Intelligent systems - Laboratory 3.
Implementing (un) informed search algorithms
09-18-2018
Fernando Miguel Arriaga Alcántara

Introduction

The objective of this lab is to implement three search algorithms in order to provide a solution to the container crane problem. The algorithms implemented were Uniform Cost Search, A* with a consistent heuristic and A* with a non-consistent heuristic. For these three algorithms, the graph version was implemented using C++.

Heuristics implementation

All the heuristics are based on the idea of misplaced crates, but each one has a different approach:

For this lab, two non-consistent heuristics were implemented:

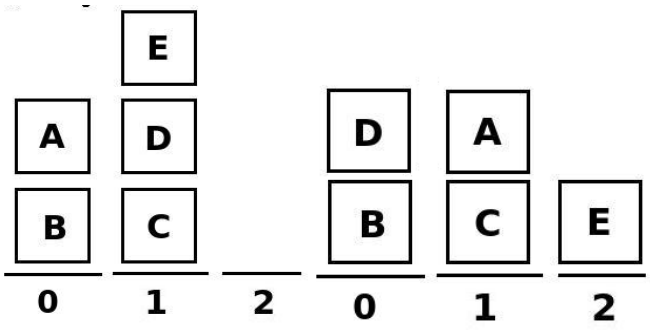
The first one estimates cost by the absolute value of the difference in size of each one of the stacks between the current state and the goal state. This difference is multiplied by two in order to be closer to the real path cost.

The second one builds upon the previous one, also taking into account the number of misplaced crates. For every crate that is not on the right place two minutes are added to the final estimation

Lastly, the consistent heuristic implemented for A* is an improvement upon the previous two heuristics. If a crate is found in the place where another one should be, two minutes are added to the final cost, and the difference in the size of the stacks is added directly to the final estimation.

If one stack of the current state is bigger than a stack from the goal state, then this means that at least another stack from the goal state will be bigger than its corresponding stack of the current state. If we only add the difference, it is guaranteed not to overestimate the cost. On the other hand, If a crate is found in the place where another one should be, two minutes are added to the final cost.

Example:



Having the initial state on the left and the goal state no the right, the result of each heuristic will be:

- H1 = 4 min (Difference in height between stack 1 of current and goal state is 1, this is multiplied by two and the same occurs on stack 2, having four as a result).
- H2 = 8 min (Same as before but two minutes are added for each crate occupying the space of another one, so it is +2min for A and +2 min for D)
- H3 = 6 min (D and A add 2 each one and the difference between stacks is 1 for both of them, having 6 as a result)

Performance comparison

A set of problems were used to test the algorithms giving the following results:

Problem 1.

3
(B, A); (C, D, E); ()
(E); (C, B, A); (D)

ALGORITHM	NODES EXPANDED	EXECUTION TIME	SOLUTION COST
UCS	1204	9.85 sec.	17 min.
A* (NON-CONSISTENT 1)	374	1.44 sec.	17 min.
A* (NON-CONSISTENT 2)	190	0.97 sec.	17 min.
A* (CONSISTENT)	133	0.78 sec.	17 min.

Problem 2.

4
(A, B); (C, D); ()
(C, D); (A, B); ()

ALGORITHM	NODES EXPANDED	EXECUTION TIME	SOLUTION COST
UCS	754	5.38 sec.	17 min.
A* (NON-CONSISTENT 1)	470	1.93 sec.	17 min.
A* (NON-CONSISTENT 2)	404	2.20 sec.	17 min.
A* (CONSISTENT)	158	0.69 sec.	17 min.

Problem 3.

4
(A, B); (C, D); (); ()
(C, D); (A, B); (); ()

ALGORITHM	NODES EXPANDED	EXECUTION TIME	SOLUTION COST
UCS	6960	665.33 sec.	17 min.
A* (NON-CONSISTENT 1)	1914	21.19 sec.	17 min.
A* (NON-CONSISTENT 2)	1468	21.50 sec.	17 min.
A* (CONSISTENT)	596	3.567 sec.	17 min.

Problem 4.

5
(A, B, C, D, E); (); ()
(); (X); (D, B, E)

ALGORITHM	NODES EXPANDED	EXECUTION TIME	SOLUTION COST
UCS	2102	25.81 sec.	18 min.
A* (NON-CONSISTENT 1)	606	2.02 sec.	18 min.
A* (NON-CONSISTENT 2)	72	0.526 sec.	18 min.
A* (CONSISTENT)	277	1.318 sec.	18 min.

Conclusions

The general behavior of the tests show that a better heuristic can lead to smaller processing times as well as a smaller number for nodes expanded. A* with a consistent heuristic performed better than the rest almost on each case, except for the last test. However, only A* (consistent) and UCS are guaranteed to give an optimal solution. Even though the other algorithms also gave the optimal solution, it is very important to state that this is not guaranteed for every problem as the algorithms are implemented using a graph-search.

Having a better heuristic means that only the nodes that could give an optimal solution are expanded, that is why the number of nodes expanded is reduced for every next algorithm, except for the last test. This could be because the second non-consistent heuristic makes an estimate that might be closer to the actual cost, leading to a quick solution, but then again, this will not guarantee for solution to be optimal.

It can be observed from the result that the algorithm with the less complex heuristic (1) can find a result in a similar time than the other two, even though it expands more nodes. It requires much less time to calculate the value of the heuristic as the other two, which explains its very decent performance with respect to execution time.

The results show how a simpler algorithm like UCS can and will find an optimal solution if there is one; however, its time and space requirements are quite large.
This also shows the advantages of simpler heuristics as they will not require too much time to design and they will find a result in a reasonable amount of time when compared to ones that are more complex. If design and implementation time is short, maybe a simple heuristic could do the job. However, I think that it is still the best choice to implement a very good heuristic as it will bring optimal results with very minor and space requirements.