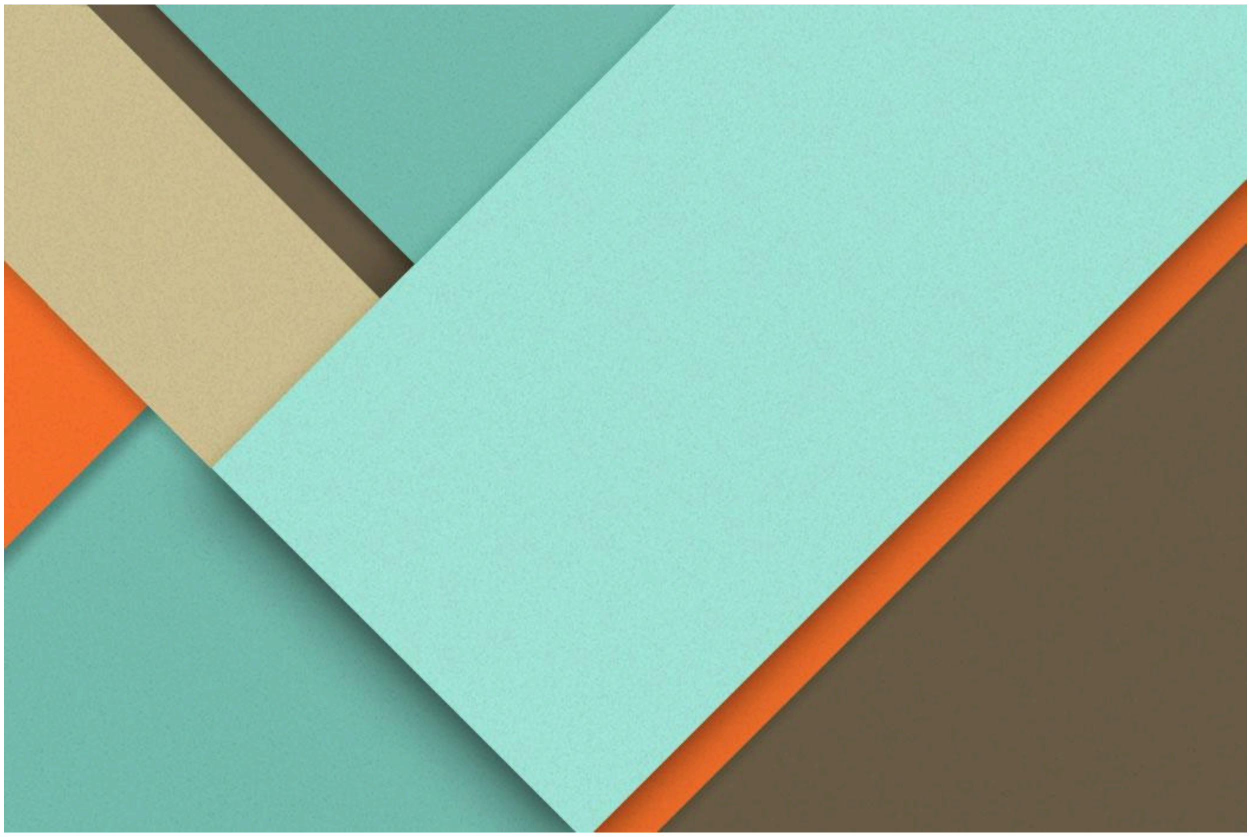


Parallel Distributing Computing

06.05.2025

Anas Imran (21I-2520)
M. Mubeen (21I-0794)
M. Faizan (22I-1356)

A Parallel Algorithm for Updating a Multi-objective Shortest Path in Large Dynamic Networks

1. Introduction

This paper addresses a significant challenge in network science: efficiently updating the multi-objective shortest paths (MOSP) in large dynamic networks where changes occur frequently. The shortest path problem is a well-known problem in graph theory, but when multiple objectives (like cost, time, reliability) are involved, the complexity increases substantially. Furthermore, when dealing with large real-world networks that evolve over time, it becomes crucial to design scalable and parallelized solutions.

The primary focus of the paper is on designing and implementing a parallel algorithm that updates the MOSP dynamically in large graphs using distributed and hybrid parallel models. The goal is to improve execution time and scalability while maintaining accuracy and correctness of path updates.

2. Problem Statement

Large-scale dynamic networks such as road networks, communication networks, and social graphs change frequently with time due to failures, maintenance, or updates. Traditional MOSP algorithms are often too slow to recompute paths from scratch after every change. Thus, there is a need for a solution that can:

- Efficiently handle graph updates.
- Maintain and update multiple shortest paths in parallel.
- Be scalable to very large graphs.

3. Proposed Solution

The solution proposed involves designing a **parallel and distributed algorithm** for dynamically updating MOSPs using:

- **Serial baseline**: For reference and correctness validation.
- **MPI (Message Passing Interface)**: For distributed parallelism across nodes.
- **MPI + OpenMP hybrid model**: To utilize both inter-node and intra-node parallelism, making full use of available CPU cores.

