MA308 Mini Project
Report

Designing a shortest-path algorithm for large-scale graphs

Submitted by

Roll No.	Names of Students
I22MA023	Abhinav Kumar
I22MA038	Raj Kumar
I22MA062	Chandra Pratap

Under the guidance of
Dr. Sushil Kumar

Department of Mathematics
Sardar Vallabhbhai National Institute of Technology, Surat

Even Semester 2024

Acknowledgement

Department of Mathematics

Sardar Vallabhbhai National Institute of Technology, Surat

Certificate

This is to certify that this is a bonafide record of the project presented by the students whose names are given below during the even semester of 2024 in partial fulfilment of the requirements of the degree of Integrated Masters of Science in Mathematics.

Roll No	Names of Students
I22MA023	Abhinav Kumar
I22MA038	Raj Kumar
I22MA062	Chandra Pratap

Dr. Sushil Kumar
(Project Guide)

Dr. Ramakanta Meher
(Course Coordinator)

Date: 20 January, 2025

Abstract

- Brief overview of the project.
- Key motivation, methodology, and outcomes.
- Highlights of the hybrid algorithm approach and its applications.

Contents

List of Figures

Chapter 1

Problem Definition

1.1 Graph Representation

- Define the graph G=(V, E) where V is the set of vertices (nodes) and E is the set of edges.
- Edge weights represent costs (e.g., travel time, distance, tolls).

1.2 Objectives

- Minimize computation time for shortest path queries.
- Incorporate real-time edge weight updates.
- Allow customizable routing based on user-defined cost functions.

Chapter 2

Introduction

2.1 Background

- Importance of shortest path calculations in large-scale graphs.
- Challenges in real-world applications: scale, dynamics, and user preferences.

2.2 Objective

- To develop and analyze a hybrid shortest path algorithm combining preprocessing, efficient querying, and dynamic updates.

2.3 Scope

- Focus on road networks, real-time traffic updates, and customizable routing.

Chapter 3

Literature Review

3.1 Summary of existing shortest path algorithms

- Dijkstra's, Bellman-Ford, A*, and Bidirectional Search.
- Preprocessing techniques: Contraction Hierarchies (CH), ALT.
- Real-time and dynamic algorithms.

Refer figure **??**.

3.2 Gaps in current approaches

- Limited scalability in dynamic graphs.
- Lack of flexibility for user preferences.
- High preprocessing overhead in large graphs.

Chapter 4

Algorithm Design

4.1 Preprocessing Phase

- Explain the choice of preprocessing techniques (e.g., Contraction Hierarchies, ALT).
- Describe how the graph is simplified or augmented with shortcuts/landmarks.
- Discuss trade-offs in preprocessing time and memory usage.

4.2 Query Execution

- Detail the hybrid algorithm combining preprocessing results with A* or bidirectional search.
- Include pseudocode for the query phase.

4.3 Dynamic Updates

- Describe methods for updating edge weights and recalculating shortest paths efficiently.
- Outline incremental update mechanisms.

4.4 Customization

- Define the composite cost function and how it integrates with the algorithm.

Chapter 5

Implementation

5.1 Tools and Technologies

- Programming language and libraries/frameworks used (e.g., Python, NetworkX, CUDA for parallelism).
- Hardware setup for experiments.

5.2 Code Structure

- Modular design: preprocessing, query execution, dynamic updates, and customization.

5.3 Dataset

- Description of datasets used (e.g., OpenStreetMap, SNAP datasets).
- Data preprocessing steps: parsing, cleaning, and formatting.

Chapter 6

Result and Analysis

6.1 Performance Metrics

- Preprocessing time and memory usage.
- Query execution time (average, worst-case).
- Accuracy of paths (for heuristic methods).
- Scalability with graph size and density.

6.2 Comparative Analysis

- Benchmark hybrid algorithm against standalone algorithms (e.g., plain Dijkstra's, A*).
- Use tables and plots to illustrate improvements.

6.3 Sensitivity Analysis

- Impact of graph properties (e.g., number of nodes, edge density).
- Effect of dynamic updates on performance.

Chapter 7

Discussion

7.1 Strengths

- Highlight significant improvements in speed, accuracy, or flexibility.
- Discuss adaptability to various real-world scenarios.

7.2 Limitations

- Discuss preprocessing overhead or constraints in memory usage.
- Identify scenarios where the hybrid approach might underperform.

7.3 Further Improvements

- Enhancing scalability for distributed systems.
- Incorporating machine learning to predict edge weights dynamically.
- Extending to multimodal routing or 3D graphs.

Chapter 8

Conclusion

- Recap the problem, approach, and key findings.
- Reiterate the significance of combining multiple algorithms for shortest path calculations.
- Highlight practical implications and potential impact.

References

[1] Cite all academic papers, libraries, datasets, and tools used. `<urlhere>`

Appendices

.1 Pseudocode

Detailed pseudocode for key components.

.2 Additional Figures

Visualizations of the graph, preprocessing results, and query paths.

.3 Proofs

Proofs for bounds on time and space complexity.

.4 Raw Data

Further Performance metrics and analysis data.