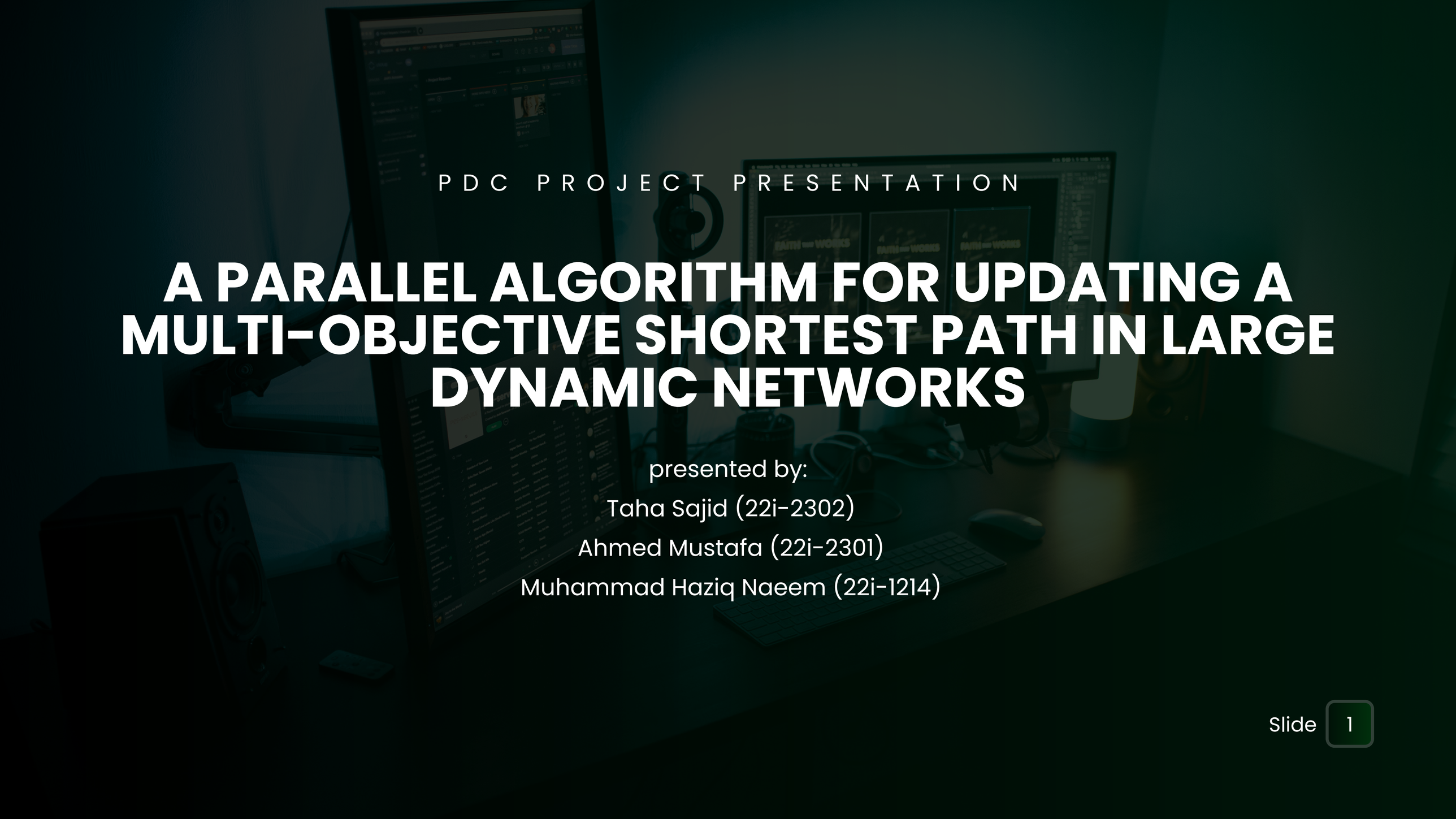


PDC PROJECT PRESENTATION

A PARALLEL ALGORITHM FOR UPDATING A MULTI-OBJECTIVE SHORTEST PATH IN LARGE DYNAMIC NETWORKS

presented by:

