PROJECT-1 REPORT

Fundamentals of Artificial Intelligence

Anjan Shrestha  
 Computer Science  
 University of North Texas  
 Denton Texas   
 anjanshrestha@my.unt.edu

Vaishnavi Anne  
 Computer Science  
 University of North Texas  
 Denton Texas   
 vaishnavianne80@my.unt.edu

**APPROACH**

* We used two-dimensional array to store grid layout of 6x6.
* Then Created a loop for 1000 episodes
* Moreover, for each episode, we generated a random number between one and the total number of item types and used that number to get random order items.
* Then Created an array of size 2 named as “current\_robot\_position” to store agent position where first index determines the row and second determines the columns of the agent position.
* As well created episode\_score and episode\_path variable to store current episode score and path of the agent.
* Likewise created a list of available items in the available\_items list
* Along with created shortest\_path, longest\_path, shortest\_path\_score, longest\_path\_score, total\_score to store its respective values.
* When the agent finds the order, then it will remove the item from the list and iterates after the order item list is empty.
* For each item in the list, if robot finds the item in the grid, then it will increment episode score by 3 and remove the item from the order item list. If not, then it will deduct 1 from episode\_score.
* Next, it will try to identify the next move using current robot position, order items list and warehouse grid.
* To identify next move, we find the available position for next move based on current robot position, filters the

position that are outside the grid and sense moves based on the sensor accuracy.

* To sense the move based on accuracy, first we generated the random number from 1 to 10 where 1-8 represents 80% of the time robot is accurate, 9 represents 10% of the time where agent is false positive where it senses items even if it’s not there and 10 represents another 10% of the time where agent is false negative where it doesn’t sense item even it it’s there.
* After finding the possible next moves list, we get the random move using random number and assign it to current robot position and append the position as a tuple to episode path.
* Furthermore, update the shortest\_path and shortest\_path\_score if length of shortest\_path is greater than length of current episode path.
* Accordingly update the longest\_path and longest\_ path \_score if length of longest\_path is less than length of current episode path.
* On top of that add the current episode score to the total score and use it to calculate average score.
* To calculate average brute force score, create the function that takes number of items as input and returns the brute force score. To get the average number of items, add all the number of items generated randomly and divided by number of episodes i.e., 1000.
* As a result, print shortest\_path, longest\_path, shortest\_path\_score, longest\_path\_score and average\_score to show the output.
* For modified grid layout, created modified\_warehouse\_grid variable to store a new pattern and created modified\_available\_items to store new available items and send this info to the agent to run for 1000 episodes with random number of items. After that, calculated the corresponding scores.

**ASSUMPTIONS**

1. Warehouse grid is 6 by 6.
2. Agent can move into the grid that contains item
3. Number of items in each order can be taken as a random number between one and total number of item types
4. 80% of the time the robot will correctly sense whether a shelf exists in its neighborhood
5. Assume that the neighborhood consists of a maximum of 4 grid positions which are to the left, right, up and down from its current position
6. 10% of the time the sensor fails, and the robot thinks that a shelf is present when it is not the case
7. 10% of the time a shelf exists but the sensor fails to detect it until it lands on the grid position containing the shelf.
8. If any of the grid positions in its neighborhood refers to an item ordered, then it moves to that grid position. If two or more neighboring positions contain ordered items, then the tie is broken by making a random choice between the positions involved
9. If none of the neighboring positions contain ordered items, then a random choice is made to move to the next node
10. If none of the neighboring positions contain ordered items, then a random choice is made to move to the next node
11. Program stops after executing 1000 episodes
12. Agent can move to already visited grid.

**TEAM CONTRIBUTION**

* Anjan Shrestha (Build agent next move logic using 80, 10, 10 percentage rule and output next move. Also, worked on project report)
* Vaishnavi Anne (Build the episode score, episode path calculation and comparing shortest and longest path of the agent. Also, worked on project report)

**QUESTIONS AND ANSWERS**

1. The average score taken across 1000 episodes.
   * The average score is found to be **-97.528** from one specific agent run with 1000 episodes and random orders.
   * (Attached screenshot on Question 3)
2. Display the shortest path (i.e., the sequence of grid positions visited) across the 1000 episodes and its corresponding score.
   * The shortest path is found to be [(0,0), (1,0), (1,1)] for order items [‘A’].
   * (Attached screenshot on Question 3)
3. Display the longest path across the 1000 episodes and its corresponding score.
   * Since longest path is too long, please find the screenshot below.

­

Graphical user interface, text, application

Description automatically generated

1. Demonstrate that your program is generic enough to cope with a new layout of shelving as shown below. This should be done by carrying out requirements 1, 2 and 3 above for the new layout.
   * To demonstrate that our program is generic enough, we have created a new warehouse grid and new list of available items and passed that info to the agent to run. And we got average score, shortest path, longest path and so on. The output is shown in below screenshot.

Graphical user interface, text, application

Description automatically generated

1. Give two possible reasons why the average score differs from the average score computed across the old layout.
   * The two possible reasons are as follows:
2. The old layout contains only 10 available items but the new one contains total of 16 items. So, while calculating random number of items for each episode out of 1000, there is higher chance of getting a greater number of items in the new layout due to which score differs. (More number of items means more score as score increases by 3 for each item found and grid layout is constant i.e., 6x6)
3. Due to inaccurate sensor (80% accurate) and sometimes being in a loop of the same path (as agent knows about its current state only), the old and new score will differ.
4. What data structure(s) did you use to ensure that your program can cope with any given layout?
   * We have used two-dimensional array which can cope with any given layout. Since two-dimensional array can represent grid layout, it will work if the layout has been modified.