symmetry

MDPI

Article

Algorithm for Detecting Communities in Complex Networks Based on Hadoop

Mo Hai, Haifeng Li *, Zhekun Ma and Xiaomei Gao

School of Information, Central University of Finance and Economics, Beijing 100081, China; haimo@cufe.edu.cn (M.H.); mzk@cufe.edu.cn (Z.M.); gxm@cufe.edu.cn (X.G.)

* Correspondence: mydlhf@cufe.edu.cn

Received: 9 October 2019; Accepted: 6 November 2019; Published: 7 November 2019

Abstract: With the explosive growth of the scale of complex networks, the existing community detection algorithms are unable to meet the needs of rapid analysis of the community structure in complex networks. A new algorithm for detecting communities in complex networks based on the Hadoop platform (called Community Detection on Hadoop (CDOH)) is proposed in this paper. Based on the basic idea of modularity increment, our algorithm implements parallel merging and accomplishes a fast and accurate detection of the community structure in complex networks. Our extensive experimental results on three real datasets of complex networks demonstrate that the CDOH algorithm can improve the efficiency of the current memory-based community detection algorithms significantly without affecting the accuracy of the community detection.

Keywords: community detection; complex networks; Hadoop; modularity increment

1. Introduction

In the era of Web 2.0, objects are connected to each other by various technologies such as the Internet and the Internet of Things, and form a variety of complex networks such as interpersonal interaction, essay reference, transportation, and protein interaction networks. Various complex networks are widely used in sociology, management, computer science, operations, biology, and other disciplines, while their wide application prospects have attracted the interest of many researchers. For example, Watts and Strogatz [1] applied the complex network theory in the field of biology and considered the nervous system to be a complex network of large numbers of nerve cells connected by nerve fibers. Faloutsos [2] applied the method of complex network analysis to study computer networks and evaluated their stability by analyzing their robustness. Sen et al. [3] mapped the transportation network to a complex network and implemented an optimal planning and configuration of the transportation network using dynamic analysis of the complex network. Xiao et al. [4] constructed a directed and weighted complex network based on the Beijing traffic network, analyzed the traffic network load-bearing pressure, and mined the corresponding regional centers, which provided a theoretical support for optimizing urban public transport network systems. Based on the characteristic analysis of the complex network itself, Ruguo [5] proposed a method for social coordination governance and provided ideas for solving mass public events based on a characteristic analysis of complex networks.

Many studies analyzed the inherent characteristics of complex networks and discovered the relationships between node attributes and connections within networks. To discover feature of complex networks, several community detection algorithms have been proposed. The so-called "community" is a sub-network composed of a group of nodes closely connected with their internal nodes and sparsely connected with other external community nodes. The community structure is a common feature of complex networks made up of one or more communities. The accurate identification of

the community structure in complex networks play an important theoretical role for public opinion monitoring, interest recommendation, identification of the network internal structure and other related research. As a result, many researchers have studied community detection algorithms from the aspects of modularity and edge structure. For examples, Newman and Girvan [6] proposed the concept of modularity and mined the complex network community structure, Yang et al. [7] introduced a method for analyzing the edge structure and node properties allowing to improve the accuracy of the detection of the complex network community structure. The accurate identification of the community structure in complex networks have broad applications, such as influence maximization, influences discovery within a community, interest recommendation, edge intelligence empowered recommendation [8], and so on.

However, the existing studies about complex network community detection algorithms focused on small-scale data sets and limited to the improvement of the community detection accuracy while neglecting its efficiency. At the same time, the number of nodes in complex networks demonstrates an explosive growth trend considering the advent of big data era, increasing number of network users, and exponential increase of the generated contents. At present, many social networking platforms such as WeChat, Weibo, Facebook, and Twitter, have more than 100 million on-line users and various interaction forms, including follow-ups, comments, and sharing. The large-scale complex network data sets generated by such platforms have the characteristics of node diversity, complex structure, multi-complexity fusion, which challenges the accuracy of the traditional complex network community detection algorithms. Furthermore, the traditional community detection algorithms are based on matrix iterations, which make the algorithms unable to adapt to the requirements of real-time and flexibility.

In this paper, we propose a new complex network community detection algorithm based on Hadoop framework (called Community Detection on Hadoop (CDOH)). Hadoop is a distributed system infrastructure developed by the Apache Foundation. Our contributions are as follows:

- Based on the idea of the maximum modularity, and combining the distributed characteristics of the Hadoop platform, a new modularity matrix update method is proposed and a corresponding community merging strategy is constructed to implement a fast and accurate detection and discovery of complex network community structures;
- We theoretically analyze our proposed CDOH algorithm, and show the computational cost of our algorithm can achieve $O(n)$ computational cost when we use enough parallel nodes;
- Experimental results on 3 real datasets demonstrate that CDOH significantly outperforms the traditional complex network community detection algorithm in terms of both the efficiency and accuracy of the community detection of complex networks.

The rest of our paper are organized as follows. Section 2 introduces the related works. Section 3 describes our proposed CDOH algorithm and analyzed its computational complexity. In Section 4, we show the experimental results with theoretical analysis. Section 5 concludes the paper and presents the future works.

2. Related Works

Since Newman [6,9] proposed the module optimality algorithm, the modularity-based community detection approach has been used in many network community mining algorithms such as the classic fast Newman community division algorithm [9] and CNM algorithm [10]. The fast Newman community detection algorithm is an agglomerative hierarchical clustering algorithm that starts with a state, in which each node is the sole member of n communities, and repeatedly joins communities together in pairs, choosing a joint at each step, which results in the greatest increase (or smallest decrease) in modularity. Recently, domestic researchers such as Lei et al. [11] implemented an edge community mining algorithm based on the local information of the considering network. Xiong [12] proposed a community discovery algorithm that combined the user closeness with clustering algorithms. Weiping [13] proposed the concept of new gravity of users for an accurate

community discovery; Leng [14] proposed a new network community detection algorithm based on a greedy optimization technology. Zhang et al. [15] further improved the fast Newman algorithm by introducing an improved index for the closeness centrality to classify overlapping nodes; the proposed method demonstrated a high classification accuracy in detecting overlapping communities with a time complexity of $O(n^2)$.

