Community Detection in Signed Networks: an Error-Correcting Code Approach

Cheng-Shang Chang, Duan-Shin Lee, Li-Heng Liou, and Sheng-Min Lu
Institute of Communications Engineering, National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
Email: cschang@ee.nthu.edu.tw; lds@cs.nthu.edu.tw; dacapo1142@gmail.com; s103064515@m103.nthu.edu.tw

***Abstract*—In this paper, we consider the community detection problem in signed networks, where there are two types of edges: positive edges (friends) and negative edges (enemies). One renowned theorem of signed networks, known as Harary's theorem, states that structurally balanced signed networks are clusterable. By viewing each cycle in a signed network as a parity-check constraint, we show that the community detection problem in a signed network with two communities is equivalent to the decoding problem for a parity-check code. We also show how one can use two renowned decoding algorithms in error-correcting codes for community detection in signed networks: the bit-flipping algorithm, and the belief propagation algorithm. In addition to these two algorithms, we also propose a new community detection algorithm, called the Hamming distance algorithm, that performs community detection by finding a codeword that minimizes the Hamming distance. We compare the performance of these three algorithms by conducting various experiments with known ground truth. Our experimental results show that our Hamming distance algorithm outperforms the other two.**

I. Introduction

As the advent of on-line social networks, structural analysis of networks becomes an important research topic. In a social network, an edge between two nodes usually represents friendly interactions between these two nodes, and structural analysis of networks with both directed or undirected edges has been studied extensively in the literature (see e.g., [1], [2], [3] and references therein). However, as pointed out in the recent survey paper [4], structural analysis of signed networks has received a lot of attention lately in various areas, including sociology, physics, biology, and computer science. In signed networks, there are two types of edges: positive edges (friends) and negative edges (enemies). With these two types of edges in signed networks, researchers can better characterize the interactions between two persons in social networks.

A signed network is called *structurally balanced* if every cycle in the network contains an even number of negative edges [5], [2]. One of the most renowned theorems of signed networks is Harary's theorem [6]. Harary's theorem states that a structurally balanced signed network is clusterable and it can be separated into several communities, where the edges within a community are positive and the edges between two different communities are negative (see the node coloring algorithm in Algorithm 1 for more detailed explanations). But what if a signed network is not structurally balanced? How do we address the community detection problem in a signed network that is not structurally balanced?

In this paper, we focus on the community detection problem in signed networks that might not be structurally balanced. For such a problem, it was proposed in [7] a partition criterion (and an associated partitional algorithm) that finds a partition of the nodes to minimize the (weighted) sum of the following two types of errors: (i) the number of positive edges between two different communities and (ii) the number of negative edges within a community. Such an intuition for the two types of errors in signed networks can be easily incorporated into the notion of modularity [8] for community detection in unsigned networks. In particular, it was proposed in [9] that community detection in signed networks could be formulated as an optimization problem that maximizes the positive modularity and minimizes the negative modularity. In addition to modularity maximization, a signed Laplacian matrix was proposed in [10] and the community detection problem in signed networks was formulated as a normalized cut for such a matrix.

One of the main contributions of this paper is to address the community detection problem in signed networks by using error-correcting codes (ECC). To the best of our knowledge, this is the first paper that links error-correcting codes to the community detection problem in signed networks. The fundamental insight of this is that each cycle in a signed network can be viewed a parity-check constraint in an error-correcting code and every structurally balanced signed network can be viewed as a legitimate codeword. A signed network that is not structurally balanced can then be "corrected" back to a structurally balanced sign network by changing a "minimum" number of the signs of the edges. In this paper, we show how one can use two renowned decoding algorithms in error-correcting codes for community detection in signed networks with two communities: (i) the bit-flipping algorithm, (ii) the belief propagation algorithm. In addition to these two algorithms, we also propose a new community detection algorithm, called the Hamming distance algorithm, that performs community detection by finding a codeword that minimizes the Hamming distance. We compare the performance of these three algorithms by conducting various experiments with known ground truth. Our experimental results show that our Hamming distance algorithm outperforms the other two.

The rest of the paper is organized as follows. In Section II, we introduce the related backgrounds for signed networks

and Harary's theorem. In Section III, we establish the links between the community detection problem in signed networks and error-correcting codes. We also illustrate how one can use two renowned decoding algorithms for community detection in signed networks. Experimental results for these three algorithms are presented in Section IV. The paper is concluded in Section V.

II. Signed networks

In this paper, we consider community detection in signed networks. A signed network $G_s = (V, E, W)$ consists of a set of nodes V, a set of edges E, and a function W that maps every edge in E to the two signs $\{+, -\}$. An edge (u, v) with the sign $+$ is called a positive edge and it is generally used for indicating the *friendship* between the two nodes u and v. On the other hand, an edge with the sign $-$ is called a negative edge. A negative edge (u, v) indicates that u and v are enemies and it is better not to cluster these two nodes in the same community. One of the most important results of signed networks is Harary's theorem (see e.g., the book [2]). For a signed network, it is said to be *structurally balanced* if it contains only cycles (loops) with even numbers of negative edges. Harary's theorem says that a structurally balanced signed network is *clusterable* in the sense that it can be divided into connected groups of nodes such that all edges between members of the same group are positive and all edges between members of different groups are negative. The converse statement for Harary's theorem is in general not true, e.g., a signed network with three nodes connected by three negative edges has a cycle of three negative edges. But it is true for a signed networks with two groups. This is stated in the following proposition.

