AI Project 1 : Astar algorithm and relaxation technique

Leena Dighole

Strategy 3: Relaxation based strategy

3.1 Aim: We tried to develop a heuristics based upon a relaxation strategy and tried to observe the working of the Astar algorithm in higher dimensions.

We named it as "Cubes inside a cube!!"

3.2 Motivation:

We generated the Astar algorithm with geometric heuristics (Manhatten distance and Euclidean distance calculations) and observed that we are surprisingly able to solve the mazes up to dimensions 2000*2000 successfully!

Also we observed the relaxation technique to determine the heuristics for A star algorithm using p-thinning. We randomly removed some of the blockages and tried to observe the performance of the method. Surprisingly as the number of blockages were small, the technique worked very well in giving successfully solved results for higher dimensions (dim = 1000, p =0.3, rho = 0.4)

Now as this relaxation strategy worked well in the problem of solving a maze using Astar technique, we tried to think about another possible relaxation strategy for even better results.

The relaxation technique allows us to put some more constraint on the present situation in order to save a number of search space nodes.

With the similar motivation we tried to work for the **"cubes inside a cube"** strategy

3.3Algorithmic Thinking:

1. Create original Maze with a given dimension(dim) and blockage probability (p)
2. Convert the original dim dimension maze into a sufficiently smaller dimensions but make sure to retain its original blockage properties
3. Design a method to retain the newly formed superblocks original blockage properties
4. Find out the optimal path in the relaxed version of the maze
5. Implement the Astar in original maze with the nodes of the optimal path of its relaxed version
6. Observe the performance accuracy

Original maze (dim = 200 , p =0.3)

```
Enter dimension of the maze (dim): 30
Enter obstacle probability (p): 0.2
Enter heuristic (Manhattan (m) / Euclidean (e)): m
The Dimensions of Original maze : 30
Lets check the relaxation heuristic with new dimension : 3
```

