Brenda Guadalupe Flores Parada A01206527
Daniel Gámez Serna A01206539
Report Lab 3

Implementing (Un)informed Search Algorithms

Introduction

The following text shows the result of applying the knowledge about different search algorithms in order to solve different variants of the crane puzzle and some analysis the team went through.

Which heuristics did you use for the A* algorithm?

For a consistent heuristic we decided to use the number of stacks that are different from the goal state.

For an inconsistent heuristic we just put a random number with a range from 1 to 100.

Test your program with a couple of different problems. Increase the size of the problem to test the limits of your program. Make a table comparing how many nodes are searched to find the answer for each problem. For this table, you should compare a number of different problems (at least 3) to avoid a statistical bias. Which of the three algorithms (UCS, A with consistent and and A with an inconsistent heuristic) searches the least nodes and which one take the most?

Problems	A* search (Cons heu)	A* search (Incons heu)	UCS
3 (A);(B);(C);(D) (C,A);(B);(D);( )	(0, 1); (2, 0); (1, 0); (3, 2) (4 moves)	(1, 3); (0, 2); (2, 1); (2, 0); (3, 2); (1, 0); (2, 1); (3, 2) (8 moves)	(0, 1); (2, 0); (3, 2); (1, 0) (4 moves)
4 (A,B);(C,D);(E,F) (A,E,C);( );(D,B,F)	(1, 0); (2, 1); (2, 1); (0, 2); (0, 2); (1, 0); (1, 2); (1, 0) (8 moves)	(0, 2); (1, 0); (2, 1); (0, 1); (2, 1); (1, 0); (2, 0); (1, 2); (1, 0); (0, 2); (0, 2); (2, 1); (0, 1); (1, 2); (1, 0); (1, 0) (16 moves)	(1, 0); (2, 1); (2, 1); (0, 2); (0, 2); (1, 0); (1, 2); (1, 0) (8 moves)

1 **(A);(B);(C);()** **();(C);(B);(A)**	**(0, 3); (1, 0); (2, 1); (0, 2)** **(4 moves)**	**(2, 3); (0, 2); (1, 0); (3, 1); (2, 3); (0, 2)** **(6 moves)**	**(0, 3); (2, 0); (1, 2); (0, 1)** **(4 moves)**
3 **(A,B,C);(D);(E,F)** **(F,A);(B);(C,D,E)**	**(2, 1); (2, 1); (0, 2); (1, 2); (1, 2); (0, 1); (0, 1); (2, 0); (1, 0); (1, 2); (1, 0); (2, 1); (2, 1); (0, 2); (1, 2)** **(15 moves)**	**(2, 1); (2, 1); (0, 2); (1, 2); (0, 1); (1, 2); (1, 0); (1, 0); (2, 1); (0, 1); (0, 2); (1, 0); (2, 0); (2, 1); (0, 1); (0, 2); (0, 2); (1, 0); (2, 0); (1, 2)** **(20 moves)**	**(2, 1); (2, 1); (0, 2); (1, 2); (1, 2); (0, 1); (0, 1); (2, 0); (1, 0); (1, 2); (1, 0); (2, 1); (2, 1); (0, 2); (1, 2)** **(15 moves)**

With each test case we tested the algorithm the results between A* with a consistent heuristic and UCS are the same, while A* search with a inconsistent heuristic is not, taking it more movements than the other two algorithms to solve the problem.

Why does this happen?

UCS takes the same number of movements than A* search because it uses the same algorithm with a constant heuristic of 1. A* with an inconsistent heuristic takes more movements to solve the problem because the heuristic affects the priority queue, so it don't get the optimal solution.

Which algorithms are optimal? Why?

A* and UCS algorithm are optimal because they look-up for the shortest path to reach the goal.

They maintain a priority queue of options for movements, ordered by how good they might be. Those algorithms keep searching until it finds a route to the goal that's so good that none of the other options could possibly make it better.

In your opinion, what are the benefits of simpler algorithms versus more complex ones?

Using a more complex algorithm you can solve almost any kind of problem by taking into account more variables in order to reach a goal state but it takes a lot more thinking while making a simpler algorithm can solve easier problem and sometimes faster or slower depending on the problem, an optimal solution is not granted.