# Clustering Algorithms for Large-Scale Graphs Using MapReduce

Chris Joseph[1] (undergraduate) • Stephen Krucelyak[2] (undergraduate) • Enyue Lu[2] (faculty mentor) • Jack Slettebak[3] • Matthias K. Gobbert[3]
[1]Department of Computer Science • Stony Brook University • Stony Brook, USA
[2]Department of Math and Computer Science • Salisbury University • Salisbury, USA
[3]Department of Mathematics and Statistics • University of Maryland, Baltimore County • Baltimore, USA

## Abstract

**Large-scale graphs representing data from real-world networks are usually non-uniform and contain underlying structures. Graph clustering, a process of identifying these structures, has many applications. Given the massive sizes of modern graph data sets, which consist of millions or billions of vertices and edges, it is difficult or even impossible to process them on a single computer. In this project we take advantage of MapReduce, a programming model suitable for processing large data sets, for graph clustering. The aim of our project is to demonstrate efficient and scalable implementations of clustering algorithms using MapReduce and analyze their performance on the distributed cloud computing platform Amazon EC2. Three different types of graph clustering algorithms are considered: graph theory based k-Trusses clustering, physics model based Barycentric clustering, and probability based Markov clustering. We also compare the scalability, effectiveness, and accuracy of these algorithms for identifying clusters in various data sets acquired from Stanford SNAP.**

## Algorithm Implementation Platform

- **Clustering algorithms were implemented on Amazon Elastic Cloud Compute (EC2) service.**
- **Each virtual machine on EC2 was fitted with 8 64-bit virtual CPU cores, 7GB of memory and a minimum of 100Mbps bandwidth.**
- **Experimental results were obtained by running algorithms on 1, 2, 4, 8 and 16 machines.**
- **Figure 1 shows the scalability of Barycentric clustering algorithm on Google+ dataset (13,673,453 edges and 107,614 vertices).**
- **Figure 2 shows the scalability of K-Trusses and Barycentric clustering algorithms on various datasets using 16 machines.**

**Figure 1: Scalability on Multiple Machines**

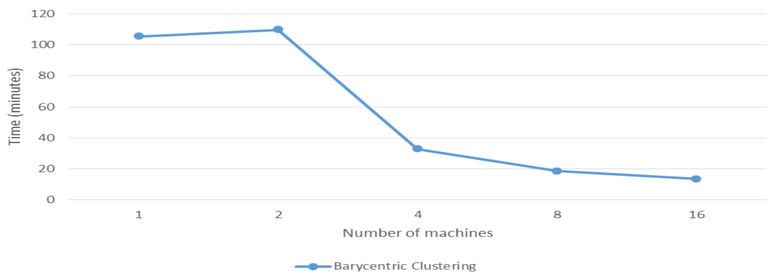
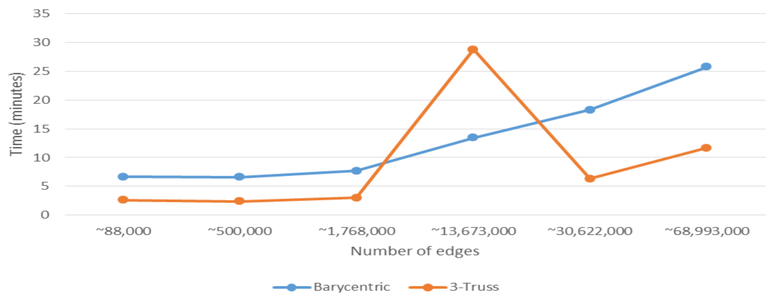



**Figure 2: Scalability on Various Data Sizes**




## Markov MapReduce Algorithm

**Markov clustering is based on simulated stochastic flow in an intuitive notion that if a person were to randomly walk along the edges of the graph, after several such walks, they are more likely to keep walking within their current cluster than they are to leave for a different cluster.**

**The main steps of the Markov clustering MapReduce algorithm:**

1. **Create a matrix that represents the graph**
2. **Normalize each column to represent the probabilities for random walks**
3. **Simulate a random walk by multiplying the matrix**
4. **Normalize the columns again to strengthen intra-cluster edges and weaken inter-cluster edges**
5. **Repeat steps 3-4 until the matrix converges to reach a steady state**

## Barycentric MapReduce Algorithm

**Barycentric clustering is based on the physics ball and spring model, which treats vertices as balls and the edges between them as springs. Intuitively, balls mutually connected by many springs will naturally clump together.**

