EAST WEST UNIVERSITY

Assignment 1 Report

Course Name: Artificial Intelligence

Course Code: CSE-366

Section - 3

Assignment Name: Enhanced Dynamic Robot Movement Simulation

Submitted By

Name: Ismail Mahmud Nur

ID: 2021-2-60-052

Dept. of Computer Science & Engineering

Submitted To

Dr. Mohammad Rifat Ahmmad Rashid

Assistant Professor,

Department of Computer Science and Engineering

East West University

Enhanced Dynamic Robot Movement Simulation

Introduction

The Enhanced Dynamic Robot Movement Simulation aims to simulate the movement of a robot in a dynamic environment using various search algorithms. This report provides an overview of the code functionality, implementation details, and the results of running simulations.

Theory

In robotics and artificial intelligence, pathfinding algorithms play a crucial role in enabling autonomous agents to navigate complex environments efficiently. The Enhanced Dynamic Robot Movement Simulation project utilizes two common pathfinding algorithms: A* (A-star) and Uniform Cost Search (UCS). Below, we provide a theoretical overview of these algorithms and their significance in robotic motion planning.

A* Algorithm: A* is a widely-used algorithm for finding the shortest path between two points on a graph. It combines the advantages of both Dijkstra's algorithm and greedy best-first search. A* maintains two lists: open and closed. The open list contains nodes to be evaluated, while the closed list contains nodes that have already been evaluated. The algorithm selects the node with the lowest cost from the open list for expansion until it reaches the goal node or exhausts all possible paths.

Components:

- **Cost Function:** A* uses a cost function that combines the cost of reaching a node from the start node (known as the 'g' value) and the estimated cost of reaching the goal node from the current node (known as the 'h' value or heuristic).
- **Heuristic Function:** A heuristic function estimates the cost of reaching the goal from a given node. It guides the search towards the goal node by prioritizing nodes that are closer to the goal.
- **Optimality:** A* guarantees finding the shortest path if the heuristic function satisfies two conditions: admissibility (never overestimates the true cost) and consistency (satisfies the triangle inequality).

Uniform Cost Search (UCS): UCS, also known as Dijkstra's algorithm, is a classic algorithm for finding the shortest path in a weighted graph. Unlike A*, UCS does not use heuristics and instead explores all possible paths from the start node until reaching the goal node. It maintains a priority queue of nodes sorted by their path cost, expanding the node with the lowest cost first.

Components:

- **Cost Function:** UCS evaluates nodes based solely on the cost of reaching them from the start node.
- **Exploration Strategy:** UCS explores nodes in increasing order of path cost, ensuring that the shortest path is found when the goal node is reached.
- **Optimality:** UCS guarantees finding the shortest path in weighted graphs with non-negative edge costs.

Code Overview

The code consists of two main classes: Environment and Robot, as well as a script simulation.py to run simulations.

- **Environment Class:** This class represents the environment in which the robot operates. It includes methods for generating obstacles, visualizing the environment, and checking the validity of robot movements within the environment.

- **Robot Class:** This class represents the robot's behavior and movement. It includes methods for moving the robot, searching for the optimal path using different algorithms (A* and UCS), and simulating the robot's movement in the environment.

- **Simulation Script:** The simulation.py script serves as the main entry point for running simulations. It initializes the environment, creates a robot instance, and runs simulations using different search algorithms. It prompts the user to choose the algorithm and displays the simulation results.

Implemented Functionality

1. **Environment Generation:** The Environment class generates random obstacles within the environment grid.

2. **Robot Movement:** The Robot class allows the robot to move in different directions (up, down, left, right) within the environment grid while avoiding obstacles.

3. **Path Finding Algorithms:** The Robot class implements two search algorithms: A* (A-star) and Uniform Cost Search (UCS) to find the shortest path from the starting point to the destination.

4. **Battery Management:** The program also includes battery management for the robot during its movement in the environment. The battery management system ensures that the robot efficiently utilizes its power resources while navigating the environment and accounts for recharging when necessary. Here is the detail overview of how the battery management system works below:

- **Battery Level:** The robot's battery level is initialized to 100% at the start of the simulation. With each movement, the battery level decreases by 10 units to simulate power consumption.

- **Battery Depletion:** If the battery level reaches 0% during the simulation, the robot's movement is halted, and the system prompts for recharging. This prevents the robot from continuing its operation with insufficient power.

- **Recharging:** When the battery is depleted, the robot automatically initiates a recharging process. The battery is restored to 100% capacity, allowing the robot to resume its movement in the environment.