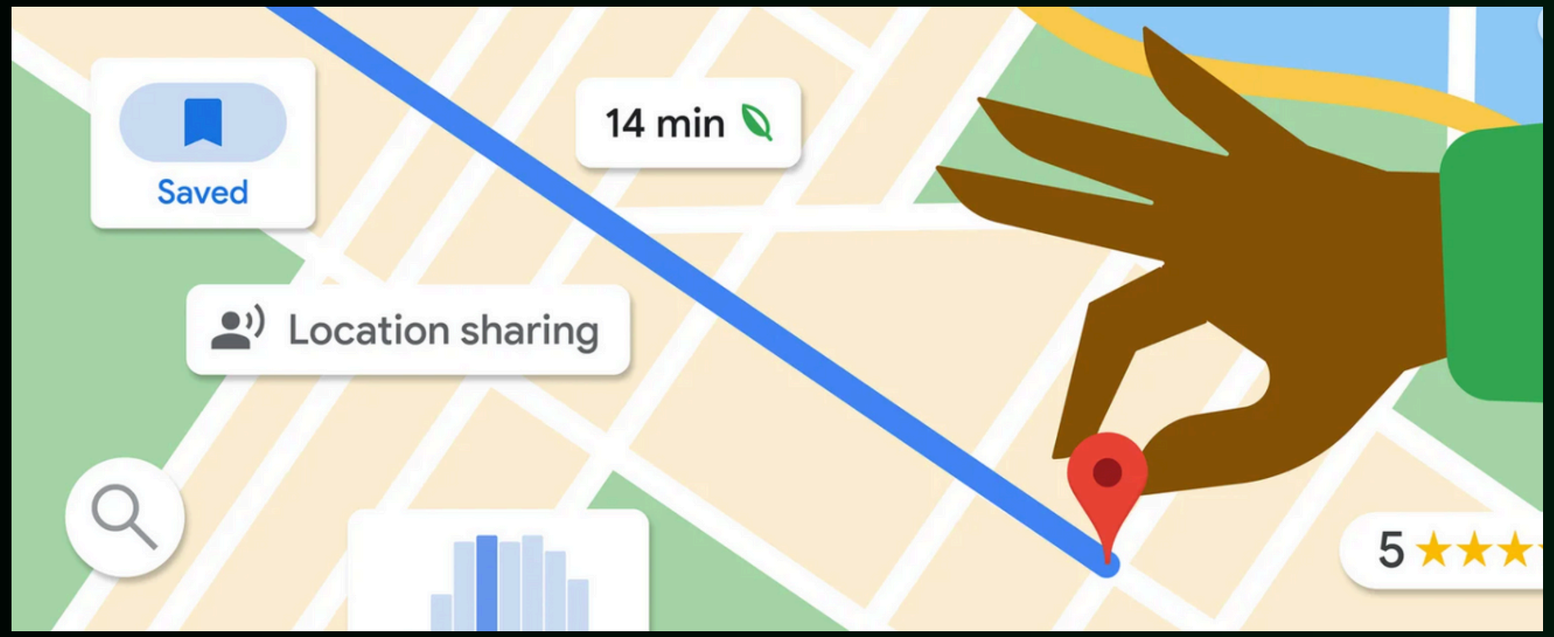
Taha Sajid (22i-2302)

Ahmed Mustafa (22i-2301)

Muhammad Haziq Naeem (22i-1214)