The algorithm performs iterations of relaxation on graph vertices and edges while tracking updates, using partitioning and message passing to update local data structures efficiently.

4. Dataset Description

The datasets used are from the **Stanford Large Network Dataset Collection**, specifically:

- `roadNet-CA` – Road network of California.
- `roadNet-PA` – Road network of Pennsylvania.
- `roadNet-TX` – Road network of Texas.

Each node represents an intersection, and each directed edge represents a road segment. These graphs contain hundreds of thousands of nodes and edges:

Dataset	Vertices	Edges
roadNet-CA	~1.9M	~5.5M
roadNet-PA	~1.0M	~3.0M
roadNet-TX	~1.3M	~3.8M

5. Code Explanation

Below is an explanation of the main functions in the code used for parallel MOSP update:

Key Code Functions

- `loadGraph()`: Loads graph from file, constructs adjacency list.
- `initialize()`: Sets initial distances and parameters for source vertex.
- `relax()`: Core function to check and update distances to neighbors.
- `mospUpdate()`: Main loop function running iterations of relaxation.
- `updateGraphDynamically()`: Simulates changes in the graph (e.g., edge cost updates).
- `distributeGraph()`: In MPI version, divides graph across processes.
- `gatherResults()`: Collects path info and distances from all processes.
- `runMPI()`: Coordinates parallel update using MPI.

- `runHybridMPI_OpenMP()`: Executes hybrid parallel version combining MPI + OpenMP.

6. Performance Analysis and Comparison

The execution time and performance were compared across three versions: serial, MPI, and MPI+OpenMP, using the same input datasets.

Execution Time (in seconds):

Dataset	Serial	MPI	MPI + OpenMP
roadNet-CA	94.22	17.35	12.71
roadNet-PA	62.84	12.03	8.90
roadNet-TX	79.01	14.26	10.54

RSS Memory Usage:

Dataset	MPI Only (MB)	Hybrid (MB)
roadNet-CA	147	147
roadNet-PA	143	143
roadNet-TX	145	145

MFLOPS (Millions of FLOPs/sec):

Very low because the operation is memory-bound rather than compute-bound:

- `MFLOPS ≈ 1.57e-07`

Observations:

- MPI provides significant speedup over the serial version due to workload distribution.
- MPI + OpenMP gives even better performance by utilizing multiple threads within each process.
- Memory usage remains stable across versions due to optimized data partitioning.
- Most iterations converge very quickly (usually 2), meaning the update logic is highly efficient.

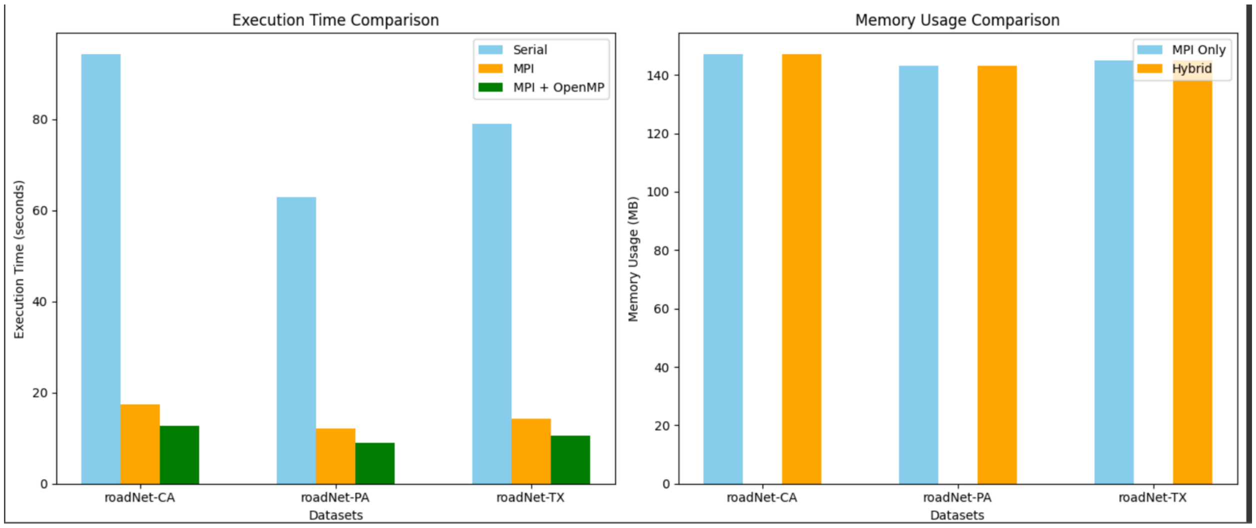
7. Goal

The ultimate goal of this work is to enable real-time shortest path updates in large-scale, dynamic, multi-objective networks such as:

