PROJECT 1: DIE-MAZE PROBLEM

Srinath Kanna Dhandapani(sd2689)

Shivankar Ojha(sxo4955)

Problem definition: The basic representation of our problem makes use of a class called, 'node', which in general is the representation of the state of our maze and our dice in one particular configuration. It consists of the position that the dice is currently in, in the form of integer values i and j; a dictionary 'die' having three values with keys top, front and right respectively, with integer values being the corresponding values to these keys representing the number on these faces of the die; a variable called 'previous' which contains a link to the parent; a list called 'successors' to store the successors of this particular node; the 'heuristicCost' and 'nodeCost' variables which are storing the cost to get to the current node and the heuristic cost of this node to get to the goal respectively; and finally, a string called move taken which stores the move taken by the dice to get to this configuration. Our solution basically takes the starting position on the maze and creates a node for that point, generating its successors from there on and looking for the optimal solution using the A* algorithm and the designed heuristics.
For our implementation, the states and state space are as follows:
State: The state, as described above is defined by an instance of the node class, which stores the information of the state of the dice and its position in the maze.
State space: The state space is a set of nodes, generated by the successors function, which generates all possible moves from the starting configuration, and these are the nodes in which we look for our solution.

Heuristics:

The mathematical formulas used in implementing our heuristics are as follows:
Current node position=(i,j)

Goal node position=(a,b)

c=max(abs(a-i), abs(b-j))

d=min(abs(a-i),abs(b-j))

Now, the three Heuristics that we wrote for our implementation are as follows:

(1)<u>The first heuristic:</u> We came up with a unique heuristic based on the area between the current point and the goal point in the maze. The mathematical representation of the heuristic function is as follows:
h(current node)=(c*d)/2
The logic behind this heuristic is that for any two points, we can always be able to traverse the distance between them in a maximum of (area)/2 steps. This is what makes the heuristic admissible.
<u>(2)The second heuristic</u>: The mathematical definition of the second heuristic is as follows: h(current node)=3*d.

We know that c+d= Manhattan distance by definition. But in some cases, this overestimates the cost path of the problem, and gives a suboptimal solution, as was happening in puzzle 5. We know: c>d (since c is the maximum value)

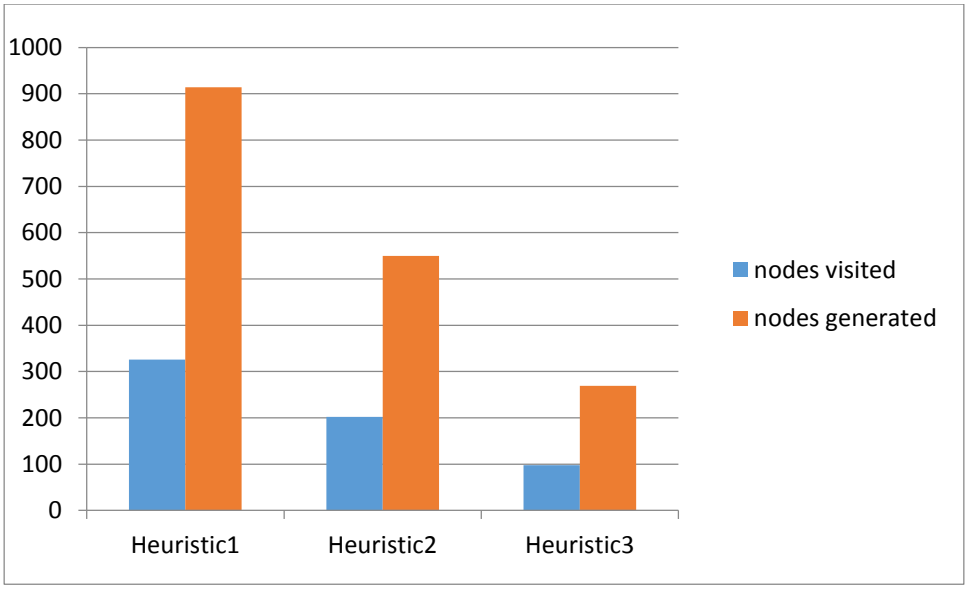
So, c+d>2d. But 2d, we observed underestimated the distance and cause excessive node generation for our heuristics. In order to deal with this drawback, we used 3d as our Heuristic, which was able to optimize our solutions.

(3) <u>The third heuristic:</u> function is defined mathematically as:
h(node)=(d^2)*square root(2)+abs(c-d)
Our logic in coming up with this heuristic was to incorporate the area between the starting and the goal node, as well as the length of the path to be traversed. Now we consider an area for the movement of the die which is customized by taking the product of the d value, with square root two, i.e, d*square root(2); with the value of d, which comes out to)=(d^2)*square root(2). This represents the area we consider; the movement of the die is represented in the heuristic by the absolute value of (c-d).

<u>Performance Metrics:</u>
Shown below is a table depicting the number of nodes generated and the number of nodes visited, for each Heuristic function, corresponding to each of the 5 puzzles.

Heuristic1 Visited	Heuristic1 generated	Heuristic2 visited	Heuristic2 generated	Heuristic3 visited	Heuristic3 generated	
24	52	23	50	16	35	Puzzle1
71	165	89	216	69	164	Puzzle2
5	5	4	4	4	4	Puzzle3
162	371	201	467	103	235	Puzzle4
326	914	202	550	98	269	Puzzle5

From the above table, we have drawn a bar chart for the puzzle 5, depicting the number of nodes generated and visited for each heuristic, which is shown below:



Graph for puzzle 5

Discussion:

1. For the heuristics we defined, the third heuristic turned out to be the most effective of all, not just solving the last puzzle effectively, but also generating just 267 nodes, while traversing only 98 of them.

2. Initially, while generating the nodes for all the puzzles, and working with the Manhattan and Euclidean distance heuristics, we were getting suboptimal solution of 28 moves for puzzle 5. That is why we decided to implement heuristics which we came up with ourselves and we were able to generate the optimal moves of 26 steps for puzzle 5 with them.

3. The second and third heuristics generated 4 nodes before stopping in comparison to the first heuristics which generated 5, for puzzle 3 which had no

solution.
4. The first heuristic also was a step up above the Euclidean and the Manhattan distances, since it made use of the minimal estimation, with respect to the area between the starting and goal nodes, and we observed that the distances measured by this heuristic were always lesser than that of Manhattan and Euclidean distances.

5. In comparison to the Manhattan and the Euclidean distance metrics as Heuristic functions, the three heuristics defined by us gave estimates of distances which were more and more accurate as we got closer to the goal.

6. In all the heuristics, we compute the area that is available for the die to travel and also take it into consideration along with the path so that these heuristics better deal with such problems.

7. The number of nodes generated for puzzle 5 by each of the three heuristics differed drastically, even though the ratio of the nodes visited to the nodes generated came out to be 0.36 for all three of them.

8. The nodes generated drops to half from one heuristic to the next and same for the number of nodes visited by them, again emphasizing on the superiority heuristic 3.

9. Heuristic 1 performs better than heuristic 2 for some puzzles and vice versa. This implies that Heuristic 2 is best suited to puzzles with minimal or very few obstacles.

10. We observe that all the heuristics behave similarly by giving us the same optimal solution if it exists, but their efficiencies are different which is evident by the number of nodes each of them generates and traverses.