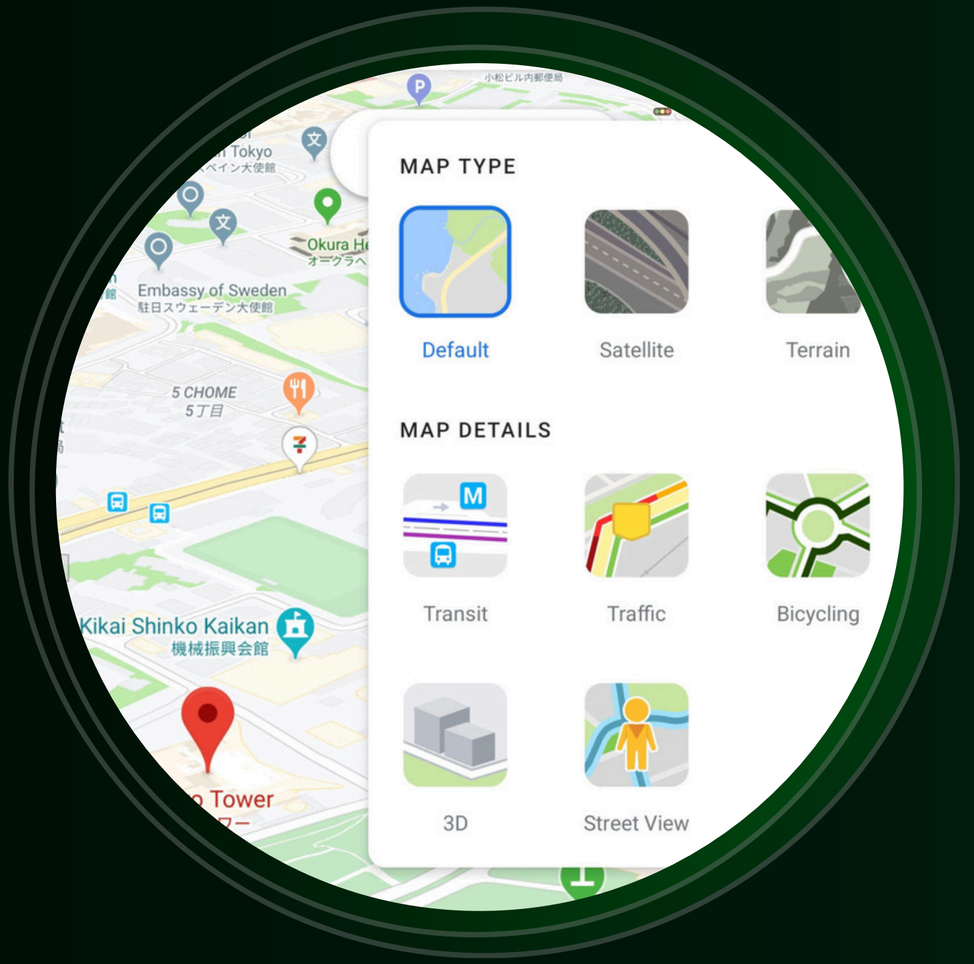
Shortest Path

The quickest, cheapest, and the most efficient route between two points

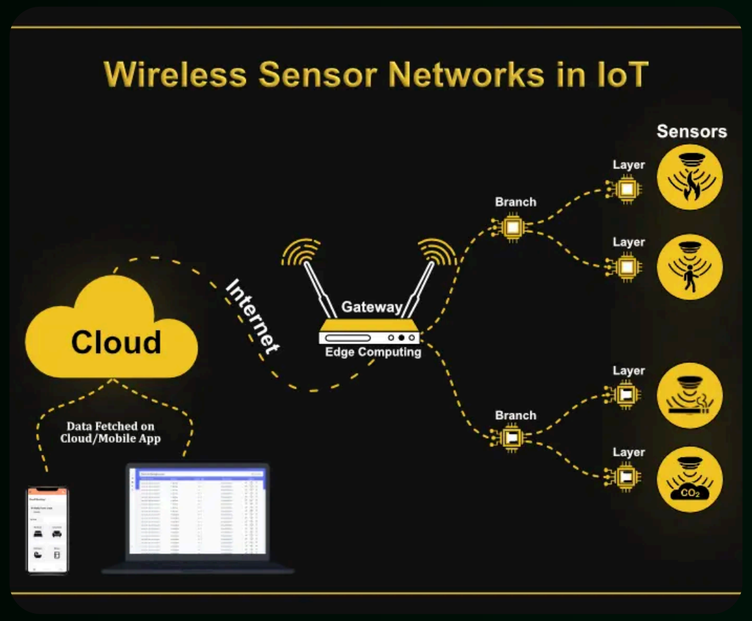


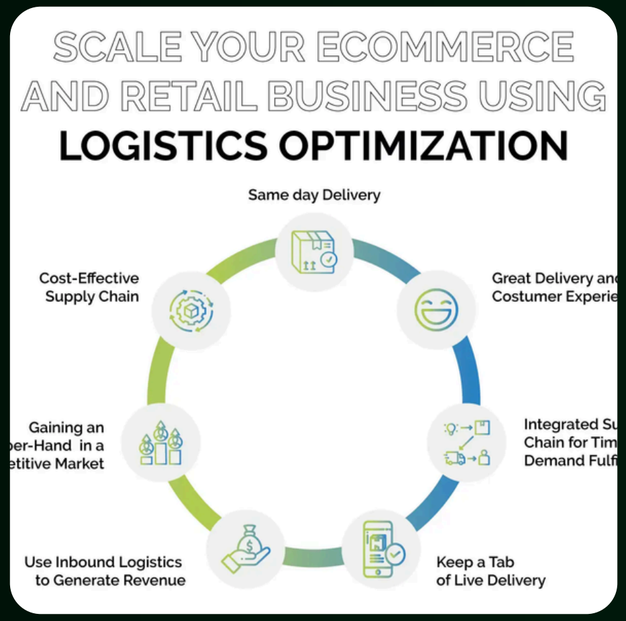
Understanding the Shortest Path Problem (SOSP & MOSP)

- SOSP: Finding the shortest path based on just ONE factor (e.g. time, cost or distance)
- MOSP: Finding best path based on multiple factors (e.g. time + cost + safety).



