A Probabilistic Framework for Structural Analysis and Community Detection in Directed Networks

Cheng-Shang Chang, *Fellow, IEEE,* Duan-Shin Lee, *Senior Member, IEEE,* Li-Heng Liou, Sheng-Min Lu, and Mu-Huan Wu

***Abstract*—There is growing interest in structural analysis of *directed* networks. Two major points that need to be addressed are (i) a formal and precise definition of the graph clustering and community detection problem in directed networks, and (ii) algorithm design and evaluation of community detection algorithms in directed networks. Motivated by these, we develop a probabilistic framework for structural analysis and community detection in *directed* networks based on our previous work in *undirected* networks. By relaxing the assumption from *symmetric* bivariate distributions in our previous work to bivariate distributions that have the same marginal distributions in this paper, we can still formally define various notions for structural analysis in directed networks, including centrality, relative centrality, community, and modularity. We also extend three commonly used community detection algorithms in *undirected* networks to *directed* networks: the hierarchical agglomerative algorithm, the partitional algorithm, and the fast unfolding algorithm. These are made possible by two modularity preserving and sparsity preserving transformations. In conjunction with the probabilistic framework, we show these three algorithms converge in a finite number of steps. In particular, we show that the partitional algorithm is a linear time algorithm for large sparse graphs. Moreover, the outputs of the hierarchical agglomerative algorithm and the fast unfolding algorithm are guaranteed to be *communities*. These three algorithms can also be extended to general bivariate distributions with some minor modifications. We also conduct various experiments by using two sampling methods in directed networks: (i) PageRank and (ii) random walks with self-loops and backward jumps.**

keywords: centrality, community, modularity, PageRank

I. Introduction

As the advent of on-line social networks, structural analysis of networks has been a very hot research topic. There are various notions that are widely used for structural analysis of networks, including centrality, relative centrality, similarity, community, modularity, and homophily (see e.g., the book by Newman [2]). In order to make these notions more mathematically precise, we developed in [3], [4] a probabilistic framework for structural analysis of *undirected* networks. The key idea of the framework is to model a network as a graph and "sample" the graph to generate a bivariate distribution $p(v, w)$ that specifies the probability that a pair of two nodes v and w are selected from a sample, e.g., the probability that v and w appear respectively at the two ends of a "randomly" selected path in the graph. The bivariate distribution $p(v, w)$ can be viewed as a normalized *similarity* measure [5] between the two nodes v and w and thus provides a certain viewpoint to the graph. A graph G associated with a bivariate distribution $p(\cdot, \cdot)$ is then called a *sampled* graph.

In [3], [4], the bivariate distribution is assumed to be *symmetric*. Under this assumption, the two marginal distributions of the bivariate distribution, denoted by $p_V(\cdot)$ and $p_W(\cdot)$, are the same and they represent the probability that a particular node is selected in the sampled graph. As such, the marginal distribution $p_V(v)$ can be used for defining the *centrality* of a node v as it represents the probability that node v is selected. The relative centrality of a set of nodes S_1 with respect to another set of nodes S_2 is then defined as the *conditional* probability that one node of the selected pair of two nodes is in the set S_1 given that the other node is in the set S_2. Based on the probabilistic definitions of centrality and relative centrality in the framework, the *community strength* for a set of nodes S is defined as the difference between its relative centrality with respect to itself and its centrality. Moreover, a set of nodes with a *nonnegative* community strength is called a *community*. Intuitively, if the bivariate distribution $p(v, w)$ is the probability for v and w to appear respectively at the two ends of a randomly selected path, then a community is a set of nodes with the property that it is more likely to find the other end in the same community given one of the two ends in a randomly selected path is already in the community. In the probabilistic framework, the *modularity* for a partition of a sampled graph is defined as the average community strength of the community. As such, a high modularity for a partition of a graph implies that there are communities with strong community strengths. It was further shown in [4] that the Newman modularity in [6] and the stability in [7], [8] are special cases of the modularity for certain sampled graphs.

As pointed out in the recent survey [9], most networks in sociology, biology, neuroscience and computer science, are *directed* and one major point that needs to be addressed is "a formal and precise definition of the graph clustering and community detection problem in directed networks." Motivated by this, our main objective of this paper is to extend the probabilistic framework in [3], [4] to *directed* networks, where the sampling bivariate distributions could be *asymmetric*. Our main finding is that we can relax the assumption from *symmetric* bivariate distributions to bivariate distributions that have the same marginal distributions. By using such a weaker assumption, we show that the notions of centrality, relative centrality, community and modularity can be defined in the same manner as before. Moreover, the

C.-S. Chang, D.-S. Lee, L.-H. Liou, S.-M. Lu and M.-H. Wu are with the 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; u9661106@oz.nthu.edu.tw.

The conference version of this paper was presented in IEEE International Conference on Communications 23-27 May 2016, Kuala Lumpur, Malaysia [1].

equivalent characterizations of a community still hold.

Another point that needs to be addressed in [9] is algorithm design and evaluation of community detection algorithms in directed networks. For this, we extend three commonly used community detection algorithms in *undirected* networks to *directed* networks: (i) the hierarchical agglomerative algorithm [10], (ii) the partitional algorithm [11], and (iii) the fast unfolding algorithm [12]. Since the bivariate distribution could be *asymmetric*, these community detection algorithms cannot be directly applied. For this, we show two modularity preserving transformations: (i) the transformation from a sampled graph that has the same marginal distributions to another sampled graph that has a symmetric bivariate distribution, (ii) the transformation from a sampled graph with a larger number of nodes to another sampled graph with a smaller number of nodes by node aggregation. These two transformations not only preserve modularity but also preserve sparsity (of the sampled graph). As such, the computational complexity of these three algorithms for directed networks remains the same as that of their undirected counterparts. In conjunction with the probabilistic framework, we show these three algorithms converge in a finite number of steps. In particular, we show that the partitional algorithm is a linear time algorithm for large sparse graphs. Moreover, the outputs of the hierarchical agglomerative algorithm and the fast unfolding algorithm are guaranteed to be *communities*. Though there are several variants of fast unfolding algorithms in the literature, it seems (to the best our knowledge) that our fast unfolding algorithm is the first one that has provable guarantees and also is general enough to be applicable to directed networks.

Further extension to the setting with general bivariate distributions is possible. However, the results are not as elegant as the setting with bivariate distributions that have the same marginal distributions. The good news is that the notions of *community* and *modularity* can be extended in the same manner. Moreover, the hierarchical agglomerative algorithm, the partitional algorithm, and the fast unfolding algorithm can also be extended to the setting with some minor modifications.

To test these algorithms, we consider two methods for sampling a directed network with a bivariate distribution that has the same marginal distributions : (i) PageRank and (ii) random walks with self-loops and backward jumps. Though PageRank [13] has been very successful in ranking nodes in directed networks, our experimental results show that its performance in community detection is not as good as sampling by a random walk with self-loops and backward jumps. This might be due to the fact that PageRank adds weak links in a network and that changes the topology of the network and thus affects the results of community detection. In [14], the authors pointed out a similar problem for the original PageRank and provided various ways to modify PageRank to tackle this problem.

We summarize the logic flow of using sampled graphs for structural analysis and community detection in Figure 1. For a graph $G = (V_g, E_g)$, we provide a viewpoint by sampling the graph with the bivariate distribution $p(\cdot, \cdot)$. This results in a sampled graph. Based on this sampled graph, we can then define the notions of relative centrality and centrality. The notions of community and modularity are built upon the notions of relative centrality and centrality. Ranking algorithms and community detection algorithms can then be developed and written in codes by using sampled graphs as inputs. If one is not satisfied with the community detection result from a sampled graph, e.g., the resolution, one has the freedom to choose another viewpoint for the graph and try out the analysis again. There is no need to rewrite the codes for the community detection algorithm.

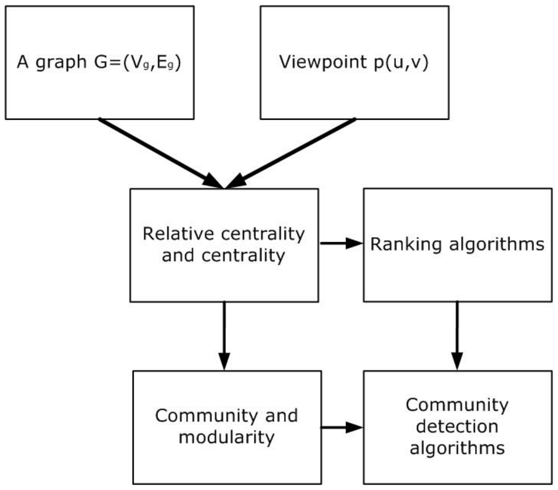


Fig. 1. The logic flow of using sampled graphs for structural analysis and community detection.

The rest of the paper is organized as follows. In Section II, we introduce the sampled graph with a bivariate distribution that has the same marginal distributions. Various notions for structural analysis in directed networks are then defined in Section III. In Section IV, we propose three community detection algorithms for directed networks. Extensions to sampled graphs with general bivariate distributions are addressed in Section V. We then conduct various experiments to test these algorithms in Section VI. The paper is concluded in Section VII.

II. Sampling networks by bivariate distributions with the same marginal distributions

In [4], a probabilistic framework for network analysis for undirected networks was proposed. The main idea in that framework is to characterize a network by a *sampled graph*. Specifically, suppose a network is modelled by a graph $G(V_g, E_g)$, where V_g denotes the set of vertices (nodes) in the graph and E_g denotes the set of edges (links) in the graph. Let $n = |V_g|$ be the number of vertices in the graph and index the n vertices from $1, 2, \ldots, n$. Also, let $A = (a_{ij})$ be the $n \times n$ adjacency matrix of the graph, i.e.,

$$a_{ij} = \begin{cases} 1, & \text{if there is an edge from vertex } i \text{ to vertex } j, \\ 0, & \text{otherwise.} \end{cases}$$

A sampling bivariate distribution $p(\cdot, \cdot)$ for a graph G is the bivariate distribution that is used for *sampling* a network by randomly selecting an ordered pair of two nodes (V, W), i.e.,

$$\mathsf{P}(V = v, W = w) = p(v, w). \tag{1}$$

Let $p_V(v)$ (resp. $p_W(w)$) be the marginal distribution of the random variable V (resp. W), i.e.,

$$p_V(v) = \mathsf{P}(V = v) = \sum_{w=1}^{n} p(v, w), \tag{2}$$

and

$$p_W(w) = \mathsf{P}(W = w) = \sum_{v=1}^{n} p(v, w). \tag{3}$$

Definition 1: **(Sampled graph)** A graph $G(V_g, E_g)$ that is sampled by randomly selecting an ordered pair of two nodes (V, W) according to a specific bivariate distribution $p(\cdot, \cdot)$ in (1) is called a *sampled graph* and it is denoted by the two-tuple $(G(V_g, E_g), p(\cdot, \cdot))$.

For a given graph $G(V_g, E_g)$, there are many methods to generate sampled graphs by specifying the needed bivariate distributions. In [4], the bivariate distributions are all assumed to be *symmetric* and that limits its applicability to *undirected* networks. One of the main objectives of this paper is to relax the *symmetric* assumption for the bivariate distribution so that the framework can be applied to *directed* networks. The key idea of doing this is to assume that the bivariate distribution has the same marginal distributions, i.e.,

$$p_V(v) = p_W(v), \quad \text{for all } v. \tag{4}$$

As pointed out in [15], the bivariate distribution that has the same marginal distributions, called *circulation* in [15], plays an important role in analyzing directed networks. Note that a symmetric bivariate distribution has the same marginal distributions and thus the assumption in (4) is much more general.

Extensions to sampled graphs with general bivariate distributions will be addressed in Section V. One particular sampling method is the uniform edge sampling method with $p(v, w) = a_{vw}/m$, where m is the total number of edges in the directed network. Such a sampling method corresponds to the modularity in [16]. If one uses the uniform edge sampling method, then there is a one-to-one mapping between the bivariate distribution $p(v, w)$ and the adjacency matrix of the network (if the total number of edges is known). In that case, one still has the complete information of the network. On the other hand, if one uses the random walk with self-loops and backward jumps or the sampling method with paths of length 2 (described later in the paper), then the adjacency matrix of the network cannot be recovered. The loss of the complete information of the adjacency matrix might not be crucial for the community detection problem.

