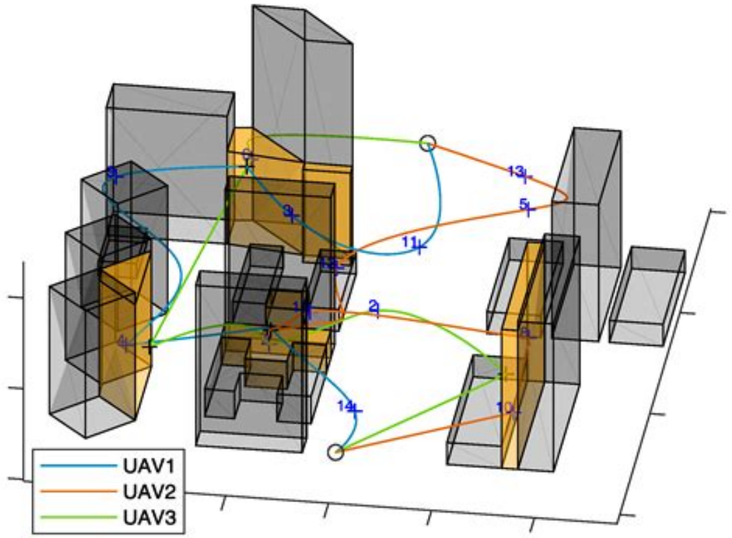
Electrical Engineering
Association

UAV
PATH PLANNING

By Abhinav Mishra

Github Fork

INTRODUCTION



The given problem statement explores the various present Path Planning Algorithms for the use case of Unmanned Aerial Vehicles.

It aims to discuss the various search algorithms such as BFS, DFS, Djikstras and A* for the use of finding the least time and least energy pathways that a UAV can take to go from a given point A to B

BACKGROUND

GENERAL OVERVIEW

At it's heart the problem concerns with leveraging Graph Theory and traversals algorithm to find the most efficient path in a given environment optimizing for either time or energy or if possible then both.

HISTORY

Integrates old graph traversal algorithms such as A* and Djikstras to tackle modern tasks including but not limited to obstacle avoidance and searching optimized pathways.

GOALS

The goal of this project is to bridge the gap between fundamental graph-search algorithms—such as BFS, DFS, and A*—and their practical application in UAV coordinate mapping and navigation. By implementing these algorithms, the project aims to solve the optimization challenge of finding efficient, obstacle-free trajectories for autonomous drones in simulated environments. Ultimately, the objective is to enable precise path planning and complex 3D maneuvers, such as epicycle approximations for formation-based aerial shows.

GRAPHS

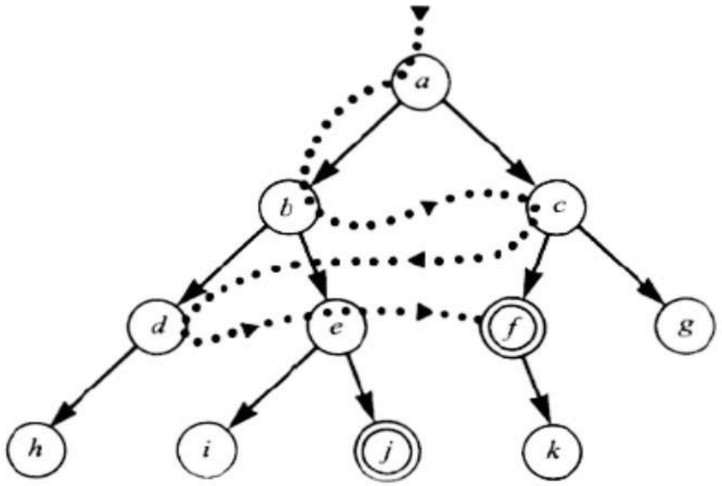
A graph is a structure consisting of a set of objects (nodes or vertices) where some pairs of objects are connected by links (edges). They are used to represent networks, such as road maps or UAV flight paths.

Types of Graphs

- Undirected Graph: Edges have no direction: the connection between nodes is mutual.
- Directed Graph (Digraph): Edges have a specific direction (arrows), showing one-way movement.
- Weighted Graph: Edges have "weights" (costs) representing distance, time, or energy.
- Unweighted Graph: All edges are considered equal in cost.
- Cyclic Graph: Contains at least one "cycle" (a path that starts and ends at the same node).
- Bipartite Graph: A graph whose nodes can be split into two groups such that no two nodes in the same group are connected.

