ME5413: Autonomous Mobile Robot

Homework 3: Planning

Name: WANG Yexiang Matric Number: A0304719A

Task 1: Global Planning

Overview

A* (A-star) is an efficient pathfinding algorithm widely used in graph search and path planning. It combines the advantages of Breadth-First Search (BFS) and Greedy Best-First Search, ensuring both optimality and efficiency.

A* evaluates each node using the following cost function:

$$f(n)=g(n)+h(n)$$

where:
$g(n)$ is the actual cost from the start node to the current node n.
$h(n)$ is the heuristic estimate of the cost from node n to the goal, guiding the search direction.

Algorithm Steps

1. **Initialization**: Insert the start node into a priority queue (Min-Heap), setting g(start)=0 and $f(start)=h(start)$.

2. **Search Loop**:
1) Extract the node with the smallest f-value (i.e., the current best path node).
2) If this node is the goal, the search ends, and the path is reconstructed.
3) Otherwise, add its neighbors to the queue and update their ***g*** and ***f*** values if a better path is found.

3. **Repeat** step 2 until the goal is found or the queue is empty (no solution).

Characteristics

・**Optimality**: If the heuristic function h(n) is admissible (i.e., never overestimates the actual cost to the goal), A* guarantees the shortest path.
・**Efficiency**: A well-designed heuristic can significantly reduce the search space and improve speed.

Below are the results of the heuristic algorithm using Euclidean distance:

	start	snacks	store	movie	food
start	0.0	705.26	772.15	892.11	1094.44
snacks	705.26	0.0	571.63	532.72	648.50
store	772.15	571.63	0.0	1042.60	554.35
movie	892.11	532.72	1042.60	0.0	556.90
food	1094.44	648.50	554.35	556.90	0.0

In the code for this assignment, Manhattan distance and other heuristics can also be used as the heuristic function.

Task 2 (Bonus): The “Travelling Shopper” Problem

This is a variant of the classic Traveling Salesman Problem (TSP), where we need to start from an initial location, visit all given locations, and return to the starting point while minimizing the total travel distance.

Problem Modeling

- We have five locations: **start** (starting/ending point), **snacks**, **store**, **movie**, and **food**.
- The distances between each pair of locations have already been calculated in Task 1.
- Objective: Find a path that starts at start, visits all other locations exactly once, and returns to start with the shortest total distance.

Possible Solutions

1. **Brute Force**: Enumerate all possible paths, calculate their total distances, and select the shortest one. This method is feasible for a small number of locations.
2. **Dynamic Programming (Held-Karp Algorithm)**: Uses a DP-based approach with a time complexity of $O(n^2 2^n)$, significantly faster than brute force.
3. **Greedy Algorithm (Nearest Neighbor)**: Always chooses the nearest unvisited location at each step. It is simple and fast but does not guarantee the optimal solution.
4. **Genetic Algorithm**: Simulates an evolutionary process to solve complex combinatorial optimization problems.
5. **2-opt Algorithm**: Starts with an initial path and iteratively improves it by swapping two edges to reduce the total distance.

In this task, I will implement the brute force method and the nearest neighbor greedy algorithm to solve this problem. The final results are as follows.

```
Brute Force Enumeration:
Shortest Path: start -> snacks -> movie -> food -> store -> start
Total Distance: 3121.38 meters

Nearest Neighbor Greedy Algorithm:
Path: start -> snacks -> movie -> food -> store -> start
Total Distance: 3121.38 meters
```

Comparative Analysis

Optimality:

- The brute force method guarantees finding the global optimal solution (shortest path).
- The nearest neighbor algorithm follows a greedy strategy, which may get stuck in a local optimum and does not necessarily find the global best solution.

Time Complexity:

- **Brute force method**: $O((n-1)!)$, which grows rapidly as the number of locations increases.
- **Nearest neighbor algorithm**: $O(n^2)$, much more efficient and suitable for large-scale problems.

Results Comparison:

- In this small-scale problem with only 5 locations, the brute force method is likely to find the global optimal solution.
- Due to its greedy nature, the nearest neighbor algorithm usually produces a near-optimal solution, but it is not guaranteed to be the best.
- By comparing the total path lengths of both algorithms, we can observe the difference in their performance.