A. PageRank

One approach for sampling a network with a bivariate distribution that has the same marginal distributions is to sample a network by an *ergodic Markov chain*. From the Markov chain theory (see e.g., [17]), it is well-known that an ergodic Markov chain converges to its steady state in the long run. Hence, the joint distribution of two successive steps of a *stationary and ergodic* Markov chain can be used as the needed bivariate distribution. Specifically, suppose that a network $G(V_g, E_g)$ is sampled by a stationary and ergodic Markov chain $\{X(t), t \geq 0\}$ with the state space $\{1, 2, \ldots, n\}$ being the n nodes in V_g. For this Markov chain, let p_{ij} be the transition probability from state i to state j and π_i be the steady state probability of state i. Then we can choose the bivariate distribution

$$\begin{aligned} &\mathsf{P}(V = v, W = w) = p(v, w) \\ &= \mathsf{P}(X(t) = v, X(t+1) = w). \end{aligned} \tag{5}$$

As the Markov chain is stationary, we have

$$\mathsf{P}(X(t) = v) = \mathsf{P}(X(t+1) = v) = p_V(v) = p_W(v). \tag{6}$$

It is well-known that a random walk on the graph induces a Markov chain with the following state transition probabilities:

$$p_{ij} = \frac{a_{ij}}{k_i^{out}}, \tag{7}$$

where

$$k_i^{out} = \sum_{j=1}^{n} a_{ij}, \tag{8}$$

is the number of outgoing edges from vertex i. In particular, if the graph is an undirected graph, i.e., $a_{ij} = a_{ji}$, then the induced Markov chain is reversible and the steady state probability of state i, i.e., π_i, is $k_i/2m$, where $m = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$ is the total number of edges of the undirected graph.

One problem for sampling a *directed* network by a simple random walk is that the induced Markov chain may not be ergodic even when the network itself is weakly connected. One genuine solution for this is to allow random jumps from states to states in a random walk. PageRank [13], proposed by Google, is one such example that has been successfully used for ranking web pages. The key idea behind PageRank is to model the behavior of a web surfer by a random walk (the random surfer model) and then use that to compute the steady state probability for a web surfer to visit a specific web page. Specifically, suppose that there are n web pages and a web surfer uniformly selects a web page with probability $1/n$. Once he/she is on a web page, he/she continues web surfing with probability λ. This is done by selecting *uniformly* one of the hyperlinks in that web page. On the other hand, with probability $1-\lambda$ he/she starts a new web page *uniformly* among all the n web pages. The transition probability from state i to state j for the induced Markov chain is then

$$p_{ij} = (1 - \lambda)\frac{1}{n} + \lambda \frac{a_{ij}}{k_i^{out}}, \tag{9}$$

where $a_{ij} = 1$ if there is a hyperlink pointing from the i^{th} web page to the j^{th} web page and $k_i^{out} = \sum_{j=1}^{n} a_{ij}$ is the total number of hyperlinks on the i^{th} web page. Let π_i be steady probability of visiting the i^{th} web page by the web surfer. It then follows that

$$\pi_i = (1 - \lambda)\frac{1}{n} + \lambda \sum_{j=1}^{n} \frac{a_{ji}}{k_j^{out}} \pi_j. \tag{10}$$

PageRank then uses π_i as the centrality of the i^{th} web page and rank web pages by their centralities. Unlike the random walk on an undirected graph, the steady state probabilities in (10) cannot be explicitly solved and it requires a lot of computation to solve the system of linear equations.

The sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ by using PageRank then has the following bivariate distribution

$$p(v, w) = \pi_v p_{vw}, \tag{11}$$

where p_{vw} is defined in (9) and π_v is the solution of (10). Such a sampled graph was called LinkRank in [18].

B. Random walks with self-loops and backward jumps

Another way to look at the Markov chain induced by PageRank in (9) is that it is in fact a random walk on a different graph with the adjacency matrix $\tilde{A}$ that is constructed from the original graph with additional edge weights, i.e.,

$$\tilde{A} = (1-\lambda)\frac{1}{n}\mathbf{1} + \lambda D^{-1}A, \tag{12}$$

where $\mathbf{1}$ is an $n \times n$ matrix with all its elements being 1 and $D = (d_{ij})$ is the diagonal matrix with $d_{ii} = k_i^{out}$ for all $i = 1, 2, \ldots, n$.

In view of (12), another solution for the ergodic problem is to consider a random walk on the graph with the adjacency matrix

$$\hat{A} = \lambda_0\mathbf{I} + \lambda_1 A + \lambda_2 A^T, \tag{13}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix and A^T is the transpose matrix of A. The three parameters $\lambda_0, \lambda_1, \lambda_2$ are positive and

$$\lambda_0 + \lambda_1 + \lambda_2 = 1.$$

A random walk on the graph with the adjacency matrix $\hat{A}$ induces an ergodic Markov chain if the original graph is weakly connected. Also, with the additional edges from the identity matrix and the transpose matrix, such a random walk can be intuitively viewed as a random walk on the original graph with self-loops and backward jumps.

In addition to random walks, one can also consider using *diffusion* in a network to obtain the corresponding Markov chain. Specifically, let $k_{\max}^{out} = \max_{1\le i\le n} k_i^{out}$, $\mathbf{I}$ be the $n \times n$ identity matrix and $D = (d_{ij})$ be the diagonal matrix with $d_{ii} = k_i^{out}$ for all $i = 1, 2, \ldots, n$. The transition probability matrix is then $\mathbf{I} - \alpha(D - A)$, where $\alpha \le 1/k_{\max}^{out}$ is a normalization coefficient and $D - A$ is the graph Laplacian. For other Laplacian related methods, we refer to the survey paper [9].

In general, one can generate a bivariate distribution that has the same marginal distributions by using a *stationary* and *ergodic* stochastic process, e.g., a semi-Markov process or a hidden Markov process.

III. The framework for directed networks

A. Centrality and relative centrality

Centrality [19], [20], [2] is usually used as a measure for ranking the importance of a set of nodes in a (social) network. Under the assumption in (4), such a concept can be directly mapped to the probability that a node is selected as in [4].

Definition 2: **(Centrality)** For a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with the bivariate distribution $p(\cdot,\cdot)$ that has the same marginal distributions in (4), the *centrality* of a set of nodes S, denoted by $C(S)$, is defined as the probability that a node in S is selected, i.e.,

$$C(S) = \mathsf{P}(V \in S) = \mathsf{P}(W \in S). \tag{14}$$

As a generalization of centrality, relative centrality in [4] is a (probability) measure that measures how important a set of nodes in a network is with respect to another set of nodes.

Definition 3: **(Relative centrality)** For a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with the bivariate distribution $p(\cdot,\cdot)$ that has the same marginal distributions in (4), the *relative centrality* of a set of nodes S_1 with respect to another set of nodes S_2, denoted by $C(S_1|S_2)$, is defined as the conditional probability that the randomly selected node W is inside S_1 given that the randomly selected node V is inside S_2, i.e.,

$$C(S_1|S_2) = \mathsf{P}(W \in S_1 | V \in S_2). \tag{15}$$

We note that if we choose $S_2 = V_g$, then the relative centrality of a set of nodes S_1 with respect to V_g is simply the *centrality* of the set of nodes S_1.

Example 4: **(Relative PageRank)** PageRank described in Section II-A has been commonly used for ranking the importance of nodes in a directed network. Here we can use Definition 3 to define relative PageRank that can be used for ranking the relative importance of a set of nodes to another set of nodes in a directed network. Specifically, let π_i be the PageRank for node i in (10) and $p_{i,j}$ be the transition probability from state i to state j for the induced Markov chain in (9). Then the relative PageRank of a set S_1 with respect to another set S_2 is

$$\begin{aligned} C(S_1|S_2) &= \mathsf{P}(W \in S_1 | V \in S_2) \\ &= \frac{\mathsf{P}(W \in S_1, V \in S_2)}{\mathsf{P}(V \in S_2)} = \frac{\sum_{i\in S_2}\sum_{j\in S_1}\pi_i p_{ij}}{\sum_{i\in S_2}\pi_i}. \end{aligned} \tag{16}$$

Analogous to the relative centrality in [4], there are also several properties of relative centrality in Definition 3. However, the reciprocity property in Proposition 5(iv) is much weaker than that in [4]. The proof of Proposition 5 is given in Appendix A of the online-only supplement.

Proposition 5: For a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with the bivariate distribution $p(\cdot,\cdot)$ that has the same marginal distributions in (4), the following properties for the relative centrality defined in Definition 3 hold.

(i) $0 \le C(S_1|S_2) \le 1$ and $0 \le C(S_1) \le 1$. Moreover, $C(V_g|S_2) = 1$ and $C(V_g) = 1$.

(ii) (Additivity) If S_1 and S_2 are two disjoint sets., i.e., $S_1 \cap S_2$ is an empty set, then for an arbitrary set S_3,

$$C(S_1 \cup S_2|S_3) = C(S_1|S_3) + C(S_2|S_3). \tag{17}$$

In particular, when $S_3 = \{1, 2, \ldots, n\}$, we have

$$C(S_1 \cup S_2) = C(S_1) + C(S_2). \tag{18}$$

(iii) (Monotonicity) If S_1 is a subset of S_1', i.e., $S_1 \subset S_1'$, then $C(S_1|S_2) \le C(S_1'|S_2)$ and $C(S_1) \le C(S_1')$.

(iv) (Reciprocity) Let $S^c = V_g \backslash S$ be the set of nodes that are *not* in S.

$$C(S)C(S^c|S) = C(S^c)C(S|S^c).$$

B. Community strength and communities

The notions of community strength and modularity in [4] generalizes the original Newman's definition [10] and unifies various other generalizations, including the stability in [7], [8]. In this section, we further extend these notions to directed networks.

Definition 6: **(Community strength and communities)** For a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (4), the *community strength* of a set of nodes $S \subset V_g$, denoted by $Str(S)$, is defined as the difference of the relative centrality of S with respect to itself and its centrality, i.e.,

$$Str(S) = C(S|S) - C(S). \tag{19}$$

In particular, if a subset of nodes $S \subset V_g$ has a nonnegative community strength, i.e., $Str(S) \geq 0$, then it is called a *community*.

To see the intuition for the definition of community strength, suppose that we use a Markov chain to sample a network. For a set S that is relatively small to the total number of nodes in a network, we have $C(S) \approx 0$. In this case,

$$\begin{aligned} Str(S) &\approx C(S|S) = \mathsf{P}(W \in S | V \in S) \\ &= 1 - \frac{\mathsf{P}(W \in S^c, V \in S)}{\mathsf{P}(V \in S)} = 1 - \phi(S), \end{aligned}$$

where $\phi(S) = \mathsf{P}(W \in S^c, V \in S)/\mathsf{P}(V \in S)$ is known as the *conductance* of the Markov chain. Thus, a (small) set with a strong community strength is a set with low conductance of the Markov chain. In particular, suppose we ignore the directions of the edges in a directed network and consider the random walk on the corresponding undirected network. Then such a random walk is a reversible Markov chain with the transition probabilities

$$p_{vw} = \frac{A_{vw}}{k_v},$$

where A_{vw} is the number of edges between node v and node w, and k_v is the degree of node v. The steady state probability for such a reversible Markov chain to visit node v is

$$\pi_v = \frac{k_v}{2m},$$

where m is total number of edges in the undirected network. For such a reversible Markov chain,

$$\begin{aligned} \phi(S) &= \frac{\mathsf{P}(W \in S^c, V \in S)}{\mathsf{P}(V \in S)} \\ &= \frac{\sum_{v \in S} \sum_{w \notin S} A_{v,w}}{\sum_{v \in S} k_v}, \end{aligned}$$

which is exactly the graph conductance for a small set S in the undirected network. The graph conductance is a commonly used measure in the literature (see e.g., [21], [22]) to detect communities that are densely connected inside and sparsely connected outside.

In the following theorem, we show various equivalent statements for a set of nodes to be a community. The proof of Theorem 7 is given in Appendix B of the online-only supplement.

Theorem 7: Consider a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (4), and a set S with $0 < C(S) < 1$. Let $S^c = V_g \backslash S$ be the set of nodes that are not in S. The following statements are equivalent.

(i) The set S is a community, i.e., $Str(S) = C(S|S) - C(S) \geq 0$.

(ii) The relative centrality of S with respect to S is not less than the relative centrality of S with respect to S^c, i.e., $C(S|S) \geq C(S|S^c)$.

(iii) The relative centrality of S^c with respect to S is not greater than the centrality of S^c, i.e., $C(S^c|S) \leq C(S^c)$.

(iv) The relative centrality of S with respect to S^c is not greater than the centrality of S, i.e., $C(S|S^c) \leq C(S)$.

(v) The set S^c is a community, i.e., $Str(S^c) = C(S^c|S^c) - C(S^c) \geq 0$.

(vi) The relative centrality of S^c with respect to S^c is not less than the relative centrality of S^c with respect to S, i.e., $C(S^c|S^c) \geq C(S^c|S)$.

