Rutgers School of Engineering

ECE 573 : Project Proposal

Adaptive Incremental Shortest Path Algorithm for Dynamic Network Optimization

Joel Joseph Kinny (jk2112)
Sumedh Marathe (sam792)
Apurv Paliwal (ap2523)

TABLE OF CONTENTS

Revision History

Version	Date	Reason	Sign Off
1.0	3/24/2024	Initial Draft	
1.1	3/31/2024	Changes for Incremental Algo	

Introduction

The goal of this project is to develop an adaptive, incremental algorithm for computing the shortest paths in a graph that efficiently updates the shortest paths when the graph changes in real-time. This approach is crucial for applications where graph data frequently changes, such as in airline or train route planning, where schedules and costs are subject to modification. By selectively choosing the most efficient algorithm based on the density of the graph, and focusing on incremental updates, this project aims to significantly reduce the computational overhead associated with dynamic shortest path calculations.

Background

The Floyd-Warshall algorithm is a well-known method for finding shortest paths between all pairs of vertices in a weighted graph. Despite its effectiveness for dense graphs, its cubic time complexity (O(V^3)) makes it less suitable for graphs that undergo frequent updates. For sparse graphs, algorithms like Johnson's Algorithm, which combines Bellman-Ford and Dijkstra's algorithms, offer more efficient computation by initially reweighting the graph to eliminate negative weights and then applying Dijkstra's algorithm from each vertex.

This project seeks to bridge the gap between static shortest path algorithms and the dynamic nature of real-world graphs. By developing an incremental approach to updating shortest paths in response to graph changes, and by adaptively choosing between Floyd-Warshall and Johnson's Algorithm based on graph density, the project aims to offer a more efficient solution for real-time shortest path computations.

Methodology

- Graph Representation and Initial Computation:
    - Implementation of graphs using adjacency matrix/compressed sparse row
    - Implement both Floyd-Warshall algorithms for base case and full computation.
    - Create a synthetic dataset representing a transportation network (e.g., airfares or train fares) with varying densities to serve as the input graph.
    - Calculate the initial shortest paths using the appropriate algorithm based on the initial density of the graph.
- Incremental Update Mechanism:
    - Implement an incremental update algorithm that efficiently updates shortest paths in response to changes in the graph (e.g., addition, removal, or weight change of edges).
    - The update mechanism should minimize the need to recompute shortest paths from scratch.
    - The Incremental update mechanism should be able to handle incremental updates such as node insertions/deletions and update to edge weights without having to recompute the shortest path distance matrix from scratch.
    - The candidate algorithms that would be explored in this project would be the LPA (Lifelong Planning A*) algorithm and Ramalingam and Reps algorithm (DynamicSWSF-FP).

    - These algorithms would be assessed and compared for the best use for the All Pairs Shortest Path problems
- Adaptive Algorithm Selection:
    - Implement a mechanism to assess the density of the graph dynamically.
    - Based on the density, automatically select between Floyd-Warshall and Johnson's algorithm for both initial computation and subsequent updates.
- Evaluation:
    - Evaluate the performance of the proposed incremental update algorithm in terms of computation time and accuracy, comparing it against the baseline full recomputation with both Floyd-Warshall and Johnson's algorithms.
    - Analyze the efficiency of adaptive algorithm selection based on graph density changes.

Making Algorithms Incremental

Making classical algorithms like the Floyd-Warshall algorithm incremental involves adapting them to efficiently handle updates to the input data, such as adding or removing vertices or edges in a graph. The Floyd-Warshall algorithm itself is a dynamic programming algorithm for finding the shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles). The classical version of the algorithm computes the shortest paths between all pairs of vertices, but it does so in a batch manner and is not designed to be inherently incremental. When handling edge weight updates, addition/removal of edges, or addition/removal of vertices in a graph, one must ensure that shortest paths are updated accordingly. This involves checking for shorter paths due to weight decreases, reevaluating affected paths for weight increases or edge removals, and initializing or updating shortest paths when adding or removing vertices, typically represented using adjacency matrices or adjacency lists.

Implementation Strategy

The key to implementing these updates efficiently lies in maintaining a data structure that allows for quick identification of affected paths and vertices. For example, keeping track of the predecessors for each shortest path can help in quickly identifying which paths are affected by a change.We would be implementing a priority queue data structure to keep track of the affected vertices with the changes.