Proposition 1: Consider a signed network that can be divided into *two* connected groups of nodes such that all edges between members of the same group are positive and all edges between members of different groups are negative. Then every cycle of the signed network has an even number of negative edges. Thus, the signed network is structurally balanced.

Proof. Image the two groups of nodes as two islands and the negative edges as bridges between these two islands. As there are exactly two groups in the signed network, a cycle that starts from a node in one group must cross the bridges an even number of times in order to get back to where the cycle is started. ∎

In view of Proposition 1, a structurally balanced signed network with two groups can be easily detected by the *node coloring* algorithm (see e.g., the book [11]) that colors all the nodes in two colors, say black and white. It starts from coloring a node with one color (e.g., black) and repeatedly coloring a neighbor of a colored node by the same (resp. the other) color if it is connected by a positive (resp. negative) edge. This is summarized in Algorithm 1.

ALGORITHM 1: The Node Coloring Algorithm for a Structural Balanced Signed Network with Two Groups

Input: A structurally balanced signed network $G_s = (V, E, W)$.
Output: A partition $\mathcal{P} = \{S_1, S_2\}$.
(1) Initially, choose a node w and assign that node to S_1.
(2) Traverse the graph by using the breadth-first search (BFS) with node w as the root.
(3) For each neighbor v of a traversed node u, assign node v to the set that u is assigned if (u, v) is a positive edge. Otherwise, assign node v to the other set.

III. Community detection

In the previous section, we have shown that a structurally balanced signed network with two groups can be easily detected by the node coloring algorithm in Algorithm 1. But what if the input signed network is not structurally balanced? Our idea is to treat this as an error-correcting code problem and "correct" a signed network that is not structurally balanced into another structurally balanced signed network.

A. Parity-check codes

In this section, we briefly review the parity check codes (see e.g., [12] for more details). The Galois field GF(2) defines two operations $\oplus$ (the exclusive-OR operation) and $\cdot$ (the AND operation) on the set $\{0, 1\}$. These two operations act similarly to the usual addition operation and the usual multiplication operation as they satisfy various algebraic properties, including the associative law, the commutative law and the distributive law. As such, we can add, subtract, multiply and divide in GF(2) as in rational numbers. Specifically, for two binary m-vectors $\mathbf{h} = (h_1, h_2, \ldots, h_m)$ and $\mathbf{w} = (w_1, w_2, \ldots, w_m)$, its inner product is defined as

$$\mathbf{h} \cdot \mathbf{w}^T = (h_1 \cdot w_1) \oplus (h_2 \cdot w_2) \oplus \ldots \oplus (h_m \cdot w_m), \quad (1)$$

where $\mathbf{w}^T$ is the transpose of $\mathbf{w}$. The matrix multiplication can also be defined similarly.

Now consider an $\ell \times m$ matrix $\mathbf{H} = (h_{i,j})$ with all its elements in GF(2) and the set of vectors in the null space of $\mathbf{H}$, i.e., $\{\mathbf{w} : \mathbf{H} \cdot \mathbf{w}^T = \mathbf{0}_\ell\}$, where $\mathbf{0}_\ell$ is the zero vector with dimension ℓ. The matrix $\mathbf{H}$ is called the *parity-check matrix* and the set of vectors in the null space of $\mathbf{H}$ are called the *codewords* of the parity-check code with the parity-check matrix $\mathbf{H}$. To see the intuition behind such an error-correcting code, suppose we transmit a codeword $\mathbf{w}$ through an error-prone channel and receive another vector $\mathbf{w}'$. The vector $\mathbf{w}'$ may not be in the null space and thus we may be able to correct it by selecting a codeword that is "closest" to $\mathbf{w}'$. There are a vast amount of papers in the literature addressing the issue of how to "decode" the received vector. In this paper, we will consider three renowned decoding algorithms: the bit-flipping algorithm, the belief propagation algorithm, and the Hamming distance algorithm.

B. Cycle basis and parity-check codes

In this section, we establish the connections between signed networks and error-correcting codes. Consider a graph $G = (V, E)$ with m edges and n nodes. Index these m edges from $1, 2, \ldots, m$ and n nodes from $1, 2, \ldots, n$. Then every (simple) cycle of the graph can be represented by a binary m-vector $\mathbf{h} = (h_1, h_2, \ldots, h_m)$, where h_i indicates whether the i^{th} edge is in the cycle. Suppose that $\mathbf{h}$ and $\mathbf{h}'$ represent two *non-disjoint* cycles of the graph. Then it is clear that $\mathbf{h} \oplus \mathbf{h}'$ also represents another cycle of the graph, where $\oplus$ is the bit-wise exclusive-OR operation of these two vectors. In fact, any linear combination of cycles (in the field of GF(2)) is also a cycle (or a collection of disjoint cycles). Thus, a collection of the maximum number of linearly independent cycles forms a cycle basis. For a connected graph with n nodes and m edges, the maximum number of linearly independent cycles is known to be $m - n + 1$ [13]. Thus, every collection of $m - n + 1$ linearly independent cycles forms a cycle basis. One way to find a cycle basis is to first construct a spanning tree of the connected network with m edges and n nodes. The number of edges in the spanning tree is $n - 1$ and there are exactly $m - n + 1$ edges that are not in the spanning tree. Adding each of these $m - n + 1$ edges to the spanning tree forms a linearly independent cycle.