**The main steps of the Barycentric clustering MapReduce algorithm:**

1. **Generate random physical positions for each vertex**
2. **For each vertex:**
   - **2.1. Calculate the forces from its neighbors**
   - **2.2. Update its position based on the neighbors' forces**
   - **2.3. Calculate the local average length based on the forces**
   - **2.4. Delete edges that exceed the local average length**
3. **Repeat step 2 for 5 iterations**

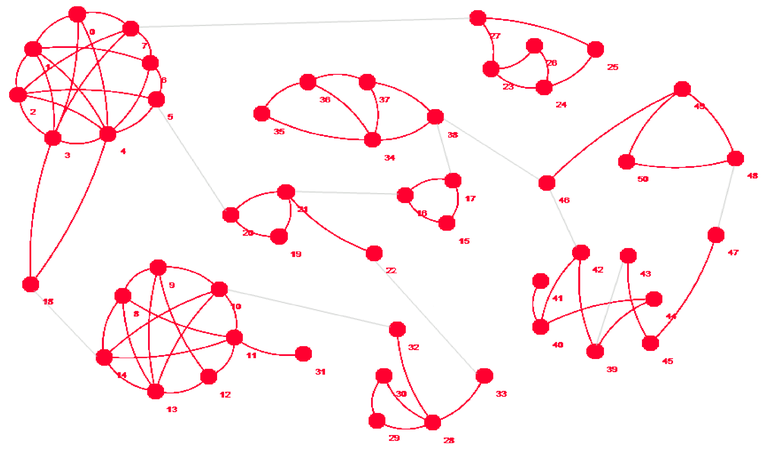
## K-Trusses MapReduce Algorithm

**In a K-Truss, all edges belong to at least K-2 triangles. Trusses are a relaxation of cliques (complete sub-graphs), which are computationally hard to find. Trusses capture most of the characteristics of cliques without being overly restrictive. Larger K values identify more tightly connected clusters.**

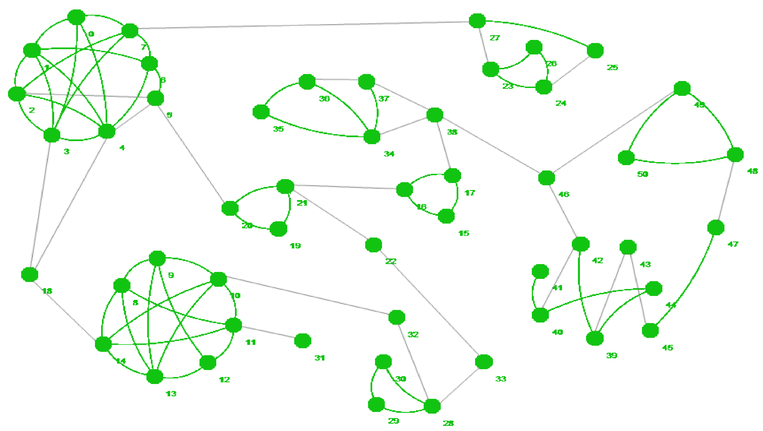
**The main steps of the K-Trusses clustering MapReduce algorithm:**

1. **Augment each edge by computing the degrees of its end vertices**
2. **Find all triangles in the graph using augmented edges**
3. **For each edge, count the number of triangles containing that edge**
4. **If any edge is not part of K-2 triangles, delete it**
   - **4.1. If any edges were deleted, return to step 1**
   - **4.2. If no edge was deleted, the algorithm terminates**

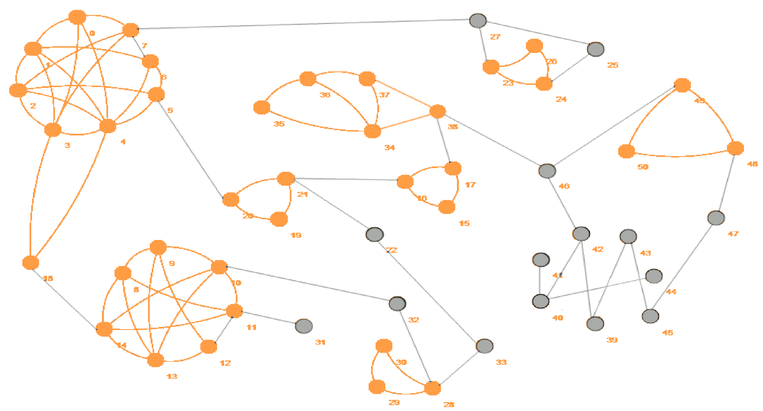
## Markov Clusters



## Barycentric Clusters



## K-Trusses Clusters



**Acknowledgement**: This work is funded by NSF CCF-1460900 under Research Experiences for Undergraduates Program.