The exact implementation details can vary based on the specific requirements, such as whether the graph is dense or sparse, and the frequency and type of updates. In some cases, especially for frequent updates or in highly dynamic graphs, other algorithms designed specifically for dynamic graphs might offer better performance than adapting the Floyd-Warshall algorithm. Algorithms like the DynamicSWSF-FP, LPA* D* Lite would be explored further for the correct implementation.

Evaluation Strategy

The evaluation of the algorithms would involve test cases containing directed graphs of varying numbers of nodes, edges and edge weights. Ideally, the naive test case would be to start with graphs of 4 nodes and 4 edges. There would be an increase in number of nodes, number of edges and edge weights for the baseline evaluation. After establishing the baseline full computation, we would perform incremental updates to the already existing shortest path graph by varying number of updates (insertion of nodes/deletion of nodes/changes to edge weights).

Expected Outcomes

- A comprehensive comparison of the efficiency and accuracy of the incremental update algorithm versus traditional full recomputation methods.
- Insights into the impact of graph density on the choice of shortest path algorithm and the overall computational efficiency.
- A scalable solution for real-time shortest path updates in dynamic graphs, applicable to various domains such as transportation routing and network optimization.

Team

- Joel Joseph Kinny (Github - joelkny97)
    - Research and Development of LPA* and RR Algorithms
    - Empirical Analysis of Algorithms
    - Test Case generation
    - Documentation
- Apurv Paliwal (Github - apurvpaliwal)
    - Research and Development of Graph Representation
    - Test Case generation.
    - Integration of Incremental Search
- Sumedh Marathe (Github - sumedh-21)
    - Research and Development of Floyd-Warshall algorithm
    - Test Case generation for Floyd Warshall
    - Dataset generation of Integration Test Case

Project Repository

https://github.com/apurvpaliwal/ECE-573-DSA-finalproject

Milestones

Description	Start Date	End Date	Duration
Project Start	3/25/2024		
Research on Floyd-Warshall Algorithm	3/25/2024	4/1/2024	1 week
Development of Test Case Framework	3/25/2024	4/1/2024	1 week
Research on LPA* and RR Algorithm	3/25/2024	4/1/2024	1 week
Phase 1 Complete	3/25/2024	4/1/2024	1 week
Development of Floyd-Warshall Algorithm	4/2/2024	4/8/2024	1 week
Setup of Graph Representation and Test Case Code	4/2/2024	4/8/2024	1 week
Development of LPA* and RR Algorithm	4/2/2024	4/8/2024	1 week
Test Case generation and Comparison for all Algorithms	4/8/2024	4/11/2024	3 days
Documentation of Report and Research Paper	4/11/2024	4/15/2024	4 days
Phase 2 Complete	4/1/2024	4/12/2024	2 weeks
Development of Adaptive Density Algorithm	4/16/2024	4/22/2024	1 week
Test Case Generation for Integration Tests	4/23/2024	4/26/2024	3 days
Documentation of Report and Research Paper	4/27/2024	4/29/2024	4 days
Project Presentation	4/27/2024	4/29/2024	2 days
Project Delivery		4/29/2024	

References

1. G. Ramalingam and T. Reps, "An Incremental Algorithm for a Generalization of the Shortest-Path Problem," in Proceedings of the 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '94), 1994, pp. 17-29.
2. G. Ausiello, G. F. Italiano, A. Marchetti Spaccamela, and U. Nanni, "Incremental algorithms for minimal length paths," Journal of Algorithms, vol. 12, no. 4, pp. 615-638, 1991.
3. I. H. Toroslu, "Improving The Floyd-Warshall All Pairs Shortest Paths Algorithm," https://arxiv.org/abs/2109.01872
4. S. Sunita and D. Garg, "Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queue," inJournal of King Saud University - Computer and Information Sciences, vol. 13, no. 3, pp 364-373, 2021
5. I. Abraham, S. Chechik, and S. Krinninger, "Fully dynamic all-pairs shortest paths with worst-case update-time revisited," https://arxiv.org/abs/1607.05132
6. Baeldung. "Graph Density." https://www.baeldung.com/cs/graph-density.
7. Greco, S., Molinaro, C., Pulice, C. *et al.* Incremental maintenance of all-pairs shortest paths in relational DBMSs. *Soc. Netw. Anal. Min.* **7**, 36 (2017). https://doi.org/10.1007/s13278-017-0457-y