COMS 576 - Motion Planning for Robotics and Autonomous Systems,
Homework 1

Mobina Amrollahi

Deadline: 09/06/2024, 11:59 PM

Contents

1 8 points

Consider a robot navigating a grid-like warehouse, represented as a 2D grid. Each location in the warehouse is identified by a coordinate (x, y), where $x \in \{0, 1, \ldots, X_{\max}\}$ represents the horizontal position (x-axis) and $y \in \{0, 1, \ldots, Y_{\max}\}$ represents the vertical position (y-axis) for some natural numbers $X_{\max}$ and $Y_{\max}$. The robot can move in the following directions:

- **Up:** This corresponds to adding 1 to the y-coordinate,
- **Down:** This corresponds to subtracting 1 from the y-coordinate,
- **Left:** This corresponds to subtracting 1 from the x-coordinate, or
- **Right:** This corresponds to adding 1 to the x-coordinate.

Some locations may contain obstacles, making them inaccessible to the robot. The goal is to compute an optimal path for the robot from a given starting location $l_i = (x_i, y_i)$ to a given target location $l_g = (x_g, y_g)$, ensuring that the path is collision-free.
The definition of "optimal" can vary depending on the criteria used. For each of the following criteria, your task is to:

- Identify the elements of the planning problem, including the state space X, the set U of actions, the action space $U(x)$ for each state $x \in X$, the state transition function f, and if applicable, the transition cost.
- Determine the most efficient algorithm for solving the problem based on the given criteria. Provide the time complexity of your chosen algorithm, and clearly state any assumptions made to derive the runtime.

Criteria:

(a) **Minimizing the Number of Moves:** Find a path that minimizes the total number of moves (or steps) required to reach the target location from the starting location.

(b) **Minimizing the Total Time:** Find a path that minimizes the total time taken to reach the target location, considering that the time required for each move may vary depending on both the move and the location where it is made.

(c) **Minimizing the Total Cost:** Each move has associated time and energy costs. Given a location l and a move m, let $c_{time}(l, m)$ and $c_{energy}(l, m)$ represent the time and energy, respectively, for making a move m at location l. For a sequence of moves $m_1 m_2 \ldots m_k$ that leads the robot through the corresponding path $p = l_1 l_2 \ldots l_{k+1}$, the total cost of the path is defined as

$$c(p) = w_{\text{time}} \sum_{i=1}^{k} c_{\text{time}}(l_i, m_i) + w_{\text{energy}} \sum_{i=1}^{k} c_{\text{energy}}(l_i, m_i),$$

where c_{time} and c_{energy} are non-negative functions and $w_{\text{time}}, w_{\text{energy}} \in \mathbb{R}_{\geq 0}$ are weights representing the relative importance of time and energy in the cost function.

(d) (**COM S 5760 only**) **Minimizing the Hierarchical Cost:** In this scenario, time is considered to be strictly more important than energy. Therefore, you need to select a path that has the least energy consumption among all paths that achieve the minimum total time. Specifically, let P_{time} denote the set of paths that minimize the total time taken to reach the target location. Then, select the one with the least energy consumption among all paths in P_{time}.

Hint: You do not need to explicitly identify the set P_{time}. Instead, compute the optimal path directly by considering both time and energy costs in a hierarchical manner, where minimizing time is the primary objective and minimizing energy is the secondary objective among those time-optimal paths.

1.1 Answer

Since the grid, the way the robot moves, and the location of obstacles are fixed. The first four elements of planning, state space X, set U of actions, action space $U(x)$, and state transition function f are the same despite different criteria:

- State space X: In this problem, the state space is finite and countable. Based on the definition, $X = \{(x, y) \mid x \in \{0, 1, \ldots, X_{\max}\}\{0, 1, \ldots, X_{\max}\}, y \in \{0, 1, \ldots, Y_{\max}\}\}\backslash O$, where O refers to the set of obstacles.
- Set U of actions: $U = \{\underbrace{(0,1)}_{\text{Up}}, \underbrace{(0,-1)}_{\text{Down}}, \underbrace{(-1,0)}_{\text{Left}}, \underbrace{(1,0)}_{\text{Right}}\}$
- Action space $U(x)$: $\forall x \in X, U(x) = \{u \in U \mid f(x, u) \in X\}$ and $U = \cup_{x \in X} U(x)$.
- State transition function f: $f(x, u) = x + u$ in which $x \in X$ and $u \in U$. This is due to the discrete movement of the robot in the four directions.

1.1.1 (a) Minimizing the Number of Moves

Elements of the planning problem:

- The transition cost (if applicable): transitions have uniform cost which is 1 for this particular criteria.