Blondel et al. [16] improved the modular incremental solution method by merging communities iteratively using a new calculation formula to achieve good results. Parsa et al. [17] used a probability vector model based on a single variable edge distribution algorithm, that combines an evolutionary algorithm with a community discovery method to enable the community detection; Oliverira et al. [18] used an improved Kuramoto coupled oscillator synchronization model to analyze networks from their dynamic factors and implemented a method for community discovery in complex networks. Ling Xing et al. [19] proposed a method that combines the sliding time-window method with the hierarchical encounter model based on association rules to increase the fidelity of the extracted networks by alleviating the homophily effect. Yuhui Gong et al. [20] focused on the customers' conformity behaviors in a symmetry market where customers are located in a social network. Simulation results have shown that topology structure, network size, and initial market share have significant effects on the evolution of customers conformity behaviors. Recently, Aceto et al. [21,22] and Ruoyu Wang et al. [23] applied deep learning and machine learning technologies in the research about social networking.

Recently, researchers have proposed complex network community detection algorithms based on big data platforms. Clauset [24] proposed a community-based parallel detection method based on the CNM algorithm. The basic idea of the algorithm proposed in [24] is to calculate the maximum community modularity in parallel and recognize the communities of large-scale networks by decreasing the communication overhead. The limitation of this algorithm is that it fails to run when the network scale increases and the amount of data rises to a certain level. Jinpeng [25] proposed a link community recognition algorithm based on the Hadoop platform. While this algorithm resolves the limitation of the linked community method that cannot store and process large matrices when analyzing big networks, its efficiency is still not efficient enough. Furthermore, its processing time reaches more than 5000 seconds when the scale of nodes reaches 15,000. Riedy et al. [26] used servers with multi-core processors to calculate the maximum community modularity in parallel to identify communities. However, the proposed method has strong hardware dependencies.

Moon et al. [27] proposed a parallel GN algorithm [6] based on Hadoop that can be divided into 4 stages. Each stage includes the map and reduce process. In the first stage, the tuples of all node pairs are generated; in the second and third stages, the edges with large edge betweenness values are identified and removed, respectively; in the fourth stage, the tuples are recalculated according to the new network. The experiment results demonstrated that the efficiency of the algorithm increases linearly with the increase of the number of reducers which are in charge of reduce process. Weijiang et al. [28] proposed a parallel Louvain algorithm that solved the main time-consuming problem of calculating the modularity and ergodic modularity increment in the Louvain algorithm [29]. This proposed algorithm outputs the information about all neighbors of a node in the map phase and decides the new home community of the node in the reduce phase accordingly. When computing a new community of a node, it is necessary to ensure that the neighbor's community is up-to-date, which is hard to be guaranteed in a distributed environment. Therefore, it is easy to face the problems of "community interchange" and "community ownership delay," which can be solved by resolving the associated connected graph. To solve the problem of high complexity of the fast Newman algorithm [10] in calculating the modularity of nodes. Bingzhou [30] proposed a parallel fast Newman algorithm based on Hadoop that calculates the modularity increment of each node merged with its neighbors in the map stage in parallel. In the reduce stage, the 2 nodes with the largest modularity increment are found and merged. The map and reduce processes are executed iteratively until all nodes are merged into 1 community. To deal with the problems of the fast-unfolding algorithm in processing large-scale networks. Bingzhou [30] also proposed a parallel fast-unfolding algorithm based on Hadoop and the

divide and conquer principle. First, a large-scale network is partitioned and merged separately, then the network is reconstructed according to the merging results of each partition, and finally the network is merged iteratively and reconstructed until the structure of community does not change any more. Conte et al. [31] proposed an algorithm which was able to find large k-plexes of very large graphs in just a few minutes and scale up to tens of machines with tens of cores each. Vincenzo el al. [32] proposed a novel algorithm for community detection in social networks based on game theory, and showed this algorithm outperformed other algorithms in terms of computational complexity and effectiveness. However, this algorithm cannot scale to a huge number of nodes and edges.

The traditional community detection algorithms focused on small-scale data sets and hard to scale to a large scale data sets. While parallel community detection algorithms are more scalable, they cannot achieve a good trade-off between the efficiency and accuracy. In order to overcome the shortcomings of traditional community detection algorithms and parallel community detection algorithms, we propose a new complex network community detection algorithm based on Hadoop, which effectively implements a fast and accurate detection of complex network community structure. Compared with traditional community detection algorithms, it can scale to a large scale data set. Compared with parallel community detection algorithms, it achieves a good trade-off between efficiency and accuracy.

3. Complex Network Community Detecting Algorithm Based on Hadoop

The proposed CDOH algorithm is based on the idea of the maximal modularity increment, which employs a new modularity matrix updating method and a community merging strategy.

3.1. Definitions

This section provides formal definition of the basic concepts involved in the proposed complex network community detection algorithm. The symbols and their meanings are shown in Table 1.

Table 1. Symbols and Definitions.

Symbols	Meanings
N	A complex network
V	a set of nodes
v_i	node i
E	a set of edges
e_{ij}	Denotes the connection between node v_i and node v_j, if they are connected, e_{ij} is 1; Otherwise e_{ij} is 0.
d_i	the node degree of node v_i
M	the modularity of a network
C	the set of detected network communities
c_i	a community i
l_c	the total number of edges interconnected between nodes within the community c
m	the total number of edges in the network
D_c	the sum of the node degrees of all nodes in the community c
a_c	The ratio of the sum of degrees of all nodes in the community c to the sum of degrees of all nodes in N
$\triangle M$	the modularity increment
R_{ij}	the number of connection edges between communities c_i and c_j

Definition 1. *(Complex network) A complex network is a network consisting of a series of nodes and their interconnected edges denoted as* $N = (V, E)$. *Here,* $V = \{v_i \mid i = 1, 2, \cdots, n\}$ *represents a set of nodes in a complex network, and* $E = \{e_{ij} \mid v_i, v_j \in V\}$ *represents a set of edges in a complex network, where* e_{ij} *denotes the connection between nodes* v_i *and* v_j. *If they are connected, then* $e_{ij} = 1$; *otherwise,* $e_{ij} = 0$.