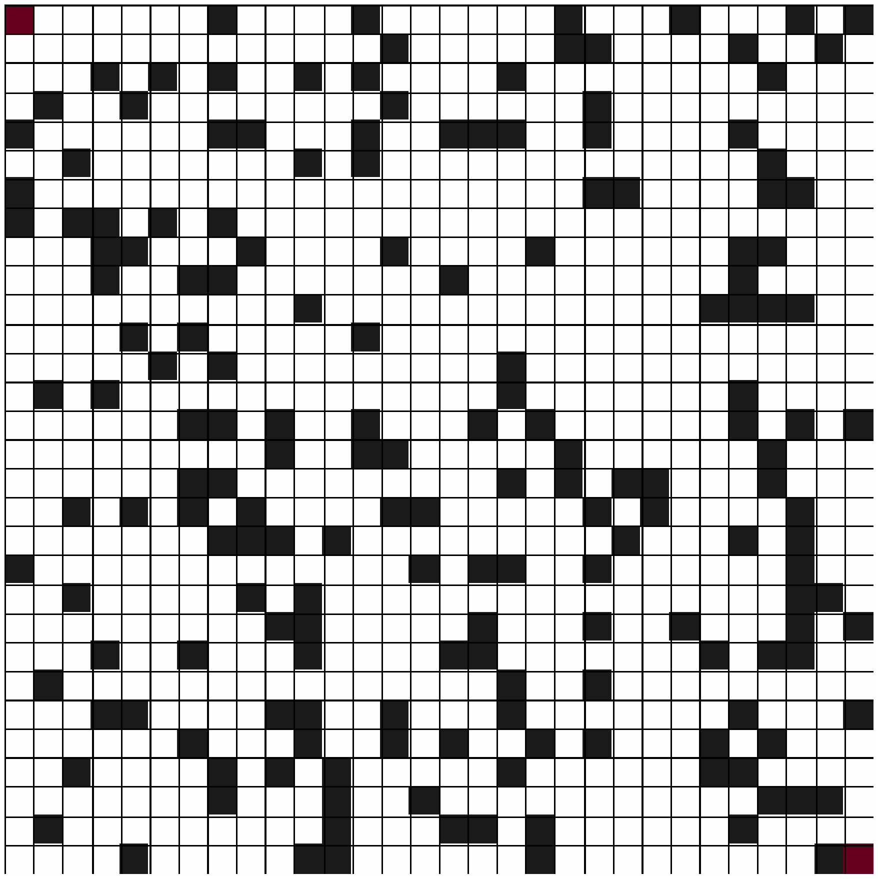
Figure 1:Original Maze with dim = 30

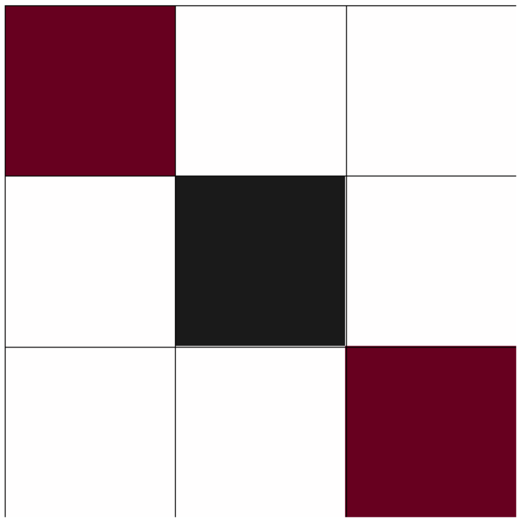
Figure 2: Relaxed version of a maze with new dim =3

3.4 Retaining Blockages Properties:

We tried implementing 2 methods:

1. Blockage Probability as a weight
2. Blocked Neighbor Weight

3.4.1 Blockage Probability as a weight

The motivation behind this method is simple. We calculated the probability of the blocked cells Inside newly formed superblock and tried to assign weightage about blockages accordingly.

Algorithm: Method 1

Aim : Try to assign the blockage cost to new superblocks

To Do:

1. Calculate number blocked and open nodes inside a superblock

2. Calculate its blocked probability "p_superblock = (block cells/Total cells)"

3. Set a "Fixed Threshold value" using p_superblocks

4.Assign weightage cost to all superblocks

5.Display Newly weightage relaxed version of the original maze!

6.Calculate number of "saved nodes" in the relaxed version of the maze

Result: Poor results as dim increases p_superblocks "converges" to original p and fails to assign proper costs to new superblocks due to small variance.

If p_original = 0.2 then p_superblocks also converges to 0.2

Thus we can’t perfectly assign the weightage to the blocks and thus using this method proves no additional cost savings in the search algoritm.

Whats next option then?

→Think about Another method

3.4.2 Blockage Probability as a weight

The motivation behind this method is simple. We calculated the Calculate weightage of the blocked node according to its immediate blocked neighbors. We are trying to find out the

Algorithm: Method 2

Aim : Try to assign the blockage cost to new superblocks

To Do:

1. Calculate number of immediate blocked nodes of the particular blocked cell
2. Calculate its mode value and maximum value
3. Set a "Dynamic Threshold value" using mode value
4. Assign weightage cost to all superblocks
5. Display Newly weightage relaxed version of the original maze!
6. Calculate number of "saved nodes" in the relaxed version of the maze

Result: Better results for higher values of dimensions. The nodes with some blockage patterns will be given higher weightage in terms of saving some search nodes.

As here we are not considering the any threshold value based upon the input (p) value.

