Research Article

Semisupervised Community Detection by Voltage Drops

Min Ji, 1 Dawei Zhang, 1 Fuding Xie, 2 Ying Zhang, 3 Yong Zhang, 1 and Jun Yang 2

1 School of Computer Science and Technology, Liaoning Normal University, Dalian, Liaoning 116081, China
2 School of Urban and Environmental Science, Liaoning Normal University, Dalian, Liaoning 116029, China
3 College of Business Administration, Dalian University of Finance and Economics, Dalian, Liaoning 116622, China

Correspondence should be addressed to Fuding Xie; xiefd@lnnu.edu.cn

Received 3 November 2015; Revised 28 December 2015; Accepted 30 March 2016

1. Introduction

From the point of view of mathematics, many real-world systems in nature and society can be effectively modeled as complex networks or graphs. Specifically, the entities of the system are represented by the vertices and the interactions between the entities are represented by the edges. Examples include social relationships, spreading of viruses and diseases, the World Wide Web, author cooperation networks, citation networks, and biochemical networks. It has been shown that many real-world networks have a structure of modules or communities, where the nodes within a community are higher connected to each other than the nodes among communities. The community structures play an important role in the functional properties of complex network, and finding such a structure could be of significant practical importance.

Identifying community structure in special networks has a considerable merit of practice because it gives us insights to the structure-functionality relationship. In the past decades, plenty of techniques have been proposed to detect the community structure hidden in networks. The more typical algorithms for community detection can be found in [1]. Very recently, Chen et al. [2] defined the antimodularity as a quantitative measure of anticomunity partitioning on a network and showed the reliability of antimodularity as a measurement of the quality of an anticomunity partitioning. A vertices similarity probability model to find community structure without the prior knowledge of the type of complex network structure was presented [3]. By studying the community structure in Chinese character network, Zhang et al. [4] found that community structure was always considered as one of the most significant features in complex networks, and it played an important role in the topology and function of the networks. Palla et al. [5] revealed that complex network models exhibited an overlapping community structure, also called fuzzy community. These complicated structures actually make it harder to appropriately construct algorithms to uncover them. Along this way, researchers have made great contributions to the community detection [6–10].

The methods mentioned above belong to unsupervised community detection methods since the topological information of the network is used only and its background knowledge is ignored. In fact, some prior information is of great value in identifying the community structure. Based on the discussion of an equivalence of the objective functions of the symmetric nonnegative matrix factorization and the maximum optimization of modularity density, Ma et al. [11] introduced a semisupervised clustering algorithm for
community structure detection. In [12], Silva and Zhao presented a technique for semisupervised classification tasks, by using the modularity measure of complex networks, originally designed for unsupervised learning tasks. Zhang [13, 14] developed a method that implicitly encoded the pairwise constraints by modifying the adjacency matrix of the network, which could also be regarded as the denoising process of the consensus matrix of the community structures. A novel semisupervised community detection algorithm was proposed based on the discrete potential theory [15]. It effectively incorporated individual labels, the labels of corresponding communities, to guide the community detection process for achieving better accuracy. Although these existing semisupervised community detection methods can improve the community identification accuracy, some of them have limitations in high time complexity. Therefore, it is worthwhile to introduce the novel algorithm to identify community structures in complex network rapidly.

The application of discrete potential theory to detect community in network can be traced back to Wu and Huberman’s work [16]. They presented a method that allowed for the discovery of communities within graphs of arbitrary size in times that scale linearly with their size. Their method was based on notions of voltage drops across networks that were both intuitive and easy to solve regardless of the complexity of the graph involved. Zhang et al. [17] applied it to directed networks; they presented a new mechanism for the local organization of directed networks and designed the corresponding link prediction algorithm. Wang and Zhang [18] came up with a semisupervised clustering method based on generalized point charge models for text data classification. Liu et al. [15] recently proposed a linear time algorithm to find the community in network based on discrete potential theory. As data sets get larger and larger, it is still necessary to develop the efficient semisupervised learning methods.

Motivated by Wu and Huberman’s work [16] and Liu’s method [15], in this paper, we present a simple and fast semisupervised algorithm for detecting the community in complex network by discrete potential theory and voltage drops. The complexity of the proposed algorithm is $O(|V| + |E|)$ for the sparse network with $|V|$ vertices and $|E|$ edges. Similar to the membership degree in fuzzy $c$-means algorithm, the voltage value of each vertex in network implies the relationship between this vertex and community. The main contributions of the proposal are as follows: (1) the proposed algorithm is a simple and fast semisupervised approach to discover community structures in complex network. (2) The proposal gets rid of the limitation of positive definite matrix which is needed to solve a linear system by conjugate gradient decent algorithm. To some extent, this approach remedies the deficiency of Liu et al’s work [15]. (3) The unsupervised Wu-Huberman algorithm is extended to semisupervised learning case. The experimental results demonstrate the effectiveness of the proposed algorithm.