Definition 2. *(Node degree) In a complex network $N = (V, E)$, the node degree d_i of each node v_i is defined as the number of edges connected to node v_i, which is defined by Equation (1),*

$$d_i = \sum_{v_j \in V, i \neq j} e_{ij} \tag{1}$$

Figure 1 illustrates a simple network community structure. According to Definitions 1 and 2, there are 12 nodes in the network (from v_1 to v_2), where $e_{12} = 1$, $e_{19} = 0$, and v_1 has a node degree $d_1 = 4$.

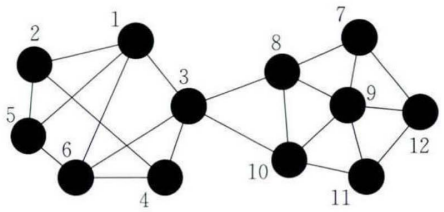

Figure 1. A Simple Network Community Structure.

Definition 3. *(Modularity) The modularity of a network M is defined by Equation (2).*

$$M = \sum_{c \in C} \left(\frac{l_c}{m} - a_c^2 \right) \tag{2}$$

Here, $C = \{c_i \mid i = 1, 2, \cdots, k\}$ denotes the detected set of network community structures, l_c denotes the total number of edges interconnected between nodes within the community c, m denotes the total number of edges in the network, and

$$a_c = \frac{D_c}{2m} \tag{3}$$

where D_c denotes the sum of the node degrees of all nodes in the community c, and D_c equals to 2 times of the sum of l_c and the total edges of connecting the community c and other external communities.

According to Equation (2), the modularity of complex networks measures the degree of closeness within the community and the degree of sparseness between the communities. The closer the internal connection of the community is and the thinner the connection between the communities is, the greater the modularity M is, and vice versa. Thus, when the modularity M of a complex network is the largest, the community detection results are optimal. However, it is quite difficult to determine directly whether M has reached its maximum. Therefore, the concept of the modularity increment $\triangle M$ proposed by Newman is adopted, where the increase or decrease in the modularity M caused by merging communities c_i and c_j, which is defined as Equation (4).

$$\triangle M = \frac{2R_{ij}}{m} - 2 \times a_i \times a_j \tag{4}$$

Here, R_{ij} denotes the number of connection edges between communities c_i and c_j in which $i \neq j$. Then the modularity M increases progressively when $\triangle M > 0$. On the contrary, if $\triangle M < 0$, the modularity M is the maximum and the process of the community detection ends.

When the number of nodes and edges in a complex network are kept the same, and different communities are merged to form a new community, the number of edges among nodes within the new community is the sum of the number of edges within the 2 merged communities and the number of edges between the 2 merged communities. Accordingly, [14] points out that when the number of nodes and edges are kept the same, the increase of the modularity between the new communities formed by merging multiple known communities and other communities can be established as Equation (5).

$$\triangle M[c_z][c_k] = \begin{cases} \triangle M[c_z][c_k] + \triangle M[c_i][c_k], < c_i, c_k > \in E, c_i \in c_z \\ \triangle M[c_z][c_k] - 2 \times a_i \times a_k, < c_i, c_k > \notin E, c_i \in c_z \end{cases} \quad (5)$$

Here, c_z denotes the new community after merging, c_k denotes the old community that does not belong to c_z, c_i denotes the old community merged to c_z, and $< c_i, c_k >$ denotes the edge set from community c_i to community c_k.

Taking the network structure in Figure 1 as an example, we can see that each node represents a community. Equation (4) can be used to calculate the modularity increment $\triangle M$ among any 2 communities and form a matrix as shown in Table 2, where the first row and column represent the community number. We focus only on the 2 same communities need to be merged, and the changes within the community need not be considered, so the diagonal of the matrix can be initialized to 0. From the values of the matrix, we can observe that communities that can be merged in this example are c_2 and c_4, c_2 and c_5, c_7 and c_{12}, c_{11} and c_{12}, where $\triangle M$ is the maximal value, that is, 0.036. Taking the community c_{13} formed by merging c_2 and c_4 as an example, the results after merging are listed in Table 3.

As can be noticed from Tables 2 and 3, the modularity increment between the community c_{13} and other communities is the sum of the modularity increment between the communities c_2, c_4, and the corresponding communities. For example, in Table 3, the modularity increment of the communities c_1 and c_{13} is 0.021, which is the sum of the modularity increment, 0.033, of c_1 and c_2, and the modularity increment, -0.012, of c_1 and c_4, as shown in Table 2.

Considering that the modularity matrix update algorithm has the characteristics of merging communities in parallel and conforms to the characteristics of parallel processing on the Hadoop platform, we select the modularity incremental update method represented by Equation (5) to construct the proposed CDOH algorithm. According to the modularity increment represented by Equation (4), we initialize the entire network, treat each node as a community, and calculate the modularity increment when merging any 2 communities. Then, we iterate consecutively to find new communities. Based on the MapReduce parallel programming model, all the 2 communities with the maximum modularity increment are identified and merged in parallel. Equation (5) is used to update the modularity increment when merging any 2 communities in parallel. The community discovery process ends when the maximum modularity increment is negative. Finally, the CDOH algorithm stores the node set V as (vId, cId), where vId denotes the node number and cId denotes the community number, and the edge set E is represented as $(s, d, \triangle M)$, where s denotes the source node of the edge, d denotes the destination node of the edge, and $\triangle M$ is the modularity increment corresponding to this edge.

