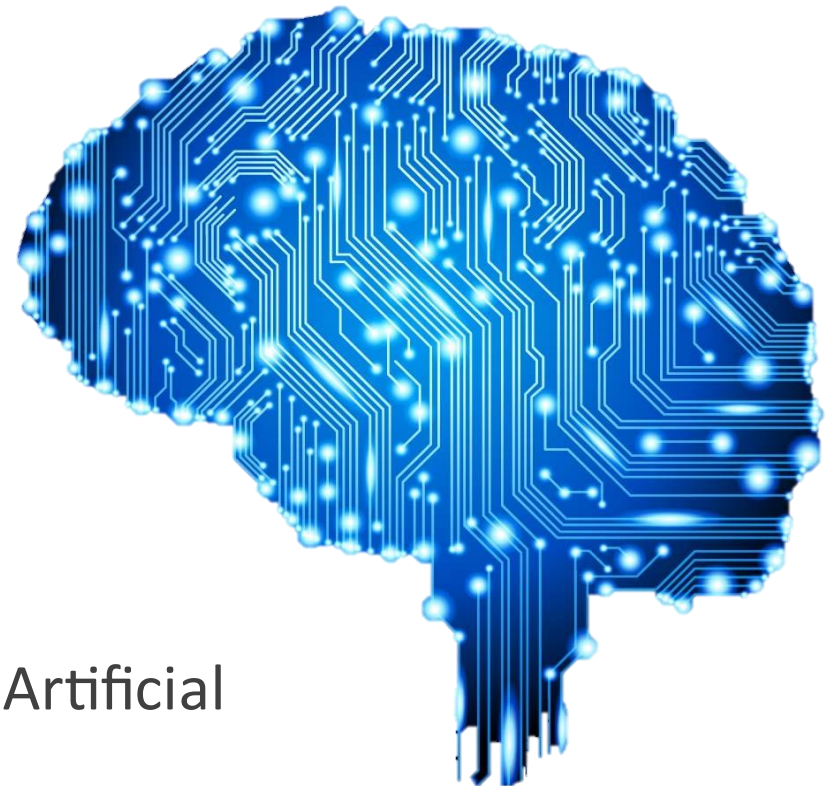
# Misioneros y Puzzle con AIMA



Inteligencia Artificial
2016/2017

Raúl Gil Fernández
Alberto Rodríguez – Rabadán Manzanares

# 1. Misioneros y caníbales: Demo no informada.

| | Path cost | Nodes expanded | Queue size | Max. queue size | Solution found |
|---|---|---|---|---|---|
| **Recursive DLS** | 0 | 1463 | - | - | No |
| **Iterative DLS** | 11 | 8119 | - | - | Yes |
| **Breadth first search** | 11 | 13 | 1 | 3 | Yes |
| **Uniform cost** | 11 | 14 | 0 | 3 | Yes |
| **Depth first search w/ graph** | 11 | 12 | 2 | 3 | Yes |
| **Depth first search w/ tree*** | - | - | - | - | No |

*Este algoritmo entra en bucle ya que no tiene control de ciclos.

A excepción del primer y último algoritmo, todos los demás han encontrado la solución en once pasos. Como al algoritmo de profundidad limitada recursivo sólo le dejamos llegar hasta profundidad 9 (valor por defecto), no conseguía alcanzar la solución. En cuanto al algoritmo primero en profundidad con árboles, se quedaba en bucle sin llegar a encontrar nunca ninguna solución.

Los DLS, son, de lejos, los que más nodos llegan a expandir de todos los algoritmos usados en comparación con el resto.

# 2. Misioneros y caníbales: demo informada.

| | Path cost | Nodes expanded | Queue size | Max. queue size | Solution found |
|---|---|---|---|---|---|
| **Greedy best first search (infinite boats heuristic) w/ graph** | 11 | 13 | 1 | 3 | Yes |
| **Greedy best first search (infinite boats heuristic) w/ tree** | 11 | 24 | 28 | 29 | Yes |

| | | | | | |
|---|---|---|---|---|---|
| **Greedy best first search (Non attack heuristic) w/ graph** | 11 | 12 | 2 | 3 | Yes |
| **Greedy best first search (Non attack heuristic) w/ tree** | 11 | 14 | 17 | 18 | Yes |
| **A* search (infinite boats heuristic) w/ graph** | 11 | 14 | 0 | 3 | Yes |
| **A* search (infinite boats heuristic) w/ tree** | 11 | 2428 | 2864 | 2865 | Yes |
| **A* search (non attack) w/ graph** | 11 | 13 | 1 | 3 | Yes |
| **A* search (non attack heuristic) w/ tree** | 11 | 48 | 55 | 56 | Yes |

## Heurística de no ataque (non attack)

Para el desarrollo de esta heurística, asumimos que los caníbales nunca atacarían a los hombres con lo cuál en cada viaje transportamos a dos personas independientemente de sin son caníbales o misioneros. La función nos quedaría del tipo: 2(# misioneros + # caníbales) – orilla
El valor de la orilla será 1 si está en la izquierda, 0 si está en la derecha.

## Heurística de barcos infinitos (infinite boats)

En esta heurística decidimos que nunca hubiera que hacer un viaje de vuelta para recoger al resto de personas, sino que cada vez que un pareja llegaba a la orilla contraria, otro barco estaría disponible en la otra orilla y otra de las parejas lo podría utilizar.

# 3. BlackAndWhite puzzle: demo no informada.

| | Path cost | Nodes expanded | Queue size | Max. queue size | Solution found |
|---|---|---|---|---|---|
| **Recursive DLS** | 14 | 141142 | - | - | Yes |
| **Iterative DLS** | 14 | 214838 | - | - | Yes |
| **Breadth first search** | 14 | 123 | 10 | 27 | Yes |
| **Uniform cost** | 14 | 133 | 3 | 28 | Yes |
| **Depth first search w/ graph** | 29 | 96 | 30 | 50 | Yes |
| **Depth first search w/ tree** | - | - | - | - | No |

En este caso y al igual que en el caso anterior, la búsqueda primero en profundidad no llega a encontrar la solución ya que se queda en un bucle. Como se puede observar, el coste del camino para el primero en profundidad es el doble que para el resto de algoritmos aunque expande menos nodos.

# 4. BlackAndWhite puzzle: demo informada.

| | Path cost | Nodes expanded | Queue size | Max. queue size | Solution found |
|---|---|---|---|---|---|
| **Greedy best first search (misplaced tile heuristic) w/ graph** | 18 | 17 | 27 | 28 | Yes |
| **Greedy best first search (misplaced tile heuristic) w/ tree** | - | - | - | - | No |
| **A* search (bad positioned heuristic) w/ graph** | 14 | 130 | 6 | 26 | Yes |

| | | | | | |
|---|---|---|---|---|---|
| **A* search (bad positioned heuristic) w/ tree** | 14 | 251343 | 838035 | 838036 | Yes |
| **A* search (misplaced tile heuristic) w/ graph** | 14 | 103 | 28 | 31 | Yes |
| **A* search (misplaced tile heuristic) w/ tree** | 14 | 19459 | 68088 | 68089 | Yes |

## Heurística de piezas descolocadas (misplaced tiles)

Esta heurística es simplemente el número de piezas descolocadas - 1

## Heurística de la mala posición final (bad positioned)

En esta heurística, para que las piezas blancas estén colocadas de forma correcta, sólo pueden ir de la posición 0 a la posición 3 consecutivamente y lo mismo para las negras (de la 4 a la 6). Por lo tanto se cuentan las fichas que estén fuera de ese rango