Postprint of Local Community Detection Algorithm Based on Graph Traversal

Authors: Wu Jian, Wang Ziquan, Yi Yi, Sun Haixia

Date: 2018-05-24T00:00:00+00:00

Abstract

Discovering community structures in networks is of great significance for understanding the structure and function of complex networks. To address the problem that current local community detection algorithms have slow expansion speed and are not suitable for large-scale networks, a graph traversal-based local community detection algorithm is proposed. The algorithm first identifies the node with the lowest degree in the network, uses this node as a starting point to divide the nodes in the network into community nodes and boundary nodes through an influence function, forming a preliminary community partition, and then determines the community of boundary nodes through a fitness function to obtain the final partition result. Experimental results show that when tested on real networks, the algorithm can not only effectively detect community structures in the network but also has a relatively fast speed.

Full Text

Preamble

Title: Local Community Detection Algorithm Based on Graph Traversal

Authors: Wu Jian[1], Wang Ziquan[1]†, Yi Yi[1], Sun Haixia[2]
[1] College of Communication & Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2] College of Information Engineering, Xizang Minzu University, Xianyang, Shaanxi 712082, China

Abstract: Detecting community structure is significant for understanding the structures and functions of complex networks. To address the slow expansion speed of current local community detection algorithms and their unsuitability for large-scale networks, this paper proposes a local community detection algorithm based on graph traversal. The algorithm first identifies the node with the lowest degree in the network, then uses this node as a starting point to classify nodes

into community nodes and border nodes through an influence function, forming an initial community partition. Finally, the fitness function determines the communities of border nodes to obtain the final partition results. Experimental results demonstrate that when tested on real-world networks, the algorithm can not only effectively mine community structures but also operate with relatively fast speed.

Keywords: complex network; modularity; community detection; graph traversal

0 Introduction

With the advent of the big data era, the scale of complex networks has gradually increased, making global network information increasingly difficult to grasp. Consequently, global community detection algorithms have become inadequate for large-scale networks, prompting researchers to propose local community detection algorithms. Community structure is a ubiquitous topological property of complex networks, and identifying such structures has become a fundamental problem. These structures play crucial roles in controlling complex networks, analyzing network topology, predicting individual behaviors, and deeply understanding network functions. Typically, this structure manifests as a network that can be partitioned into many groups where intra-group connections are relatively dense while inter-group connections are relatively sparse [1-3]. Community detection utilizes topological information embedded in networks to discover modular structures, which is extremely important for controlling the spread of diseases and network viruses.

Due to different interpretations of community structure definitions, researchers follow various criteria when performing community partitioning, which can be divided into two categories: global community detection algorithms and local community detection algorithms [4]. Global community detection algorithms require obtaining complete network information beforehand and perform community partitioning from a global perspective, yielding highly credible results [5]. However, local community detection algorithms operate without prior global information, instead expanding communities based on local information of nodes (or subgraphs). Compared with global algorithms, local community detection algorithms offer greater advantages in runtime, applicability, and algorithmic flexibility, making them a current research hotspot.

Based on different partitioning philosophies, existing algorithms can be categorized into label propagation algorithms, clique percolation algorithms, and local expansion algorithms. The Label Propagation Algorithm (LPA) first assigns corresponding labels to all nodes in the network, then repeatedly updates node labels according to propagation rules until labels no longer change throughout the network. Nodes with identical labels constitute a community. Since LPA restricts each node to possessing only one label simultaneously, it cannot discover

overlapping structures. To address this limitation, Steve Gregory et al. proposed the COPRA algorithm [6], which allows nodes to hold multiple labels simultaneously, enabling them to belong to multiple communities and thus discover overlapping structures.

Clique percolation algorithms are based on percolation theory, defining a k-clique as a fully connected subgraph containing k nodes, where communities are collections of fully connected subgraphs sharing nodes. This algorithm first searches for all k-cliques in the network, then constructs a new graph with k-cliques as cores, connecting two k-cliques with an edge if they share k-1 nodes. Ultimately, each connected subgraph forms a community.

Local expansion algorithms start from given nodes (or subgraphs) and gradually merge neighboring nodes according to propagation rules and evaluation functions to discover community structures. For example, Lancichinetti et al. proposed the LFM algorithm [7], which uses seed nodes as centers and employs a metric function to discover community structures. However, since LFM randomly selects seed nodes from the network, its results lack stability and require multiple runs to obtain optimal results. To address this issue, Conrad Lee et al. proposed the GCE algorithm [8], which first identifies k-cliques in the network and then expands these k-cliques using LFM' s fitness function. Although GCE exhibits high stability, it suffers from long runtime, and the quality of its partitioning results heavily depends on the selection of the k value.

