The Bridge Edge Label Propagation for Overlapping Community Detection in Social Networks

Jui-Le Chen[1,2], Jen-Wei Hu[1], and Chu-Sing Yang[1]

[1] Institute of Computer and Communication Engineering, National Cheng Kung University, Taiwan
{q38991051,q38001050,csyang}@mail.ncku.edu.tw
[2] Department of Computer Science, Tajen University, Taiwan

Abstract. Overlapping community detection aims to discover a set of groups in which each node belongs to at least one group. There are more proposed methods that interest in overlapping community detection to find out the groups which are not necessarily disjoint. In this paper, we propose a modify method that provides the detection results would be the same for each run. The accuracy for experimental result of overlapping community detection is better but not much time consume.

Keywords: Overlapping community, Label Propagation Algorithm.

1 Introduction

Community is formed by members which within a group interact with each other more frequently than with the others outside the group. Community detection aims to discover groups in a network where is given a set that contains members with connection property. However, most of the work has been done on disjoint community detection. Instead of one member just belongs to single community, it is possible for each member to have many communities simultaneously. Overlapping community detection aims to discover a set of groups in which each node belongs to at least one group. For the reason, there are more proposed methods that interest in overlapping community detection to find out the groups which are not necessarily disjoint.

In recent years, there are many kinds of methods which try to identify the overlapping community. (1) Clique percolation method[1] is based on the assumption that a community consists of fully connected sub-graphs and detects overlapping communities by searching for adjacent cliques. It is a popular approach for analyzing the overlapping community structure of networks. (2) Local expansion method[2] used the iterative scan algorithm(IS) to improve . (3) Fuzzy clustering method[3] provided the fuzzy c-mean algorithm. to embed the graph into low dimensionality Euclidean space. (4) Link partitioning method[4] using links instead of nodes to discover communities. (5) Dynamical Algorithms [5] and Speaker-listener Label Propagation Algorithm)[6] which are the label propagation algorithms use labels to discover communities.

J.-S. Pan et al. (eds.), *Intelligent Data Analysis and Its Applications, Volume 1*, Advances in Intelligent Systems and Computing 297,
DOI: 10.1007/978-3-319-07776-5_11,

The SLPA(Speaker-listener Label Propagation Algorithm)[6] method proposed a method for detecting both individual overlapping nodes and overlapping communities using the underlying network structure alone. SLPA accounts for overlap by allowing each node to possess multiple labels. In each communication step, one node is a speaker (information provider), and the other is a listener (information consumer). each node has a memory of the labels received in the past and takes its content into account to make the current decisions. However, the method although gives the complexity: O(Tnk) with maximum iteration(T), the average node degree(k) and the total number of nodes(n) that towards linear time but not provides the stable detection results.

For the reason, we propose a modify method that provides the detection results would be the same for each run. The accuracy for experimental result of overlapping community detection is better than SLPA but not much time consume.

2 Problem Definition

In this section, we present basic definitions that will be used throughout the paper. Given a network or undirected graph $G = E, V$, V is a set of n nodes and E is a set of m edges. In the case of overlapping community detection, the set of clusters found is called a cover or partition $C = \{c_1, c_2, ..., c_k\}$, in which a node may belong to more than one cluster.

3 The Proposed Method

In the proposed method, each node is assigned a unique community id as label and maintains a group list with size n, the number of nodes in network. At the first, the group list of each node is initialized with a null label. Then, the following steps are repeated until the maximum iteration T is reached: (a) Each node detects the label of each neighbor, if both are not the same then raising the bridge edge problem and determinates the two nodes are the same community or not. (b) Each node has a group list of the labels to record these labels of its neighbors at each position. Finally, based on the labels in the group list is applied to output the communities by using post-processing and community detection.

The Bridge Edge Problem is defined as the edge connecting two communities. When the bridge edge problem is arising , it means that both node of edge's side would be made decision for belongs to which community. Therefor, the overlap determination can be deal with the kind of the bridge edges. The good judgement is introduce by [7] with average degree for the concept of partition density. The definition of average degree is as follows: where c is a community, $E(c)$ is the number of edges in the community, and $|c|$ is the number of nodes of the community. If adding to the community makes $|AD(c)|$ increase, we suppose that the node contributes to the community.