As discussed in [4], the social meaning of the first statement in Theorem 7(i) is that a community is a group of people who consider themselves much more important to themselves than to random people on the street. The second statement in Theorem 7(ii) says that a community is a group of people who consider themselves much more important to themselves than to the other people not in the community. The third statement in Theorem 7(iii) says that the other people not in a community are much less important to the people in the community than to random people on the street. The fourth statement in Theorem 7(iv) says that people in a community are much less important to the other people not in the community than to random people on the street. The fifth statement in Theorem 7(v) says that the other people not in a certain community are also a community. Finally, the sixth statement says that the other people not in a community are much more important to themselves than to the people in a community.

C. Modularity and modularity preserving transformations

As in [4], we define the modularity for a partition of a network as the average community strength of a randomly selected node in Definition 8.

Definition 8: **(Modularity)** Consider a sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a bivariate distribution $p(\cdot, \cdot)$ that has the same marginal distributions in (4). Let $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$, be a partition of $\{1, 2, \ldots, n\}$, i.e., $S_c \cap S_{c'}$ is an empty set for $c \neq c'$ and $\cup_{c=1}^{C} S_c = \{1, 2, \ldots, n\}$. The modularity $Q(\mathcal{P})$ with respect to the partition S_c, $c = 1, 2, \ldots, C$, is defined as the weighted average of the community strength of each subset with the weight being the centrality of each subset, i.e.,

$$Q(\mathcal{P}) = \sum_{c=1}^{C} C(S_c) \cdot Str(S_c). \tag{20}$$

We note the modularity in (20) can also be written as follows:

$$Q(\mathcal{P}) = \sum_{c=1}^{C} \mathsf{P}(V \in S_c, W \in S_c) - \mathsf{P}(V \in S_c)\mathsf{P}(W \in S_c)$$

$$= \sum_{c=1}^{C} \sum_{v \in S_c} \sum_{w \in S_c} (p(v,w) - p_V(v)p_W(w)). \tag{21}$$

As the modularity for a partition of a network is the average community strength of a randomly selected node, a good partition of a network should have a large modularity. In view of this, one can then tackle the community detection problem by looking for algorithms that yield large modularity. As the definition of modularity is the weighted sum of community strength of each community, one should note that the larger communities also have larger weights (in terms of $C(S)$). Thus, as illustrated in our experimental results later, large communities can also be found by modularity maximization even though a smaller set of nodes tends to have a strong community strength. We also note that if a set S is a community, then by Theorem 7(v) the set S^c is also a community. It is possible that the community S^c contains several communities with much larger community strengths. By doing modularity maximization, it is possible that S^c can be further partitioned into several communities with much larger community strengths.

For sampled graphs with *symmetric* bivariate distributions, there are already various community detection algorithms in [3], [4] that find local maxima of the modularity. However, they cannot be directly applied as the bivariate distributions for sampling directed networks could be *asymmetric*. For this, we show two modularity preserving transformations: (i) the transformation from a sampled graph that has the same marginal distributions to another sampled graph that has a symmetric bivariate distribution, (ii) the transformation from a sampled graph with a larger number of nodes to another sampled graphs with a smaller number of nodes by node aggregation.

In the following lemma, we first show that one can construct another sampled graph with a *symmetric* bivariate distribution so that for any partition of the network, the modularity remains the same as that of the original sampled graph. The proof of Lemma 9 is given in Appendix C of the online-only supplement.

Lemma 9: Consider a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a bivariate distribution $p(\cdot,\cdot)$ that has the same marginal distributions in (4). Construct the sampled graph $(G(V_g, E_g), \tilde{p}(\cdot,\cdot))$ with the symmetric bivariate distribution

$$\tilde{p}(v,w) = \frac{p(v,w) + p(w,v)}{2}. \tag{22}$$

Let $Q(\mathcal{P})$ (resp. $\tilde{Q}(\mathcal{P})$) be the modularity for the partition $\mathcal{P} = \{S_c, c = 1, 2, \dots, C\}$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ (resp. the sampled graph $(G(V_g, E_g), \tilde{p}(\cdot,\cdot))$). Then

$$\tilde{Q}(\mathcal{P}) = Q(\mathcal{P}). \tag{23}$$

We note the transformation in Lemma 9 is also sparsity preserving. Specifically, let m_0 be the total number of nonzero entries in the $n \times n$ matrix $P = (p(v,w))$. Then the number of nonzero entries in the $n \times n$ matrix $\tilde{P} = (\tilde{p}(v,w))$ is at most $2m_0$. Such a sparsity property is crucial for reducing the computational complexity of the community detection algorithms described in the next section.

Now we show another modularity preserving and sparsity preserving transformation by node aggregation.

Definition 10: **(Node aggregation)** Consider a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a bivariate distribution $p(\cdot,\cdot)$ that has the same marginal distributions in (4), and a partition $\mathcal{P} = \{S_c, c = 1, 2, \dots, C\}$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Now we aggregate the nodes in S_c into a giant node c, $c = 1, 2, \dots, C$, and define the bivariate distribution $\hat{p}(\cdot,\cdot)$ with

$$\hat{p}(c_1, c_2) = \sum_{u \in S_{c_1}} \sum_{v \in S_{c_2}} p(u,v), \tag{24}$$

for $c_1, c_2 = 1, 2, \dots, C$. Let $\hat{V}$ be the collection of the giant nodes c, $c = 1, 2, \dots, C$, and $\hat{E}$ be the collection of pairs (c_1, c_2) with $\hat{p}(c_1, c_2) > 0$. The new sampled graph $(G(\hat{V}, \hat{E}), \hat{p}(\cdot,\cdot))$ is called the $\mathcal{P}$-aggregated sampled graph of the sample graph $(G(V_g, E_g), p(\cdot,\cdot))$.

It is clear from Definition 10 that the new sampled graph $(G(\hat{V}, \hat{E}), \hat{p}(\cdot,\cdot))$ also has a bivariate distribution $\hat{p}(\cdot,\cdot)$ that has the same marginal distributions. In the following lemma, we further show that the node aggregation in Definition 10 is a modularity preserving transformation. Its proof is given in Appendix D of the online-only supplement.

Lemma 11: Consider a sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a bivariate distribution $p(\cdot,\cdot)$ that has the same marginal distributions in (4), and a partition $\mathcal{P} = \{S_c, c = 1, 2, \dots, C\}$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Suppose $\mathcal{P}^a = \{S_k^a, k = 1, 2, \dots, K\}$ for some $K \le C$ is a partition of the $\mathcal{P}$-aggregated sampled graph of the sample graph $(G(V_g, E_g), p(\cdot,\cdot))$. Then $\mathcal{P}^a$ induces a partition $\mathcal{P}^o = \{S_k^o, k = 1, 2, \dots, K\}$ on the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$, where

$$S_k^o = \cup_{c \in S_k^a} S_c, \tag{25}$$

for $k = 1, 2, \dots, K$. Let $\hat{Q}(\mathcal{P}^a)$ be the modularity of the partition $\mathcal{P}^a$ of the $\mathcal{P}$-aggregated sampled graph and $Q(\mathcal{P}^o)$ be the modularity of the induced partition $\mathcal{P}^o$ of the sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. Then

$$\hat{Q}(\mathcal{P}^a) = Q(\mathcal{P}^o). \tag{26}$$

Note from (24) that the number of nonzero entries in the $C \times C$ matrix $\hat{P} = (\hat{p}(\cdot,\cdot))$ is not larger than the number of nonzero entries in the $n \times n$ matrix $P = (p(\cdot,\cdot))$. Thus, such a transformation is also sparsity preserving.

IV. Community detection

In this section, we extend three commonly used modularity maximization algorithms for community detection in *undirected* networks to *directed* networks: (i) the hierarchical agglomerative algorithm [10], (ii) the partitional algorithm

[11], and (iii) the fast unfolding algorithm [12]. The keys are the modularity preserving transformations in Lemma 9 and Lemma 11. As $\tilde{Q}(\mathcal{P}) = Q(\mathcal{P})$ in Lemma 9, one can then use the community detection algorithms for the sampled graph $(G(V_g, E_g), \tilde{p}(\cdot,\cdot))$ with the symmetric bivariate distribution to solve the community detection problem for the original sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$. For this, we define the correlation measure between two nodes v and w as follows:

$$\begin{aligned} q(v,w) &= \tilde{p}(v,w) - \tilde{p}_V(v)\tilde{p}_W(w) \\ &= \frac{p(v,w) - p_V(v)p_W(w) + p(w,v) - p_V(w)p_W(v)}{2}. \end{aligned} \tag{27}$$

For any two sets S_1 and S_2, define the correlation measure between these two sets as

$$q(S_1, S_2) = \sum_{v \in S_1} \sum_{w \in S_2} q(v,w). \tag{28}$$

With this correlation measure, we have from Lemma 9, (21) and (28) that the modularity for the partition $\mathcal{P} = \{S_c, c = 1,2,\dots,C\}$ is

$$Q(\mathcal{P}) = \tilde{Q}(\mathcal{P}) = \sum_{c=1}^{C} q(S_c, S_c), \tag{29}$$

Moreover, a set S is a community if and only if $q(S,S) \geq 0$.

ALGORITHM 1: The Hierarchical Agglomerative Algorithm

```
Input: A sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a
       bivariate distribution $p(\cdot,\cdot)$ that has the same
       marginal distributions in (4).
Output: A partition $\mathcal{P} = \{S_c, c = 1,2,\dots,C\}$ for some
        $C \leq n$.
(H1) Initially, $C = n$; $S_i = \{i\}$, $i = 1,2,\dots,n$;
(H2) Compute the correlation measures
  $q(S_i, S_j) = q(\{i\},\{j\})$ from (27) for all
  $i, j = 1,2,\dots,n$;
while there exists some $i$ and $j$ such that $q(S_i, S_j) > 0$
 do
    (H3) Merge $S_i$ and $S_j$ into a new set $S_k$, i.e.,
      $S_k = S_i \cup S_j$;

      $q(S_k, S_k) = q(S_i, S_i) + 2q(S_i, S_j) + q(S_j, S_j)$;   (30)
    for each $\ell \neq k$ do
        $q(S_k, S_\ell) = q(S_\ell, S_k) = q(S_i, S_\ell) + q(S_j, S_\ell)$;
                                                                 (31)
    end
    $C = C - 1$;
end
Reindex the $C$ remaining sets to $\{S_1, S_2, \dots, S_C\}$;
```

A. A hierarchical agglomerative algorithm

Analogous to the hierarchical agglomerative algorithms in [10], [12], we first propose a hierarchical agglomerative algorithm for community detection in directed networks.

The hierarchical agglomerative algorithm in Algorithm 1 has the following properties.

Theorem 12:

(i) For the hierarchical agglomerative algorithm in Algorithm 1, the modularity is non-decreasing in every iteration and thus converges to a local optimum.

(ii) When the algorithm converges, every set returned by the hierarchical agglomerative algorithm is indeed a *community*.

The proof of Theorem 12 is given in Appendix E of the online-only supplement. For (i) and (ii) of Theorem 12, it is not necessary to specify how we select a pair of two sets with a nonnegative correlation. In our experiments in Section VI, we will use the greedy selection that selects the two sets with the *largest* correlation measure to merge in (H3) of Algorithm 1. Such a selection results in the largest increase of the modularity in each merge. Even though the hierarchical agglomerative algorithm in Algorithm 1 produces communities, there are still two drawbacks:

(i) The hierarchical agglomerative algorithm only uses the "merge" operation to increase the modularity. As such, once a "merge" operation is done, there is no way to reverse it. As such, the order of the "merge" operations might have a serious effect on the final outcome.

(ii) In the initialization stage, every node is assumed to be a community itself. There are roughly $O(n)$ communities at the early stage of the hierarchical agglomerative algorithm. Thus, its computational complexity is high at the early stage of the algorithm. For a graph with n nodes, it is well-known (see e.g., [2]) that the computational complexity of a näive implementation is $O(n^3)$ [23] for a greedy hierarchical agglomerative algorithm and it can be reduced to $O(n^2 \log n)$ (or $O(m \log n)$ in a sparse graph) by implementing priority queues [24].

B. A partitional algorithm

As mentioned in the previous section, there are two drawbacks of the hierarchical agglomerative algorithm in Algorithm 1. To tackle these problems, we propose a partitional algorithm in Algorithm 2 that has a much lower computational complexity. Like the K-means algorithm and the K-sets algorithm in [11], our partitional algorithm also allows nodes to move from communities to communities.

