Algorithms Analysis and Design

Instructor: Hammad Khan

Team Members: Ali Raza, Taha Hunaid

Paper Selection Proposal

Checkpoint 1

March 30, 2025

Paper Selection Proposal

Paper Details

- **Title:** On Dynamic Shortest Paths Problems
- **Authors:** Liam Roditty, Uri Zwick
- **Conference:** ESA 2004 (European Symposium on Algorithms)
- **Year:** 2004
- **DOI:** 10.1007/978-3-540-30140-0_52

Summary

This paper explores how to efficiently maintain shortest paths in dynamic graphs, where edges can be inserted or removed over time. Instead of recomputing shortest paths from scratch after every change, the authors present new techniques to update paths incrementally, leading to faster algorithms for real-world applications like network routing and graph-based search.

The paper makes three key contributions:

- **Hardness Results:** It establishes that solving incremental and decremental *Single-Source Shortest Paths* (SSSP) problems is just as difficult as solving *All-Pairs Shortest Paths* (APSP) in static graphs. This suggests that certain improvements in dynamic graph algorithms might be fundamentally limited.
- **A Randomized Fully Dynamic APSP Algorithm:** The paper introduces an algorithm for unweighted directed graphs that maintains shortest paths efficiently with an amortized update time of approximately $O(m\sqrt{n})$ and query time $O(n^{3/4})$. This makes it significantly faster than recomputing everything from scratch.
- **Deterministic Spanner Construction:** The authors propose a way to construct an $O(\log n)$-spanner with only $O(n)$ edges in weighted undirected graphs, using incremental SSSP. This helps reduce storage and computation while preserving path accuracy.

Justification

This paper is ideal for our algorithms project due to its blend of theoretical hardness and practical solutions, relevant to network analysis (e.g., routing, social graphs). Exploring reductions, randomization, and spanners enhances our grasp of graph theory and optimization.

Implementation Feasibility

The paper includes clear pseudocodes (Figures 2, 3, and 4), which provides a structure for implementing its algorithms. We plan to:

- Implement the APSP algorithm in Python using the NetworkX library. The use of BFS and random sampling makes it easier to work with using standard graph processing tools.
- We may use public datasets from SNAP (Stanford Network Analysis Project) to test and benchmark performance.
- Implement the spanner construction using a greedy approach combined with incremental SSSP, which aligns with existing shortest path methods in Python libraries.

Team Responsibilities

- **Ali Raza:** Read paper, summarize theorems.
- **Taha Hunaid:** Set up GitHub, source datasets, prototype APSP algorithm.
- **Both:** Research and work on the reports.