Motion Planning – Dijkstra's Algorithm

Leo Jaeeun Kim
University of Michigan
General Motors

ABSTRACT

Motion planning is a critical function in autonomous systems, enabling intelligent agents such as mobile robots, autonomous vehicles, and drones to navigate safely and efficiently through complex environments while minimizing cost metrics such as distance, time, or energy. Within the broader decision-making architecture, motion planning operates at the lowest level of a hierarchical framework, following vehicle routing at the high level and behavior planning at the mid-level. This layered structure reflects increasing granularity in decision-making, where vehicle routing addresses abstract task assignments and navigation across road networks, behavior planning manages tactical decisions like lane changes and overtaking, and path planning produces collision-free routes in the immediate operational space.

Among classical deterministic methods, Dijkstra's algorithm remains prominent due to its completeness and ability to compute globally optimal paths in known, static environments with non-negative edge weights. This study examines the theoretical underpinnings and practical applications of Dijkstra's algorithm within robotic navigation and Advanced Driver Assistance Systems (ADAS) features such as autonomous emergency braking, lane-change assistance, and path-following control. The algorithm is implemented on discretized 2D grid maps and evaluated across varying obstacle densities to assess key performance indicators including path optimality, node expansion, and computational time. A comparative analysis with A*, a heuristic-augmented extension of Dijkstra's method, reveals the trade-offs between optimality and computational efficiency.

To provide clarity on its functional scope, this work also distinguishes between path planning and trajectory optimization. While path planning focuses on geometric routes without explicit time considerations, trajectory optimization involves time-parameterized trajectories that satisfy vehicle kinematic and dynamic constraints—often formulated as optimal control problems. The study further explores the use of graph-based methods such as A*, BFS, and DFS; sampling-based methods like RRT and PRM; and learning-based approaches incorporating reinforcement learning. To address scalability in embedded systems and high-resolution environments, enhancements including hierarchical frameworks and hybrid search strategies are discussed. Case studies involving mobile robots and autonomous vehicles underscore the practical relevance of Dijkstra's algorithm in structured and constrained settings. Overall, the findings reaffirm its foundational role in motion planning while contextualizing its integration within hybrid, probabilistic, and learning-based planning architectures.

INTRODUCTION

Motion planning is a core component of robotic and autonomous systems, responsible for computing feasible and often optimal trajectories that guide an agent from a start location to a destination while avoiding obstacles and adhering to motion constraints. Real-world motion planning problems become significantly more complex due to dynamic environments, nonlinear system dynamics, and real-time execution requirements. To manage this complexity, motion planning is typically framed within a hierarchical planning architecture composed of three primary layers: vehicle routing, behavior planning, and path planning. This structure mirrors the level of abstraction required at each stage—vehicle routing deals with high-level navigation across road networks, behavior planning focuses on tactical maneuvers in dynamic scenarios, and path planning resolves local motion within constrained environments.

In structured environments with complete map availability, deterministic planning algorithms such as Dijkstra's offer valuable guarantees. Proposed in 1956 by Edsger W. Dijkstra, the algorithm is a greedy, label-setting method that solves the single-source shortest path problem on graphs with non-negative edge weights. In robotic systems, the environment is typically discretized into a 2D occupancy grid or graph where nodes represent spatial positions and edges encode valid transitions with associated costs. Dijkstra's algorithm iteratively expands the node with the lowest tentative cost, guaranteeing the discovery of an optimal path once the goal node is reached.

Despite its optimality and deterministic nature, Dijkstra's algorithm is computationally intensive in large or high-resolution spaces due to exhaustive exploration. In contrast, heuristic-based methods like A* reduce search effort by guiding node expansion toward the goal. Both methods are widely used in path planning, which differs fundamentally from trajectory optimization. Path planning focuses on generating a geometric sequence of waypoints from start to goal, abstracting away time and dynamic feasibility. In contrast, trajectory optimization involves time-parameterized paths that respect vehicle dynamics, making them physically executable. These are often solved using techniques from optimal control theory.

