1 Background and Motivation

Let $G = (V, E, w)$ be a weighted connected graph where $w$ is a function from $E$ to positive real numbers. Given a spanning tree $T$ of $G$ we define the distance in $T$ between $u, v \in V \text{dist}_T(u, v)$ to be the length of the unique path in $T$ from $u$ to $v$.

We then define the stretch of an edge $(u, v) \in E$ as

$$\text{stretch}_T(u, v) = \frac{\text{dist}_T(u, v)}{w(u, v)}$$

When studying the stretch of spanning trees, the important value to look at is the average stretch of the spanning tree, defined as

$$\text{ave-stretch}_T(G) = \frac{1}{|E|} \sum_{(u, v) \in E} \text{stretch}_T(u, v)$$

There are several slight variations on the definition of stretch that are used in the literature, however transformations on the weight function accounts for most, if not all, of these variations (e.g. $w \rightarrow 1/w$).

There is considerable interest in generating low-stretch spanning trees for graphs. Applications include approximately solving linear systems efficiently, MCT approxi-
information, and message passing models [EST04].

2 Project Description

The project aims to improve upon existing algorithms for low-stretch spanning trees. Specifically, implementations of these algorithms, while have good theoretical bounds on the stretch, can ‘fail’ on certain graphs and produce relatively high-stretch (although still asymptotically good). Taking motivation from [FRT03], [BGS16], and [MPX13], the current plan is to implement a randomized algorithm loosely based off of Prim’s algorithm to tackle the problem. After this, the algorithm will be thoroughly tested for performance (both in speed and stretch). If the algorithm seems to successfully produce low-stretch trees, an optimistic goal would be a proof as to its correctness. If the algorithm does not seem to work, then an analysis of the graphs on which it fails might lend insight into how it might be modified and improved upon.

3 Approach

The current plan is to implement a randomized algorithm based upon Prim’s Algorithm. Prim’s algorithm is a greedy-algorithm that, starting at some node, adds edges to the spanning tree greedily, namely the smallest edges adjacent to the current connected component. A randomized form of Prim’s is implemented in Laplacians currently, where edges are added with probability inversely proportional to their weight. Unfortunately, it does not perform as well as desired.

The approach plans to modify the algorithm by using multiple starting points. Currently the number and location of these starting points is not fully decided upon. The plan is to use exponential random variables as a means of defining an arrival process, which will determine the order in which each cluster will take a step and add an edge.
4 Deliverables

The deliverables will include the code (in Julia) added to Professor Spielman’s Laplacians library that implements some form of the algorithm for low-stretch spanning tree. Other deliverables would be numerical tests and analysis of the algorithm as well as other algorithms to see if and where they produce high-stretch trees, in hopes of characterizing these graphs.

Optimistically, if the algorithm works, a long-shot deliverable would be a proof of the correctness of said algorithm.

References


