Parallel Community Mining in Social Network using Map-reduce

1 Wen Cui, 2 Guoyong Wang, 3 Ke Xu

1 Computer Information Engineering Department, LuoYang Institute of Science and Technology, Luoyang, Henan, China, cuiwen@sina.com
2 Computer Information Engineering Department, LuoYang Institute of Science and Technology, Luoyang, Henan, China, wgy@lit.edu.cn
3 College of Computer Science, South-Central University for Nationalities, Wuhan, China, cole.xu@gmail.com

Abstract

The mobile social network plays an essential role as the spread of information and relationship. This paper proposes a parallel algorithm based on MapReduce for finding community in a mobile social network where individuals communicate with one another using mobile phones with register identification information and analyzes the behavior of the communication patterns with taking the actual call detail records received and dialed by users into account. The proposal algorithm is composed of three main components - map, reduce and merge for mining groups and a dynamic programming algorithm for selecting subgroups to combine into a big community. Empirical studies on a large real-world mobile social network show that performance of our algorithm is an effective and fast algorithm for mining community in social network.

Keywords: Social Network, Community Mining, Map-Reduce

1. Introduction

Recently mining on sub-graphs or communities has been emerging as a prevailing interest and produced many practical applications, whose emphasis points are on designing novel algorithms and revealing underlying patterns of real-world graphs [1, 16]. Among most of the applications in graph mining, an efficient and principled method for detecting communities in networks was proposed in [2, 5], many networks display community structure-groups of vertices within which connections are dense but between which they are sparser-and highly sensitive computer algorithms have in recent years been developed for detecting such structure [9, 11]. Mining community structure in networks is a topic of considerable recent interest within the physics community, but most methods proposed so far are unsuitable for very large networks because of their computational cost [15, 17].

A social network is a social structure connecting individuals or organizations. Mobile social network is a typical social network where one or more individuals of similar interests or commonalities, conversing and connecting with one another using the mobile phone [3, 12]. Social network analysis (SNA) is the mapping and measuring of relationships and flows between people, groups, organizations, computers, or other information/knowledge processing objects [7, 18]. Some typical problems in SNA include discovering groups of individuals sharing the same properties [8] and evaluating the importance of individuals [6, 14, 20].

We propose a fast parallel algorithm based on MapReduce for finding community in mobile social networks of China Mobile Communications Corporation. We use our metric to analyze received and dialed call records within an organization to extract social hierarchy. We analyze the behavior of the communication patterns with taking the actual mobile phone call detail records. The resulting data provide a multi-dimensional and temporally fine grained record of human interactions on an unprecedented scale. Further, in principle, the methods we discuss here could be applied to hundreds of millions of mobile phone users.

2. Related works

Many real-world networks are sparse and hierarchical, discovery and analysis of community structure in networks is a topic of considerable recent interest within the physics community, but most methods proposed so far are unsuitable for very large networks [15, 17]. An algorithm is proposed in
to extract meaningful communities from this network, revealing large-scale patterns present in
the purchasing habits of customers. Karrer and Newman has witnessed the emergence, primarily in on-
line communities, of new types of social networks that require for their representation more complex
graph structures than have been employed in the past [21, 23].

A set of algorithms for discovering community structure in networks were proposed for dividing
natural divisions of network nodes into densely connected subgroups [13]. MapReduce [22] is a
programming model that enables easy development of scalable parallel applications to process vast
amounts of data on large clusters of commodity machines. Analyzing this kind of data such as net logs,
web document repositories, telephone call records, search logs or picture repositories often can not be
one on a relational database because of the large amount of data [10]. Hadoop is a popular open-source
map-reduce implementation which is being used in companies like Yahoo, Facebook etc. to tore and
process extremely large data sets on commodity hardware [19]. It focuses on providing the necessary
minimum functionality, other open source components are available, which address e.g., key-based
data access, or more complex job and data schema management.

