Path Planning

Mahmoud Esam Ali

September 19, 2023

Abstract

path planning algorithms are used in mobile robots to generate a geometric path, from an initial point to a final point safely and collision-free. Choosing the most fit algorithm help robot to navigate safely and effectiveness. Choosing the algorithm depends on the robot geometry and computing constraints. Keywords: path planning; A*; D*; dijkstra; RRT; genetic; ant colony; Firefly

1 Introduction

Path planning is the process of finding a collision-free path from a start state to a goal state in a given environment. It is a fundamental problem in robotics, autonomous vehicles, and other applications where robots or vehicles need to move around safely and efficiently. Path planning algorithms can be classified into two main categories:

1.1 Search-based algorithms

These algorithms explore the environment systematically to find a path. Some common search-based algorithms include Dijkstra's algorithm, A*, and IDA*.

1.2 Sampling-based algorithms

These algorithms randomly sample the environment to find a path. Some common sampling-based algorithms include Rapidly-exploring Random Trees (RRTs) and Probabilistic Roadmaps.

The choice of path planning algorithm depends on a number of factors, including the environment in which the robot or vehicle is operating, the constraints on the path (e.g., distance, time, and energy), and the computational resources available.

2 Path Planning Algorithm

2.1 Search-based algorithms

Search-based algorithms explore the environment systematically to find a path. They work by building a graph of the environment, where the nodes in the graph represent possible states of the robot or vehicle and the edges in the graph represent possible transitions between states. The algorithm then searches the graph to find a path from the start state to the goal state. Some common search-based algorithms.

2.1.1 Dijkstra's algorithm

Dijkstra's algorithm is a greedy algorithm that finds the shortest path from a single source node to all other nodes in a graph. It works by iteratively expanding a frontier of nodes, starting from the source node. At each iteration, the algorithm adds the node with the shortest distance from the source node to the frontier. The algorithm terminates when the frontier reaches the goal node.

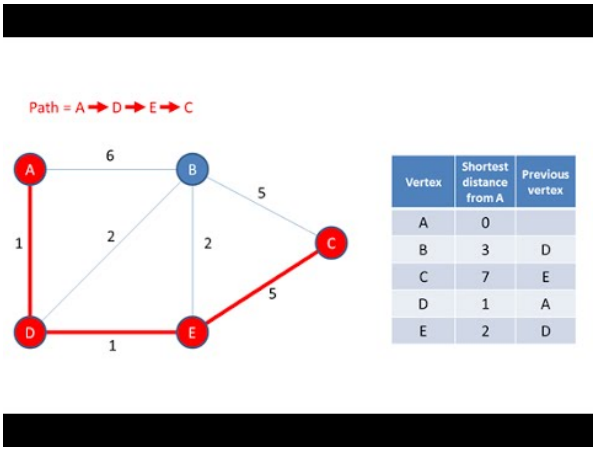
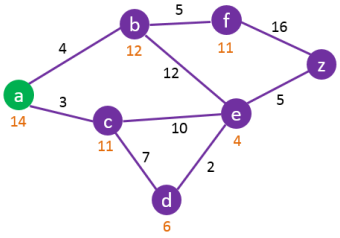


Figure 1: Dijkstra's short path algorithm.

2.1.2 A algorithm

The A* algorithm is an informed search algorithm that combines Dijkstra's algorithm with a heuristic function to estimate the distance from a node to the goal node. This allows the A* algorithm to find shorter paths than Dijkstra's algorithm.



Node	Status	Shortest Distance From A	Heuristic Distance to Z	Total Distance*	Previous Node
A	Current	0	14	14	
B		∞	12		
C		∞	11		
D		∞	6		
E		∞	4		
F		∞	11		
Z		∞	0		

* Total Distance = Shortest Distance from A + Heuristic Distance to Z

Figure 2: A* search algorithm.

2.1.3 IDA algorithm

The IDA* algorithm is a depth-first search algorithm that finds the shortest path from a single source node to a goal node in a graph. It works by iteratively increasing a depth bound and searching for a path to the goal node within the depth bound. The algorithm terminates when a path to the goal node is found within the depth bound.