BFS

Breadth First Search (BFS)

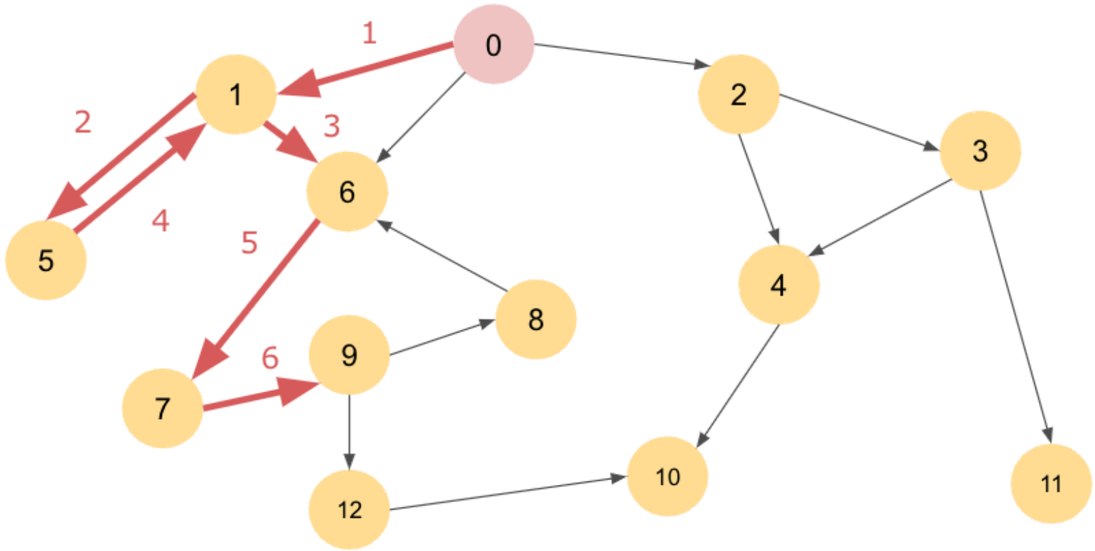


BFS

- Breadth-First Search (BFS) is a fundamental graph traversal algorithm that explores nodes layer-by-layer, starting from a source vertex and visiting all its immediate neighbors before moving to the next level of depth. This strategy ensures that when a goal is reached in an unweighted graph, the path taken is the shortest possible in terms of the number of edges. In the context of UAV path planning, BFS is utilized to systematically search for valid coordinates or grid cells, often employing a queue-based data structure to track the next nodes to be explored.

DFS

Depth First Search

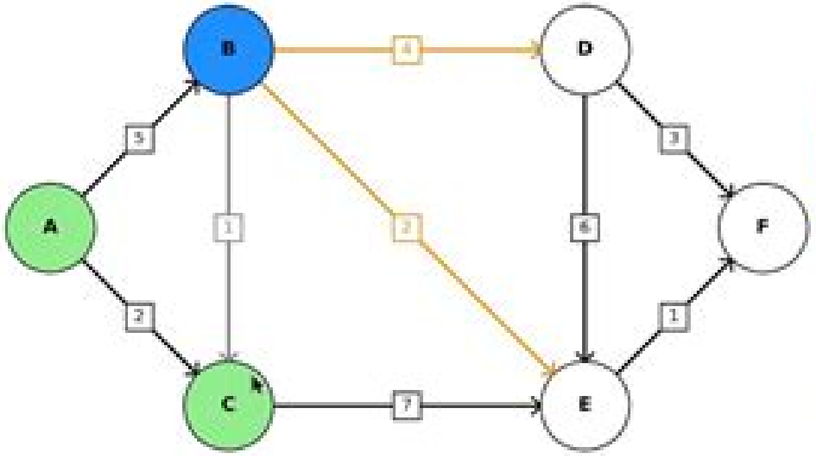


DFS

Depth-First Search (DFS) is a graph traversal algorithm that explores as far as possible along each branch before backtracking. Starting from a root node, it moves to an unvisited neighbor and continues deeper into the graph until it reaches a leaf node or a node with no unvisited neighbors. In UAV path planning, DFS is particularly useful for tasks like identifying cycles in a network of waypoints or checking for the general connectivity of a search space.

