

Parallel Community Detection on Massive Graphs

Bing Tian^{1,2}

¹China Coal Technology Engineering Group Chongqing Research Institute, Chongqing 400039, China;

²State Key Laboratory of Gas Disaster Emergency Information Technology, Chongqing 400039, China.

tb_ccteg@163.com

Keywords: community detection, modularity, parallel computing

Abstract. Community detection groups a network into node sets according to their connections. It is an effective way to understanding and analyzing graph-structured data, such as social networks, collaboration networks, and bioinformatic networks. With the flourishing development of social network applications, it has become more desirable to explore graphs from a community-level view. However, based on sequential algorithms, most existing community detection methods are not suitable for massive graphs. In this paper, we propose a Parallel Community Detection approach, named ParCoDe. Just like the native sequential algorithm, it uses “community modularity” as the metric. The detecting process starts from each single node and performs in a bottom-up way. In order to improve its performance, we propose an approximate solution to accelerate the speed of detection with little loss of accuracy. We have implemented ParCoDe on Giraph. Comprehensive experiments on both real and synthetic datasets demonstrate that ParCoDe is of well scalability and is efficient for community detection.

Introduction

Community detection is a kind of topology analysis approach by dividing a network into node groups according to the connections among them. The main goal of community detection is to make nodes of the same group densely connected, while connections between groups are sparse [1]. It is an effective way to understanding and analyzing graph-structured data. In social networks, community detection can be used to find people with same interests and make advertisements more personalized [2]. For collaboration networks, with the help of community detection, it is much easier to reveal collaboration patterns in further complex analysis [3]. In bioinformatics, a promising use of community detection is to predict the unknown functions of genes and proteins [4]. Recently, prosperous social network applications make it more desirable to explore graphs from a community-level view. Taking Twitter which has more than 200M users [5], and Facebook which has more than 500M users [6], as examples, communities are more tractable than various users at such scale.

Existing approaches on community detection can be mainly classified as partitioning methods, clustering methods, spectral methods, and modularity-based methods [1]. However, all these methods are based on sequential algorithms. Just like most clustering algorithms, community detection is computationally expensive sometimes NP-hard. Since detecting communities on a single machines could easily run out the computational resources, it is challenging to find communities on massive graphs. Recently, several parallel community detection approaches have been proposed based on MapReduce [7] programming model [8, 9, 10]. But, as it is an iterative process, BSP [11] programming model are more suitable for graphs for the consideration of extra cost. Therefore, in this paper, we propose a Parallel Community Detection approach (ParCoDe) based on BSP programming model. The main contributions can be summarized as follows:

- (1) Based on the classical CNM (Clauset-Newman-Moore) algorithm, we propose a parallel community detection algorithm with adjustments on modularity updating and community merging;
- (2) In order to improve the performance of ParCoDe, we introduce an approximate approach to accelerate its efficiency with little accuracy penalty;
- (3) We conduct exhaustive experiments on both real and synthetic datasets and demonstrate that our approach is of well scalability and efficiency.

The rest of this paper is summarized as follows: we first introduce several basic definitions in Preliminaries section; ParCoDe is detailed in the following section; then it is empirically studied in Experimental Evaluation section; Related Work section is a survey of existing community detection approaches; the paper is concluded by Conclusion section.

Preliminaries

In this section, we introduce several basic definitions including modularity, CNM algorithm and BSP programming model.

Modularity and CNM Algorithm. Modularity is first proposed by Newman [12]. It is a major quality metric for community detection. High modularity values correspond to good community detection results. Specifically, the modularity Q of a community detection result is calculated as:

$$Q = \sum_i (e_{ii} - a_i^2),$$

where e_{ii} denotes the fraction of edges that connect nodes in community i and j , a_i denotes the fraction of edges incident to nodes in community i . Since detecting communities with maximum modularity is an NP-hard problem, CNM algorithm is a greedy solution [13]. It maintains a matrix of ΔQ_{ij} , which is the modularity increment of merging community i and j . Then the overall community detection process repeatedly selects the largest ΔQ_{ij} and after merging, updates each ΔQ_{jk} by:

$$\Delta Q_{jk} = \begin{cases} \Delta Q_{ik} + \Delta Q_{jk}, & \text{if community } k \text{ is connected to } i \text{ and } j; \\ \Delta Q_{ik} - 2a_j a_k, & \text{if community } k \text{ is connected to } i \text{ not } j. \end{cases},$$

where a_i denotes the fraction of degrees incident to community i . The algorithm terminates when all ΔQ_{jk} s are negative.