Where This Applies?

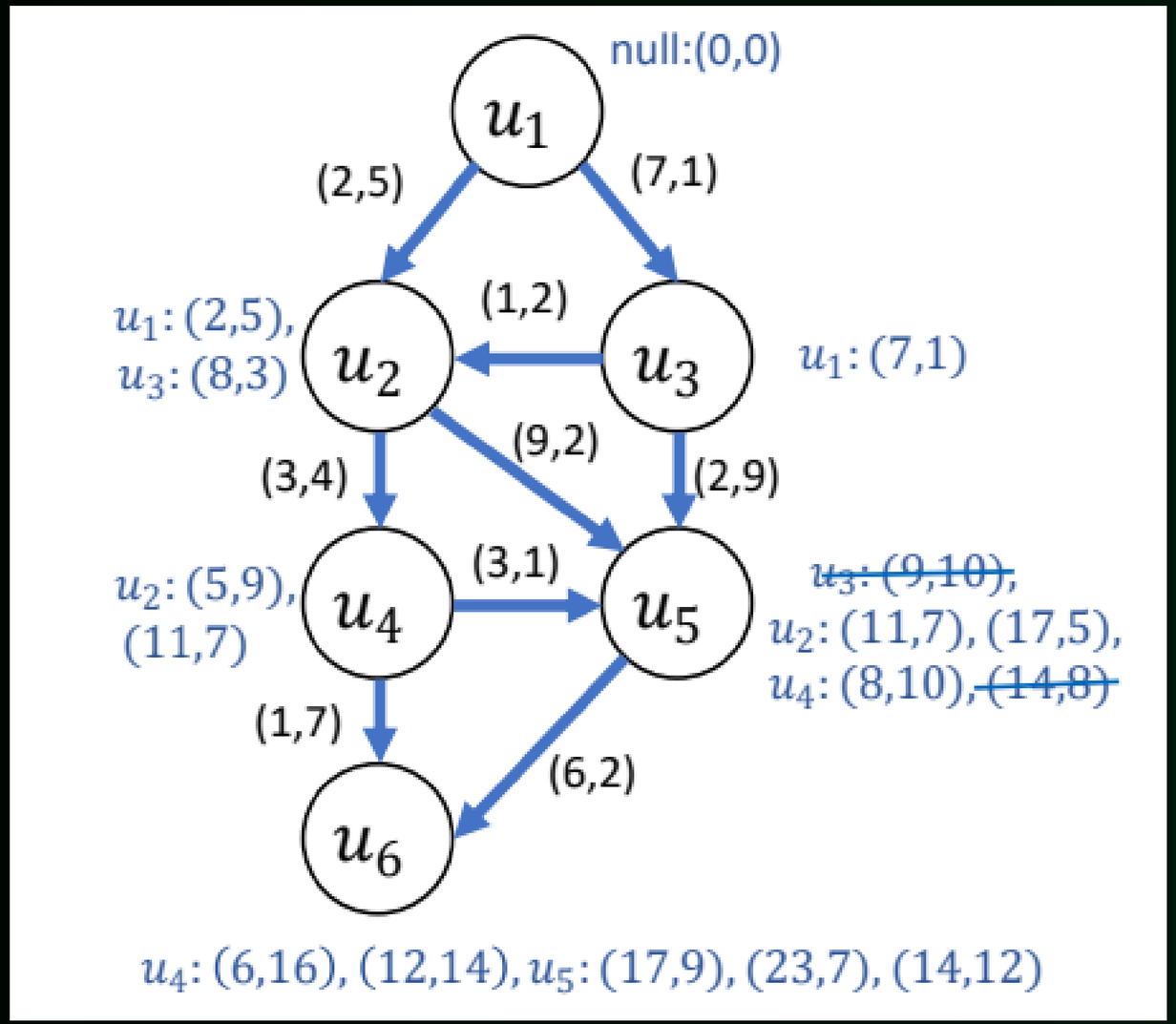




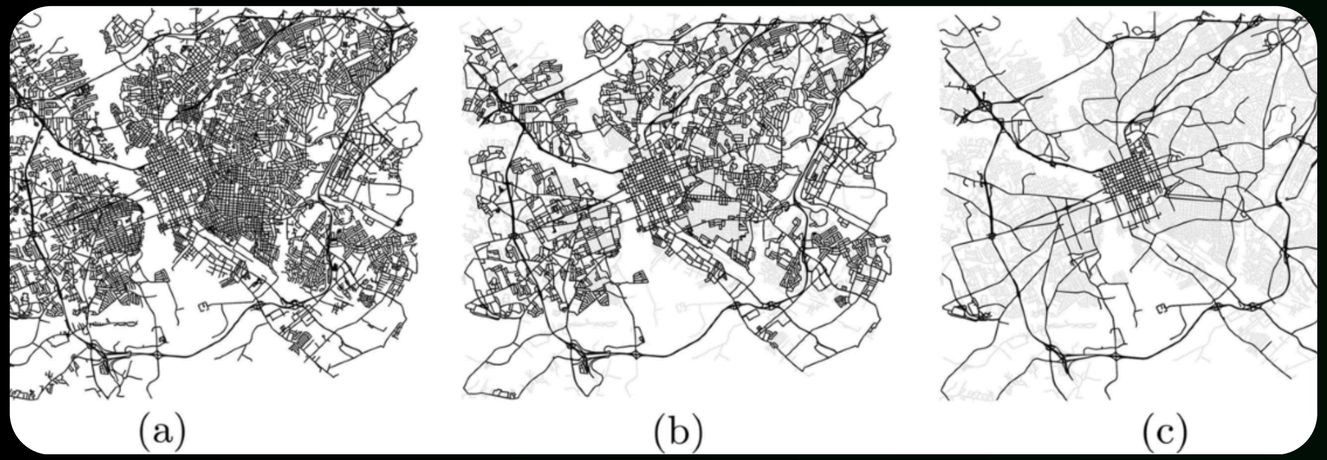
Before Diving In: Know These Terms

- •Pareto Optimality: Best trade-offs — can’t improve one factor without worsening another (e.g. Speed and fuel consumption).
- •Dominated Paths: Eliminated from Pareto Optimality distance set (e.g. (8,10) dominates (9, 10))

MOSP Graph


null:(0,0)
u_1
(2,5)
(7,1)
u_1: (2,5),
u_3: (8,3)
u_2
(1,2)
u_3
u_1: (7,1)
(9,2)
(3,4)
(2,9)