We first consider another correlation measure $q_0(\cdot,\cdot)$ from the original correlation measure $q(\cdot,\cdot)$ in (27) by letting $q_0(v,w) = q(v,w)$ for $v \neq w$ and $q_0(v,w) = 0$ for $v = w$. Also let

$$q_0(S_1, S_2) = \sum_{v \in S_1} \sum_{w \in S_2} q_0(v,w). \tag{32}$$

In view of (29), we have for any partition $\mathcal{P} = \{S_c, c = 1,2,\dots,C\}$ that

$$Q(\mathcal{P}) = \sum_{c=1}^{C} q(S_c, S_c) = \sum_{c=1}^{C} q_0(S_c, S_c) + \sum_{v=1}^{n} q(v,v). \tag{33}$$

ALGORITHM 2: The Partitional Algorithm

```
Input: A sampled graph $(G(V_g, E_g), p(\cdot,\cdot))$ with a
       bivariate distribution $p(\cdot,\cdot)$ that has the same
       marginal distributions in (4), and the (maximum)
       number of communities $K$.
Output: A partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ for some
        $C \le K$.
(P0) Initially, choose arbitrarily $K$ disjoint nonempty sets
  $S_1, \ldots, S_K$ as a partition of the sampled graph
  $(G(V_g, E_g), p(\cdot,\cdot))$.
(P1) for $v = 1, 2, \ldots, n$ do
    for each neighboring set $S_j$ of node $v$ do
        Compute the correlation measures $q_0(v, S_j)$.
    end
    Find the neighboring set to which node $v$ has the
      largest correlation measure. Assign node $v$ to that
      set.
end
(P2) Repeat from (P1) until there is no further change.
```

Thus, maximizing the modularity is equivalent to maximizing $\sum_{c=1}^{C} q_0(S_c, S_c)$.

To describe the partitional algorithm in Algorithm 2, we let

$$\text{Nei}(v) = \{w : \tilde{p}(v,w) > 0\} \tag{34}$$

be the set that contains the "neighboring" nodes of v, where $\tilde{p}(v,w)$ is the symmetric bivariate distribution defined in (22). A set S is called a neighboring set of a node v if there exists a node $w \in S$ such that $\tilde{p}(v,w) > 0$. In each iteration, every node is assigned to one of its neighboring set to which it has the largest correlation measure. It might happen (in fact very often) that some sets of the initial partition become empty when all their set members are moved to other sets. Intuitively, the "merge" operation that is used in the hierarchical agglomerative algorithm is done in a microscopic manner in the partitional algorithm. As such, the chance for the partitional algorithm to be trapped in a bad "merge" operation is smaller than that for the hierarchical agglomerative algorithm.

The partitional algorithm has the following property and its proof is given in Appendix F of the online-only supplement.

Theorem 13: In the partitional algorithm in Algorithm 2, the modularity is non-decreasing when there is a change, i.e., a node is moved from one set to another. Thus, the algorithm converges to a local optimum of the modularity in a finite number of steps.

Now we turn our attention to the computation complexity of the partitional algorithm in Algorithm 2. Let m be the total number of nonzero entries in the $n \times n$ matrix $\tilde{P} = (\tilde{p}(v,w))$. Even though the total number of nonzero entries in the modularity matrix Q is $O(n^2)$, we will show that the computation complexity of the partitional algorithm is only $O((n+m)I)$, where I is the number of iterations in Algorithm 2. This is done with the memory complexity $O(n+m)$. Thus, Algorithm 2 is a linear time algorithm for a sparse graph with $m = O(n)$ and it is much more efficient than the hierarchical agglomerative algorithm for large sparse graphs.

Let

$$\begin{aligned}\tilde{p}(S_1, S_2) &= \sum_{v \in S_1} \sum_{w \in S_2} \tilde{p}(v,v), \text{ and} &(35)\\ \tilde{p}_V(S) &= \sum_{v \in S} \tilde{p}_V(v). &(36)\end{aligned}$$

To store the nonzero entries of the $n \times n$ matrix $\tilde{P} = (\tilde{p}(v,w))$ requires $O(m)$ memory complexity. In addition to this, we also store $q(v,v)$, $\tilde{p}_V(v)$ for all $v = 1, 2, \ldots, n$, and $\tilde{p}_V(S_k)$ for all $k = 1, 2, \ldots, K$. Thus, the memory complexity is $O(n+m)$. Now suppose node v is moved from S_1 to S_2. As $\tilde{p}_V(v)$ for all $v = 1, 2, \ldots, n$, are all stored in the memory, one can perform the following updates:

$$\begin{aligned}\tilde{p}_V(S_1) &\Leftarrow \tilde{p}_V(S_1) - \tilde{p}_V(v), \text{ and}\\ \tilde{p}_V(S_2) &\Leftarrow \tilde{p}_V(S_2) + \tilde{p}_V(v).\end{aligned}$$

These updates can be done in $O(1)$ steps. Thus, the total number of updates in each iteration of Algorithm 2 requires $O(n)$ steps.

Now we argue that the computational complexity for the correlation measures in each iteration is $O(n+m)$. Note that

$$\begin{aligned}q(v, S_k) &= \sum_{w \in S_k} q(v,w)\\ &= \tilde{p}(v, S_k) - \tilde{p}_V(v)\tilde{p}_V(S_k)\end{aligned}$$

and that

$$\begin{aligned}\tilde{p}(v, S_k) &= \sum_{w \in S_K} \tilde{p}(v,w)\\ &= \sum_{w \in S_k \cap \text{Nei}(v)} \tilde{p}(v,w).\end{aligned}$$

As now $\tilde{p}_V(v)$ and $\tilde{p}_V(S_k)$ are stored in the memory and there are $|\text{Nei}(v)|$ "neighboring" nodes of v, one can compute the correlation measures $q(v, S_k)$ for all the neighboring sets S_k of node v in $|\text{Nei}(v)|$ steps. The correlation measure $q_0(v, S_k)$ can be computed by using $q(v, S_k)$ for $v \notin S_k$ and $q(v, S_k) - q(v,v)$ for $v \in S_k$. Thus, the computational complexity for the correlation measures for the n nodes in each iteration is $O(n+m)$.

One possible initial partition for the partitional algorithm is to choose $K = n$ and every node is in a different set. Such a choice was in fact previously used in the fast unfolding algorithm in [12]. The fast unfolding algorithm is also a modularity maximization algorithm and it consists of two-phases: the first phase is also a partitional algorithm like the one presented here. The difference is that our partitional algorithm uses the probabilistic framework. Such a framework not only provides the needed physical insights but also allows a much more general treatment for modularity maximization. Once the first phase is completed, there is a second phase of building a new (and much smaller) network whose nodes are the communities found during the previous phase. This will be addressed in the next section.

ALGORITHM 3: The fast unfolding (Louvain) algorithm

```
Input: A sampled graph $(G(V_g, E_g), p(\cdot, \cdot))$ with a
       bivariate distribution $p(\cdot, \cdot)$ that has the same
       marginal distributions in (4).
Output: A partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ for some
       $C \leq n$.
(F0) Initially, choose the partition $\mathcal{P}$ with $C = n$ and
    $S_i = \{i\}$, $i = 1, 2, \ldots, n$;
(F1) Run the partitional algorithm in Algorithm 2 with
    the initial partition $\mathcal{P}$. Let $\mathcal{P}' = \{S_1, S_2, \ldots, S_C\}$ be its
    output partition.
(F2) If $\mathcal{P} = \mathcal{P}'$, then there is no improvement and return
    $\mathcal{P}$ as the output partition.
(F3) Otherwise, do node aggregation to generate the
    $\mathcal{P}'$-aggregated sampled graph of the sampled graph
    $(G(V_g, E_g), p(\cdot, \cdot))$. Run the fast unfolding algorithm in
    Algorithm 3 with the $\mathcal{P}'$-aggregated sampled graph as
    its input. Suppose it outputs the partition
    $\mathcal{P}^a = \{S^a_k, k = 1, 2, \ldots, K\}$ on the $\mathcal{P}'$-aggregated
    sampled graph. Return the induced partition
    $\mathcal{P}^o = \{S^o_k, k = 1, 2, \ldots, K\}$ on the sampled graph
    $(G(V_g, E_g), p(\cdot, \cdot))$, where $S^o_k = \cup_{c \in S^a_k} S_c$.
```

C. A fast unfolding algorithm

The outputs of the partitional algorithm may not be *communities* when it converges. For this, we consider another phase that uses node aggregation as in the fast unfolding algorithm in [12]. As shown in Lemma 11, node aggregation is a modularity preserving and sparsity preserving transformation. The modularity preserving property in Lemma 11 allows us to further increase the modularity by recursively applying the partitional algorithm in Algorithm 2 and the node aggregation in Definition 10. This is outlined in the (recursive) fast unfolding algorithm in Algorithm 3.

Since the transformation by node aggregation is also sparsity preserving, the computational complexity of each recursive call of Algorithm 3 is $O(m+n)$ (as in the partitional algorithm). Thus, the overall computational complexity depends on the depth of the recursive calls. A rigourous proof for the upper bound on the depth of the recursive calls appears to be difficult. The original fast unfolding algorithm is a special case of ours (by using the uniform edge sampling) and it is generally conjectured to be $O(\log m)$ for the depth of the recursive calls. But we do not have a formal proof for that. However, in our experiments, the depth is quite small.

We have the following properties for Algorithm 3. Its proof is given in Appendix G of the online-only supplement.

Theorem 14:

(i) In the fast unfolding algorithm in Algorithm 3, the modularity is non-decreasing when there is a change of the partition. Thus, the algorithm converges to a local optimum of the modularity in a finite number of steps.

(ii) When the algorithm converges, every set returned by Algorithm 3 is indeed a *community*.

The intuition behind the fast unfolding algorithm is to increase the modularity by moving a single node one at a time at the first level and then moving a block of nodes in each recursive level. As it gets deeper, the block size is larger. The fast unfolding algorithm stops at its deepest level when no more blocks of nodes can be moved to increase the modularity. In view of this, it is not necessary to start the fast unfolding algorithm from the initial partition that contains n sets with each node in a set. The modularity can be further increased by running the fast unfolding algorithm again with its first output partition as its input partition. However, the gain from doing another iteration of the fast unfolding algorithm is in general quite limited. Also, as pointed out in [12], two tricks to reduce the computational cost are (i) to stop the partitional algorithm when the gain of modularity is below a certain threshold, and (ii) removing the nodes with degree 1 and then adding them back later.

In comparison with the original fast unfolding algorithm in [12], our fast unfolding algorithm has the following advantages:

(i) Generalization: based on our probabilistic framework, our fast unfolding algorithm provides users the freedom to choose the sampled graph that might be more suitable for their viewpoints and resolution requirements. By doing so, our fast unfolding algorithm is also applicable for directed networks.

(ii) Theoretical guarantees: our framework provides a rigorous proof for the convergence of the fast unfolding algorithm through modularity maximization and node aggregation. Moreover, when the algorithm converges, it returns *communities* that are formally defined in our framework.

(iii) Probabilistic insights: our probabilistic framework defines the concept of *correlation*. The partitional algorithm in the first phase of the fast unfolding algorithm is simply to *repeatedly* move each node to the most positively correlated set. Also, the variables that need to be stored in the memory have clear probabilistic meanings that can then be used for speeding up the computation of correlation. More importantly, the node aggregation phase can be understood as a modularity preserving and sparsity preserving transformation.

(iv) Reduction of computational cost: by setting the diagonal elements of the modularity matrix to be 0, the computation for the largest increase of modularity, ΔQ in [12], is now reduced to the computation of the largest correlation of a node to its neighboring sets.

D. Weak communities and outliers

If the output partition of the fast unfolding algorithm is satisfactory, then we can simply stop there. If otherwise, we can perform the following post-processing to find another and possibly better initial partition. The idea of our post-processing is to examine the contribution of each community to the modularity and reassign the nodes in the communities with small contributions. Specifically, suppose that the fast unfolding algorithm returns a partition of C communities $S_1, S_2, \ldots, S_C$. The contribution of S_c to the modularity is then $q(S_c, S_c)$, where $q(\cdot, \cdot)$ is defined in (28). Then we can sort $q(S_c, S_c)$, $c = 1, 2, \ldots, C$ in the ascending order and perform a "sweep" to cut these C communities (at the maximum gap) into *strong* communities and *weak* communities. Suppose that the strong communities are $S'_1, \ldots, S'_K$ with $K < C$. The nodes in the weak communities are then *iteratively* reassigned to the most *positively correlated* strong community, i.e., node i is assigned to S'_j if $q(i, S'_j) = \max_{1 \le \ell \le K} q(i, S'_\ell)$ and $q(i, S'_j) > 0$. After this is done, we have K communities, $S''_1, \ldots, S''_K$ with $S'_\ell \subset S''_\ell$, $\ell = 1, 2, \ldots, K$. There might be still some nodes that are *negatively correlated* to S''_ℓ for all $\ell = 1, 2, \ldots, K$. The set of nodes $\{i : q(i, S''_\ell) \le 0, \ell = 1, 2, \ldots, K\}$ can be classified as *outliers*. For outliers, we can either remove them from the graph or reassign them to the most correlated strong community. In either case, we have a new initial partition that we can use for another run of the fast unfolding algorithm.

