Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 874217, 12 pages
http://dx.doi.org/10.1155/2014/874217

Research Article

An Efficient Hierarchy Algorithm for Community Detection in Complex Networks

Lili Zhang,[1] Qing Ye,[1] Yehong Shao,[2] Chenming Li,[1] and Hongmin Gao[1]

[1] *College of Computer and Information Engineering, Hohai University, Nanjing 211100, China*
[2] *Arts and Science, Ohio University Southern, Ironton, OH, USA*

Correspondence should be addressed to Lili Zhang; lilzhang@hhu.edu.cn

Received 31 March 2014; Revised 19 July 2014; Accepted 14 September 2014; Published 23 October 2014

Academic Editor: Ren-Jieh Kuo

Community structure is one of the most fundamental and important topology characteristics of complex networks. The research on community structure has wide applications and is very important for analyzing the topology structure, understanding the functions, finding the hidden properties, and forecasting the time-varying of the networks. This paper analyzes some related algorithms and proposes a new algorithm—CN agglomerative algorithm based on graph theory and the local connectedness of network to find communities in network. We show this algorithm is distributed and polynomial; meanwhile the simulations show it is accurate and fine-grained. Furthermore, we modify this algorithm to get one modified CN algorithm and apply it to dynamic complex networks, and the simulations also verify that the modified CN algorithm has high accuracy too.

1. Introduction

Today, there are many complex networks in our life, such as, the Internet, the WWW, electrical power grids, transportation, neural networks, metabolic networks, and research collaboration networks. These networks have complex structures, large number of nodes, and complex interactions among the nodes. The research on complex networks has become one of the most popular areas because of their applications in our life and the importance to other research areas. These complex networks share some concepts or research methods. And the study of the complex network provides insight into their structures, properties, and emergent behaviors [1–4]. Of particular interest are the rigorous methods for uncovering and understanding structure or subnetwork structure. For example, in social networks, the important structure is the social connection or community structure. Extracting community structure and leveraging them for predicting the emergent and critical casualties of large networks in a dynamic setting is of growing importance.

Community detection in complex networks has been the focus of many studies over the past eighty years [5]. With the development of technology, we are getting more and more data in the complex networks. Nowadays, community detection, especially in large and time-varying social networks, plays a very important role in real life. Community does not have a unique definition, but generally speaking, it consists of a group of nodes that are relatively densely connected to each other but sparsely connected to other dense groups in the network. Communities are often given different physical meanings in different networks. For example, community represents different research areas in the network of citations between papers; it represents different research community in the research collaboration network; it represents some people studying or working at the same university, and so forth. The research of community structure is to analyze network topology, to understand the function of networks, and to discover the hidden patterns in networks and predict the behavior of networks. Community structure also has broad applications. It has currently been used in terrorist organization detection, organizational structure management, network analysis [6], search engines [7–9], and many other fields.

With the development of technology, we are getting more and more data from all kinds of networks, such as Facebook, twitter, QQ, and Weixin (two kinds of social networks used in China). But to our best knowledge, the big data processing is

one challenge in many research fields nowadays, because the efficiency of previous algorithms can be not guaranteed with the increasing of the data and the networks. So, in complex networks, the efficient algorithm for community detection is also needed.

2. Related Work

The main algorithms to find community structure include traditional graph partitioning method, spectral bisection method, split method, clustering algorithm, and module optimization algorithm. These algorithms have the different assumptions and advantages when they applied in different occasions, so they have different applications in real life [10]. Next we will review and analyze some of the existing methods for community detection.

For convenience, we firstly assume the network as a graph $G = (V, E)$, where V is the node set which corresponds to objects in the network, and E is the edge set which corresponds to the relationship between nodes. Generally, based on the different applications, the networks can be modeled as undirected graphs, directed graphs, weighted graphs, or hypergraphs. This paper mainly uses an undirected graph to represent a complex network and find the community structure of the network efficiently.

2.1. The Kernighan-Lin Algorithm. The Kernighan-Lin algorithm, or the KL algorithm, proposed by Kernighan and Lin [11], originated from the graph partitioning algorithm, in other words, it is a graph partitioning method. Also, the KL algorithm is a heuristic optimization method, which divides the network into two given size communities, and the optimization goal is to minimize the difference of the number of connections between communities and the number within communities. Starting from the initial solution, it iterates until it cannot find any better candidate solution. The search strategy of the candidate solution is to move the nodes or exchange the nodes. So the solution it finds is often a local optimum rather than a global optimum. Meanwhile, we need to know the number and the size of existing communities in the network in advance, but it is hard to give the number and the size of the communities, and it is impossible in dynamic networks. So the KL algorithm is very sensitive to initial conditions and it is of low efficiency when applying it in some uncertain networks or dynamic networks.

2.2. Spectral Bisection Method. Spectral bisection method is also one of the famous graph partitioning algorithms, which has been applied to complex networks clustering analysis [12] in recent years. The idea of the spectral bisection method mainly applies quadratic optimization techniques to minimize the given "cut" function. And a partition with the minimum "cut" is considered to be optimal. But minimizing "cut" function is NP-complete [13]. Based on matrix theory, the spectral bisection method converts the minimum "cut" problem into quadratic optimization problem with constraints, and the approximate optimal solution of the quadratic optimization problems (i.e., the approximation of the optimal partition of the network) can be obtained by calculating the second smallest Laplace matrix eigenvectors of the network graph. Similar to the Kernighan-Lin algorithm, the spectral bisection method also partitions the network into two subnetworks. If we want a network to be divided into more, we need to apply the algorithm to subclusters; meanwhile, the spectral bisection method cannot automatically recognize the number of communities in the network. In other words, the spectral bisection method can be applied only to balanced networks. However, in reality, power law distribution is very common in complex networks, and there are many networks whose topology structures are unbalanced and complex. So graph partitioning methods have obvious limitation.

2.3. Divisive Algorithms. In 2002, Flake and other scholars firstly proposed the MFC clustering algorithm based on max-flow min-cut theorem [14] in graph theory. MFC is one of the frequently used divisive algorithms. It assumes that the network's maximum flow is determined by the network "bottleneck" capacity. By the maximum flow minimum cut theorem, we can get the community structure by simply calculating the minimum cut [13]. After that, Palla et al. proposed the GN algorithm in 2002 based on the similar method [6] and the difference is that the GN algorithm uses heuristic rules. Based on the assumption, the edge betweenness between the communities should be larger than that inside the communities. The GN algorithm yields a hierarchical clustering tree (dendrogram) in a top-down manner in the end.

Both MFC algorithm and GN algorithm are only applied in small and medium-sized networks because their time complexity. Besides that, neither of them gives a solution of how to obtain community structure of the networks from the obtained dendrogram finally.

2.4. Agglomerative Algorithm. As for the agglomerative algorithm, the main works are contributed by Newman. Due to the higher time complexity of the GN algorithm, Newman proposed a fast algorithm [15], which can be used for complex networks. The algorithm is greedy, and begins by initializing the network as n communities. The main process is to merge two of the communities by the modularity value. At last the communities will be merged into one, and then the community structure can be obtained by manually choosing different locations in the hierarchical clustering tree. It decreases the time complexity of the algorithm to $(O(m + n)n)$, but it is hard to get a good result if we want to get communities by computer from final hierarchical clustering tree, because the algorithm does not give the method of how to choose the suitable position in the hierarchical clustering tree for dividing the nodes into different communities.

