CATS: A Big Network Clustering Algorithm based on Triangle Structures

Mincheol Shin[1, *], Jeongwoo Kim[1, *], Jungrim Kim [1]

Dongmin Seo [2], Chihyun Park [2], Seok Jong Yu [2], Sanghyun Park [1, †]

{smanioso, jwkim2013, kimgogo02}@yonsei.ac.kr, {dmseo, chihyun.park, codegen}@kisti.re.kr, sanghyun@yonsei.ac.kr

ABSTRACT

A huge amount of data, known as "big data," has been generated from various areas. A network is a popular data structure for presenting and analyzing big data. However, the conventional network analysis algorithms cannot cover the size of big data. To address this limitation, we propose in this paper a network clustering algorithm for a big data network using a parallel distributed computation model. To consider parallel computation concepts, we change the paradigm of the conventional clustering algorithm using triangle structures. We demonstrate that the proposed algorithm can cover a big data network that cannot be otherwise implemented using a conventional algorithm. Experimental results show that the proposed algorithm is faster than the conventional algorithm.

CCS Concepts

• Theory of computation → Design and analysis of algorithms → Distributed algorithms

Keywords

Network, Clustering, Parallel Distributed Computation, Big Data

1. INTRODUCTION

Recently, the term "big data" has been widely used to describe a huge amount of data. These data have been generated in several areas, such as biology, marketing, social media, and wireless sensor data analysis. Owing to the size of big data, conventional algorithms are unsuitable to process them. To address this challenge, several studies employed a MapReduce programming model as a big data analysis model. MapReduce is a parallel distributed computation model that employs map and reduce functions in the manner of batch processing. By using the MapReduce model, we can solve some big data problems. However, the MapReduce model is inefficient for an iterative mining algorithm because the model must read and write the result of each iteration, which requires a significant amount of time, for each iterative step. This issue can be solved by using Apache Spark as a parallel distributed model. Apache Spark applies a computing paradigm consisting of transformations of immutable data, which is called a resilient distributed dataset (RDD). By using RDD, Spark resolves limitations in the MapReduce computing paradigm. Spark additionally achieves an efficient fault-tolerance scheme using lineage, which is a sequence of transformations. When a failure occurs, Spark recalculates the failed RDD from ancestral RDDs with a lineage [1]. We can therefore design efficient iterative mining algorithms using Spark.

Meanwhile, a network can be used to describe many types of big data with edges and nodes. Furthermore, a network provides opportunities to analyze data by applying several analytical measures, such as centralities, which include degrees, closeness, betweenness, eigenvectors, clustering, and propagation. Of these measures, clustering is widely used to analyze a network in several fields. In the case of biology, we can infer protein modules by analyzing a protein–protein interaction network. In addition, social groups can be identified by using network clustering in the social network. For this reason, several studies have been conducted to analyze the network data by using network clustering algorithms.

In this paper, we propose the CATS clustering algorithm to analyze a big data network in Spark. The proposed algorithm is based on a clustering algorithm using structure similarity. The goal of this study is to develop a clustering algorithm to process big data in Spark. To this end, we designed a clustering algorithm using a parallel distributed concept. In addition, we changed the algorithm paradigm using a triangle structure.

The main contributions of this work include:

- Developing a clustering algorithm for big data network analysis in Spark.

- Using a triangle structure to calculate structure similarity.

[1] Department of Computer Science, Yonsei University, Seoul, Korea
Tel: +82-2-2123-7757
[2] Korea Institute of Science and Technology Information, Daejeon, Korea
Tel: +82-42-869-1796
[*] These authors equally contributed to this paper.
[†] Corresponding author

SAC 2017, April 03-07, 2017, Marrakech, Morocco

© 2017 ACM. ISBN 978-1-4503-4486-9/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3019612.3019893

2. Methods

In this section, we describe the CATS algorithm. Figure 1 shows the outline of the method, which consists of three steps.