u_2: (5,9),
(11,7)
u_4
(3,1)
u_5
~~u_3: (9,10),~~
u_2: (11,7), (17,5),
u_4: (8,10), ~~(14,8)~~
(1,7)
(6,2)
u_6
u_4: (6,16), (12,14), u_5: (17,9), (23,7), (14,12)

What’s a Dynamic Network?



- Edges/Nodes can be added or removed (like road openings or closures).

What’s the Point of a Dynamic Graph Algorithm?

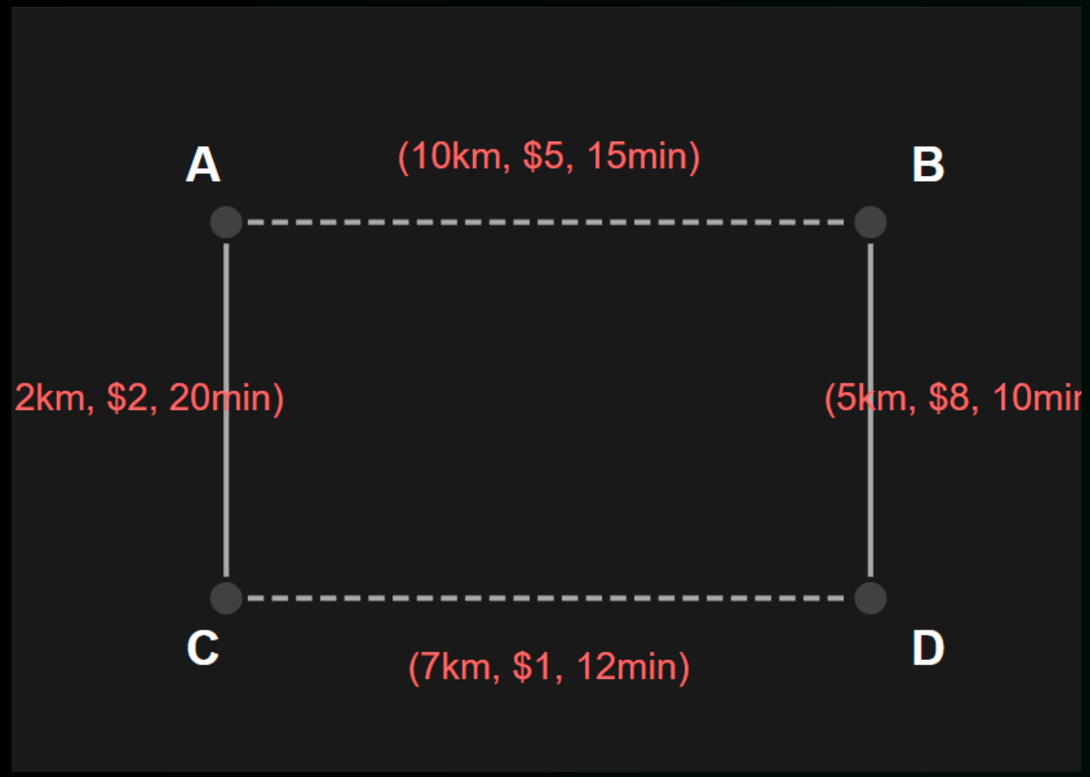
In a static graph, when you want to compute something like the shortest path, you compute it once and you’re done.