2. The Graph and Discrete Potential Method

The graph can be mathematically represented as $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices and $E \subset V \times V$ denotes the set of edges. Generally, the graph can be expressed by its adjacent matrix $A$, whose elements $A_{ij}$ are equal to 1 if $v_i$ points to $v_j$ and 0 otherwise. We denote $d_i$ as the degree of vertex $v_i$. The degree matrix $D$ is a diagonal matrix containing the vertex degree $d_i$ ($i = 1, 2, \ldots, n$) of a graph on the diagonal. Then the Laplacian matrix $L$ can be defined as

$$L = D - A.$$

Denote $x_i^s$ as the potential of vertex $v_i$ in the electrostatic field generated by vertices with label $s$. Assign the potentials of all labeled vertices with labels other than $s$ to zero and the $s$-labeled vertices to have a unit potential. The process of potential transmission for each electrostatic field is a circuit theory problem and can be modeled by combinatorial Dirichlet [15]. By using the Laplacian matrix $L$, a combinatorial formulation of the Dirichlet integral is in the form [15, 19]

$$D[x] = \frac{1}{2}x^TLx,$$

where $x$ is the potentials of all vertices minimizing (2). Reassigning the order of all vertices of the graph and putting the labeled vertices forward, (2) can be rewritten into

$$D[x] = \frac{1}{2}\begin{bmatrix} L_L & B \\ B^T & L_U \end{bmatrix} \begin{bmatrix} x_L \\ x_U \end{bmatrix}$$

(3)

$$= \frac{1}{2}\begin{bmatrix} x_L^TL_Lx_L + 2x_U^TB^Tx_L + x_U^TL_Ux_U \\ x_U^TL_Lx_L \end{bmatrix},$$

where $x_L$ and $x_U$ are two vectors whose elements represent the potentials of labeled vertices and unlabeled vertices, respectively. Setting the derivative of $D[x]$ with respect to $x_U$ equal to zero, one can obtain a system of linear equations

$$L_Ux_U = -B^Tx_L,$$

(4)

where $x_U$ is a $|V_U|$ dimensional vector whose elements are unknown quantities needing to be solved. If the graph is connected, or if every connected component contains a seed, then (4) will be nonsingular.

For each label $s$, a system of linear equations can be established as

$$L_Ux^s = -B^T\mu^s.$$

(5)

If one assigns a unit potential to the labeled vertices with label $s$ and zero to other labeled vertices, it will generate an electrostatic field. The potentials of unlabeled vertices can be obtained by the solution of (5). By comparing the potentials of each unlabeled vertex, its label is assigned the same as the labeled vertex corresponding to the greatest potential. Thus the community structure is detected.

From the perspective of discrete potential theory, the solution to (5) can be interpreted as a circuit theory. Based on the three fundamental equations of circuit theory, Kirchhoff’s Current Law, Ohm’s Law, and Kirchhoff’s Voltage Law, one can also get an equivalent system of (5) [15, 19].

In [15], the solutions of (5) have been obtained by conjugate gradient decent algorithm, and a novel semisupervised community detection algorithm was proposed. Several experimental results demonstrate the effectiveness of their approach.
3. The Proposed Algorithm

It should be noted that the coefficient matrix $L$, in (5) must be a symmetric positive definite matrix while solving the nonhomogeneous linear equations (5) by conjugate gradient decent algorithm. Obviously, Laplacian matrix $L$ is not a positive definite matrix since every row of the Laplacian matrix sums to zero, $0$ is always its eigenvalue, and the corresponding eigenvector is $(1, 1, \ldots, 1)$. This fact compels us to develop a new method to detect communities in network while considering the network as an electric circuit. In [16], Wu and Huberman introduced an unsupervised method to solve the system like (5) to discover the communities in complex network in linear time. Since there is no class information in advance, they employed bipartite strategy and some superb skills for the case of multiple communities. In this work, we extend their work to the case of semisupervised community detection.

In what follows, we would like to present a novel method to find community structure in complex networks by the process of voltage transmission.

For a given network, we suppose each edge to be a resistor with the same resistance. One attaches all the labeled vertices with label $s$ to anode of a battery and other labeled vertices to negative pole so that they have fixed voltages, say 1 and 0. Based on these assumptions, the network can be viewed as an electric circuit with current flowing through each edge (resistor). By solving Kirchhoff equations, one can obtain voltages of a vertex for the different labels if there are $n$ classes. In semisupervised learning methods, it is required that at least one sample must be labeled in each class. This indicates that the class parameter $k$ is known previously.

Physically, if node $v_i$ connects to $m$ neighbors $v_1, v_2, \ldots, v_m$ in an electric circuit, the Kirchhoff equation [20] tells us that the total current flowing into $v_i$ should sum up to zero; that is,

$$\sum_{j=1}^{m} I_j = \sum_{j=1}^{m} \frac{V_j - V_i}{R} = 0, \quad (6)$$

where $I_j$ is the current flowing from $v_j$ to $v_i$ and $V_j$ is the voltage at neighbor node $v_j$.

