CSE 4618: Artificial Intelligence Lab

Lab 2

Name: K.M.Tahlil Mahfuz Faruk

ID:200042158

Islamic University of Technology, OIC

Gazipur, Bangladesh

Feb 27, 2024

Introduction

A* Search:

The problem asks us to implement the A* search algorithm.

Finding all the corners:

In this problem, the pacman has to find the four corners of a maze and eat the food pallets considering the walls in of the maze.

Corners Problem Heuristics:

In this problem we have to define a non-trivial heuristic for the cornersProblem.

Eat all the dots:

In this problem we have to eat all the dots in the maze exploring as minimum nodes as possible.

Suboptimal Search:

In this problem we have to implement the findPathToClosestDot function.

Analysis of the problem

A* Search:

In this problem a heuristic value and the problem is sent as parameters. The function tries to find the optimal path from the start state to the goal state using the A* search algorithm. It takes the cumulative value of heuristic value and the actual cost under consideration to find the optimal path.

Finding all the corners:

In the CornersProblem class we need to implement getStartState, isGoalState, getSuccessors.

Corners Problem Heuristics:

The heuristic needs to be admissible and consistent. So we can use the Manhattan distance or the Euclidean distance as our heuristic.

Eat all the dots:

In this problem we have to define the heuristic that will help to eat all the dots. But the maze consists of a lot of blocks between pacman and the foodpalets. We need to consider the max distance of pacman and the food pallets considering the walls also.

Suboptimal Search:

In this problem we have to find the closest dot from pacman. That can be done using any search algorithms such as A* search.

Explanation of the solutions

A* Search:

- The priority queue ensures that the minimum cumulative cost is found at the top of the queue that leads to the goal comparatively quicker than other algorithms.
- The heuristic function guides the search from start to finish. A well designed heuristic can reduce the search space effectively that finds the comparatively more relevant path and gets to the goal more efficiently. An admissible heuristic generally leads to faster convergence. Inadmissible ones might find a solution but may take longer or provide suboptimal paths solutions.
- Tracking of visited nodes helps to not re-explore an already explored path that reduces the time complexity of the algorithm.

Finding all the corners:

- To keep track of the visited corners we initialize a dictionary of visited corners. Key would be the dimension of the corner and value would be a boolean that signifies the corner is visited or not.
- In getStartingState we return the startState and the visitedCorner tuple.
- In the isGoalState we check if all the visitedCorners value is true.
- In the getSuccessors, we update the position of pacman. But we need to consider the walls. So we check if the updated pacman dimension is hitwall dimension or not.
- We also need to check if the nextPosition is a corner or not.

Corners Problem Heuristics:

- In this solution we have used the maximum manhattan distance as our heuristic.
- Manhattan Distance= abs(x1-y1)+abs(x2-y2); where (y1,y2) is the dimension of the corners.

Eat all the dots:

- We have the food pallet co-ordinates.
- Also there is a function called mazeDistance that returns the mazeDistance of one point to another point considering the walls as well.
- We use that function to find the distance of pacman and food pallets and get the maximum value that will take us closer to the actual summation of pacman to all the food palettes.

Suboptimal Search:

We have our already implemented A* search function. In the findClosestPathToDot function we just need to call the aStartSearch algorithm to find the path.

Challenges and Solutions

A* Search:

During this problem no significant problem was encountered.

Finding all the corners:

During this problem no significant problem was encountered.

Corners Problem Heuristics:

In this problem, when I was trying to run it could not pass all the tests when I was using any combination of max,min of euclidean heuristic or min of Manhattan Heuristic. Only when I took the max value of manhattan distance from the corner to pacman all the test cases passed.

Eat all the dots:

Manhattan Distance was not working in this case. But the mazeDistance function works as it considers the walls as well.

Suboptimal Search:

During this problem no significant problem was encountered.

Behavior for Different Hyperparameters

A* Search:

- The behavior of the code varies based on the problem and the heuristic function.
- A well designed admissible heuristic can significantly improve the efficiency of the search.
- An inadmissible heuristic may converge but might provide the optimal solution.

Finding all the corners:

- Depending on the problem passed as a parameter the solution will vary.

- Walls, grids,starting position of pacman, position of corners will vary the behavior of the code.

Corners Problem Heuristics:

Similar to the previous problem.

Eat all the dots:

Similar to the previous problem.

Suboptimal Search:

Similar to the previous problem.

Interesting Findings

A* Search:

It combines the benefits of UCS and the Greedy Search that helps it to find the optimal solution of a given problem depending on the heuristic function.

Finding all the corners:

No interesting findings for this problem.

Corners Problem Heuristics:

In this problem, only when we use max of manhattan distance all the test cases passed. Why? Basically the max value will take us to the actual cost the most. The summation of cost from pacman to all four corners will be our target to achieve. But most of that cost can be achieved only if we take the maximum value.

Eat all the dots:

MazeDistance function was a fortunate finding for me.

Suboptimal Search:

No interesting finding for this problem.