V. General bivariate distribution

In this section, we discuss how to extend our framework of sampled graphs to the setting with general bivariate distributions. As the marginal distributions may not be the same, many results derived in this paper are not as elegant as before. For instance, now we have to distinguish the notion of *centrality* into two types: the out-centrality (with respect to the marginal distribution of V) and the in-centrality (with respect to the marginal distribution of W). The notion of *relative centrality* also needs to be extended in the same manner. However, even with such an extension, the reciprocity property in Proposition 5(iv) is not true any more.

The good news is that the notions of *community* and *modularity* can be extended in the same manner as before. There is no need to distinguish an in-community and an out-community. Specifically, we can still define a set S as a community if

$$\mathsf{P}(V \in S, W \in S) - \mathsf{P}(V \in S)\mathsf{P}(W \in S) \ge 0,$$

and define the modularity $Q(\mathcal{P})$ with respect to the partition S_c, $c = 1, 2, \ldots, C$, by using (21). However, the equivalent characterizations for a community in Theorem 7 are no long valid as the reciprocity property in Proposition 5(iv) is not true any more.

With the general definition of the modularity, one can still apply the hierarchical agglomerative algorithm in Algorithm 1, the partitional algorithm in Algorithm 2, and the fast unfolding algorithm in Algorithm in 3 for community detection in a sampled graph with a general bivariate distribution. Moreover, the results in Theorem 12, Theorem 13, and Theorem 14 also hold. However, there are two minor modifications:

(i) The modularity preserving transformation of sampled graphs in Lemma 9 cannot be directly applied. As such, one needs to skip the transformation and directly apply the definition of the correlation measure between two nodes v and w, i.e., $q(v, w)$, in (27) to construct the correlation matrix. Also, for any two sets S_1 and S_2, the correlation measure between these two sets, i.e., $q(S_1, S_2)$, is still defined as in (28). By doing so, a set S is community if and only if $q(S, S) \ge 0$ and the modularity for a partition $\mathcal{P} = \{S_c, c = 1, 2, \ldots, C\}$ can still be computed by $\sum_{c=1}^{C} q(S_c, S_c)$ as described in (29).

(ii) The definition of "neighboring" nodes of a node v in (34) should be generalized as follows:

$$\text{Nei}(v) = \{w : p(v, w) > 0 \text{ or } p(w, v) > 0\}. \tag{37}$$

Also, a set S is called a "neighboring" set of a node v if there exists a node $w \in S$ such that w is a "neighboring" node of v. In order to show that the partitional algorithm is still a linear-time algorithm, one needs to store in memory the nonzero entries of the $n \times n$ matrix $P = (p(v, w))$, $q(v, v)$, $p_V(v)$, $p_W(v)$ for all $v = 1, 2, \ldots, n$, and $p_V(S_k)$, $p_W(S_k)$ for all $k = 1, 2, \ldots, K$. Then use those to compute the correlation measure

$$\begin{aligned} &q(v, S_k) \\ &= \frac{p(v, S_k) + p(S_k, v) - p_V(v)p_W(S_k) - p_W(v)p_V(S_k)}{2}. \end{aligned}$$

As pointed out by one of the reviewers, there are two different types of communities for directed networks that are commonly addressed in the literature (see e.g., [25], [26]): structural communities (based on the densities of edges) and flow communities (based on the flow of probabilities). These two types of communities correspond to two different types of sampled graphs in our probabilistic framework. For flow communities, they can be detected by using random walks as a random walker tends to be "trapped" in a community with a high probability. On the other hand, structural communities can be detected by using uniform edge sampling in [16] to produce communities that are densely connected inside and sparsely connected outside. A very enlightening example was shown in Fig. 4 of [26]. We note that the computation complexity of constructing the desired bivariate distribution to detect a certain type of communities might vary. For instance, the computation complexity of finding $p(v, w)$ by using uniform edge sampling in [16] is $O(1)$. On the other hand, the computation complexity of finding $p(v, w)$ by using a random walk requires solving the steady state probabilities of a Markov chain, which is in general $O(n^3)$.

VI. Experimental results

A. The hierarchical agglomerative algorithm

In this section, we use the hierarchical agglomerative algorithm to compare the sampling methods by PageRank in Section II-A and random walks with self-loops and backward jumps in Section II-B. We conduct various experiments based on the stochastic block model with two blocks. The stochastic block model, as a generalization of the Erdos-Renyi random graph, is a commonly used method for generating random graphs that can be used for benchmarking community detection algorithms. In a stochastic block model with two blocks (communities), the total number of nodes in the random graph are evenly distributed to these two blocks. The probability that there is a directed edge between two nodes within the same block is p_{in} and the probability that there is a directed edge between two nodes in two different blocks is p_{out}. These edges are generated independently. Let $c_{in} = np_{in}$ and $c_{out} = np_{out}$.

In our experiments, the number of nodes n in the stochastic block model is 2000 with 1000 nodes in each of these two blocks. The average degree (the sum of the average in-degree and the average out-degree) of a node is set to be 3. The values of $c_{in} - c_{out}$ of these graphs are in the range from 2.5 to 5.9 with a common step of 0.1. We generate 20 graphs for each $c_{in} - c_{out}$. Isolated vertices are removed. Thus, the exact numbers of vertices used in this experiment might be slightly less than 2000. For PageRank, the parameter λ is chosen to be 0.9. For the random walk with self-loops and backward jumps, the three parameters are $\lambda_0 = 0.05$, $\lambda_1 = 0.75$ and $\lambda_2 = 0.2$. We run the greedy hierarchical agglomerative algorithm in Algorithm 1 until there are only two sets (even when there do not exist nonnegative correlation measures between two distinct sets). We then evaluate the overlap with the true labeling. In Figure 2, we show the experimental results, where each point is averaged over 20 random graphs from the stochastic block model. The error bars are the 95% confidence intervals. From Figure 2, one can see that the performance of random walks with self-loops and backward jumps is better than that of PageRank. One reason for this is that PageRank uniformly adds an edge (with a small weight) between any two nodes and these added edges change the network topology. On the other hand, mapping by a random walk with backward jumps in (13) does not change the network topology when it is viewed as an undirected network.

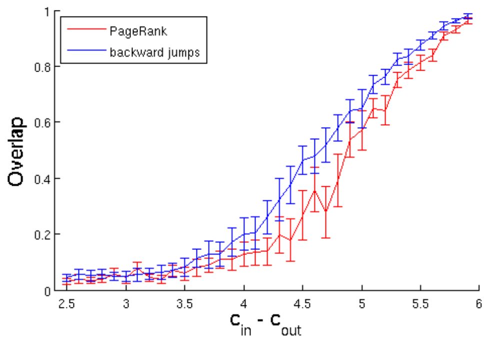


Fig. 2. Community detection of the stochastic block model by using PageRank in (12) and a random walk with self-loops and backward jumps in (13).

We also test the hierarchical agglomerative algorithm by using a real world dataset [27], [28]. The directed network in [27], [28] is a neural network that consists of 297 nodes and 2345 directed edges. In Figure 3(a), we show the community detection result for the neural network by using PageRank in (12). The parameter λ for PageRank is chosen to be 0.9. There are five communities from the hierarchical agglomerative algorithm by using PageRank. These five communities have the community strength 0.3379, 0.4603, 0.4202, 0.0934, and 0.3539. The modularity for such a partition is 0.3025. In Figure 3(b), we show the community detection result for the neural network by a random walk with backward jumps in (13). On the other hand, the three parameters for a random walk with backward jumps are $\lambda_0 = 0$, $\lambda_1 = 0.9$ and $\lambda_2 = 0.1$. There are two communities with the community strength 0.3763 and 0.4457. The modularity for such a partition is 0.4086. Once again, the mapping by a random walk with backward jumps in (13) performs better than the mapping by using PageRank in (12) as there is a rather weak community (with community strength 0.0934) by using PageRank.

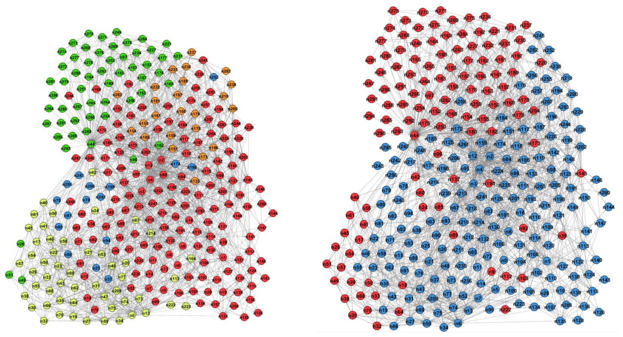

(a) PageRank (b) random walk with backward jumps

Fig. 3. The community detection result for the neural network.

B. The fast unfolding algorithm

In this section, we evaluate the fast unfolding algorithm by using two different benchmark graphs: (i) the six cluster model, (ii) the LFR model [29], and (iii) the SNAP collection [37].

1) The six cluster model: We generate a graph with six communities on a plane. We consider six unit circle centered at $(2, 0)$, $(4, 0)$, $(6, 0)$, $(2, 2)$, $(4, 2)$, and $(6, 2)$, respectively. For the first (resp. last) three circles, we generate 1000 (resp. 800) random nodes. A node within a circle is generated by first selecting its distance to its center uniformly in $[0, 1]$ and then its angle uniformly in $[0, 2\pi]$. Clearly, there are six "ground truth clusters" of these 5400 nodes as shown in Figure 4(a). Now we construct a directed (neighborhood) graph by connecting two nodes with a directed edge if their distance is not greater than a constant ϵ. The direction of the edge is chosen with an equal probability of 0.5.

In our first experiment, we choose $\epsilon = 0.5$. We use the random walk with backward jumps for generating the sampled graph. The three parameters are $\lambda_0 = 0$, $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$. In view of (13), such a sampled graph has the following *symmetric* bivariate distribution

$$p(u, v) = \frac{a_{u,v} + a_{v,u}}{2m}, \tag{38}$$

where $A = (a_{u,v})$ is the adjacency matrix of the directed graph and m is the total number of edges. This is also equivalent to the edge sampling [4] of an undirected graph with the adjacency matrix $B = A + A^T$ (with A^T being the transpose of the matrix A) that chooses the two end nodes of a uniformly selected edge. The modularity for edge sampling is thus the same as the Newman's modularity [6]. The modularity of the partition with the six "ground truth clusters" for such a graph is 0.8105. We run the fast unfolding algorithm in Algorithm 3 for such a graph and it outputs a partition of 6 communities with the modularity 0.8106 (see Figure 4(b)). The fast unfolding algorithm in facts ends when the partitional algorithm in Algorithm 2 ends after 6 iterations. In other words, node aggregation is not used for this experiment of the fast unfolding algorithm.

In our second experiment, we construct another graph that is more difficult to detect communities. This is done by choosing $\epsilon = 0.25$ so that the graph is much more sparse than the one with $\epsilon = 0.5$. Once again, we use the sampling in (38) for this new graph. The modularity of the partition with the six "ground truth clusters" for such a graph is 0.8197. We run the fast unfolding algorithm in Algorithm 3 for such a graph and it outputs a partition of 11 communities with the modularity 0.8246 (see Figure 4(c)), which is much larger than the modularity of the ground truth partition 0.8197. As can be seen from Figure 4(a) and (c), the fast unfolding algorithm outputs a partition (with a much larger modularity value) that is not even close to the ground truth partition. In view of this, one should carefully examine the output partition of a modularity maximization algorithm.

For this new graph with $\epsilon = 0.25$, the fast unfolding algorithm needs 5 levels of node aggregations and the partitional algorithm in the first level has 20 iterations, which is significantly larger than the graph with $\epsilon = 0.5$. By examining the 11 communities (see Table I), we find that there are 5 communities (communities 3,4,6,7 and 9) that have relatively weak contributions to the modularity. This does not mean the community strengths of these 5 communities are small. In fact, their community strengths are comparable to the other 6 communities (communities 1,2,5,8,10 and 11). However, their sizes are relatively smaller. After the process of eliminating communities with weak contributions to the modularity in Section IV-D, we have a partition of six clusters with the modularity 0.8175 (see Figure 4(d)). We then use that partition as the initial partition for another run of the fast unfolding algorithm and it outputs a partition of six clusters with the modularity 0.8200 (see Figure 4(e)). One lesson learned from this experiment is that the fast unfolding algorithm (based on modularity maximization) might produce small communities with relatively strong community strengths if the total number of communities is not known in advance. These small communities are also negatively correlated to the large communities and thus cannot be merged into larger communities. In view of this, post processing that carefully examines the results produced by modularity maximization algorithms (such as the fast unfolding algorithm) is crucial to improving the quality of the community detection result.