Currently, most local community detection algorithms add neighbor nodes to communities individually and randomly, processing only one node per iteration. This leads to numerous repeated calculations, low efficiency, slow community expansion, and difficulty in applying to large-scale networks. To solve this problem, this paper proposes a local community detection algorithm based on graph traversal. The fundamental idea is to use the node with minimum degree as the starting point, mark nodes in the network as community nodes or border nodes based on the proposed influence function and a threshold value r to form an initial community partition, and then obtain final results through the fitness function F.

---

1 Local Community Detection Algorithm Based on Graph Traversal

As indicated in literature [6, 11, 12], most existing local community detection algorithms can only determine the community affiliation of one node at a time during partitioning. To discover overlapping community structures, certain nodes require repeated evaluation, resulting in excessive iterations and significantly increased runtime, making them unsuitable for large-scale networks or networks with numerous communities. To address this limitation, this paper proposes a local community detection algorithm based on graph traversal that improves efficiency.

1.1 Basic Definitions

To accurately understand the proposed algorithm, we first provide the following definitions.

Definition 1 (Network Representation). Let G represent a network as G(V,E), where V = {v , v , v , ···, v } denotes the set of nodes and E = {e , e , e , ···, e } denotes the set of edges.

Definition 2 (Node Influence NIS(v)). The influence received by node v is defined as:

$$NIS(v) = \frac{N(v)}{degree(v)}$$

where N(v) represents the number of influenced neighbor nodes of v, and degree(v) represents the degree of node v. Clearly, NIS(v) always ranges between 0 and 1. For the starting node, its NIS value is zero. If all neighbor nodes of v are influenced, then NIS(v) equals 1.

Definition 3 (Border Node). If the influence NIS(v) received by node v is less than a predetermined threshold r (0 r 1), then this node is considered a border node (BN).

Definition 4 (Community Node). If the influence NIS(v) received by node v is greater than or equal to a predetermined threshold r (0 r 1), then this node is considered a community node, with its label determined based on specific circumstances.

Definition 5 (Overlapping Modularity). Overlapping modularity (EQ) is a function proposed by Shen et al. [9] that improves upon standard modularity. This function addresses the limitation that ordinary modularity (Q) [10] cannot evaluate overlapping communities. The EQ function considers nodes that simultaneously belong to multiple communities. Since such overlapping nodes belong to multiple communities, their impact on community structure density should be weakened when calculating modularity. Based on the principle that higher node overlap leads to smaller impact on modularity, the overlapping modularity is defined as:

$$EQ = \frac{1}{2m} \sum_{c} \sum_{i,j \in c} \frac{1}{O_i O_j} \left(A_{ij} - \frac{k_i k_j}{2m} \right)$$

where A represents elements of the network' s adjacency matrix, m represents the total number of edges, O represents the number of communities to which node i belongs, and k represents the degree of node i. Compared with the Q function, the EQ function reduces the influence of overlapping nodes on community structure density. When no overlapping nodes exist, the EQ function is equivalent to the Q function.

Definition 6 (Fitness Function). The fitness function, proposed by Lancichinetti, measures the quality of internal versus external connections of a community:

$$F_G = \frac{K_G^{in}}{(K_G^{in} + K_G^{out})^{\alpha}}$$

where K_G^{in} represents the internal degree of community G, K_G^{out} represents the external degree of community G, and is a resolution parameter (typically set to 1).

1.2 Initial Community Partition

During the initial community partition phase, the algorithm defines a novel node marking rule that can simultaneously determine the community affiliation of all neighbor nodes of a selected node, performing initial community partitioning on the network and reducing algorithmic iterations. After initial partitioning, network nodes are divided into two categories: border nodes and community nodes. Border nodes, receiving low influence, cannot have their community affiliation determined temporarily and require further evaluation. Community nodes with identical labels are considered stable and require no further determination during final partitioning, reducing the number of nodes to be evaluated in subsequent stages.

