ANALYSIS OF GRAPH PARTITIONING ALGORITHMS

Zishi Deng[1] Torsten Suel[2]

1. Undergraduate Research Intern at NYU 2.Professor at NYU Tandon School of Engineering

Abstract

In this project, we study several Graph partitioning methods proposed in recent papers, in particular, Balanced Label Propagation [1], LEOPARD [2], a Bayesian Sharding approach [3], and METIS [4]. We implement these algorithms and run them on several large social graphs including LiveJournal, Orkut, and Pokec, each with millions of nodes and tens of millions of edges. We then compare and contrast the advantages and drawbacks for each algorithm. Finally, we further optimize the algorithms to improve partition quality.

Importance of Graph Partitioning

Social network like Facebook, Google+ and Twitter have large amounts of user data that make their social graphs impossible to be stored on a single machine. Thus, companies have built large distributed systems to store these graphs, and to run queries on them. However, due to bandwidth constraints and communication overheads, querying nodes across machines takes significantly more time than querying nodes locally. Hence, to minimize communication costs, data needs to be partitioned such that the total number of edges cutting across partitions is minimized while also satisfying constraints on the maximum amount of data that can be stored on each node.

Approaches

1.Balanced Label Propagation (BLP)

- An iterative approach
- Move as many nodes as possible to maximize gain in colocation count (number of neighbors in the same shard) while respecting the constraints of each partition size.
- Randomly initialize partition by applying an integer hash to nodeID and then taking modulus of number of partitions k.
- Sort increase in colocation count when moving each node from partition i to j, representing how willing a node wants to be placed into partition j.
- Solve a linear program with size constrains to decide how many nodes should be relocated.

2.Lightweight Edge-Oriented Partitioning and Replication for Dynamic Graphs (LEOPARD)

- A dynamic graph partitioning approach
- A scoring function to calculate the attractiveness of each partition
- Assign new node to the partition with the highest score
- Periodically re-examine the partition of the incoming node and propagate to all its neighbors
- Replication was allowed to ensure fault tolerance and to improve edge locality

3. Bayesian Sharding Approach (BSA)

- Stochastic Block Model (SBM) based on probability concepts
- Iteratively update community assignment via a weighted, discounted vote over their neighbors' membership until convergence
- Assign the whole community to shard using a greedy algorithm
- Replication of local node with the most number of incoming edges

* Predicted Results	BLP	LEOPARD	BSA	METIS
Static or Dynamic	S	D	S	S
Replication allowed	N	Y	Y	N
Iterative approach	Y	N	Y	Y
Partition Quality	✓✓	* ✓✓	* ✓✓	✓✓✓
Running time	ϟ	* ϟϟ	* ϟϟ	ϟϟϟ

Methodology

We implement randomly initialized Balanced Label Propagation using c++ and python. It consists of 4 sub-programs: Random Initialization, Linear Program Producer, Apply Move, and Bokeh Graph Plot, glued together in a bash shell script. We test the program with Orkut social graph, which contains 3.1 Million nodes and 117 Million edges, on MacOS Sierra v10.12 using 2.9 GHz Intel Core i5 CPU and 8 GB 2133 MHz LPDDR3 Memory. Due to the massive input, each iteration takes roughly 2 minutes to complete.

Result shows that there is an diminishing return in running Balanced Label Propagation (BLP) across iterations and the final result converges to a fairly good locality ratio. The graphs below were produced by running BLP on Orkut social graph into 5 partitions for 15 iterations with a partition size leniency f = 0.10. The result shows that there is a significant decrease in node movement and an increase in edge locality for the first few iterations and then slowly converges.

Figure 2: Decrease of nodes movement

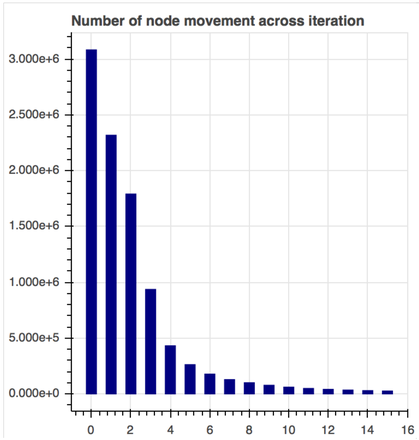
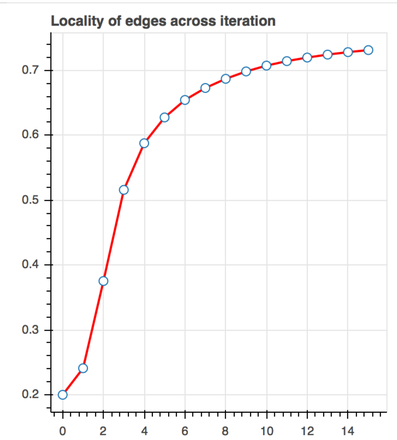


Figure 3: Improvement of locality of edges



Future Work

We will test Balanced Label Propagation (BLP) on top of METIS, as we see the potential of combining the speed of METIS and gradual refinement ability of BLP. Further, we plan to implement the other two algorithms and verify their predicted partition quality and running time.

Works Cited

[1]Johan Ugander, Lars Backstrom, Balanced Label Propagation for Partitioning Massive Graphs
[2]Jiewen Huang, Daniel J. Abadi, LEOPARD: Lightweight Edge-Oriented Partitioning and Replication for Dynamic Graphs
[3]Quang Duong, Sharad Goel, Sharding Social Networks
[4]George Karpis Vipin Kumar, Unstructured Graph Partitioning and Sparse Matrix Ordering System

Acknowledgement

The authors thank NYU Tandon School of Engineering's Office of Undergraduate Academics for generous funding of the project, and Stanford SNAP for its free dataset provided.