Project Report

Introduction to Algorithm Engineering

Bharat Sahlot — Kyrylo Shyvam Kumar

1 Introduction

The paper[1] presents a data structure and relevant algorithms for fast t-distance queries. We implemented the algorithms for connected undirected weighted graphs represented using Adjacency List.

2 Data Structure

The data structure presented is called *Oracle*. It requires a pre-process step. The pre-process step does most of the work, therefore queries are really fast.

The data structures takes as input a connected undirected weighted graph G and an integer K.

2.1 Pre process Algorithm

According to the paper this step takes time $O(kmn^{\frac{1}{k}})$. The actual time complexity depends on the shortest path algorithm used. In our case we went with djikstra, therefore we have a slightly worse time complexity.

The algorithm selects vertices randomly with a probability of $\frac{1}{k}$. We used `std::bernoulli_distribution` for this. We used STL data structures like `std::vector`, `std::priority_queue` and `std::unordered_map` heavily. We also use `boost::heap::fibonacci_heap` for implementation using Fibonacci Heap.

2.2 Query Algorithm

This is a simple algorithm. It takes two vertices as input and returns the k-spanner distance between them.

3 Performance

We have two implementations of the algorithm (one using `std::priority_queue` and other using `boost::heap::fibonacci_heap`), we compare them against Floyd-Warshall and Djikstras algorithm.

3.1 Random Graphs

We generate random connected graphs of different vertices and edges, and we obtained the following performance graph:

3.2 Real World Graphs

We used real world graphs from the **Graph** directory on the server, and we obtained the following performance graph:

References

[1] Mikkel Thorup and Uri Zwick. “Approximate Distance Oracles”. In: *J. ACM* 52.1 (Jan. 2005), pp. 1–24. ISSN: 0004-5411. DOI: 10.1145/1044731.1044732. URL: https://doi.org/10.1145/1044731.1044732.