Community Mining from Signed Social Networks

Bo Yang, William K. Cheung, and Jiming Liu

Abstract—Many complex systems in the real world can be modeled as signed social networks that contain both positive and negative relations. Algorithms for mining social networks have been developed in the past; however, most of them were designed primarily for networks containing only positive relations and, thus, are not suitable for signed networks. In this work, we propose a new algorithm, called FEC, to mine signed social networks where both positive within-group relations and negative between-group relations are dense. FEC considers both the sign and the density of relations as the clustering attributes, making it effective for not only signed networks but also conventional social networks including only positive relations. Also, FEC adopts an agent-based heuristic that makes the algorithm efficient (in linear time with respect to the size of a network) and capable of giving nearly optimal solutions. FEC depends on only one parameter whose value can easily be set and requires no prior knowledge on hidden community structures. The effectiveness and efficacy of FEC have been demonstrated through a set of rigorous experiments involving both benchmark and randomly generated signed networks.

Index Terms—Community mining, agent-based approach, random walk, signed social networks.

1 Introduction

Many complex systems in the real world can be modeled as social networks [1], [2]. A social network can be defined as a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of vertices, and E is the set of edges connecting pairs of vertices. For example, in a human social network, each vertex denotes an individual or a node, and each edge denotes a relation or a link between two nodes. In weighted social networks, each link is attached with a real number called *weight*. In the field of social science, the networks that include only positive links are also called *positive social networks*, and the networks with both positive and negative links are called *signed social networks* [3] or *signed networks* for short. Positive links in signed networks denote a "positive relationship" such as "friendship," whereas negative links may denote a "negative relationship" such as "hostility." For example, in the Gahuku-Gama subtribes network [4], the weights of positive links denote rankings of "political alliance relation." In contrast, the weights of negative links denote the rankings of "political opposition relation." Analyzing and mining the signed networks created based on the relationships can help gain further understanding on the characteristics of the social networks (see Section 3.2).

Community structure is an important topological property of social networks. Positive social network communities are defined as the groups of nodes within which the links are dense but between which the links are less dense [5]. Signed network communities are defined not only by the density of links but also by the signs of links. Qualitatively speaking, they are defined as the groups of nodes within which the positive links are dense and between which negative links are also dense. The challenge of mining signed network communities lies in the fact that it is natural to have some negative links within groups and, at the same time, some positive links between groups. Finding the best and yet robust partition of the network for discovering the hidden community structure is by no means straightforward.

1.1 Related Studies on Social Network Analysis

In essence, the problem of network community mining can be translated into that of partitioning a fully connected graph with positive and negative links into clusters such that the intracluster links are positive, and the intercluster links are negative. This problem is known as *correlation clustering* [6], [7], where positive and negative links correspond to agreements and disagreements between nodes (or items), respectively. In [6], [7], Bansal et al. provided a constant-factor approximation for minimizing the number of disagreements and a polynomial-time approximation scheme (PTAS) for maximizing the total number of agreements (that is, intracluster positive links and intercluster negative links) that can achieve any constant error gap in polynomial time. Their algorithm can be extended to the case of minimizing weighted (that is, real-valued) disagreements or maximizing weighted agreements.

Also related to social-network-based interrelationship analysis are studies on trust evaluation and propagation in a distributed environment. Kamvar et al. [8] studied a peer-to-peer file-sharing network, in which a peer assigns a trust value to those peers who have provided it with authentic files. They proposed an EigenTrust algorithm that considers the entire history of uploads with individual peers by aggregating the normalized local trust values of *all* the users (for example, the trust values assigned by friends' friends).

- *B. Yang is with the College of Computer Science and Technology, Jilin University, Changchun, China 130012. E-mail: ybo@jlu.edu.cn.*
- *W.K. Cheung and J. Liu are with the Department of Computer Science, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: {william, jiming}@comp.hkbu.edu.hk.*

Manuscript received 18 Dec. 2005; revised 23 Oct. 2006; accepted 9 Apr. 2007; published online 2 May 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0552-1205.
Digital Object Identifier no. 10.1109/TKDE.2007.1061.

The resulting aggregated global trust vector can be computed by calculating the left principal eigenvector of a matrix of normalized local trust values. In this vector, each element reflects a unique global trust value that the system as a whole places on a peer. Using the distributed EigenTrust algorithm, all peers in the network can update and store the global trust vector with minimal overhead.

Based on a general framework of trust propagation scheme, Guha et al. [9] addressed the problem of predicting the trust between any two people (nodes) in a social network connected by ratings or trust/distrust scores. The scheme is formulated based on a weighted chaining of four well-defined atomic propagations (matrix operations), that is, direct propagation, cocitation, transpose trust, and trust coupling, to a set of belief (represented as a matrix) that users hold about each other. Based on this chaining, a final matrix that contains the trust or distrust of any two people can be derived after a number of propagations. Experimental validation using the Epinions data has shown that with a small number of expressed trusts per individual, it is possible to predict the trust between any two people in the system.

1.2 Related Algorithms for Detecting Network Communities

In the literature, many algorithms have been developed to detect positive network communities. They can generally be divided into three main categories: 1) graph theoretic methods such as random walk methods and physics-based methods [10], [11], [12], the spectral methods [13], [14], [15], [16], and the Kernighan-Lin algorithm [17], 2) hierarchical methods such as agglomerative methods based on the structural similarity metrics [18], [19], [20] and divisive methods based on betweenness metrics, like the Girvan-Newman (GN) algorithm [5], the Tyler algorithm [21], and the Radicchi algorithm [22], and 3) methods for detecting hyperlink-based Web communities such as the maximum flow communities (MFC) algorithm [23], the hyperlink-induced topic search (HITS) algorithm [24], the spreading activation energy (SAE) algorithm [25], and others [26], [27]. In this section, we provide details on some representative algorithms that are more related to our work.

Pons and Latapy [10] reported a community finding method using random walk. It starts with single-node communities and repeatedly performs the merging of a pair of adjacent communities that minimizes the mean of the squared distances between each node and its community. At each step, the distance between adjacent communities is updated. Here, the notion of the distance between two nodes corresponds to their similarities, that is, the directional transition probabilities between them in a discrete random walk process.

There have been other studies on community identification from complex networks that utilize physics-based methods. For example, Palla et al. [11] recently introduced the concept of *clique percolation* to the problem of identifying communities, where one node can belong to more than one community (like many social networks). They viewed a community as a union of all k-cliques (that is, complete subgraphs of size k) and studied the statistical features (for example, the cumulative distributions of the community size, community degree or number of overlap links, overlap size, and membership number or number of communities) of the interwoven sets of overlapping communities involving highly overlapping cohesive groups of nodes. The proposed method first identifies all cliques of the network and performs a standard component analysis of the clique-clique overlap matrix to discover a set of k-clique-communities.

As a common method used in identifying groups from a network, Kim and Jeong [12] developed a *matrix block diagonalization* and applied it to weighted stock networks. Their method constructs a network of stocks and identifies stock groups with a percolation approach based on a filtered empirical stock correlation matrix. The eigenvector-based filtering method removes the random noise and the marketwide effect. In the percolation approach, the sequence of stocks for block diagonalization is optimized by considering the "attraction force" (correlation) between them and minimizing the total "energy" (measuring the compactness of grouping correlated items) for a stock sequence.

Finding optimal cuts in a graph has been shown to be NP-complete [16], and strategies for computing approximate solutions efficiently are needed. For instance, the spectral methods solve the problem through calculating the eigenvector with the second smallest eigenvalue of the Laplacian matrix or its variants. In general, computing eigenvectors of a matrix takes $O(n^3)$ time. For sparse matrices, the second smallest eigenvector can be computed with the time complexity of $O(m/(\lambda_3 - \lambda_2))$ by using the Lanczos method [28], where λ_2 and λ_3 are the second and third smallest eigenvalues of the Laplacian matrix, respectively, and dominate the performance of spectral methods. They will produce a better approximation of the best cut when λ_2 is large, and they will run very slowly when the gap between λ_2 and λ_3 is small.

Doreian and Mrvar [3] presented an approach to partitioning a signed network by using the local search method. In essence, their approach is a greedy optimization algorithm, like the Kernighan-Lin algorithm, which divides a graph into several parts so as to minimize a predefined error function: the sum of the absolute weight values of within-group negative links and between-group positive links. The algorithm first randomly partitions all the nodes of a signed network into a given number of groups and computes the error based on the initial partition. To minimize the error function, it moves nodes from one group to another or interchanges the group IDs of nodes in different groups. The node moving process repeats until the error cannot be further decreased in the local sense. Inherited from the Kernighan-Lin algorithm, the algorithm runs moderately fast, with a time complexity of $O(iteration \times n^2)$, where *iteration* is the number of rounds of node moves. One limitation of this algorithm is that it needs to know the number of groups beforehand in order to initialize a reasonable starting partition. Like the Kernighan-Lin algorithm, this algorithm is also sensitive to the initial partition. Starting with a good initial layout, it will find a locally optimal partition. Otherwise, it may not. In addition, this algorithm takes into account only the signs of links for partitioning signed networks, neglecting

the density of links, which, in many cases, is a salient feature in partitioning.

1.3 Our Contributions

Although there have been many techniques proposed for social network analysis, most of them focus merely on link density but not on the signs of links as their clustering attributes. Applying the existing techniques to signed networks, say, by adding a large-enough positive number to the weight of each link, is not feasible, since the transforming process will inevitably change the underlying connectivity structure of the original network (for example, some negative relationships might become positive relationships and, thus, associated nodes that should be separated from each other might be aggregated together), hence resulting in unreasonable clustering results.

In this paper, we present a novel efficient algorithm that considers both link density and signs for mining signed network communities. The proposed method adopts an agent-based approach for formulating the community identification problem. Based on the notion of sink node that we introduce, a heuristic is proposed as an iterative equation for estimating the probability for an agent starting from an arbitrary node of the network to reach the sink.

The underlying assumption is that it should be difficult for an agent to "move" across its own community boundary, whereas it should be easy for it to reach other nodes within its community, as link density within a community should be high. As compared to Doreian and Mrvar's method [3], the proposed method has been demonstrated via a set of rigorous experiments to be more effective regarding its community identification accuracy. Also, it is efficient, with a time complexity that is linear with respect to the number of nodes and links. In addition, it requires no prior knowledge on the community structure (for example, the number of groups and a good initial partition), making itself easier to be applied to different networks.

The remaining parts of this paper are organized as follows: Section 2 gives the problem definition and the related background. Then, the details of the proposed method are described. Section 3 provides experimental results obtained by applying the proposed method to a number of benchmark signed networks. Also, some important experiments, including a sensitivity analysis on a key parameter, are presented. Finally, Section 4 concludes the paper by highlighting the major advantages of our method and future work.