We can implement this algorithm in higher dimensions where in method 1 p_blockage was converging to original value of p. But this method is now independent of that convergent factor and hence gives better results compared to the method 1

3.5 Observations:

Threshold code logic:

```
#Dynamic Threshold value
        if self.block_node_max >= 3:        #Dense maze
          new_threshold = 2
        elif self.block_node_max < 3:    #less dense maze
          new_threshold = 1
```

Case 1: dim = 100, p =0.1

Comparing number of nodes traversed:

1. Astar in original maze = 239   :)

2. Astar in relaxed maze = 217 (savings!!)   :)

Analysis: Startegy is viable with given threshold logic

Case 2: dim = 100, p =0.3

1. The Astar in original maze has optimal path (253 nodes) :)

2. The Astar in relaxation maze (superblock version) has a optimal path :)

3. Astar in the reduced version of original maze using information from relaxed version optimal path (refers 2.) the maze is unsolvable! :(

Anaysis: The same threshold logic in case 1 gives the inference that as original maze has optimal path so does the relaxed version. But restricting search space only in the optimal path may not give us the required solution for solving the original maze.

Possibilities for failure:

1. Threshold logic should be changed and see the performance (But how to decide optimal threshold satisfying all scenarios?)

2. Too much restrictions may not give us the optimal path with p >= 0.3

Case 3: dim = 100, p = 0.2

1. The Astar in original maze = 229 :)

2. The Atsar in relaxed maze = 269   :)

3. 40 extra nodes searching takes place than original form of maze   :(

Analysis: The relaxation startegy is correctly predicting the solvability of the maze but due to restricting the search space the total no of nodes traversed in optimal path is more than that of original path

Case 4: dim =100, p = 0.4

1. The original maze is unsolvable   :)

2. The relaxed version of maze is also unsolvable :)

Analysis: As the original form of the maze is unsolvable and so does its superblock version is reflecting that the maze is unsolvable. Thus the relaxation technique is predicting the original maze problem upto desired results.

This shows that the relaxation technique is viable strategy as its mininmizing our search space and giving optimal solution.

Case 5: dim =1000, p =0.1

1. The original maze is solvable = 2167 nodes   :)

2. The relaxed version is sometimes not solvable   :(

Analysis:For higher dimension the threshold value affects the results drastically. We should think about coming up with the good strategy to retain the original blockage properties of the maze(dynamic threshold value)

Note: Kindly refer the attached html reports for these 4 sample scenarios to view the actual images of Astar implementation in these cases!

3.6 Some important Observations:

1. The relaxed version of the maze gives us the optimal solving route for the nodes with less cost as expected.
2. The relaxation based strategy for generating a heuristic is a viable approach as from above example of dim = 100, p =0.1/0.2/0.3/0.4
3. The relaxed version of the maze is solvable and thus so our original maze too!
4. This technique proves helpful in dealing with the mazes with higher dimensions such as dim >=100
5. We tried to solve this strategy for the highest dim = 2000 successfully and with p = 0.2 the maze was not solvable which is as expected.
6. As the p value increases from 0.1 to 0.3, the solvability rate of the maze decreases when repeated for every 10 different mazes with same dim value.
7. As the dim value increases the solvability rate of dim ranging from 100 to 1000 decrease s for the same value of p which seems reasonable observation
8. The mazes with p>= 0.4 are almost resulted into failed to solve the maze as expected.

3.7 Scope of brainstorming:

1. Optimal technique to decide value of the threshold for superblock version of the maze

Note: Two methods for calculating relaxation cost are used

1)Blocked Neighbor Weight

2)Blockage Probability as a weight

2. How to attempt the conditions where following the relaxed version optimal path gives o/p "maze is not solvable" but in its original version its still solvable.(Again in my opinion this will get solved if we have a optimal way of the first scope of brainstorming)

Note: We tried to show the Astar in relaxed version and combined version(relaxed + original) in order to compare its solvability and accuracy of performance!

We tried to observed the relaxation technique implementation depending upon the different methods for threshold decision and tried to compare its potential advantages and disadvantages.

Thus the usage od relaxation technique for solving a maze and generating a potentially good heuristic is a viable idea and can be done with several different ways. We tried to solve using one technique and analyzed its performances and results.