2.5. The Analysis of the Previous Works. Following the analysis described above, we can see that all the algorithms except Newman's fast algorithm quantify the community structure by the number of edges. We use Figure 1 (where the small circles represent the nodes, and the large dot line circle means one cutting line between the node set S and the other nodes in

graph G) to demonstrate their same results if we only quantify the edges as the main strategy to find community structure.

In graph theory, an edge-cut is the set of the edges between two disjoint set of nodes. Therefore, the size of the edge-cut and the two disjoint node sets do not contribute to whether there is a community structure between the two node sets. Therefore, the size of the edge-cut is not a sufficient condition for the tightness of the two node sets. As shown in Figure 1, nodes 1, 2, and 3 form a complete graph, so the connectivity of the graph formed by nodes 1, 2, and 3 reaches highest with three nodes. But the existing algorithms will most likely divide node 1 and nodes 2 and 3 to different communities. Therefore, in addition to using the set of edges to quantify the community structure, we also need more criterions to improve the accuracy of the algorithm.

For the above reasons, we propose one hierarchy efficient algorithm for community detection which is different from the previous works in complex networks. Our algorithm not only considers the quantitative description of communities, but also considers the qualitative description based on the connectivity of the neighborhood of the node. The combination of them will make the algorithm more fine-grained. And it will work on unbalanced networks, so it is a universality. In the end, we will output the different community structures according to the different conditions on the different levels automatically, which keeps from the disadvantage of manually dividing the nodes into communities. Moreover, this process gives a better understanding of the generating process of communities and the relationship among all communities, which is helpful for understanding the evolution of the community structures.

3. CN Agglomerative Algorithm

3.1. Definitions and Notations. We use an undirected simple graph $G = (V, E)$ to model the network, so there is exactly one edge between any two nodes which have relationship. In this paper, we assume that G is connected, so there is at least one path between any two nodes, unless otherwise specified. For any two different nodes in the graph G with size n, if there is one edge between them, then G is a complete graph, denoted by k_n. For any node i, the degree of node i is the number of nodes (edges) connected to k, denoted by d_k, and the set of nodes connected to node k is called the neighborhood of node k, denoted by $N(k)$. Clearly, $d_k = |N(k)|$. If $H = (V_1, E_1)$ is a graph satisfying (1) $V_1 \subseteq V$ and (2) $E_1 \subseteq E$ then H is a subgraph of G. And if H is the subgraph of G, then the neighborhood of node k in H is defined as the intersection of H and the neighborhood of node k in G is denoted by $N_H(k)$.

Proposition 1 ($|N_H(k)| = |N(k) \cap V(H)|$). *By the definition of the adjacency matrix of simple undirected graph G, we can get the following result.*

Proposition 2. *Suppose that the adjacency matrix of G is* $A = [a_{ij}]$. *Then* $d_k = \sum_j a_{ij}$.

Divide the neighborhood of node i into two subsets, and one of them is the neighborhood of node i in the subgraph H, denoted by d_k^{in}, *the other is the neighborhood of node i in G-H, denoted by* d_k^{out}, *where G-H means the graph after deleting the nodes of H from G. Thus* $d_k^{in} = \sum_{j \in H} a_{ij}$, $d_k^{out} = \sum_{j \notin H} a_{ij}$ *and* $d_k^{in} + d_k^{out} = d_k$.

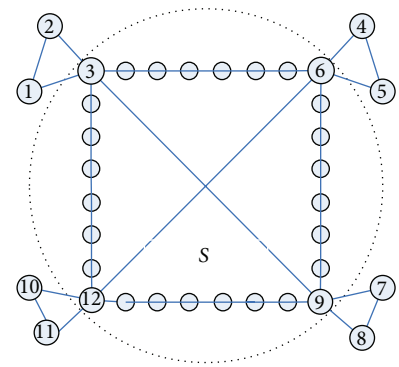


Figure 1: Example: graph G.

Although the communities in a network do not have widely accepted unique definition, generally, we can assume that community members (the nodes belonging to one community) are usually more tightly connected to its group than to other parts of the network. There were many suggestions about the quantitative definition of community structure in networks [15]. The following is one of the frequently used ones.

Definition 3. For any subgraph H of a graph G, if any node k of H satisfies $\sum_{k \in H} d_k^{\text{in}} > \sum_{k \in H} d_k^{\text{out}}$, then we say H is a community of G.

The description of community focuses on the connection relationship within the community or between them, but Definition 3 defines the community based on the in-degree and out-degree of some nodes in one subgraph. Figure 1 has shown the problem if we only consider the number of the edges between two node sets, so Definition 3 sometimes cannot show the tightness of connection relationship within one group and between groups. Then the accuracy and universality of the algorithm cannot be guaranteed if the algorithm designed is exactly based on Definition 3. Therefore, we propose an algorithm for community detection which not only considers the number of edges within the subgraph, but also considers the local connectedness in the graph.

Definition 4. If G has a sequence of nodes $v_1, v_2, \ldots, v_n$ (the subscripts are all distinct except the initial and the final nodes) such that v_i and v_{i+1} are adjacent for any i, then the sequence $v_1 v_2, \ldots, v_n$ is called a path of G; if $v_1 = v_n$, then the sequence $v_1 v_2, \ldots, v_n$ is called an n-cycle of G. A 3-cycle is called a triangle. If G has a k-cycle for any k ($3 \leq k \leq n$), then G is pancyclic. For any k ($3 \leq k \leq n$), if any node of G lies in a k-cycle, then G is vertex pancyclic.

Definition 5. Let $G = (V, E)$ be a graph and X, Y be two nonempty subsets of G, and any edge of G has one end node in X and the other in Y, then G is called a bipartite graph, denoted by $K_{m,n}$, where m, n denote the number of nodes in X and Y, respectively. If $m = 1, n > 2$, then the bipartite graph is called a star; if $n = 3$, then $K_{1,3}$ is called a claw.

Definition 6. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. If there exists a correspondence between V_1 and V_2, and under the correspondence, there is a correspondence between E_1 and E_2 as well, then G_1 and G_2 are isomorphic. If G does not contain any subgraph isomorphic to a claw, then G is claw-free.

Theorem 7 (see [16]). *Let $G = (V, E)$ be a connected claw-free graph. If the neighborhood of any node of G is a connected subgraph of G, then G is vertex pancyclic.*

From Definitions 4 and 6 and Theorem 7, if the node neighborhood forms a connected graph, the nodes must lie in a triangle when the graph is claw-free; conversely, if a node lies in a triangle, we can always choose one set such that the neighborhood of any node in the set induces a connected subgraph. Therefore, the connectedness of the neighborhood in one graph also implies the connection relationship between the node and its neighborhood and describes qualitatively community structure to some extent. For example, in any complete graph, the neighborhood of any node is connected. In Figure 1, when the distance between nodes is relatively large in the subgraph S, the community structure of the network is not obvious, or in other words, there exist some nodes that do not belong to any community, so the community structure is hierarchical and has a priority. But the previous works do not consider them. In this paper, we will consider the hierarchy and priority of the communities in the complex network based on the degree of the node and the connectivity of neighborhood. After that, we will give an agglomerative algorithm to find communities in complex networks. The algorithm will merge communities step by step based on the degree of nodes and connectedness of the neighborhood of the nodes, until all the communities merge into one.

