Robot Motion Planning Capstone Project

Plot and Navigate a Virtual Maze

Capstone Proposal

James Tooles
January 9, 2018

Domain Background

This project's core outline[1] was created by Udacity and the idea for this project is mainly derived from Micromouse[2] competitions. In the competitions, a robot mouse agent is given the task of planning and plotting the best path from a corner of the environment (a maze) to its center. The purpose of the competition was for the agent to discover the environment during the first run and to use the best planned path to reach the center of the environment in subsequent runs. The goal of this project is to emulate these constraints of the completion and achieve the goal of the agent obtaining the fastest times possible in a series of test environments. The competition places the same restrictions on the agent as this project's restrictions, which will be covered in the following sections.

Problem Statement

This project is tasked with solving the problem of a robot mouse agent plotting a path from the bottom left corner of a maze to its center. The agent is allowed to embark on two runs of any given maze. The first run of the maze, the agent will explore, map, and analyze the environment to determine the best path plans to reach the center of the maze. The second run of the same maze, the agent will attempt to navigate the environment as quickly and efficiently as possible in order to reach the center of the maze.

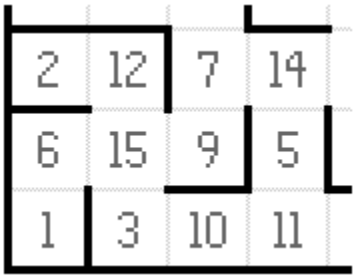
Datasets and Inputs

The dataset and inputs for this project consists of the environment, the maze. The maze is an environment of an n x n grid of squares, where n is even. The environment can have an n value range of twelve to sixteen. The state space for this problem is equal to n * n * 4, where n is the size of the environment and 4 is the number of actions the agent can take in any given location within the environment. The state space is between, 576 and 1024, for environment sizes of 12 x 12 and 16 x 16 respectively.

The environments are accessed through text files. The first line of the text file is a number which describes the number of n squares within the environment. The first data row in the text file is the leftmost column of the environment, with the first element being the bottom-left corner of the environment. The remaining n lines will be n comma-delimited numbers which describe the passible (open) edges of the environment. Each number represents a four-bit number that has a bit value of 0 if an edge is impassible (closed) and 1 if an edge is passible (open). The 2^0 is the upward side, 2^1 is the right side, 2^2 is the bottom side, 2^3 is the left side.

Ex. 1010 ⇔ $(2^3 + 0 + 2^1 + 0)$ ⇔ $(8 + 0 + 2 + 0)$ ⇔ 10

{Left: Open, Bottom: Closed, Right: Open, Upward: Closed}



The environment will always have impassible walls on the outer board of the grid, which blocks all movement. Walls will be placed throughout the environment to block movement of the agent, creating a maze. The robot agent will always start in the bottom-left corner square of the environment grid, and be oriented upwards. The starting square will always have walls on the left, right, and bottom, which will allow for the first action to always be upwards. The environment will have a center within the environment grid consisting of a 2 x 2 square, this is the goal for the agent to position itself within the center 2 x 2 square.

Solution Statement

The information of the environment is initially only given to the agent as the dimensions of the environment, this makes the environment partially observable. Finding the correct path can be done during the first run for the agent by exploring the environment, partially until the agent reaches the goal or fully to guarantee an optimal solution. The path planning algorithm that will be implemented as a solution. Approaches such as, Decision Tree[3], Neural Network[4], Q Learning[5], or A*[6] algorithms will be explored as possible solutions for path planning. The environment will be analyzed during the first run, and the agent will then utilize the path plan created with the least amount of actions to reach the goal.

Benchmark Model

The Benchmark Model for the agent directly relates to how the evaluation metrics are done for scoring the agent's path planning regarding steps taken within the environment. The agent is restricted to a

total of one thousand time steps total for both runs within the environment. One baseline model would be that of taking five hundred actions for each run, maximizing the allowed one thousand time steps. These actions could be random actions to create a true baseline model, which would be random guessing path planning. If the solution for the agent does better than this baseline model, it would be a successful model. The following score would result for the baseline model:

$$Score = \#actions_2 + (1/13) * \#actions_1$$

$$Score_{Baseline} = 500 + (1/13) * 500$$

$$Score_{Baseline} = 538.46$$

Evaluation Metrics

The agent will explore multiple environments and must complete two runs on each. During the first run of the environment, the agent will be allowed to freely explore in order to build a map of the environment. At someone point during the exploration, the agent must enter the goal area but, the agent is free to continue exploring the environment after entering the goal area.

The second run of the environment, the agent will be returned to the starting position and orientation. The agent's goal is to then navigate to the goal area in fastest time possible, minimizing actions (time steps) taken by the agent.

The agent's score is the number of time steps required to execute the second run, plus one thirtieth the number of actions taken during the first run. The maximum actions allowed for a completion of both runs is one thousand actions for both runs for a single environment.

$$Score = \#actions_2 + (1/13) * \#actions_1$$

where lower Score is better

Project Design

The agent is given the problem of solving a randomized maze environment in the least amount of actions. The agent is given access to the environment dimensions initially, then allowed to explore the environment in the first run. The second run, the agent should utilize the optimal path in order to take the minimum actions necessary to reach the goal. Thus, receiving the best (lowest) score possible during the second run. The following theoretical high level workflow will be taken:

1. Dimension of environment will be taken in on the initialization of the agent
2. The goal location area will be derived from the environment dimension
3. First run of the environment:

    a. The agent will explore the environment
        i. This could be done with random actions, actions that prefer to move towards the middle until encountering a wall, or another means of selecting actions
    b. The agent will record the environment during exploration, i.e. where there are walls
    c. The agent will not attempt to run into walls discovered
    d. The agent will reach the goal
        i. The agent can stop exploration
        ii. The agent can continue exploration until the environment is completely discovered
4. The path planning algorithm will take in the recorded environment data and decide the optimal path for the agent
    a. Decision Tree[3]
    b. Neural Network[4]
    c. Q Learning[5]
    d. A*[6]
5. Second run of the environment:
    a. The agent will follow the path planned by the algorithm
    b. The agent will stop once the goal is reached

References

1. https://docs.google.com/document/d/1ZFCH6jS3A5At7_v5IUM5OpAXJYiutFuSIjTzV_E-vdE/pub
2. https://en.wikipedia.org/wiki/Micromouse
3. https://en.wikipedia.org/wiki/Decision_tree_learning
4. https://en.wikipedia.org/wiki/Artificial_neural_network
5. https://en.wikipedia.org/wiki/Q-learning
6. https://en.wikipedia.org/wiki/A*_search_algorithm