Artificial Intelligence for Robotics - Assignment 04

Deepan Chakravarthi Padmanabhan
Muhammad Umer Ahmed Khan

November 2018

1. Give an explanation to prove the following statements:

   - **Breadth-first search is a special case of uniform-cost search:**
     Uniform cost search(UCS) utilizes a priority queue managed depending on the cumulative cost to reach the node. Uniform Cost Search gives the minimum cumulative cost the maximum priority. In Uniform cost search, the evaluation function that guides the search f(n) is given as below:
     Let the path be considered as: 1-2-3.

     $$f(n) = g(n) = \text{path cost of node 1-2} + \text{path cost of node 1-2-3}$$

     Here, g(n) is the total cost to reach a node 'n' from the start and no heuristic function is used here. UCS expands the node with the lowest g(n).
     BFS primarily visits the node with the shortest path length (number of nodes) from the root node, UCS first visits the node with the shortest path costs (sum of edge weights) from the root node. Here, BFS explores vertices in the order of their distance from the source vertex. The distance is the minimum length of a path from source vertex to the node.
     Breadth First Search(BFS) is a special case of uniform-cost search when all edge costs are positive and identical. In this case, the evaluation function is given as below: For instance, step cost =1

     $$f(n) = g(n) = 1 * depth(n)$$

   - **Breadth-first search, depth-first search, and uniform-cost search are special cases of Greedy Best-First Search:**
     BFS, DFS and UCS are uninformed search methods. However, all the searches are guided by the evaluation function f(n). The uninformed search methods- BFS, DFS and UCS are special cases of Greedy Best-First search. The f(n) of Greedy Best-First Search is given by:

     $$f(n) = h(n)$$

     h(n) is the heuristic function. This is an estimation of cost from n state to the closest goal. The below f(n) illustrates the BFS, DFS and UCS working conditions of Greedy Best first search .

     f(n)= depth(n) ; then Greedy Best First Search works as BFS

     f(n)= -depth(n) ; then Greedy Best First Search works as DFS

     f(n)= g(n) ; then Greedy Best First Search works as UCS

   - **Uniform-cost search is a special case of A* search:**
     The evalution function for A* search is given as:

     $$f(n) = g(n) + h(n)$$

     Here g(n) is the actual cost to reach node n from the start and h(n) is the estimate of cost from n to the closest goal. A* Search reduces to uniform cost search when the heuristic function is zero everywhere, i.e. h(n) = 0 for all n. This heuristic is clearly admissible since it always underestimates the distance remaining to reach the goal. Thus, UCS is A* with h(n)=0.

2. Answer the following questions regarding A* search:

   - When is A* complete?
     The 2 completeness conditions of A* states are- no infinite paths have finite cost and consistency of the heuristic. When the heuristic is inconsistent, the graph-search algorithm has to be modified to ensure completeness. The consistency of a heuristic is analogous to the triangular inequality and ensures f(n) increases along any path. Hence, if there are no infinitely many nodes with f ≤ f (G), where G indicates the shortest optimal goal, the algorithm is complete. This ensures that A* returns a solution.

   - When does A* end the search process?
     A* determines the expansion path based on the the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A* selects the path that minimizes

     $$f(n) = g(n) + h(n)$$

where n is the next node on the path, g(n) is the cost of the path from the start node to n. h(n) is a heuristic function that estimates the cost of the cheapest path from n to the goal. A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended.The heuristic function is problem-specific. If the heuristic function is admissible, h(n) never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal.

A* algorithm expands all nodes with f (n) < C*, where C* is the cost of the optimal solution path. Sometimes, it might then expand some of the nodes right on the goal contour (where f (n) = C*) before selecting a goal node. If this is satisfied, the algorithm ends the searching process. A* expansion criteria is as given below:

$$\text{If f(n) is consistent, A* does not expand nodes for f(n)} > C^*$$
$$\text{A* expand nodes for f(n)} < C^*$$
$$\text{A* expand some nodes for f(n)} = C^*$$

3. During the lecture you have discussed two heuristics for the 8-puzzle: Manhattan distance and misplaced tiles (see slides if not yet covered in class). Your tasks for this week are:
• Implement a Greedy and A* agent for the 8-puzzle. The agents should be able to switch between both heuristics. Make sure to produce proper output to "visualize" the working of your program.
• Compare the performance of the solvers and the two heuristics. Provide data in your report to support your arguments (number of visited nodes, path cost, execution time, etc). Which works better?

Algorithm	Heuristic	Transitions/ Cost	Time	Frontier Push	Frontier Pop	Extensions
A*	Manhattan	15	00:00:00.03	107	66	174
A*	Misplaced Tiles	15	00:00:52.60	5874	3699	10048
Greedy	Manhattan	23	00:00:00.05	142	83	224
Greedy	Misplaced Tiles	15	00:00:52.83	5874	3699	10048

From the statistics provided, it is evident that Manhattan distance has less cost in terms of time compared to misplaced tiles. Also, it shows that A* provides the optimal solution, while in Greedy algorithm, the optimality is not guaranteed.

The code has been attached inside the src folder. The transitions of tile states can be viewed inside the src_output folder files. In the end of each file is a summary of their statistics which are more detailed compared to the provided above.

The code can be executed using Python3. The files output files were generated using the following commands in terminal,

- python Main.py A* ManhattanDistance > src_out/Astar_Manhattan.txt
- python Main.py A* NumberOfTiles > src_out/Astar_NumberOfTiles.txt
- python Main.py Greedy NumberOfTiles > src_out/Greedy_NumberOfTiles.txt
- python Main.py Greedy ManhattanDistance > src_out/Greedy_Manhattan.txt

The input and goal states can also be varied by changing the Input.py file.

References:
1. PDF Document: CompSci 171: Intro AI-PDF, Accessed on: 15-11-2018, \\URL: https://www.ics.uci.edu/~welling/teaching/271fall09/HW3_sol.pdf\\
2. Stuart Russell and Peter Norvig, 'Artificial Intelligence A Modern Approach- Third Edition', Pearson Education Limited, ISBN:1292153962, 2016.