It is easy to rewrite (6) into the following form:

$$V_i = \frac{1}{m} \sum_{j=1}^{m} V_j, \quad (7)$$

That is to say, the voltage of a node is the average of those voltages of its neighbors.

Suppose the number of communities to be $k$; then the label set $K = \{1, 2, \ldots, k\}$. In addition, we also assume that there must be at least one labeled vertex in each community. Divide the vertex set $V$ into two parts, $V_L = \{(v_1, y_1), (v_2, y_2), \ldots, (v_{n_1}, y_1)\}$ (labeled vertices), where $y_i \in K$ is the label of vertex $v_i$ and $V_U = \{v_{n_1+1}, \ldots, v_n\}$ (unlabeled vertices) such that $V_L \cap V_U = \emptyset$. One also defines set $V_L^s = \{v_i \mid v_i \in V_L, y_i = s\}$.

For a given network, suppose each edge to be a resistor with the same resistance. One attaches all the labeled vertices with label $s$ to anode of a battery and other labeled vertices to negative pole so that they have fixed voltages, say 1 and 0. Based on these assumptions, the network can be viewed as an electric circuit with current flowing through each edge (resistor). By solving Kirchhoff equations, one can obtain voltages of a vertex for the different labels if there are $n$ classes. In semisupervised learning methods, it is required that at least one sample must be labeled in each class. This indicates that the class parameter $k$ is known previously.

Physically, if node $v_i$ connects to $m$ neighbors $v_1, v_2, \ldots, v_m$ in an electric circuit, the Kirchhoff equation [20] tells us that the total current flowing into $v_i$ should sum up to zero; that is,

$$\sum_{j=1}^{m} I_j = \sum_{j=1}^{m} \frac{V_j - V_i}{R} = 0, \quad (6)$$

where $I_j$ is the current flowing from $v_j$ to $v_i$ and $V_j$ is the voltage at neighbor node $v_j$.

It is easy to rewrite (6) into the following form:

$$V_i = \frac{1}{m} \sum_{j=1}^{m} V_j, \quad (7)$$

That is to say, the voltage of a node is the average of those voltages of its neighbors.

Suppose the number of communities to be $k$; then the label set $K = \{1, 2, \ldots, k\}$. In addition, we also assume that there must be at least one labeled vertex in each community. Divide the vertex set $V$ into two parts, $V_L = \{(v_1, y_1), (v_2, y_2), \ldots, (v_{n_1}, y_1)\}$ (labeled vertices), where $y_i \in K$ is the label of vertex $v_i$ and $V_U = \{v_{n_1+1}, \ldots, v_n\}$ (unlabeled vertices) such that $V_L \cap V_U = \emptyset$. One also defines set $V_L^s = \{v_i \mid v_i \in V_L, y_i = s\}$.

For a given network, suppose each edge to be a resistor with the same resistance. One attaches all the labeled vertices with label $s$ to anode of a battery and other labeled vertices to negative pole so that they have fixed voltages, say 1 and 0. Based on these assumptions, the network can be viewed as an electric circuit with current flowing through each edge (resistor). By solving Kirchhoff equations, one can obtain voltages of a vertex for the different labels if there are $n$ classes. In semisupervised learning methods, it is required that at least one sample must be labeled in each class. This indicates that the class parameter $k$ is known previously.

Physically, if node $v_i$ connects to $m$ neighbors $v_1, v_2, \ldots, v_m$ in an electric circuit, the Kirchhoff equation [20] tells us that the total current flowing into $v_i$ should sum up to zero; that is,

$$\sum_{j=1}^{m} I_j = \sum_{j=1}^{m} \frac{V_j - V_i}{R} = 0, \quad (6)$$

where $I_j$ is the current flowing from $v_j$ to $v_i$ and $V_j$ is the voltage at neighbor node $v_j$.

It is easy to rewrite (6) into the following form:

$$V_i = \frac{1}{m} \sum_{j=1}^{m} V_j, \quad (7)$$

That is to say, the voltage of a node is the average of those voltages of its neighbors.

Suppose the number of communities to be $k$; then the label set $K = \{1, 2, \ldots, k\}$. In addition, we also assume that there must be at least one labeled vertex in each community. Divide the vertex set $V$ into two parts, $V_L = \{(v_1, y_1), (v_2, y_2), \ldots, (v_{n_1}, y_1)\}$ (labeled vertices), where $y_i \in K$ is the label of vertex $v_i$ and $V_U = \{v_{n_1+1}, \ldots, v_n\}$ (unlabeled vertices) such that $V_L \cap V_U = \emptyset$. One also defines set $V_L^s = \{v_i \mid v_i \in V_L, y_i = s\}$.