2 Mining Signed Social Networks

2.1 Problem Definition

The community structure of a signed network is characterized by the way in which the nodes of the network are connected. A signed network is called *partitionable* or *clusterable* if it has a K-way partition. A K-way partition $\pi = \{P_1, P_2, \cdots, P_K\}$ of a signed network is defined such that the following conditions hold:

$$\begin{cases} w_{ij} > 0 & < v_i, v_j > \in E \wedge (v_i \in P_k) \wedge (v_j \in P_k) \\ w_{ij} < 0 & < v_i, v_j > \in E \wedge (v_i \in P_k) \wedge (v_j \in P_l) \wedge (k \neq l), \end{cases}$$

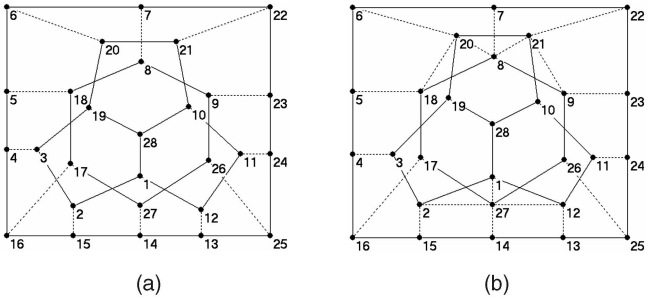


Fig. 1. The schematic representation of two illustrative signed networks. In the figures, solid lines denote positive links, and dashed lines denote negative links. The node labels are randomly assigned. The network shown in (a) is partitionable because it can be divided into three groups including (6, 7, 22, 23, 24, 25, 13, 14, 15, 16, 4, 5), (8, 9, 26, 27, 17, 18), and (20, 21, 10, 11, 12, 1, 2, 3, 19, 28). In the three-way partition, the weights of links within groups are all positive, and the weights of links between groups are all negative. The network is also balanced because there exits a reasonable two-way partition: (6, 7, 22, 23, 24, 25, 13, 14, 15, 16, 4, 5) and (8, 9, 26, 27, 17, 18, 20, 21, 10, 11, 12, 1, 2, 3, 19, 28). The network shown in (b) is also partitionable because there exists the same three-way partition as that of (a). However, it is not balanced because we cannot find a two-way partition that can bipartition it according to the definition mentioned above.

where w_{ij} is the weight of link $< v_i, v_j >$ and $1 \leq k, l \leq K$. Thus, for the ideal case, if the signed network is partitionable, the issue of mining signed network communities can be defined as equivalent to finding a K-way partition. In addition, a signed network is called *balanced* if it has a two-way partition. Fig. 1 gives two examples. The network shown in Fig. 1a is *partitionable* and *balanced*. The network shown in Fig. 1b is *partitionable* but not *balanced*.

Intuitively, identifying communities in partitionable or balanced signed networks can easily be achieved by cutting all the negative links. The subgraphs obtained will contain only positive links and form communities. However, there are scenarios that make the identification task nontrivial: 1) the signed social networks are nonpartitionable and 2) the signed social networks are partitionable, but the best (or the most natural) partition cannot be arrived at only by cutting negative links. In fact, many signed networks often result in several large subgraphs accompanied with a few isolated nodes after all negative links are cut out. More robust subgraph partitions should properly disregard and retain some positive and negative links (or properly discount their influence) so as to identify more natural communities.

In the literature, there exist a number of robust graph clustering methods for positive networks, as stated in Section 1. However, as mentioned above, they are not suitable for the networks with both positive and negative weights. Thus, we need some more effective network analysis methods for analyzing signed networks.

2.2 The Main Idea

A signed network is often composed of communities, where there are many positive links within communities but not too many between them. To extract communities from such a signed network, our main idea is to start with an arbitrary node and then extract its associated community by exploiting its connectivity with other nodes in the network. In particular, an agent-based approach is adopted for modeling and solving the problem. An agent performs a

TABLE 1
The FEC Algorithm for Community Mining from a Signed Network

Algorithm FEC (A_0, A)
A_0: Input, the initial adjacency matrix of a signed network;
A: Output, the final adjacency matrix of a signed network with identified communities;
1. FC(A_0, A); /* for finding a community to be described in Section 2.3 */
2. EC(A, pos); /* for extracting a community to be described in Section 2.4 */
3. If $pos = dim(A)$ then return A;
4. Divide A at pos such that $A = [A_{11} \quad A_{12}; A_{21} \quad A_{22}]$;
5. FEC (A_{11}, A_U);
6. FEC (A_{22}, A_L);
7. $A = [A_U \quad A_{12}; A_{21} \quad A_L]$ and return A.

random walk starting from one node and carries on for a number of steps. At each step before visiting the next node, the agent makes the selection based on the transition probability distribution that depends on the degree of connectivity.

Due to the topological structure of a community, the aggregated probability for remaining in the same community, that is, an agent starts from any node and stays in the same community after a number of transitions, is greater than that for going out to a different community. Thus, based on this observation, we can find communities by examining localized aggregated transition probabilities. That is, the greater such probabilities, the more likely the existence of communities.

The aggregated transition probability can be computed through iterative operations on the adjacency matrix of the graph. In our present work, we have proposed an efficient method for identifying communities in signed networks. It contains two main phases: 1) the FC phase transforms the adjacency matrix to compute their transition probability vector and sorts them for each row and 2) the EC phase applies a cutoff criterion to the transformed adjacency matrix and divides it into two block matrices, which correspond to two subgraphs. One of these subgraphs is the identified community, and another is the matrix to be processed, recursively, following the above-mentioned two phases.

Table 1 provides an overview of our proposed algorithm FEC for identifying hidden communities in a signed social network. Note that the algorithm is a recursive one: Communities identified early on will be further analyzed until no more communities can be found.

2.3 Finding a Community

2.3.1 Algorithm

An agent-based approach is here adopted to model the problem of finding the community that contains a specific node (which can be randomly chosen). An imaginary agent walks freely from one node to another, following the links of a given network. The agent's walk can be viewed as a stochastic process defined based on the links' attributes. In particular, when the agent arrives at a node, it will select one of its neighbors at random and then go there. Let p_{ij} be the probability of the agent walking from node i to its neighbor, node j. In an unsigned, unweighted, and undirected network, this probability can simply be computed as

$$p_{ij} = \frac{1}{d_i}, \tag{2.1}$$

where d_i is the degree of node i. Note that the sum of the probabilities of the links going out from node i will be 1; that is, $\sum_j p_{ij} = 1$. Here, we define the destination node of the agent as the *sink node*.

Then, let $P_t^l(i)$ be the probability that the agent starting from node i can eventually arrive at a specific sink node t within l steps. The value of $P_t^l(i)$ can be estimated iteratively by

$$P_t^l(i) = 1 \cdot I_{\{i=t\}} + \sum_{<i,j>} p_{ij} \cdot P_t^{l-1}(j) \cdot I_{\{i \neq t\}}, \tag{2.2}$$

where $< i, j >$ denotes the link connecting nodes i and j, and note that $< i, j >$ should be replaced by $i \rightarrow j$ for directed graphs. As the link density within a community is, in general, much higher than that between communities, agents that start from nodes within the community of a sink node should have more paths to choose from in order to reach the sink node within some l steps, where the value of l cannot be too large. On the contrary, agents that start from nodes outside the community of the sink node have a much lower probability of eventually arriving at the sink. In other words, it will be hard for an agent that falls on a community to pass those "bottleneck" links and leave the existing community. Mathematically speaking, $P_t^l(i)$ should follow the property as

$$\underset{i \in C_t}{\forall} \underset{j \notin C_t}{\forall} \{P_t^l(i) > P_t^l(j)\}, \tag{2.3}$$

where C_t denotes the community where node t is situated.

Based on this idea, the procedure for finding the community containing a specific sink node t can be described as follows: 1) calculate $P_t^l(i)$ for each node i and 2) rank all the nodes according to their associated probability values. After the two steps, the members of the community that we want should be ranked high in step 2. Then, by properly setting a cutoff point (to be explained in the next section), we can extract the community's nodes, together with their links from the original network.

The method discussed above can easily be extended to deal with weighted networks by replacing (2.1) with the following:

$$p_{ij} = \frac{w_{ij}}{D_i}, \tag{2.4}$$

where w_{ij} is the weight of link $< i, j >$ and D_i is the weighted degree of node i that can be calculated as

$$D_i = \sum_{<i,j>} w_{ij}. \tag{2.5}$$

To further extend it to deal with weighted and directed networks, (2.1) can be replaced by the following:

$$p_{ij} = \frac{w_{ij}}{k_i}, \tag{2.6}$$

where w_{ij} is the weight of the link $i \to j$, and k_i is the weighted outdegree of node i, given as

$$k_i = \sum_{i \to j} w_{ij}. \tag{2.7}$$

For signed networks, the agent-based approach described above needs to be modified slightly, as negative probability is not defined. We can restrict the agent's walk only along the positive links during the entire process. By this modification, (2.1) for calculating the probability p_{ij} can be replaced by the following:

$$p_{ij} = \frac{\max(0, w_{ij})}{K_i}, \tag{2.8}$$

where K_i is the positive weighted outdegree of node i, given as

$$K_i = \sum_{<i,j>} \max(0, w_{ij}). \tag{2.9}$$

Note that although this step looks equivalent to simply removing the negative links, the influence of negative links remains important for computing a robust cutoff for community extraction, as will be discussed in the Section 2.4.

Let $\psi^{(l)} = (P_t^l(i))_{n\times 1}$ denote the l-step transfer probability distribution and $A = (w_{ij})_{n\times n}$ denote the adjacency matrix of a signed network containing n nodes. One can then rewrite (2.2) in a matrix form as

$$\psi^{(l)} =_{(t)} (B\psi^{(l-1)}), \tag{2.10}$$

where

$$B = (B_{ij})_{n\times n} = (\max(0, A_{ij}) / \sum_j \max(0, A_{ij}))_{n\times n}.$$

For a given vector V, the ith component of the new vector $({}_{(t)}V)$ is defined as $({}_{(t)}V)(i) = 1 \cdot I_{\{i=t\}} + V(i) \cdot I_{\{i\neq t\}}$.

There is no special requirement for selecting a sink: We can select one at random. However, in order to obtain a determinate result, we set the sink to be always node 1 without loss of generality. In this case, the iterative form for computing the l-step transfer probability distribution of an agent's random walk is given as follows:

$$\begin{cases} \psi^{(0)} = (1, 0, \cdots, 0)_{n\times 1} \\ \psi^{(l)} =_{(1)} (B\psi^{(l-1)}). \end{cases} \tag{2.11}$$

Based on the above discussions, the main steps of the FC subroutine for identifying the community associated to node 1 are listed as follows:

Step 1. Compute the l-step transfer probability distribution $\psi^{(l)}$.

Step 2. Transform the initial adjacency matrix A_0 into an output adjacency matrix A by sorting the nodes based on their corresponding values in $\psi^{(l)}$.