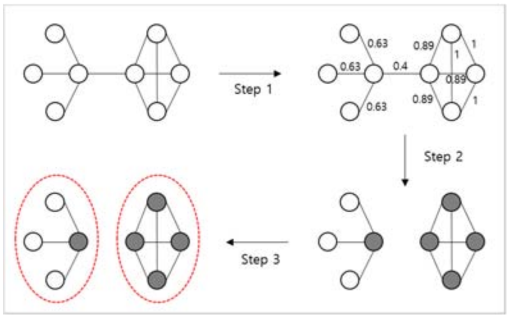

Figure 1. Outline of the CATS method

In the first step, the structure similarity between nodes is calculated based on the number of the common neighborhood. If node C has two edges that are linked with nodes A and B, node C is called a common neighborhood for nodes A and B. In the next step, the network is pruned by epsilon, which indicates the threshold for the structure similarity value. In the above figure, epsilon is 0.6. In this step, a node is assigned a label, which has a core and non-core. If a node has neighborhoods that have a greater structure similarity than epsilon, the node is considered a core; if not, the node is non-core. In the third network of the figure, the black node indicates a core, and the white node indicates a non-core. In the last step, the network is re-constructed by core and non-core information. Clusters are generated by finding the connected component in the re-constructed network.

2.1 Calculating Structure Similarity based on Triangle Structure

The similarity is calculated based on the neighbor node set, which includes itself. The equation is below.

$$\mathrm{Sim}(\mathrm{v},\mathrm{w}) = \frac{|Nei(v) \cap Nei(w)|}{\sqrt{|Nei(v)||Nei(w)|}}$$

In the equation, Sim(v, w) indicates the structure similarity between node v and w. Nei(v) denotes the neighbor nodes for node v. The values in the second network of Figure 1 indicate the similarity between nodes.

To calculate the similarity, the common neighbor nodes are needed. The common neighbor nodes can be extracted by considering the triangle structures in the network. Figure 2 shows the principle for extracting common neighbor nodes using triangle structures.

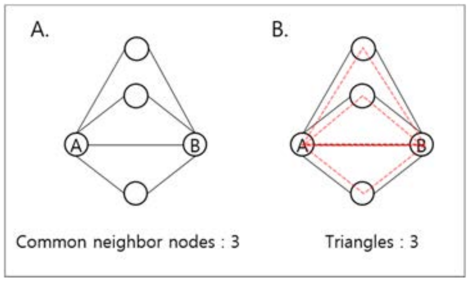

Figure 2. The example of triangle structure

As shown in Figure 2, we can find the number of common neighbor nodes between nodes A and B by using a triangle structure. The number of triangles, which includes the edge between nodes A and B, indicates the number of common neighbor nodes for nodes A and B. Therefore, we can calculate the structure similarity between nodes A and B using the triangle structure.

2.2 Network Re-Constructing

After calculating the structure similarity, we prune the edges using the epsilon value. Then, we assign a label for each node. If a node has neighborhoods with a structure similarity that is more than epsilon, the node is considered a core; if not, the node is non-core. After labeling, we filter the edges using core and non-core labels.

We select edges that include core nodes because the clustering step is conducted based on these nodes. The clustering rules are the following:

- The neighbors of a core node are assigned in the same cluster.

- If a neighbor of a non-core node is non-core, the edge is pruned.

Thus, the selected edges have two types. One is a core–core edge; the other is a core and non-core edge. Using the selected edges, we reconstruct the network in the next step.

2.3 Connected Component

We can generate clusters by finding the connected component in the reconstructed network. The connected component indicates the sub-network in which any two nodes are connected by paths. The edges between sub-networks do not exist.

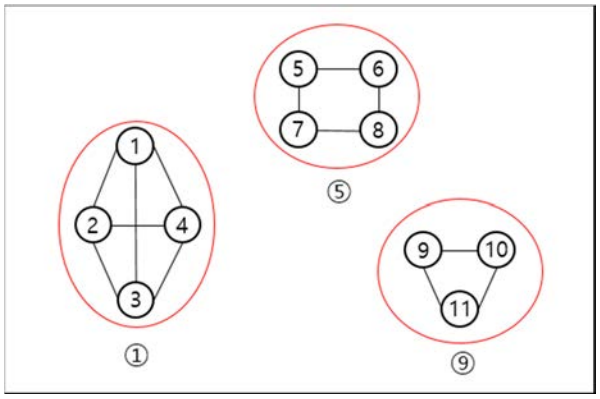

Figure 3. The example of connected component

Figure 3 shows the network with three connected components. In the connected component, a node label is assigned as the smallest node number among the nodes that are included in the same connected component. In the example, the number in the circle indicates the label of the connected components. The label is used as a cluster identifier, and the connected component is the cluster.

3. EXPERIMENTAL SETTING

All experiments were conducted on a cluster of five computers, which include one master cluster and four slave clusters. Each of the computers included quad-core 3.4 GHz processors and 32 GB of memory. Hadoop 2.7.2, Java 1.8.0, Scala 2.10, and Spark 1.6.0 were used in the experiments.