Though the fast unfolding algorithm is a nearly-linear time algorithm for large sparse graphs with edge sampling, it suffers from the problem of resolution limit [30] due to its limited visibility to "see" more than one step (as illustrated in the previous experiment). To address the problem of resolution limit, *stability* was then proposed in [7], [8] based on continuous-time random walks. As shown in [4], the stability is also a special case of the modularity for a certain sampled graph. However, the sampled graph that corresponds to the stability in [7], [8] is no longer sparse and thus the computational cost of the fast unfolding algorithm is increased dramatically. One reasonable compromise might be to consider the following sampling method with paths of length two:

$$p(u,v) = \frac{\breve{a}_{u,v}}{2\breve{m}}, \tag{39}$$

where

$$\breve{A} = (\breve{a}_{u,v}) = (A + A^T) + 0.5(A + A^T)^2$$

and

$$\breve{m} = \sum_u \sum_v \breve{a}(u,v)/2.$$

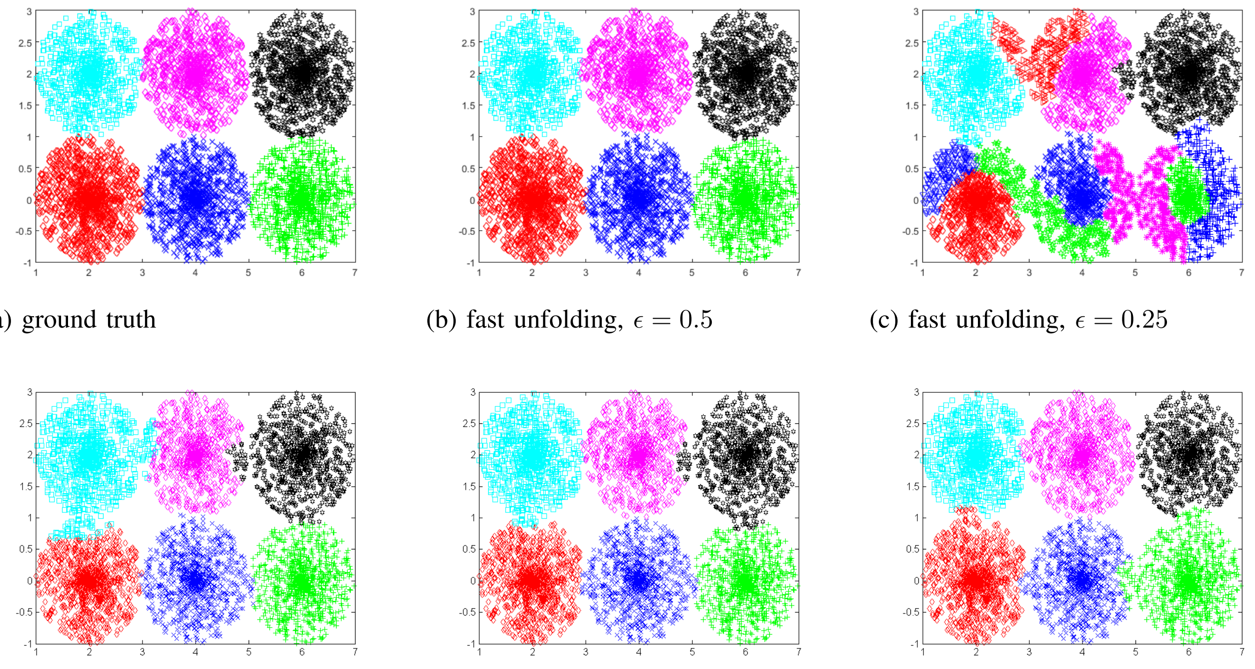
In Figure 4(f), we show the partition generated by the fast unfolding algorithm for the sampled graph in (39). Since the matrix A is the adjacency matrix of the graph with $\epsilon = 0.25$, the matrix $\breve{A}$ is capable of "seeing" nodes within the distance 0.5. As shown in Figure 4(a) and (f), its performance is comparable to that by the fast unfolding algorithm for the more dense graph with $\epsilon = 0.5$.

Our observations from the experiments for the six "ground truth" clusters also appear in our experiment for the real dataset from the political blogs in [31]. For such a directed network, we first remove the nodes with degree 0 and there are 793 nodes left. These nodes are in two ground truth communities with 351 nodes and 442 nodes respectively. We first use the random walk with backward jumps for generating the sampled graph. The three parameters are $\lambda_0 = 0$, $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$. The fast unfolding algorithm yields 11 communities. After the process of eliminating communities with weak contributions to the modularity in Section IV-D, we have two communities. These two communities are very close to the two ground truth communities and the overlap is 0.9672. We then use edge sampling for the directed network to generate another sampled graph. Specifically, we have the bivariate distribution (that may not have the same marginal distributions) $p(u,v) = a_{u,v}/m$, where $A = (a_{u,v})$ is the adjacency matrix of the directed network and m is the total number of directed edges in the network. Such a sampling method yields the same modularity in a direct network in [16]. This time the fast unfolding algorithm yields 10 communities. Again, after the process of eliminating communities with weak contributions, we have two communities that are exactly the same as that from the random walk with backward jumps. For these two sampled graphs, we need to go through the process of eliminating communities with weak contributions. For the third sampled graph, we use the sampling method with paths of length two in (39). The fast unfolding algorithm for this sampled graph yields two communities directly with the overlap 0.9697. This shows that the capability of the sampling method to "see" nodes more than one step away is very helpful in achieving the needed resolution for community detection, especially for sparse graphs.

2) The LFR model: In [12], the authors showed that the original fast unfolding algorithm is capable of dealing with large scale real networks, even when the number of edges is up to one billion. To provide convincing evidence that our fast unfolding algorithm can not only produce good community detection results but also has the same capability for dealing with large scale networks, we conduct two experiments by using the LFR benchmark graphs [29]. There are several

TABLE I
THE 11 COMMUNITIES BY THE FAST UNFOLDING ALGORITHM FOR A GRAPH WITH SIX CLUSTERS IN FIGURE 4(C).

community	1	2	3	4	5	6	7	8	9	10	11
contribution	0.1441	0.1397	0.0222	0.0229	0.1345	0.0225	0.0096	0.1116	0.0127	0.1044	0.1004
strength	0.8029	0.8007	0.7984	0.7908	0.7888	0.7961	0.8046	0.8662	0.8641	0.8631	0.8800
size	717	621	352	324	476	372	148	756	230	598	806



(a) ground truth (b) fast unfolding, $\epsilon = 0.5$ (c) fast unfolding, $\epsilon = 0.25$

(d) eliminating communities, $\epsilon = 0.25$ (e) 2nd fast unfolding, $\epsilon = 0.25$ (f) another sampled graph, $\epsilon = 0.25$

Fig. 4. The community detection result for a graph with six "ground truth clusters."

parameters that need to be specified in the LFR benchmark graphs. The node degrees and the (ground truth) community sizes in the LFR model are distributed according to the power law, with exponents γ and β, respectively. The symbol $\langle k \rangle$ denotes the average degree. The mixing parameter μ is used for characterizing how the built-in communities in a graph are mixed. Specifically, each node has a fraction $1-\mu$ (resp. μ) of its edges connected to the other nodes of its community (resp. the other nodes of the network). The direction of the edge is chosen with an equal probability of 0.5.

For both experiments, we use the bivariate distribution in (39). The objective of the first experiment is to show that our fast unfolding algorithm can produce good community detection results for various parameter settings. In the first experiment, the two exponents (γ, β) are chosen to be (2,1) and (3,2), respectively. The rest of the parameters, including the number of nodes, the maximum degree, $\langle k \rangle$, are set to be 100,000, 300, and 100, respectively. The values of mixing parameter μ of these graphs are in the range from 0.1 to 0.75 with a common step of 0.05. We generate 20 graphs for each μ. Then, we run our fast unfolding algorithm on each graph and compute the normalized mutual information measure (NMI) by using a built-in function in igraph [32]. The results are shown in Figure 5. The error bars are the 95% confidence intervals. As shown in Figure 5, our fast unfolding algorithm can produce good community detection results when the mixing parameter μ is small. Also, increasing the average degree $\langle k \rangle$ also increases the NMIs of the community detection results. In Figure 6, we show the number of the iterations of performing the partition algorithm at various recursive levels. As shown in Figure 6, the number of iterations increases when the mixing parameter μ increases.

In the second experiment, we show that our fast unfolding algorithm is capable of dealing with large scale networks. In this experiment, γ, β, the number of nodes, the maximum degree, the average degree, and the mixing parameter μ are set to be 2, 1, 10,000,000, 500,100, 0.5, respectively. As such, there are roughly half a billion edges in a graph with 10 million nodes. The number of ground-truth communities is 57187. As it takes more than 150 hours to generate such a large graph, we only generate a single LFR benchmark graph in this experiment. We use a random walk with self-loops to obtain the needed bivariate distribution for our probabilistic framework. In each node, the random walker will either stay at the same node or go along with one of its edges to an adjacent node, with probabilities λ and $1-\lambda$, respectively. Hence, the transition probability matrix of such a Markov chain can be characterized by $\lambda I + (1-\lambda)D^{-1}A$. The steady state probability for the random walker to visit node v is $\pi_v = k_v/2m$, where k_v is the degree of node v and m is the total number of edges in the network. In this experiment, we set $\lambda = 2999/3000$ so that the random walker has a very high probability to stay at the same node. We then compare our algorithm with the original fast unfolding algorithm that uses Newman's modularity [12]. The original fast unfolding algorithm, to the best of our knowledge, is one of the fastest community detection algorithms in the literature. Our algorithm is implemented in C++, while the original fast unfolding algorithm is obtained from built-in functions of igraph [32], which are implemented in C. We perform the experiment on an Acer Altos-T350-F2 machine with two Intel(R) Xeon(R) CPU E5-2690 v2 processors (only one single thread is used). We compare the performance of these two algorithms in Table II. Note that the I/O time is excluded from the running time in Table II. As shown in Table II, our algorithm achieves better performance than the original fast unfolding algorithm, both in accuracy (in terms of NMI) and the running time. To dig further, we observe that the original fast unfolding algorithm (that aims to maximize Newman's modularity) fails to detect small communities because it keeps recursively merging communities until there are only 3395 communities left. This phenomenon is known as the resolution limit previously reported in [30]. To see the intuition behind this, note that Newman's modularity in the original fast unfolding algorithm is equivalent to using the bivariate distribution $p_{vw} = A_{vw}/(2m)$ in our fast unfolding algorithm and this corresponds to the random walker without self-loops. Intuitively, increasing the self-loop probability of the random walker decreases its mobility and the mobility of the random walker can be used to control the size of detected communities. Decreasing (resp. Increasing) the mobility behaves like "zooming in" (resp. "zooming out") on the graph, and results in the detection of "local communities" (resp. "global communities"), which are small (resp. large) in size. Given this, our probabilistic framework has the flexibility to choose the resolution and might be more suitable for practical applications than other algorithms that have fixed resolutions.

3) The SNAP collection: In addition to synthetic networks, we use several well-known real-world networks from the Stanford Network Analysis Project Collection (SNAP) [37] to evaluate our algorithm. The collection contains six networks: Amazon, DBLP, Friendster, LiveJournal, Orkut, and Youtube. All the networks come with top 5000 ground-truth communities with the highest quality. Since all the ground-truth communities are *overlapping*, we need a preprocessing step to generate the ground-truth *disjoint* communities in these networks.