Moreover, in our algorithm, we will output the different community structures by the different level conditions to reflect the hierarchy of the community structures. Because the largest difference of our algorithm is to consider the connectedness of the neighborhood of the nodes besides the in-degree and out-degree of the nodes within one group and between two groups, therefore the algorithm is referred to as the CN (connectedness of the neighborhood) agglomerative algorithm.

3.2. Algorithm and Analysis. Let $G = (V, E)$ be a graph model of a network and $G[H_1, H_2]$ be the induced graph of the edges with one end in H_1 and the other end in H_2. The main idea of our algorithm is given as follows.

(1) In the initialization, we divide the network into several communities, and each community is a complete subgraph. This will improve the efficiency of the algorithm. Of course, the worst case is that each community consists of a single node, as the graph exactly contains a node is also considered as a complete graph to some extent. It is rare to find a graph with community structure but does not have any complete subgraphs, so this step is helpful to reduce the time of algorithm. If we want to find more complete subgraph, we can use the output-sensitive algorithm [17] to find the first complete subgraph in graph G, and apply the algorithm to G-H; repeat the above step until the graph is empty. Because the output-sensitive algorithm is polynomial, so the time complexity of this step is polynomial as well.

(2) As for the hierarchy of our algorithm, it not only means we can get a hierarchy clustering tree eventually, but also means we can output the community structures on demand timely. We will get the community structures by quantitative description and qualitative description, respectively. Firstly, based on the degree condition we get some communities and the corresponding community structure; secondly, we will get more communities and the corresponding community structure by the connectedness between nodes and neighborhood and then output the community sets and community structures, respectively.

(3) In the end, the communities will be merged into one and get the hierarchy clustering tree.

Applying the above hierarchy method to get the communities and different community structures will show a simple evolution process of the community structures and the different community structures by the different conditions. We design our algorithm to output the communities in Steps 2 and 3 respectively and choose the optimum result of the actual community structure as needed. This avoids the inconvenience caused by manually choosing right positions like the GN algorithm or Newman fast algorithm. For example, without knowing the number of communities, the GN algorithm does not know when to stop deleting the edges that have the highest edge betweenness, and in the final hierarchical clustering tree achieved by Newman fast algorithm, in order to get the community structure it has to select the right positions and then disconnect it. Both of these algorithms using the manual method, different people may get different results, and the accuracy of the community analysis will be affected.

The algorithm for static complex networks will be given as follows in detail. First we give some notations and symbols used in our algorithm.

Generally, we use C_k ($k \in \{0, 1, 2, 3, \ldots\}$) to denote the community structure and H_i ($i \in \{1, 2, \ldots\}$) to denote one of communities in one community structure, which corresponds to one block in the adjacency matrix. Let d_v represent the degree of node v in graph G and d_v^{in} represent the degree of node i in one community H_i. It means the intersection of the neighborhood of node k and the community H_i. In the adjacency matrix, it is corresponding to the number of 1's which are located on the intersection of node k and the nodes of community H_i.

Let $|V(H_i)|$ denote the number of the nodes in community H_i. We refer to $|N(i) \cap V(H_i)|$, the number of neighborhood of node i in community H_i as $d_v^{\text{out}}[H_j]$ (Algorithm 1).

In our CN agglomerative algorithm, we use the complete subgraph as the core and origin of the communities. Then we use the degree of the node in each community as one

Input Adjacency matrix of the graph model
Output Communities, community structures, hierarchy clustering tree
(1) $C_0 \leftarrow \{H_1, H_2, \ldots\}, k \leftarrow 0$
(2) **while** $|C_k| > 1$
(3) perform a BFS traversal over the graph to compute for each vertex $v \in H_i$ (1) d_v^{in}, (2) $d_v^{\text{out}}[H_j]$, (3) $T(i, j)$, defined as $|\{v \in H_i \mid d_v^{\text{in}} \leq d_v^{\text{out}}[H_j]\}|$, (4) $M(i, j)$, defined as the minimum value of elements in $\{d_v^{\text{out}}[H_j]/d_v^{\text{in}} | v \in H_i, d_v^{\text{in}} \leq d_v^{\text{out}}[H_j]\}$
(4) find an i such that $T(i, j) \geq |V(H_i)|/2$ for some j
(5) **if** such i does not exist **then**
(6) **break;**
(7) **else**
(8) find j to maximize $\{M(i, j) | T(i, j) \geq |V(H_i)|/2\}$
(9) merge H_i into H_j
(10) let the resulting set of communities be C_{k+1}
(11) $k \leftarrow k + 1$
(12) **endif**
(13) **endwhile**
(14) $C_{\text{first}} \leftarrow C_k$ //the first level community structure
(15) **while** $|C_k| > 1$
(16) perform a BFS traversal over the graph to compute for each vertex $v \in H_i$ (1) d_v^{in}, (2) $d_v^{\text{out}}[H_j]$, (3) $T'(i, j)$, defined as $|\{v \in H_i \mid d_v^{\text{in}} \geq d_v^{\text{out}}[H_j] \geq (d_v^{\text{in}}/2) + 1\}|$, (4) $M'(i, j)$, defined as the minimum value of elements in $\{d_v^{\text{out}}[H_j]/d_v^{\text{in}} \mid v \in H_i, d_v^{\text{in}} > d_v^{\text{out}}[H_j] \geq (d_v^{\text{in}}/2) + 1\}$
(17) find an i such that $T'(i, j) < |V(H_i)|/2$ for some j
(18) **if** such i does not exist **then**
(19) **break;**
(20) **else**
(21) find j to maximize $\{M'(i, j) \mid T'(i, j) < |V(H_i)|/2\}$
(22) merge H_i into H_j
(23) let the resulting set of communities be C_{k+1}
(24) $k \leftarrow k + 1$
(25) **endif**
(26) **endwhile**
(27) $C_{\text{second}} \leftarrow C_k$ //the second level community structure
(28) **while** $|C_k| > 1$
(29) select two communities of C_k and merge them
(30) let the resulting set of communities be C_{k+1}
(31) $k \leftarrow k + 1$
(32) **endwhile**
//now $C_1, C_2, \ldots, C_k$ consist of the hierarchy clustering tree

Algorithm 1: CN agglomerative algorithm.

of the conditions to get bigger communities, line 4 in CN agglomerative algorithm gives the specific condition about the quantitative description. After that, line 17 gives the condition of qualitative description for communities. Therefore, we can get the different community structures by the different conditions.