Suppose for a connected graph $G = (V, E)$ with m edges and n nodes, we have found a cycle basis $\{\mathbf{h}_i, i = 1, 2 \ldots, m - n + 1\}$, where $\mathbf{h}_i = (h_{i,1}, h_{i,2}, \ldots, h_{i,m})$. Form the $(m - n + 1) \times m$ matrix $\mathbf{H} = (h_{i,j})$. Such a matrix $\mathbf{H}$ is called a fundamental cycle matrix for the graph $G = (V, E)$. Now for a signed network $G_s = (V, E, W)$, we can also associate each positive edge with the weight 0 and each negative edge with the weight 1. Then the weights of the m edges can be represented by a binary vector $\mathbf{w} = (w_1, w_2, \ldots, w_m)$. Such a vector $\mathbf{w}$ is called the weight vector of the signed network $G_s = (V, E, W)$. Clearly, the statement that a cycle represented by $\mathbf{h} = (h_1, h_2, \ldots, h_m)$ contains an even number of negative edges is equivalent to

$$(h_1 \cdot w_1) \oplus (h_2 \cdot w_2) \oplus \ldots \oplus (h_m \cdot w_m) = 0. \tag{2}$$

Writing (2) in the inner product of two vectors in GF(2) yields

$$\mathbf{h} \cdot \mathbf{w}^T = 0, \tag{3}$$

$\mathbf{w}^T$ is the transpose of $\mathbf{w}$.

In the following lemma, we show that a structurally balanced signed network is a codeword of a parity-check code.

Lemma 2: Consider a signed network $G_s = (V, E, W)$. Let $\mathbf{H}$ be any fundamental cycle matrix for the graph $G = (V, E)$ and $\mathbf{w}$ be the weight vector of the signed network $G_s = (V, E, W)$. Then the signed network $G_s = (V, E, W)$ is structurally balanced if and only if

$$\mathbf{H} \cdot \mathbf{w}^T = \mathbf{0}_{m-n+1}, \tag{4}$$

where the matrix product is in GF(2) and $\mathbf{0}_{m-n+1}$ is the $m - n + 1$-column vector with all its elements being 0.

Proof. ($\Rightarrow$) If the signed network $G_s = (V, E, W)$ is structurally balanced, then every cycle in G_s consists of an even number of negative edges. Note that $\mathbf{H}$ is a fundamental cycle matrix of G_s with its row vector $\mathbf{h_i}$ representing the i^{th} cycle. By (2), we have $\mathbf{h_i} \cdot \mathbf{w}^T = 0$, for $1 \leqslant i \leqslant m - n + 1$. Thus, $\mathbf{H} \cdot \mathbf{w}^T = \mathbf{0}_{m-n+1}$.

($\Leftarrow$) Assume $\mathbf{H} \cdot \mathbf{w}^T = \mathbf{0}_{m-n+1}$. Then $\mathbf{h_i} \cdot \mathbf{w}^T = 0$, for $1 \leqslant i \leqslant m - n + 1$. Rewrite it as in (2), i.e.,

$$(h_1 \cdot w_1) \oplus (h_2 \cdot w_2) \oplus \ldots \oplus (h_m \cdot w_m) = 0.$$

It implies that the i^{th} cycle consists of an even number of negative edges for all i. Thus, the signed network $G_s = (V, E, W)$ is structurally balanced. ■

In view of (4), a fundamental cycle matrix $\mathbf{H}$ for a graph can be viewed as the parity-check matrix of a parity-check code. Also, a vector $\mathbf{w}$ that satisfies (4) is a codeword for the parity-check matrix $\mathbf{H}$ (or simply a codeword for the graph $G = (V, E)$ in this paper). In the following lemma, we further show that a two-way partition of a network corresponds to a codeword of a parity-check code.

Lemma 3: Consider a two-way partition $\mathcal{P} = \{S_1, S_2\}$ of the nodes in a graph $G = (V, E)$. Construct a signed network $G_s(V, E, W)$ by assigning all the edges within the same set to be positive and all the edges between two sets to be negative. Let $\mathbf{w}(\mathcal{P})$ be the weight vector of the signed network $G_s = (V, E, W)$. Then $\mathbf{w}(\mathcal{P})$ is a codeword for the graph $G = (V, E)$.

Proof. As a direct result of Harary's theorem in Proposition 1, the signed network $G_s = (V, E, W)$ constructed from the partition $\mathcal{P} = \{S_1, S_2\}$ is structurally balanced. Also, in the structurally balanced network, the multiplication of a fundamental cycle matrix $\mathbf{H}$ and $\mathbf{w}(\mathcal{P})$ is zero by Lemma 2. Thus, we conclude that $\mathbf{w}(\mathcal{P})$ is a codeword for the graph $G = (V, E)$. ■

From Lemma 3, we know every two-way partition corresponds to a codeword. This leads to the following method to generate codewords for a graph $G(V, E)$ from an n-binary vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. For the n-binary vector, we assign node i the value x_i for $i = 1, 2, \ldots, n$. Suppose that the two ends of the j^{th} edge are node u and v. Then we assign $w_j = x_u \oplus x_v$ for all $j = 1, 2, \ldots, m$. Note that both the two binary vectors $\mathbf{0}_n$ and $\mathbf{1}_n$ generate the same codeword, i.e., the zero codeword. Excluding these two binary vectors, we let $S_1 = \{i : x_i = 0\}$ and $S_2 = \{i : x_i = 1\}$ and this yields a two-way partition and a codeword $\mathbf{w}(\mathcal{P})$. We can write this in the following matrix form:

$$\mathbf{w} = \mathbf{x} \cdot \mathbf{G}, \tag{5}$$

where $\mathbf{G} = (g_{i,j})$ is $n \times m$ generator matrix with $g_{i,j} = 1$ when node i is one end of the j^{th} edge and 0 otherwise. Such a generator matrix was considered in [14] for decoding binary node values.

