Artificial Intelligence

Report for 1st part of Project – "*Peg Solitaire*"

Heuristic

To help guide and speed up the informed searches, a **heuristic** is used to estimate how close a board state might me to a possible solution, which consists of the sum of three different heuristics. This way, the final heuristic is given by the formula:

$$h(n) = h_n(n) + h_c(n) + h_g(n)$$

The first heuristic, h_n , consists of the **number of pegs remaining** on the board.

The second heuristic, h_c , consists of the **number of pegs that are in *corner positions***. A *corner position* is a position that cannot be jumped over, which means it can only be captured by a move starting in itself.

The third heuristic, h_g , consists of the **number of different *peg groups***. A *peg group* is a group of adjacent pegs that can only be jumped over by pegs within the same group. Its definition relies on *Merson regions*, a region of the board in which, if all positions are occupied, every position can only be jumped over by a move starting in a position inside itself. In simpler terms, a *peg group* is a group of adjacent pegs that are surrounded by empty or blocked positions.

A heuristic is only admissible if it **never overestimates** the cost of reaching a possible solution. Which forces the conclusion that the heuristic used is **not admissible** by analysis of a simple scenario. In that scenario, in which there are only four pegs, all adjacent to each other and forming a square, and one of them in a *corner position*, the heuristic value would be 6, but the minimum number of moves to reach a solution is just 3. In this example, the heuristic used **overestimates** the cost of reaching a possible solution by 3. And because the heuristic is not admissible, it **cannot be consistent** either, which may result in a non-optimal solution, if there are many solutions possible.

Algorithms

There will be three different search algorithms used, which are ***Depth-First Search***, ***A* Search*** and ***Greedy Search***. While *A* Search* and *Greedy Search* are **informed search** algorithms (which means a heuristic is used to estimate how close a state is to a possible goal), *Depth-First Search* is not. The difference between *A* Search* and *Greedy Search* is that, while *Greedy Search* only considers the heuristic value, *A* Search* considers the sum of the heuristic value and the total cost from the start to the given node.

Formally, the algorithms used are **not complete**, because there are cases, such as the existence of cycles and/or states with infinite depth, in which they cannot find a solution. However, since the given problem is a game of "Peg Solitaire", there is the guarantee that it is not possible to have cycles or states with infinite depth, which makes the algorithms considered **complete in the scope of the problem**.

Algorithm	Time Complexity	Space Complexity
DFS Search	$O(b^m)$	$O(b \times m)$
A* Search	$O(b^{\Delta})$	$O(b^m)$ *
Greedy Search	$O(b^m)$	$O(b^m)$

Table 1: Comparison of the three algorithms used by their time and space complexities.

b = branching factor; m = maximum depth; Δ = heuristic error (difference between real cost and heuristic value)

* If the heuristic was perfect, the complexity would be $\mathrm{O(n)}$, where n would the dimension of the shortest path.

Result Analysis

Below are presented the empiric results, for **time** (in seconds), **generated nodes** and **nodes expanded**, obtained from running the three algorithms presented on all given boards. All the tests were done on the same system, with identical load. It is to be noted that the generated nodes do not include the first node.

Board	Algorithm	Time (s)	Generated Nodes	Nodes Expanded
11 Pegs 5x5 Board	DFS Search	$\approx 5.64 \times 10^{-4}$	21	14
	A* Search	$\approx 2.65 \times 10^{-3}$	23	18
	Greedy Search	$\approx 1.91 \times 10^{-3}$	19	14
14 Pegs 4x4 Board	DFS Search	$\approx 1.76 \times 10^{-1}$	7 943	7 919
	A* Search	$\approx 1.33 \times 10^{-2}$	156	129
	Greedy Search	$\approx 1.41 \times 10^{-2}$	156	133
16 Pegs 4x5 Board	DFS Search	≈ 3.04	106 278	106 249
	A* Search	$\approx 4.67 \times 10^{-3}$	59	16
	Greedy Search	$\approx 4.62 \times 10^{-3}$	59	16
20 Pegs 4x6 Board	DFS Search	≈ 980.70	32 689 061	32 689 018
	A* Search	$\approx 4.34 \times 10^{-1}$	2 523	2 337
	Greedy Search	≈ 20.17	155 639	155 554

Table 2: Results of the three searches on all given boards.

From analyzing the results, there comes the first conclusion, which is that the ***Depth-First Search*** algorithm **scales very poorly** comparing to the other algorithms used. This is to be expected, as this algorithm performs an **uninformed search**, whereas both other algorithms perform informed searches. However, in very small problems, such as the 5x5 board with 11 pegs, there is a big percentual gain to be had from *Depth-First Search*, even though it does not equate to much due to their size, being almost neglectable.

Comparing the ***A* Search*** **and** ***Greedy Search*** algorithms, there is **little to no difference** between them on **small to medium sized** problems. However, the *Greedy Search* algorithm starts to scale very poorly with bigger sized problems, such as the 4x6 board with 20 pegs, while the *A* Search* algorithm scales a lot better. This is to be expected because of the difference between *A* Search* and *Greedy Search* explained earlier, meaning that the ***A* Search*** **will be less affected by a non-optimal heuristic** than the *Greedy Search*, which relies exclusively on heuristic values. The results then lead to the conclusion that there is the possibility of the heuristic used not being optimal, meaning there are considerable gains to be had from improving it.

For this class of problems specifically, it is also verified that the **complexity of the problem relies** almost entirely **on the number of pegs** rather than on the size of the board. This makes sense, as **the number of moves required** to reach a possible solution, which **equates to the cost of a solution**, depends mostly on the number of pegs that need to be removed to reach a solution, which is a state in which there is only one peg left.

Teacher: *Manuel Cabido Lopes*

Group 27: *João Galinho* (87667) and *Filipe Henriques* (87653)