Table 2. $\triangle M$ Matrix before Network Merging.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0.000	0.033	0.025	−0.012	0.033	0.029	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012
2	0.033	0.000	−0.015	0.036	0.036	−0.012	−0.009	−0.012	−0.015	−0.012	−0.009	−0.009
3	0.025	−0.015	0.000	0.030	−0.015	0.025	−0.015	0.025	−0.026	0.025	−0.015	−0.015
4	−0.012	0.036	0.030	0.000	−0.009	0.033	−0.009	−0.012	−0.015	−0.012	−0.009	−0.009
5	0.033	0.036	−0.015	−0.009	0.000	0.033	−0.009	−0.012	−0.015	−0.012	−0.009	−0.009
6	0.029	−0.012	0.025	0.033	0.033	0.000	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012
7	−0.012	−0.009	−0.015	−0.009	−0.009	−0.012	0.000	0.033	0.030	−0.012	−0.009	0.036
8	−0.017	−0.012	0.025	−0.012	−0.012	−0.017	0.033	0.000	0.025	0.029	−0.012	−0.012
9	−0.021	−0.015	−0.026	−0.015	−0.015	−0.021	0.030	0.025	0.000	0.025	0.030	0.030
10	−0.017	−0.012	0.025	−0.012	−0.012	−0.017	−0.012	0.029	0.025	0.000	0.033	−0.012
11	−0.012	−0.009	−0.015	−0.009	−0.009	−0.012	−0.009	−0.012	0.030	0.033	0.000	0.036
12	−0.012	−0.009	−0.015	−0.009	−0.009	−0.012	0.036	−0.012	0.030	−0.012	0.036	0.000

Table 3. $\triangle M$ Matrix after Merging c_2 and c_4.

	1	**3**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	0	0.025	0.033	0.029	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012	0.021
3	0.025	0	−0.015	0.025	−0.015	0.025	−0.026	0.025	−0.015	−0.015	0.015
5	0.033	−0.015	0	0.033	−0.009	−0.012	−0.015	−0.012	−0.009	−0.009	0.027
6	0.029	0.025	0.033	0	−0.012	−0.017	−0.021	−0.017	−0.012	−0.012	0.021
7	−0.012	−0.015	−0.009	−0.012	0	0.033	0.03	−0.012	−0.009	0.036	−0.019
8	−0.017	0.025	−0.012	−0.017	0.033	0	0.025	0.029	−0.012	−0.012	−0.025
9	−0.021	−0.026	−0.015	−0.021	0.03	0.025	0	0.025	0.03	0.03	−0.031
10	−0.017	0.025	−0.012	−0.017	−0.012	0.029	0.025	0	0.033	−0.012	−0.025
11	−0.012	−0.015	−0.009	−0.012	−0.009	−0.012	0.03	0.033	0	0.036	−0.019
12	−0.012	−0.015	−0.009	−0.012	0.036	−0.012	0.03	−0.012	0.036	0	−0.019
13	0.021	0.015	0.027	0.021	−0.019	−0.025	−0.031	−0.025	−0.019	−0.019	0

3.2. The CDOH Algorithm

Based on the research framework of complex network community detection algorithm on the Hadoop platform shown in the Section 3.1. The CDOH has 4 steps, that is, first, we will initialize the parameters; second, we will find the maximum modularity increment; third, we will merge the communities and update the modularity increment; finally, we will generate the final community discovery results. Step 2 and step 3 will be repeated to find new communities until the maximum modularity increment is negative. We shown the flow charts of CDOH algorithm in Figure 2. Here, step 1 (Parameter initialization), step 2 (Finding the maximum modularity increment), and step 3 (Merging communities and updating the modularity increment) are implemented based on MapReduce parallel programming model of Hadoop.

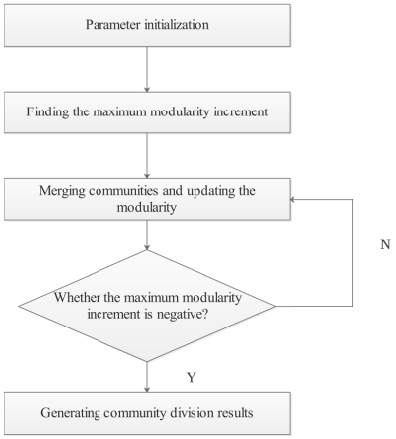

Figure 2. Flow charts of Community Detection on Hadoop (CDOH) Algorithm.

3.2.1. Parameter Initialization

The initialization phase is responsible for calculating the necessary parameters of the algorithm, which includes the total number of nodes n, total number of edges m, degree d of each node, vector a, and the modularity increment $\triangle M$ between each pair of nodes. The process is listed in Algorithm 1, the main steps of which include the following:

- First, we load the complex network data from the input file, then calculate the number of nodes n and edges m of the complex network, and broadcast the number of edges (m) to all nodes;
- Second, we calculate the degree d of each node and the vector a according to Equation (3);

- Finally, we use Equation (4) to calculate the modularity increment $\triangle M$ between each pair of nodes, and construct a new network N using this modularity increment.

Algorithm 1 Initialization of CDOH Parameters

Input:

D: Preprocessed network data;

Output:

$\triangle M$: Modularity increment;

N: Network;

N = networkLoad(D);
n = getVertices(N);
m = getEdges(N);
Broadcast the number of edges m to all nodes in the cluster;
for each Node i in N **do**
    k_i = getDegree(i);
    $a_i = \frac{k_i}{2m}$;
for each Edge e in N **do**
    $\triangle M_{ij} = \frac{R_{ij}}{m} - 2 \times a_i \times a_j$;

Here, we firstly divide the $n \times n$ matrix into multiple sub matrix, then we deploy multiple mappers, and let each mapper calculate the vector a of each node and calculate the modularity increment between each pair of nodes of each sub matrix. Each mapper works in parallel.