Proposition 2.1. *The time complexity of the FC subroutine is* $O(l(n+m))$*, where* l *is the number of steps that the agent travels, and* n *and* m *correspond to the numbers of nodes and links of the signed network, respectively.*

Proof. Step 1 for computing $\psi^{(l)}$ is the most computationally costly step. Each component of $\psi^{(l)}$ is updated using (2.2). For node i with d_i neighbors, FC has to spend $O(d_i)$ time for getting $P_t^{l-1}(j)$ of all neighbors and $O(1)$ time for calculating $P_t^l(i)$. Therefore, the corresponding time complexity is the time spent in Step 1, which is bounded by

$$\sum_{i=1}^{n} O(d_i + 1) = O\left(\sum_{i=1}^{n} d_i + \sum_{i=1}^{n} 1\right) = O(m+n).$$

Step 2 sorts all nodes according to their probability values, and linear-time sort algorithms such as bin sort or counting sort can be adopted. Thus, the overall complexity of FC is $O(l(n+m))$. □

l is the only parameter required by the FC subroutine. Obviously, the value of l should not be too small because walking only for a few steps will exclude the agents that start a bit far from the sink node. However, if the value of l is too large, then the agent can arrive at any specified sink from any node, as far as there exists a path without any negative link leading to that node. Thus, we have

$$\psi^{(l)} \to (1)_{n\times 1} \quad \text{when} \quad l \to \infty.$$

In this case, the probability values of all the nodes will be almost identical and, thus, we will have no way to distinguish the communities. One reasonable choice for l can be made with the aid of the average distance of a network. The distance between two nodes refers to the number of links along the shortest path connecting them.

As most social networks are small-world networks, the average distance between two nodes has shown to be around 6, according to the theory of "six degrees of segregation [29]." For scale-free networks, the average distance is usually small too. The World Wide Web is one of the biggest scale-free networks that we have found so far. However, the average distance of such a huge network is actually about 19 [30]; that is, we can get to anywhere we want through 19 clicks on the average. Thus, based on the above general observations, the good options for the value of l that we propose, in practice, should somehow fall in the range of

$$6 \le l \le 20.$$

In addition, the l-value that we are considering is actually the average distance between nodes *within* a community instead of the whole network. In most of our experiments, we have found that $l = 10$ is, in general, good enough for all the tested cases. We will also examine the validity of setting the interval to be [6, 20] via a sensitivity analysis, as further described in Section 3.5.

Note that the results of the FC subroutine can be visualized with the aid of a matrix. Let A_0 be the initial adjacency matrix of a signed network and A be the output matrix of the FC subroutine. Compared with A_0, A should be a much more regular matrix with a diagonal structure of two block matrices, A_{11} and A_{22}, as shown in Fig. 2. A_{11} and A_{22} correspond to two dense matrices with more positive elements, whereas the remaining parts of A, A_{12}, and A_{21} are sparse with more zero or negative elements. The

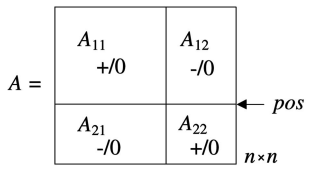


Fig. 2. The FC subroutine transforms the adjacency matrix of a signed network into one with a diagonal structure, in which A_{11} and A_{22} are two dense matrices with more positive elements, whereas the remaining contain more zero or negative elements.

subgraph corresponding to A_{11} is the community containing the sink node initially selected.

2.3.2 Examples of Applying FC to Signed Networks

To illustrate how the FC subroutine works, we have applied it to the two signed networks, as shown in Fig. 1, and the results are provided in Fig. 3. Nodes 1 and 6 are, respectively, chosen as the sink nodes in different cases. As compared with their initial matrices, the two output matrices are highly regular with two diagonal blocks, as shown in Figs. 3a and 3b. In the figures, the labels on the left-hand side denote the node indices, and those on the right-hand side are the values of the probability of being able to reach the sink. The top 10 nodes in the matrix in Fig. 3a compose the community (including node 1) corresponding to the set of nodes (1, 12, 2, 28, 11, 3, 19, 10, 21, 20) in Fig. 1a. The top 12 nodes in the matrix in Fig. 3b make up the community (6, 7, 5, 22, 4, 23, 16, 24, 15, 25, 14, 13), which is again consistent with what is shown in Fig. 1b.

This proposed FC subroutine can also work on positive social networks. We have applied it to two benchmark positive social networks that are widely used by other researchers [5], [22], [31]. Fig. 4a shows the karate club network, which represents the social interaction such as friendship between members within a karate club at an American university, as observed by Wayne Zachary in the 1970s [32]. The network contains two clubs of roughly equal size formed due to a dispute that arisen between the club's administrator, represented as squares, and its principal karate teacher, represented by circles, as shown in Fig. 4a. We select node 1, the administrator, as the sink. Fig. 4c gives the output of the FC subroutine, in which the top 16 nodes are the 16 members led by the administrator. Then, we select node 19 as the sink. By referring to Fig. 4d, the top nodes correspond to all the 18 members led by the teacher.

We have also applied FC to the network of a US college football association in the 2000 season [5]. The network includes 115 nodes and 613 links representing football teams and games among those teams, respectively. All the 115 teams are divided into 12 conferences. Generally, the matches played within a conference are much greater than those played between conferences. Therefore, each conference naturally forms a community of the network. Fig. 5a shows the adjacency matrix of the football association network. Fig. 5b shows the output matrix transformed by FC with node 8_ArizonaState selected as the sink. In Fig. 5b, we can see that all the 10 teams of the Pac10 conference, including ArizonaState on the top.

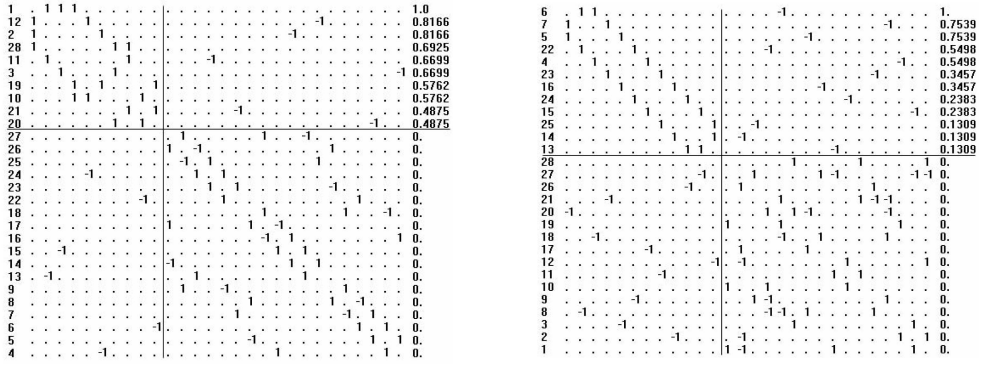
2.4 Extracting the Sink Community

Based on the ranked list of the nodes obtained from FC, the community that contains the specified sink can be distilled by properly setting the cutoff point along the ranked list of nodes. For partitionable and balanced networks, the cutoff point can be set so as to satisfy the partition condition:

$$C1 : P_t^l(\cdot) > 0. \tag{2.12}$$

That is, we extract only those nodes with nonzero value of $P_t^l(i)$ from the entire matrix. For instance, in Figs. 3a and 3b, all the nodes on the top are in the community (including sinks) and have the associated values of $P_t^l(i)$ greater than zero. However, those of the remaining nodes are all zero. $C1$ is also suitable for some nonpartitionable signed networks such as the Gahuku-Gama subtribes network and the Slovene parliamentary party network, as shown in Figs. 6a and 6b, respectively. Unfortunately, $C1$ does not work well for positive networks such as those in Figs. 4c, 4d, and 5b.

For a more robust criterion function for setting the cutoff, one can adopt the error function used in [3] for evaluating signed network clustering results, given as



(a) (b)

Fig. 3. Transforming the signed networks in Fig. 1 by the FC subroutine. (a) Transformed adjacency matrix for the signed network in Fig. 1a. (b) Transformed adjacency matrix for the signed network in Fig. 1b.

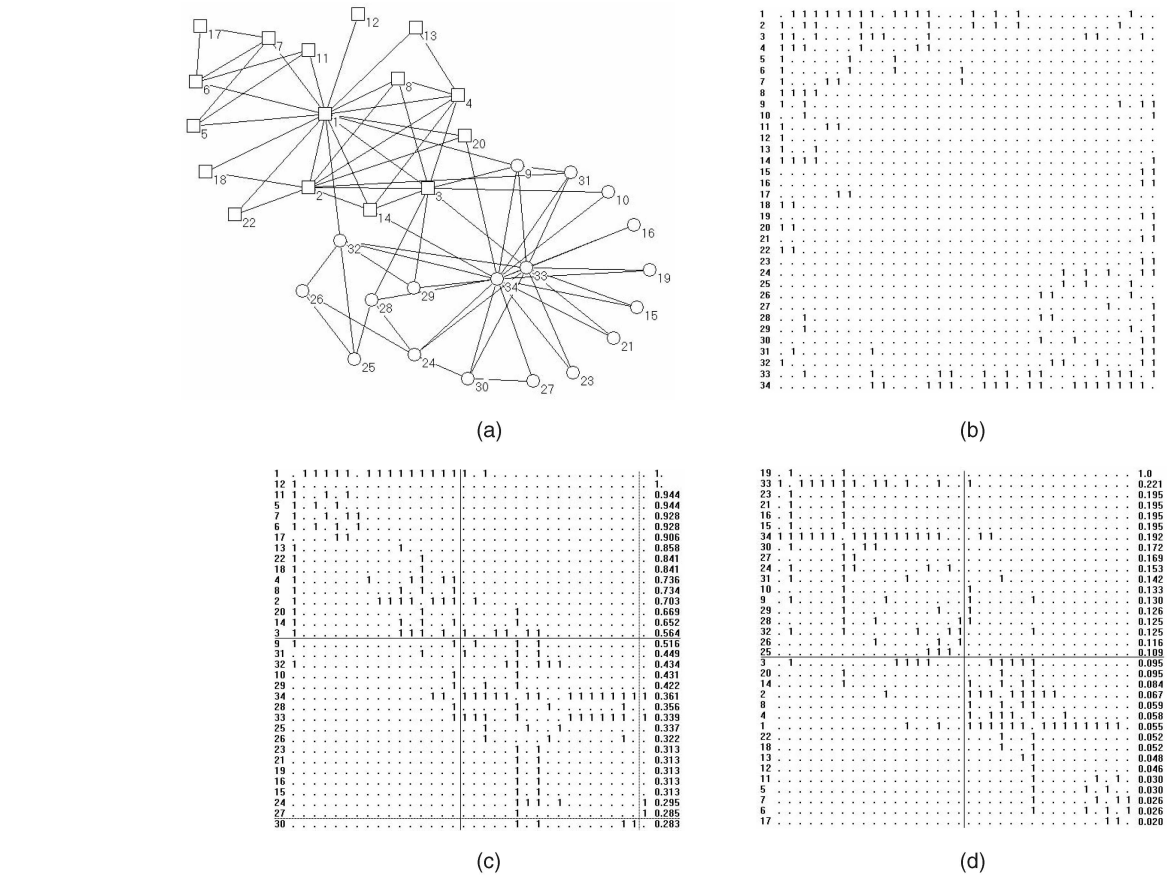
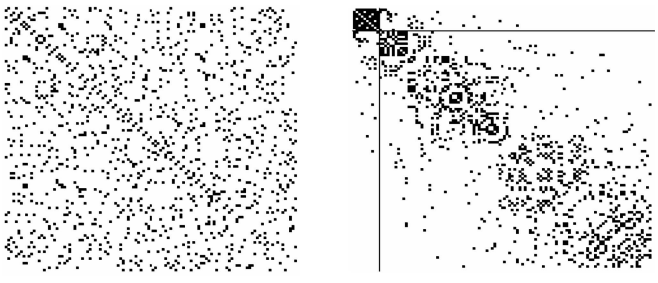