The initial community partition marking rules are as follows:

a) The starting node must be a border node, marked as BN.

b) If a node is a border node and its neighbor nodes are also discovered to be border nodes through this node, mark these neighbor nodes as BN.

c) If a node is a border node and its neighbor nodes are discovered to be community nodes through this node, change this node' s status from border node to community node, updating its label from BN to its current label, and mark its neighbor nodes with the same label as this node.

d) If a node is a community node and its neighbor nodes are discovered to be border nodes through this node, mark these neighbor nodes as BN while keeping this node' s label unchanged.

e) If a node is a community node and its neighbor nodes are also discovered to be community nodes through this node, mark these neighbor nodes with this node' s current label.

f) Each node is allowed to carry only one label.

1.3 Final Community Partition

After initial partitioning, network nodes are divided into community nodes and border nodes. To determine the community affiliation of border nodes and discover overlapping structures in the network, this paper adopts Lancichinetti' s fitness function F. Border nodes are merged into adjacent community nodes, and the F value is calculated before and after merging. If the F value after merging is greater than before merging, a merged community label is added to the border node. Each border node can possess multiple labels. After all border nodes have been merged, all BN labels are removed from the network. Nodes with identical labels are then partitioned into the same community.

1.4 Algorithm Steps

Input: Network G(V,E), threshold r
Output: Community partition results of network G

a) Calculate the degree of all nodes in the network and select the node with minimum degree as the starting node. If multiple nodes have the same minimum degree, randomly select one. Since no nodes are labeled at this point, mark this node as a border node (BN).

b) Starting from this node, calculate the influence received by all its neighbor nodes using Equation (1), and mark neighbor nodes according to the node marking rules.

c) Check whether currently marked nodes have unmarked neighbor nodes. If yes, calculate the influence received by all unmarked neighbor nodes using Equation (1), assign appropriate labels according to the rules, and repeat step c). If no, proceed to step d).

d) Select border nodes in the network and merge them with neighboring communities. Calculate the fitness function (F) before and after each merge using Equation (3). If the F value of the new community after merging is greater than before merging, update the label of the merged node according to the rules.

e) When all border nodes in the network have been merged, remove all BN labels. Nodes with identical labels are partitioned into the same community.

1.5 Time Complexity Analysis

As described above, the proposed algorithm consists of two main phases: initial community partitioning and final community partitioning. For a network containing n nodes and m edges, finding the starting node requires calculating the degree of all nodes and identifying the node with minimum degree, with time

complexity O(n). Initial community partitioning requires calculating NIS values for all nodes and marking nodes based on these values and threshold r, with time complexity O(n). Final community partitioning compresses community nodes with identical labels into new nodes and merges nodes in the network, calculating the F value after each merge until F can no longer increase. Considering the worst-case scenario where each node forms an independent community requiring pairwise merges, this phase has time complexity O(m). Therefore, the overall time complexity of the proposed algorithm is O(n + m). For sparse networks, the complexity can be approximated as O(n).

---

2 Algorithm Testing

2.1 Threshold Analysis

As the algorithm' s concept indicates, the threshold r significantly impacts both the quality of community structure and final partitioning results. Therefore, determining an appropriate initial value for r is essential. We use representative real-world datasets: Karate, Football, Dolphins, and Netscience, with detailed information provided in Table 1 .

The Karate dataset is a social network constructed from observations of member interactions in a university club, where nodes represent club members and edges represent friendship relationships. The Football dataset is a complex social network constructed by Newman et al. based on American college football leagues, where nodes represent football teams and edges indicate matches played between teams. The Dolphins dataset is a social network of dolphin interactions compiled by Lusseau et al., where nodes represent dolphins and edges represent frequent communication. The Netscience dataset is a co-authorship network of researchers working on network theory and experiments, where nodes represent researchers and edges indicate co-authored publications.

By running the algorithm 30 times on each of the four datasets and averaging the threshold r values, we found that when r ranges between 0.4 and 0.8, the obtained EQ values are relatively good. When threshold r = 0.7, the algorithm produces optimal community structures on most datasets (as shown in Figure 1 [Figure 1: see original paper]). Therefore, we select r = 0.7 for subsequent experiments.

