CSCI 630 Foundation of Intelligent Systems
Project-1 Report

Author: Sandhya Murali, Rohan Shiroor

Problem Definition:

The project demonstrates finding the shortest path in a maze from start to goal using A* search technique. The A* search algorithm uses both node cost and heuristic in order to determine the best move. Therefore different heuristic functions are applied in order to determine the shortest path from start to goal.

The state space consists of an NxM state puzzle, where N is the number of rows of the maze and M is the number of columns. The puzzle consists of '.' which signify an empty location, ' * ' signify an obstacle, S represents start state , G represents goal state. Now there are O obstacles in the puzzle. So the number of states the die can be in is (NxM)-O. The die has 6 faces which are North, South East, West,Top and Bottom. One can move in 4 directions namely Right, Left, Up and Down. The die has a constraint of never having 6 on the top and not rolling through blocks. It must also have one on the top when goal state is reached. Since 6 can never be on top and there are 4 directions where the die can move in the maze, there are 5x4 = 20 possible orientations for each move. If we multiply with the permissible states of the die we get (20x(NxM)-O) states.

In our project, we have decided to establish upper left corner of the grid as (0,0). Use this as a reference point for the die.

Initial State:

Die with 1 on top, 2 facing north, 5 facing south, 3 facing east and 4 facing west at the start of the maze (marked with S). There is also a maze with size MXN. Therefore we have a class of Die that stores all the 6 orientations (East, West, North, South,Top and Bottom). We have a Node class that stores the location (row,column), Current Dice Orientation,Parent Node,Parent Coordinates, Dice Top and the heuristic values (f,g,h) which will help us select the optimum path. We have also implemented a Frontier Queue that stores this Node representation in a sorted order based on the f value of the node for the current dice representation since this f value will help us determine our next optimum node to reach our goal.

Therefore each cell in a maze with a configuration of the die is considered as a state for the cell.

Transitions:

The die can be rolled in North,South,East and West directions. Additionally the move can be only in an empty space of maze and not involve moves to cell which has obstacles. The die cannot be rolled if rolling it will place 6 on the top and also should have one on the top when destination is reached.

Therefore, to move in each of the maze, we need to roll the die. The die will be tipped and rolled back if the given constraints (1 on top when destination is reached and 6 should never be on the top) has not been met. This moves will be continued until goal state with die configuration of 1 on the top has been met.

Path Cost:

Total cost to reach the goal of the maze (G) considering the constraints of the die that the face 6 should never be on top and goal state dice configuration constraint that 1 should be on top of the die.

Goal Test:

The goal state will be met if the current cell in the maze is denoted by 'G' and satisfy goal state dice configuration constraint that 1 should be on top of the die and any other configuration for the remaining. At this point the heuristic will be zero. For each of the Heuristics, they may have different behaviour between start and the goal state.

Heuristic Functions:

The characteristics for Heuristic function is that it should be admissible and consistent. Admissibility determines the optimality in the code which means it should not overestimate the distance to the goal. Consistency is that the total path cost should be less than or equal to cost from source to intermediate node and from intermediate node to goal.

The Heuristics chosen for the project are:

Manhattan Distance

It finds the absolute value of difference between the x and y coordinates of the goal state and the x and y coordinates of the Goal state. That is, it counts the number of vertical and horizontal blocks to reach the goal. This heuristic assumes that there are no obstacles between the die and that the dice can move only in 4 directions(left,right,up and down).

The equation for the Manhattan Distance is given by

$$h(n) = abs(current\_cell.x - goal\_cell.x) + abs(current\_cell.y - goal\_cell.y)$$

This heuristic is admissible because it finds the number of shortest empty spaces to reach the goal node from current state. It cannot overestimate because the path through the empty space cannot be any shorter. That is, from a node n1 to goal, the value at best would be equal to the original Manhattan distance. Hence the cost from n1 to goal (cost to reach n1 plus Manhattan Distance from n1 to goal) will be at best equal to the original Manhattan distance (from n1 to goal). Therefore we can say that it is consistent.

Euclidean Distance

It finds the straight line distance between the current node and the goal node. The equation for the Euclidean distance is given by the formula

$$h(n) = \sqrt{(current\_cell.x - goal\_cell.x)^2 + (current\_cell.y - goal\_cell.y)^2}$$

Euclidean distance is admissible heuristic because it calculates the straight line distance between the current and the goal node. The straight line distance can be the shortest distance between any two points in the grid. If the die could move diagonally, it could make the estimated cost higher than actual cost. Since the die cannot move diagonally, this Heuristic is considered admissible. This heuristic is consistent and does not overestimate there will be no distance that is shorter. That is if there is a shortest path between two points, adding another node will cause the sum (by adding from start to node and from node to end) greater than the original. Hence this heuristic can never overestimate. The Euclidean distance assumes that there are no obstacles between the current and goal state, the die moves in a straight line and the die just slides instead of rotating.