To do so, we follow the *maximum independent set (MIS)* method in [38] and use their code to generate the ground-truth *disjoint* communities. For each graph G, the MIS method considers each of the 5000 ground-truth community as a giant vertex, and assigns an edge between two giant vertices if and only if two communities share at least one vertex, i.e., they are two overlapping communities. Specifically, let C_i, $i = 1, 2, \ldots, 5000$, be the 5000 ground-truth communities in G. The MIS method constructs another graph $G'(V', E')$ with $V' = \{C_1, C_2 ... C_{5000}\}$ and $E' = \{(C_i, C_j) | C_i \cap C_j \neq \phi\}$. Then, it finds the maximum independent set of the graph G' and removes all the giant vertices that are not in the maximum independent set. This then leads to a graph with disjoint communities. We summarize the parameters of these six graphs by using the MIS method in Table III, including the number of vertices, the number of edges and the number of ground-truth communities. The setup of this experiment is the same as that in the second experiment of the LFR model in the previous section. Specifically, we use a random walk with self-loops to obtain the bivariate distribution for our probabilistic framework. In each node, the random walker will either stay at the same node or go along with one of its edges to an adjacent node, with probabilities λ and $1 - \lambda$, respectively. In this experiment, the values of $1-\lambda$ are selected from $2^{-\ell}$, $\ell = 0, 1, \ldots, 18$. The NMI results are shown in Figure 7, and the running time for each data point in Figure 7 is within two seconds. Note that for $\ell = 0$, we have $\lambda = 0$ and the random walk with self-loops degenerates to the original fast unfolding algorithm. As such, the first data point of each curve corresponds to that from the original fast unfolding algorithm. As shown in this figure, the NMI values of each curve form an inverted U-shaped curve that gradually increases to its maximum and then decreases. When the value of $1 - \lambda$ is lower than a particular threshold, the curve will suddenly level off. To explain this phenomenon, we manually examine the output communities. We find that the number of the communities increases when the value of $1 - \lambda$ decreases. This is because the random walker has a higher probability to stay at the same node for a smaller $1-\lambda$. If the value of $1-\lambda$ is below a particular threshold, the algorithm outputs each single node as a community itself. As such, each curve levels off when $1-\lambda$ is below that threshold. Moreover, when each curve reaches its maximum NMI, the output of the algorithm has roughly the same number of communities as that of the ground-truth communities. For the two networks with less than 1000 communities, i.e., Amazon and Orkut, the original fast unfolding performs well since both curves reach their maximums with relatively small values of λ. On the other hand, for the other four networks with more than 1000 communities, i.e., DBLP, Friendster, LiveJournal, and Youtube, the number of communities resulted from the original fast unfolding is smaller than the number of ground-truth communities. For these four networks, the fast unfolding algorithm based on a random walk with self-loops performs much better than the original fast unfolding algorithm. In view of this, when the number of detected communities is smaller than expected, one might consider using another "viewpoint" to detect the ground-truth communities in a real-world network (such as the random walk with self-loops in this experiment).

VII. Conclusion

In this paper, we extended our previous work in [3], [4] to *directed networks*. Our approach is to introduce bivariate distributions that have the same marginal distributions. By doing so, we were able to extend the notions of centrality, relative centrality, community strength, community and modularity to

TABLE II
PERFORMANCE COMPARISON BETWEEN THE ORIGINAL FAST UNFOLDING ALGORITHM AND THE FAST UNFOLDING ALGORITHM IN OUR PROBABILISTIC FRAMEWORK.

performance metric	NMI	running time (hours)
fast unfolding (Newman's modularity)	0.86	1.45
fast unfolding (probabilistic framework)	1	0.50

TABLE III
SUMMARY OF THE SIX NETWORKS AFTER THE PREPROCESSING STEP.

	Amazon	DBLP	Friendster	Livejournal	Orkut	Youtube
number of vertices	8258	23479	79803	38713	6722	10206
number of edges	22129	72116	992461	633195	73522	15959
number of communities	992	3021	3529	2520	375	2883

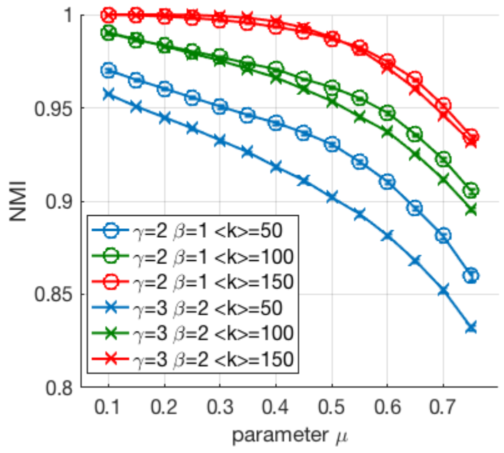


Fig. 5. The NMIs of our fast unfolding algorithm as a function of the mixing parameter μ.

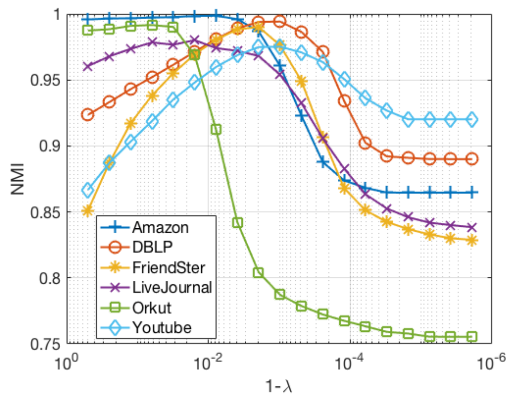


Fig. 7. The NMIs of the six networks from the SNAP collection. The values of $1 - \lambda$ are selected from $2^{-\ell}$, $\ell = 0, 1, \ldots, 18$.

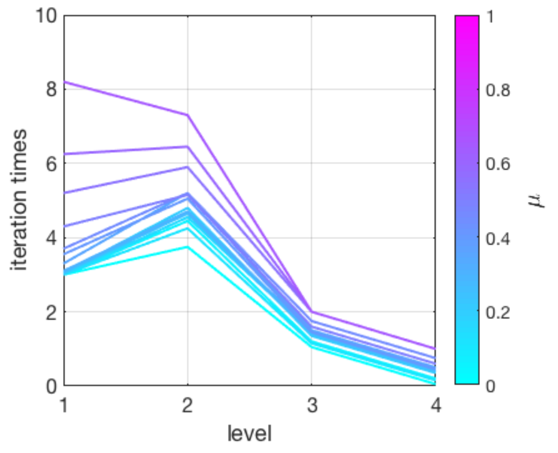


Fig. 6. The number of iterations in each level of recursive calls. The parameters γ, β, $\langle k \rangle$ are set to be 2, 1, 100, respectively.

directed networks. For community detection, we also extended three commonly used algorithms in *undirected* networks to *directed* networks: the hierarchical agglomerative algorithm, the partitional algorithm, and the fast unfolding algorithm. These three algorithms were shown to converge in a finite number of steps. Moreover, the outputs of the hierarchical agglomerative algorithm and the fast unfolding algorithm are guaranteed to be *communities*. We illustrated how these three algorithms can also be extended to the setting with general bivariate distributions. We also conducted various experiments by using two sampling methods in directed networks: (i) PageRank and (ii) random walks with self-loops and backward jumps. The experimental results showed that sampling by random walks with self-loops and backward jumps perform better than sampling by PageRank for community detection.

One of the most important factors that affect the performance of our probabilistic framework is the selection of the bivariate distribution (viewpoint). We also note that the prior domain knowledge is important for choosing the right bivariate distribution. As discussed in our second LFR experiment, one should choose a random walker with low mobility as there are many small ground-truth communities in the LFR dataset. On the other hand, in the experiment with six clusters, knowing the number of communities is very helpful for eliminating weak communities in the post-processing step and thus crucial to the success of community detection. Thus, the more we know about the application, the better we can detect communities for that application. There are various practical applications of community detection, e.g., identifying important places from cellular network data [33], routing in delay tolerant networks [34], and information spreading in social networks [35], [36]. Finding the right bivariate distribution for each of these applications definitely requires further investigation.

References

[1] C.-S. Chang, D.-S. Lee, L.-H. Liou, S.-M. Lu, and M.-H. Wu, "A probabilistic framework for structural analysis in directed networks," in *Proc. IEEE ICC*, May 2016.

[2] M. Newman, *Networks: an introduction*. OUP Oxford, 2009.

[3] C.-S. Chang, C.-Y. Hsu, J. Cheng, and D.-S. Lee, "A general probabilistic framework for detecting community structure in networks," in *Proc. IEEE INFOCOM*, 2011, pp. 730–738.

[4] C.-S. Chang, C.-J. Chang, W.-T. Hsieh, D.-S. Lee, L.-H. Liou, and W. Liao, "Relative centrality and local community detection," *Network Science*, vol. 3, no. 4, pp. 445–479, 2015.

[5] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proc. ACM CIKM*, 2003, pp. 556–559.

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

[7] R. Lambiotte, "Multi-scale modularity in complex networks," in *Proc. 8th Int. Symp. Model. Optimization Mobile, Ad Hoc Wireless Netw.*, IEEE, 2010, pp. 546–553.

[8] J.-C. Delvenne, S. N. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," *Proc. Nat. Acad. Sci.*, vol. 107, no. 29, pp. 12 755–12 760, 2010.

[9] 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.

[10] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, p. 066133, 2004.

[11] C.-S. Chang, W. Liao, Y.-S. Chen, and L.-H. Liou, "A mathematical theory for clustering in metric spaces," *IEEE Transactions on Network Science and Engineering*, vol. 3, no. 1, pp. 2–16, 2016.

[12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech.*, vol. 2008, no. 10, p. P10008, 2008.

[13] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *In Proc. WWW7 Conf. Computer Networks*, 1998.

[14] R. Lambiotte and M. Rosvall, "Ranking and clustering of nodes in networks with smart teleportation," *Phys. Rev. E*, vol. 85, no. 5, p. 056107, 2012.

[15] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Ann. Combinatorics*, vol. 9, no. 1, pp. 1–19, 2005.

[16] E. A. Leicht and M. E. Newman, "Community structure in directed networks," *Phys. Rev. Lett.*, vol. 100, no. 11, p. 118703, 2008.

[17] R. Nelson, *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer Verlag, 1995.

[18] Y. Kim, S.-W. Son, and H. Jeong, "Finding communities in directed networks," *Phys. Rev. E*, vol. 81, no. 1, p. 016103, 2010.

[19] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[20] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1979.

[21] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Proc. 47th IEEE FOCS*, 2006, pp. 475–486.

[22] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proc. ACM 19th WWW*, 2010, pp. 631–640.

[23] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, p. 066133, 2004.

[24] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, p. 066111, 2004.

[25] M. Rosvall and C. T. Bergstrom, "Maps of information flow reveal community structure in complex networks," *Proc. Nat. Acad. Sci.*, vol. 105, pp. 1118–1123, 2008.

[26] R. Lambiotte, J.-C. Delvenne, and M. Barahona, "Random walks, Markov processes and the multiscale modular organization of complex networks," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 2, pp. 76–90, 2014.

[27] J. White, E. Southgate, J. Thomson, and S. Brenner, "The structure of the nervous system of the nematode caenorhabditis elegans: the mind of a worm," *Phil. Trans. R. Soc. Lond*, vol. 314, pp. 1–340, 1986.

[28] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[29] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, p. 046110, 2008.

[30] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proc. Natl. Acad. Sci.*, vol. 104, no. 1, pp. 36–41, 2007.

[31] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proc. ACM Int. Workshop Link Discovery*, pp. 36–43.

[32] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: http://igraph.org

[33] S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky, "Identifying important places in peoples lives from cellular network data," in *Int. Conf. on Pervasive Computing*. Springer, 2011, pp. 133–151.

[34] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: a social network perspective," in *Proc. ACM MobiHoc*, 2009, pp. 299–308.

[35] N. P. Nguyen, T. N. Dinh, Y. Xuan, and M. T. Thai, "Adaptive algorithms for detecting community structure in dynamic social networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2282–2290.

[36] N. P. Nguyen, G. Yan, M. T. Thai, and S. Eidenbenz, "Containment of misinformation spread in online social networks," in *Proc. ACM WSC*, 2012, pp. 213–222.

[37] J. Yang, J. Leskovec. "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181-213, 2015.

[38] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, N. Samatov, "Community detection in largescale networks: a survey and empirical evaluation," *Wiley Interdiscip. Rev. Comput. Stat.*, vol. 6, no. 6, pp. 426-439, 2014.

Cheng-Shang Chang (S'85-M'86-M'89-SM'93-F'04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, USA, in 1986 and 1989, respectively, all in electrical engineering.

From 1989 to 1993, he was employed as a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Since 1993, he has been with the Department of Electrical Engineering, National Tsing Hua University, Taiwan, where he is a Tsing Hua Distinguished Chair Professor. He is the author of the book Performance Guarantees in Communication Networks (Springer, 2000) and the coauthor of the book Principles, Architectures and Mathematical Theory of High Performance Packet Switches (Ministry of Education, R.O.C., 2006). His current research interests are concerned with network science, big data analytics, mathematical modeling of the Internet, and high-speed switching.

Dr. Chang served as an Editor for Operations Research from 1992 to 1999, an Editor for the *IEEE/ACM TRANSACTIONS ON NETWORKING* from 2007 to 2009, and an Editor for the *IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING* from 2014 to 2017. He is currently serving as an Editor-at-Large for the *IEEE/ACM TRANSACTIONS ON NETWORKING*. He is a member of IFIP Working Group 7.3. He received an IBM Outstanding Innovation Award in 1992, an IBM Faculty Partnership Award in 2001, and Outstanding Research Awards from the National Science Council, Taiwan, in 1998, 2000, and 2002, respectively. He also received Outstanding Teaching Awards from both the College of EECS and the university itself in 2003. He was appointed as the first Y. Z. Hsu Scientific Chair Professor in 2002. He received the Merit NSC Research Fellow Award from the National Science Council, R.O.C. in 2011. He also received the Academic Award in 2011 and the National Chair Professorship in 2017 from the Ministry of Education, R.O.C. He is the recipient of the 2017 IEEE INFOCOM Achievement Award.