3.2.2. Find the Maximum Modularity Increment

After completing the modularity increment calculation, we initiate the iterative community discovery, and find multiple community pairs with the largest modularity increment, and merge them into the corresponding new communities. Taking the network shown in Figure 1 as an example. According to the $\triangle M$ matrix shown in Table 2, communities c_2 and c_4, c_2 and c_5, c_7 and c_{12}, c_{11} and c_{12} can be merged. Clearly, communities c_2, c_4, c_5 and c_7, c_{11}, c_{12} should be merged to the community c_{13} and community c_{14}, respectively.

Algorithm 2 describes the steps involved in finding the modularity increment, which has 4 steps.

- First, we compare the $\triangle M$ value of each edge e in network N, find the maximum modularity increment $max(\triangle M)$, and broadcast it to all nodes in the cluster;
- Second, we get the cartesian product T of the edge set E and node set V, $T = (s, sc, d, dc, \triangle M)$, s denotes the number of the source node, d denotes the number of destination node, sc and dc denote the community numbers of the source node and destination node respectively, and $\triangle M$ denotes the modularity increment between the source node and destination node;
- Third, we find the sub-set MC in the set T, where $\triangle M$ equals to $max(\triangle M)$;
- Finally, to organize the merged communities, we obtain the community number (i) of the source node and the community number (j) of the destination node, which represent the current communities to be merged. If i or j already belongs to a new community in C, we will get the new community to merge i and j into it, or merge i and j into another new community, whose number is $n + 1$. The final output is the community C after merging.

Algorithm 2 Find the Maximum Modularity Increment and Communities that need to be Merged

Input:

$\triangle M$: Modularity increment;

$N(E, V)$: Network;

Output:

$C = \{c_1, c_2, \cdots, c_l\}$: Communities;

$max(\triangle M)$: Maximum Modularity increment;

```
max(△M) = searchMaxDeltaM(N);
Broadcasting △M to all nodes in the cluster;
T = E × V;
for each quintuple t in T do
    if getDeltaM(t) == max(△M) then
        MC = insert(t);
for each quintuple t in MC do
    (i, j) = getCommuNum(t);
    if i ∈ C or j ∈ C then
        k = Get the new number of community i or j from C;
        c_k = insert(i,j);
    else
        n = n+1;
        c_n = insert(i, j);
```

Here, we find the maximum modularity increment $max(\triangle M)$ based on the MapReduce. After dividing the $n \times n$ matrix into multiple sub matrix, in the map phrase, each mapper finds the maximum modularity increment of each sub matrix and output the results to the reducer, and then in the reduce phrase, the reduce output the maximum modularity increment $max(\triangle M)$. Afterwards, we find the community pairs with the largest modularity increment based on MapReduce. Each mapper finds the community pairs with the largest modularity increment of each sub matrix in parallel.

3.2.3. Merging and Updating Communities

Merging and updating communities are the core of the proposed algorithm. Since after step 2, the community pairs with the maximum modularity increment are identified to be merged, the mapper updated the number of the communities that need to be merged and the community number of the corresponding nodes to their corresponding new community number in parallel, and the $\triangle M$ of any 2 communities are updated by the mapper in parallel.

The steps of merging and updating of communities listed in Algorithm 3 are the following.

- First, we obtain the Cartesian product T of the node set V and edge set E. Then, we look for the new community number corresponding to sc and dc in $t = (s, sc, d, dc, \triangle M)$. Let X to be the set of community numbers to be merged in this round contained by the new community of the community $t.sc$ and Y to be the set of community numbers to be merged in this round contained by the new community of the community $t.dc$;
- Second, using Equation (5), we will merge and update community i in X and community j in Y. If there is an edge connecting communities i and j, then the modularity increment between new communities X and Y should include the modularity increment between communities i and j. However, if there is no edge connecting communities i and j, the modularity increment between new communities X and Y should be reduced by the doubled product of vector value a_i of community i and vector value a_j of community j.

Algorithm 3 Merging and Updating Communities

Input:

$C = \{c_1, c_2, \cdots, c_l\}$: Communities;

N(E,V): Network;

Output:

$N(E, V)$: Updated Network;

Update the number of the communities that need to be merged and the community number of the corresponding nodes to their corresponding new community number;
$T = V \times E$;
for each quintuple t in T **do**
  $tsc = getNewCommuNum(t.sc)$;
  $tdc = getNewCommuNum(t.dc)$;
  if ($tsc \in C$ or $tdc \in C$) and $tsc \neq tdc$ **then**
    X = a set of community numbers to be merged in this round contained by the new community corresponding to $t.sc$;
    Y = a set of community numbers to be merged in this round contained by the new community corresponding to $t.dc$;
    for each community i in X and each community j in Y **do**
      if there exists at least an edge connecting i and j **then**
        $\triangle M_{XY} = \triangle M_{XY} + \triangle M_{ij}$
      else
        $\triangle M_{XY} = \triangle M_{XY} - 2 \times a_i \times a_j$

3.2.4. Generating Community Discovery Results

After the community discovery finishes, redundant data in the data set (primarily the matrix data) should be cleared, while the initial node set and their community number should be kept. Here, the node storage structure in the network is considered to be $V = (vId, cId)$, where vId denotes the node number and cId denotes the community number indicating which community each node belongs to. Algorithm 4 presents the process of generating the results of the community partitions, which has 2 steps:

- We will first traverse all nodes and keep the nodes with the same community number cId together. If cId is already in C, it means that the corresponding community of cId has already appeared. The node Ids in the community cId that have been stored in C need to be taken out, merged with the current node Id, and then stored in C; otherwise they are stored in C directly;
- Then we store the community and community's node set on the Hadoop distributed file system (HDFS) one by one. Thus, CDOH stores the final results of community discovery with a set of the tuple $(cId, vIds)$, and finishes the detection and discovery of complex network communities on Hadoop platform.