Modern motion planning systems increasingly integrate various algorithmic families to balance performance and feasibility. These include graph-based planners (Dijkstra, A*, BFS, DFS), sampling-based planners (RRT, RRT*, PRM), and learning-based planners that leverage reinforcement learning for generalization in unstructured or high-dimensional environments. As planning systems evolve to meet the demands of embedded and real-time applications, methods such as

hierarchical search and hybrid planning architectures become essential. This paper provides an in-depth exploration of Dijkstra's algorithm within this context, highlighting both its foundational role and adaptability in complex planning pipelines.

METHOD

The algorithm is widely used in various applications due to its deterministic behavior and optimal path generation. However, its main limitation lies in its exhaustive exploration, which can be computationally expensive for large or high-dimensional spaces. This paper aims to provide a detailed examination of Dijkstra's algorithm in the context of motion planning, analyze its strengths and limitations, and compare it with heuristic-based methods such as A*. Dijkstra's algorithm is designed to find the shortest path from a source node to all other nodes $v \in V$ in a weighted graph $G = (V,E)$, where each edge $(u, v) \in E$ has a non-negative weight $w(u, v)$.

Let $d(v)$ is the tentative shortest distance from s to v, Q for a priority queue(min-heap) of nodes ordered by d(v), and $\pi(v)$ the predecessor of node v in the shortest path.

Initialize with $d(s) = 0$, $d(v) = \infty$, $\pi(v) = NULL$. Insert all nodes into Q based on their $d(v)$ values.

In main loop, while Q is not empty, extract the node u with the minimum d(u) from Q. For each neighbor $v \in Adj(u)$:

$$if\ d(u) + w(u,v) < d(v) then:$$
$$d(v) \leftarrow d(u) + w(u,v)$$
$$\pi(v) \leftarrow u$$
$$Update\ v\ in\ Q$$

The algorithm terminates once all nodes have been visited or when the goal node's shortest distance is finalized. The resulting set of $\pi(v)$ pointers can be traced back from the goal to reconstruct the optimal path.

Gird-Based Discretization

In practical robotic applications, the continuous environment is discretized into a 2D occupancy grid. Each cell (i,j) in the grid corresponds to a node in the graph. Free cells are connected to their neighbors (4 or 8-connected) using edges with the following weights:

$$w\big((i,j),(i',j')\big) = \begin{cases} 1, & if\ |i-i'| + |j+j'| = 1 \\ \sqrt{2}, & if\ |i-i'| = |j+j'| = 1 \\ \infty, & if\ either\ cell\ is\ an\ obstacle \end{cases}$$

Edge Cost Representation

For generality, the cost of moving from node u to v is given by:

$$w(u,v) = c_d + c_o$$

where c_d is distance-based cost and c_o is additional cost due to terrain difficulty, slope, or other domain-specific penalties.

IMPLEMENTATION

The implementation of Dijkstra's algorithm in a grid-based environment utilizes an implicit graph representation, where each node corresponds to a free cell in the occupancy grid and edges are derived from the neighboring cells, either in 4-connected or 8-connected fashion. The graph structure is efficiently handled through adjacency lists or neighborhood queries during runtime. To ensure efficient retrieval of the minimum-cost node, a binary heap is used to implement the priority queue, which maintains the frontier of unexplored nodes sorted by their tentative distances. A mapping of predecessor nodes, $\pi(v)$, is preserved throughout the execution to reconstruct the optimal path upon completion of the search. The algorithm achieves a time complexity of $O((V+E)logV)$, where V and E denote the number of nodes and edges, respectively, assuming a min-heap is used for the priority queue. This implementation setup provides the foundation for evaluating the performance of Dijkstra's algorithm in varied environmental configurations, including scenarios with increasing map size and obstacle density.

