COMP30024 Artificial Intelligence Project Part A Report

Alexander Westcott 994344 & Cameron Chandler 993990

April 2020

Our implementation of Part A of the project uses code sourced from the Artificial Intelligence: A Modern Approach GitHub repository: github.com/aimacode/aima-python.

The Game as a Search Problem

The game has been formulated as a search problem considering each board configuration (position and height of black and white stacks) as a state. Each legal white move in a given board configuration is one of the actions that can be taken in that state, and will lead to another state. The goal test simply compares the current board configuration to an empty board. The path cost is one unit per action. In this way, a search algorithm has sufficient structure to search through game configurations until a solution is found, given a starting configuration.

The Algorithm

A* was chosen as the search algorithm because it is complete in this search space with finite nodes (checks were made for repeated states in the algorithm). Empirically, A* was found to be more time efficient in finding solutions than other algorithms like iterative deepening search. Our implementation of A* is not optimal as the heuristic we developed is not admissible. The chosen heuristic calculates the manhattan distance between all black pieces to their respective closest white piece, summing these distances. This is as closeness of white pieces to black pieces is typically desirable, although exceptions can be found such that the heuristic value is greater than the optimal solution from a state (consider one white piece adjacent to a line of black pieces that could all be removed with one boom). $h(node)$ is defined below:

$x_{1,i}$ is defined to be x coordinate of i^{th} black piece $i \in 1, \ldots, n$
$y_{1,i}$ is defined to be y coordinate of i^{th} black piece $i \in 1, \ldots, n$
$x_{2,j}$ is defined to be x coordinate of j^{th} white piece $j \in 1, \ldots, m$
$y_{2,j}$ is defined to be y coordinate of j^{th} white piece $j \in 1, \ldots, m$
Let S_i be the set of Manhattan distances from the i^{th} black piece to all white pieces.
$S_i = \{|x_{2,j} - x_{1,i}| + |y_{2,j} - y_{1,i}| : j \in 1, \ldots, m\}$

$$h(node) = \sum_{i=1}^{n} \min(S_i)$$

Time and Space Requirements

Let b be the branching factor, h be the heuristic function, h^* be the least path cost to a solution, $\varepsilon = \left(\frac{|h-h^*|}{h^*}\right)$ be the heuristic error and d be the depth of the least cost solution.

For A* search, many factors impact the time and space requirements. The time and space complexity for A* with an admissible heuristic is given by $O(b^{\varepsilon d})$ – they are the same as every node is stored in memory for the duration of the search. We see that the time and space requirements

depend on the branching factor, and exponentially on both the error in the heuristic and the depth of the optimal solution. The branching factor depends on the number of white pieces – increasing with more white pieces – and can vary slightly depending on the board configuration. e.g. how the pieces are stacked, distance to the edge of the board and black pieces in the way of movements. Generally, with an increase in the number of white pieces on the board, we see an increase in the time and space requirements of the search due to the increasing branching factor. It is worth noting that in reality the effective branching factor is much lower than b if a good heuristic is chosen, the heuristic will allow pruning of counterproductive branches and reducing the time and space complexity. As the heuristic gets further away from the true value of the remaining cost, the heuristic error increases, thus increasing the worst case time and space requirements. The value of d depends upon the board configuration – problems with solutions that can be found in few steps require exponentially less time and space than problems with solutions that can be found in at least more steps.

For a non-admissible heuristic, the big O upper bound on complexity is even higher, as we can go to a depth greater than d in the search – the algorithm is no longer optimal. However, this does not imply poor performance as solutions can still be found efficiently, just without the certainty of being optimal. Increases in d, b, and the error in the heuristic will nevertheless generally increase the time and space requirements of the algorithm.