HARSHDEEP J.
SE21UARI044

Why one search technique is better than the other one?

Introduction

Best First Search:

Only nodes which seem closest to goal state are expanded. The evaluation function estimates the cost from present state to the final state.

A* Search:

It is a combination of Uniform Cost Search and Best First Search. It is the sum of least cost from start state to the present state and the estimated cost to go from present state to goal state.

File Name Given to each program:

- Best First Search:
  - Using Heuristic h1: BFS_h1.cpp (Output in "Output h1.txt")
  - Using Heuristic h2: BFS_h2.cpp (Output in "Output h2.txt")
- A* Algorithm:
  - Using Heuristic h1+g(n): A_star_h1.cpp (Output in " Output A_h1.txt")
  - Using Heuristic h2+g(n): A_star_h2.cpp (Output in " Output A_h2.txt")
- Hill Climb Search
  - Using Heuristic h1: Hill_Climb_Search_h1.cpp (Output in " Output Hill_Climb_Search_h1.txt")
  - Using Heuristic h2: Hill_Climb_Search_h2.cpp (Output in " Output Hill_Climb_Search_h2.txt")

Comparison:

Best First Search		Criteria	A* Search	
h1(n)	h2(n)		g(n)+h1(n)	g(n)+h2(n)
Success	Success	Success/Failure	Success	Success
21	717	Number of States Explored	51	26
369		Average Number of States Explored	38	
11	73	Optimal Path Cost	11	11
42		Average Optimal Path Cost	11	
0.005	0.018	Time taken for execution	0.007	0.013
0.0115		Average Time taken for execution	0.0100	

Reason why A* Search turns out to be better than Best First Search

A* Search turns out to be better than Best First Search as all the steps taken in A* consider both, the number of steps taken so far, as well as the number of estimated steps need to be taken to reach goal node. This not only helps to move towards the destination but also to take into consideration the steps taken so far and see if there is possibility of being in a better place if we had taken the same number of steps so far in any other direction. This guarantees(given that heuristic is admissible and consistent) that the path given by A* is optimal. Adding increasing values of heuristic guarantees that the algorithm is complete and the Time Complexity of A* Search is exponential to the path length and all nodes are stored in it.

A* Search			
Complete	Optimal	Time Complexity	Space Complexity
Yes	**Yes* *: if heuristic is admissible and consistent**	**O(b^m) where b = branching factor and d = depth**	**O(b^m) where b = branching factor and d = depth**

Whereas, in BFS, only the estimated steps towards the goal node from the present node is taken into consideration and this can lead to a path, which might be reaching towards the goal but might not be optimal as we dont consider the number of steps taken so far.

Best First Search			
Complete	**Optimal**	**Time Complexity**	**Space Complexity**
Yes* *:if cycle checking is done	**No**	**O(b^m) where b = branching factor and d = depth**	**O(b^m) where b = branching factor and d = depth**

What if the search algorithm got stuck into Local optimum? Is there any way to get out of this?

There are many ways to get out of Local optimum:

1) Start moving in a particular direction until you get out of Local optimum.
2) Start moving randomly in any direction until you get out of Local optimum.
3) Trying Simulated Annealing and move to the next neighbour with probability exp{-ΔE/T)
    a) This returns solution even if we are stuck in local optimum
    b) Time constraint is present.
4) Trying Genetic Algorithms to get a successor by combining any two states available and try mutate it.