nty paper 11 12 13 14 DynPar StrHyp ProStr

Database Technology [2ID35]

Project part 2

Michiel Fortuin — 0812105
Fengjun Wang — 0925350
Ferry Timmers — 0637586
Myrthe van Delft — 0657742

May 20, 2015

1 Introduction

As part of the database and technology course of 2014-2015, we have to investigate a research paper from the scientific community. For our project we were assigned the paper Clustering Streaming Graphs[1]. This paper presents an algorithm for 'online' clustering of (large) streaming graphs. In this first progress report we will present a brief literature survey, the main outline of the results of the paper, and a rough estimate of the planning of the project. We may refer to [1] as 'the paper' or 'our paper'.

2 Literature Survey

In the paper, a method for clustering online, streaming graphs is presented. A streaming graph is a graph where the updates to the graph are given in the form of a *stream* of edge or vertex additions or deletions. Handling of such rapidly changing graphs is challenging because of its dynamic, online nature and massive scale. An algorithm thus would necessarily need to be incremental and extremely fast, and preferably amenable to parallel or distributed implementations.

The graph clustering problem has been a subject of extensive research, but mostly assuming an *offline* setting where the entire graph is given beforehand. The paper presents an algorithm which is suitable to an online setting, and is capable of maintaining a decent graph clustering while updates are performed on the graph.

In [3] random sampling was used to solve this problem in a offline fashion. This random sampling was first used in this paper. In [4] solves the same problem, but then in an online fashion, using an Erdös-Réyni model. Problem with this algorithm in [4] is that it does not allow deletion or modification of the graph. Also it does not scale well with

the number of clusters. The algorithm presented in [5] solves the same problem in an online fashion. This hash-compressed micro-clusters and find structurally similar graphs in a stream of large number of small graphs. In [6] reservoir sampling is also used. This algorithm is called the AZY algorithm. Our paper is based on top of this algorithm. It can be seen as an extension. The AZY does not consider deletions in the graph.

In [8] the algorithm presented in our paper is used for analyzing social network. The weighted graph is used, which is based on the chat frequency and access recency. In [9] the algorithm is adapted for use with hypergraphs. The paper [10] describes another clustering algorithm similar to ours, only the query's aren't about a snapshot of the graph, but deletions in the past and additions in the future are taken into account with the clustering.

3 The paper

The research problem is clustering large-scale and rapidly changing "streaming" graphs where the updates to a graph are given in form of a stream of vertex or edge additions and deletions.

The algorithm in the paper is proposed to solve the clustering problem and also satisfies 3 optimization requirements:

a it should be simple to adjust to parallelizations and distributed environments

b it should handle massive inputs, with a relatively low overall space complexity

c it should handle high throughput streams, with a relatively low time complexity per update or query

To test the algorithm, it is tested against two other clustering algorithms. The first is METIS, a standard clustering algorithm, with some slight modifications to make it suitable to an 'online' environment. The second is AZY, a clustering algorithm which is suited to 'online' environment, but with expected much worse performance. The tests are performed on 4 different datasets:

1 cit-HepPh, which shows a large increase in the number of vertices over time. This graph has (in total) $34k$ vertices and $420k$ edges.

2 web-NotreDame. This graph has (in total) $330k$ vertices and $1.5M$ edges.

3 replies, a very sparse and not well clustering graph. This graph has (in total) $1.9M$ vertices and $1.6M$ edges.

4 DNS Edges, containing many duplicate edges but with different time stamps. This graph has (in total) $180k$ vertices and $4.8M$ edges.

3.1 Results

The performance of the algorithms is measured in two different aspects: quality and performance. Furthermore, some tuning experiments with adjusted p-values are done.

3.1.1 Quality

The paper tests the quality of the 4 datasets on the Structural Sampler, METIS, and AZY, with different clustering group size bounds, while fixing the sampling threshold parameter p at 1. The results show that the cut-size quality of the Structural Sampler is almost as well as METIS, which is assumed to be close to the best clustering. AZY performance significantly worse than both METIS and the Structural Sampler.

Furthermore, the paper tests the cut-size quality of the Structural Sampler, METIS, and AZY on the cit-HepPh and DNS Edges datasets, using different p-values (the sampling threshold). For both METIS and the Structural Sampler, applying a lower sampling threshold could bring a worse clustering quality. However, the AZY is mostly uninfluenced by a lower sampling threshold.

3.1.2 Performance

The paper tests the performance of both the Structural Sampler and METIS on cit-HepPh and web-NotreDame, using different ratios for the number of queries versus the number of updates. The throughput of the Structural Sampler out-performs METIS, increasingly so with an increasing query vs. update ratio, up to more an improvement more than 3 orders of magnitude.

