Grundlagen der Wissensverarbeitung

Blatt 3

Daniel Speck, Lena Niermeyer

31.10.2015

Aufgabe 1.3 (Blind Search)

1. to 3.: see .py-files attached

4. Problems of search strategies

BFS:
- every visited node has to be saved in a set (could take vast amounts of memory), otherwise the search will end in loops.
- in a tree where each node has an infinite number of children, but the tree has a finite height, BFS might not terminate. It will only visit the children of the root node and if the node we're looking for is the child of some other node, it won't be reached.
- BFS is much worse memory-wise. DFS is linear space, BFS may store the whole search space.

DFS:
- in opposite to BFS not optimal, because DFS may produce a much longer path than BFS
- in a tree where each node has a finite number of children, but the height of the tree is infinite, DFS might never find the node we are searching for - it will just keep visiting the first child of every node it sees, so if the one we're looking for isn't the first child of its parent, it will never get there.
- In the worst-case BFS is always better than DFS in time complexity

A*-Search:
- will always find an optimal solution, but only if there is a solution (goal)
- As A* keeps all generated solutions in memory it runs out of space long before it runs out of time.

5. Other Problems of search strategies in other environments

i.) New environment: Different costs for different ways.
Problem: The shortest way is not longer the best way. There might be a longer way with lower costs.
ii) New environment: Number of goals unclear.
Problem: Since the number of goals is unclear, the agent does not know when

to stop searching. He has to search the hole maze.

iii) New environment: infinite maze.
Problem: Search algorithm might never terminate, because the maze is infinite.

6. Possible ways to cope with identified problems

regarding i) Search Variation: Uniform-cost search. When all steps don't have the same cost, at every point we can expand the node of the lowest path cost instead of the closest to the origin. Equivalent to the Dijkstra algorithm.
regarding ii) choose DFS before BFS, because with DFS every node is crossed maximum twice and through the complete maze has to be searched the needed memory space will be smaller than for BFS.
regarding iii) choose BFS before DFS, because BFS is complete.