Fig. 4. Transform the karate club network by FC. (a) The karate club network. (b) The adjacency matrix of the karate club network. (c) Output matrix selecting node 1 as the sink. (d) Output matrix selecting node 19 as the sink.

$$P(C) = \sum_{k} \sum_{i,j \in C_k} \max(0, -w_{ij}) + \sum_{r \neq s} \sum_{i \in C_r, j \in C_s} \max(0, w_{ij}), \tag{2.13}$$

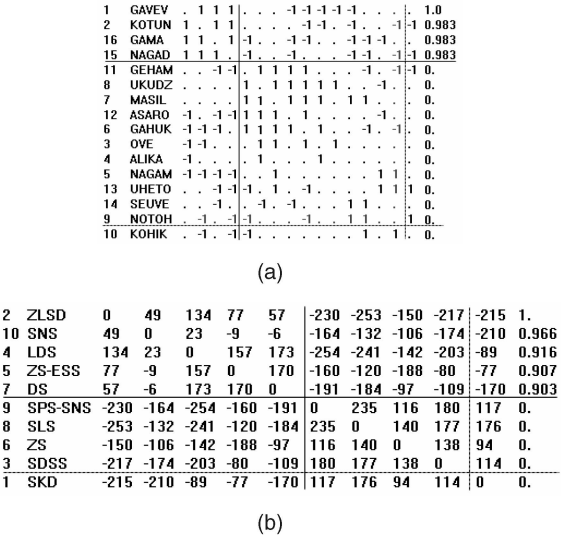
where C is a possible partition of the signed network, and w_{ij} is the weight of link $< i, j >$. This error function takes into account the signs of links for clustering signed networks, but the density of links, which is another very important criterion used for clustering positive networks, is neglected.

To take into account the link density, a number of criterion functions have been proposed for positive networks. For instance, one can perform graph clustering based on the minimization of a number of proposed cut criteria such as the *average cut* and *normalized cut* adopted



(a) (b)

Fig. 5. Transform the football association network by the FC subroutine. (a) The adjacency matrix of the football association network. (b) The output obtained by selecting node 8_ArizonaState as the sink. The dots in both matrices denote positive links.



(a) (b)

Fig. 6. The FC output of (a) the Gahuku-Gama subtribes network, with node 1 being the sink. (b) The Slovene parliamentary party network, with node 2 being the sink. In (a) and (b), the solid crosses denote the real partitions, and the dashed crosses denote the partitions obtained by normalized cut only based on link density.

by the spectral methods [13], [14], [15], [16]. In graph theory, a *cut* corresponding to a bipartition of a network with the node set V is defined as

$$cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w_{ij}, \tag{2.14}$$

where $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \phi$, and w_{ij} is the weight of the link $< i, j >$.

The optimal bipartition of a network is the one that minimizes the cut value, which is also called the *minimum cut*. In some cases, *minimum cuts* will lead to unnatural biased clustering results, where some partitions are simply isolated nodes. For example, in Fig. 4c, the optimal bipartition obtained by the *minimum cut* method is represented as the dashed cross, where node 30 is separated from the other nodes, and the cut value is 4. Obviously, this is not a good partition. In fact, the cut value corresponding to a more natural split of the matrix, as indicated by the solid cross, is found to be 10, which is much greater than 4.

To alleviate the limitation of the *minimum cut*, the *average cut*, and the *normalized cut* have been proposed, which both compute the density of links instead of the number of links, and are defined, respectively, as

$$Acut(V_1, V_2) = \frac{cut(V_1, V_2)}{|V_1|} + \frac{cut(V_1, V_2)}{|V_2|}, \tag{2.15}$$

$$Ncut(V_1, V_2) = \frac{cut(V_1, V_2)}{assoc(V_1, V)} + \frac{cut(V_1, V_2)}{assoc(V_2, V)}, \tag{2.16}$$

where $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \phi$, and

$$assoc(V_1, V) = \sum_{u \in V_1, t \in V} w(u, t).$$

Using these two cut criteria, we can find better partitions for the matrices shown in Figs. 4c and 4d, as indicated by the solid crosses. Direct adoption of these two criteria for signed networks is not good enough because they consider only link density, with the signs of links neglected. For example, in the matrices shown in Figs. 6a and 6b, the partitions obtained by these two criteria are represented as the dashed crosses. Compared with the real partitions represented by the solid crosses, they are far from satisfactory.

For the reasons discussed above, a new cut criterion for extracting a sink community from a signed network is proposed. In Fig. 2, matrix A is split into two block matrices, A_{11} and A_{22}, at the position pos, where A_{11} corresponds to the community containing the sink node. According to the definition of a community in a signed network, the within-community positive links should be dense, and the within-community negative links should be sparse. Now, let $1 \le i \le pos$, and

$$a = \sum_{1 \le j \le pos} \max(0, A_{ij}), \qquad b = \sum_{pos < j \le n} \max(0, A_{ij}),$$
$$c = \sum_{1 \le j \le pos} \min(0, A_{ij}), \qquad d = \sum_{pos < j \le n} \min(0, A_{ij}).$$

Note that a (c) corresponds to the sum of the positive (negative) elements of the ith row of the matrix A from column 1 to column pos. Note that a must be a positive value and c must be a negative value. $a + c$ can be interpreted as the sum of the weights of the positive links being discounted by the sum of the weight value of negative links within a community. A good partition should result in a very positive a and a not-so-negative c and, thus, the value of $a + c$ must be large. Also, between communities, we expect the positive links to be sparse and the negative links to be dense. b (d) corresponds to the sum of the positive (negative) elements of the ith row of the matrix A from column $pos + 1$ to column n. $b + d$ can then be interpreted as the sum of the weights of positive links being discounted by the sum of the weight value of negative links between the two communities. Again, a good partition should result in a not-so-positive b and a very negative d, and thus, the value of $b + d$ must be small. Thus, for the first row of A, we expect

$$a + c \ge b + d. \tag{2.17}$$

By requiring the first pos rows of A to satisfy this condition, (2.17) can simply be rewritten as

$$\sum_{1 \le j \le pos} A_{ij} \ge \sum_{pos < j \le n} A_{ij} \quad \text{for} \quad 1 \le i \le pos. \tag{2.18}$$

By a similar argument, we require the remaining rows of A to satisfy the following condition:

$$\sum_{1 \le j \le pos} A_{ij} \le \sum_{pos < j \le n} A_{ij} \quad \text{for} \quad pos < i \le n. \tag{2.19}$$

Thus, the *signed cut* that we propose here for extracting A_{11} from the entire matrix A is the cut that satisfies

$$C2: \underset{1 \le i \le pos}{\forall} \left(\sum_{1 \le j \le pos} A_{ij} \ge \sum_{pos < j \le n} A_{ij} \right) \wedge \underset{pos < i \le n}{\forall} \left(\sum_{1 \le j \le pos} A_{ij} \le \sum_{pos < j \le n} A_{ij} \right). \tag{2.20}$$

To compute the position that satisfies the *signed cut* criterion proposed above, one most direct way is to check all the n possible positions one by one. However, this is very time consuming, with $O(n^3)$ time, since each checking process requires $O(n^2)$ time for scanning the entire matrix. Fortunately, we have found a very efficient way to complete the heavy task through at most thrice scanning the entire matrix, including two top-down scanning and one bottom-up scanning, respectively.

We first define two $(n-1)$-dimensional vectors, λ_1 and λ_2, which can help us efficiently find a good signed cut position. The posth component of λ_1 is defined as

$$\lambda_1(pos) = \left\| \left\{ i, 1 \le i \le pos \,\middle|\, \sum_{1 \le j \le pos} A_{ij} \ge \sum_{pos < j \le n} A_{ij} \right\} \right\|, \quad \text{for } 1 \le pos < n. \tag{2.21}$$

TABLE 2
Procedure for Computing λ_1

```
λ1 = zeros(1,n);  /* initialize a zero vector */
for r = 1:n-1   /* r - row index */
   sum_left = 0; sum_right = 0;
   for c = 1:n-1  /* c - column index */
      sum_left = sum_left + A_rc;
      if c >= r
         sum_right = D(r) - sum_left;
         if sum_left >= sum_right
            λ1(c) = λ1(c) + 1;
         endif
      endif
   endfor
endfor
```

The posth component of λ_2 is defined as

$$\lambda_2(pos) = \left\| \left\{ i, pos < i \le n \left| \sum_{1 \le j \le pos} A_{ij} \le \sum_{pos < j \le n} A_{ij} \right. \right\} \right\|, \quad \text{for } 1 \le pos < n, \tag{2.22}$$

where $\|S\|$ denotes the cardinality of set S.

Actually, $\lambda_1(pos)$ and $\lambda_2(pos)$ denote the numbers of cut positions satisfying (2.18) and (2.19), respectively. As an example, for the matrix shown in Fig. 3b, we have

$$\begin{aligned}\lambda_1 = [&0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 12, 13, 13, 13,\\ &15, 16, 16, 17, 17, 17, 20, 21, 21, 21, 22],\\ \lambda_2 = [&22, 22, 21, 21, 20, 19, 18, 17, 16, 15, 16, 16, 15,\\ &14, 13, 10, 9, 9, 7, 7, 6, 5, 5, 4, 3, 2, 0].\end{aligned}$$

λ_1 and λ_2 can be quickly figured out through at most three scanning of the entire matrix. The first top-down matrix scanning process is for computing the degree vector D such that $D(i) = \sum_{1 \le j \le n} A_{ij}$. The second top-down matrix scanning process is for computing the vector λ_1, as listed in Table 2. In a similar manner, we can compute the vector λ_2 through a bottom-up matrix scanning process that is omitted here due to the space limitation.

Based on λ_1 and λ_2, a vector S that measures how well different positions can fit the signed cut criterion given by (2.20) can be computed as

$$S(pos) = \left(\frac{\lambda_1(pos)}{pos} + \frac{\lambda_2(pos)}{n - pos} \right) \cdot I_{\{1 \le pos < n\}} + 2 \cdot I_{\{pos = n\}}. \tag{2.23}$$

It is easy to show that $0 \le S(pos) \le 2$ for $1 \le pos \le n$. The candidate positions that satisfy the signed cut criterion are those with S-values equal to 2. Again, for the example matrix shown in Fig. 3b, we have

$$\begin{aligned}S = [&0.81, 1.84, 1.84, 1.86, 1.87, 1.86, 1.86, 1.85, 1.84, 1.83, 1.94,\\ &2, 1.92, 1.93, 1.87, 1.65, 1.70, 1.79, 1.62, 1.73, 1.67, 1.61,\\ &1.87, 1.88, 1.84, 1.81, 0.81, 2].\end{aligned}$$

In this example, $pos = 12$ is one of the candidate positions and is also the optimal cut position. For more examples to demonstrate the effectiveness of the proposed method, Figs. 3a, 3b, 4c, 4d, 5b, 6a, and 6b show the signed cut positions found by this method, which are represented by the solid crosses. We can see that all of them are identical to those real splits.