$$AD(c) = \frac{2 * E(c)}{|c|} \tag{1}$$

Algorithm 1. Bridge Edge Label Propagation Algorithm

Each node is assigned a unique community id as label.
Each node maintains a group list with size n, the number of nodes in network and initialized with a null label.
while the termination criterion is not met($t < T$) **do**
    for each Node $N_i, i = 1$ to n **do**
        for each neighbor N_k of N_i **do**
            if (N_i, N_k) is not same group **then**
                Calculate $AD(c_k) = 2 * E(c_k)/|c_k|$
                Calculate $AD(c_{i,k}) = 2 * E(c_{i,k})/|c_{i,k}|$
                if $AD(c_k) < AD(c_{i,k})$ **then**
                    add Node(K) into C(i)
                end if
            end if
        end for
    end for
end while
Post-processing and community detection
(a)Detect the transitive property:
Transform all of node of $C(i)$ into $C(j)$ if $C(i) \subset C(j)$
(b)Construct a maximum community:
Two communities are adjacent if they share k-1 nodes, $|C(i)| = k < |C(j)|$
(c)Make a histgram(i) but except node(i) itself for each nodeś $grouplist[i]$:
In histgram(i), find out the frequent >= 2 then output the community id.

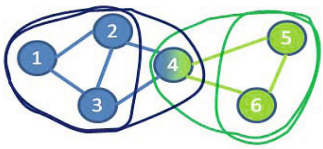
For examples, in Figure 1, there are two Communities 1:$\{1, 2, 3\}$ and 2:$\{4, 5, 6\}$ with bridge edges $e(2, 4), e(3, 4)$. At the result, we can find that node 4 is a member of both community 1 and 2, but node 3 and 2 just belong to community 1.

Another example showed in Figure 2: Node 3 is contained in community 1 and Node 4 is in community 2. Because node 3 decreases the $|AD(c)|$ of community 2, while adding into community 2, then node 3 should be just belong to community 1, which is the most reasonable partition. For the same reason, node 5 decreases the $|AD(c)|$ of community 1, while joins to community 1, and then node 5 would be in community 2.

3.1 Complexity Evaluation

The initialization of labels requires $O(n)$, where n is the total number of nodes. Each node has a group list of size n. The operation executed by each node in each iteration. Detects the label of each neighbor is the same or not which requires $O(N * k)$, where k is the average degree of node. If not, the bridge edge problem raised then processing the determination. Each bridge node would find out all edges in the same group which requires $O(2 * k * n_b)$, where n_b is the total number of bridge edges. The n_b would equal to the number of community h so that $O(2 * k * n_b)$ would be the $O(2 * k * h)$. At the result, the complexity

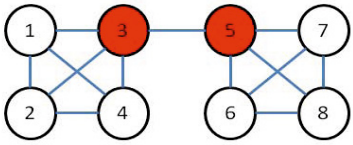
Two communities with bridge edges e(2,4), e(3,4)
(a)Community 1: {1,2,3}, AD(c) = 2*(3/3)=2
Adding node 4 to community 1:
AD(c) = 2*(5/4)=5/2 > 2, then
node 4 is also a member in community 1
(b) Community 2: {4,5,6}, AD(c) = 2*(3/3)=2
Adding node 2 to community 2:
AD(c) = 2*(4/4) = 2, then
node 2 does not belongs to community 2.
Same as node 3 not belongs to community 2



Group List	1	2	3	4	5	6
Node 1	1	1	1	1	0	0
Node 2	1	2	1	1	0	0
Node 3	1	1	3	1	0	0
Node 4	1	1	1	4	5	5
Node 5	0	0	0	5	5	5
Node 6	0	0	0	5	5	6
#freq > 2	1	1	1	1,5	5	5

Fig. 1. The overlap determination for network size=6

Two communities with bridge edges e(3,5)
(a)Community 1: {1,2,3,4}, AD(c) = 2*(6/4)=3
Adding node 5 to community 1:
AD(c) = 2*(7/5)=14/5 < 3, then
node 5 does not belong to community 1
(b) Community 2: {5,6,7,8}, AD(c) = 2*(6/4)=3
Adding node 3 to community 2:
AD(c) = 2*(7/6) = 14/5 < 3, then
node 3 does not belong to community 2.



Group List	1	2	3	4	5	6	7	8
Node 1	1	1	1	1	0	0	0	0
Node 2	1	2	1	1	0	0	0	0
Node 3	1	1	3	1	0	0	0	0
Node 4	1	1	1	4	5	0	0	0
Node 5	0	0	0	5	5	6	6	6
Node 6	0	0	0	0	6	6	6	6
Node 7	0	0	0	0	6	6	7	6
Node 8	0	0	0	0	6	6	6	8
#freq > 2	1	1	1	1	6	6	6	6

Fig. 2. The overlap determination for network size=8

in each iteration would be $O(N * k^2 * h)$ at the worst case(upper bound). The complexity of the other methods for overlapping detection show in the table 1.