BSP Programming Model. BSP is short for Bulk Synchronous Parallel [14]. It consists of three components: concurrent computation, communication and barrier synchronization. These three operations are iteratively performed in one round called superstep. The model is well-suited for distributed-memory computing. A popular implementation of this model is Pregel [11], which is proposed by Google. It is a node centric programming framework and well-designed for iterative applications. There are also many open-source Pregel-like platforms, such as Apache Hama, Apache Giraph, and etc. In this paper, we choose to implement ParCoDe on Giraph.

The Parallel Community Detection Approach (ParCoDe)

In this section, we present the algorithm of parallel community detection which is an approximation approach. Suppose that each node is initialized and contained two fields: a_i and ΔQ_{ij} . It can be easily achieved in Pregel-like platforms. Hence we omit the details of initialization and mainly focus on the selecting and updating operations.

The Algorithms. The community selection algorithm is presented in Algorithm 1. It is used to select the communities with maximal modularity increment to be merged. Algorithm 2 is used to merge the selected communities and update all the modularity value held by each node. The two algorithms run alternatively. Obviously, the overall cost of selection and update is four supersteps. In

the native CNM algorithm, each time it merges only two communities. Thus, $(n - 1)$ rounds of iteration are required. To further optimize its performance, we propose a simple but effective approximation approach.

Algorithm 1. Community Selection of ParCoDe on node v

// *masterNode* is a specific node of G ;

// *DeltaQMessage* is a triple of $(C_i, C_j, \Delta Q)$;

SuperStep 0:

1. search all the maximal ΔQ_{ij} and assign it to $\max \Delta Q$;
2. **sendMessage** (*masterNode*, **new** *DeltaQMessage*($\max \Delta Q$));
3. **voteToHalt**();

SuperStep 1:

4. **if** ($v = \text{masterNode}$)
 5. find the maximal $\max \Delta Q$ and the corresponding node set V_{\max} ;
 6. **end if**
 7. **for each** (node u in G)
 8. **sendMessage** (u , **new** *DeltaQMessage*(V_{\max}));
 9. **end for**
 10. **voteToHalt**();
-

Algorithm 2. Modularity Update of ParCoDe on node v

// *CMessage* is a set of nodes (u_1, u_2, \dots, u_k)

SuperStep 2:

1. **if** (v in V_{\max})
2. select the node with smaller id as *mergedNode*;
3. **sendMessage** (*mergedNode*, **new** *CMessage*(C_v));
4. **sendMessage** (*mergedNode*, **new** *DeltaQMessage*(v));
5. **else**
6. **if** (v is connected to $u, u' \in V_{\max}$)
7. $\Delta Q_{vu} = \Delta Q_{vu} + \Delta Q_{vu'}$;
8. **else**
9. $\Delta Q_{vu} = \Delta Q_{vu} - 2a_v a_{u'}$;
10. **end if**
11. **end if**
12. $a_u = a_u + a_{u'}$;
13. **voteToHalt**();

SuperStep 3:

14. add all received nodes to v ;
 15. **voteToHalt**();
-

An Approximation approach. The main idea of our approximation approach is to select more than two communities to be merged. For $\varepsilon > 0$, communities with ΔQ_{ij} satisfies $|\Delta Q_{ij} - \max \{\Delta Q_{ij}\}| \leq \varepsilon$ are selected. Such an approximation approach may lead to a penalty of accuracy. But as demonstrated in our algorithm, it shows that with an appropriate ε , the accuracy penalty could be neglected.

Experimental Evaluation

In this section, we evaluate our algorithm on both real and synthetic datasets. Details on the datasets are presented in Table 1. Our machine cluster consists of one master and ten slaves. Each machine runs the Ubuntu Linux 12.04 LTS and is equipped with a memory size of 16G, disk storage of 500G and AMD Opteron 4180 2.6GHz CPU.

Table 1 Details of Datasets

Dataset	Nodes	Edges	Description
Twitter	42M	1202M	Follow-Follower relationships on Twitter.
uk-2002	18M	298M	web pages of .uk domain
RMat	50M	250M	synthetic graph generated using R-MAT algorithm

Accuracy on varying ε . We shows how parameter ε affects the accuracy of ParCoDe compared with the native approach. As demonstrated in Figure 1, it shows that with larger ε , the accuracy of ParCoDe decreased. By our observation, ε should be chosen in the interval $[0.1, 0.15]$. As a default value, ε is set to be 0.125.