Algorithm 4 Generating Community Discovery Results

Input:

$N(E, V)$: Network;

Output:

$C = \{c_1, c_2, \cdots, c_l\}$: Communities;

1: **for** each $v = (vId, cId)$ in N **do**
2: **if** $cId \in C$ **then**
3: g = getNodeId(C, cId);
4: c = insert(g, vId);
5: C = insert(cId, c);
6: **else**
7: C = add(cId, vId);
8: **for** each community c in C **do**
9: output c;

3.3. Computational Complexity Analysis of the CDOH Algorithm

As presented before, in step 1, we let multiple mappers take charge of the initializing process of $n \times n$ sub-matrix. Supposed the matrix is divided into m matrices, and let each mapper takes charge of each sub-matrix in parallel, so the computational complexity of the initializing process of the matrix is the computational complexity of the initializing process of the sub-matrices, that is $O(\frac{n^2}{m})$. In step 2, the maximum modularity increment $max(\triangle M)$ and the community pairs with the largest modularity increment is found based on MapReduce. Again, if we divide the matrix into m sub-matrices, and let each mapper takes charge of each sub-matrix in parallel, the computational complexity of step 2 is also $O(\frac{n^2}{m})$. In step 3, the mapper updated the number of the communities that need to be merged and the community number of the corresponding nodes to their corresponding new community number in parallel, whose computational complexity is $O(1)$. After merging, the $\triangle M$ values of any 2 communities are updated by the mapper in parallel. Supposing that each mapper works on a sub-matrix, the computational complexity of updating $\triangle M$ is $O(\frac{n^2}{m})$. In step 4, all nodes are traversed and the nodes with the same community number are kept together, whose computational complexity is $O(n)$. Since step 2 and step 3 are repeated until the the maximum modularity increment $max(\triangle M)$ becomes negative, and after some iterations, the $n \times n$ matrix will shrink to a constant computing cost. As a result, our algorithm can achieve a performance that is in reverse proportion to the number of sub-matrices, which is determined by the number of nodes in the Hadoop platforms. Supposing we have n nodes to conduct the parallelly computing, we can achieve a $O(n)$ computing cost.

4. Experimental Results

4.1. Datasets and Evaluation Algorithms

To evaluate the accuracy and running time of CDOH, 3 real complex network data sets obtained from the Stanford Network Analysis Project (SNAP) were selected. The data sets contain the nodes and connection status of real complex networks and mark the communities to which the nodes belong. Table 4 gives the characteristics of the data sets used in the experiments.

Table 4. Characteristics of Datasets.

Dataset	No. of Nodes	No. of Edges	Node Average Degree	Description
Soc-Epinions	75,879	508,837	13.4118	Epinions.com Date Set
Web-NotreDame	325,729	1,497,134	9.1925	Web Graph Data Set
Soc-Pokec	1,632,803	30,622,564	37.5092	Poke Social Data Set

To evaluate our algorithm, we use 2 state-of-the-art algorithms in our experiments, that is, the traditional complex network community detection algorithm Fast Community Detection (FCD) proposed by Newman [9] and the non-overlapping community detection algorithm Non-Overlapping Community Detection Idea (OCDI) proposed by Zhang et al. [15].

All the algorithms were implemented with Java, and our algorithm was deployed on Hadoop cluster made of 3 different computers, of which 1 serving as a master node and the other 2 serving as slave nodes. The following experimental results are represented as average across the 10 runnings.

4.2. Analysis of Community Detection Accuracy

We used the community detection accuracy (CDA) metric to measured the accuracy of community detection. CDA is defined as the ratio of the number of nodes in the correctly identified communities to the total number of nodes in the network, which is shown in Equation (6).

$$CDA = \frac{\sum_{i=1}^{k} max\{|C_i \cap C'_j| \mid C'_j \subset C_i\}}{n}, j = 1, 2, \cdots, l \tag{6}$$

Here, $C = \{c_1, c_2, \cdots, c_k\}$ denotes the original and accurate community set, $C' = \{c'_1, c'_2, \cdots, c'_l\}$ denotes the community set identified by the community detection algorithm, $max\{|C_i \cap C'_j| \mid C'_j \subset C_i\}$ denotes the maximum number of the common nodes between all community sets and the i-th accurate community c_i, and n denotes the number of nodes. As can be seen, the larger the value is, the higher the accuracy of a community detection algorithm is and the better the quality of the resulting community is. Figure 3 shows the community discovery accuracies of the considered algorithms on the 3 different data sets.

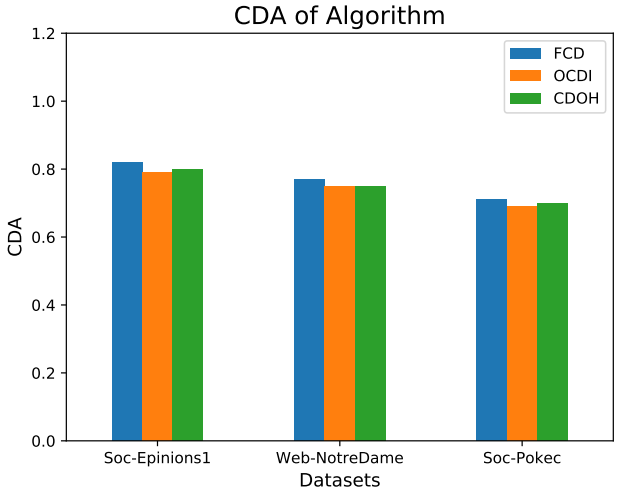

Figure 3. Comparison of the Accuracy of the Community Detection Algorithms.

It can be noticed from Figure 3 that the accuracy of the CDOH algorithm is slightly lower than that of the FCD algorithm (on average by 1.7%) and similar to that of OCDI. The reason for this is

that CDOH and OCDI have similar community merging strategies and module update principles. While multiple communities are merged at one time in the same iteration according to CDOH and OCDI, FCD only supports one-time merging of 2 communities in a single iteration, which results in the accuracy gap between FCD and the other 2 algorithms.

