CS-487 (Introduction to AI)
Fall 2023

Assignment 1

1. **Deadline:** Saturday, 21/10/2023 on e-learn (https://elearn.uoc.gr/)
2. **Deliverables:** Submit a zip file containing a report in PDF with the answers AND all code files (included the `.ipynb`) written by you in the scope of the assignment.

1 Theoretical Exercises

Exercise 1

Both the performance measure and the utility function measure how well an agent does. Explain how they differ.

Exercise 2

Can such graph exist in which A* extends more nodes than the Depth-First Search algorithm? If so, draw an example of such a graph. If not, explain why this is not possible.

Exercise 3

1. Suppose we run a greedy search algorithm with $h(n) = g(n)$ (recall that $g(n)$ gives the path cost from the start node to node n). What sort of search will the greedy search emulate?

2. Sometimes there is no good evaluation function for a problem, but there is a good comparison method: a way to tell if one node is better than another, without assigning numerical values to either. Show that this is enough to do a best-first search. What properties of best-first search do we give up if we only have such a comparison method?

2 Programming Exercises

Exercise 4 - Comparison of Search Algorithms

In this exercise you will run various informed and uninformed search algorithms to solve the 8-puzzle problem. You can use the code of the book given to you (Note that the `Assignment_1.ipynb` file needs to be inside the `aima-search` folder). The algorithms you can use can be found on `search.py`. You will have to run the code 100 times for the 8-puzzle, for random puzzles each time. Because some algorithms may take too long to solve a problem, we set an upper limit on the number of states that each algorithm can visit (10^7 in our case). In case the algorithm fails to find a solution (it has reached the limit of the number of states it can visit, it has reached the limit of the memory it can use or for some other reason) the value -1 is stored as the number of states and the size of the solution.

Use at least 4 algorithms of your choice you have to compare the algorithms based on the above measurements in terms of the complexity and quality of the solution, since a solution has been found. Also calculate how many times each algorithm found a solution. How you compare them is up to you. For example, you can calculate the transaction factor, various statistics (mean values, standard deviation, etc.), create graphs (see also in the book how search algorithms are compared). You can use whatever tools you want for this purpose and modify the given code accordingly. One should be able to decide which algorithm to use based on your comparison. Also answer the following questions:

- What results would you expect based on the theory?
- Are they verified by experiments? Comment on that.
- Which algorithm would you choose to use? Justify your answer.
- How does the size of the optimal solution affect the performance of the algorithms?
- Experiment with the maximum number of states (variable `max_actions`) (i.e. run the algorithms several times with different variable values). What do you notice? Justify your answers.