CDS DS 210 — Programming Data Science

Name: Emmy Blumenthal | **BU ID:** U87312711
Due Date: Dec 15, 2022 | **Email:** emmyb320@bu.edu

Final Project Report

Implementing Karger's Algorithm on the Facebook Social Circles Graph Dataset

Contents

1 Karger's Algorithm and Minimum Cuts **1**
1.1 Introduction to Karger's Algorithm 1
1.2 Nature of my Implementation of Karger's Algorithm 2

2 Relationship to Original Project Proposal **2**

3 The Facebook Social Circles Graph Dataset **2**

4 Performance and Results **3**

1 Karger's Algorithm and Minimum Cuts

1.1 Introduction to Karger's Algorithm

Karger's algorithm is a well-known randomized algorithm for finding a minimum cut in an undirected graph. In an undirected graph, a minimum cut is a partition of the vertices into two disjoint sets, such that the number of edges crossing the partition is minimized. In other words, a minimum cut is a way of dividing the vertices of the graph into two groups, such that the number of edges connecting the two groups is as small as possible. The algorithm works by repeatedly contracting randomly-selected edges in the graph until only two vertices remain. Karger's algorithm has a running time of $O(n^2m)$, where n is the number of vertices and m is the number of edges in the graph. Although the algorithm is simple and efficient, it is not guaranteed to find the minimum cut with high probability. However, the probability of finding the minimum cut can be increased by running the algorithm multiple times and taking the minimum cut found across all runs. Karger's algorithm has numerous applications in various fields, including network design, image segmentation, and data mining. It is an important tool in the field of graph theory and has been extensively studied and utilized in practice.

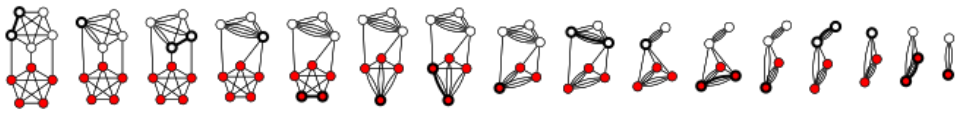

Figure 1: Karger's algorithm successfully identifying the minimum cut between sparsely connected complete graphs with five vertices. Image source: *Wikipedia* (`https://commons.wikimedia.org/wiki/File:Single_run_of_Karger%E2%80%99s_Mincut_algorithm.svg#/media/File:Single_run_of_Karger\OT1\textquoterights_Mincut_algorithm.svg`)

1.2 Nature of my Implementation of Karger's Algorithm

In my implementation of Karger's algorithm, although the input of the function is an adjacency list, I represent the graph using a nested hashmap. Specifically, each vertex is taken as a key in a hashmap whose associated value is another hashmap whose keys represents vertices to which it is connected and whose values are the multiplicity of connections between the vertices. It is necessary to record the multiplicity of connections between the vertices so that I am able to sample uniformly over the set of contracted edges instead of sampling over the set of contracted vertices; this sampling is an essential part of the statistical properties of Karger's algorithm. In order to represent contraction of edges, I also have a vector of integers which represent nodes that have been contracted into; this helps us specify which vertices have been deleted as part of edge contraction without having to iterate through a hashmap. At each iteration, edge contraction is performed by randomly sampling a first node weighted by how many connections it has (including multiplicity), random sampling a second, connected node also weighted by how many connections it has to the already-sampled node (including multiplicity), dropping the first node from the list of un-contracted vertices, replacing all references in other nodes to the first node with a reference to the second node (including multiplicity), and moving all connections from the first node to connections from the second node (including multiplicity). The final result of the algorithm is a vector of length two whose elements are vectors which contain integers representing the vertices on either side of the minimum cut.

2 Relationship to Original Project Proposal

In the originally-submitted project proposal, I proposed using the HCS (Highly Connected Subgraphs) clustering algorithm on a dataset of gene expressions in embryonic mouse brain cells; however, this project was computationally and temporally infeasible for two reasons: dependencies of the HCS algorithm and the size of the embryonic mouse brain cells dataset.