But in a dynamic graph, edges may be added or removed constantly. You don’t want to recompute everything from scratch each time.

So What Does the Paper Do?

- Proposes a Parallel Algorithm to update SOSP efficiently.
- Extends it with a Heuristic Approach for MOSP.
- Tested on large real and synthetic networks.

A Quick Example



Path	Distance	Cost	Time
A → B → D	15 km	$13	25 min
A → C → D	19 km	$3	32 min

So Works with 3 Basic Steps

Invalidates closed paths

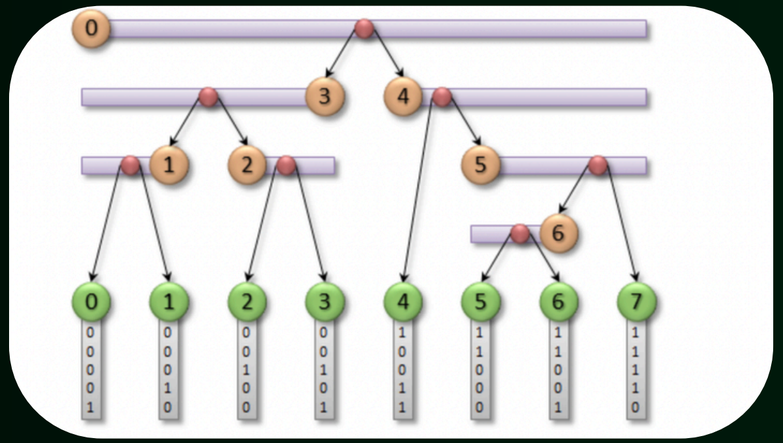
Re-evaluates alternatives

Updates your best path quickly and efficiently

Parallel SOSP Algorithm (Simplified)

Steps:

1. Group Changed Edges
2. Process in Parallel
3. Iteratively Propagate Updates



Output: Updated shortest path tree (fast and efficient)

