Parallel Distributed Computing

Implementation and Analysis of Parallel Graph Algorithms

Research: A Parallel Algorithm Template for Updating Single-Source Shortest Paths in Large-Scale Dynamic Networks

Muaz Ahmed **Ayan Asif**

Muhammad Abbas

Motivation & Problem Overview

- SSSP in Dynamic Networks
  - Traditional algorithms assume static graphs
  - Real-world networks change over time (edge insertions/deletions)
- Challenges
  - Efficiently updating SSSP without full recomputation
  - Balancing load and avoiding synchronization overhead
- Need for Parallelism
  - Exploit multiple cores and GPUs for faster processing

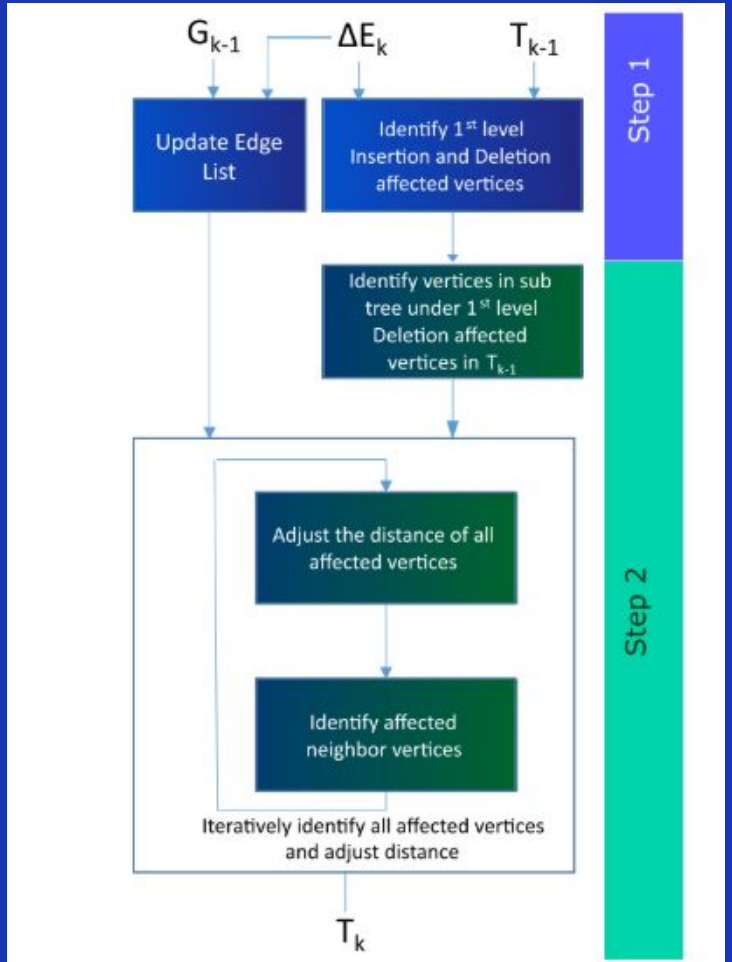


Fig. 1. A Parallel framework to update SSSP in dynamic networks.

Parallel Algorithm Design & Key Contributions

- Two-Step Framework:
  - Identify the affected vertices
  - Iteratively update the SSSP tree
- Key Innovations:
  - Avoiding heavy locking by iterative updates
  - Using a rooted-tree data structure to maintain SSSP
  - Preventing cycle formations during concurrent updates