Theorem 8. *The computational complexity of CN agglomerative algorithm is $O(mn)$, where m is number of edges and n is the number of vertices.*

Proof. Initially, C_0 contains at most n communities, so there are at most $n-1$ merging operations. Consequently, there are at most $n-1$ iterations of all the three while loops. Further note that each iteration of every while loop runs in $O(m)$ time. Hence, the complexity is $O(mn)$. □

4. Modified CN Algorithm for Dynamic Networks

The study of the community structure for a time-varying or dynamic network is still in its infancy. One of the reasons that hinder its development is the different controversial theories about communities, and lack of real world data reflecting the change between the time and the network [10]. There is lack of efficient algorithm to detect the communities in time-varying networks [18]. CN agglomerative algorithm can be modified to be applied in the time-varying networks. The CN agglomerative algorithm mainly finds the communities or community structures through the degree and the connectedness of the neighborhood of the nodes, so it is local and distributed, and the time complexity will not increase

Input Adjacency matrix of snapshot of the network
Output Community structures
(1) let G be the graph of the snapshot of the network
(2) apply CN agglomerative algorithm to graph G and get the
first level and second level community structures, respectively.
Let all the communities in the two community structure be one set
$C_0 = \{H_1, H_2, \ldots\}$
(3) compute the minimum degree of the node in each community
H_i, denoted by δ_{H_i}
(4) initialize a special set S to be an empty set
//When new node u is added to the network
(5) let $G = G[V(G) \cup u]$
(6) **if** there exists some i such that $|N(u) \cap V(H_i)| \geq \delta_{H_i}$ **then**
(7) among all such is, find the one that
maximizes $|N(u) \cap V(H_i)|$, let the found index be j
(8) insert node u into community H_j
(9) **else**
(10) insert node u into the special set S
(11) **if** $|S| > N$ // when more than N new nodes are inserted into S, where N is a user-provided threshold
then
(12) apply CN agglomerative algorithm to S, and get the
first level community structure and the second level community structure
(13) output the first level community structure and the second level community structure
(14) **endif**
(15) **endif**
(16) **if** new community appears
(17) output new community set
(18) **endif**

Algorithm 2: Modified CN algorithm.

as fast as the node number grows when applying it to time-varying network. The version of CN agglomerative algorithm applied in time-varying networks is called the modified CN algorithm, for convenience.

First of all, we can consider the network as static network when the time is fixed, so we can apply CN agglomerative algorithm in the static network to get the initial community structures on the two different levels.

Furthermore, we will merge the new nodes and edges to some communities and get new community structures with the networks changing. When a new node is added, we need to find the community that the new node may belong to. In the modified CN algorithm, we also firstly use the in-degree and out-degree of the new node in community detection, and then use the connectedness of the neighborhood of the new node in different communities.

Because the time-varying network frequently changes, and it is good to know the community structure at any time and evolution of the community structures, we will need to apply CN agglomerative algorithm to the snapshot of the network firstly, and then find the evolution of community structure by the modified CN algorithm (Algorithm 2).

Theorem 9. *When a new node is added into the network, modified CN agglomerative algorithm runs in either $O(n)$ time or $O(mn)$ time, depending on the current network structure, where m is the number of edges and n is the number of vertices.*

Proof. There are three cases when a new node is added: (1) Lines 7-8 are executed; (2) Line 10 is executed; (3) Lines 10, 12, and 13 are executed. In the first two cases, the algorithm terminates in $O(n)$ time, since there are at most n communities. For the third case, CN agglomerative algorithm is called, so we have a running time of $O(mn)$. □

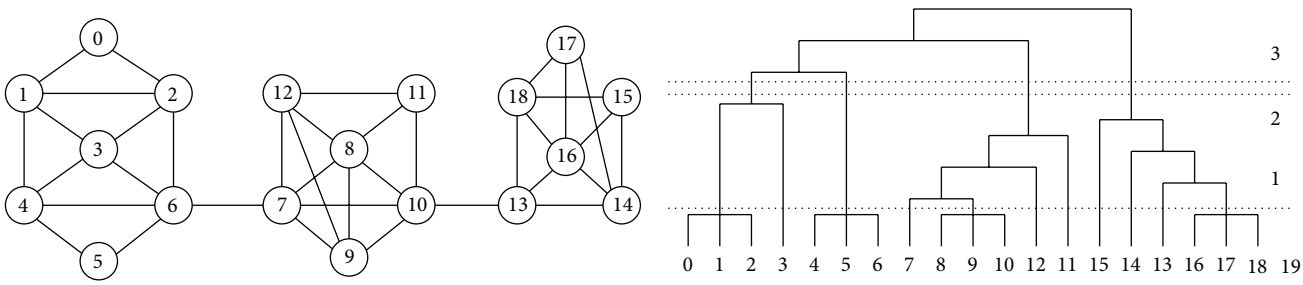
5. Simulation and Evaluation

The experimental environment for the simulation in this section uses the Intel Core 2 Duo 2.26 GHz processor, 2 GB 1067 MHz DDR3 memory, 160 G hard drive, Mac OS X operating system, C + + 4.2.1 compiler, and Cytoscape drawing software.

5.1. Simulation on CN Agglomerative Algorithm. First of all, we apply the CN agglomerative algorithm to the classic graph G [19] and graph H. The results are shown in Figures 2 and 3, where level 1 in the hierarchical clustering tree indicates the initialization of the corresponding community structure, and level 2 and level 3 show the corresponding community structures after the implementation of Step 2 and Step 3.

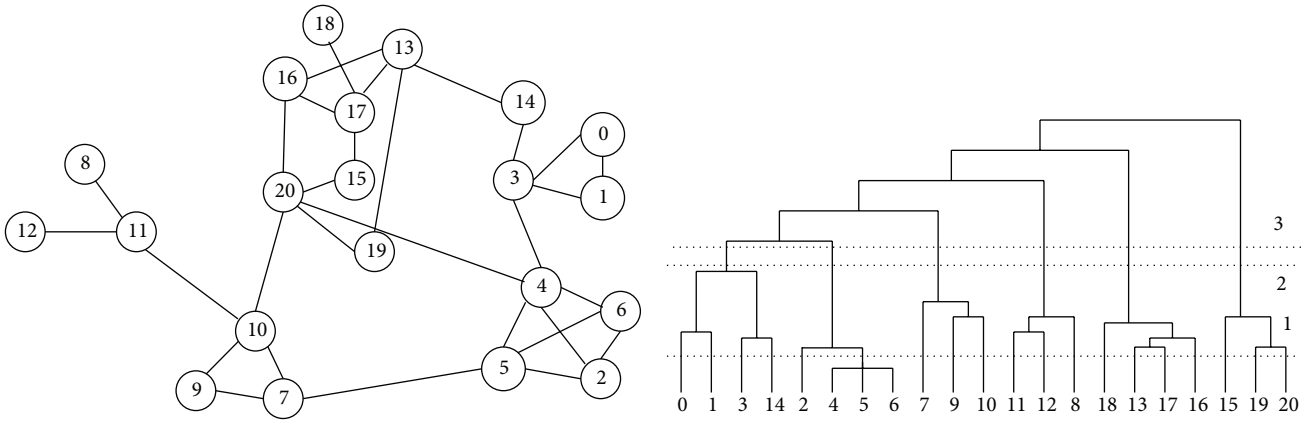
From Figures 2 and 3, the graph G and graph H both have clear community structures, because the first level community structure is the same as the second level community structure.

For graph G, we compared the efficiency of the different algorithms and the results are shown in Figure 4. Nodes



(a) Graph G (b) Community structure of graph G

Figure 2: Graph G and its community structure got by CN agglomerative algorithm.