Our work is related to community mining. There are a host of algorithms for community
mining. However, it remains to be a challenging problem for a large social network. In a mobile
social network, the number of communities and the size of communities are unknown. Also,
these methods are quite expensive when applied to a large network. In this paper, we propose a
fast parallel community mining in mobile social network. This parallel method is developed
using MapReduce, which is a simplified parallel program paradigm for large scale, data
intensive, parallel computing jobs. MapReduce hides the parallel machine from the programmer
by simplifying the parallel programming model to two functions: the map function and the
reduce function. Given a list of keys and associated values, the map function produces an
intermediate set of keys and values. The reduce function then combines these intermediate
values into a final result.

3. Mining community in mobile network

Community structure is a basic property of a CMSN and communities represent real circles of social
groups in which members are more likely to have common interest with each other [20]. There exist a
number of algorithms for community finding, and they partition graphs based on the node connections.
MapReduce-based community finding (MRCF) framework aims to implement sub-group finding on
complex graphs based on Hadoop.

3.1. Campus mobile social network

We extract a Campus Mobile Social Network from the call log and model it as a directed weighted
diagram: a phone user corresponds to a node; a directed edge from node $u$ to node $v$ is established, if there
exists communication from $u$ to $v$, with the corresponding communication time as the weight of the edge.
We denoted the graph as $G = (V, E, W)$, where $V$, $E$, and $W$ represent nodes, edges, and weights,
respectively.

3.2. Extract social network feature

Degree of node

Degree($i$) is defined as unique dialer and receiver who had mobile phone call communication
with node $i$. That is, it is the simple average of the in-degree and out-degree of the node:

$$\text{Degree}(i) = \frac{\sum_{j} E_{ij} + E_{ji}}{2} \quad (1)$$

Belonging Degree

Let $\mathcal{A}$ be the initial set of active nodes. The belonging degree of set $\mathcal{A}$ is computed as:
\[ B(A) = \frac{\nu_A}{N} \]  

(2)

where \( \nu_A \) is the number of nodes influenced by \( A \) during information diffusion process.

**Combination-Entropy**

If a node \( i \) activates its neighbor \( j \), we label the edge \( E_{ij} \) as live. If \( E_{ij} \) is live and \( i \) belongs to community \( C_m \), but \( j \) belongs to a different community \( C_l \), we say that \( j \) is a live node of \( C_m \). Let \( A[C_m] \) be the set of live nodes of \( C_m \). The combination entropy of community \( C_l \) to \( C_m \) is defined as:

\[
ComBetw(C_m^c, C_l) = \max_{i \in C_m, j \in (C_m, j \in C_l)} \frac{B_m(\{i\})}{B_m(\{j\})}
\]

(3)

where \( B_m(\{i\}) \) denotes the betweenness of node \( i \) in its community \( C_m \). \( B_m(\{j\}) \) denotes the betweenness of node \( j \) outside its community \( C_m \). A node with high betweenness means that the corresponding person is a contact point between different social groups.

### 3.3. MapReduce

MapReduce is a parallel programming paradigm, originally introduced by Google [19, 22], whose central focus is to simplify the processing of large datasets on inexpensive cluster computers. The map function takes as input a set of key-value pairs, designated as \( k_1 \) and \( v_1 \), provided directly from the user-defined input files. Within the map function, the user specifies what to do with these keys and values. The map function outputs another set of keys and values, designated as \( k_2 \) and \( v_2 \). The reduce function sorts the key value pairs by \( k_2 \). All of the associated values \( v_2 \) are reduced and emitted as value \( v_3 \). The map and reduce functions are as follows:

\[
\text{Map} \ (k_1, v_1) \rightarrow [(k_2, v_2)]
\]

(4)

\[
\text{Reduce} \ (k_2, [v_2]) \rightarrow [v_3]
\]

(5)

At the MapReduce run-time level, the map operations are distributed by the master-server to the chunk-servers. The scheduler makes an effort to schedule computation on the same node where the data is stored. Meanwhile, other chunk-servers assigned to the reduce phase begin to take the \( (k_2, v_2) \) value pairs and sort them by \( k_2 \). These sorted arrays of \( v_2 \) values are passed to the reduce functions on these same assigned nodes. These outputs are finally saved on the GFS. It is quite common for an application to string together many simpler MapReduce operations.