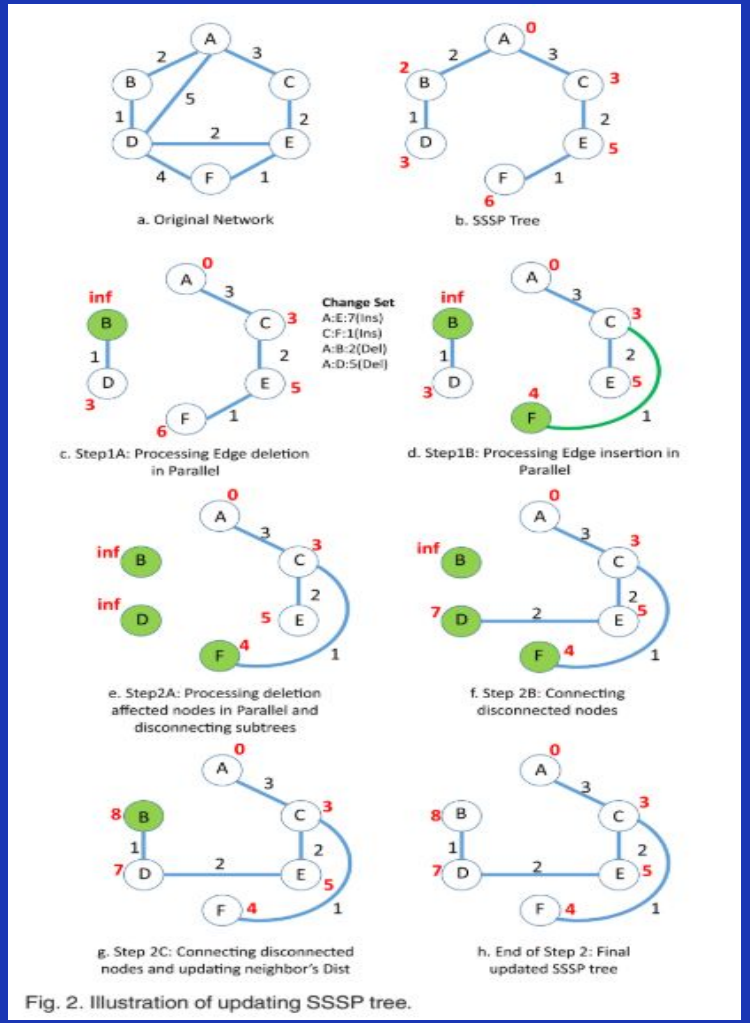
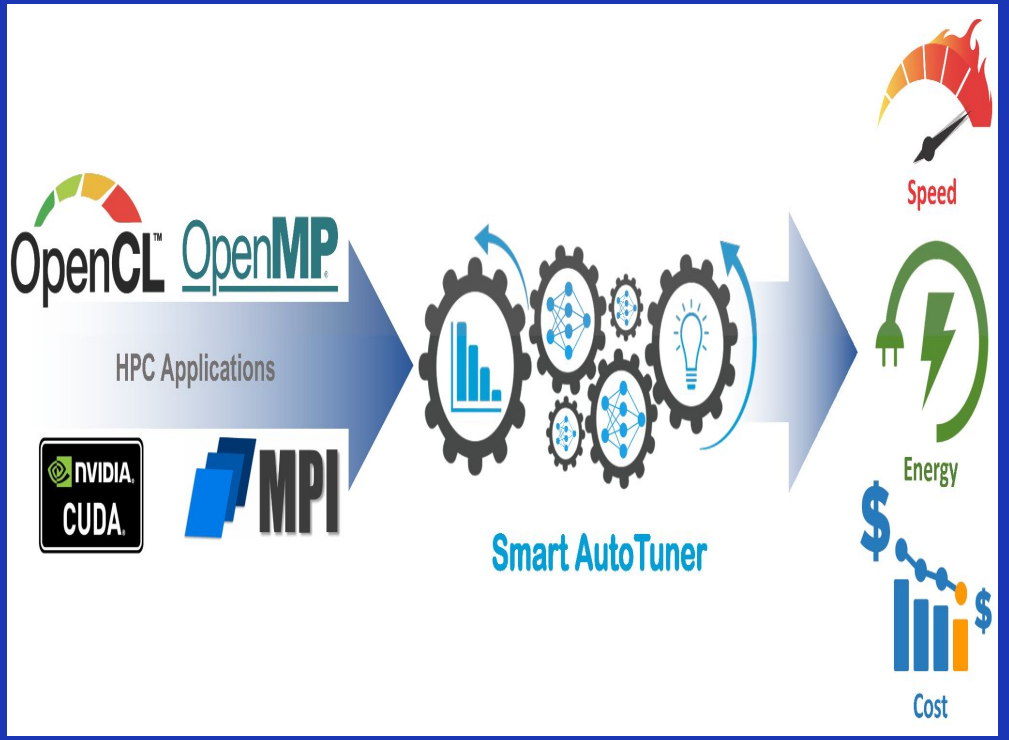


Fig. 2. Illustration of updating SSSP tree.

Implementation Strategy & Tools

- Programming Models
  - MPI
  - OpenCL
  - OpenMP
- Graph Partitioning
  - METIS
- Version Control
  - GitHub



Scalability Analysis & Performance Evaluation

- **Algorithm Complexity:**
  - Expect O(m/p) time complexity
  - Expect O(D·(x·d)/p)

- **Evaluation Plans:**
  - Targeting **significant speedup**, particularly on GPU architectures
  - Will explore the performance differential in **edge insertion-heavy** vs **deletion-heavy** scenarios

- **Scalability Focus Areas:**
  - Investigate how **edge change ratios** affect performance
  - Integrate **load balancing and dynamic scheduling** strategies

Conclusion: Plans for Execution

Phase 1

Planning

- Define problem space
- Identify use cases
- Select benchmark datasets
- Set evaluation goals

Phase 2

Framework Architecture

- Partition graph (METIS)
- Design parallel model
- Combine MPI + OpenMP/OpenCL
- Enable lock-free updates

Phase 3

Implementation & Testing

- Develop core algorithm
- Run scalability tests
- Vary update conditions
- Benchmark against baselines

Thank You

Questions?