Path Smoothing with Splines

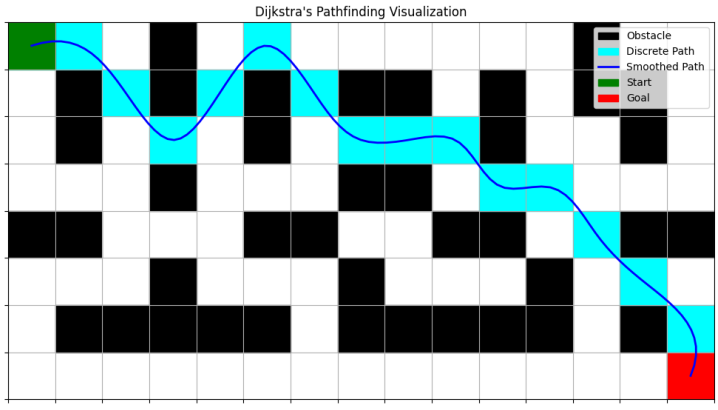
Once the discrete path is computed, a cubic B-spline is fit through the waypoints using SciPy's splprep() and splev() functions. This smooths the transitions between grid points, producing a continuous and differentiable curve suitable for control layers or trajectory tracking.

The B-spline smoothing generates a function $\gamma(u)=(x(u),y(u))$ for $u\in[0,1]$, ensuring:

- Smoothness: continuous first and second derivatives
- Accuracy: passes through or near original waypoints
- Feasibility: eliminates abrupt turns in sharp corners

Grid and Visualization

The grid is visualized using matplotlib, where obstacles are rendered in black, the raw grid path in cyan, and the smoothed path as a blue curve. Start and goal positions are highlighted in green and red, respectively.



The example code for Dijkstra algorithm in motion planning can be found in:
https://github.com/woa0425/Leo_ADAS_script/blob/main/Motion_Planning/Motion_Planning_Dijkstra_python/Motion_Planning_Dijkstra.py

To assess the effectiveness of the proposed motion planning algorithm, we applied the 8-directional Dijkstra's algorithm with spline-based path smoothing to a structured 15×8 grid environment populated with static obstacles. The environment mimics constrained navigation scenarios relevant to ADAS applications, such as parking layouts or road edges, where obstacle avoidance and smooth steering are critical.
The planner successfully identified a valid and optimal path from the top-left corner (start) to the bottom-right corner (goal), circumventing multiple obstacles. The path was constructed using both cardinal and diagonal steps, minimizing total cost under Euclidean metrics. After computing the discrete path, a cubic B-spline was fitted to the resulting waypoints, generating a continuous trajectory with smooth curvature transitions.
The reported total path cost is 15.00, calculated as the sum of Euclidean distances over all segments in the discrete path. The smooth curve maintains close adherence to the optimal path while providing continuous curvature for downstream controllers.

CONCLUSION

This work presents a curvature-aware motion planning strategy built upon a classical Dijkstra's algorithm with 8-directional movement and B-spline-based trajectory smoothing. The planner guarantees optimality within the grid discretization and addresses practical ADAS challenges by producing geometrically smooth and kinematically feasible paths.

The implementation demonstrates strong performance in structured static environments, with the ability to navigate narrow passages and generate clean, steerable paths without the need for heuristic tuning. The smooth output makes it directly compatible with vehicle controllers that require continuous curvature input for actuation.

Future extensions will involve dynamic re-planning under moving obstacles, adaptive grid resolution, and integration with vehicle dynamics constraints for curvature, speed, and jerk limits. These enhancements will further bridge the gap between discrete path planning and real-time motion execution in modern ADAS platforms.

REFERENCES

1. E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, vol. 1, 1959.
2. S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.
3. P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Trans. Systems Science and Cybernetics, 1968.
4. H. Choset et al., *Principles of Robot Motion*, MIT Press, 2005.