### 3.4. MapReduce-Based Community Finding (MRCF) algorithm

The methods described in this paper all assume that we are given a network structure that we wish to divide into communities in such a way that every vertex belongs to one of the communities. The problem of finding good divisions of networks has been studied for some decades now in computer science. Computer scientists refer to this task as graph partitioning. Graph partitioning problems arise for example in the optimal allocation of processes to processors in a parallel computer.

Here we define the size of a community by its vertices number. After loading a dataset of a network, MRCF uses one MapReduce pass to parse the dataset and form three information tables.

MapReduce-based community finding (MRCF) framework aims to implement sub-group finding on complex graphs based on Hadoop. It's more interesting and representative to apply this framework on directed graphs. For clearly depicting MRCF, we show the serial community finding algorithm in Algorithm 1.

**Algorithm 1** Map function for the Customers’ call-detail-records dataset.

1: map(const Key& key, const Value& value)/* OwnerNumber,call-detail-records */
2: { OwnerNumber = key;
3: CallNumber = value. CallNumber; /* compute CallDuration using call-detail-records */
output_key = (CallNumber, OwnerNumber);
output_value = (CallDuration);
Emit(output_key, output_value);

Algorithm 2 Reduce function for the Customers’ call-detail-records dataset.
reduce(const Key& key, const ValueIterator& value) /* (CallNumber, OwnerNumber) */
{ betweenness = /* compute betweenness for a contact point between different social groups */
  Emit(key, (betweenness)); } }

Algorithm 3 Merge function for community combination.
merge(const FirstKey& fristKey, /* (fristCommunityID, nodeList) */
const FirstValue& fristValue, /* sets of nodes of one community */
const SecondKey& secondKey, /* (secondCommunityID, nodeList) */
const SecondValue& secondValue /* sets of nodes of other community */) {
if (betweenness > θ or communityNo < Cm) {
  fristCommunity = fristCommunity ∪ secondCommunity;
  Emit(fristKey.communityID, nodeList);
} }

Step 1: Distributed storage. MRCF is based on Hadoop, a Google's GFS implementation, is being used in companies like Yahoo, Facebook etc. to store and process extremely large data sets [19, 22]. In Step 1, the target network is stored as textual files in a specific format. Using Hadoop, the file can be easily divided into a set of blocks with the same size and distributed on nodes of the cluster to keep load balance in the cluster. Hadoop can process the blocks concurrently on nodes where the data is located [19].

Step 2: Neighbor vertices finding and community initialization. In this step we use a MapReduce pass to do two tasks, one is to find adjacent neighbor of each vertex to form an adjacent vertices table, the other is to find community of size two (one edge and two vertices) and their sub-groups. Each mapper inputs one block of the dataset. It is used to detect the groups on the borders of blocks and to guarantee our algorithm to be complete (against losing patterns). Community Set and nodeList are updated by Step 3 respectively.

Step 3: community extension. It is the key step of the MRCF. This step also takes one MapReduce pass. The map stage working with reduce stage extends patterns of size $i$ to $i+1$ or $i+m$.

Mapper - extend the communities of size $i$ to $i+1$ or $i+m$, calculate their patterns and produce a group key with the community and sub-groups. Each mapper outputs one or more key-value pairs, and the pairs with the same key will automatically be grouped into the same reducer.

Reducer - remove the duplicated nodes. Since different nodes may get the same subgraph of size $i+1$ or $i+m$ when the sub-group $i$ are extended. During the grouping process mentioned above, we compare the canonical label of each match and keep just one of the same sub-group. The outputs of reducers are grouped into different files according to the community label.

Merger - a merger reads from two sets of reducer outputs that cover the same key range. Since these reducer outputs are sorted already, this merger simply does the merge part of the community combination.