If we reassign the order of all vertices of the graph and put the labeled vertices forward and labeled vertices with label $s$ first, following (6), one can get the system:

$$V_i^s = 1, \quad (8)$$

$$V_i^s = 0, \quad (9)$$

$$V_i^s = \frac{1}{d_i} \sum_{j \in \mathcal{N}(v_i)} V_j^s A_{ij}, \quad (10)$$

for $i = l + 1, l + 2, \ldots, n$. Equation (10) is a linear system with $n - l$ variables and can be put into a symmetrical form as follows:

$$V_i^s = \frac{1}{d_i} \sum_{j=1}^{n} V_j^s A_{ij} + \frac{1}{d_i} A_{i1} + \cdots + \frac{1}{d_i} A_{ir}, \quad (11)$$

for $i = l + 1, l + 2, \ldots, n$.

Define

$$V^s = \begin{pmatrix} V_{l+1}^s \\ \vdots \\ V_n^s \end{pmatrix}, \quad (12)$$

$$B = \begin{pmatrix} A_{l+1,l+1} & \ldots & A_{l+1,n} \\ \vdots & \ddots & \vdots \\ A_{n,l+1} & \ldots & A_{n,n} \end{pmatrix},$$

$$C = \begin{pmatrix} d_{l+1} \\ \vdots \\ d_n \end{pmatrix}, \quad (13)$$

and then the matrix form of Kirchhoff equation is

$$V^s = BV + C, \quad (14)$$

which has the solution

$$V^s = (I - B)^{-1} C. \quad (15)$$

Generally, it will take $O((n - l)^3)$ time to solve this system. Wu-Huberman algorithm [16] skillfully avoids this difficulty by solving (8)–(10) for $r = 1$ and $l = 2$. This method seems naturally to be a semisupervised learning method. We now extend it to the case of semisupervised learning.

Specifically, we first set $V_i^s = 1$, for $v_i \in V_L$, and $V_i^s = 0$, for $v_i \in V - V_L$.
Starting from \( N(V_j) (v_j \in V_j^k) \), one consecutively updates the voltages of \( v_j \in V_j \) to
\[
V_i^s = \frac{1}{d_i} \sum_{v_j \in N(v_i)} V_j^s = \frac{1}{d_i} \sum_{j} V_j^s A_{ij}
\]  

The updating process adopts breadth-first search algorithm and it will end when we get voltages for all vertices in \( V_j \). This process is called a round. One spends an amount of \( O(d_i) \) time calculating neighbor voltage of vertex \( v_j \) and \( |V| \) time setting initial voltages; therefore the complexity in one round is \( O(|V| + |E|) \). After repeating the updating process for a finite number of rounds, one will reach an approximate solution of (14) within a certain precision which only depends upon the number of iteration rounds.

Unlike Wu and Huberman’s method [20], we do not need to compute the ideal voltage gap and know roughly the size of each community. As a result, we get a \((n-l)\)-dimensional voltage vector. The component \( V_i^s \) reflects the relationship of vertex \( v_i \) and \( s\)th community. For each label \( s \) in label set \( K = \{1, 2, \ldots, k\} \), we repeat this process. Therefore, for each vertex \( v_i \), we obtain a voltage vector \( (V_i^1, V_i^2, \ldots, V_i^k) \). The element \( V_i^j \) can be considered as the membership degree which vertex \( v_i \) belongs to the \( j\)th community. The vertex \( v_i \) is within the \( j\)th community if \( V_i^j = \max V_i^s \), \( 1 \leq s \leq k \). That is to say, largest voltage of each vertex indicates to which community the vertex \( v_i \) should belong.

4. Experiments

To validate the proposed algorithm, one would like to test it on four real networks and three benchmarks which are widely used to test the validity of various community division methods. The experimental platform is based on Windows 7 Ultimate Service Pack 1 with Intel® Core™ i5-3470 CPU 3.20 GHz, 4.00 GB memory, ×64 Operating system, and Java 1.8 Eclipse RCP Luna srl.

4.1. Three Evaluation Indices of Clustering. To assess the quality of partition, we here use the \( F\)-measure, \( P\)-measure, and modularity \( Q\) to quantify the clustering results. The \( F\)-measure is a harmonic combination of the precision and recall values used in information retrieval [21].

If \( n_i \) is the number of the members of class \( i \), and \( n_{ij} \) is the number of the members of class \( i \) in cluster \( j \), then the precision \( P_{ij} \) and recall \( R_{ij} \) can be defined as
\[
P_{ij} = \frac{n_{ij}}{n_j}, \quad R_{ij} = \frac{n_{ij}}{n_i}
\]

\( F_{ij} \) is denoted by
\[
F_{ij} = \frac{2 \times P_{ij} \times R_{ij}}{P_{ij} + R_{ij}}
\]
The corresponding \( F\)-measure (FM) of the whole clustering result is defined as
\[
FM = \frac{\sum_{i} n_{ij} \max_j F_{ij}}{N}
\]

where \( N \) is the total number of the members in the data set.

In general, the high value of \( F\)-measure indicates the better cluster result.

The purity of a cluster represents the fraction of the cluster corresponding to the largest class of data assigned to that cluster; thus the purity of cluster \( j \) is defined as
\[
P_j = \frac{1}{n_j} \max_{i} n_{ij}.
\]
The purity of the whole clustering result is defined as
\[
PM = \frac{\sum_{j} n_j}{N} P_j.
\]
In general, the larger the purity value is, the better the clustering result is.

