Efficient MapReduce Implementation on a Graph Algorithm

Cheng Chen[2], Chenyi Gu[1], Michela Taufer[2]
[1]Department of Physics and Astronomy, University of Tennessee, Knoxville
[2]Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville

***Abstract* – Para-clique is a graph theory algorithm that can impute the potential new link by analyzing the dense subgraphs. Speeding up para-clique algorithm can largely extend researcher's ability to reach out the biological mechanism information. In this project, we are tackling two problems:**

1. **How MapReduce speed up the para-clique algorithm;**
2. **How clique size influence the algorithm running time.**

Our results show that efficient implementation of MapReduce will speed up the para-clique algorithm, especially when working with large cliques set.

***Keywords* – Clique; Para-clique; MapReduce; PySpark**

I. Introduction

Clique is a graph where every vertexes are connecting to each other. In our implementation, we only consider unweighted and undirected graph for simplicity. Maximal clique is a type of dense sub-graph and cannot be included in a larger clique. Vertices in a dense sub-graph are normally strong related with each other.

In life science area, studying dense subgraphs is a widely used and effective way to reveal the biological mechanism behind the complicated large data. For instance, instead of working alone, proteins normally collaboration to perform a job. Previous researchers finding the hierarchy and organization of biological processes and cellular components by analysing the densely interconnected proteins clusters [1]. Another example is that in drug-target network, the drugs in a same clique may have the same targets [2].

Paraclique is a densely-connected subgraph where every vertexes are connecting to each other except a small number of edges [3]. The number of these unconnected vertexes (member vertexes of clique) is called glom term. Examples of Para-clique are shown in Fig 1.

Paraclique algorithm can impute the potential new link by the original cliques. It can be used in researches such as gene-disease relationship analysis. The previous graph theory algorithm used for finding paraclique (paraclique algorithm) is augments a maximum clique [3], which is the maximal clique in the graph with largest size. One of the reasons of the thought para-clique is only applied on maximum clique and is not applied on maximal cliques is that, the number of maximal cliques in a graph can be enormously and paraclique process requires long computing time.

However, the maximal cliques have no less information than maximum clique and should not be omitted. If we could develop a implantation that can finish para-clique on a large number of cliques quickly, the researchers could dig more results from the data. Therefore we are going to implement MapReduce which enables parallelization across large amount of servers in a Hadoop cluster.

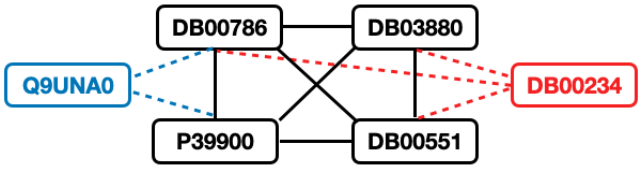

Fig. 1. Para-clique examples. Four black squared vertexes "DB00786", "P39900", "DB03880" and "DB0051" are combined into one clique. Vertex "Q9UNA0" is a para-clique result of clique with glom = 2 since vertexes "DB0051" and "DB03880" in the clique does not connect to "Q9UNA0". Vertex "DB00234" is a para-clique result of the clique with glom = 1 because only "P39900" in the clique does not connect to "DB00234".

II. Preprocessing

The raw data is downloaded from Stanford Biomedical Network Dataset Collection [4]. We used ChG-Miner(drugs-genes), DCh-Miner(diseases-drugs), andDG-Miner(disease-genes) three networks. The infection disease vertices in these three networks and they connected genes and drugs vertices made our input graph. The clique dataset of the input graph contains 89,811 cliques with imbalanced size (number of nodes inside one clique) as shown in Fig 2.

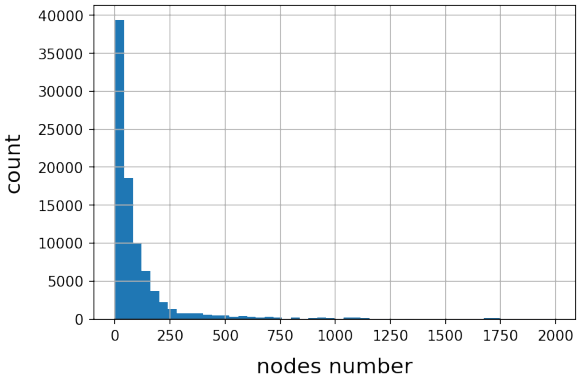

Fig. 2. Distribution of clique sizes. 70% cliques of total cliques is smaller than 100, 20% of total is in between 100 and 250, and 10% of total is bigger than 250.

We divide the clique dataset into 3 parts – small, middle, large dataset – based on the cliques sizes:

- Small: clique size $n < 100$;
- Middle: clique size $100 \leq n < 250$;
- Large: clique size $n \geq 250$.

We will run algorithms with respect to each subset and study how clique size will influence the para-clique algorithm running time.

III. Methods: MapReduce

We try three different paraclique implementations, the workflows of each implementation is shown in Fig 3:

1. WithoutMR: sequentially apply paraclique on each clique;
2. ForMR: applied MapReduce to parallel compute the clique set, and the process of each individual clique is the same with WithoutMR;
3. MR: transform cliques rdd to clique nodes rdd and applying Para-clique on every single nodes parallel.