Diagonal Distance

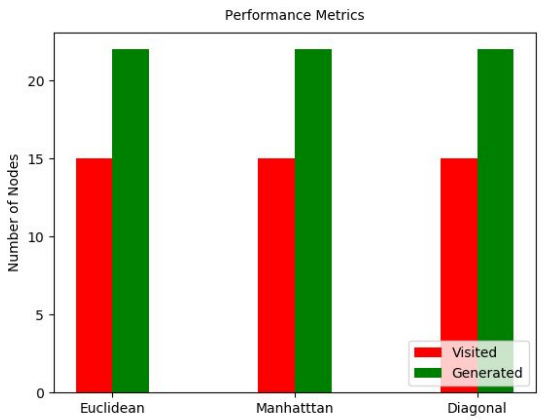
It finds the maximum of absolute values of differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively. That is, it counts the number of vertical,horizontal and diagonal blocks to reach the goal. This heuristic assumes that there are no obstacles between the die and roll and that the dice can move in all 8 directions.The formula is given by:

$$h(n) = max(\ abs(current\_cell.x - goal\_cell.x)\ , abs(current\_cell.y - goal\_cell.y))$$

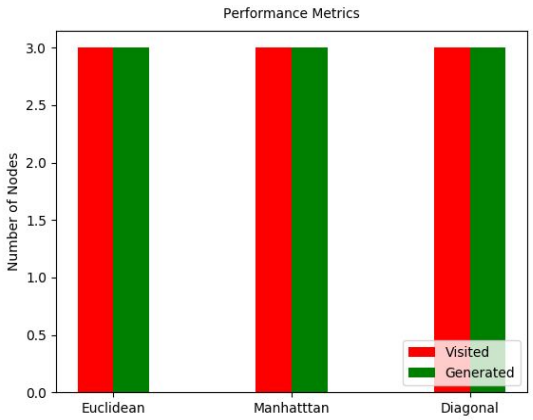
The diagonal distance is considered as admissible because the heuristic value generated after computing h(n) is lesser than Euclidean and Manhattan distance which is considered admissible. Since Euclidean computes shortest path between two nodes and Manhattan counts the number of horizontal and vertical blocks, it is considered admissible. Since Diagonal distance is a combination of Manhattan and Euclidean, we can say that Diagonal distance is admissible. Hence, Diagonal distance considers the direction with maximum distance and penalizes the direction with minimum value.Therefore it will never overestimate the cost since this cost will always be lesser than or equal to actual cost. Diagonal Distance assumes that there are no obstacles between current and goal state and die moves in a straight line.

Performance Metrics

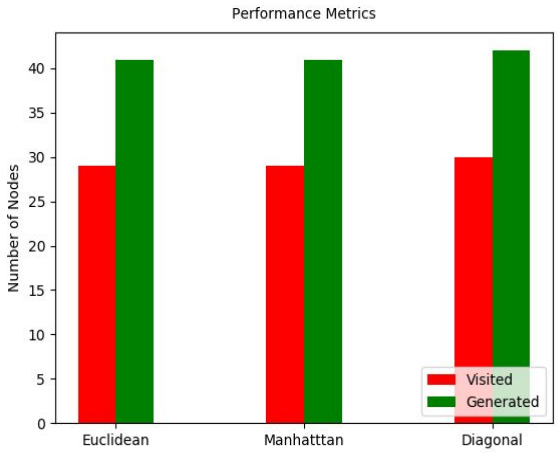
For each cell in the maze, the total cost is computed using the heuristic function and actual cost. The node with the lowest f value where f(n)=g(n)+h(n) is popped from the frontier queue and marked as visited. The neighbours of the visited nodes are explored and if the configuration is not visited, we roll the die in that direction, and we push the node in the frontier queue if that configuration is not present. Below is the performance metrics provided for puzzles 1,2,3,4 and 5.