To illustrate this, let us consider a graph with five nodes and nine edges in Figure 1. For this graph, we choose the spanning

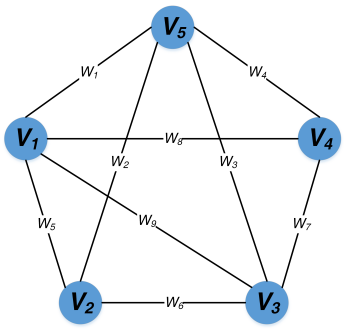


Fig. 1. A graph with five nodes and nine edges.

tree with the edges connected to vertex 5. Such a spanning tree is a star graph and every fundamental cycle contains exactly three edges. For this graph, we have the following parity check matrix

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

and the generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{7}$$

Our approach for the (two-way) community detection problem for a signed network $G_s(V, E, W)$ is to treat a signed network as a received signal and then decode such a signal by finding the most likely codeword. In Section III-C and Section III-D, we discuss two commonly used decoding algorithms for low-density parity-check (LDPC) codes in the literature (see e.g., [15], [16]): (i) the bit-flipping algorithm, and (ii) the belief propagation algorithm. In Section III-E, we propose our decoding algorithm based on Hamming distance.

C. The bit-flipping algorithm

For a signed network $G_s(V, E, W)$, let $\mathbf{w}$ be its weight vector and $\mathbf{H}$ be a fundamental cycle matrix. As discussed before, we may consider $\mathbf{w}$ as the received signal of a parity-check code. To decode such a signal, we first compute the syndrome $\mathbf{s} = (s_1, s_i, \ldots, s_{m-n+1})$ in GF(2):

$$s_i = \sum_{j=1}^{m} h_{ij} w_j. \tag{8}$$

Rewriting (8) in the inner product form yields

$$\mathbf{s} = \mathbf{H} \cdot \mathbf{w}^T, \tag{9}$$

where the matrix product is in GF(2). The i^{th} syndrome component, s_i, indicates whether the i^{th} cycle in $G = (V, E)$ is structurally balanced. If s_i equals to one, the i^{th} cycle is not structurally balanced. In view of (4), all of the parity-check constraints are satisfied if all syndrome components are zero.

ALGORITHM 2: The Bit-Flipping Algorithm

Input: A signed network $G_s = (V, E, W)$.
Output: A partition $\mathcal{P} = \{S_1, S_2\}$.
(1) Compute the syndrome $\mathbf{s}$ in (9). If all the syndrome components are zero, then stop decoding.
(2) Find the number of unbalanced cycles u_k for each edge k by using (10).
(3) Find the edge with maximum u_k and then flip the sign of that edge.
(4) Repeat Steps 1 to 3 until all of the syndrome components are zero or a predefined maximum number of iterations is reached.
(5) Detect two groups by the *node coloring* algorithm in Algorithm 1.

With the syndrome, we can then compute u_k as follows:

$$u_k = \sum_{i=1}^{m-n+1} s_i h_{ik}, \tag{10}$$

The physical meaning of u_k is the number of unbalanced cycles that traverse through the k^{th} edge. Thus, we can find out the edge that has the largest number of unbalanced cycles and then flip the sign of that edge. Intuitively, such a greedy correction will reduce the number of unbalanced cycles and hopefully it will converge to a codeword (that has no unbalanced cycles). Specifically, let k^* be the index of the edge that has the largest number of unbalanced cycles, i.e.,

$$k^* = \arg\max_k u_k. \tag{11}$$

We then flip its sign from 1 to 0 or from 0 to 1, i.e.,

$$w_k \leftarrow 1 - w_k. \tag{12}$$

We summarize the bit-flipping algorithm in Algorithm 2. However, this algorithm does not guarantee convergence. As pointed out in [16], usually there is a design parameter δ that stops the algorithm when the number of unbalanced cycles is within δ.

D. The belief propagation algorithm

The belief propagation algorithm, first introduced by Gallager [15], is a soft decision method. In this paper, we use the computer program written for the sum-product algorithm in [17] (we thank Prof. H.-C. Lee for providing us the computer program). The sum-product algorithm is to compute the maximum a posteriori probability for each codeword bit.

For a signed network $G_s(V, E, W)$, let $\mathbf{w}$ be its weight vector and $\mathbf{H} = (h_{ij})$ be a fundamental cycle matrix. We can create a bipartite graph as depicted in Figure 2 with m variable nodes and $m - n + 1$ check nodes, where v_j denotes the j^{th} variable node, and c_i denotes the i^{th} check node. If $h_{ij} = 1$, there is an edge between variable node j and check node i.

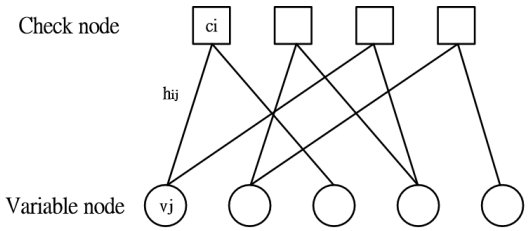


Fig. 2. The construction of the bipartite graph from $\mathbf{H}$

Denote by $\mathcal{N}(v_j)$ the neighboring check nodes that connect to variable node v_j, i.e., $\mathcal{N}(v_j) = \{c_i : h_{ij} = 1\}$. Similarly, denote by $\mathcal{N}(c_i)$ the neighboring variable nodes that connect to check node c_i, i.e., $\mathcal{N}(c_i) = \{v_j : h_{ij} = 1\}$.