(a) Graph H (b) Community structure of graph H

Figure 3: Graph H and its community structure got by CN agglomerative algorithm.

within a big circle belong to the same community; there are different community structures applying different algorithms on the graph G. From Figure 4, we can see that the CN agglomerative algorithm is both accurate and fine-grained compared to the other algorithms.

Furthermore, in order to verify the universality of our algorithm, we test the following random network, in which each community random network is defined as RN (C, s, d, pin), where C is the number of network communities, s is the number of nodes, and d is the average degree of nodes in the network. The higher the pin value is, the more obvious the community structure of the random network is. Otherwise, the fuzzier the community structure is. In particular, when pin <0.5, the random network does not have community structures. Figure 5 shows the topology structure of the random graph with p decreasing. The simulation results on the accuracy of CN agglomerative algorithm and the community structures of network are shown in Figures 6 and 7, respectively, where p represents pin value. Figure 6 shows that the accuracy of the CN agglomerative algorithm is always high although p takes different values. In Figure 7, the number in the rectangle represents the size of the community; the first column shows the first level community structure, and the second column shows the second level community structure.

As the community structure is still not universally defined, the calculation of the accuracy has inevitable errors, but from Figure 7, we can see that the CN agglomerative algorithm is fine-grained compared to the graph in Figure 5, especially when $p = 0.6$. When $p = 0.6$, we can see that the topology structure of the random graph is becoming more random, and the community structure is becoming not obvious. But in Figure 7, we can find the difference between the first level community structure and the second level community structure when $p = 0.6$; it means our algorithm can still work when the network topology is random. Meanwhile, applying the different community structure, we can predict the evolution of the communities.

Moreover, in order to verify the accuracy of our algorithm in the real network, we collect data from QQ users. Let G' denote this real network, and the size of G' is $n_1 = 210$. Figure 8 shows the topology structure of G'. The ellipses in Figure 8 represent the communities according to the real network, and the letters in ellipses mean the name of the communities. According to the topology structure of G', we know that $|V(H_1)| = 37$, $|V(H_2)| = 42$, $|V(H_3)| = 31$, $|V(H_4)| = 52$, and $|V(H_5)| = 48$, respectively. Applying CN agglomerative algorithm on graph G', there are two people in H_1 that are divided into H_2, others are in the same community as the real. And the first level community structure is the same as the second level community structure, so it shows

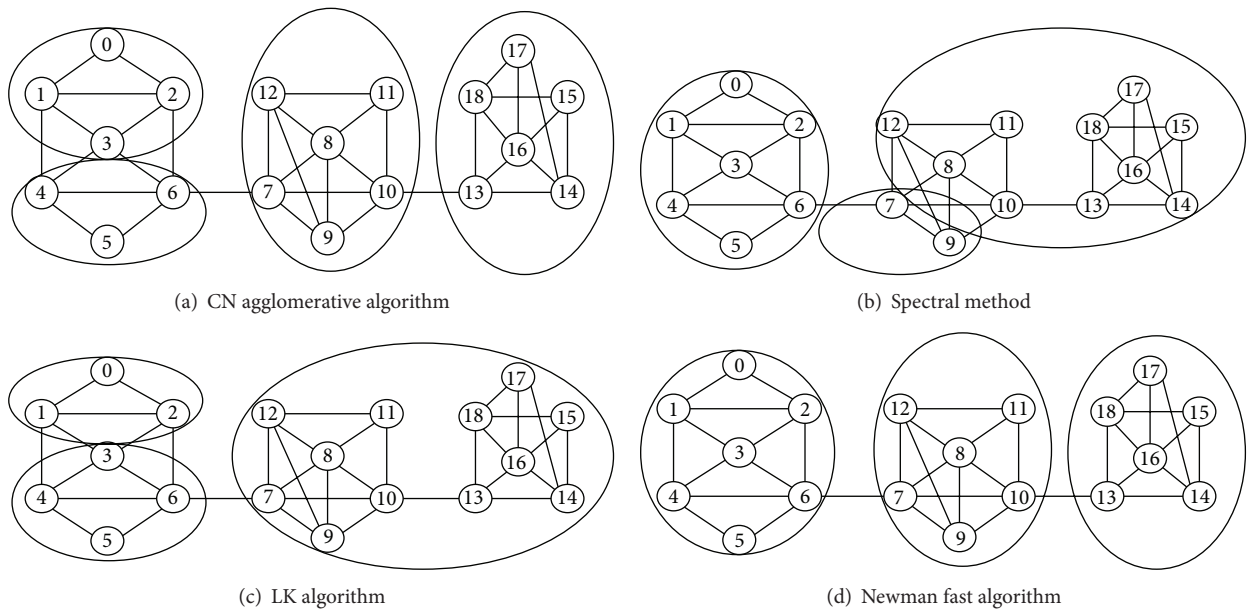


Figure 4: Comparison of algorithms on graph G.

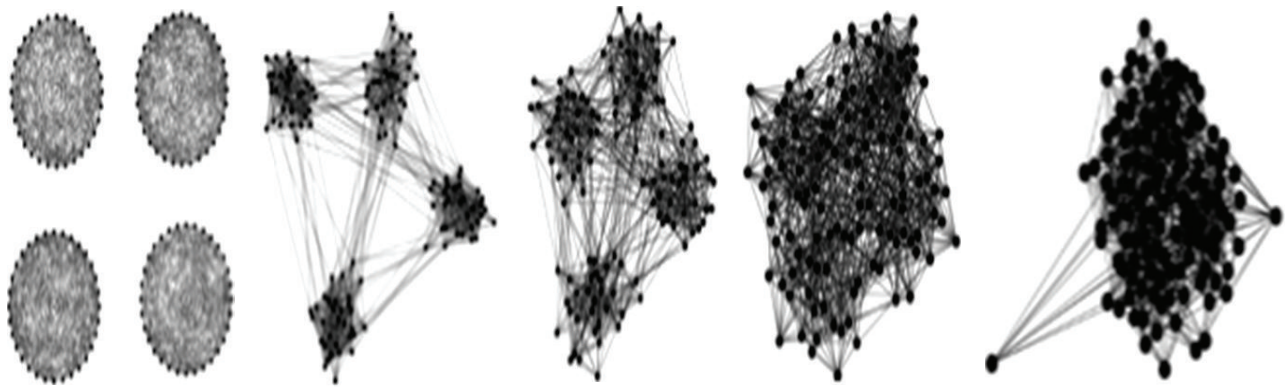


Figure 5: Random graphs with different p.

that graph G' has the obvious community structure. The simulation results of CN agglomerative algorithm and the accuracy of each community are shown in Figures 9 and 10. In Figure 9, letter A means the real network, CN means the simulation results.

5.2. Simulation on Modified CN Algorithm. In this section, we apply the modified CN algorithm to growing random network RN (C, s, d, pin), which is one classic kind of time-varying network, and the parameters have the same meaning as the static random network.

In the growing random graphs, the pin value remains unchanged when the network grows, as shown in Figure 11, where pin = 0.9. For convenience, we denote pin as p, where the network growth process of the network topology increases from 80 nodes to 128 nodes.