We also used the normalized mutual information (NMI) to evaluate our algorithm in comparison to the other 2 algorithms. NMI [33] is a standard factor which is often used to detect the difference between the results of the division and the true partition of the network. NMI can be described in Equation (7), in which $H(X)$ is the entropy of X, and $H(X|Y) = H(X,Y) - H(Y)$.

$$NMI(X,Y) = \frac{H(X) - H(X|Y) + H(Y) - H(Y|X)}{2max(H(X),H(Y))} \tag{7}$$

We can see from Figure 4 that the NMI of the 3 algorithms can reach at least 75%. Our algorithm, CDOH, has a very similar NMI score to the FCD algorithm and has a slightly higher score than OCDI. Again, we consider this is due the fact that our algorithm has similar community merging strategies and module update principles.

However, the computing cost of our algorithm is much better than that of FCD, which will be discussed in Section 4.3.

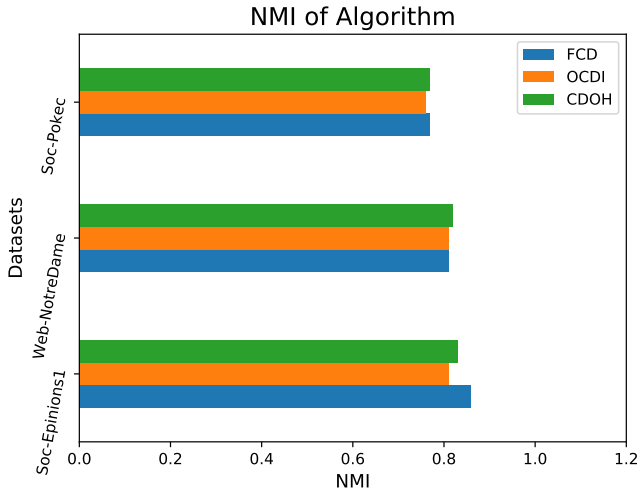

Figure 4. Comparison of the normalized mutual information (NMI) of the Community Detection Algorithms.

4.3. Analysis of Community Detection Efficiency

CDOH is a community detection algorithm based on Hadoop platform for large-scale complex networks. For processing large scale data, the run time of the algorithm is an important metric to evaluate its performance of the algorithm. Figure 5 shows the comparison of the run time of the 3 considered algorithms.

It can be noticed from Figure 5 that CDOH is highly efficient. To compared with OCDI and FCD, we can see that CDOH is about 2.1 times and 3.2 times faster, respectively, which is mainly determined by the number of slave nodes on the Hadoop platform. Compared with the traditional community detection algorithms, CDOH uses significantly less time required for community merging and modularity updating.

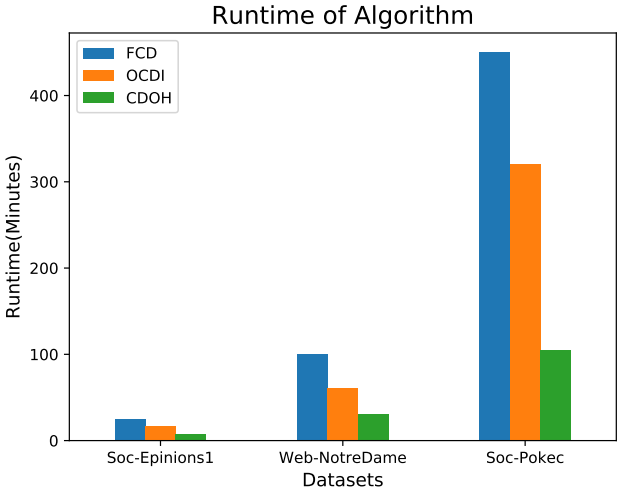

Figure 5. Comparison of the Runtime of Community Detection Algorithms.

5. Conclusions and Future Works

5.1. Conclusions

In this paper, we proposed a community detection algorithm called CDOH based on the Hadoop platform to implement accurate and fast community identification in large-scale complex networks. The algorithm was based on the modularity increment calculation method, which employed the theory of complex networks to find multiple communities satisfying certain merging conditions. The parallel merging and modularity updating of communities based on MapReduce used in the proposed algorithm reduce the number of iterations. CDOH was compared with traditional complex network algorithms using real large-scale complex networks. The experimental results evaluated the effectiveness of CDOH in large-scale network community detection.

5.2. Future Works

Our proposed CDOH algorithm is independent of the underlying big data platform. To prove its effectiveness and efficiency, we implemented the CDOH algorithm and other complex network community detection algorithms based on the Hadoop platform. However, in the Hadoop platform, the MapReduce intermediate results are first stored in disk files, and a large number of I/O operations will affect the whole calculation time; while in the Spark platform, the intermediate results are stored in memory, which avoids the performance overhead brought by I/O. In the future, we will implement the CDOH algorithm on the Spark platform and evaluate the efficiency. Furthermore, our proposed CDOH algorithm focused on static complex network community discovery, in the future, we plan to adapt the proposed algorithm to the evolving community networks.

Author Contributions: Conceptualization, M.H.; methodology, H.L.; software, Z.M.; validation, M.H. and X.G.; formal analysis, M.H.; investigation, H.L.; resources, H.L.; data curation, H.L.; writing–original draft preparation, M.H.; writing–review and editing, M.H., H.L. and Z.M.; visualization, H.L.

Funding: This research is supported by the National Natural Science Foundation of China (61100112,61309030), Beijing Higher Education Young Elite Teacher Project (YETP0987), the Top Discipline Construction Project of Central University of Finance and Economics in 2019 (Key Technologies and Application of Independent Controllable Block Chain), the Fundamental Research Funds for the Central Universities, the Education and Teaching Reform Fund of Central University of Finance and Economics in 2018(2018GRZDJG06).