The HCS algorithm requires two substantial components whose implementations are non-trivial: finding minimum cuts and testing if an undirected subgraph is highly connected. When I was setting out to implement the HCS algorithm, I began exploring algorithms for finding minimum cuts and quickly learned that exact algorithms are computationally expensive, so I decided to implement Karger's algorithm. After attempting implementation, I repeatedly ran into errors and statistical subtleties (like dealing with multiplicity of connections). As I neared successfully implementing Karger's algorithm, I realized that the dataset I was interested in had just over 1,000,000 vertices and dense connections, and preliminary tests led to estimates that computations of the algorithm would take at least a day to complete on my laptop. I additionally regularly encountered integer overflow when recording the multiplicity of connections as the number of connections grows exponentially as edges become repeatedly contracted; in the current implementation, I use integers of type u128 to account for such larger numbers. These issues caused me to search for a smaller dataset on the Stanford Network Analysis Project website on which I successfully implemented Karger's algorithm. At the time I had implemented Karger's algorithm on the new dataset, I concluded that my implementation would satisfy the appropriate bounds of the project and that, given my other responsibilities, it would be temporally infeasible to proceed with implementing the entire HCS algorithm. Therefore, I am turning in my current implementation of Karger's algorithm as my project. In further work or as a personal project, I could use what I have implemented here as an essential component of the HCS algorithm.

3 The Facebook Social Circles Graph Dataset

The Stanford Network Analysis Project describes the Facebook Social Circles dataset (`https://snap.stanford.edu/data/ego-Facebook.html`):

> *This dataset consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using this Facebook app. The dataset includes node features (profiles),*

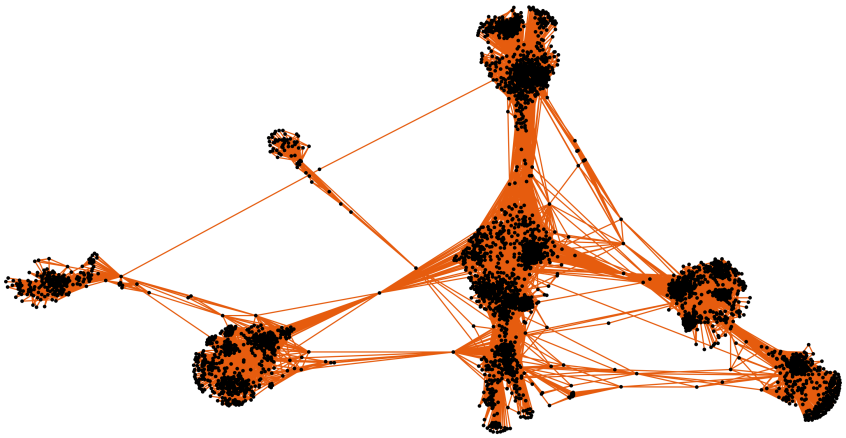

Figure 2: Visualization of Facebook Social Circles graph dataset generated in *Mathematica* using the GraphPlot command.

> *circles, and ego networks. Facebook data has been anonymized by replacing the Facebook-internal ids for each user with a new value.*

The dataset has 4039 nodes and 88234 edges, making it a reasonable size for my implementation and significantly smaller than the embryonic mouse brain cell dataset.

4 Performance and Results

When running in release mode, the algorithm took approximately 18 seconds to run on the Facebook Social Circles dataset and returned data in the proper format: a vector of length two whose elements are vectors of integers.

A comment on tests

As Karger's algorithm is randomized, it is not possible to get consistent results from a unit test for a non-trivial graph. Therefore, I have only implemented one test, 'cut_a_simple_graph,' which checks that a contraction does indeed occur on a graph with two vertices and one edge.