For the ease of computation, we deal with the log-likelihood ratios (LLRs). Using LLRs as messages offers implementation advantages over using probabilities or likelihood ratios, because multiplications are replaced by additions and the normalization step is eliminated. The log-likelihood ratio is shown below:

$$L(w) = \log(\frac{p(w=0)}{p(w=1)}). \tag{13}$$

If $p(w=0) > p(w=1)$ (resp. $p(w=1) > p(w=0)$) then $L(w)$ is positive (resp. negative). The larger the magnitude of $L(w)$, the higher the probability that w would equal to zero (resp. one).

From the Bayes formula, a posteriori probability is positively correlated to likelihood and a priori probability. Then, we can compute a posteriori LLR as follows:

$$\log \frac{p(w^*=0|w)}{p(w^*=1|w)} = \log \frac{p(w|w^*=0)}{p(w|w^*=1)} + \log \frac{p(w^*=0)}{p(w^*=1)}, \tag{14}$$

where w^* is the original sign of the edge and w is the observed sign. On the right-hand side, the first term is the log-likelihood ratio and the second term is a priori LLR. The log-likelihood ratio is called the intrinsic information and can be computed for the binary symmetric channel as follows:

$$\log \frac{p(w_j|w_j^*=0)}{p(w_j|w_j^*=1)} = \log(\frac{1-p}{p})w_j. \tag{15}$$

We denote a posteriori LLR as the variable-to-check message $L_{v_j \to c_i}$ that propagates from variable node v_j to check node c_i. Such a message can be calculated according to

$$L_{v_j \to c_i} = \sum_{c_a \in \mathcal{N}(v_j) \setminus c_i} m_{c_a \to v_j} + L_{v_j}. \tag{16}$$

where L_{v_j} is the intrinsic LLR of variable node v_j.

A priori LLR is denoted as the check-to-variable message $m_{c_i \to v_j}$ that propagates from check node c_i to variable node v_j. Such a message is generated according to

$$m_{c_i \to v_j} = 2 \tanh^{-1} \left(\prod_{v_b \in N(c_i) \setminus v_j} \tanh \left(\frac{L_{v_b \to c_i}}{2} \right) \right). \tag{17}$$

ALGORITHM 3: The Belief Propagation Algorithm

Input: A signed network $G_s = (V, E, W)$.
Output: A partition $\mathcal{P} = \{S_1, S_2\}$.
(1) Initially, set $L_{v_j} = w_j \log(\frac{p}{1-p})$.
(2) For each check node c_i, update $m_{c_i \to v_j}$ by using (17).
(3) For each variable node v_j, update $L_{v_j \to c_i}$ by using (16). and update L_{v_j} by using (18)
(4) Compute the decision value $\hat{w}_j$ by using (19).
(5) Repeat Steps 1 to 4 until all of the parity-check constraints are satisfied or a predefined maximum number of iterations is reached.
(6) Detect two groups by the *node coloring* algorithm in Algorithm 1.

Thus, we can compute the LLR value of variable node v_j by using

$$L(v_j) = \sum_{c_i \in \mathcal{N}(v_j)} m_{c_i \to v_j} + L_{v_j}. \tag{18}$$

To estimate the value of the j^{th} variable node v_j, we can make a hard decision according to

$$\hat{w}_j = \begin{cases} 0, & \text{if } L(v_j) \geq 0 \\ 1, & \text{if } L(v_j) < 0 \end{cases}. \tag{19}$$

With a hard decision value $\hat{\mathbf{w}}$, we can compute the syndrome s by using (9). If all of the syndrome components are zero, then $\hat{\mathbf{w}}$ is a codeword for the graph $G = (V, E)$. Conversely, If $\hat{\mathbf{w}}$ does not satisfy all the parity-check constraints, we continue the iteration until all parity-check constraints are satisfied or a predefined maximum number of iterations is reached. We summarize the belief propagation algorithm in Algorithm 3.

E. The Hamming distance algorithm

The Hamming distance between two binary vectors is the number of different bits between these two binary vectors. For a signed network $G_s(V, E, W)$, let $\mathbf{w}$ be its weight vector and $\mathbf{w}(\mathcal{P})$ be the weight vector associated with the partition $\mathcal{P} = \{S_1, S_2\}$ of the nodes in the graph $G = (V, E)$. The Hamming distance algorithm aims to find a codeword that has the minimum Hamming distance to the received signal. In the following lemma, we first show how to compute the Hamming distance by computing the number of positive (resp. negative) edges between two sets.

Lemma 4: The Hamming distance between $\mathbf{w}$ and $\mathbf{w}(\mathcal{P})$, denoted by $d(\mathcal{P})$, can be computed as follows:

$$\begin{aligned} d(\mathcal{P}) &= \frac{1}{2}\Big(N^-(S_1, S_1) + N^+(S_1, S_2) \\ &\quad + N^+(S_2, S_1) + N^-(S_2, S_2)\Big), \end{aligned} \tag{20}$$

where

$$N^-(S_i, S_j) = \sum_{u \in S_i, v \in S_j} \mathbf{1}\{w(u, v) = 1\} \tag{21}$$

is the number of negative edges from the set S_i to the set S_j, and

$$N^+(S_i, S_j) = \sum_{u \in S_i, v \in S_j} \mathbf{1}\{w(u,v) = 0\} \tag{22}$$

is the number of positive edges from the set S_i to the set S_j.