In addition, we employed a considerable amount of undirected network data. These are described in Table 1 [2].

Table 1. Dataset

Dataset	Node	Edge
Yeast-2	2,223	7,049
Drosophila	6,600	19,820
DIP	22,585	69,148
DBLP	317,080	1,049,866
As-skitter	1,696,415	11,095,298

4. RESULTS AND DISCUSSIONS

In this section, we describe the experimental results for the CATS algorithms. To validate our algorithm, we present comparison results for the proposed algorithm and a conventional clustering algorithm. We applied our algorithm to five datasets: Yeast-2 [1], Drosophila [3], DIP [6], DBLP [7], and as-skitter [4].

MCL is a well-known conventional clustering algorithm [5]. The algorithm was implemented based on a single machine. To validate the proposed algorithm, we present in Figures 4.

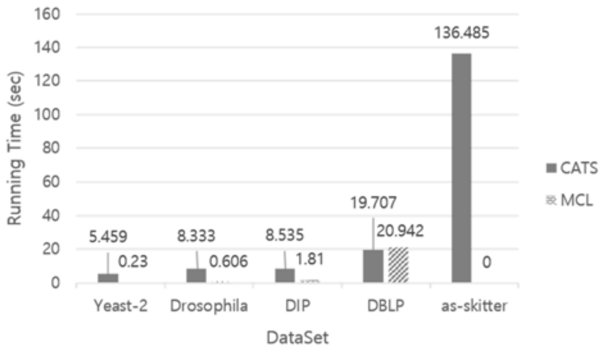

Figure 4. Running time for CATS and MCL

As shown in Figure 4, MCL is faster than CATS for three small datasets: Yeast-2, Drosophila, and DIP. The proposed algorithm requires partitioning and collection time to conduct parallel computation. When the dataset is small, the partitioning and collection time are longer than the clustering running time. In the case of the small dataset, the proposed algorithm is therefore slow. However, CAT is faster than MCL in the case of large datasets. Figure 4 shows that the proposed algorithm is faster than MCL for a big data network that has nodes (>300 k) and edges (>1 M). Furthermore, we confirmed that MCL is not conducted by a memory limitation in the case of as-skitter data.

5. CONCLUSIONS

In this paper, we proposed the CATS algorithm, which is a network clustering algorithm for a big data network in Spark. To consider parallel distributed computation, the paradigm of the algorithm is changed. We applied our algorithm to several network datasets and demonstrated that CATS is a more useful algorithm than a conventional algorithm for clustering in a big data network. CATS was shown to be an effective algorithm for analyzing a big data network. Owing to limitations of the experimental environments, we conducted experiments using five computers. In future work, we will use more computers for experiments that compare CATS to state-of-the-art algorithms. It is expected that the experimental results will be better than the current ones. Furthermore, directed network clustering is additionally important for analyzing network data. For this reason, we will strive to design a directed network clustering algorithm for a big data network.

6. ACKNOWLEDGMENTS

This work was supported by the “Development of biomedical data network analysis technology based on high performance computing for dementia researches (K-16-L03-C02-S02)” funded by Korea Institute of Science and Technology Information.

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (NRF-2015R1A2A1A05001845).

7. REFERENCES

[1] Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., ... & Li, G. (2003). Topological structure analysis of the protein–protein interaction network in budding yeast. Nucleic acids research, 31(9), 2443-2450.

[2] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[3] Giot, L., Bader, J. S., Brouwer, C., Chaudhuri, A., Kuang, B., Li, Y., ... & Vijayadamodar, G. (2003). A protein interaction map of Drosophila melanogaster. science, 302(5651), 1727-1736.

[4] Leskovec, J., Kleinberg, J., & Faloutsos, C. (2005, August). Graphs over time: densification laws, shrinking diameters 0and possible explanations. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (pp. 177-187). ACM.

[5] Van Dongen, S. M. (2001). Graph clustering by flow simulation.

[6] Xenarios, I., Fernandez, E., Salwinski, L., Duan, X. J., Thompson, M. J., Marcotte, E. M., & Eisenberg, D. (2001). DIP: the database of interacting proteins: 2001 update. Nucleic acids research, 29(1), 239-241.

[7] Yang, J., & Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. Knowledge and Information Systems, 42(1), 181-213.