Artificial intelligence

LAB 2: Search

Part 1,

Breadth-first search is the shortest path technique that follows the first in first out. It works into levels if the one vertex is visited and marked then it will go to the adjacent node and mark it as visited.

While on the other end depth-first search goes inside the node to another node. It works in two steps first send the node into the stack and then if there is no node it will pop that node.

In lab 2 we implemented the BFS (Breadth-first search) and DFS (Depth-first search). First, we check either the frontier is empty or not, we checked the condition until the frontier is not empty. Afterward, we added a node into to frontier. After adding the node we identify the state of the current node and if the goal state is equal to the node, means we found the node otherwise we will add as explored. We loop through all the reachable states and check if the node is not already explored. Then we assign the Boolean variable insert front if the variable is true then it will work like DFS or if it's false then it will treat as BFS and return the path by the end.

LAB 2, Part 2 - Theory

1. In a vacuum cleaner we have multiple states and actions, some of the states like dirt, bump, and clear which are some of the conditions that agent encounters while cleaning the floor, while on the other end actions are forward, left, and right. The branching factor can be defined as the maximum number of nodes which are the successor of the current node.

2. Breadth-first search(BFS) can be considered as the search technique where you take the current node and extract the nodes which are successors of the current node considering the cost of each action as same. But in Uniform cost search(UFS) can be considered as a breadth-first search which takes into consideration the path cost. But in the question, we have the path cost as 1 for the uniform cost search and we can conclude that BFS and UFS are the same if we have the path cost as one.

3. An admissible heuristics (h(n)) is the one that never overestimates the cost to reach a goal node from the current node.

a) If h1 and h2 are admissible heuristics then we can conclude that (h1+h2)/2 is also admissible and consistent.

b) Even if h1 is admissible we cannot say that 2\*h1 is admissible since we cannot say that 2\*h1 is less than or equal to the actual cost of a least-cost path from node n to a goal.

c) max (h1,h2) can be admissible since h1 and h2 are admissible so we can derive that the max of h1 and h2 is also admissible.

4. A\* search is an informed search strategy where we have a function f(n) which is the sum of g(n), the cost to reach the node n, and h(n), the cost to get from the node n to the goal node. Hence f(n) should be the estimated cost of the cheapest solution through n.

In the vacuum domain, we can define the h(n) as the no of steps or cost to reach from the current node to the one specific square which is the goal.

The cost function g(n) be the cost of reaching the current square from the initial starting square position. The choice of h(n) is not admissible if we are using the uninformed search techniques such as BFS and DFS and hence not admissible they do not consider the cost of reaching the goal state.

Since we are using A star it is guaranteed to give an admissible heuristic because each time when an action is made, the sum f(n) of the cost function, g(n) and heuristic function, h(n) is considered and so, in the end, it will return an admissible heuristic since the value of f(n) will be the lowest in A star search.

5. We have selected three main search strategies which are A star, Breadth-first search, and Depth-first search.A picture containing text, whiteboard

Description automatically generated

As seen in the above image we have the f(n)= g(n) + h(n), where g(n) is the cost function and h(n) is the heuristic function. Our goal is to find the Node F from the starting state A. So we are calculating the function f(n) every time from starting from node A. We then move on the node which gives the lowest value of f(n) holding the other value in the memory. We calculate the same for the subsequent nodes until it reaches the goal state and chooses the path which has the least value for f(n), hence in the example we choose the path from A to F directly as it has the lowest value 9.

Text, letter

Description automatically generated

Here we consider the case of BFS where we are moving in the order as shown in the above example, in a level ordered manner. It is an uninformed search and considers the cost of each action to be equal. It is a First in First out queue in the memory as it always pops the first element of the queue. We can see that the memory requirement is more for BFS than DFS since BFS expands all the child nodes and keeps it in the memory.

DFS uses a stack where the Last In First Out method is used since it iteratively grows only one child until it cannot move forward and then it backtracks. Therefore, it has a lower memory requirement than BFS. But both DFS and BFS have the same time complexity, a similar time to find the goal node.

6.

Breadth-First Search is complete since it expands all the nodes until the last level and hence it will definitely find the solution if the number of nodes is finite. They are optimal only if all the actions have the same cost otherwise we cannot say that the goal node is not necessarily optimal since it goes in a level ordered manner.

Depth First Search is not complete because it can keep on going on an infinite path. They are also not optimal because it expands an entire left subtree node after one even though we have the goal node in the right subtree.

Uniform cost search is also complete only if the branching factor is finite and if the cost of step is something positive. Uniform cost search is optimal since it keeps on checking even if after finding the goal node in order to find the most optimal path.

The best first search is optimal and complete if the heuristic function is admissible and consistent. Since the optimal sum of path cost and heuristic is always selected.

Iterative deepening search is complete when the branching factor is finite and optimal when the path cost is a non-decreasing function of the depth of the node or in other words, step cost is all identical. It is used when the search space is large and the depth of the solution is unknown.

Bidirectional search is complete if the branching factor is finite and, both from the front and backward searches use the breadth-first search. It is optimal if the cost of actions is identical and both the searches use breadth-first search.

Greedy best-first search is incomplete even infinite search space and it is also not optimal since it searches to in a greedy manner, so as to reach to the goal node as soon as possible.

A\* search is both optimal and complete because it considers both the path cost to reach the current node and also the heuristic function, which is the estimated cost of the cheapest path from the current node to the goal node.

7. If we are moving to the first lab and trying to implement a search function we would definitely use the Breadth-first search since our current scenario has a finite branching factor that is known. And also we know that all the step costs are the same and hence it would be a good option to choose the BFS as it will clean all the dirt in the vacuum environment.