Proof. Note that

$$\begin{aligned} d(\mathcal{P}) &= \sum_{i=1}^{m} \mathbf{1}\{w_i = 1, w(\mathcal{P})_i = 0\} \\ &\quad + \sum_{i=1}^{m} \mathbf{1}\{w_i = 0, w(\mathcal{P})_i = 1\}. \end{aligned} \tag{23}$$

For the i^{th} edge, we know that $w(\mathcal{P})_i = 0$ if both ends of the i^{th} edge belong to the same set. Thus,

$$\begin{aligned} &\sum_{i=1}^{m} \mathbf{1}\{w_i = 1, w(\mathcal{P})_i = 0\} \\ &= \frac{1}{2} N^-(S_1, S_1) + \frac{1}{2} N^-(S_2, S_2). \end{aligned}$$

On other hand, $w(\mathcal{P})_i = 1$ if one end of the i^{th} edge belongs to one set and the other end belongs to the other set. Thus,

$$\begin{aligned} &\sum_{i=1}^{m} \mathbf{1}\{w_i = 0, w(\mathcal{P})_i = 1\} \\ &= \frac{1}{2} N^+(S_1, S_2) + \frac{1}{2} N^+(S_2, S_1). \end{aligned}$$

■

In general, there is no efficient algorithm to find the optimal $\mathbf{w}(\mathcal{P})$ that minimizes $d(\mathcal{P})$. Here we propose a heuristic algorithm in Algorithm 4 that finds a local optimum. Such an algorithm is related to the partitional algorithm in [7]. Algorithm 4 can be clearly described by defining the correlation measure between a node v and a set S as the difference of the number of positive edges and the number of negative edges from v to S, i.e.,

$$q(v, S) = N^+(\{v\}, S) - N^-(\{v\}, S). \tag{24}$$

With this correlation measure, Algorithm 4 is simply a local search algorithm that iteratively assigns each node to the most correlated set.

Unlike the bit-flipping algorithm and the belief propagation algorithm, we show in the following theorem that the Hamming distance algorithm in 4 is guaranteed to converge within a finite number of steps. The proof of Theorem 5 is given in Appendix A.

Theorem 5: In Algorithm 4, the Hamming distance is non-increasing when there is a change, i.e., a node is moved from one set to another. Thus, the algorithm converges to a local minimum of the Hamming distance in a finite number of steps.

ALGORITHM 4: The Hamming Distance Algorithm

Input: A signed network $G_s = (V, E, W)$.
Output: A partition $\mathcal{P} = \{S_1, S_2\}$.
(0) Initially, choose arbitrarily two disjoint nonempty sets S_1 and S_2 as a partition of the n nodes in $G = (V, E)$.
(1) for $v = 1, 2, \ldots, n$ **do**
  Compute the correlation measures $q(v, S_1)$ and $q(v, S_2)$ in (24).
  If the two correlation measures are the same, node v remains in the original set. Otherwise, assign node v to the set with a larger correlation measure.
end
(2) Repeat from Step 1 until there is no further change.

IV. Experimental results

A. Community detection with two communities

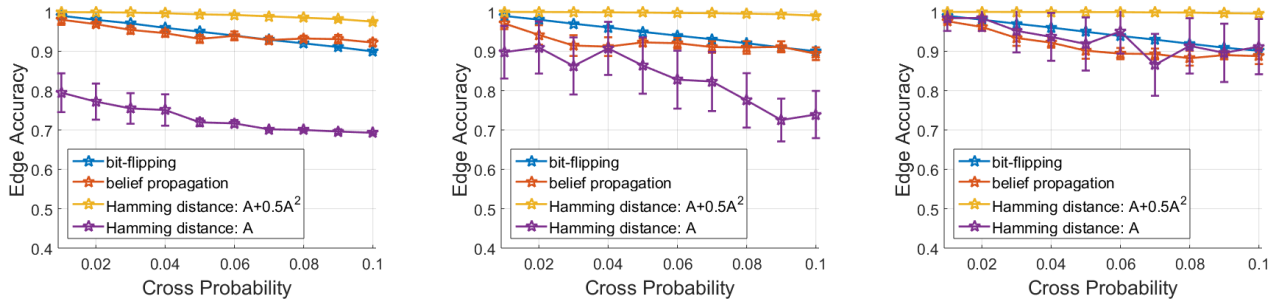
In this section, we conduct experiments for these ECC algorithms by using the stochastic block model. The stochastic block model, commonly used for benchmarking community detection algorithms, is a generalization of the Erdös-Rényi model. In the stochastic block model with n nodes and two blocks, the two blocks are equally sized with $n/2$ nodes. The parameter p_{in} is the probability that there is a positive edge between two nodes within the same block and p_{out} is the probability that there is a negative edge between two nodes in two different blocks. All edges are generated independently according to p_{in} and p_{out}. Let $c_{in} = np_{in}$ and $c_{out} = np_{out}$. Clearly, such a construction generates a structurally balanced signed network with two ground truth communities.

To generate a signed network that is not structurally balanced, we randomly flip the sign of an edge in the stochastic block model with the crossover probability p. Clearly, if p is small, the signed network is not too far from the original structurally balanced signed network and it is more likely that we can recover the original signed network. The method of random bit-flipping corresponds to the binary symmetric channel in a communication system.