DJIKSTRAS

Dijkstra's Algorithm



Node	Distance	Parent
A	0	
B	5	A
C	2	A
D	∞	
E	9	C
F	∞	

B is now the open node with the smallest distance

Closed

A	C

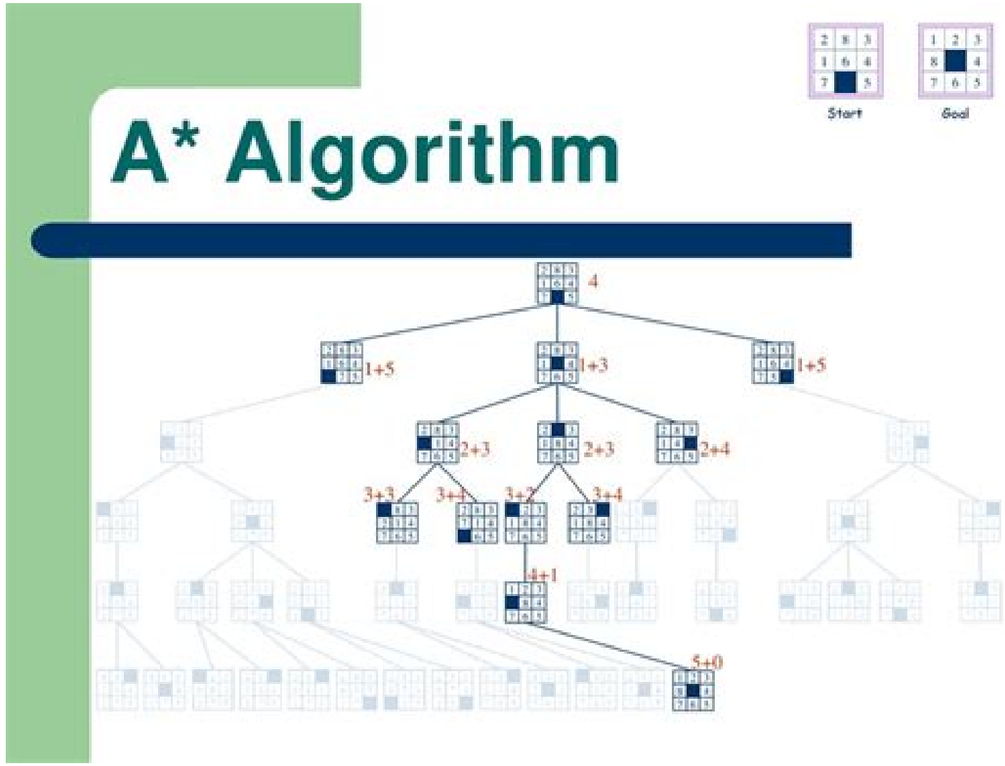
Open

B	D	E

DJIKSTRAS

Dijkstra's Algorithm is a greedy search algorithm used to find the shortest path between a starting node and all other nodes in a weighted graph. It works by maintaining a set of "visited" nodes and continuously updating the shortest known distance from the source to every other vertex, always choosing the unvisited node with the smallest cumulative cost as the next step. In UAV path planning, this is crucial for navigating environments where different movements have varying costs, such as energy consumption or distance, ensuring the drone reaches its destination via the most efficient route possible.