The simulation topology in Figure 11 shows the final community structure in a network of 128 nodes. This section simulates the searching process of community structure in order to keep the accuracy of the community structure in network growth process. The simulation results are shown in Figures 12 and 13. Figure 12 shows the accuracy of the algorithm when applied to the growing random networks, and Figure 13 shows the change of the community structure in the algorithm implementation process, where the number of nodes in rectangle in Figure 13 represents the size of the corresponding communities, and the columns correspond the community structures of the random graphs with nodes 80, 96, 112, and 128 orderly; the number in the rectangle means the size of the community.

Finally, we apply the modified CN algorithm to the time-varying network of G'. Let $T_1 = G'$ denote the initial time of network, and we know that $n_1 = 210$. With the time

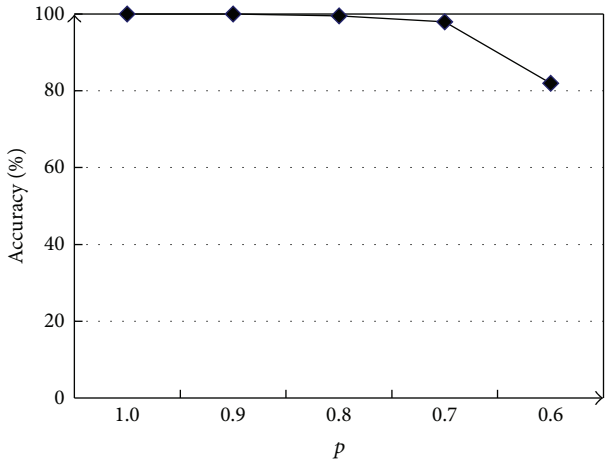


Figure 6: The accuracy of CN agglomerative algorithm to find the communities on random graph with different p.

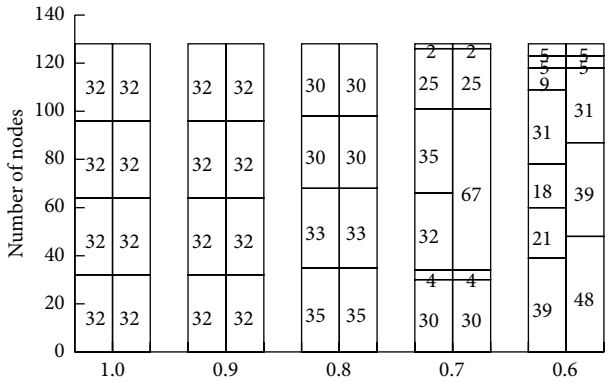


Figure 7: Community structures of the random graph with different p got by CN agglomerative algorithm.

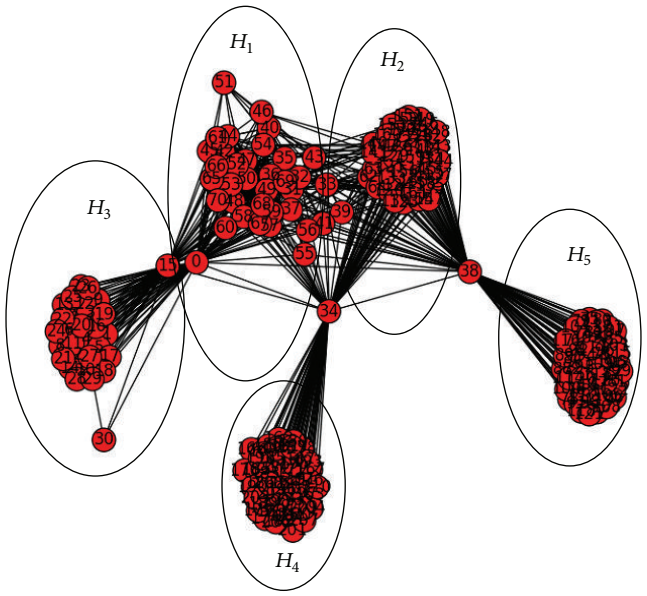


Figure 8: The topology structure of G'.

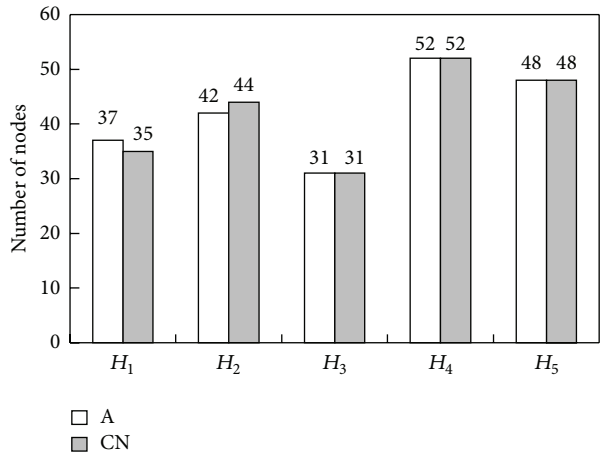


Figure 9: The comparison of community structure of G' between our algorithm and the real network.

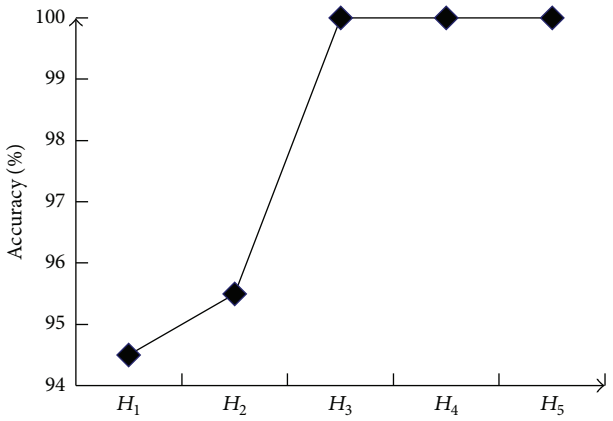
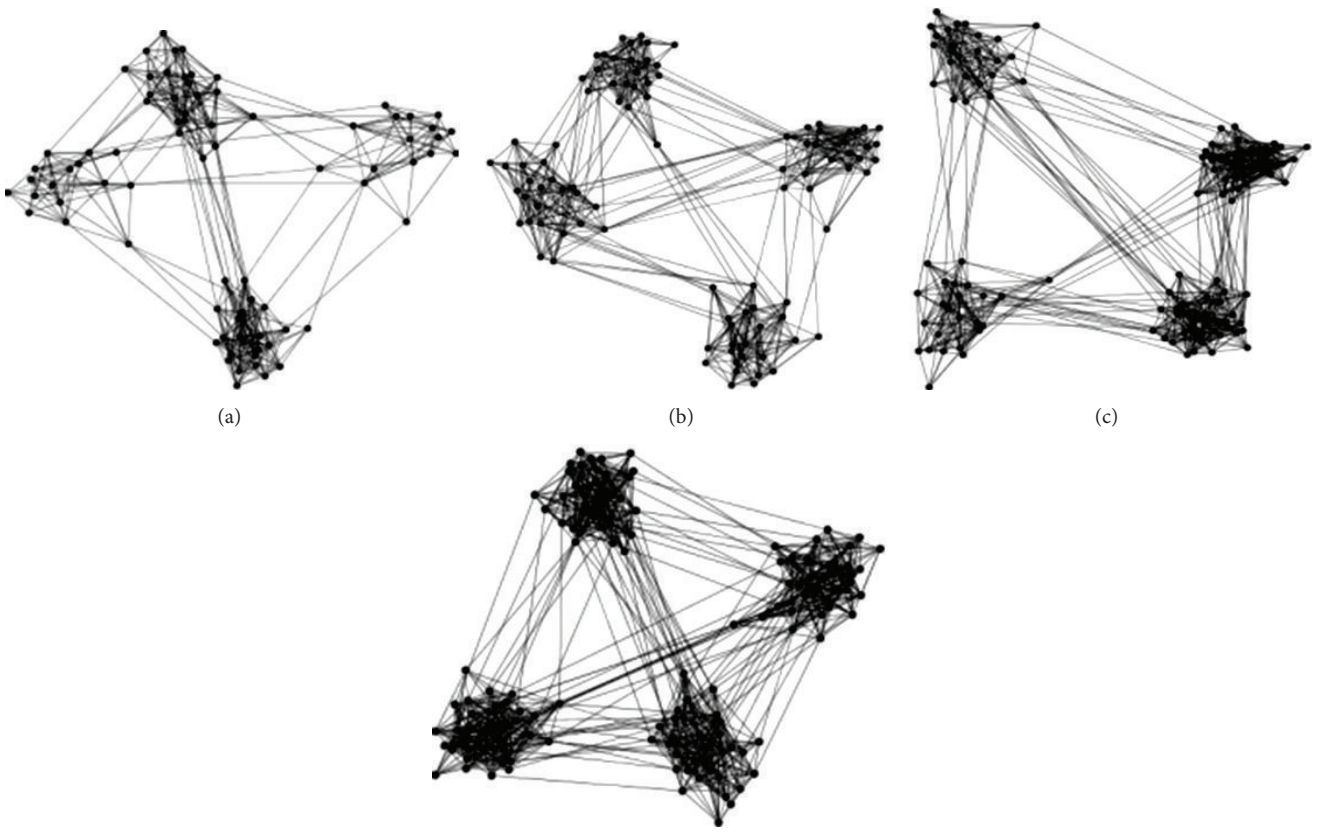