In order to quantify the validity of community division of a complex network and to optimize the chosen splitting, we use, following [22], the concept of modularity. It is defined as follows: given a network division, Let \( e_{ij} \) be the fraction of edges in the network that connect vertices in group \( i \) to those in group \( j \), and let \( a_i = \sum_j e_{ij} \). Then the modularity \( Q \) is defined as
\[
Q = \sum_i (e_{ii} - a_i)^2.
\]
It measures the fraction of edges that fall between communities minus the expected value of the same quantity in a random graph with the same community division. Obviously, the larger \( Q \) corresponds to the ideal community structure.

4.2. Experiment on Four Real Networks. Testing an algorithm essentially means analyzing a network with a well-defined community structure and recovering its communities. In this subsection, four classical complex networks with known community structures are selected to test the introduced algorithm. The description of these four networks can be found everywhere [1, 11, 16, 23]. Taking Zachary Karate Club network with two communities, for example, we first choose randomly one node in each community and label it. Afterwards, the algorithm can work on this network and a community division is detected. The values of FM, PM, and \( Q \) can be computed according to the obtained partition. It is possible that the community division may be changed with the different selection of initial labeled notes. To evaluate the validity of the proposal objectively, we calculate the average values of three indices by choosing randomly 10 groups of initial labeled notes. Along this way, we also compute three indices values by adding the number of labeled notes in each community. In Table 1, we list the average values of three indices by selecting randomly 10 groups of different labeled
Table 1: The average values of indices for four real networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>Index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolphins</td>
<td>FM</td>
<td>0.8473</td>
<td>0.9571</td>
<td>0.9783</td>
<td>0.9902</td>
<td>0.9919</td>
<td>0.9952</td>
<td>0.9968</td>
<td>0.9903</td>
<td>0.9968</td>
<td>0.9936</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.8484</td>
<td>0.9613</td>
<td>0.979</td>
<td>0.9903</td>
<td>0.9919</td>
<td>0.9952</td>
<td>0.9968</td>
<td>0.9903</td>
<td>0.9968</td>
<td>0.9936</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.2476</td>
<td>0.338</td>
<td>0.3634</td>
<td>0.3757</td>
<td>0.3743</td>
<td>0.3805</td>
<td>0.3773</td>
<td>0.3743</td>
<td>0.3788</td>
<td>0.3774</td>
</tr>
<tr>
<td>Football</td>
<td>FM</td>
<td>0.8503</td>
<td>0.9179</td>
<td>0.9561</td>
<td>0.9667</td>
<td>0.9845</td>
<td>0.9913</td>
<td>0.9914</td>
<td>0.9939</td>
<td>0.9957</td>
<td>0.9936</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.867</td>
<td>0.9235</td>
<td>0.9574</td>
<td>0.9669</td>
<td>0.9843</td>
<td>0.9913</td>
<td>0.9913</td>
<td>0.9939</td>
<td>0.9957</td>
<td>0.9936</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.5544</td>
<td>0.5738</td>
<td>0.5763</td>
<td>0.5744</td>
<td>0.5657</td>
<td>0.5628</td>
<td>0.5608</td>
<td>0.5612</td>
<td>0.5582</td>
<td>0.554</td>
</tr>
<tr>
<td>Zachary</td>
<td>FM</td>
<td>0.764</td>
<td>0.8112</td>
<td>0.9409</td>
<td>0.9369</td>
<td>0.9615</td>
<td>0.9764</td>
<td>0.9853</td>
<td>0.9853</td>
<td>0.9882</td>
<td>0.9882</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.7235</td>
<td>0.7941</td>
<td>0.9412</td>
<td>0.9382</td>
<td>0.9618</td>
<td>0.9765</td>
<td>0.9853</td>
<td>0.9853</td>
<td>0.9882</td>
<td>0.9882</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.1655</td>
<td>0.2162</td>
<td>0.352</td>
<td>0.3449</td>
<td>0.3574</td>
<td>0.3642</td>
<td>0.3686</td>
<td>0.3705</td>
<td>0.3691</td>
<td>0.3704</td>
</tr>
<tr>
<td>Polbooks</td>
<td>FM</td>
<td>0.7003</td>
<td>0.772</td>
<td>0.8541</td>
<td>0.8588</td>
<td>0.8805</td>
<td>0.8825</td>
<td>0.8989</td>
<td>0.8994</td>
<td>0.9021</td>
<td>0.9249</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.7095</td>
<td>0.7762</td>
<td>0.8676</td>
<td>0.8629</td>
<td>0.88</td>
<td>0.8838</td>
<td>0.8934</td>
<td>0.8971</td>
<td>0.8943</td>
<td>0.921</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.2732</td>
<td>0.3703</td>
<td>0.4593</td>
<td>0.4704</td>
<td>0.4581</td>
<td>0.472</td>
<td>0.4688</td>
<td>0.4632</td>
<td>0.4772</td>
<td>0.4623</td>
</tr>
</tbody>
</table>

