# FID3008 Advanced Topics in Distrbuted Systems

Susanna Pozzoli

April 2020

- **Task 4:** write a critical review of the papers that covers in particular the summary of contributions, solutions, significance, and technical/experimental quality.

## 1 Introduction

[LXG14] *Large-Scale Frequent Subgraph Mining in MapReduce*,

[Zha+16] *Frequent Subgraph Mining Based on Pregel*, and

[RZ16] *Leveraging Multiple GPUs and CPUs for Graphlet Counting in Large Networks*.

[LXG14], [Zha+16], [RZ16] are distinct distributed algorithms that address the issue of Frequent Subgraph Mining (FSM), which has large numbers of applications, such as chemistry and analysis of social networks.

While [Mal+10] and [Zha+16] focus on subgraph isomorphism, [RZ16] concentrates on induced subgraphs, as detailed below.

It is worth remembering that both vertices and edges are labeled in case of [Mal+10] and [Zha+16].

## 2 *Large-Scale Frequent Subgraph Mining in MapReduce*

[LXG14] is one of the first to present a MapReduce-based algorithm that is able to mine frequent subgraphs whose support is greater than $\theta$, where $\theta$ is the threshold, from a collection of small graphs, unlike [Zha+16], [RZ16], and the majority of the FSM algorithms, which are designed for single large graphs. Unlike previous work, MapReduce Frequent Subgraph Mining (MRFSM) requires a fixed number of steps.

There are three steps in MRFSM, each of which uses MapReduce.

1. *Filter.* Graphs are divided into partitions, one partition per worker. Workers determine which subgraphs are probable to be frequent.

2. *Sorting.* Sort keys, which are subgraphs, and values, which are frequencies, by subgraph size.

3. *Refinement.* Workers compute the local and then the global support of the subgraphs. It is possible to use either a top-down or a bottom-up methodology to mark the bounds of the subsets of graph that it is necessary to search, to decrease the number of subgraph isomorphism tests.

Unfortunately, there is a risk of a large number of false positives, because it is possible for subgraphs to be frequent at local but infrequent at the global level. One of the contributions of [LXG14] is to predict whether subgraphs that are locally frequent are globally frequent, too, by binomial distribution.

### 2.1 Strengths and Weaknesses

S Based on MapReduce, which is widely adopted.

S Authors made one first attempt to filter the subgraphs in order that disadvantages, *i.e.*, false positives, may be highlighted.

S Communication overhead is decreased by including an optimization based on compression of canonical labelling.

W It is not possible to extend MRFSM to graphs that do not fit in memory.

## 3 *Frequent Subgraph Mining Based on Pregel*

[Zha+16] address FSM in the hypothesis that there is a single (labeled) graph. As the title implies, the difference between [Mal+10] and [Zha+16] is that the latter starts from Pregel [Mal+10], which is a framework for graph processing. However, by its vertexcentric nature, Pregel poses a number of challenges to FSM. This is why it is necessary to extend Pregel itself, to make it work.

Vertices apart, there are $n$ workers in this case and only one master. It is responsible for the big picture, which is derived from aggregation of data at worker level. Apache Giraph, which is the open-source equivalent of Pregel, offers aggregators, by default.

Basically, the idea behind pegi is to alternate between *pattern growth* and *embedding discovery*, iteratively. While master orchestrates, workers are used to compute the number of subgraphs at the local level. Vertices employ message passing to explore their neighborhood.

### 3.1 Strengths and Weaknesses

S Optimizations allow them to decrease overheads and to minimize the number of supersteps and thus of synchronization barriers.

W Unfortunately, Pregel is not good with tasks that imply structure. See [Gao+14]. This is why it is necessary to hack Pregel in order that vertices may be either masters or workers.

W pegi is compared to the state of the art of 2014, *i.e.*, GrapMi [Els+14], which is not distributed.

# 4 *Leveraging Multiple GPUs and CPUs for Graphlet Counting in Large Networks*

Graphlets are, by definition, induced subgraphs. As such, graphlet count is a specialization of FSM.

Authors present an algorithm that is able to count the number of connected and disconnected graphlets of size $k = \{2, 3, 4\}$ at the edge level and at the graph level. It is uncommon to count the number of disconnected graphlets, since they are, by definition, connected subgraphs.

One of the contributions of this article is to demonstrate that edgecentrism enables them to do a better load balancing.

Both CPUs and GPUs, with their pros and cons, are used to count how many graphlets are present. Networks are known to show a degree distribution that is a power law, and as a result it makes sense to assign edges with a large number of neighbors to CPUs and thus to assign edges with a small number of neighbors to GPUs.

$$\Pi = \{\underbrace{e_1, \ldots, e_k,}_{\Pi_{CPUs}} \overbrace{e_{k+1}, \ldots, e_{j-1},}^{\text{unprocessed } (j-k)-1} \underbrace{e_j, e_{j+1}, \ldots, e_M}_{\Pi_{GPUs}}\} \text{ [RZ16]}$$

Edges are in descending order of difficulty. While GPUs pop from the tail, CPUs pop from the head of the job queue, *i.e.*, $\Pi$.

For each edge, *i.e.*, $e_i$, they compute $T$, which is the set of vertices that are in triangles with $e_i$, and $S_u$ is the set of vertices that are in two-stars, whose center is $u$, with $e_i$. Given $T$ and $S_u$, it is possible to derive the number of graphlets at the edge level and at the graph level.

## 4.1 Strengths and Weaknesses

S Triangles and two-stars are sufficient for graphlet count.

S Authors present a series of algorithms for CPUs as well as a series of algorithms for GPUs, which are designed to leverage their respective devices. For example, hash algorithms are used for CPUs and search algorithms are used for GPUs.

S Thanks to $b_{CPUs}$ and $b_{GPUs}$, which are batch sizes, it is possible to switch from single GPU, to multi-GPU, and to hybrid multi-core CPU-GPU.

W It is parallel but not distributed, and it may be hard to share $\Pi$ with $n$ machines.

W It is trivial that the number of connected graphlets of size 2 is equal to the number of edges, *i.e.*, $M$, and the number of disconnected graphlets of size 2 is equal to the complementary of $M$.

# References


[Els+14] Mohammed Elseidy et al. "GraMi: Frequent Subgraph and Pattern Mining in a Single Large Graph". In: *Proceedings of the VLDB Endowment* 7.7 (2014), pp. 517–528.

[Gao+14] Jun Gao et al. "Continuous Pattern Detection over Billion-Edge Graph Using Distributed Framework". In: *2014 IEEE 30th International Conference on Data Engineering*. 2014, pp. 556–567.

[LXG14] Wenqing Lin, Xiaokui Xiao, and Gabriel Ghinita. "Large-Scale Frequent Subgraph Mining in MapReduce". In: *2014 IEEE 30th International Conference on Data Engineering*. 2014, pp. 844–855.

[Mal+10] Grzegorz Malewicz et al. "Pregel: A System for Large-Scale Graph Processing". In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. 2010, pp. 135–146.

[RZ16] Ryan A. Rossi and Rong Zhou. "Leveraging Multiple GPUs and CPUs for Graphlet Counting in Large Networks". In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2016, pp. 1783–1792.

[Zha+16] Xiang Zhao et al. "Frequent Subgraph Mining Based on Pregel". In: *The Computer Journal* 59.8 (2016), pp. 1113–1128.