Efficient MapReduce Implementation on a Graph Algorithm

Cheng Chen [1] Chenyi Gu [2] Michela Taufer [1]

[1]*Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville*
[2]*Department of Physics and Astronomy, University of Tennessee, Knoxville*

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Abstract

We are tackling two problems:

1. How we speed up the para-clique algorithm with MapReduce;
2. Study the influence of cliques size onto the running time.

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

Introduction

Clique is a graph where every vertexes are connecting to each other. Maximal clique cannot be included in a larger clique. Maximum clique the maximal clique in the graph with largest size. For simplicity in our implementation, we only consider unweighted and undirected graph.

Para-clique is a graph theory algorithm that augments a maximum clique with non-member vertices adjacent to all but a specified number of member vertices, known as the glom term.[1]

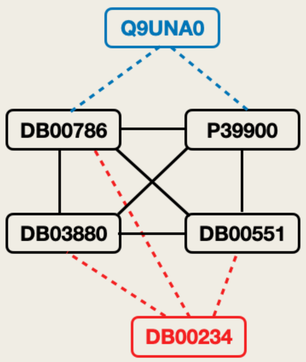


Figure 1. para-clique examples

Clique example:
As shown in Fig. 1, the black squared vertexes ”DB00786”, ”P39900”, ”DB03880” and ”DB0051” are one clique.

Para-Clique example:
As shown in Fig. 1, vertex ”Q9UNA0” is a para-clique result of clique with glom $= 2$, vertex ”DB00234” is a para-clique result of the clique with glom $= 1$ because only P39900 in the clique does not connect to DB00234.

The raw data is downloaded from Stanford Biomedical Network Dataset Collection[2]. The Clique dataset contains around $90,000$ cliques with imbalanced size of each cliques (number of edges inside one clique) as shown in Fig. 3.

Significant of speeding up Paraclique:

- Cliques contain precious information like indication of strongly related targets. And Para-clique can impute the potential new link attach to the cliques.
- Previously Para-clique was normally used on maximum clique instead of maximal cliques since the number of maximal cliques can be enormously.
- While the maximal cliques also contains precious information and should not be omitted. So a speedup implantation can extend the algorithm application range.

Method / Preprocessing

MapReduce: We try three different para-clique algorithms:

1. WithoutMR: sequentially apply Para-clique on each clique;
2. ForMR: applied for loop and 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.

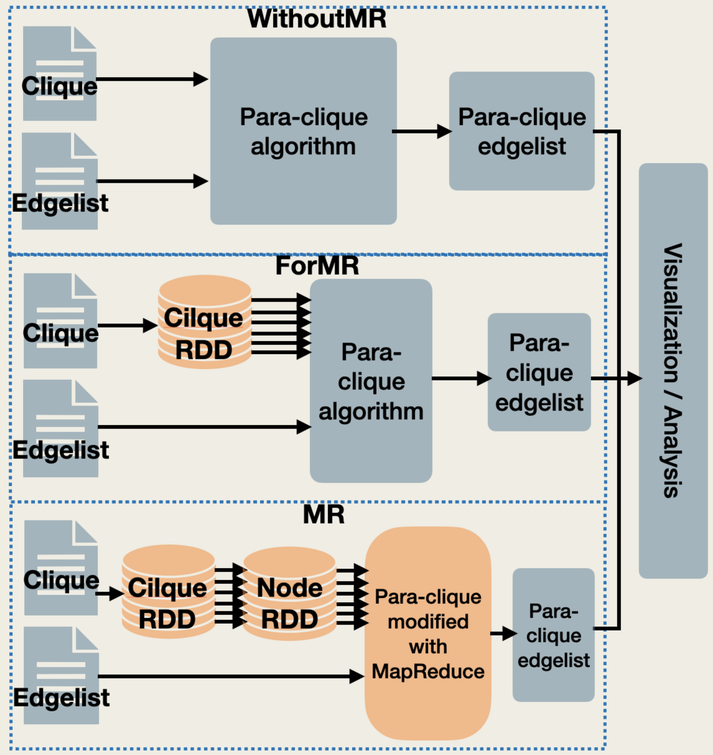
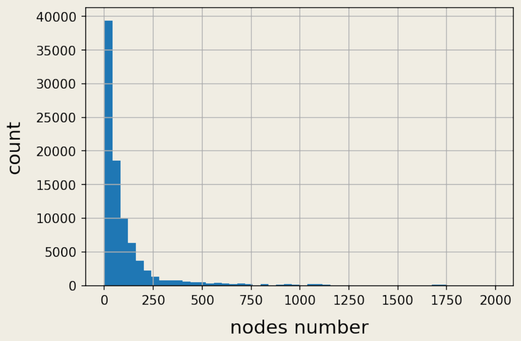


Figure 2. Workflows of three para-clique algorithms.

We divide the clique dataset into 3 sample groups based on the cliques sizes.

- Small (70%): clique size $n < 100$;
- Middle (20%): clique size $100 \leq n < 250$;
- Large (10%): clique size $n \geq 250$.

Figure 3. Distribution of clique sizes.



Results

- The larger the input clique size and clique number are, the more efficient MapReduce implementation behaves;
- For large dataset, ForMR and MR algorithm outperform than WithoutMR when number of cliques as 100 and 800 respectively. For small/middle dataset, 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.

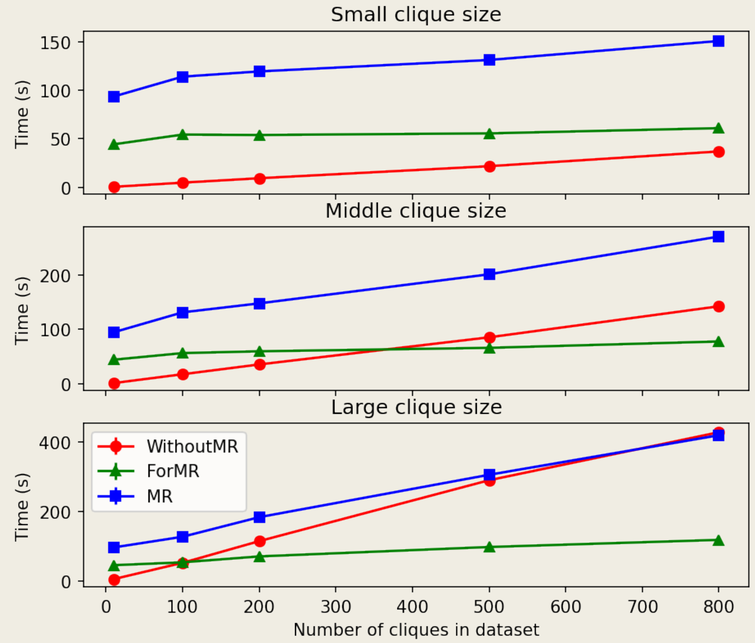


Figure 4. Results of the running time for small/middle/large dataset with three algorithms as a function of the number of cliques in each running sample dataset. Average and standard deviation are calculated with 10 random sample datasets.

Conclusion

1. Efficient MapReduce implementation will speed up the para-clique algorithm, especially working with large cliques set.
2. Our ForMR algorithm works better than MR algorithm. The potential reasons are
   - we use several actions that some transformations are recomputed;
   - para-clique algorithm without MapReduce is efficient that makes no enough difference to small or large cliques or our cliques are too small.

References

[1] 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.

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