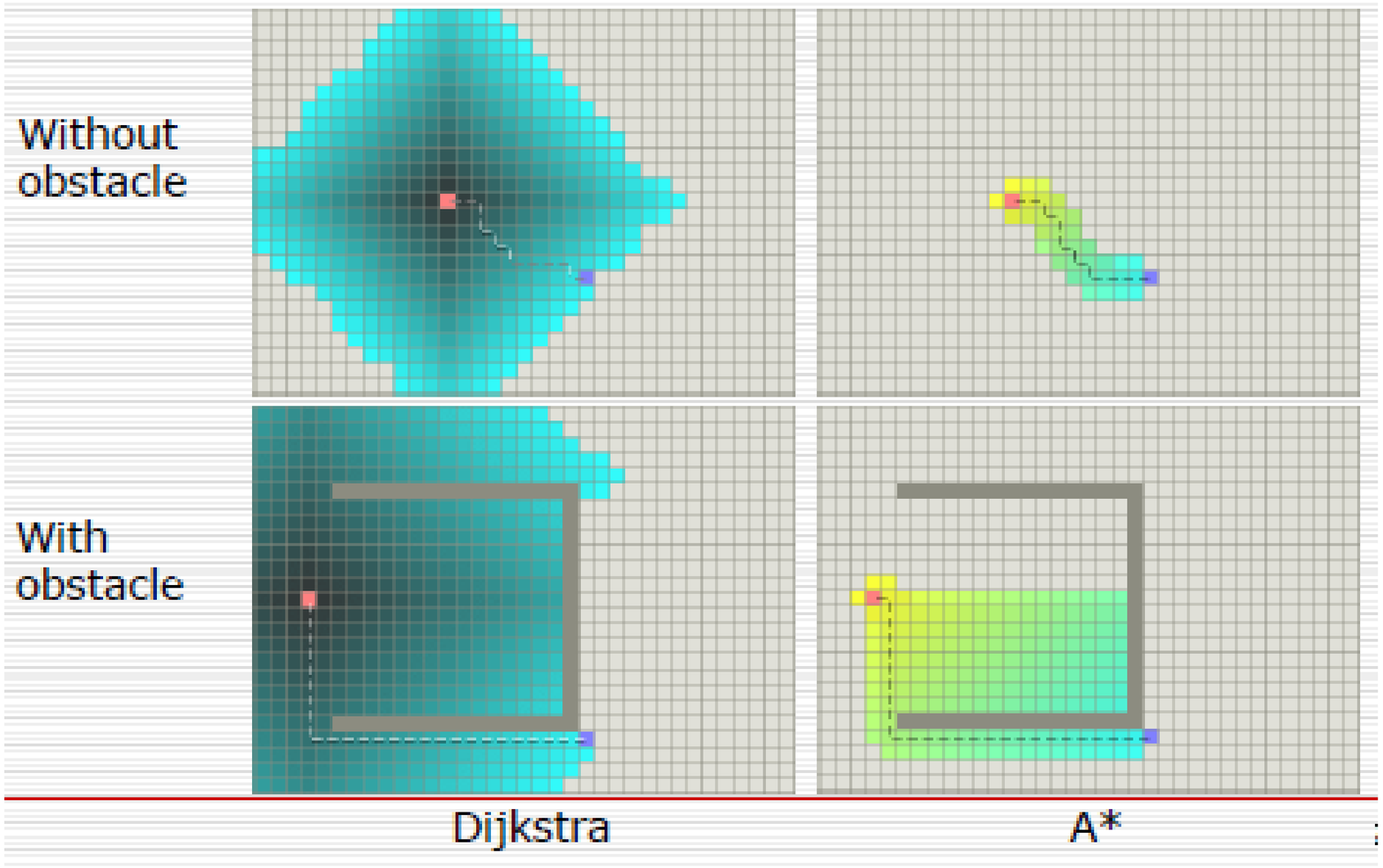
A*



A*

The A* (A-star) algorithm is an "informed" search method that improves upon Dijkstra's by using a heuristic to guide its search. It determines the best path by minimizing the function $f(n) = g(n) + h(n)$, where $g(n)$ is the actual cost from the start to the current node, and $h(n)$ is the estimated cost (heuristic) to the goal. In UAV path planning, $h(n)$ is often the straight-line Euclidean distance to the destination, allowing the drone to ignore irrelevant directions and focus its search toward the target. This makes A* significantly more computationally efficient than Dijkstra's while still guaranteeing the shortest possible path, provided the heuristic never overestimates the actual cost.

COMPARISON



COMPARISON

Informed vs. Uninformed Search: Dijkstra's is an uninformed search that only knows the cost from the start to the current node (g(n)); A* is an "informed" search that uses a heuristic (h(n)) to estimate the distance remaining to the goal.

Search Efficiency: Dijkstra's explores nodes in all directions equally (like an expanding circle), whereas A* uses its heuristic to prioritize nodes that appear to be on a direct path to the destination.

Reduced Complexity: By focusing the search toward the target, A* "prunes" or ignores many irrelevant nodes that Dijkstra's would otherwise spend time calculating.

Total Cost Function: A* makes decisions based on the combined cost f(n) = g(n) + h(n), whereas Dijkstra's only considers f(n) = g(n).

Speed in UAV Planning: In practical UAV applications, such as those involving the coordinate mapping and pathfinding tasks you are working on, A* finds the optimal trajectory faster and with less computational memory than Dijkstra's.

FUTURE WORK

- **Dynamic Obstacle Avoidance and 3D Complexity:** Transition from static maps to dynamic environments where UAVs can detect and bypass moving obstacles in real-time. This involves expanding coordinate mapping to include complex, multi-layered 3D urban environments with varying altitudes and no-fly zones.
- **Multi-UAV Swarm Coordination and Advanced Formations:** Implement decentralized algorithms for synchronized "swarms" to perform light shows or search-and-rescue. This would build upon your existing epicycle approximation work to allow for intricate 3D formation transitions.
- **Energy-Aware Planning and Path Smoothing:** Refine search algorithms like A* and Dijkstra's to prioritize energy conservation (accounting for battery life and wind). Additionally, apply smoothing techniques like cubic splines to turn discrete search results into fluid, flyable trajectories.

Electrical Engineering
Association

THANK YOU

By Abhinav Mishra

Github Repository