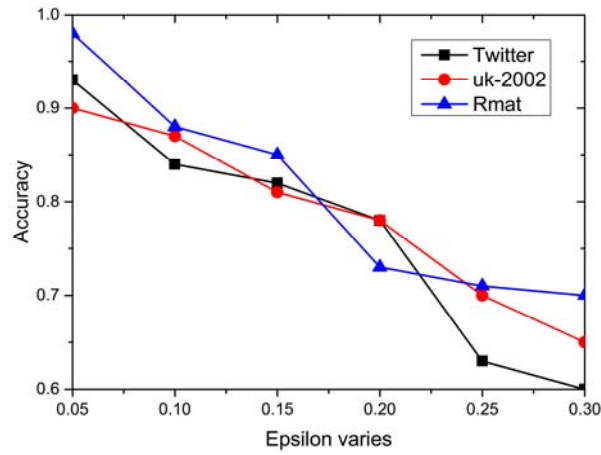


Figure 1. Accuracy of approximation approach as ε varies

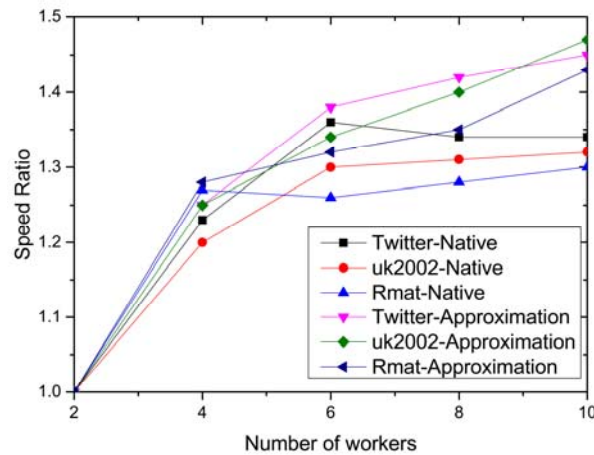


Figure 2. Speed ratio of ParCoDe as the number of workers grows

Scalability. We demonstrate the scalability of ParCoDe in Figure 2. It shows the performance of parallel community detection with the number of works raising from 2 to 10. We notice that for the native ParCoDe, the runtime is decreased sharply as the number of machines increased from 2 to 6. As the number of workers continually increased, the improvement increases slightly. As to the approximation approach, the scalability is obviously better.

References

- [1] Fortunato, Santo. "Community detection in graphs." *Physics Reports* 486.3 (2010): 75-174.
- [2] Lancichinetti, Andrea, et al. "Characterizing the community structure of complex networks." *PloS one* 5.8 (2010): e11976.
- [3] Gleiser, Pablo M., and Leon Danon. "Community structure in jazz." *Advances in complex systems* 6.04 (2003): 565-573.
- [4] Gulbahce, Natali, and Sune Lehmann. "The art of community detection." *BioEssays* 30.10 (2008): 934-938.
- [5] Korula, Nitish, and Silvio Lattanzi. "An efficient reconciliation algorithm for social networks." *Proceedings of the VLDB Endowment* 7.5 (2014): 377-388.
- [6] Hongladarom, Soraj. "Personal identity and the self in the online and offline world." *Minds and Machines* 21.4 (2011): 533-548.
- [7] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.
- [8] Moon S, Lee J -G, Kang M. Scalable community detection from networks by computing edge betweenness on MapReduce//Proceedings of the 2014 International Conference on Big Data and Smart Computing. Bangkok, Thailand, 2014:145-148.
- [9] Staudt C L, Meyerhenke H. Engineering parallel algorithm for community detection in massive networks. *IEEE transactions on Parallel and Distributed Systems*, 2015, IEEE Early Access Article:1-14.
- [10] Chen W -Y, Song Y, Bai H, Lin C, Chang Y. Parallel spectral clustering in distributed systems. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 2011, 33(3): 568-586.
- [11] Malewicz, Grzegorz, et al. "Pregel: a system for large-scale graph processing." *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010.
- [12] Newman M E J, Girvan M. Finding and evaluating community structure in networks. *Physical Review E*, 2004, 69(2): 026113.
- [13] Clauset A, Newman M E J, Moore C. Finding community structure in very large networks. *Physical review E*, 2004, 70(6):066111.
- [14] Leslie G. Valiant, A bridging model for parallel computation, *Communications of the ACM*, Volume 33 Issue 8, Aug. 1990.