Table 2: The average run time of the proposed algorithm for four real networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>Label number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dolphins</td>
<td>0.7015</td>
</tr>
<tr>
<td>Football</td>
<td>1.4094</td>
</tr>
<tr>
<td>Zachary</td>
<td>0.1174</td>
</tr>
<tr>
<td>Polbooks</td>
<td>1.7469</td>
</tr>
</tbody>
</table>

From Table 1, it is easy to see that we can detect an ideal community division for these four networks by the proposed algorithm when we label 3 nodes in each community. The accuracy of network partition is greater than or equal to 94% except polbooks network. Three indices values are ascending or varying slightly with the increasing of labeled nodes. These results also show that one can detect a good network partition by labeling a small quantity of nodes in each community. For football network, we can get the same partition accuracy as in [15] while the number of the labeled vertices randomly selected is from 1 to 4.

In Table 2, the average run times of the proposed algorithm for four real networks are presented. It is shown that the run times decrease with the increase of labeled nodes. This is reasonable because the number of nodes that need to be divided is reduced.

In what follows, we apply the proposed algorithm to detect the communities on this benchmark. For each fixed number of labeled nodes, one also selects randomly 10 groups of different initial labeled nodes to compute the average values of three indices. The benchmark can be thought of as the network with apparent community structure if mixing parameter $\mu < 0.5$. From Table 3, one can see that four communities in this benchmark are detected accurately when mixing parameter $\mu < 0.25$ and the number of labeled nodes is equal to or greater than 4. If we take $\mu = 0.5$ and label 10 nodes in each community, 90% of nodes in this benchmark can be partitioned correctly. When $\mu > 0.5$, this benchmark is with overlapping community structures. Although the partition accuracy becomes higher and higher with the increasing of number of labeled nodes, we can not find ideal communities in this network. Particularly, our algorithm fails to divide it into four groups when $\mu = 1$ and number of labeled nodes in each group is less than 10.

Assuming that both the degree and the community size distributions are power laws, Lancichinetti et al. [25] designed a more general benchmark for testing community detection algorithms on graphs. Some parameters used in this benchmark are explained as follows:

- $N$: number of nodes,
- $k$: average degree,
- $\text{max } k$: maximum degree,
In this benchmark, $N$, $k$, max $k$, and $\mu$ have to be specified. For the others, the program can use default values: $t_1 = 2$; $t_2 = 1$; $\text{on} = 0$; $\text{om} = 0$; min $c$ and max $c$ will be chosen close to the degree sequence extremes.

If we set parameters $N = 128$, $k = 16$, max $k = 16$, min $c = 32$, and max $c = 32$, a kind of Girvan-Newman benchmark will be obtained.

To test the validity of our algorithm on large network, we apply the proposed algorithm to this benchmark with parameters $N = 10^5$, $k = 20$, max $k = 10^3$, and $t_2 = 1$. The mixing parameter $\mu$ is varied from 0.1 to 0.6. For each fixed $\mu$, one takes $t_1 = 2$ and $t_1 = 3$, respectively. Unlike the GN benchmark, the community size is power laws in this network. Therefore, it is proper to label nodes in each community in terms of node proportion. The minimal proportion which we will take is 10% because of the requirement that there
Table 3: The average values of indices on GN benchmark.

<table>
<thead>
<tr>
<th>( \mu )</th>
<th>Index</th>
<th>FM</th>
<th>PM</th>
<th>Label number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0625</td>
<td>( \mu )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.1250</td>
<td>Q</td>
<td>0.6875</td>
<td>0.6875</td>
<td>0.6875</td>
</tr>
<tr>
<td>0.1875</td>
<td>FM</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.2500</td>
<td>Q</td>
<td>0.5625</td>
<td>0.5625</td>
<td>0.5625</td>
</tr>
<tr>
<td>0.3125</td>
<td>FM</td>
<td>0.9733</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.3750</td>
<td>Q</td>
<td>0.4135</td>
<td>0.4686</td>
<td>0.4973</td>
</tr>
<tr>
<td>0.4375</td>
<td>FM</td>
<td>0.8592</td>
<td>0.8994</td>
<td>0.914</td>
</tr>
<tr>
<td>0.5000</td>
<td>Q</td>
<td>0.7277</td>
<td>0.8017</td>
<td>0.8659</td>
</tr>
<tr>
<td>0.5625</td>
<td>FM</td>
<td>0.8601</td>
<td>0.8665</td>
<td>0.8617</td>
</tr>
<tr>
<td>0.6250</td>
<td>Q</td>
<td>0.7336</td>
<td>0.8031</td>
<td>0.8664</td>
</tr>
<tr>
<td>0.6875</td>
<td>FM</td>
<td>0.9264</td>
<td>0.9734</td>
<td>0.9977</td>
</tr>
<tr>
<td>0.7500</td>
<td>Q</td>
<td>0.3112</td>
<td>0.3417</td>
<td>0.3819</td>
</tr>
<tr>
<td>0.8125</td>
<td>FM</td>
<td>0.5788</td>
<td>0.6648</td>
<td>0.761</td>
</tr>
<tr>
<td>0.8750</td>
<td>Q</td>
<td>0.495</td>
<td>0.1614</td>
<td>0.1882</td>
</tr>
<tr>
<td>0.9375</td>
<td>FM</td>
<td>0.4698</td>
<td>0.5567</td>
<td>0.6344</td>
</tr>
<tr>
<td>1.0000</td>
<td>Q</td>
<td>0.415</td>
<td>0.3126</td>
<td>0.3298</td>
</tr>
</tbody>
</table>