Based on its definition, $S(n)$ is always equal to 2. If the position n is found to be the unique candidate in the extraction step, then it indicates that a current network is already cohesive enough and need not be further divided. This corresponds to Step 3 of the FEC algorithm given in Table 1. This way, the prior knowledge such as the number of communities is not required for determining when to stop the recursive bisections.

Based on the above discussions, the main steps of the EC subroutine, which can efficiently find the best signed cut position, are summarized as follows:

Step 1. Compute the degree vector D through a top-down matrix scanning process.

Step 2. Compute the vector λ_1 through a top-down matrix scanning process.

Step 3. Compute the vector λ_2 through a bottom-up matrix scanning process.

Step 4. Compute the vector S based on λ_1 and λ_2.

Step 5. Return the position, with S-value being equal to 2.

Proposition 2.2. *The time complexity of the EC subroutine is* $\mathrm{O}(n + m)$.

Proof. The matrix scanning processes in the EC subroutine are the most computationally costly steps. With the help of the implementation using an adjacency list, in which only nonzero elements are stored, each scanning will take only $\mathrm{O}(n + m)$ time. Thus, the overall time required by the EC subroutine is bounded by $\mathrm{O}(n + m)$ or $O(n)$ for sparse networks. □

Proposition 2.3. *The time complexity of the FEC algorithm for community mining from a signed network (as shown in Table 1) is* $\mathrm{O}(Kl(n + m))$, *where* K *is the number of communities detected.*

Proof. Let $T(A_0)$ be the time complexity of FEC and A_0 be the adjacency matrix of the network to be classified. We have

$$T(A_0) = \begin{cases} T_1 + T_2 + T_4 + T(A_{11}) + T(A_{22}) & pos < n \\ T_1 + T_2 + T_4 & pos = n, \end{cases}$$

where T_1, T_2, and T_4 are the time required by Steps 1, 2, and 4 of FEC, respectively. Based on Propositions 2.1 and 2.2, we have $T_1 + T_2 = \mathrm{O}(l(n + m))$ and $T_4 = \mathrm{O}(n)$. Thus, the worst-case time complexity of FEC is

$$T(A_0) = \begin{cases} \mathrm{O}(l(n + m)) + T(A_{11}) + T(A_{22}) & pos < n \\ \mathrm{O}(l(n + m)) & pos = n. \end{cases}$$

In a recursive manner, one can easily show that

$$T(A_0) < \mathrm{O}(Rl(n + m)),$$

where R is the total number of times recursively calling of FEC by Steps 5 and 6 during the course of finding all

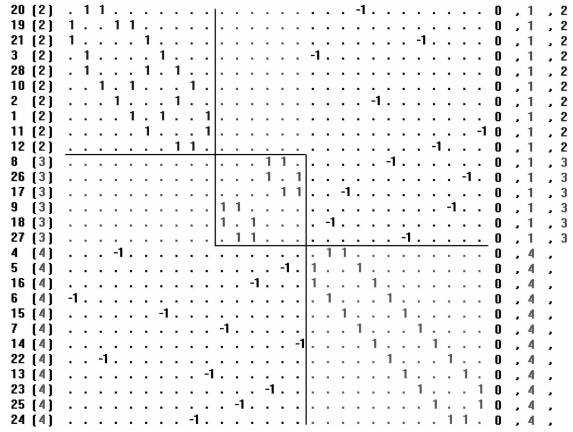


Fig. 7. The FEC output for the illustrative signed network, as shown in Fig. 1a.

K communities. It can easily be shown that $R = 2K - 1$. Therefore,

$$T(A_0) < \mathrm{O}(Kl(n+m)).$$

□

3 Experiments

We have tested our proposed community mining algorithm FEC with a number of benchmark networks and randomly created networks so as to evaluate its effectiveness and efficiency. The proposed FEC requires no parameter setting, except for l. We have empirically found that setting l to 10 worked sufficiently well for all experiments that we have tested, which is substantiated by the sensitivity analysis on the value of l to be presented in Section 3.5. To measure the partitioning quality, we define the error rate of a partition C of a signed network as

$$error(C) = \frac{P(C)}{\sum_i \sum_j |A_{ij}|} \times 100\%, \tag{3.1}$$

where $P(C)$ is defined in (2.13), and A is the adjacency matrix of the signed network. Obviously, for partitionable signed networks, the smaller the value of error(C) is, the better the partitioning quality becomes. If all the within-community links are negative, and all the between-community links are positive, then $P(C) = \sum_i \sum_j |A_{ij}|$, and the error will be 100 percent. If all the within-community links are positive, and the between-community links are negative, then $P(C) = 0$, and the error will be zero. However, for nonpartitionable networks, the error rate of the best partition will be more than 0 due to some negative within-community links or positive between-community links.

3.1 Evaluation on an Illustrative Network

We have applied the FEC algorithm to the illustrative signed network, as given in Fig. 1a, for validation. Fig. 7 presents the FEC outputs for the network. In the figure, the labels on the left are the node indices, followed by their corresponding community IDs. For example, in the first row of Fig. 7, "20" [2] denotes "node 20" belonging to "community 2." As FEC is a recursive algorithm, the labels on the right indicate the sequence of extracting communities one by one. In addition, the matrix in Fig. 7 contains diagonal blocks corresponding to different communities. In particular, three communities have been detected by FEC, which are community 2 (20, 19, 21, 3, 28, 10, 2, 1, 11, 12), community 3 (8, 26, 17, 9, 18, 27), and community 4 (4, 5, 16, 6, 15, 7, 14, 22, 13, 23, 25, 24). In this case, the error of the partition is 0.

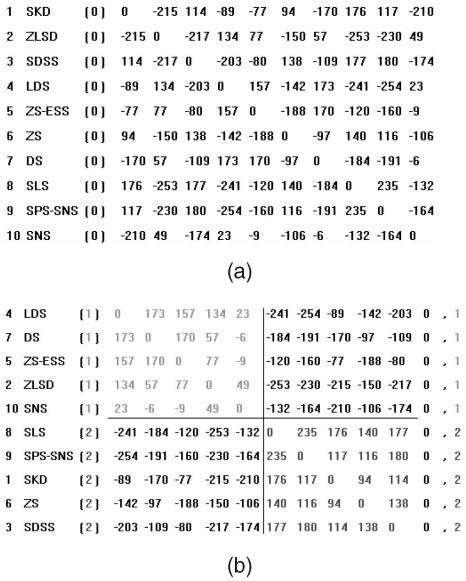


Fig. 8. Community mining from the Slovene Parliamentary party network using the FEC algorithm. (a) The initial adjacency matrix of the unclassified Slovene Parliamentary party network. SKD, ZLSD, SDSS, LDS, ZS-ESS, ZS, DS, SLS, SPS-SNS, and SNS, respectively, denote the abbreviated names of the 10 parities. (b) The output adjacency matrix of the FEC algorithm. In this matrix, two communities have been detected: (LDS, DS, ZS-ESS, ZLSD, SNS) and (SLS, SPS-SNS, SKD, ZS, SDSS). The error ratio of this partition is 0.231 percent.

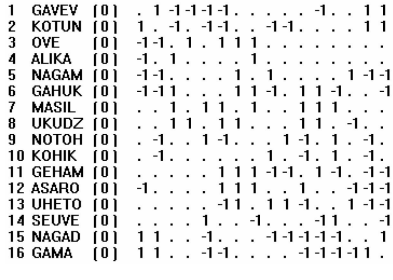
3.2 Evaluation on Existing Social Networks for Benchmarking

3.2.1 Slovene Parliamentary Party Network

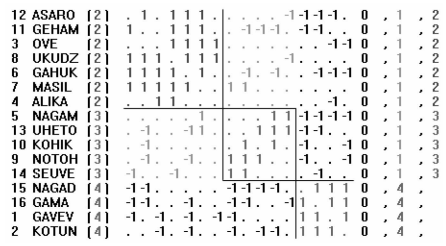
Fig. 8a presents the relation network of 10 parties of the Slovene Parliamentary in 1994 [33], which was established by a group of experts on Parliament activities. The weights of links in the network were estimated by 100 multiplying the average of the distance between each pair of parties on the scale from -3 to $+3$, where $\{-3, -2, -1\}$ mean that the parties are "very dissimilar," "quite dissimilar," and "dissimilar," respectively. 0 means that parties are neither dissimilar nor similar (somewhere in between). $\{+1, +2, +3\}$ mean that parties are "similar," "quite similar," and "very similar," respectively. Thus, it is a signed network. Using the FEC algorithm, we analyzed this network, and the experimental results were presented in Fig. 8b. In this case, FEC detected two communities, which are identical to the analysis results given by Kropivnik and Mrvar [33].

3.2.2 Gahuku-Gama Subtribes Network

This network was created based on Read's study on the cultures of highland New Guinea [4] (obtained from http://mrvar.fdv.uni-lj.si/sola/info4/andrej/prpart5.htm). It describes the political alliances and oppositions among 16 Gahuku-Gama subtribes, which were distributed in a particular area and were engaged in warfare with one



(a)



(b)

Fig. 9. Community mining from the Gahuku-Gama subtribes network using the FEC algorithm. (a) The initial adjacency matrix of the unclassified Gahuku-Gama subtribes network. (b) The output matrix of the FEC algorithm. In this matrix, three communities can be detected: (ASARO, GEHAM, OVE, UKUDZ, GAHUK, MASIL, ALIKA), (NAGAM, UHETO, KOHIK, NOTOH, SEUVE), and (NAGAD, GAMA, GAVEV, KOTUN). The error ratio of the partition is 3.45 percent.

another in 1954, as shown in Fig. 9a. The positive and negative links of the network correspond to political arrangements with positive and negative ties, respectively. Using the FEC algorithm, we analyzed this signed network and detected its community structure, as presented in Fig. 9b. Three communities were detected as a result, which was identical to the three-way partition of the same network reported by Doreian and Mrvar [3].

3.2.3 Karate Club Network and Football Association Network

To further evaluate the effectiveness of the FEC algorithm on positive social networks, two benchmark networks, namely, karate club network and football association network, were tested. Fig. 10a presents the community structure of the karate club network (refer to Fig. 4a) extracted by FEC, where the nodes were divided into two groups, as equivalent to the actual split shown in Fig. 4a after the first call of FEC. In addition, FEC, after being called recursively, suggests partitioning community B further into two smaller groups. Fig. 10b shows the original football association network. Fig. 10c shows the community structure of the football association network, as extracted by FEC. Fig. 10d presents the network, with different communities shown using different gray degrees. The identified 12 blocks correspond to 12 different conferences. Most of the conferences were correctly extracted, except for a few teams such as five teams of Independent conference, teams 28 and 58 of Western Athletic, and team 110 of Texas Christian. The errors were due to the fact that those teams played more interconference matches than intraconference matches.