From SOSP to MOSP

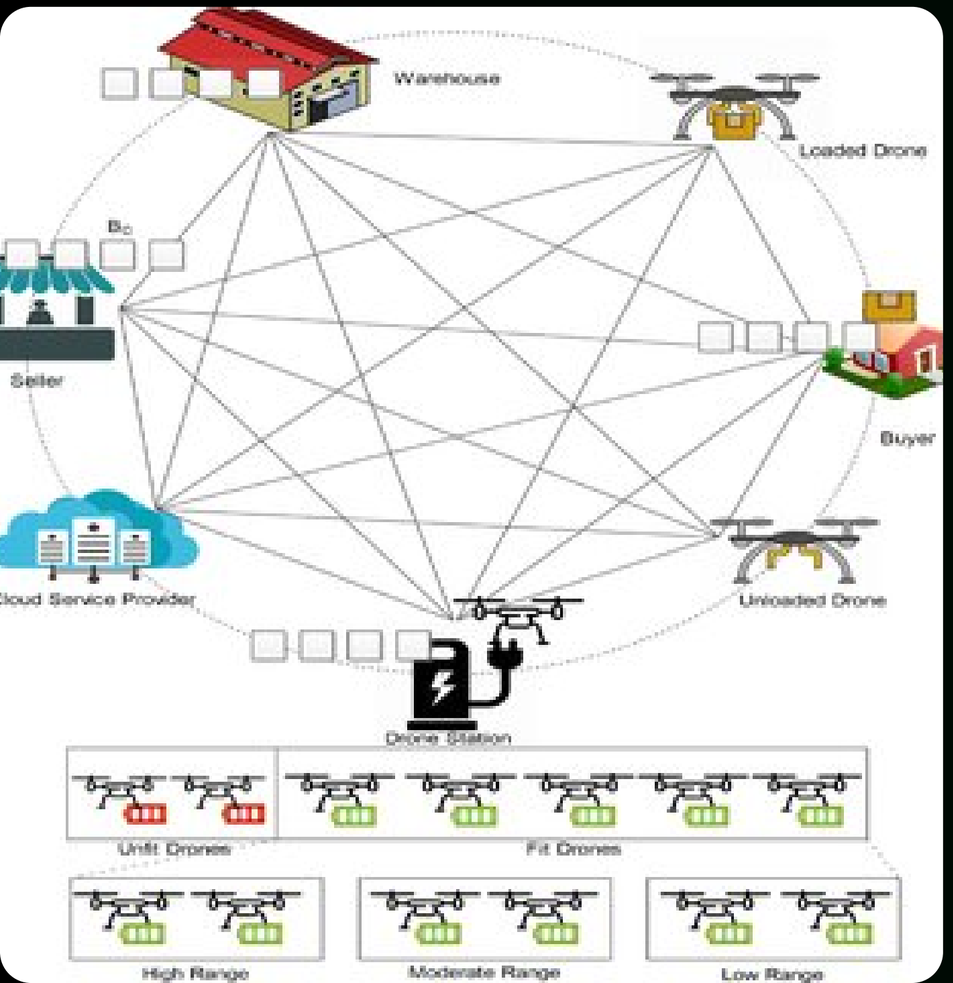
Step Diagram:

1. Build separate trees for each objective (time, fuel, etc.)
2. Merge into a combined weighted graph
3. Run single SOSP on merged graph

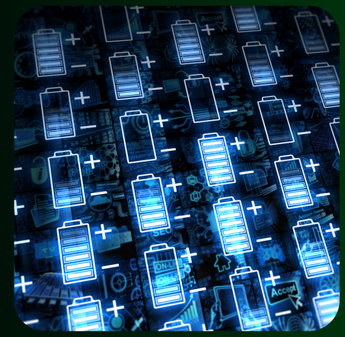
Result:

One near-optimal multi-objective path — fast, no full recomputation.

Drone Delivery Example

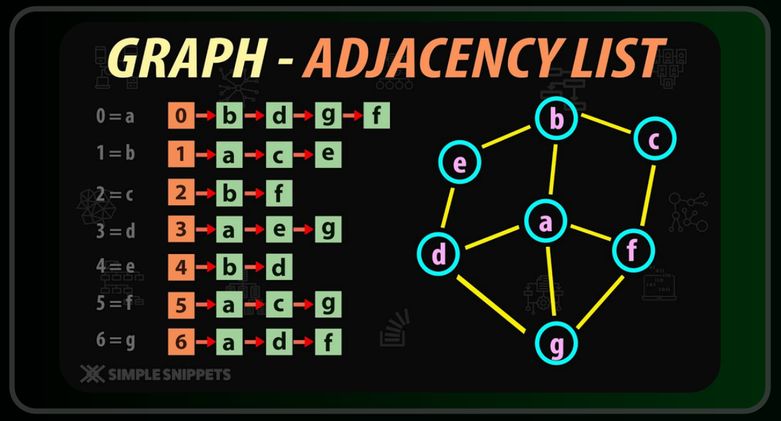


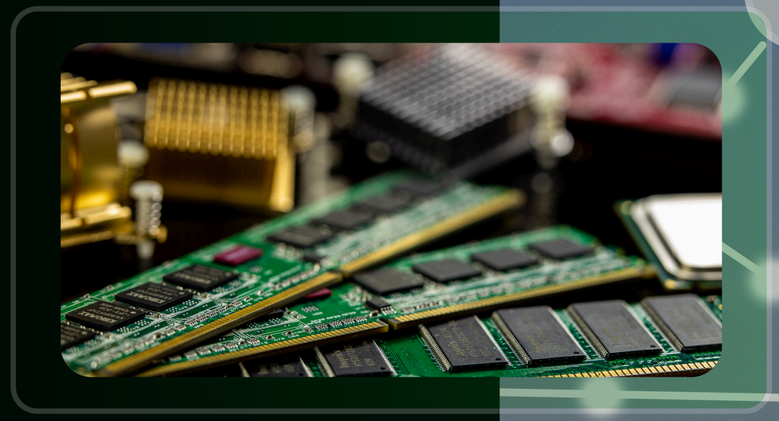
- Dynamic changes (e.g., wind, no-fly zones)
- Update time & battery SOSP trees
- Merge, get best balanced route
- Drone adapts in real time