In our experiments, the total number of nodes in the stochastic block model is 2000 with 1000 nodes in each block. The parameter c denotes the average degree of a node that is set to be 6, 8, and 10. The value of $c_{in} - c_{out}$ is set to be 5. The crossover probability p is in the range from 0.01 to 0.1 with a common step of 0.01. We generate 20 graphs for each p and c. We remove isolated nodes, and thus the exact numbers of nodes in the experiments might be less than 2000. We show the experimental results with each point averaged over 20 random graphs. The error bars represent the 95% confident intervals.

For the bit-flipping algorithm (resp. belief propagation algorithm), the maximum number of iterations is set to be 20 (resp. 100). To test the Hamming distance algorithm, we conduct our experiments with two adjacency matrices: one with the original adjacency matrix A and the other with the two-step adjacency matrix

$$\widehat{A} = A + 0.5A^2. \tag{25}$$



(a) average degree=6 (b) average degree=8 (c) average degree=10

Fig. 3. Community detection with two communities.

The intuition of using the two-step adjacency matrix is that it allows us to "see" more than one step relationship between two nodes. According to Heider's balance theory [18], "an enemy of my enemy is likely to be my friend" and "a friend of my friend is likely to be my friend." The two-step adjacency matrix in (25) somehow predicts the two-step relationship between two nodes and thus makes the signed network more dense and complete.

We show our experimental results in Figure 3. The Hamming distance algorithm with the two-step adjacency matrix has the largest edge accuracy (defined as the ratio of the number of accurately decoded edges to the total number of edges) for the entire range of the cross probability. It seems that the corrupted edges are all corrected by using the Hamming distance algorithm with the two-step adjacency matrix. As explained before, the reason that the Hamming distance algorithm with the two-step adjacency matrix is better than that with the original adjacency matrix is because the former can "see" more than one step relationship between two nodes. From Figure 3, both the bit-flipping algorithm and the belief propagation algorithm do not perform well for community detection in signed networks. One possible explanation is that signed networks in general do not correspond to good error-correcting codes. This is because there might exist the girth-4 problem (see e.g., [19], [20]), i.e., the bipartite graph constructed from a fundamental cycle matrix might have cycles of length four and messages are likely to be trapped in short cycles. The girth-4 problem can be avoided by using some known methods in [19], [20] by constructing a new parity-check matrix. However, these methods cannot be used here as the parity-check matrix $\mathbf{H}$ is constructed from the spanning tree of a random graph.

From our experimental results, it seems that the performance of the bit-flipping algorithm is slightly better than that of the belief propagation algorithm. Also, increasing the degree in the stochastic block model seems to have a positive effect on the Hamming distance algorithm with the original adjacency matrix. This might be due to the fact that the tested signed networks are more dense. However, increasing the degree in the stochastic block model seems to result in little improvement for the other two algorithms.

B. Community detection with a real dataset

In this section, we report the experimental results for the three ECC algorithms based on the political blogs dataset [21]. The dataset contains a directed citation network which is based on a single day snapshot of 1494 political blogs. Each link is established if there is hyperlink from one blog to another blog. For each node, there is an attribute indicating the political orientation (i.e., conservative or liberal) of the blog. To generate a signed network, we convert the network into an undirected graph and label the edges within the same community (resp. between two communities) to +1 (resp. -1). Then, we flip the signs of edges randomly as we did in Section IV-A. To ensure the connectivity, we remove all the isolated nodes. By doing so, the number of nodes is down to 1222 and the average degree is about 31. The sizes of these two communities are 586 and 636, respectively. All the remaining setup and parameters for each algorithm are the same as in Section IV-A. The main difference between the political blogs dataset and the stochastic block model is that the political blogs dataset has different edge densities in the two communities. The conservative community has a denser edge density than that of the liberal community. The experimental results in Figure 4 are consistent with our early findings in Section IV-A for the stochastic block models.

V. Conclusion

In this paper, we considered the community detection problem in signed networks. By using Harary's theorem, we showed that the community detection problem in a signed network with two communities is equivalent to the decoding problem for a parity-check code. To the best of our knowledge, this is the first result that links error-correcting codes to the community detection problem in signed networks. We also showed how the bit-flipping algorithm, the belief propagation algorithm, and the Hamming distance algorithm can be used for community detection in signed networks. We compared the performance of these three algorithms by conducting various experiments with known ground truth. Our experimental results show that the Hamming distance algorithm outperforms

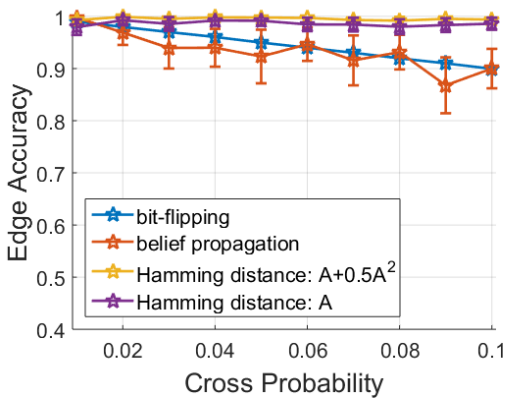


Fig. 4. Community detection for the political blogs dataset [21].

the other two. One possible explanation for this is that signed networks are in general not good error-correcting codes as there might be short cycles in such networks. As such, the bit-flipping algorithm and the belief propagation algorithm do not work well for community detection in signed networks.

APPENDIX A

In this section, we prove Theorem 5.