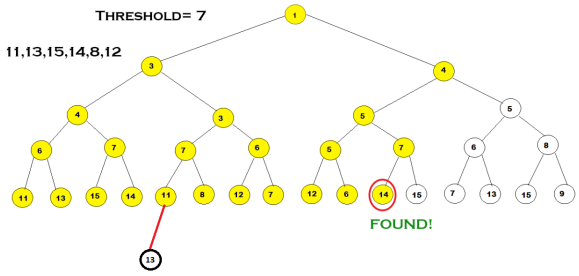


Figure 3: IDA* search algorithm.

2.2 Sampling-based algorithms

Sampling-based algorithms randomly sample the environment to find a path. They work by generating a set of random samples in the environment and then connecting the samples with edges to form a graph. The algorithm then searches the graph to find a path from the start state to the goal state. Some common sampling-based algorithms

2.2.1 Rapidly-exploring Random Trees (RRTs)

A Rapidly-exploring Random Tree (RRT) is a data structure and path planning algorithm that is designed for efficiently searching paths in nonconvex high-dimensional spaces. RRTs are constructed incrementally by expanding the tree to a randomly-sampled point in the configuration space while satisfying given constraints, e.g., incorporating obstacles or dynamic constraints (nonholonomic or kinodynamic constraints). While an RRT algorithm can effectively find a feasible path, an RRT alone may not be appropriate to solve a path planning problem for a mobile robot as it cannot incorporate additional cost information such as smoothness or length of the path. Thus, it can be considered as a component that can be incorporated into the development of a variety of different planning algorithms.

GENERATE_RRT($x_{init}, K, \Delta t$)
1 $\mathcal{T}$.init(x_{init});
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x_{rand}, \mathcal{T}$);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 $\mathcal{T}$.add_vertex(x_{new});
8 $\mathcal{T}$.add_edge(x_{near}, x_{new}, u);
9 Return $\mathcal{T}$

Figure 4: Pseudo code of an RRT

2.2.2 Probabilistic Roadmaps (PRMs)

PRMs are another class of sampling-based algorithms that find paths in high-dimensional spaces. They work by randomly sampling a set of nodes in the environment and connecting them with edges to form

a roadmap. The algorithm then finds a path from the start node to the goal node by following the edges of the roadmap.

Algorithm 6 Roadmap Construction Algorithm

Input:

n : number of nodes to put in the roadmap

k : number of closest neighbors to examine for each configura

Output:

A roadmap $G = (V, E)$

```
1:  V ← ∅
2:  E ← ∅
3:  while |V| < n do
4:     repeat
5:        q ← a random configuration in Q
6:     until q is collision-free
7:     V ← V ∪ {q}
8:  end while
9:  for all q ∈ V do
10:    N_q ← the k closest neighbors of q chosen from V according
11:    for all q' ∈ N_q do
12:       if (q, q') ∉ E and Δ(q, q') ≠ NIL then
13:          E ← E ∪ {(q, q')}
14:       end if
15:    end for
16: end for
```

Figure 5: Pseudo code of an PRM

2.3 Choice of path planning algorithm

2.3.1 Environment

The environment in which the robot or vehicle is operating will affect the choice of path planning algorithm. For example, if the environment is static, then a search-based algorithm may be a good choice. However, if the environment is dynamic, then a sampling-based algorithm may be a better choice.

2.3.2 Constraints on the path

The constraints on the path, such as distance, time, and energy, will also affect the choice of path planning algorithm. For example, if the path needs to be found quickly, then a sampling-based algorithm may be a better choice.

2.3.3 Computational resources available

The computational resources available will also affect the choice of path planning algorithm. For example, if the robot or vehicle has limited computational resources, then a sampling-based algorithm may be a better choice

2.4 Conclusion

Path planning is a fundamental problem in robotics, autonomous vehicles, and other applications where robots or vehicles need to move around safely and efficiently. There are a variety of path planning algorithms available, each with its own strengths and weaknesses. The choice of path planning algorithm depends on the specific application.

3 References

Path Planning with Rapidly-exploring Random Trees (RRTs) by Steven M. LaValle. Planning Algorithms by Steven M. LaValle. A Survey of Path Planning Algorithms for Mobile Robots by Karthik Karur.