3.3 Evaluation on Randomly Generated Signed Networks

All the experiments reported so far involve only tens of nodes and links. To test the performance of FEC further, we have applied FEC to random signed networks with larger sizes. The use of randomly generated networks is common in testing the performance of algorithms for detecting community structures [5], [22], [31]. However, most of the existing studies considered only networks with positive links instead of signed networks. To generate signed networks with controlled community structures, we have modeled the generation process based on a set of community structure-conscious parameters. We denote a random signed network as $SG(c, n, k, p_{in}, p_-, p_+)$, where c is the number of communities in the network, n is the number of nodes in each community, k is the degree of each node, p_{in} is the probability of each node connecting other nodes in the same community, p_- denotes the probability of negative links appearing within communities, and p_+ denotes that of positive links appearing between communities. The weights of the generated links within communities are positive, whereas those between communities are negative. Based on the network generation process, there will be $c \times n \times k \times p_{in} \times p_-$ negative links within communities and $c \times n \times k \times (1 - p_{in}) \times p_+$ positive links outside communities. Obviously, all the networks defined by $SG(c, n, k, p_{in}, 0, 0)$ are *partitionable*, as the two noise parameters p_+ and p_- are set to zero. p_{in} is the parameter controlling the cohesiveness of the communities inside the generated signed network. When $p_{in} \to 0$, their community structures become increasingly ambiguous and, thus, it is more difficult to identify them. Given a fixed p_{in}, we can control the noise level (that is, the number of negative links within communities and the number of positive links between communities). In general, it will be more difficult to extract communities correctly when the values of p_- and p_+ are large.

Fig. 11a shows the adjacency matrix of a random signed network created as $SG(4, 16, 16, 0.7, 0, 0)$. Thus, this is a partitionable network. The weights of links are represented by different gray degrees. As shown in Fig. 11b, FEC gave exactly this result, with four communities being extracted, and the error ratio of this partition was zero. The rightmost gray degree bar of the output matrix denotes the sequence of extracting communities one by one.

Fig. 12a shows another random network generated as $SG(4, 16, 16, 0.2, 0, 0)$. Note that in this case, p_{in} is 0.2, which is a relatively low probability value for within-community links. Fig. 12b corresponds to the output as obtained by FEC, in which four predefined communities were all detected. The error ratio of the partition was again zero. This indicates that FEC is also effective for identifying communities from signed networks with a less well-defined community structure.

We have also applied FEC to randomly generated signed networks that are *nonpartitionable*. Fig. 13b gives the output matrix of $SG(4, (32, 64, 96, 128), 24, 0.7, 0.2, 0.2)$, which is a special random network that contains four communities with different sizes and noises, as shown in Fig. 13a. Fig. 14b presents the FEC output for the random network $SG(25, 30, 20, 0.6, 0.3, 0.3)$ containing significant noises, as shown in Fig. 14a. In both cases, the FEC algorithm succeeds in finding all predefined communities.

In order to evaluate the *robustness* of FEC for community extraction, we have applied the proposed FEC algorithm to another random network $SG(4, 64, 32, p_{in}, 0, 0)$, with p_{in}, p_+,

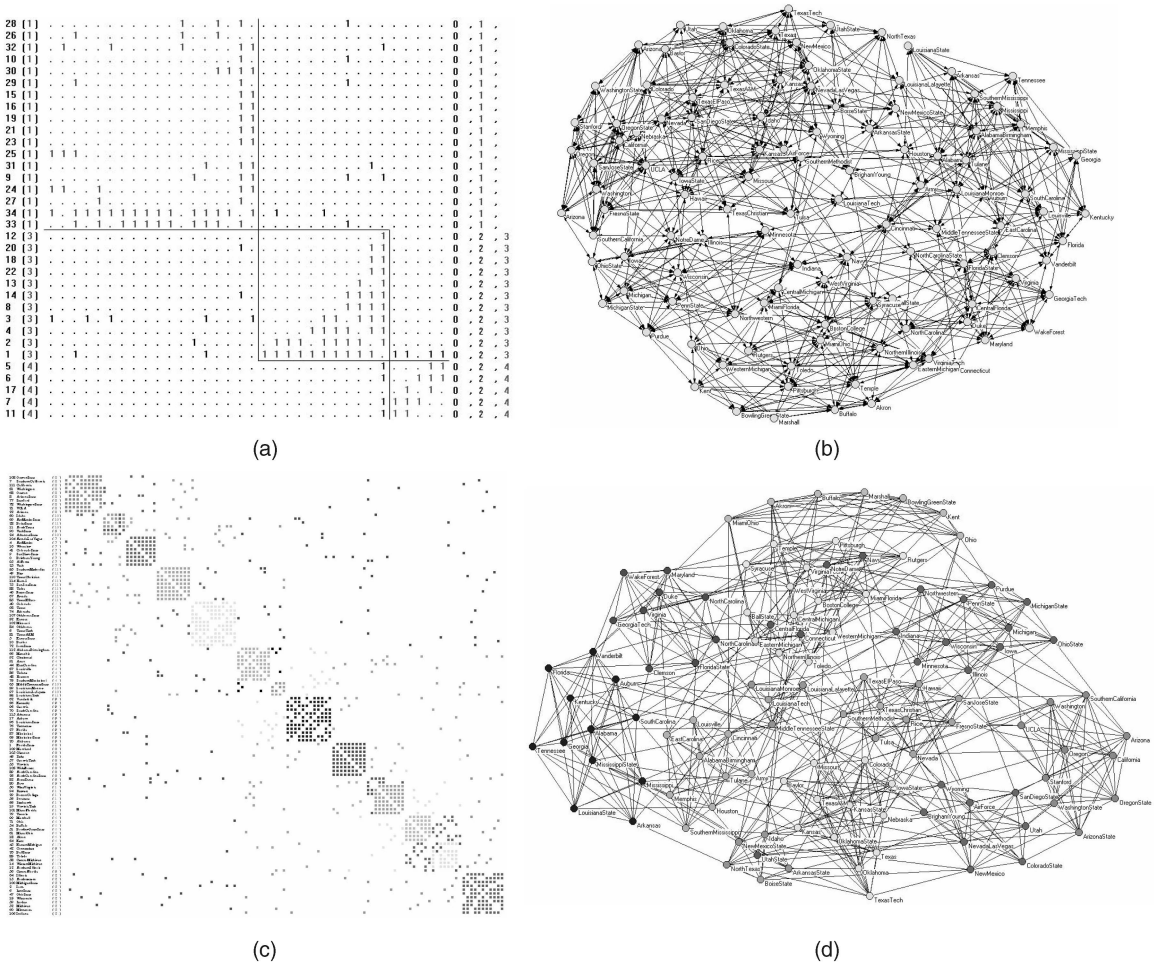


Fig. 10. Extracting communities from two positive social networks using the FEC algorithm. (a) The FEC output of the karate club network. (b) The original football association network. (c) The FEC output of the football association network. (d) The football association network with its communities indicated using different gray degrees.

and p_{-} changing gradually from 0 to 1. In addition, we have repeated the same testing using the Doreian-Mrvar (DM) algorithm [3].

Our obtained empirical results are shown in Fig. 15. We note that as p_{in} decreases from 1 to 0, the community structure in the network becomes less cohesive. As a result, it becomes increasingly difficult to cluster them correctly, and the clustering error increases, as expected. The y-axis in Fig. 15a denotes the fraction of nodes correctly clustered by these two algorithms: Each point in the curves was obtained by taking the average over 100 random networks, where the variations were due to the stochastic nature of the network generation process. We observed that in most cases, the clustering accuracy of FEC was much better than that of the DM algorithm. FEC correctly clustered all the nodes when p_{in} was not less than 0.5. In the range of $0.2 \leq p_{in} \leq 0.4$, where the network structures were quite ambiguous, the accuracy of FEC was found to be still acceptable, that is, not less than 60 percent.

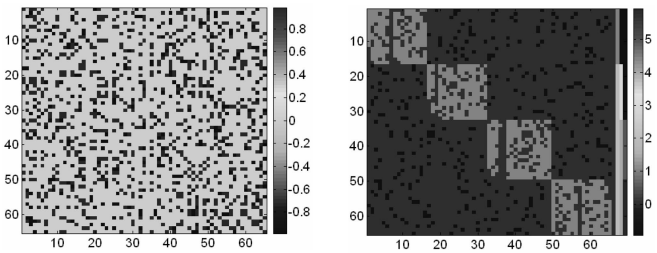


Fig. 11. Identifying the community structure from a random signed network $SG(4, 16, 16, 0.7, 0, 0)$. (a) The initial adjacency matrix. (b) The FEC output.

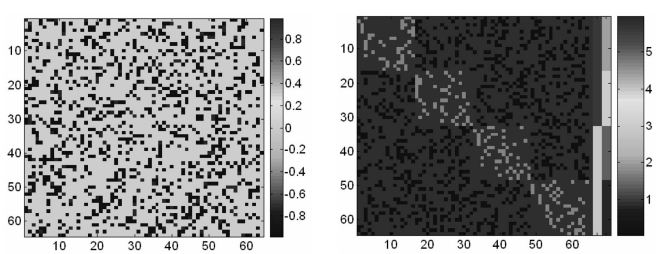


Fig. 12. Identifying the community structure from a random signed network $SG(4, 16, 16, 0.2, 0, 0)$. (a) The initial adjacency matrix. (b) The FEC output.

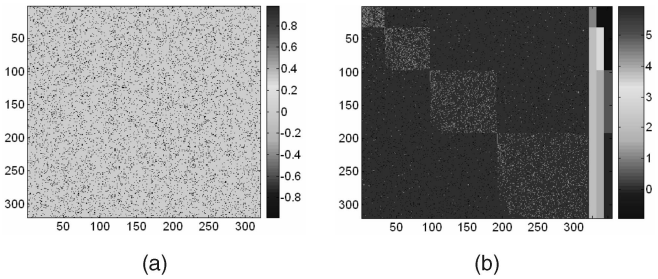


Fig. 13. Identifying the community structure from a random signed network $SG(4,(32,64,96,128),24,0.7,0.2,0.2)$. (a) The initial adjacency matrix. (b) The FEC output.

The DM algorithm only performed well for p_{in} not less than 0.65.