Implementation Overview

- Language: C++ + OpenMP
- Graph: Adjacency List
- Hardware: 64 threads (Dual 32-core AMD), 64GB RAM
- Datasets: Road & Sensor Networks





Speedup: Up to 15× on large graphs (64 threads)

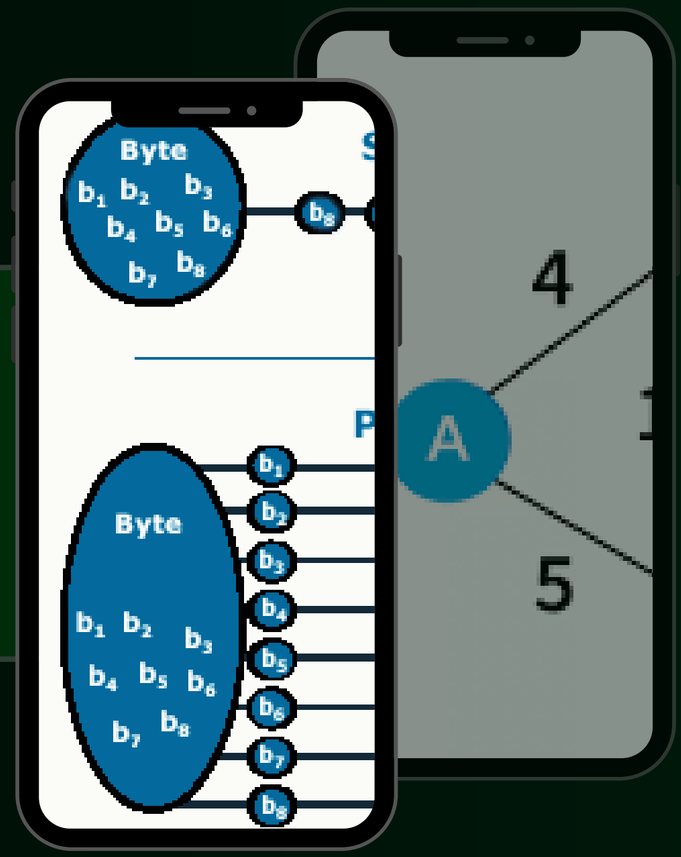
Most Time: Updating SOSP Trees

Scalable & Lightweight: Efficient on dynamic networks

Does It Actually Work?

How It Compares to Prior Research?

- Multi-objective Shortest Paths (MOSP)
- Parallel SOSP
- Dynamic MOSP
- Parallel + Dynamic MOSP (← Our focus)



Helps real-time apps like GPS, Drones, IoT systems.

Combines speed (via parallelism) + accuracy (multi-objectives).

Saves time, energy, and resources in large networks.

Why It Matters?

Proposed Parallelization Strategies

MPI – Message Passing Interface

OpenMP / OpenCL

METIS

MPI (Message Passing Interface) – For Parallelism Across Multiple Computers (Inter-node)

- You break the map into parts using a tool like METIS. Each computer (node) gets one part.
- Each computer updates its own section of the map (multi-objective shortest paths).
- But some roads (edges) connect cities (nodes) from different computers.
- In those cases, computers talk to each other using MPI, like sending messages:
- MPI_Sendrecv: to exchange updates on shared boundaries.
- MPI_Bcast: to share common changes to all nodes when needed.
- This way, every computer keeps its section updated without doing duplicate work.

OpenMP – For Parallelism Within One Computer (Intra-node)

OpenMP (Good for CPUs)

- Inside each computer, we have multiple CPU cores.
- Instead of one core doing all the work, we use OpenMP to divide the update tasks across cores.
- For example, if you have to update shortest paths for 1,000 nodes:
- You say: #pragma omp parallel for
- Now all cores start updating different nodes at the same time.
- You just have to be careful with shared data, so two cores don’t overwrite each other’s work.

METIS – For Breaking the Graph into Chunks

- METIS helps you split a large graph into balanced smaller parts, kind of like cutting a pizza into even slices.
- Each slice goes to one computer.
- METIS makes sure:
  - The slices are evenly sized (each computer has similar work).
  - Few connections are cut, so computers don’t have to talk too much.
- Along with each slice, we also pass info about connections to other slices, so border updates work correctly.

THANK YOU

FOR YOUR ATTENTION

From Team: Taha, Ahmed, Haziq