Duan-Shin Lee (S'89-M'90-SM'98) received the B.S. degree from National Tsing Hua University, Taiwan, in 1983, and the MS and Ph.D. degrees from Columbia University, New York, in 1987 and 1990, all in electrical engineering. He worked as a research staff member at the C&C Research Laboratory of NEC USA, Inc. in Princeton, New Jersey from 1990 to 1998. He joined the Department of Computer Science of National Tsing Hua University in Hsinchu, Taiwan, in 1998. Since August 2003, he has been a professor. He received a best paper award from the Y.Z. Hsu Foundation in 2006. His current research interests are social networks, network science, game theory and data science. He is a senior IEEE member.

Li-Heng Liou received his bachelor's degree in electrical engineering with a double major in computer science from the National Tsing-Hua University, Hsinchu, Taiwan, in 2013. He is currently pursuing the Ph.D. degree in the Institute of Communications Engineering, National Tsing-Hua University. His research interest is in efficient clustering algorithms and deep learning algorithms.

Sheng-Min Lu received the B.S. degree in Electrical Engineering and M.S. degree in Communications Engineering from the National Tsing-Hua University, Hsinchu, Taiwan, in 2014 and 2016, respectively. His research interest is community detection.

Mu-Huan Wu received his B.S. (2011) in electrical engineering and M.S. (2013) in communication engineering from the National Tsing-Hua University , Hsinchu, Taiwan. He has been employed by Airoha Technology Corp. since 2013. His work is mainly on the development of SoC applications.

Appendices

Appendix A

In this section, we prove Proposition 5. Since the relative centrality is a conditional probability and the centrality is a probability, the properties in (i),(ii) and (iii) follow trivially from the property of probability measures.

(iv) From (15) and (14), it follows that

$$\begin{aligned} &C(S)C(S^c|S) = \mathsf{P}(W \in S)\mathsf{P}(W \in S^c|V \in S) \\ &= \mathsf{P}(V \in S)\mathsf{P}(W \in S^c|V \in S) \\ &= \mathsf{P}(V \in S, W \in S^c). \end{aligned} \tag{40}$$

Similarly, we also have

$$C(S^c)C(S|S^c) = \mathsf{P}(V \in S^c, W \in S). \tag{41}$$

Thus, it suffices to show that

$$\mathsf{P}(V \in S, W \in S^c) = \mathsf{P}(V \in S^c, W \in S). \tag{42}$$

Note that

$$\mathsf{P}(V \in S) = \mathsf{P}(V \in S, W \in S) + \mathsf{P}(V \in S, W \in S^c), \tag{43}$$

and

$$\mathsf{P}(W \in S) = \mathsf{P}(V \in S, W \in S) + \mathsf{P}(V \in S^c, W \in S). \tag{44}$$

Since the bivariate distribution has the same marginal distributions, we have $\mathsf{P}(V \in S) = \mathsf{P}(W \in S)$. In conjunction with (43) and (44), we prove (42).

Appendix B

In this section, we prove Theorem 7. We first prove that the first four statements are equivalent by showing (i)$\Rightarrow$ (ii)$\Rightarrow$ (iii)$\Rightarrow$ (iv)$\Rightarrow$ (i).

(i) $\Rightarrow$ (ii): Note from Proposition 5 (i) and (ii) that $C(S|S)+C(S^c|S) = C(V_g|S) = 1$ and $C(S)+C(S^c) = C(V_g) = 1$. It then follows from the reciprocal property in Proposition 5(iv) that

$$\begin{aligned} &C(S^c)(C(S|S) - C(S|S^c)) \\ &= C(S^c)C(S|S) - C(S^c)C(S|S^c) \\ &= (1 - C(S))C(S|S) - C(S)C(S^c|S) \\ &= (1 - C(S))C(S|S) - C(S)(1 - C(S|S)) \\ &= C(S|S) - C(S) = Str(S) \ge 0. \end{aligned}$$

As we assume that $0 < C(S) < 1$, we also have $0 < C(S^c) < 1$. Thus,

$$C(S|S) - C(S|S^c) \ge 0.$$

(ii) $\Rightarrow$ (iii): Since we assume that $C(S|S) \ge C(S|S^c)$, we have from $C(S|S) + C(S^c|S) = C(V_g|S) = 1$ that

$$1 = C(S|S) + C(S^c|S) \ge C(S|S^c) + C(S^c|S).$$

Multiplying both sides by $C(S^c)$ yields

$$C(S^c) \ge C(S^c)C(S|S^c) + C(S^c)C(S^c|S).$$

From the reciprocal property in Proposition 5(iv) and $C(S) + C(S^c) = C(V_g) = 1$, it follows that

$$\begin{aligned} C(S^c) &\ge C(S)C(S^c|S) + C(S^c)C(S^c|S) \\ &= (C(S) + C(S^c))C(S^c|S) \\ &= C(S^c|S). \end{aligned}$$

(iii) $\Rightarrow$ (iv): Note from the reciprocal property in Proposition 5(iv) that

$$C(S)C(S^c|S) = C(S^c)C(S|S^c). \tag{45}$$

It then follows from $C(S^c|S) \le C(S^c)$ that $C(S|S^c) \le C(S)$.

(iv) $\Rightarrow$ (i): Since we assume that $C(S|S^c) \le C(S)$, it follows from (45) that $C(S^c|S) \le C(S^c)$. In conjunction with $C(S|S) + C(S^c|S) = C(V_g|S) = 1$ and $C(S) + C(S^c) = C(V_g) = 1$, we have

$$C(S|S) - C(S) = C(S^c) - C(S^c|S) \ge 0.$$

Now we show that and (iv) and (v) are equivalent. Since $C(S|S^c) + C(S^c|S^c) = C(V_g|S^c) = 1$ and $C(S) + C(S^c) = C(V_g) = 1$, we have

$$C(S|S^c) - C(S) = C(S^c) - C(S^c|S^c) = -Str(S^c).$$

Thus, $C(S|S^c) \le C(S)$ if and only if $Str(S^c) \ge 0$.

Replacing S by S^c, we see that (v) and (vi) are also equivalent because (i) and (ii) are equivalent.

Appendix C

In this section, we prove Lemma 9.

Since $p_V(v) = p_W(v)$ for all v, it then follows from (21) that

$$\begin{aligned} Q(\mathcal{P}) &= \sum_{c=1}^{C}\sum_{v\in S_c}\sum_{w\in S_c}(p(v,w) - p_V(v)p_V(w)) &(46) \\ &= \sum_{c=1}^{C}\sum_{v\in S_c}\sum_{w\in S_c}(p(w,v) - p_V(w)p_V(v)). &(47) \end{aligned}$$

Adding (46) and (47) yields

$$2Q(\mathcal{P}) = \sum_{c=1}^{C}\sum_{v\in S_c}\sum_{w\in S_c}(p(v,w) + p(w,v) - 2p_V(v)p_V(w)). \tag{48}$$

As $\tilde{p}(v,w) = (p(v,w) + p(w,v))/2$, we have

$$\tilde{p}_V(v) = \sum_{w\in V_g}\tilde{p}(v,w) = p_V(v).$$

Thus,

$$Q(\mathcal{P}) = \sum_{c=1}^{C}\sum_{v\in S_c}\sum_{w\in S_c}(\tilde{p}(v,w) - \tilde{p}_V(v)\tilde{p}_V(w)) = \tilde{Q}(\mathcal{P}).$$

Appendix D

In this section, we prove Lemma 11.

Note from (21) that

$$\hat{Q}(\mathcal{P}^a) = \sum_{k=1}^{K} \sum_{c_1 \in S_k} \sum_{c_2 \in S_k} (\hat{p}(c_1, c_2) - \hat{p}_V(c_1)\hat{p}_W(c_2)), \quad (49)$$

where

$$\hat{p}_V(c_1) = \hat{p}_W(c_1) = \sum_{c_2=1}^{C} \hat{p}(c_1, c_2) \quad (50)$$

is the marginal distribution of the bivariate distribution $\hat{p}(\cdot,\cdot)$. In view of (24) and (25), it is easy to verify that

$$\sum_{c_1 \in S_k} \sum_{c_2 \in S_k} \hat{p}(c_1, c_2) = \sum_{u \in S_k^o} \sum_{v \in S_k^o} p(u, v). \quad (51)$$

Similarly,

$$\begin{aligned} \sum_{c_1 \in S_k} \hat{p}_V(c_1) &= \sum_{c_1 \in S_k} \sum_{c_2=1}^{C} \hat{p}(c_1, c_2) \\ &= \sum_{u \in S_k^o} \sum_{v \in V_g} p(u, v) = \sum_{u \in S_k^o} p_V(u). \end{aligned} \quad (52)$$

From (49), (51) and (52), it then follows that $\hat{Q}(\mathcal{P}^a) = Q(\mathcal{P}^o)$.

Appendix E

In this section, we prove Theorem 12.

(i) Since we choose two sets that have a nonnegative correlation measure, i.e., $q(S_i, S_j) \geq 0$, to merge, it is easy to see from (30) and (29) that the modularity is non-decreasing in every iteration.

(ii) Suppose that there is only one set left. Then this set is V_g and it is the trivial community. On the other hand, suppose that there are $C \geq 2$ sets $\{S_1, S_2, \ldots, S_C\}$ left when the algorithm converges. Then we know that $q(S_i, S_j) < 0$ for $i \neq j$.

Note from (27) and (28) that for any node v,

$$q(v, V_g) = \sum_{w \in V_g} q(v, w) = 0. \quad (53)$$

Thus,

$$q(S_i, V_g) = \sum_{v \in S_i} q(v, V_g) = 0. \quad (54)$$

Since $\{S_1, S_2, \ldots, S_C\}$ is a partition of V_g, it then follows that

$$0 = q(S_i, V_g) = q(S_i, S_i) + \sum_{j \neq i} q(S_i, S_j). \quad (55)$$

Since $q(S_i, S_j) < 0$ for $i \neq j$, we conclude that $q(S_i, S_i) > 0$ and thus S_i is a community.

Appendix F

In this section, we prove Theorem 13.

It suffices to show that if node v is in a set S_1 and $q_0(v, S_2) > q_0(v, S_1)$, then move node v from S_1 to S_2 increases the value of the modularity. Also let $\mathcal{P}$ (resp. $\mathcal{P}'$) be the partition before (resp. after) the change. Since $q_0(\cdot,\cdot)$ is symmetric and $q_0(v, v) = 0$, it follows from (33) that

$$\begin{aligned} &Q(\mathcal{P}') - Q(\mathcal{P}) \\ &= q_0(S_1 \backslash \{v\}, S_1 \backslash \{v\}) + q_0(S_2 \cup \{v\}, S_2 \cup \{v\}) \\ &\quad - q_0(S_1, S_1) - q_0(S_2, S_2) \\ &= 2\Big(q_0(v, S_2) - q_0(v, S_1)\Big) > 0. \end{aligned}$$

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

Appendix G

In this section, we prove Theorem 14.

(i) This is a direct consequence of the modularity increasing result in Theorem 13 and the modularity preserving result in Lemma 11.

(ii) Suppose that Algorithm 3 returns a partition $\{S_1, S_2, \ldots, S_C\}$. Observe from (F2) of Algorithm 3 that the algorithm terminates when the deepest level of the partitional algorithm returns the same partition as its input partition. Let $p^*(u, v)$, $\tilde{p}^*(u, v)$, and $q^*(u, v)$ be the bivariate distribution that has the same marginal distributions, the symmetric bivariate distribution (see (22)) and the correlation measure (see (27)) at the deepest level, respectively. Also, let $q_0^*(u, v) = q^*(u, v)$ for $u \neq v$ and 0 otherwise. As the initial partition in (F0) is the partition that contains a single element in each set, the condition that the output partition is the same as the input partition (see (P1) in Algorithm 2) implies that

$$q^*(u, v) = q_0^*(u, v) \leq q_0^*(v, v) = 0$$

for all $u \neq v$ and $\tilde{p}^*(u, v) > 0$. On the other hand, if $\tilde{p}^*(u, v) = 0$, then $q^*(u, v) \leq 0$. Thus, we conclude that in the deepest level $q^*(u, v) \leq 0$ for all $u \neq v$. Let S_u be the set recursively expanded from the (super)giant node u in the deepest level. Analogous to the modularity preserving proof in Lemma 11, one can easily show that $q(S_u, S_v) = q^*(u, v)$ and thus $q(S_u, S_v) \leq 0$ for all $S_u \neq S_v$. For a correlation measure, we know that $\sum_{v=1}^{C} q(S_u, S_v) = 0$. Thus, $q(S_u, S_u) \geq 0$ and S_u is indeed a community (cf. the proof of Theorem 12(ii) in Appendix E).