3.5. Macro-averaged performance metric

Macro-averaged performance (MAP) is a conventional metric for evaluating community detection in community divide to node in network. The system-made decisions on each node in network belongs to which community with respect to a specific category $L_c \in C = \{C_1, C_2, \ldots, C_m\}$ can be divided into four groups: True Positions ($TP_c$), False Positions ($FP_c$), True Negatives ($TN_c$) and False Negatives ($FN_c$), respectively. The corresponding evaluation metrics are defined as:
4. Experiment

We evaluate the Macro-averaged performance of the proposed algorithm. The data sets we used and experimental setup beforehand are described at first, and then the results with different parameters are shown. We take the Newman Clustering [11] and group detection algorithms [9] as the benchmark to evaluate the proposed algorithm.

4.1. Data sets

We have a three-month call detailed record data of a campus CMCC-V-Net from China Mobile, the largest mobile communication service provider in China. A virtual campus mobile network in which all users are from universities or colleges. The phone number belonged to which students are registered when they join in the CMCC-V-Net. We could acquire the student belongs to which university and department, major specialty, grade and class. We compare the community finding with the register information and verify the macro-averaged performance of our community mining algorithm.

We extract a mobile social network from the CDR data for 3 months using the method presented in Section 3, and obtain a network with 362581 nodes from more than 50 universities or colleges and an average degree of 235.3.

4.2. Experimental results

Our algorithm automatically divides the job and distributes them to each node. So we can dynamically add the quantity of nodes, which will enhance the performance of the algorithm. We run the program on a blade-cluster with 40 nodes. Each node has 4 processors which are equipped with Intel (R) Xeon (TM) CPU 2.80 GHZ and 4 GB memory. In the experiment, we run the algorithm to do the same task on cluster of varying nodes.

We do experiment with different $\theta$ from 0.1 to 0.8 to combine the communities generated in the reduce step. When $\theta=0$, all the communities are combined, just one community; when $\theta=0.2$, most of the communities are combined and the number of final communities is 25; when $\theta=0.8$, the generated communities are dispersed and small; when $\theta=0.5$, we get 61 communities and this appear to be an appropriate number after we check the data manually. Hence we set $\theta=0.5$ as the threshold. Community number $C$ versus $\theta$ is shown in Table 1. Another parameter to be set is the number of iterations.

<table>
<thead>
<tr>
<th>$\theta$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>10</td>
<td>25</td>
<td>31</td>
<td>43</td>
<td>61</td>
<td>79</td>
<td>152</td>
<td>195</td>
</tr>
</tbody>
</table>

We observe that it is not completely stable after 5 iterations, i.e. the reduce partitions of some nodes are still changed. To be safe, we set the number of iterations at 20 to make it more stable.

In this paper, we take the runtime of community detection as part of runtime our algorithm. The algorithm partitions the whole network into communities. We compare those phone call numbers to register information when they join in the CMCC-V-Net. We check the register information and find that most of those communities are dormitory groups in which roommates communicate frequently with each other than another student.
We also compute Precision, Recall, and Macro-average for Community Detection algorithm by carrying out experiment with different $\theta$ from 0.1 to 0.8. According to community number $C$ versus $\theta$ is shown in Table 1, we classify the campus mobile social network into different community/cluster number $C$ by Newman Clustering, and compute Precision, Recall, and Macro-average for Newman Clustering algorithm. Precision, Recall, and Macro-average of the three comparison algorithms are shown in Table 2.

We compare the Precision, Recall, and Macro-average of the three algorithms on graph. Figure 1, Figure 2 and Figure 3 is shown the Precision, Recall, and Macro-average percents.

