Discussion of AI Project 1

Soham Sadhu

January 11, 2012

Problem Definition: rolling die mazes.

1. States: The position of the agent die in the maze and its orientation that is which face of the die faces which side, will define the set of states.

2. Initial state: The initial state of the problem is the agent die which is six faced, in a grid type maze is positioned in a cell which is denoted as the start cell. Also in this start cell, the orientation of the die has to be the start orientation which is side 1 is facing up, side 2 is facing North or upwards and side 3 is facing right hand side or Southwards. Thus for agent die for start or initial state can be represented by two tuple as die(start position, start orientation).

3. Description of possible actions: There are four possible actions that our agent die can take, roll East(E) or right and move to cell at right. Roll West(W) or left and move to cell at left. Roll North(N) or upwards and move to cell above. And roll South(S) or down and move to cell that is below. Provided there is cell present in that direction which is not marked as obstacle(*). For example the following tuple can define all four actions {if cell present above and not * then N, if cell present below and not * then S, if cell present at right and not * then E, if cell present at left and not * then W}.

4. Description of what each action does: Each of the action changes the current position of the die toward the cell in the direction where the action was taken and also changes the orientation of the cell. For example action S will move the die to position just below the current position of die as well as turn the face of die pointing North to up, the face pointing South to down, the face pointing up to South and the face pointing down to North. The faces pointing East and West remain unchanged. Thus this action can be written as die(present cell, orientation, action S) = die(cell just below the previous, new orientation), this form just shows the result state change of the die which includes the change in position of the die for the action as well as the new orientation of the die. Similar action result state changes can be defined for other four actions. A important point here is the validity of the state. If in the state the orientation of die is such that face six is pointing up, then it is considered as invalid state. Thus for a new state generated the orientation of die has to checked for state validity.

5. Goal test: The goal test is checking that the current position of the die is in the cell in maze which is designated as goal and in orientation of the die the side 1 is facing up.

Thus goal state for agent die can be die(goal position cell, orientation where side 1 is facing up).

6. Path cost: From the action set a = (N, S, E, W) the die for all action rolls only once. Thus the path cost for each action is 1 or can be considered one roll of the die that is c(s, a, s') = 1; where c is cost, s and s' are current and next state respectively and a is any one action from the action set.

Heuristics: The following three heuristics were implemented.

1. Straight line or Euclidean distance: the straight line distance between any any two given points. In this case the heuristic measures the distance between the present position of the die and the cell marked goal. This distance is mathematically defined $\sqrt{(x1-x2)^2+(y1-y2)^2}$ where (x1, y1) can be abstracted as position cordinates of goal state in maze and similarly (x2, y2) as position cordinates for die. The straight line gives a indication for the die that how far the die is from the goal, thus being a heuristic. It is admissible because it never overestimates that path cost, at most it equals the real path cost but never exceeds it. The path cost is measured in rolls of a die and the die cannot roll diagonally or has to move in a perpendicular axes in which it is positioned. So if the goal and die are positioned in different rows and columns in the maze then number of rolls or path cost to goal will certainly exceed the straight line which the die cannot take. Also if the die and the goal cell are in straight line like in same row or column the above formula gives the distance which are equal to number of rolls of the die but does not exceed the path cost.

2. Manhattan distance: This distance is the number of the steps from the current position to goal position when allowed freedom movements are in perpendicular direction. This can be defined as the sum of the absolute difference between the x and y cordinates. $|x1-x2|+|y1-y2|$. The order in which you take the cordinates for difference does not matter as long as you are consistent. This heuristic is admissible because it represents the minimum number of rolls that the die will have to do, to reach the goal. Even if the die does not take the straight line row or column of cells, still it will have to move perpendicular. So even trying to reach the goal in straight line it will have to simulate the straight line with alternative perpendicular moves thus eventually covering the Manhattan distance. Thus the heuristic gives path cost equal to actual one in best case but never overestimates.

3. Least number of obstacles in a Manhattan path plus the Manhattan distance: it returns the least number of obstacles on a Manhattan path plus the Manhattan distance.

   For example
   ```
   D . . *
   . . . .
   . * * G
   ```
   In the above maze representation, from current position of die(D) there are two Manhattan paths to goal(G). Either it can roll all way to right to goal column and then roll down to goal. Or roll all way down to goal row and then roll all the way to goal column. Of the two Manhattan paths if the die takes the first one it will receive less obstacles 1 as compared to 2 obstacles if it took the second path. The heuristic function will

return 1 as it is the minimum obstacle one will find if it took a Manhattan path. In effect this heuristic does nothing than tell number of obstacles that too in Manhattan path and the Manhattan distance. This heuristic is admissible since the die will have to avoid the obstacle and has to do atleast the Manhattan distance. In case there could be obstacles on both the Manhattan paths but no obstacles in any cell on straight line between, then too the heuristic over estimates but never underestimate since minimally the Manhattan distance needs to be covered. But the heuristic may not be consistent, because in current position there may not be any obstacles on the Manhattan path but in the next position there could be many obstacles on the Manhattan path, this would increase the heuristic value from the previous state.

Performance Metrics:

For straight line distance heuristic.

Puzzle files →	Puzzle1	Puzzle2	Puzzle3	Puzzle4	Puzzle5
Nodes visited	15	60	3	101	1072
Nodes generated	22	78	3	122	1187

For Manhattan distance heuristic

Puzzle files →	Puzzle1	Puzzle2	Puzzle3	Puzzle4	Puzzle5
Nodes visited	15	58	3	99	871
Nodes generated	22	76	3	122	1104

For least obstacles on Manhattan distance plus Manhattan distance.

Puzzle files →	Puzzle1	Puzzle2	Puzzle3	Puzzle4	Puzzle5
Nodes visited	28	89	3	137	1272
Nodes generated	36	100	3	152	1278

Considering path cost only.

Puzzle files →	Puzzle1	Puzzle2	Puzzle3	Puzzle4	Puzzle5
Nodes visited	28	89	3	147	1272
Nodes generated	36	100	3	158	1278

Discussion:

1. The data from five experiments seems to be small to draw any general conclusion about heuristics or their performance.

2. Even though some heuristics are intuitively look to be worse than others. The only Manhattan distance heuristic worked the best followed by the straight line heuristic. The Manhattan distance heuristic with the least obstacle proved to be giving almost no heuristic at all, infact it added to the confusion rather than solving it.

3. Manhattan distance heuristic giving a better solution than straight line distance heuristic can be expected since heuristic 2 dominates over heuristic 1. As it can be seen from the number of states generated for the puzzle 4 and puzzle 5. However the performance of the third heuristic Manhattan with the least obstacle seems to be surprising given the fact that Manhattan with obstacle should give the same performance as that of Manhattan.

4. It was interesting to see whether the heuristics were having any impact or not. For this I sent a heuristic function that returns zero for heuristic and only sends back the path cost. And found that uniform cost search generated same number of states than A* for puzzle 4 and for puzzle 5 as that with the third heuristic.