Figure 10: The accuracy of CN agglomerative algorithm on each community.

growing, the data we collected shows that $n_2 = 223$ at time T_2, $n_3 = 230$ at time T_3, and $n_4 = 236$ at time T_4. The community structure of the time-varying network of G' according to the actual situation and the community structure of the growing G' got by the modified CN algorithm are shown in Figure 14. The first column shows the actual situation, the second column shows the community structure got by the modified CN algorithm, and the number in the rectangle means the size of the community. Figure 15 shows the accuracy of the modified CN algorithm with the time growing, and Table 1 shows the results of the modified CN algorithm, the column of A in Table 1 means the actual community structure, and CN means the simulation results. At the same time, if the number of nodes will reduce with the time growing, we can get the snapshot of the network and then output the communities again. Figures 16 and 17 show the results of our algorithm when the nodes disappeared. Every time we removed ten nodes and the corresponding edges randomly, the first level community structure is the same as the second level community structure.



(a) (b) (c) (d)

Figure 11: The network growth process when $p = 0.9$.

Table 1: The results of the modified CN algorithm.

Time	T_1		T_2		T_3		T_4	
n	$n_1 = 210$		$n_2 = 223$		$n_3 = 230$		$n_4 = 236$	
	A	CN	A	CN	A	CN	A	CN
Community structures	37	35	37	36	37	37	37	35
	42	44	42	43	42	42	44	46
	31	31	39	39	39	39	40	40
	52	52	57	57	57	57	60	60
	48	48	48	48	51	51	51	51
	/	/	/	/	4	4	4	4
Accuracy	99.04%		99.5%		100%		99.1%	

From Figures 10, 15, and 17 and Table 1, it can be seen that our algorithm can still keep high accuracy and good performance when applied on the real networks.

6. Conclusions

A good algorithm must be able to determine the existence of community structures in a graph and determine the hierarchy of the communities. However, the existing community detection methods in complex networks have some limitations [20], as most of the algorithms only consider quantitative description of the communities when designing the algorithm. In this paper, the CN agglomerative algorithm proposed considers both the quantitative and qualitative properties of the communities, and we show that our algorithm runs in polynomial time. Furthermore, the simulations also show that our algorithm is highly accurate and fine-grained. On the other hand, we proposed the modified CN algorithm for time-varying networks to find the community structure, which is a new idea for the design of time-varying network community detection algorithm. In addition, our algorithm can output the different community structures by

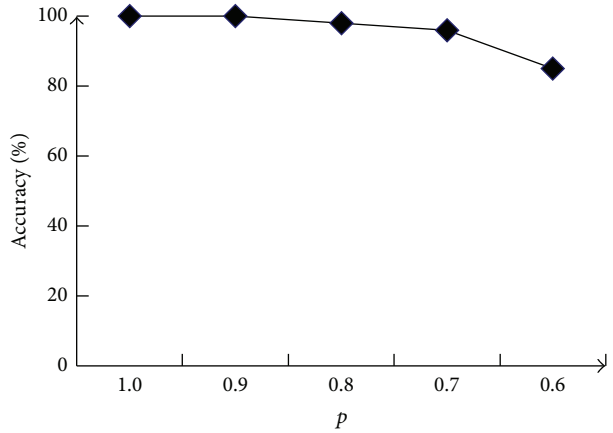


Figure 12: Accuracy of modified CN algorithm on growing random graph with different p.

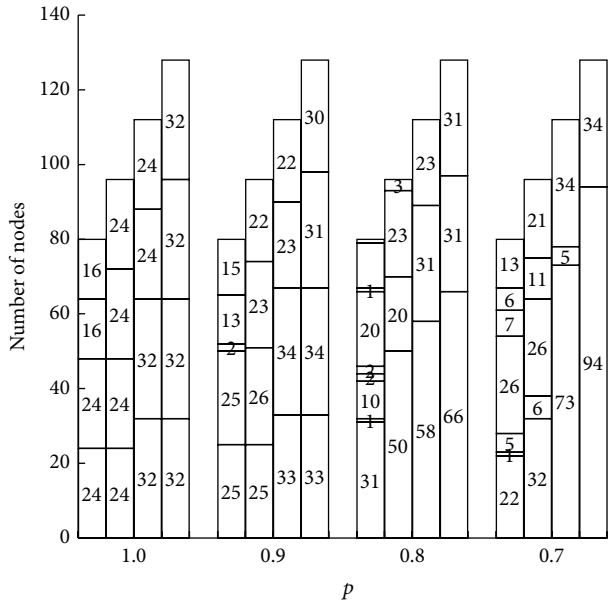


Figure 13: Community structure of the growing random graph got by the modified CN algorithm with different p.

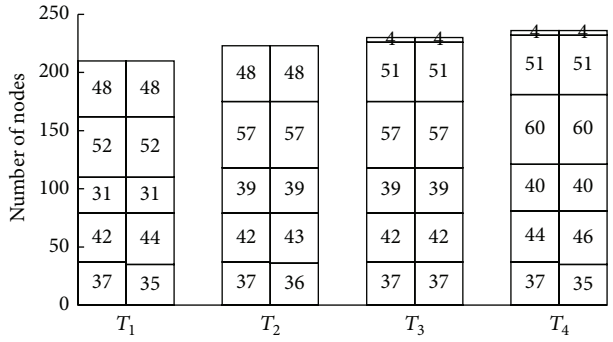


Figure 14: The comparison of community structure of the growing G' between our algorithm and the real network.