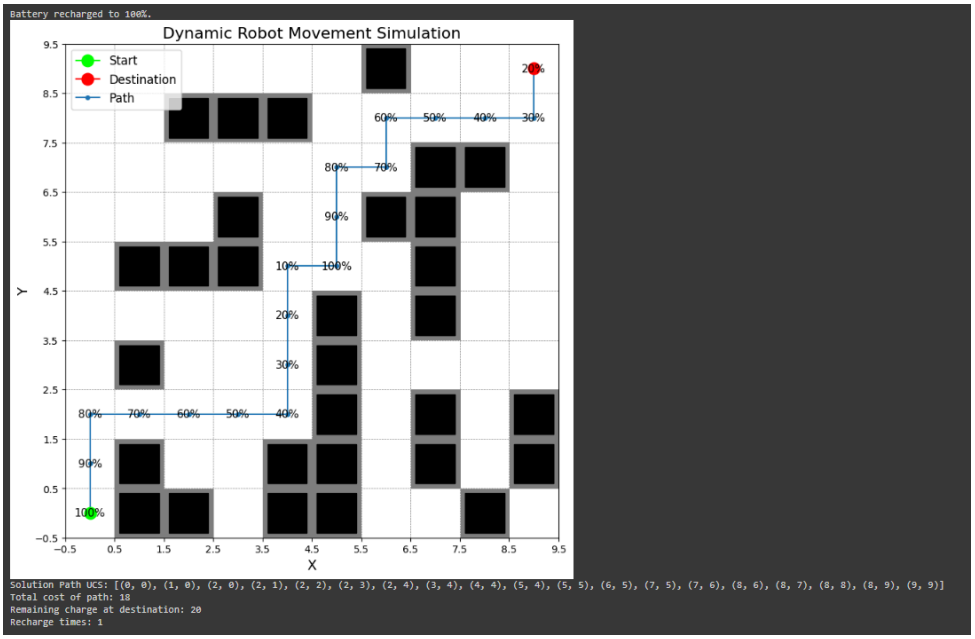
- **Recharge Times:** The system keeps track of the number of times the robot needs to recharge its battery during the simulation. This metric provides insights into the efficiency of the robot's energy management strategy.

5. **Simulation:** It allows users to run simulations using different search algorithms (A* or UCS) and visualizes the robot's movement in the environment. It provides feedback on the solution path, total cost of the path, and remaining battery level at the destination.

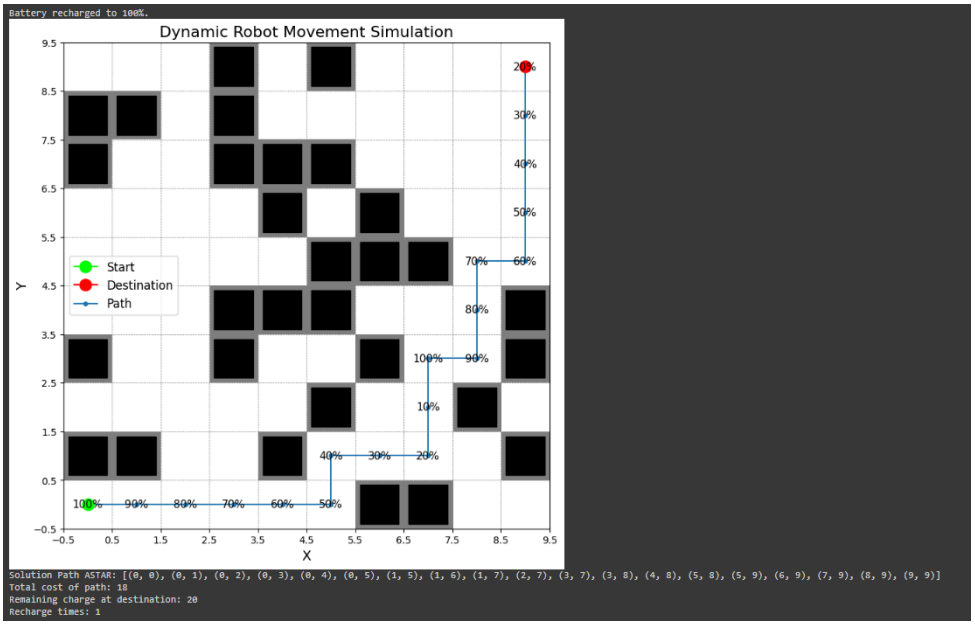
Simulation Results

Two simulations were run using both A* and UCS algorithms:

➔ **UCS Search Run:** The simulation resulted in finding a path from the starting point to the destination using the UCS algorithm. The full algorithm path, recharge times, remaining charge when reached destination for the robot during the simulation were also printed.



➔ **A* Search Run:** Similarly, the simulation using the A* algorithm found a path from the starting point to the destination. The full algorithm path, recharge times, remaining charge when reached destination for the robot during the simulation were also printed.



Conclusion

The Dynamic Robot Movement Simulation project provides a framework for simulating robotic movement in dynamic environments using various search algorithms. By incorporating functionalities such as environment generation, robot movement, pathfinding, obstacle avoidance, and simulation, the code serves as a comprehensive tool for exploring different scenarios and algorithms (UCS and A*) in robotics and artificial intelligence research. Researchers, engineers, and enthusiasts can use this to study robotic navigation strategies, optimize performance, and address real-world challenges in autonomous robotics.