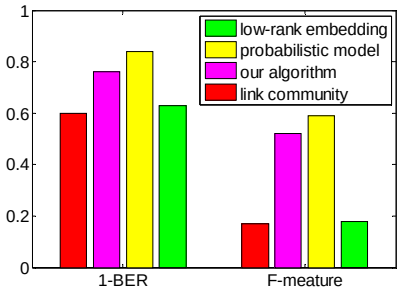


Fig. 9. The comparisons between link community, probability mode, low-rank embedding and our algorithm on 1-BER and F-measure.

Fig. 10 displays the running time of 4 algorithms. We choose 3 ego networks with different scales. When the vertex number is 66, the running time of link community, low-rank embedding and our approach is less than 1 second, while probabilistic mode needs 5 seconds. When the vertex number is 159, our algorithm needs 10 seconds, which is only a little more than link community. The running time of probabilistic mode is 79 seconds at this scale. When the scale of the network is 1046, the running time of our algorithm is 1832 seconds. It is almost twice longer than that of low-rank embedding. Probabilistic model approach needs almost 3 hours to execute the computing process.

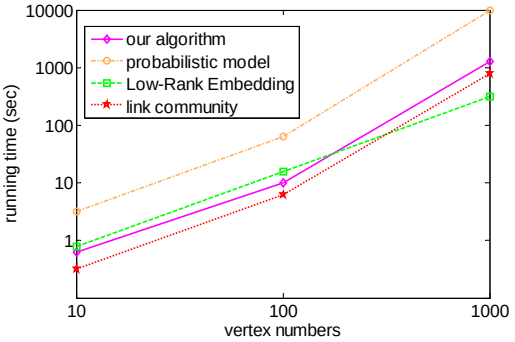


Fig. 10. The comparisons between link community, probability mode, low-rank embedding and our algorithm on running time.

IV. Conclusion and Discussion

One important feature of personal social networks is the ability to group friends and contacts into circles. Creating and maintaining these circles is a time consuming business. In this paper, we propose an efficient algorithm to analyze a user's entire list of contacts and automatically divide them up into useful circles. We improve link community algorithm by adding users' profile information to detect circles. And calculate what features cause the emergence of a circle.

The experimental results show that our algorithm is more accurate than link community and low-rank embedding method. Although our algorithm is less accurate than probabilistic mode approach, it is comparable. And the running time we need is far less than that of probabilistic mode approach. Our method gets a good balance between accuracy and efficiency. Experimental results demonstrate that characterizing social circles from the respect of edge is reasonable. Our proposed method is an effective algorithm in identifying social circles.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 60933009, 91130006, 61100157 and 61201416), Provincial Social Science Foundation of Shaanxi (No. 11M016) and the Fundamental Research Funds for the Central Universities (No. K5051106004).

Reference

[1] J. Scott, *Social Network Analysis: A Handbook* , 2000, SAGE Publications, London, UK.

[2] S. Wasserman and K. Faust, Social network analysis,1994, Cambridge University Press, Cambridge, UK.

[3] K. P. Reddy, M. Kitsuregawa, P. Sreekanth, and S. S. Rao, "A Graph Based Approach to Extract a Neighborhood Customer Community for Collaborative Filtering," In Proceedings of the Second International Workshop on Databases in Networked Information Systems, 2002, pp. 188-200.

[4] M. Girvan and M. Newman, "Community structure in social and biological networks," Proc Natl Acad Sci, Vol. 99, 2002, pp. 7821-7826.

[5] Xianghua Fu, Chao Wang, Zhiqiang Wang and Zhong Ming, "Threshold Random Walkers for Community Structure Detection in Complex Networks," Journal of Software, 2013, Vol. 8, No. 2, pp. 286-295

[6] Yueping Li, Yunming Ye and Xiaolin Du, "A New Vertex Similarity Metric for Community Discovery: a Local Flow Model," Journal of Software, 2011, Vol. 6, No. 8, pp. 1545-1553

[7] Chengying Mao, "A Heuristic Algorithm for Bipartite Community Detection in Social Networks," Journal of Software, 2012, Vol. 7, No. 1, 204-211

[8] T. Yoshida, "Toward finding hidden communities based on user profiles," In Proceeding of the IEEE International Conference on Data Mining Workshops, 2010. pp. 380-387

[9] Stephan Günnemann, Brigitte Boden , Thomas Seidl, "Finding density-based subspace clusters in graphs with feature vectors," Data Mining and Knowledge Discovery, 2012, vol.25, pp. 243-269

[10] Julian McAuley and Jure Leskovec, "Learning to discover social circles in ego networks," Neural Information Processing Systems, 2012, pp. 548-556

[11] Y. Y. Ahn, J. P. Bagrow and S. Lehmann, "Link communities reveal multiscale complexity in networks," Nature, 2010, 466:761

[12] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," Journal of the American

Society for Information Science and Technology, 2007, 58 (7): pp. 1019-1031
[13] Y. Chen and C. Lin, "Combining SVMs with various feature selection strategies," Studies in Fuzziness and Soft Computing, 2006, Vol. 207, pp. 315-324
[14] C. J. van Rijsbergen, Information Retrieval, Butterworth 1979

Yu Wang is currently a PhD student at the School of Computer Science and Technology of Xidian University Xi'an, China. She received her Master of Management from the School of Economics and Management of Xidian University (2006) and her Bachelor of Management from School of Economics and Management of Xidian University (2003). Her research interests include data mining and complex network.

Lin Gao received the B.Sc and M.Sc. in Computational Mathematics from Xi'an Jiaotong University and Northwest University in 1987 and 1990, respectively, and the. Ph. D. degree inCircuit and System from Electronics Engineering Institute from Xidian University in 2003. She was a visiting scholar at University of Guelph, Canada from 2004 to 2005. At present, she is a professor in the Department of Computer Science and Technology, Xidian University. Her research interests include bioinformatics, data mining in biological data, graph theory and intelligence computation.