Aditya Karlekar

Udacity Machine Learning Nanodegree Capstone Proposal

April 2018

Capstone Proposal – Robot Motion Planning

Domain Background:

The project is inspired by the Mircomouse competitions. The goal is to make the mouse find an optimal path from one corner of the maze to its center. The mouse is allotted two attempts to navigate the n x n maze. In the first run, the mouse surveys the maze to find the center and find the best path to reach the center. In the second run, the mouse tries to reach the center in the shortest time possible by following the most optimal path calculated with information from the first run. In this project, the mouse will navigate through a virtual maze.

Problem Statement:

The goal is to reach the center of the maze in the shortest time possible. The robot will start from the bottom left corner of the n x n maze. Two tires will be provided to accomplish this task. The first run to explore the maze and second run to reach the center as soon as possible. The maze will be of size n x n where n could be of the value of 12, 14 or 16. The center will be of 2 x 2. The maze is enclosed by walls so that the mouse cannot move out of the maze. Also, the first step will be forward as the starting square will have a wall on right, left and bottom side.

Performance Evaluation:

Each trial has an upper limit of 1000 steps. The mouse can move 3 steps forward or backward in a single movement and can turn 90 degrees clockwise or counter-clockwise.

The performance will be evaluated as the sum of one-thirtieth of steps in first run and number of steps in the second run.

Datasets and Inputs:

The maze structure is provided via a text file. The first line of the text files describes the number of squares in each dimension of the maze. The following n lines describe the structure of the maze as comma delimited values describing the open and closed squares. For the mouse to move, it should have knowledge of the environment: its exact position and possible available moves. The location of the mouse is defined by the pair of values like [1, 5].

Solution Statement:

The goal is to reach the center of the maze in the least time possible. This means we have to find the path that requires the least amount of steps from start to end position. The solution consists of two trails. In the first trial, the mouse explores the maze to get detailed knowledge of its layout. In the second run, the mouse follows the optimal path calculated on the basis of information gathered during the first run. The solution is easily quantifiable as we are keeping track of the number of steps as well as the time is taken by the mouse to navigate the maze. The results are replicable as the mouse could be subjected to additional runs to check whether the path followed by the mouse is indeed the optimal path. If the path is optimal, further runs would not affect the results.

Benchmark Model:

The benchmark will be defined as the evaluation criterion defined in Performance Evaluation section: **score = [Number of steps in trail 1 / 30] + [Number of steps in trail 2]**. The maximum number of steps for completing both runs has an upper limit of 1000. The mouse takes a random movement in each step of the first run. The mouse fails the test if it exhausts the 1000 step limit during the first run. If the mouse completes the first run under 1000 steps, then only it is allowed for the second run. As the mouse has optimized itself for the second run, the number of steps in the second run should be smaller than that in the first run. Also, irrespective of knowledge of maze learned during the first run, each maze has an optimal path i.e. the shortest path to move from one point in the maze to another.

Evaluation Metrics:

As discussed, the main evaluation criterion is minimizing the score value defined as:

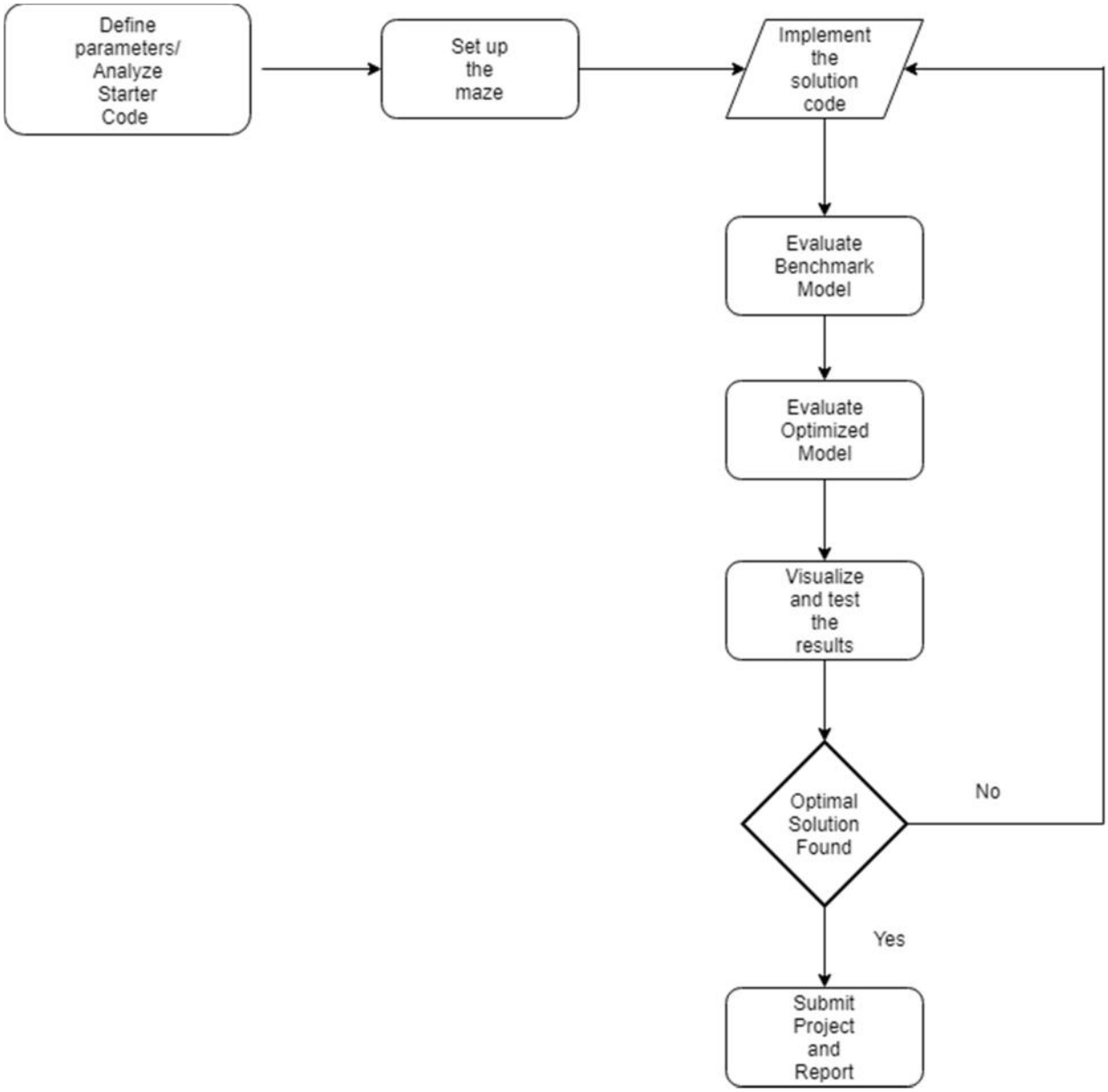
Score = [Number of steps in trail 1 / 30] + [Number of steps in trail 2]

The score is impacted by both first (exploration) and second (optimal) run value. However, in the formula, greater emphasis is given on optimal run value, hence, it will have more impact on the score. The goal is to minimize the score value.

For example, if the mouse completes the first run in 600 steps and second run in 55 steps, then the score will be **600/30** (first run) + **55** (second run). The total score would be 75.

Project Design:

I'll start with defining all the required parameters and reviewing the provided files and starter code. The starter files include robot.py, maze.py, tester.py, showmaze.py, test_maze_##.txt. I'll first run the maze.py file to set up the maze as the contents of the maze are defined in this file. Then I'll modify the robot.py file to implement my solution. I'll run the benchmark model as well as the optimized model and record the results. I'll test the results using the tester.py and visualize the results. I'll keep modifying the robot.py till the optimum solution is reached. Finally, I'll document my project in the report.



Following algorithms can be used as part of the solution:

Wall Follower:

The wall follower makes the agent to take either only the right or left turn, eventually reaching the goal state. The agent can get caught in a circular loop.

A*:

A* algorithm works by creating a tree of all paths from starting point to end point and then chooses the most optimal path from them. It's possibly the best algorithm to find the shortest path.

Dijkstra Algorithm:

Dijkstra's algorithm is used to find shortest path from one node to all nodes in the graph. In maze problem, we can use this algorithm as the algorithm will find shortest path from the starting stage to the goal stage.

Depth-first Search:

The DFS is used for searching tree-like data structure. The algorithm starts from the root node and continues searching that particular branch till either the solution is found or end of the branch is reached before traversing different branch.

Breadth-first Search:

The BFS is also used for searching tree-like data structure. The algorithm starts with the root node and then moves to the node at the same level of the tree (closest neighbor) before moving to next level.

Resources:

Udacity Project Files:

https://www.google.com/url?q=https://drive.google.com/open?id%3D0B9Yf01UaIbUgQ2tjRHhKZGlHSzQ&sa=D&ust=1523961264287000

Micromouse Wikipedia Page:

https://www.google.com/url?q=https://en.wikipedia.org/wiki/Micromouse&sa=D&ust=1523961264283000

Apec 2014 Micromouse Texas:

https://www.youtube.com/watch?v=0JCsRpcrk3s

Dijkstra's Algorithm:

https://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/

A* algorithm:

https://www.geeksforgeeks.org/a-search-algorithm/

Depth-first Search:

https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

Breadth-first Search:

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/