\( Q \) refers to the quality index, which is used to evaluate the performance of the algorithms on the GN benchmark.
Table 4: The mean values of indices on power law benchmarks.

<table>
<thead>
<tr>
<th>Network</th>
<th>Index</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>pw-0.1-2-1</td>
<td>FM</td>
<td>0.9585</td>
<td>0.9627</td>
<td>0.9707</td>
<td>0.9739</td>
<td>0.9795</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.958</td>
<td>0.9622</td>
<td>0.9703</td>
<td>0.9736</td>
<td>0.9792</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.8152</td>
<td>0.8155</td>
<td>0.8204</td>
<td>0.8233</td>
<td>0.8256</td>
</tr>
<tr>
<td>pw-0.1-3-1</td>
<td>FM</td>
<td>0.9602</td>
<td>0.9658</td>
<td>0.9709</td>
<td>0.9755</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.9524</td>
<td>0.9594</td>
<td>0.9657</td>
<td>0.9718</td>
<td>0.9775</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.789</td>
<td>0.7987</td>
<td>0.807</td>
<td>0.8148</td>
<td>0.8231</td>
</tr>
<tr>
<td>pw-0.2-2-1</td>
<td>FM</td>
<td>0.8968</td>
<td>0.9247</td>
<td>0.9332</td>
<td>0.9514</td>
<td>0.9579</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.8956</td>
<td>0.9225</td>
<td>0.9318</td>
<td>0.9499</td>
<td>0.9568</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.6756</td>
<td>0.6861</td>
<td>0.6922</td>
<td>0.701</td>
<td>0.7016</td>
</tr>
<tr>
<td>pw-0.2-3-1</td>
<td>FM</td>
<td>0.8923</td>
<td>0.9125</td>
<td>0.9241</td>
<td>0.9362</td>
<td>0.9465</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.8711</td>
<td>0.8953</td>
<td>0.9097</td>
<td>0.9252</td>
<td>0.9383</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.6079</td>
<td>0.6323</td>
<td>0.6478</td>
<td>0.6635</td>
<td>0.6772</td>
</tr>
<tr>
<td>pw-0.4-2-1</td>
<td>FM</td>
<td>0.8012</td>
<td>0.8582</td>
<td>0.8841</td>
<td>0.9066</td>
<td>0.9264</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.8125</td>
<td>0.8537</td>
<td>0.8798</td>
<td>0.9025</td>
<td>0.9231</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.5312</td>
<td>0.5539</td>
<td>0.5602</td>
<td>0.5739</td>
<td>0.5828</td>
</tr>
<tr>
<td>pw-0.4-3-1</td>
<td>FM</td>
<td>0.7064</td>
<td>0.7521</td>
<td>0.7933</td>
<td>0.8245</td>
<td>0.8574</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.6776</td>
<td>0.7228</td>
<td>0.7651</td>
<td>0.7995</td>
<td>0.8369</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.3901</td>
<td>0.4159</td>
<td>0.4466</td>
<td>0.4707</td>
<td>0.4988</td>
</tr>
<tr>
<td>pw-0.5-2-1</td>
<td>FM</td>
<td>0.7013</td>
<td>0.7532</td>
<td>0.81</td>
<td>0.8386</td>
<td>0.8712</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.7083</td>
<td>0.7537</td>
<td>0.8059</td>
<td>0.8354</td>
<td>0.8675</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.4218</td>
<td>0.4383</td>
<td>0.4562</td>
<td>0.4719</td>
<td>0.4814</td>
</tr>
<tr>
<td>pw-0.5-3-1</td>
<td>FM</td>
<td>0.7405</td>
<td>0.8253</td>
<td>0.8679</td>
<td>0.8994</td>
<td>0.9226</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.7125</td>
<td>0.7957</td>
<td>0.8401</td>
<td>0.8747</td>
<td>0.9025</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.351</td>
<td>0.4014</td>
<td>0.4291</td>
<td>0.4512</td>
<td>0.4698</td>
</tr>
<tr>
<td>pw-0.6-2-1</td>
<td>FM</td>
<td>0.5425</td>
<td>0.6709</td>
<td>0.7346</td>
<td>0.7663</td>
<td>0.8152</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.5603</td>
<td>0.671</td>
<td>0.7335</td>
<td>0.764</td>
<td>0.8119</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.2966</td>
<td>0.3432</td>
<td>0.3619</td>
<td>0.3702</td>
<td>0.385</td>
</tr>
<tr>
<td>pw-0.6-3-1</td>
<td>FM</td>
<td>0.5021</td>
<td>0.6245</td>
<td>0.6955</td>
<td>0.7542</td>
<td>0.8059</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.4752</td>
<td>0.5846</td>
<td>0.6539</td>
<td>0.7132</td>
<td>0.7689</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.2017</td>
<td>0.2365</td>
<td>0.2601</td>
<td>0.2851</td>
<td>0.3123</td>
</tr>
<tr>
<td>pw-0.7-2-1</td>
<td>FM</td>
<td>0.4258</td>
<td>0.5589</td>
<td>0.6464</td>
<td>0.7004</td>
<td>0.7551</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.4394</td>
<td>0.5607</td>
<td>0.6423</td>
<td>0.6964</td>
<td>0.7496</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.1988</td>
<td>0.2389</td>
<td>0.2642</td>
<td>0.2802</td>
<td>0.2882</td>
</tr>
<tr>
<td>pw-0.7-3-1</td>
<td>FM</td>
<td>0.3891</td>
<td>0.5211</td>
<td>0.612</td>
<td>0.6895</td>
<td>0.7537</td>
</tr>
<tr>
<td></td>
<td>PM</td>
<td>0.3665</td>
<td>0.4811</td>
<td>0.5632</td>
<td>0.6383</td>
<td>0.7084</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>0.1494</td>
<td>0.1739</td>
<td>0.191</td>
<td>0.2112</td>
<td>0.2309</td>
</tr>
</tbody>
</table>

