Erick Ibarra
Gustavo Gutiérrez

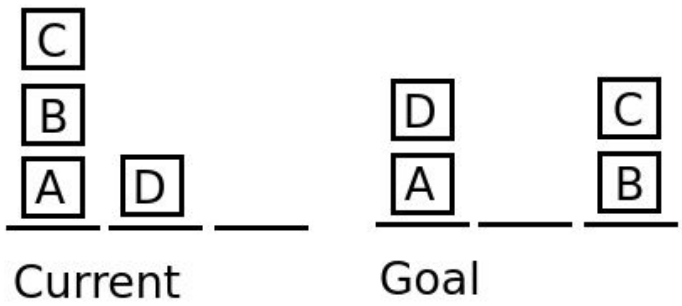
Lab 3

Implementing (Un)Informed Search Algorithms

Which heuristics did you use for the A* algorithm?

For the consistent heuristic, we used the quantity of stacks that are in an incorrect state; in other words, the stacks that do not are in their required configuration by the goal state, add 1 to the heuristic calculated at any given state. There is a special case we need to be aware of: when there is a stack in the final state that can be occupied with any boxes (or none), depicted as *'X'*, the stack is not checked in the heuristic function.

For the inconsistent heuristic, we decided to use the sum of Manhattan distances from the current state in which every box is to its final position. This heuristic proved to be inconsistent in the following case:



h(n) = 0+3+3+2 = 8

In this case, the heuristic is greater to the optimal solution, which can be completed with a cost of 4.

Problem descriptions

- Problem 1
  3
  (A); (B, E); (C); (D,F)
  (F, A, B); (C); (D); X

- Problem 2
  4
  (F); (D, E); (A, B, C)
  (A, B, C); (D, E); (F)

- Problem 3
  3
  (D); (A, C, F); (B); (G, H)
  (A, F); (B, G); (C, H); (D)

Erick Ibarra
Gustavo Gutiérrez

Problem comparison table

	Problem 1	Problem 2	Problem 3	Avg. rank
BFS	12820	4952	128977	3rd
DFS	8215	7901	12116	2nd
A* consistent	18382	5855	137127	4th
A* inconsistent	3282	1289	32783	1st

Why does this happen?

The A* Search with an inconsistent heuristic turned out to be the best on the 3 problems presented here. While the A* Search with a consistent heuristic was the worst in terms of number of expanded nodes. This was a little bit surprising at first, but as we analyze deeper, it should be a totally expected behavior. The this is that, as long as an heuristic is admissible and consistent, A* Search will always give the optimal solution (but that doesn't mean that it will be the fastest or less memory-consuming). But there is wide range of heuristics for every problem, and even though there could be various consistent heuristics, some are better than others, which can be reflected in the number of expanded nodes.

As a final thought, BFS and DFS depend heavily in the initial state in which the problem is presented, as well as the possible actions that can be made in each state. BFS or DFS could find a solution way before the other algorithms, but is often a matter of luck to expand the correct node.

Which algorithms are optimal? Why?

From the four algorithms we implemented for this lab, the A* Search algorithm will give the optimal solution (if there is one) of the problem, nevertheless it needs of an admissible heuristic in order to be optimal.

In case of DFS Search, it will be non-optimal if the limit specified is less than the level (depth) in which the solution will be found.

In the case of BFS Search, it can be optimal if the path cost is a nondecreasing function of the depth of the node. In any other case, it will not give an optimal solution every time.

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

We think that for every problem, there is a clever way to solve it, but it depends not just in the problem but its constraints, like the time that we can use to solve the problem, as well as the processing power we have, also the data that is provided beforehand plays an important role. Simpler algorithms often mean that is easier to any person to understand it, thus being easier to implement.

Erick Ibarra
Gustavo Gutiérrez

Another important thing to consider is that simpler algorithms with more data provided can beat complex algorithms with less data. This is because it can reveal a nonlinear relationship or behavior when more data is provided. This is explained further in the article Garrett Wu made, called “Why More Data and Simple Algorithms Beat Complex Analytics Models”