It suffices to show that if node v is in a set S_1 and $q(v, S_2) > q(v, S_1)$, then move node v from S_1 to S_2 decreases the Hamming distance. Also let $\mathcal{P}$ (resp. $\mathcal{P}'$) be the partition before (resp. after) the change. Note that $N^+(\cdot,\cdot)$ and $N^-(\cdot,\cdot)$ are symmetric, $N^+(v,v) = 0$, and $N^-(v,v) = 0$. It follows from (20) that

$$\begin{aligned}
&d(\mathcal{P}') - d(\mathcal{P}) \\
&= \frac{1}{2}\Big(N^-(S_1\backslash\{v\}, S_1\backslash\{v\}) + N^+(S_1\backslash\{v\}, S_2\cup\{v\}) \\
&\quad + N^+(S_2\cup\{v\}, S_1\backslash\{v\}) + N^-(S_2\cup\{v\}, S_2\cup\{v\})\Big) \\
&\quad - \frac{1}{2}\Big(N^-(S_1, S_1) + N^+(S_1, S_2) \\
&\quad + N^+(S_2, S_1) + N^-(S_2, S_2)\Big).
\end{aligned}$$

Note that

$$N^-(S_1\backslash\{v\}, S_1\backslash\{v\}) - N^-(S_1, S_1) = -2N^-(\{v\}, S_1),$$

$$N^-(S_2\cup\{v\}, S_2\cup\{v\}) - N^-(S_2, S_2) = 2N^-(\{v\}, S_2),$$

and

$$\begin{aligned}
&N^+(S_1\backslash\{v\}, S_2\cup\{v\}) - N^+(S_1, S_2) \\
&= N^+(S_1\backslash\{v\}, \{v\}) + N^+(S_1\backslash\{v\}, S_2) \\
&\quad - N^+(S_1\backslash\{v\}, S_2) - N^+(\{v\}, S_2) \\
&= N^+(S_1\backslash\{v\}, \{v\}) - N^+(\{v\}, S_2) \\
&= N^+(S_1, \{v\}) - N^+(\{v\}, S_2) \\
&= N^+(\{v\}, S_1) - N^+(\{v\}, S_2).
\end{aligned}$$

Thus,

$$\begin{aligned}
&d(\mathcal{P}') - d(\mathcal{P}) \\
&= N^-(\{v\}, S_2) - N^-(\{v\}, S_1) \\
&\quad + N^+(\{v\}, S_1) - N^+(\{v\}, S_2) \\
&= q(v, S_1) - q(v, S_2) < 0.
\end{aligned}$$

As the Hamming distance is non-increasing after a change of the partition, there is no loop in the algorithm. Since the number of partitions is finite, the algorithm thus converges in a finite number of steps.

REFERENCES

[1] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical Review E*, vol. 80, no. 5, p. 056117, 2009.

[2] M. Newman, *Networks: an introduction*. Oxford university press, 2010.

[3] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics Reports*, vol. 533, no. 4, pp. 95–142, 2013.

[4] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *arXiv preprint arXiv:1511.07569*, 2015.

[5] D. Cartwright and F. Harary, "Structural balance: a generalization of Heider's theory." *Psychological Review*, vol. 63, no. 5, p. 277, 1956.

[6] F. Harary *et al.*, "On the notion of balance of a signed graph." *The Michigan Mathematical Journal*, vol. 2, no. 2, pp. 143–146, 1953.

[7] P. Doreian and A. Mrvar, "A partitioning approach to structural balance," *Social Networks*, vol. 18, no. 2, pp. 149–168, 1996.

[8] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, 2004.

[9] S. Gómez, P. Jensen, and A. Arenas, "Analysis of community structure in networks of correlated data," *Physical Review E*, vol. 80, no. 1, p. 016114, 2009.

[10] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak, "Spectral analysis of signed graphs for clustering, prediction and visualization." in *SDM*, vol. 10. SIAM, 2010, pp. 559–559.

[11] D. B. West *et al.*, *Introduction to graph theory*. Prentice Hall Upper Saddle River, 2001, vol. 2.

[12] R. E. Blahut, *Theory and practice of error control codes*. Addison-Wesley Reading (Ma) etc., 1983, vol. 126.

[13] J. D. Horton, "A polynomial-time algorithm to find the shortest cycle basis of a graph," *SIAM Journal on Computing*, vol. 16, no. 2, pp. 358–366, 1987.

[14] E. Abbe, A. S. Bandeira, A. Bracher, and A. Singer, "Decoding binary node labels from censored edge measurements: Phase transition and efficient recovery," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 1, pp. 10–22, 2014.

[15] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[16] Y. Kou, S. Lin, and M. P. Fossorier, "Low-density parity-check codes based on finite geometries: a rediscovery and new results," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2711–2736, 2001.

[17] H.-C. Lee, Y.-L. Ueng, S.-M. Yeh, and W.-Y. Weng, "Two informed dynamic scheduling strategies for iterative LDPC decoders," *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 886–896, Mar. 2013.

[18] F. Heider, "Attitudes and cognitive organization," *The Journal of Psychology*, vol. 21, no. 1, pp. 107–112, 1946.

[19] H. Zhang and J. M. Moura, "The design of structured regular LDPC codes with large girth," in *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, vol. 7. IEEE, 2003, pp. 4022–4027.

[20] J. Fan, Y. Xiao, and K. Kim, "Design LDPC codes without cycles of length 4 and 6," *Research Letters in Communications*, vol. 2008, p. 4, 2008.

[21] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 US election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.