MELIS OKTAYOGLU 0064388
11/6/22

COMP341 – Assignment 1 Report

Q1 - ***What are some differences between DFS and BFS in terms of path cost and number of expanded nodes? When and why would you prefer BFS over DFS? When and why would you prefer DFS over BFS?***

DFS and BFS differ by the order they put and release nodes to the fringe. When the goal is at the furthest place, DFS will expand less nodes than BFS does, expanding less nodes also implies faster search. However, since it prioritizes further nodes, it might return a longer path even if there exists a shorter one as soon as it finds one. But BFS will return shortest path since it expands the closest paths in each round meaning it is optimal (for an unweighted graph). But in the case of a goal closer, BFS will expand less nodes thus becomes faster while DFS might go about searching for a very long path without a goal node. This long search, depending on the size of the problem, might result in not finding a solution at all. Thus, DFS is neither optimal nor is complete. All in all, having an idea on the nature of the problem will guide us on choosing between DFS and BFS. If we don't really have a clue where our goal is on a medium size problem, we can go about with BFS, but if we don't care about finding the shortest path but finding a far goal, we would choose DFS.

Q2 - ***What are some differences between UCS and A* in terms of path cost and number of expanded nodes? When and why would you use UCS over A*? When and why would you prefer A* over UCS?***

The main difference is using heuristics; A* uses thus is informative while UCS doesn't thus is uninformed. By using a heuristic that gives us an idea about which nodes to explore first, using a priority queue, we can optimize our search through avoiding expanding the nodes which we quantify with our heuristic to be less likely to lead us to our goal. Consequently, A* will expand less nodes than UCS. But both are optimal. So, whenever we can formulate an admissible heuristic (which generally is in terms of a distance) we should use A* for better search time.

Q3 - ***Comment on your choice of state in the finding all the corners problem. Why does it allow you to solve the problem?***

I picked the state using the currently expanding position, and the currently unvisited corners. Keeping track of unvisited corners for each state position allows me to check when to stop after reaching the goal. Without keeping track of the cornets visited (that is goal conditions) in some form, there is no way to know in which game state we reach all corners.

Q4 - ***Comment on your choice of heuristic in the finding all the corners problem. Why did you settle on that heuristic? Why is it admissible and consistent?***

My choice was Manhattan distance, it was available in the util which was convenient, and I set the max of distances as heuristic. I picked the sum because it acts as a lower bound to visit all the corners. It is admissible because of taking the lower bound it doesn't overestimate the cost to reach all goals. It is also consistent because every step is actual cost 1, but our heuristic cost is never smaller than 1 (unless we are in the goal state which other possible actions will give 1 heuristic cost).

Q5 - ***Comment on your choice of heuristic in the eating all the dots problem. Why did you settle on that heuristic? Why is it admissible and consistent?***

My logic was the same for choosing a heuristic for eating all dots problem with finding all corners problem. Thereby all the same explanation applies here. It is admissible because sum of Manhattan distances to all remaining foods are at most equal to reaching to every food, because there are also some blank distances need to be traversed to get to them all. It is again consistent because either taking the step will increase or decrease the Manhattan cost by one, and sum of our heuristic cost always will remain an upper bound.

Q6 - ***What are some practical differences between a consistent and an inadmissible heuristic, in terms of path cost and number of expanded nodes? When and why would you prefer an inadmissible heuristic over a consistent one? When and why would you prefer a consistent heuristic over an inadmissible one?***

Admissibility is picking a heuristic that overestimates the cost of reaching the goal, it isn't optimally informative and thus doesn't yield optimality of the solution. An inadmissible heuristic will make us expand more nodes. With A*, an admissible heuristic will make the search tree optimal. Consistency means that our heuristic cost would monotonically decrease by our guided picks of moves. A consistent heuristic will make the search graph in A* optimal. Also, consistency ensures admissibility. We can thus pick a consistent heuristic over and inadmissible one because it ensures admissibility.