2.2 Artificial Network Experiment

To better illustrate the algorithm' s process, we conducted experiments on a constructed simple network (Figure 2 [Figure 2: see original paper]). Table 2 shows the labels of each node at different stages during initial community partitioning when threshold r = 0.7. Starting from node 8, the network is divided into border nodes and community nodes after initial partitioning. Border nodes include nodes 3, 8, and 13, while community nodes include nodes 5 and 7 with

label 7, nodes 0, 1, 2, and 4 with label 0, and nodes 9, 10, 11, and 12 with label 10. During final partitioning, only border nodes need to be evaluated by merging them into neighboring communities and calculating the F function before and after merging using Equation (3). The final community partitioning results for this network are shown in Table 3 .

2.3 Large-Scale Network Dataset Experiments

To verify the algorithm' s effectiveness on large-scale datasets, we compared it with LFM, CPM, and COPRA algorithms, using the EQ function as the evaluation metric. Table 4 describes the real-world network data used.

Table 5 presents the test results of the proposed algorithm and the other three algorithms on three large-scale datasets. Experimental results demonstrate that the proposed algorithm achieves relatively good EQ values across different real-world datasets while requiring less runtime than the other three algorithms. The LFM algorithm randomly selects seed nodes from the network, causing its EQ values to vary across runs and making it susceptible to seed node location. Moreover, it evaluates only one node at a time during partitioning, leading to significant runtime increases as network scale grows. The CPM algorithm can mine high-quality communities in large-scale datasets but requires considerable time to find k-cliques. The COPRA algorithm consumes substantial time due to iterative label updates and performs poorly on large-scale networks. In contrast, the proposed algorithm can simultaneously determine the community affiliation of all neighbor nodes of a selected node during the initial partitioning phase, reducing iteration counts and forming a preliminary partition that decreases the number of nodes requiring evaluation in the final partitioning stage.

---

3 Conclusion

This paper proposes a local community detection algorithm based on graph traversal. The algorithm defines an influence function NIS to simultaneously update labels of multiple nodes based on received influence and threshold r, forming an initial community partition. Final partitioning is achieved through the fitness function, reducing repeated iterations when determining node community affiliation and improving algorithmic efficiency. Experiments prove that although community quality is affected by threshold r, the proposed algorithm generally achieves higher EQ values than comparative algorithms, indicating more accurate network community structures with lower time complexity. Future work will consider edge weights in networks to extend the algorithm to weighted networks.

References

[1] Zhang Xingyi, Wang Congtao, Su Yansen, et al. A fast overlapping community detection algorithm based on weak cliques for large-scale networks [J]. IEEE Trans on Computational Social Systems, 2017, 4(4): 218.

[2] Vespignani A. Complex networks: The fragility of interdependency [J]. Nature, 2010, 464(7291): 984-5.

[3] Wang Qi, Wen Zhiping. Multidimensional genetic algorithm for overlapping community detection [J]. Application Research of Computers, 2016, 33(12): 3543-3546.

[4] Liu Lihan, Fang Zhixiang, Xiao Shilun, et al. Fast communities detection algorithm with source nodes [J]. Computer Engineering and Applications, 2016, 52(23): 75-80.

[5] Li Jianhua, Wang Xiaofeng, Wu Peng. Review on Community Detection Methods Based on Local Optimization [J]. Bulletin of Chinese Academy of Sciences, 2015(2): 238-247.

[6] Kumpula J M, Kivelä M, Kaski K, et al. Sequential algorithm for fast clique percolation [J]. Physical Review E Statistical Nonlinear & Soft Matter Physics, 2008, 78(2): 026109.

[7] Lancichinetti A, Fortunato S, Kertész J. Detecting the overlapping and hierarchical community structure of complex networks [J]. New Journal of Physics, 2012, 11(3): 19-44.

[8] Lee C, Reid F, McDaid A, et al. Detecting highly overlapping community structure by greedy clique expansion [J]. arXiv preprint arXiv: 10021827.

[9] Shen Huawei, Cheng Xueqi, Cai Kai, et al. Detect overlapping and hierarchical community structure in networks [J]. Physica A Statistical Mechanics & Its Applications, 2009, 388(8): 1706-1712.

[10] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks [J]. Journal of Statistical Mechanics, 2008, 2008(10): 155-168.

[11] Wang Xiaofeng, Liu Gongshen, Li Jianhua. Overlapping community detection based on structural centrality in complex networks [J]. IEEE Access, 2017, 5: 25258-25269.

[12] Qi Jinshan, Liang Xun, Wang Yi. Overlapping community detection algorithm based on selection of seed nodes [J]. Application Research of Computers, 2017, 34(12): 3534-3537.

Note: Figure translations are in progress. See original paper for figures.