Furthermore, the paper tests Structural Sampler and METIS, with different clustering size bounds B, while fixing Query/Update-ratio at 0.5, on the datasets DNS Edges and web-NotreDame. The throughput of Structural Sampler out-performs METIS with more than 3 orders of magnitude. However, METIS has almost the same throughput for different values of the clustering size bounds B, while the performance of the Structural Sampler decreases as B increases.

Finally, the paper tests the throughput of Structural Sampler and METIS on cit-HepPh and DNS Edges, while adjusting the p-value. The effect of the sampling threshold on METIS is very limited but it does give the same trend as the Structural Sampler approach. Both show that a lower sampling threshold increases performance.

4 Planning

The planning of our project looks as follows. Depending on whether or not we receive the source code from the author, we will either execute with 3a (if we do receive the source code), or 3b (if we do not receive the source code).

1 Mail the authors with a request for the source code. *Completed*

2 Obtain the datasets used in the paper, as far as possible. *Completed*

3-a If we are capable of obtaining the source code from the authors, then:

- i Compare the performance and quality results presented in the paper with those of METIS and our implementation.

3-b If we do not receive the source code from the authors in a timely manner, then:

- i Implement the algorithm presented in the paper.
- ii Compare the quality results presented in the paper with those of METIS and our implementation.
- iii *Expansion:* Compare the performance results presented in the paper with those of METIS and our implementation.

4 Possible expansions, depending on the time it takes to implement and run the experiments:

- i Repeat the tests on different datasets. An interesting candidate is a dataset showing no obvious clustering.
- ii Testing for additional p-values aside from those used in the paper, and verifying the claims made about higher and/or lower p-values.
- iii The paper claims the algorithm can be easily parallelized or run distributed. We are interested to test the method presented for parallelizing the algorithm.

4.1 Time Schedule

Here, we list the dates and deadlines we set out for ourselves, to ensure successful completion of the project.

Date	Deadline
13 May	Finish scheduling report
24 May	Untested, but completed implementation of the algorithms Completed preprocessing the datasets Project part 3
31 May	Completed and debugged implementation of the algorithms *Start testing and preliminaries of the final report*
7 June	Start discussing final presentation
14 June	Finished main experiments
21 June	Finished final report (project part 4)

4.2 Progress 13-05

Currently, we have e-mailed the author and have been able to obtain three of the four datasets used in the paper. The fourth dataset is from an IBM process, and we do not believe it to be possible to gain access to this dataset.

Furthermore, we have received a negative response from the author, and thus we will need to create our own implementation of the presented algorithm. However, we have been able to find an implementation of the METIS algorithm, and will use this implementation as a bench mark, and not write our own.

Started on the implementation of the algorithm and the context of the algorithms.

4.3 Progress 25-05

4.4 Tools

We have decided to use git as a code repository for the project, and we will write our code in C++. The development environment we will use is Visual Studio. For writing documents and such we will use LaTeX. We might use additional benchmarking tools. Any additional datasets we will we need, we believe to be able to find on the internet.

References

[1] A. Eldawy, R. Khandekar and K. Wu, "Clustering Streaming Graphs", in *32nd IEEE International Conference on Distributed Computing Systems*, 2012.

[2] A. Jain and R. Dubes, "Algorithms for Clustering Data". Prentice-Hall, 1988.

[3] D. Karger, "Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm", in *ACM-SIAM Symposium on Discrete Algorithms*, Austin, TX, Jan. 1993, pp. 21-30.

[4] H. Zanghi, C. Ambroise and V. Miele, "Fast Online Graph Clustering Via Erdos-Rényi Mixture", *Pattern Recognition*, vol. 41, no. 12, pp. 3592-3599, 2008.

[5] C. Aggarwal, Y. Zhao and P. Yu, "On Clustering Graph Streams", in *Proceedings of the SIAM International Conference on Data Mining*, Columbos, OH, Apr. 2010.

[6] —, "Outlier Detection in Graph Streams", in *Proceedings of the Internation Conference on Data Engineering, ICDE*, Hannover, Germany, Apr. 2011, pp. 399-409.

[7] G. Karypis and V. Kumar, "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", in *SIAM Journal of Scientific Computing*, vol. 20, no. 1, pp. 359-392, 1998.

[8] M. Yuan, "Dynamic Partitioning of Social Networks", University of Illinois, Urbana, Illinois, 2012.

[9] G. Wing, "Streaming Hypergraph Partition for Massive Graphs", Kent State University, Dec. 2013.

[10] m. Yuan, K. Wu, Y. Lu and G. Jacques-Silva, "Efficient Processing of Streaming Graphs for Evolution-Aware Clustering", in *CIKM '13, Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pp. 319-328, New York, 2013