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Watts, D.J.; Strogatz, S.H. Collective dynamics of 'small-world' networks. *Nature* **1998**, *393*, 440–442. [CrossRef] [PubMed]
2. Faloutsos, M.; Faloutsos, P.; Faloutsos, C. On power-law relationships of the Internet topology. *ACM SIGCOMM Comput. Commun. Rev.* **1999**, *29*, 251–262. [CrossRef]
3. Sen, P.; Manna, S.S. Clustering properties of a generalized critical Euclidean network. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **2003**, *68*, 026104. [CrossRef] [PubMed]
4. Zheng, X.; Chen, J.; Shao, J.; Bie, L. Topological properties analysis of Beijing public transport network based on complex network theory. *J. Phys.* **2012**, *61*, 95–105.
5. Fan, R. Cooperative Innovation of Social Governance under the Paradigm of Complex Network Structure. *Soc. Sci. China* **2014**, *4*, 98–120.
6. Newman, M.E.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2003**, *69*, 17–32. [CrossRef]
7. Yang, J.; Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.* **2015**, *42*, 181–213. [CrossRef]
8. Xin, S.; Giancarlo, S.; Vincenzo, M.; Antonio, P.; Christian, E.; Chang, C. An Edge Intelligence Empowered Recommender System Enabling Cultural Heritage Applications. *IEEE Trans. Ind. Inf.* **2019**, *15*, 4266–4275.
9. Newman, M.E. Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **2003**, *69*, 066133. [CrossRef]
10. Clauset, A.; Newman, M.E.; Moore, C. Finding community structure in very large networks. *Phys. Rev. E* **2004**, *70*, 066111. [CrossRef]
11. Pan, L.; Jin, J.; Wang, C.; Xie, J. Edge Community Mining Based on Local Information in Social Networks. *J. Electron.* **2012**, *40*, 2255–2263.
12. Xiong, Z. *Community Discovery Technology and Its Application in Online Social Networks*; Central South University: Changsha, China, 2012.
13. Huang, W. *Research on Web Community Discovery Algorithms*; Beijing University of Posts and Telecommunications: Beijing, China, 2013.
14. Leng, Z. Research on network community discovery algorithm based on greedy optimization technology. *J. Electron.* **2014**, *42*, 723–729.
15. Zhang, X.; You, H.; Zhu, W.; Quiao, S.; Li, J.; Gutierrez, L.A.; Zhang, Z.; Fan, X. Overlapping community identification approach in online social networks. *Physica A Stat. Mech. Appl.* **2015**, *421*, 233–248. [CrossRef]
16. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of community hierarchies in large networks. *Comput. Res. Repos.* **2008**, abs/0803.0476.
17. Parsa, M.G.; Mozayani, N.; Esmaeili, A. An EDA-based community detection in complex networks. In Proceedings of the International Symposium on Telecommunications, Tehran, Iran, 9–11 September 2014; pp. 476–480.
18. Oliveira, J.E.M.D.; Quiles, M.G. Community Detection in Complex Networks Using Coupled Kuramoto Oscillators. In Proceedings of the International Conference on Computational Science and ITS Applications, Guimaraes, Portugal, 30 June–3 July 2014; pp. 85–90.
19. Jing-Ya, X.; Tao, L.; Lin-Tao, Y.; Davison, M. Finding College Student Social Networks by Mining the Records of Student ID Transactions. *Symmetry* **2019**, *11*, 307.
20. Yuhui, G.; Qian, Y. Evolution of Conformity Dynamics in Complex Social Networks. *Symmetry* **2019**, *11*, 299.
21. Giuseppe, A.; Domenico, C.; Antonio, M.; Antonio, P. Mobile Encrypted Traffic classification Using Deep Learning. In Proceedings of the 2018 Network Traffic Measurement and Analysis Conference (TMA), Vienna, Austria, 26–29 June 2018.
22. Giuseppe, A.; Domenico, C.; Antonio, M.; Pescapé, A. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 445–458.
23. Ruoyu, W.; Zhen, L.; Yongming, C.; Deyu T.; Jin Y.; Zhao Y. Benchmark Data for Mobile App Traffic Research. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, New York, NY, USA, 5–7 November 2018.

24. Clauset, A. Finding local community structure in networks. *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* **2005**, *72*, 026132. [CrossRef]
25. Li, J. *Research on Overlapping Community Discovery Algorithm Based on Hadoop Platform*; Jilin University: Changchun, China, 2014.
26. Riedy, J.; Bader, D.A.; Meyerhenke, H. Scalable Multi-threaded Community Detection in Social Networks. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops & Phd Forum, Shanghai, China, 21–25 May 2012; pp. 1619–1628.
27. Moon, S.; Lee, J.G.; Kang, M. Scalable community detection from networks by computing edge betweenness on MapReduce. In Proceedings of the 2014 International Conference on Big Data and Smart Computing (BIGCOMP), Bangkok, Thailand, 15–17 January 2014; pp. 145–148.
28. Wu, W.; Li, M.; Li, G. A Parallelization of Louvain algorithm. *Comput. Digit. Eng.* **2016**, *44*, 1402–1406.
29. Blondel, V.D.; Guillaume, J.L.; Lambiotee, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *10*, P10008. [CrossRef]
30. Lai, B. *Research on Parallelization of Community Discovery Algorithm Based on Hadoop*; Jiangxi University of Science and Technology: Ganzhou, China, 2017.
31. Alessio, C.;Tiziano, D.M.; Daniele, D.S.; Grossi, R.; Marion, A.; Versari, L. *D2k: Scalable Community Detection in Massive Networks via Small-Diameter k-Plexes*; KDD 2018; ACM: New York, NY, USA, 2018; pp. 1272–1281.
32. Vincenzo, M.; Antonio, P.; Giancarlo, S. Community detection based on Game Theory. *Eng. Appl. Artif. Intell.* **2019**, *85*, 773–782.
33. Mcdaid, A.F.; Greene, D.; Hurley, N. Normalized Mutual Information to evaluate overlapping community finding algorithms. *CoRR* **2011**, abs/1110.2515.