Table 1. Complexity for overlapping community detection methods

Algorithm	Time Complexity
Clique Percolation Method(CPM)	$O(n^3)$
Local Expansion	$O(mh)$
Fuzzy Clustering	$O(hn^2)$
Link Partitioning	$O(nk_{max}^2)$
Dynamical Algorithms	$O(Tnk))$
n is the number of nodes, m is the number of edges. h is the number of cliques, k_{max} is the highest degree of the n nodes	

4 Experimental Result and Discussion

In this paper, the performance of the proposed algorithm is evaluated by using it to solve the prototype generation in nearest neighbor classification problem. All the experimental results are obtained by running on an IBM X3650 machine with 2.4 GHz Xeon CPU and 16GB of memory using CentOS 6.0 with Linux 2.6.32. Moreover, all the programs are written in C++ and compiled using GNU C++ compiler.

4.1 Parameter Settings and Datasets

To study the behavior of proposed method, we conducted the experiments in synthetic networks. For synthetic random networks, we adopted the widely used LFR[8] benchmark date set. Program source code for generating benchmark data set can be get from http://sites.google.com/site/andrealancichinetti/files. The parameters show as table 2.

Table 2. LRF's synthetic benchmark data set

1. The networks with size n = 1000.
2. The average degree is kept at k = 10.
3. The node degrees and community sizes are governed by the power laws with exponents 2 and 1;
4. The maximum degree is 50;
5. The community size varies from 20 to 100;
6. The expected fraction of links of a node connecting it to other communities, called the mixing parameter μ, is set to 0.3.
7. *On* defines the number of overlapping nodes is set to 10
8. *Om* defines the number of communities to which each overlapping node belongs and varies from 2 to 8 indicating the diversity of overlap.
9. The usage for generating benchmark $./benchmark - N1000 - k10 - t12 - t21 - maxk50 - minc20 - maxc100 - mu0.3 - on500 - om8$

4.2 Experimental Results

We focus on proposed method and SLPA method to compare the performance for finding the overlap communities. In total, 5 testing sets as the benchmark were collected and tested. They are listed in Table 3.

Table 3. The performance for SLPA and proposed method

	Execution time(secs)1000/2000/3000/4000/5000	Accuracy rate*
SLPA	25/53/79/113/140	0.3/0.53/0.41/0.37/0.46
Propose Algorithm	27/60/97/132/163	0.64/0.7/0.73/0.69/0.63
*Accuracy rate : all of nodes is in the same community or not / all of nodes		

5 Conclusion

We proposed a improved method for SLPA method to archive an efficient and effective for overlapping community detection. It is important to analyze the information in the social network which can provide more helps to model the overlapping community accurately. To know that how to find out the individual overlapping community with a programmable process.

References

1. Palla, G., Derényi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043), 814–818 (2005)
2. Lee, C., Reid, F., McDaid, A., Hurley, N.: Detecting highly overlapping community structure by greedy clique expansion. arXiv preprint arXiv:1002.1827 (2010)
3. Zhang, S., Wang, R.-S., Zhang, X.-S.: Identification of overlapping community structure in complex networks using fuzzy c-means clustering. Physica A: Statistical Mechanics and its Applications 374(1), 483–490 (2007)
4. Ahn, Y.-Y., Bagrow, J.P., Lehmann, S.: Link communities reveal multiscale complexity in networks. Nature 466(7307), 761–764 (2010)
5. Gregory, S.: Finding overlapping communities in networks by label propagation. New Journal of Physics 12(10), 103018 (2010)
6. Xie, J., Szymanski, B.K.: Towards linear time overlapping community detection in social networks. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS, vol. 7302, pp. 25–36. Springer, Heidelberg (2012)
7. Cai, Y., Shi, C., Dong, Y., Ke, Q., Wu, B.: A novel genetic algorithm for overlapping community detection. In: Tang, J., King, I., Chen, L., Wang, J. (eds.) ADMA 2011, Part I. LNCS, vol. 7120, pp. 97–108. Springer, Heidelberg (2011)
8. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical Review E 78(4), 046110 (2008)