The most efficient algorithm based on the given criteria:

- Algorithm: Breadth first. This method specifies Q as a First-In First-Out (FIFO) queue. Thus, all plans that have k steps are exhausted before plans with $k + 1$ step. Therefore, the solution found with this algorithm will guarantee that it will use the smallest number of steps [2].
- Run time: In the general case, the running time of this algorithm is $\mathcal{O}(|V| + |E|)$, where $|V|$ and $|E|$ are the numbers of vertices and edges. In this problem $|V| = |X|$. However, $|E|$ can represent different things based on the transition function. In this particular problem, V is the number of vertices (grid cells) and E is the number of edges (possible moves).
- Assumptions made to derive the runtime:
  (a) Setting up the loop takes a constant time c_1 [1].
  (b) Atomic operations in the loop, such as determining which state is visited, take a constant time c_2 [2].

1.1.2 (b) Minimizing the Total Time

Elements of the planning problem:

- The transition cost: Time required for each move, which varies based on the move and location.

The most efficient algorithm based on the given criteria:

- Algorithm: Dijkstra's Algorithm. In this algorithm, the priority Q will sorted according to the cost-to-come function $C : X \to [0, \infty)$. For each state x, the value of $C^*(x)$ is called the optimal cost-to-come from the initial state x_1. In this algorithm, $C^*(x)$ is calculated based on summing edge costs over all possible paths from x_1 to x, using the path that produces the least cumulative cost.
- Run time: In the general case, the running time of this algorithm is $\mathcal{O}(|V| \log |V| + |E|)$, where $|V|$ and $|E|$ are the numbers of vertices and edges. In this problem $|V| = |X|$. However, $|E|$ can represent different things based on the transition function.
- Assumptions made to derive the runtime:
  (a) Setting up the loop takes a constant time c_1 [1].
  (b) Atomic operations in the loop, such as determining which state is visited, take a constant time c_2 [2].
  (c) The priority queue is implemented with a Fibonacci heap[2].

1.1.3 (c) Minimizing the Total Cost

Elements of the planning problem:

- The transition cost (if applicable):

$$c(p) = w_{\text{time}} \sum_{i=1}^{k} c_{\text{time}}(l_i, m_i) + w_{\text{energy}} \sum_{i=1}^{k} c_{\text{energy}}(l_i, m_i),$$

  where c_{time} and c_{energy} represent the time and energy costs for making a move m_i at location l_i, and w_{time} and w_{energy} are the weights for time and energy, respectively.

The most efficient algorithm based on the given criteria:

- Algorithm: A^* algorithm.
- Run time: $O(\mid E \mid \log \mid V \mid)$ in the worst case, but often performs much better in practice.
- Assumptions made to derive the runtime:
  (a) We have an admissible heuristic function h(n) that estimates the cost from any node to the goal.
  (b) The graph is represented using an adjacency list.

1.1.4 (d) Minimizing the Hierarchical Cost

Elements of the planning problem:

- The transition cost (if applicable): Same as (c)

The most efficient algorithm based on the given criteria:

- Algorithm: Hierarchical A* Search or a two-phase approach:
  - **Phase 1:** Use A* or Dijkstra's to find the time-optimal path.
  - **Phase 2:** Among paths with minimal time, use a secondary optimization (like another A* run) to minimize energy.
- Run time: Depends on the combined approach but can be roughly $O(|X| \log |X|)$ for each phase.
- Assumptions made to derive the runtime: Time is the primary cost and must be minimized first, then energy among the time-optimal paths.

2 8 points

Please note that the instances of abstract classes are directly used to calculate certain things related to them. I understand that changing the classes derived from these abstract classes is more efficient than changing the abstract classes directly. However, during development, grasping this concept was hard for me. The classes are all in the python main.py

3 Problem 3 (4 points)

Please note that attached is a .json file for the input, which I made after discussing it with the TA. Once the algorithm is done, the search will be running two times, once to do the path planning with Astar and once with BFS. The output will most likely be printed in your output console.

4 Use of Generative AI Tools

4.1 Writing Problem Statements

In order to avoid going back and forth between this LATEXfile and homework question files, I used chatGPT to write homework question statements for LATEXand then copied them into this file.
In addition, the TA mentioned that our code has to be able to read the input of the third question from a "JSON" file. This was my first time experiencing reading data from a JSON file, and to make sure there was nothing wrong with the file, I sought chatGPT help in this manner as well.
Queues and classes were new concepts to me, and despite reaching out to the instructor, TA, and a couple of my friends, I still needed an interactive environment to ask my questions. In this regard, Claude Sonnet 3.5 Helped me a lot. I specifically trained the model not to reveal the answer completely, but help me step by step.
For question one, I hesitated between 1 or 2 choices of algorithms, so I asked Claude Sonnet 3.5 and ChatGPT to give their answers, read through, and decide which one I'd like to choose. Specifically for Question 1, part (d), I used ChatGPT's idea.

References

[1] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[2] SM Lavalle. Planning algorithms. *Cambridge University Press google schola*, 2:3671–3678, 2006.