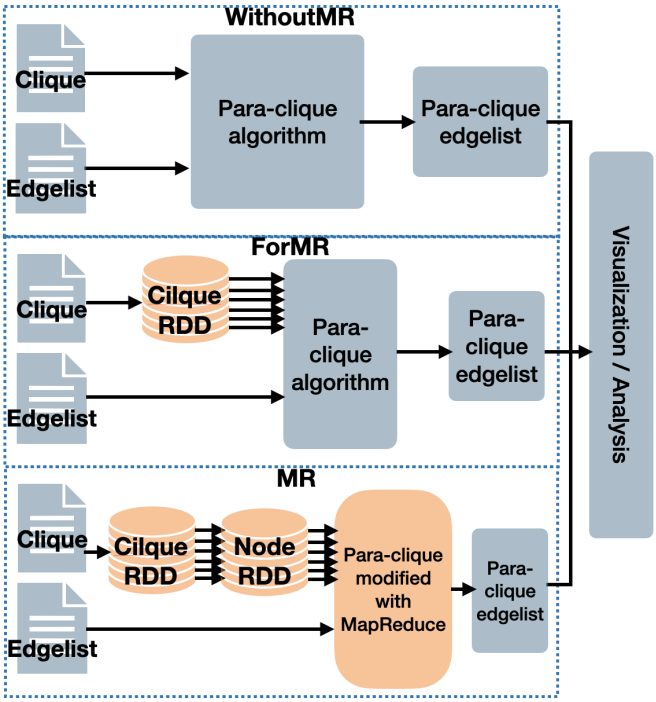

Fig. 3. Workflows of three para-clique algorithms.

IV. Results

To see how clique size will influence the algorithm running time. We randomly select 10, 100, 200, 500, 800 cliques from small, middle, and large clique sets respectively. For each target (with different number of cliques and clique size), we repeated 10 times to analysis the expectation and variation of running time.

The running time of our test are shown in Fig 4. First of all, all three implementations work in exploring paracliques and provide the same paraclique results. For all three implementations, the larger the clique size, the longer time is needed, but the larger the input clique size and clique number are, the more efficient MapReduce implementation behaves. For example, in large dataset calculation, ForMR and MR algorithm outperform than WithoutMR when number of cliques as 100 and 800 respectively. While in small/middle dataset calculation, WithoutMR and MR have similar behaves.

ForMR is outperform than MR algorithm which is out of our expectation. This can be interesting topic for further research. The potential reasons could be

- we use several actions that some transformations may be recomputed;
- para-clique algorithm without MapReduce is efficient that makes no enough difference to small or large cliques or our cliques are too small.

In conclusion, we successfully implement MapReduce algorithm in para-clique algorithm, and efficient MapReduce implementation will speed up the para-clique algorithm, especially when we are working with large cliques set.

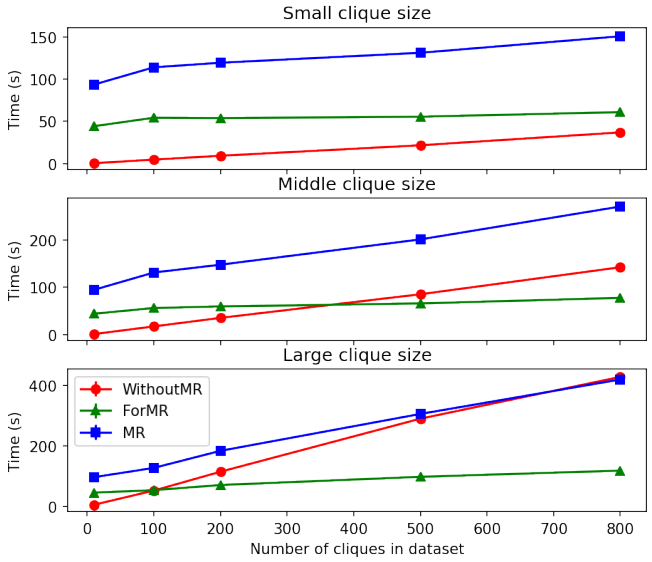

Fig. 4. Results of the running time for small/middle/large dataset with three algorithms as a function of the number of cliques in each running sample dataset. Average and standard deviation are calculated with 10 random sample datasets.

REFERENCES

[1] Xiuli Ma, Guangyu Zhou, Jingbo Shang, Jingjing Wang, Jian Peng, and Jiawei Han. Detection of complexes in biological networks through diversified dense subgraph mining. *Journal of Computational Biology*, 24(9):923–941, 2017.

[2] Cheng Chen, Stephen K. Grady, Sally R. Ellingson, and Michael A. Langston. Gene-disease-drug link prediction using tripartite graphs. In *Proceedings of the 12th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '21, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Elissa J. Chesler and Michael A. Langston. Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data. In *Proceedings of the 2005 Joint Annual Satellite Conference on Systems Biology and Regulatory Genomics*, RECOMB'05, page 150–165, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] Sagar Maheshwari Marinka Zitnik, Rok Sosič and Jure Leskovec. BioSNAP Datasets: Stanford biomedical network dataset collection. http://snap.stanford.edu/biodata, August 2018.