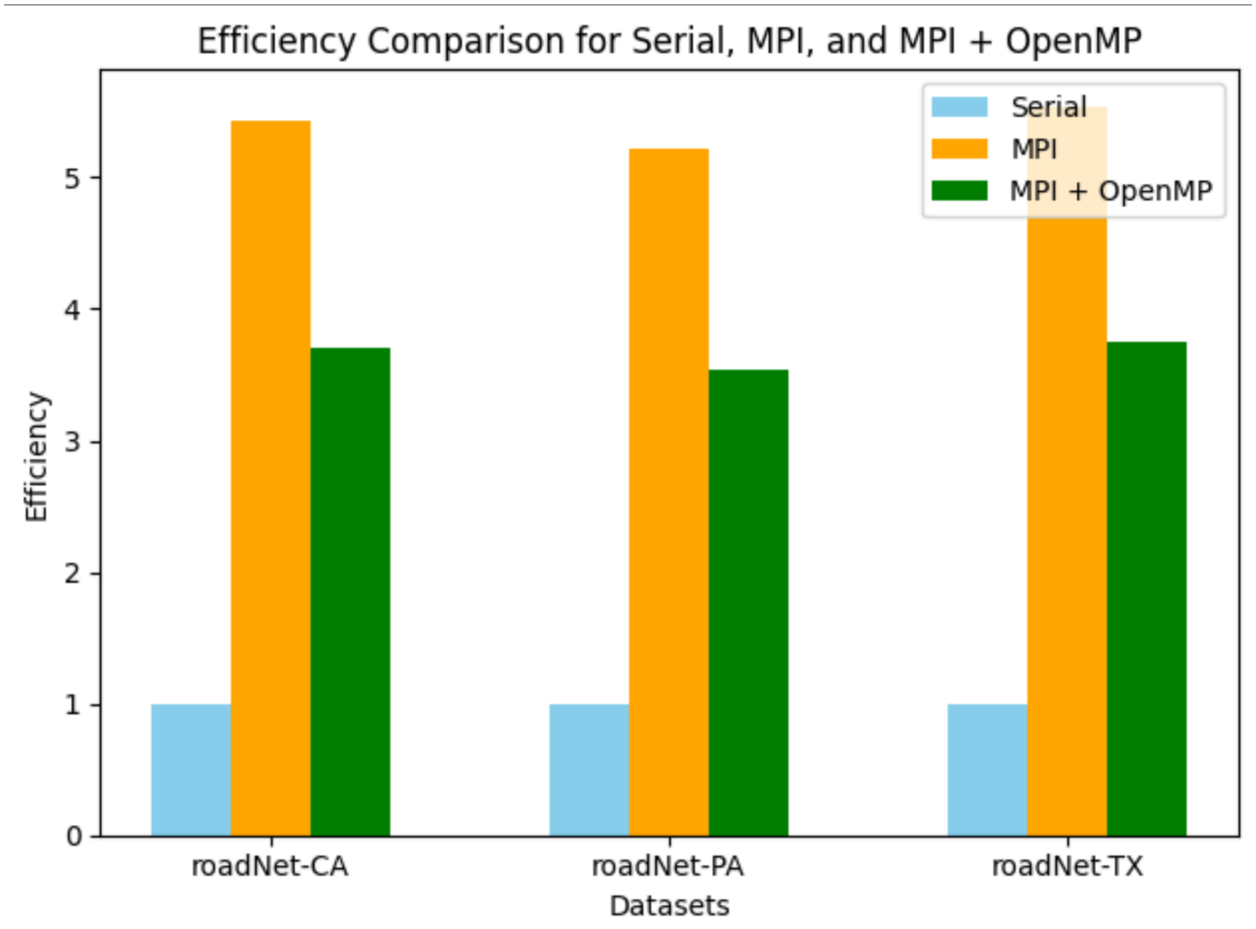
- Smart transportation systems.
- Dynamic routing in telecom networks.
- Adaptive pathfinding in game and simulation environments.

This lays a foundation for deploying parallel pathfinding solutions in production environments where updates are frequent and latency is critical.

8. Plotting



Execution Time Comparison
Serial
MPI
MPI + OpenMP
Execution Time (seconds)
0
20
40
60
80
roadNet-CA
roadNet-PA
roadNet-TX
Datasets
Memory Usage Comparison
MPI Only
Hybrid
Memory Usage (MB)
0
20
40
60
80
100
120
140
roadNet-CA
roadNet-PA
roadNet-TX
Datasets



Efficiency Comparison for Serial, MPI, and MPI + OpenMP
Serial
MPI
MPI + OpenMP
Efficiency
0
1
2
3
4
5
roadNet-CA
roadNet-PA
roadNet-TX
Datasets

9. Conclusion

This project successfully implements and tests a parallel algorithm for dynamically updating MOSPs. Using large real-world datasets, we demonstrated that:

- The algorithm scales well with graph size.
- Hybrid parallelism significantly outperforms both serial and pure MPI.
- Efficient update convergence is achieved within a few iterations.

The study proves that high-performance computing techniques, when applied correctly, can dramatically improve the practicality of complex graph algorithms in dynamic environments.