Table 4 shows the mean values of indices on power law benchmarks. The values in each column are descending with the increasing of mixing parameter $\mu$. This shows that a good network partition will not be found by the proposed algorithm for the network which communities overlap seriously.

Figure 3 presents the comparison of run time of our algorithm on two benchmarks with different parameters and label nodes number or label proportions. The increasing of labeled nodes number or label proportions implies that the number of unlabeled nodes in benchmarks is descending, and therefore it needs less and less time to partition network into groups.

We now present our experimental results on the LFR benchmark and further compare our proposal with GN algorithm [24], spectral clustering algorithm [1], NMF algorithm [20], and SNMF-SS algorithm [11] by a normalized mutual information index (NMI).

The LFR benchmark is designed by Lancichinetti et al. [25] and widely employed to test the performance of community structure identification. It allows user to specify distributions for both the community sizes and the degree distribution and then generates vertices and communities by sampling from those distributions. The mix parameter $\mu$ represents the average ratio of intracommunity adjacencies to total adjacencies. The large $\mu$ corresponds to the network with apparent community structure. In this paper, the input parameters of the LFR benchmark are the same for our algorithm and the comparative algorithms. For the different values of $\mu \in [0.50, 0.6, 0.7, 0.8, 0.9]$, we generated 50 instances for each of LFR benchmark graphs whose node degree is taken from a power law distribution with exponent 2 and community size from a power law distribution with exponent 1. Each graph has 1000 vertices, average degree of 15, maximum degree of 50, maximum for the community sizes of 50, and minimum for the community sizes of 5. The definition of NMI can be found everywhere [11, 15, 26].

From Figure 4, we can see that the values of NMI obtained by our algorithm are bigger than those gotten by the other four algorithms. The peak value of our approach is 0.732 at $\mu_{avg} = 0.9$. This value is bigger than the one 0.7 computed by SNMF-SS algorithm. Because the decrease of $\mu$ means that the LFR benchmark is with the obscure community structure, it is difficult to detect communities correctly for five algorithms. It is reasonable that the NMI values obtained by five algorithms become smaller and smaller as $\mu$ decreases. The NMF algorithm seems to be stable since it has a small decrease speed. The performance of our proposal decreases greatly while $\mu$ is greater than 0.6. This fact implies that our algorithm can not apply the networks with nonapparent community structure. However, compared with other four algorithms, our algorithm can gain the best performance.

5. Conclusions

In this paper, we propose a semisupervised community detection algorithm for partitioning network into groups. This approach amalgamates the discrete potential theory and Wu-Huberman algorithm. The complexity $O(|V| + |E|)$ of the introduced approach indicates that it can be applied to detect
community on large network. The validity of our proposal is demonstrated by applying it to four real networks and three benchmarks. The experimental results show that a good community division of a complex network is obtained by labeling a small quantity of nodes in each community. However, it is difficult to classify correctly the network with heavily overlapping communities or obscure community structure by our method. This fact can be seen from the experimental result on LFR benchmark. Therefore, it is worthwhile to further introduce new and fast algorithm to deal with this case.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This work is supported by NSFC (under Grants nos. 61373127 and 41471140).

References


Submit your manuscripts at http://www.hindawi.com