For robustness toward networks generated based on different noise levels, Figs. 15b and 15c show the results corresponding to $SG(4,64,32,0.8,p_-,p_+)$. As p_- and p_+ increase, the network structure gradually becomes more ambiguous and, again, the clustering accuracy (obtained over 100 random networks) decreases accordingly, as expected. In Fig. 15b, we can observe that 1) FEC is sensitive to p_- but insensitive to p_+. In other words, given p_-, the clustering accuracy of FEC changes slightly as the value of p_+ increases. Actually, when p_+ increases from 0 to 1, the positive links between communities become denser. In such cases, the community structure of the network will be decided not only by the signs of links but also by the density of links. As discussed above, FEC considers both clustering attributes and deals very well with positive networks. We believe that this is why FEC is insensitive to the noise level of p_+. 2) FEC works very well, even at high levels of noise. For example, in our experiments, when the values of p_- and p_+ were in the ranges of $0 \le p_- \le 0.35$ and $0 \le p_+ \le 1$, the clustering accuracy of FEC was found to be around 100 percent. For the range of $p_- = 0.4$ and $0 \le p_+ \le 0.85$, the clustering accuracy was more than 90 percent. Even for the range of $0.4 < p_- \le 0.6$ and $0 \le p_+ \le 1$, the clustering accuracy was not less than 50 percent. When compared with the results obtained by the DM algorithm, as shown in Fig. 15c, we observed that 1) the DM algorithm was more sensitive to noise and gave satisfactory results only when $0 \le p_- \le 0.25$, and $0 \le p_+ \le 0.3$ and 2) DM was very sensitive to p_+. For the range of $0 \le p_- \le 0.25$, the clustering accuracy was around 100 percent for $0 \le p_+ \le 0.3$, but this dropped to 60 percent when $0.3 < p_+ \le 1$. This was mainly because only one clustering attribute, the signs of links, was considered in DM.

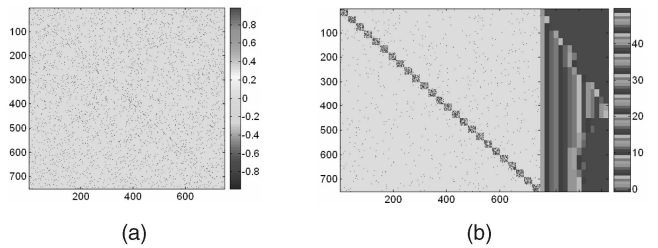


Fig. 14. Identifying the community structure from a random signed network $SG(25,30,20,0.6,0.3,0.3)$. (a) The initial adjacency matrix. (b) The FEC output.

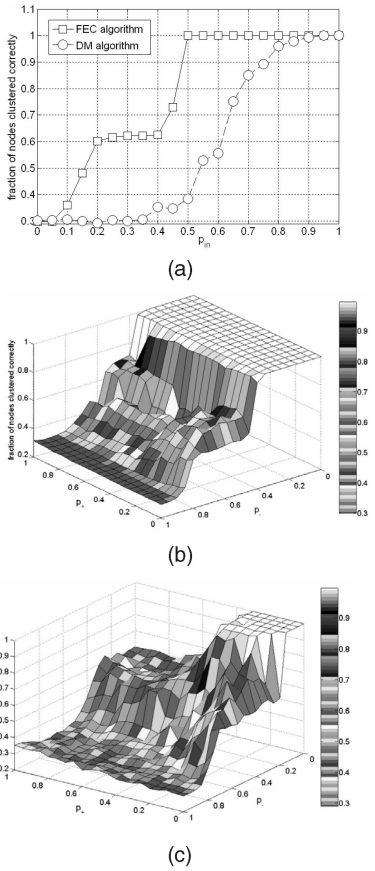


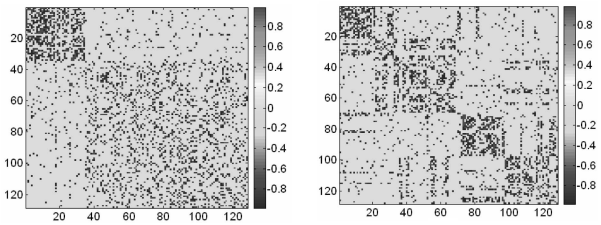
Fig. 15. Testing the clustering robustness of FEC and DM with respect to different network community structures and noise levels. (a) Clustering performance of both algorithms at different levels of within-community link density. (b) Clustering performance of the FEC algorithm at different noise levels. (c) Clustering performance of the DM algorithm at different noise levels.

3.4 Approximate Signed Cut

Unlike most of the existing criteria such as the error function, minimum cut, averaged cut, and normalized cut defined in (2.13), (2.14), (2.15), and (2.16), our proposed signed cut criterion is tightly defined based on the constraint defined in (2.20). If the community structure is too noisy, then such a position may not exist in the transformed matrix obtained by the FC subroutine, and the underlying network may be considered to be too compact to be further divided. In fact, one can soften the constraint to a more flexible one so as to define an approximate signed cut, with the revised constraint given as

$$\widetilde{C2}: \underset{1\le i\le pos}{\forall}\left(\sum_{1\le j\le pos} A_{ij} \gtrapprox \sum_{pos<j\le n} A_{ij}\right)\wedge \underset{pos<i\le n}{\forall}\left(\sum_{1\le j\le pos} A_{ij} \lessapprox \sum_{pos<j\le n} A_{ij}\right). \tag{3.2}$$

Here, $\gtrapprox$ and $\lessapprox$, respectively, denote "be approximately greater than or equal to" and "be approximately less than or equal to." Such an approximate cut can again be directly figured out with the aid of vector S defined in (2.23) by picking the position such that the S-value is the largest, given as



(a) (b)

Fig. 16. Comparing two different signed cut criteria using $SG(4, 32, 20, 0.7, 0.6, 0.6)$, a random signed network with a high level of noise. (a) The output of the FEC algorithm with respect to the signed cut criterion. (b) The output of the FEC algorithm with respect to the approximate signed cut criterion.

$$pos = \arg\max_{1 \le i < n}(S[i]). \tag{3.3}$$

The use of this approximate signed cut criterion can give better results for networks with a high noise level by being able to detect more communities with finer granularities. Note that in the new FEC algorithm incorporating the approximate signed cut, the stopping condition of recursive bipartitions should be

$$error(C_{t+1}) \ge error(C_t). \tag{3.4}$$

That is, the recursive bipartition will stop if it cannot lead to decreasing the error rate of the overall partition that has been previously computed.

As an illustration, we have applied this notion of approximate signed cut to a random signed network with a high noise level and have compared the results with those obtained by the original FEC. Fig. 16 shows the obtained experimental results. We can see that by using the original signed cut criterion, the FEC algorithm detects two communities with coarse granularities. While using the approximate signed cut criterion, the FEC algorithm can detect almost four predefined communities with quite fine granularities.

Based on the approximate signed cut criterion, we analyzed Sampson Monastery signed networks, which were created based on Sampson's study on four social relationships among a group of monks in a New England monastery [34] (obtained from http://mrvar.fdv.uni-lj.si/sola/info4/andrej/prpart5.htm). Each type of relationship has both positive and negative aspects, as shown in Table 3. As reported in [34], the 18 monks were finally split into different groups due to a "political crisis in the cloister." Group (2, 3, 17, 18) was expulsed, and group (1, 7, 14, 15, 16) left voluntarily. In the end, only monks 5, 6, 9, and 11 remained.

We applied the FEC algorithm incorporating the approximate signed cut to the four signed networks in terms of different relationships, and the experimental results are shown in Table 4. We can see that the partition results of the networks in terms of the first three relationships are quite close to the real division.

TABLE 3
Sampson Monastery Networks

Relationships	Attributes of relationship	
	Positive	Negative
friendship	like	dislike
esteem	esteem	disesteem
influence	positive	negative
sanction	praise	blame

TABLE 4
Experimental Results of the FEC Applied to Sampson Monastery Networks

Relationships	Partitions obtained by FEC	Error rates
friendship	(1,2,7,12,14,15,16), (3,17,18), (4,5,6,8,9,10,11,13)	11.11%
esteem	(1,2,7,12,14,15,16), (3,17,18), (4,5,6,8,9,10,11,13)	13.30%
influence	(1,2,7,12,14,15,16), (3,17,18), (4,5,6,8,9,10,11,13)	13.30%
sanction	(1,2,7,14,15,16,18), (3), (5,13), (17), (4,6,8,11), (9,10,12)	10.26%

3.5 Sensitivity Analysis of l-Value

The number of iteration steps l is the only parameter that is required by the FC subroutine in the proposed FEC. In each iteration, the value of ψ will be updated and, thus, the ranking of all nodes will be changed according to the current estimate of the probabilities of arriving at the selected sink. As long as all the members of the sink community are put on the top of the sequence, it will be good enough for the purpose of community mining. Thus, the convergence of the FC subroutine can be evaluated based on that of the sorted node sequence. Through experiments, we have found that this sorted node sequence in fact converges quickly.

According to Fig. 17, the x-axis refers to the l-value, and the y-axis refers to the difference between two consecutive sorted node sequences, x and y, defined as $\mathrm{nnz}(x - y)$, which counts the number of nonzeros of a given vector. We have tested different values of l for all the networks mentioned in the paper. We have observed that the sorted sequences for *most* of the networks converge within around 10 steps. Some networks need more steps. For instance, the football association network needs the largest number of steps (60). Note that even though some small $\mathrm{nnz}(x - y)$ fluctuation may still be present after around $10 \sim 20$ steps, as caused by some node reranking within communities, this will not change the communities that have been identified in the much earlier iterations. This implies that the performance of FC is insensitive to the value of the parameter l when l is greater than around $10 \sim 20$.

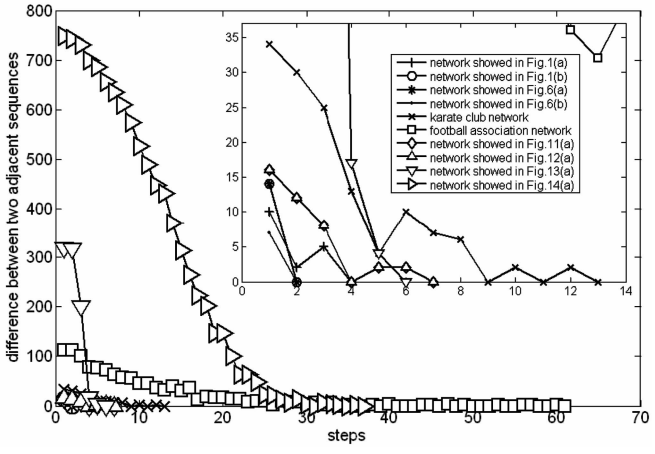


Fig. 17. Sensitivity analysis of the l-value for different problems.

TABLE 5
Average Actual Computational Time for Different Networks

Networks	Seconds
Network shown in Fig. 1a	0.043
Network shown in Fig. 1b	0.043
Karate club network shown in Fig. 4a	0.054
Football association network shown in Fig. 5a	0.181
Gahuku-Gama subtribes network in Fig. 9a	0.039
Slovene Parliamentary party network in Fig. 8a	0.038
Sampson networks shown in Table 4	0.032

3.6 Actual-Time Performance of FEC

In addition to the computational complexity analysis provided in Section 2, we have recorded the actual computational time needed for analyzing the networks. We ran all the experiments on a computer with a CPU of 2 GHz, and the memory size is 1, 024 Mbytes. The operating system was Windows XP, and the simulation was implemented and tested using Matlab 7.0. We repeated FEC 100 times for each network, and the averaged actual computational time taken is shown in Table 5.