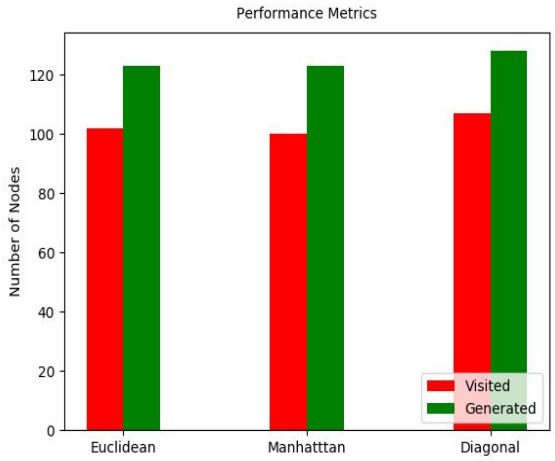
Puzzle 1



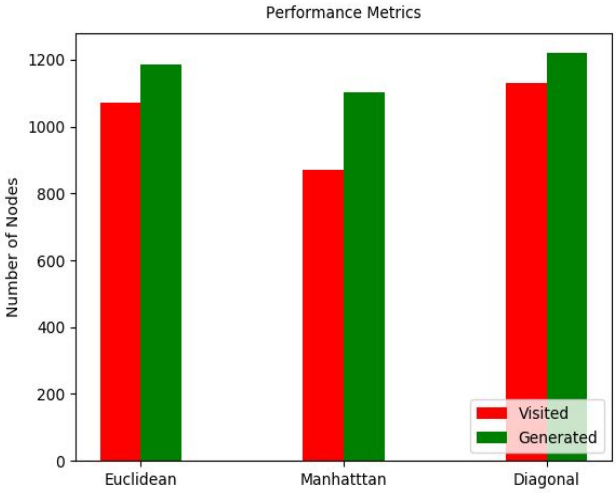
Puzzle 2



Puzzle 3



Puzzle 4



Puzzle 5

	Euclidean Distance			Manhattan Distance			Diagonal Distance		
Puzzle	Solution	Visited	Generated	Solution	Visited	Generated	Solution	Visited	Generated
1	7	15	22	7	15	22	7	15	22
2	-1	3	3	-1	3	3	-1	3	3
3	13	29	41	13	29	41	13	30	42
4	22	102	123	22	100	123	22	107	128
5	27	1072	1187	27	870	1103	27	1130	1220

From the above results, we can see that Manhattan and Euclidean give better performance as compared to Diagonal Distance. For smaller puzzles all 3 give similar results but however as size of the puzzle increases, Manhattan and Euclidean give better performance than Diagonal distance. The reason the diagonal distance does not give optimum performance is because for diagonal distance, we can consider moving in all 8 directions to consider optimum path and in our case, we restrict the die to move only in 4 directions. Therefore, it does not consider the diagonal distance which could be shorter than the horizontal or vertical distance. Due to this it tends to explore more nodes than needed in order to generate an optimal path.

Discussion

One of the main challenge we faced was to understand the representation of the cells in the maze and understand the movement of the die based on the constraints.
The solution to this was to create a class for the die, for the cells of the maze and store the representation of the die along with f,g,h values, along with the parent cell.
Given Below is a pseudocode of our project:

```
Function A*search(start_node,goal_node):
push(start_node)

While queue:
Get the node with lowest f value
Check if node is goal with die top 1:
        Generate path if true
Otherwise:
        Access its neighbours (consider only empty spaces as neighbours)
        Roll the die in that direction
        If valid die roll(no 6 on top) and node with that dice configuration is not visited:
                Calculate f,g,h value
        If node with that dice configuration not present in frontier queue
                push(node)
        Else
                Check the existing cost with the new cost
                If new cost is less than existing cost:
                        Update value in frontier queue
```

From the table generated, we can see that Euclidean distance computes the shortest path from start to goal and this cost is always less than the actual cost of the path. Manhattan distance computes close to the actual cost. Our third Heuristic is Diagonal Distance which is also considered admissible because the value obtained is always lesser than Euclidean and Manhattan which is also admissible. Also, the diagonal distance considers the direction with maximum direction and penalizes the direction with minimum distance. However it does not give effective performance as compared to Euclidean and Manhattan distance because in case of diagonal distance we can move in all 8 directions. However, since we restrict our directions to 4, it does not consider diagonal distance which could be shorter as compared to horizontal and vertical distance. Due to this, more nodes are explored and considered to reach the optimal path. For A* search it should have a heuristic that is admissible and consistent. Otherwise it will never compute the optimal path to reach the goal. If we design a heuristic that generates a path equal to the actual cost, then A* search will follow only one path since there will be only one optimum path to reach the goal. Therefore if we get a heuristic cost which is lesser than actual cost, then a smaller path may be obtained.

Over the course of the project, we can see that as the size of the puzzle increases, the puzzle complexity increases. Due to this the search tends to explore more nodes than needed. Hence the algorithm will do a poor job of estimating the actual cost when complexity and size of the puzzle increases. Therefore simpler puzzle show similar results but as complexity of the puzzle increases, we can see performance differences.

In case of puzzle 2, all the heuristics had same number of states visited and same number of states generated. This may be because, since there is no path to reach the goal state by keeping in mind the constraints, it considers all possible nodes to reach the goal.