Universitat de Barcelona

Facultat de Matemàtiques i Informàtica

Inteligència Artificial

Pràctica 1: Elementary Search Algorithms

Authors:
Martí Vila Rebellón
Martí Lázaro Bagué

Date:
October 5, 2025

Introduction

In this initial project, we were asked to investigate the A* search algorithm, an efficient method used to determine the optimal path to a target node within a graph. Using the base code provided by the university, and the example of the chess implementation, we analyzed, implemented, and tested the algorithm, to finally obtain these results.

Objectives

- Develop an example of the A* algorithm in Python to predict the best movements for the white pieces to kill the black one.
- Return a list of visited states from the origin to the target
- Return the minimum depth to reach the target
- Make additional tests with a different start position for the white king piece

1 List of visited states and minimum depth

A* move sequence:

[[[7, 0, 2], [7, 5, 6]], [[6, 4, 6], [7, 0, 2]], [[5, 3, 6], [7, 0, 2]], [[4, 2, 6], [7, 0, 2]], [[3, 3, 6], [7, 0, 2]], [[2, 4, 6], [7, 0, 2]], [[0, 0, 2], [2, 4, 6]]]

As we understand our code, the minimum depth target to the target is the one returned by the code at the end of the algorithm.

According to `dictPath` , the minimal depth is 5. The depth of the last and correct state is 5, but counting the initial state as depth 1, it would be 6.

2 A* Algorithm

```
    def AStarSearch(self, currentState):

            # frontera: PriorityQueue filled with (f, g,
                estat)
            frontera = q.PriorityQueue()
```

```
frontera.put((self.h(currentState), 0,
   currentState))

# best cost for each node
evaluated = {str(currentState): 0}

# Inital value for dictPath
# DictPath to reconstruct the path
# father depth is -1
self.dictPath[str(currentState)] = (None, -1)
depthCurrentState = 0
self.listVisitedStates.append(currentState)

while not frontera.empty():
    # take the node with the lowest f value
    f, g, node = frontera.get()
    # Get the number of moves made, for
       calculations
    depthNode = self.dictPath[str(node)][1]+1

    if depthNode > 0:
        self.movePieces(currentState,
           depthCurrentState, node, depthNode)
        # Check if we have reached the goal
    if self.isCheckMate(node):
        #The current  g value will be the final
           depth
        print("Soluci  trobada")
        # Reconstruct the path from the initial
           state to the target
        self.reconstructPath(node, g)
        return self.pathToTarget

    # Process all the neighbors
    for move in self.getListNextStatesW(node):

        #new g value
        new_g = g + 1

        # Since we start all moves from the same
           position
        # h(n) will be the same for all nodes
        # Therefore, we only need to compare g(n)
        if str(move) not in evaluated or new_g <
           evaluated[str(move)]:

            #Once we now the node hasn't been
               visited, we calculate f
            new_f = new_g + self.h(move)
```

```
                    evaluated[str(move)] = new_g
                    frontera.put((new_f, new_g, move))
                    self.dictPath[str(move)] = (node,
                        depthNode)  # save the father
                        node and its depth

                    #Add to visited nodes
                    self.listVisitedStates.append(move)
            # update currentState and depthCurrentState
            currentState = node
            depthCurrentState = depthNode

        # If there is no solution
        return None
```

3 Changing Initial State

Once changed the initial King State ([7,5] to [7,7]) we can see that the algorithm does perform the same way, giving us a similar result and the same depth (which has sense due the type of movements the King can do).

A* move sequence:

[[[7, 0, 2], [7, 7, 6]], [[6, 6, 6], [7, 0, 2]], [[5, 5, 6], [7, 0, 2]], [[4, 4, 6], [7, 0, 2]], [[3, 3, 6], [7, 0, 2]], [[2, 4, 6], [7, 0, 2]], [[0, 0, 2], [2, 4, 6]]]

As you can see it did show the same number of movements just with a slightly diferent path