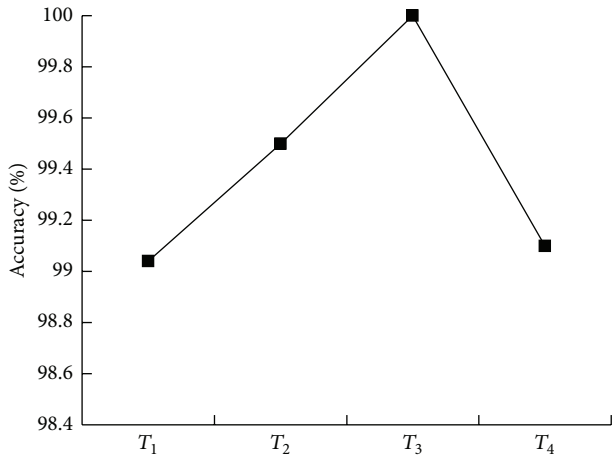


Figure 15: The accuracy of the modified CN algorithm.

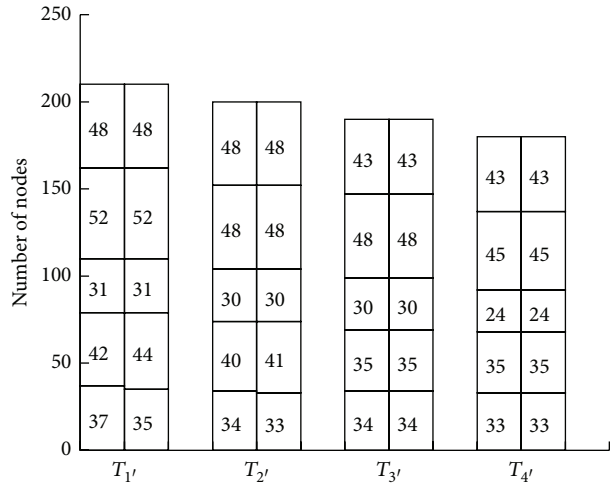


Figure 16: The comparison of community structure between our algorithm and the real network.

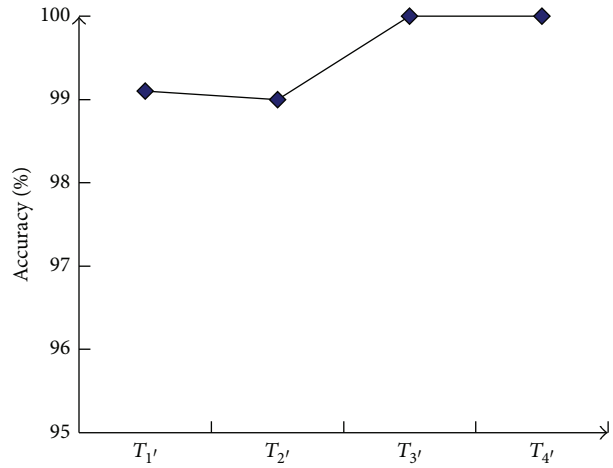


Figure 17: The accuracy of our algorithm.

the different level conditions, which can better show the evolution of the community structure and the hierarchy between the different communities.

More and more complex and dynamic networks have appeared, such as Facebook, Twitter, Weixin, and QQ. But the community detection algorithms mainly focus on static networks, so the community detection algorithm on time-varying network is urgent. In addition, finding the overlapping community structure in networks is also a popular research field. In fact, there is no unified quantitative definition for overlapping community [10]. Therefore, there are still a lot of open questions in community detection of complex networks. The CN agglomerative algorithm uses the node's neighbor information to determine the community structure, therefore, the overlap between the different communities can be converted into a problem having the same node belonging to two or more than two different communities. For the overlap of the communities in networks, how to better quantify the relationship between the nodes and the communities is very important. The authors will continue to apply the idea of the CN agglomerative algorithm to overlap community in complex networks in the future.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is partially supported by the China NSF Grant (61003224) and Innovation group project Grant (61021062).

References

[1] A. L. Barabási and E. Bonabeau, "Scale-freenetworks," *Scientic American*, vol. 288, no. 5, p. 60, 2003.

[2] A. L. Barabási and R. E. Crandall, "Linked: the new science of networks," *American Journal of Physics*, vol. 71, p. 409, 2003.

[3] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[4] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, 2006.

[5] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, Cambridge University Press, Cambridge, UK, 1994.

[6] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[7] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[8] R. B. Almeida and V. A. F. Almeida, "A community-aware search engine," in *Proceedings of the 13th Conference on the International World Wide Web Conference*, S. I. Feldman, M. Uretsky, M. Najork, and C. E. Wills, Eds., pp. 413–421, ACM, New York, NY, USA, 2004.

[9] A. Sidiropoulos, G. Pallis, D. Katsaros, K. Stamos, A. Vakali, and Y. Manolopoulos, "Prefetching in content distribution networks via Web communities identification and outsourcing," *World Wide Web*, vol. 11, no. 1, pp. 39–70, 2008.

[10] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[11] B. W. Kernighan and S. Lin, "A efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, pp. 291–307, 1970.

[12] L. Donetti and M. A. Munoz, "Improved spectral algorithm for the detection of network communities," in *Proceedings of the 8th International Conference on Modeling Cooperative Behavior in the Social Sciences*, P. L. Garrido, M. A. Muñoz, and J. Marro, Eds., vol. 779, pp. 104–107, American Institute of Physics, New York, NY, USA, 2005.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York, NY, USA, 1990.

[14] A. V. Goldberg, "Recent developments in maximum flow algorithms," in *Algorithm Theory—SWAT'98*, S. Arnborg and L. Ivansson, Eds., vol. 1432 of *Lecture Notes in Computer Science*, pp. 1–10, Springer, Berlin, Germany, 1998.

[15] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, Article ID 066133, 2004.

[16] C. Q. Zhang, "Cycles of given length in some $K_{1,3}$-free graphs," *Discrete Mathematics*, vol. 78, no. 3, pp. 307–313, 1989.

[17] K. Makino and T. Uno, "New algorithms for enumerating all maximal cliques," in *Algorithm Theory: SWAT 2004*, vol. 3111 of *Lecture Notes in Computer Science*, pp. 260–272, 2004.

[18] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha, "Robust detection of dynamic community structure in networks," *Chaos*, vol. 23, no. 1, Article ID 013142, 2013.

[19] A. Capocci, V. Sevedio, G. Caldarelli, and F. Colaiori, "Detecting communities in large networks," in *Algorithms and Models for the Web-Graph*, vol. 3243 of *Lecture Notes in Computer Science*, pp. 181–187, Springer, Berlin, Germany, 2004.

[20] B. Yang, D.-Y. Liu, J. Liu, D. Jin, and H.-B. Ma, "Complex network clustering algorithms," *Journal of Software*, vol. 20, no. 1, pp. 54–66, 2009.