Intelligent systems - Laboratory 3.
Implementing (un) informed search algorithms 09-18-2018
Fernando Miguel Arriaga Alcántara A01270913

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 and can be easily shown to overestimate the real cost.

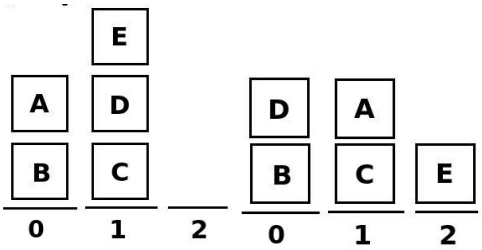
The second one adds two minutes for every crate that just needs to be moved to another place and adds a 3 for every crate that occupies the place of another crate. The way this is implemented in the code is to calculate the difference in stacks size and add it directly to the estimation. A comparison is made between the stack members of both states and adds three minutes if they are different.

This heuristic proves to be inconsistent on the following case:
Current state: (A); (B); ()
Goal state: (); (A); (B)

Lastly, the consistent heuristic implemented for A* is just the number of misplaced crates multiplied by the minimum cost (2 minutes).

The code implementation is done on the same way for the previous two heuristics. Each member of the stack is compared with the goal state and if they are different, two or three minutes are added (depending on the heuristic. This is repeated until one stack is empty and then the difference in the size of the stacks is added. 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.

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 (D and A are on the place of another crate so three minutes are added for each one, then two minutes are added because E is on the wrong place)
- H3 = 6 min (D, A and E are misplaced, adding 2 minutes for each ones gives 6 minutes 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)	75	0.66 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)	90	0.24 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)	297	1.499 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)	199	1.04 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. Even though every algorithm 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, so non-consistent heuristics might not return optimal solutions.

Having a better estimation means that an answer will be found faster, expanding less nodes on the way. It can be observed from the result that the algorithm with the less complex heuristic (1) can find a result in a decent amount of time, even though it expands significantly more nodes. It requires much less time to calculate the value of the heuristic than the other two, which explains its execution time performance, only needing about one more second to find an answer on most of the tests.

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 available 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.

Lastly, a very interesting case occurred with heuristic two, as it gives correct answers much faster than the other algorithms even though it is not consistent. It is possible that it is inconsistent for a very small amount of cases, showing a good behavior on most of the cases. This is a good example of how one can exchange the certainty of having an optimal result for a good result with small space and time needs.