In addition, we have also applied the FEC algorithm to large random networks with different sizes to examine how the actual computational time would change with respect to the network size. In this experiment, all random networks are sparse and contain 10 communities. Based on Fig. 18, we note that 1) when the FEC algorithm is applied to the network with $O(10^4)$ nodes, the actual computational time required is quite low, as bounded by $O(10^2)$ seconds and 2) the actual computational time is approximately linear with respect to the network size.

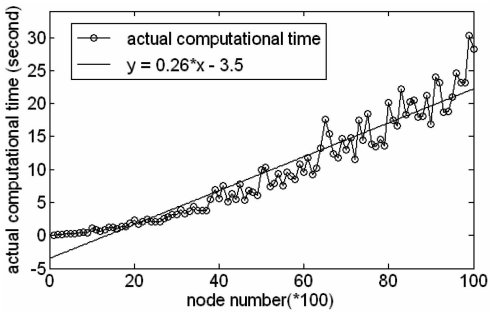


Fig. 18. Actual computational time versus the size of the network.

4 Conclusion

In this work, we have developed a new algorithm, called FEC, for identifying communities from signed social networks. The key idea behind it rests on an agent-based random walk model, based on which the FC phase can find the sink community including a specified node with a linear time complexity. Thereafter, the sink community is extracted from the entire network by the EC phase based on some robust graph cut criteria that we have derived. We have tested the FEC algorithm by using different types of networks, including both benchmark and synthetically generated signed social networks. The obtained experimental results show that our proposed FEC algorithm produces good performance in both speed and clustering capability. The distinct features of our approach can be summarized as follows.

First, FEC is very fast: Its time complexity is the linear time of $O(Kl(n+m))$, where the values of K and l are typically small, which is equivalent to saying that, in general, the complexity is $O(n+m)$.

Second, two clustering attributes, both the signs and the density of the links, are taken into consideration by the FEC algorithm; thus, our approach is suitable for not only signed networks but also positive social networks.

Third, FEC has a good clustering capability. Through the experiments shown in Section 3, we have demonstrated that FEC can work well with both partitionable and nonpartitionable signed networks, and it is insensitive to noise.

Finally, FEC is not sensitive to the particular choice of parameter values and needs no prior knowledge on the community structure (for instance, the number of communities or a good initial partition). Only one parameter is required, which is step length l, by the FC phase of the algorithm. As discussed in Section 3.5, for most networks, the best choice of l falls into the interval of [10, 20].

In our future work, we will focus on how we can use the FEC approach to further address, besides identifying communities from social networks, problems in other related domains such as image segmentation and Web mining.

A color image can readily be mapped into a weighted and directed network, possibly a signed network in some special cases, in which subnetworks correspond to physical segments of such an image. Therefore, the task of image segmentation can be translated into that of finding a natural partition of the mapped network. Especially with FEC, we may quickly extract one patch from an entire image containing some special pixels, without the need to find all segments.

So far, all the work about Web mining considers the Web as a huge directed graph including only positive links. However, the relations among the entities of the Web are miscellaneous and complex, which cannot be modeled appropriately only by positive links. For example, it would seem more reasonable to connect the homepages of two opposed or competitive sites using a negative link. From such a signed Web, we may discover some previously unknown yet profound knowledge.

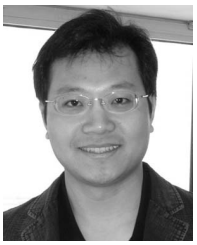
Acknowledgments

Jiming Liu was the corresponding author. The authors would like to express their thanks to the anonymous reviewers for their constructive comments and suggestions. They also thank Mark Newman for providing them with the data sets of the karate club network and the football association network. The work reported in this paper was carried out in the Center for e-Transformation Research, Hong Kong Baptist University, and supported by the Hong Kong RGC Central Allocation Grant HKBU 2/03/C. B. Yang was also supported by the National Natural Science Foundation of China Grant 60503016.

References

[1] S.H. Strogatz, "Exploring Complex Networks," *Nature,* vol. 410, pp. 268-276, 2001.
[2] D.J. Watts and S.H. Strogatz, "Collective Dynamics of Small-World Networks," *Nature,* vol. 393, pp. 440-442, 1998.
[3] P. Doreian and A. Mrvar, "A Partitioning Approach to Structural Balance," *Social Networks,* vol. 18, no. 2, pp. 149-168, 1996.
[4] K.E. Read, "Cultures of the Central Highlands, New Guinea," *Southwestern J. Anthropology,* vol. 10, no. 1, pp. 1-43, 1954.

[5] M. Girvan and M.E.J. Newman, "Community Structure in Social and Biological Networks," *Proc. Nat'l Academy of Science (PNAS),* vol. 99, no. 12, pp. 7821-7826, 2002.
[6] N. Bansal, A. Blum, and S. Chawla, "Correlation Clustering," *Proc. 43rd Ann. IEEE Symp. Foundations of Computer Science (FOCS '02),* pp. 238-247, 2002.
[7] N. Bansal, A. Blum, and S. Chawla, "Correlation Clustering," *Machine Learning,* vol. 56, no. 1-3, pp. 89-113, 2004.
[8] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks," *Proc. 12th Int'l Conf. World Wide Web (WWW '03),* pp. 640-651, May 2003.
[9] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of Trust and Distrust," *Proc. 13th Int'l Conf. World Wide Web (WWW '04),* pp. 403-412, 2004.
[10] P. Pons and M. Latapy, "Computing Communities in Large Networks Using Random Walks," *Proc. 20th Int'l Symp. Computer and Information Sciences (ISCIS '05),* pp. 284-293, 2005.
[11] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society," *Nature,* vol. 435, no. 7043, pp. 814-818, 9 June 2005.
[12] D.-H. Kim and H. Jeong, "Systematic Analysis of Group Identification in Stock Markets," *Physical Rev. E,* vol. 72, 046133, 2005.
[13] M. Fiedler, "Algebraic Connectivity of Graphs," *Czechoslovakian Math. J.,* vol. 23, no. 98, pp. 298-305, 1973.
[14] A. Pothen, H. Simon, and K.P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Matrix Analysis and Application,* vol. 11, no. 3, pp. 430-452, 1990.
[15] M. Fiedler, "A Property of Eigenvectors of Nonnegative Symmetric Matrices and Its Application to Graph Theory," *Czechoslovakian Math. J.,* vol. 25, no. 100, pp. 619-637, 1975.
[16] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligent,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[17] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical,* vol. 49, pp. 291-307, 1970.
[18] J. Scott, *Social Network Analysis: A Handbook,* second ed. Sage Publications, 2000.
[19] R.S. Burt, "Positions in Networks," *Social Forces,* vol. 55, no. 1, pp. 93-122, 1976.
[20] S. Wasserman and K. Faust, *Social Network Analysis.* Cambridge Univ. Press, 1994.
[21] J.R. Tyler, D.M. Wilkinson, and B.A. Huberman, "Email as Spectroscopy: Automated Discovery of Community Structure within Organizations," *Proc. First Int'l Conf. Communities and Technologies (C&T '03),* pp. 81-96, 2003.
[22] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, "Defining and Identifying Communities in Networks," *Proc. Nat'l Academy of Science (PNAS),* vol. 101, no. 9, pp. 2658-2663, 2004.
[23] G.W. Flake, S. Lawrence, C.L. Giles, and F.M. Coetzee, "Self-Organization and Identification of Web Communities," *Computer,* vol. 35, no. 3, pp. 66-71, 2002.
[24] J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM,* vol. 46, no. 5, pp. 604-632, 1999.
[25] P. Pirolli, J. Pitkow, and R. Rao, "Silk from a Sow's Ear: Extracting Usable Structures from the Web," *Proc. ACM Conf. Human Factors in Computing Systems (CHI),* pp. 118-125, 1996.
[26] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, "Trawling the Web for Emerging Cyber-Communities," *Proc. Eighth Int'l Conf. World Wide Web (WWW '99),* pp. 1481-1493, 1999.
[27] S. Chakrabarti, M. van der Berg, and B. Dom, "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery," *Proc. Eighth Int'l Conf. World Wide Web (WWW '99),* pp. 1623-1640, 1999.
[28] G.H. Golub and C.F. Van Loan, *Matrix Computations.* Johns Hopkins Univ. Press, 1989.
[29] S. Milgram, "The Small World Problem," *Psychology Today,* vol. 1, no. 1, pp. 60-67, 1967.
[30] R. Albert, H. Jeong, and A.-L. Barabási, "Diameter of the World Wide Web," *Nature,* vol. 401, pp. 130-131, 1999.
[31] M.E.J. Newman, "Fast Algorithm for Detecting Community Structure in Networks," *Physical Rev. E,* vol. 69, 066133, 2004.
[32] W.W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups," *J. Anthropological Research,* vol. 33, pp. 452-473, 1977.
[33] S. Kropivnik and A. Mrvar, "An Analysis of the Slovene Parliamentary Parties Network," *Developments in Statistics and Methodology,* A. Ferligoj, A. Kramberger, eds., pp. 209-216, 1996.
[34] S. Sampson, "Crisis in a Cloister," PhD dissertation, Cornell Univ., 1969.



Bo Yang received the BSc, MS, and PhD degrees in computer science from Jilin University in 1997, 2000, and 2003, respectively. He is an associate professor in the College of Computer Science and Technology, Jilin University. His main research interests include agent-based systems, autonomy-oriented computing (AOC), complex networks analysis, and knowledge engineering.

William K. Cheung received the BSc and MPhil degrees in electronic engineering from the Chinese University of Hong Kong and the PhD degree in computer science in 1999 from the Hong Kong University of Science and Technology. He is an associate professor in the Department of Computer Science, Hong Kong Baptist University. He has served as the cochair and a program committee member of a number of international conferences, as well as a guest editor of journals in areas including artificial intelligence, Web intelligence, data mining, Web services, and e-commerce technologies. Also, he has been on the editorial board of the *IEEE Intelligent Informatics Bulletin* since 2002. His research interests include artificial intelligence and machine learning, as well as their applications to collaborative filtering, Web mining, distributed data mining, planning under uncertainty, and dynamic Web/grid service management.

Jiming Liu is currently a professor and the head of the Department of Computer Science, Hong Kong Baptist University. He has served as the editor-in-chief of *Web Intelligence and Agent Systems* (IOS Press), *Annual Review of Intelligent Informatics* (World Scientific), and, from 2002 to 2006, the *IEEE Intelligent Informatics Bulletin* (IEEE Computer Society TCII). He is an associate editor of the *IEEE Transactions on Data and Knowledge Engineering* (IEEE Computer Society) and several other journals. His research interests include autonomy-oriented computing (AOC) paradigm, Web intelligence and the Wisdom Web, multiagent systems, self-organizing complex systems, and autonomous networks. He has published more than 200 research articles in refereed international journals and conferences, and a number of books including authored research monographs, for example, *Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling* (Kluwer Academic/Springer), *Spatial Reasoning and Planning: Geometry, Mechanism, and Motion* (Springer), *Multi-Agent Robotic Systems* (CRC Press), and *Autonomous Agents and Multi-Agent Systems: Explorations in Learning, Self-Organization and Adaptive Computation* (World Scientific).