**Table 2 Precision, Recall, and Macro-average for Newman Clustering and Community Detection algorithm**

<table>
<thead>
<tr>
<th>$\theta$</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster size</td>
<td>10</td>
<td>36</td>
<td>52</td>
<td>69</td>
<td>91</td>
<td>129</td>
<td>186</td>
<td>217</td>
</tr>
<tr>
<td>Precision</td>
<td>NC</td>
<td>52.1</td>
<td>59.0</td>
<td>63.2</td>
<td>69.2</td>
<td>71.9</td>
<td>66.5</td>
<td>61.6</td>
</tr>
<tr>
<td></td>
<td>GD</td>
<td>43.2</td>
<td>53.5</td>
<td>66.5</td>
<td>73.0</td>
<td>78.1</td>
<td>73.5</td>
<td>67.4</td>
</tr>
<tr>
<td></td>
<td>MRCF</td>
<td>47.2</td>
<td>56.5</td>
<td>68.1</td>
<td>75.4</td>
<td>79.3</td>
<td>76.2</td>
<td>70.2</td>
</tr>
<tr>
<td>Recall</td>
<td>NC</td>
<td>61.3</td>
<td>68.1</td>
<td>73.7</td>
<td>79.6</td>
<td>81.4</td>
<td>76.7</td>
<td>71.8</td>
</tr>
<tr>
<td></td>
<td>GD</td>
<td>53.5</td>
<td>63.7</td>
<td>76.2</td>
<td>82.5</td>
<td>87.6</td>
<td>83.5</td>
<td>77.5</td>
</tr>
<tr>
<td></td>
<td>MRCF</td>
<td>56.4</td>
<td>65.2</td>
<td>79.3</td>
<td>85.6</td>
<td>89.2</td>
<td>85.4</td>
<td>79.3</td>
</tr>
<tr>
<td>Macro-average</td>
<td>NC</td>
<td>56.1</td>
<td>63.3</td>
<td>68.6</td>
<td>74.2</td>
<td>76.7</td>
<td>71.3</td>
<td>66.4</td>
</tr>
<tr>
<td></td>
<td>GD</td>
<td>47.8</td>
<td>58.2</td>
<td>71.8</td>
<td>74.7</td>
<td>82.5</td>
<td>78.2</td>
<td>72.8</td>
</tr>
<tr>
<td></td>
<td>MRCF</td>
<td>52.5</td>
<td>60.7</td>
<td>74.5</td>
<td>80.2</td>
<td>85.6</td>
<td>81.7</td>
<td>76.4</td>
</tr>
</tbody>
</table>

**Figure 1.** Precision for Newman Clustering, Community Detection Algorithm and MRCF Algorithm

**Figure 2.** Recall for Newman Clustering, Community Detection Algorithm and MRCF algorithm
As it is shown in Table 1 and those graphs, we get 61 communities when $\theta=0.5$ and classify the campus mobile social network into 61 clusters. The Precision, Recall, and Macro-average percent of the two algorithms when $\theta=0.5$ is higher than other conditions. It verifies that we set $\theta=0.5$ as the threshold in this appear to be an appropriate number.

Figure 4 is shown time (s) versus network size when using community detection algorithm and MRCF algorithm for finding community. As we can see from this graph, time for community detection algorithm grows speed fast when the size of network goes big compared to the MRCF algorithm.

5. Conclusions

In this paper, we propose a fast parallel algorithm called MapReduce-Based Community Finding (MRCF) for mining community in mobile social network. MRCF algorithm is based on MapReduce which is a parallel model and often used in data mining. Empirical studies on a large real-world campus mobile social network show that performance of our algorithm is better than the state-of-the-art Newman Clustering algorithm and group detection algorithms [9] for mining community in mobile social network.

We anticipate that the methods outlined here will have a major impact in the social sciences, providing insight into the underlying relational dynamics of organizations, communities and, potentially, societies. At the micro level these methods, for example, provide a new approach to studying collaboration and communication within organizations—allowing the examination of the evolution of relationships over time. More dramatically, these methods allow for an inspection of the dynamics of macro networks that were heretofore unobservable.

This work opens to several interesting directions for future work. Notably, it is relevant to take...
spatial information of mobile customers into consideration, and construct locations based social networks to find influential nodes; it is also interesting to study the evolution of influential nodes over time.

6. Acknowledgments

This research was supported by the Natural Science Foundation of LuoYang Institute of Science and Technology (Grant No. 2008QZ28, 2008YZB11). Ke Xu is supported by "the Fundamental Research Funds for the Central Universities", South-Central University for Nationalities (CZY12008).

7. References


