Mini-Project 2 Report

Jordan Elijah Greene
jgreene88@gatech.edu

***Abstract*—I was tasked with creating an agent to solve the Block World Problem. I will describe how my agent approaches the problem, how well it performs in various categories, and how well it compares to a human.**

1 AGENT OVERVIEW

The agent has 32 lines of code. All the code used to solve the problem is inside Solve(). The only library used for this assignment is the copy module which I used to manipulate the state of the initial and goal stacks without interfering with them directly. The task of the agent is when given an arrangement of blocks in a series of stacks, rearrange them based on a goal state provided.

1.1 Dissecting The Agent

My agent's approach overall is slightly different from the lectures. It does not deploy Means-End Analysis, but it does use a type of Problem Reduction. For all cases regardless of the length of the initial or goal stacks, it would unstack all the blocks in the initial arrangement in a DFS manner. This means it unstacks one block at a time as opposed to unstacking one block for each stack across the entire initial arrangement. After it unstacks all the blocks, it reassembles each stack based on the arrangement of the goal state in a bottom-up approach.

Overall, the agent has a time complexity of $O(n^2)$. It loops through the entirety of the initial arrangement and then loops through the entirety of the goal arrangement.

In terms of the time taken to complete each task, a graph below describes it best.

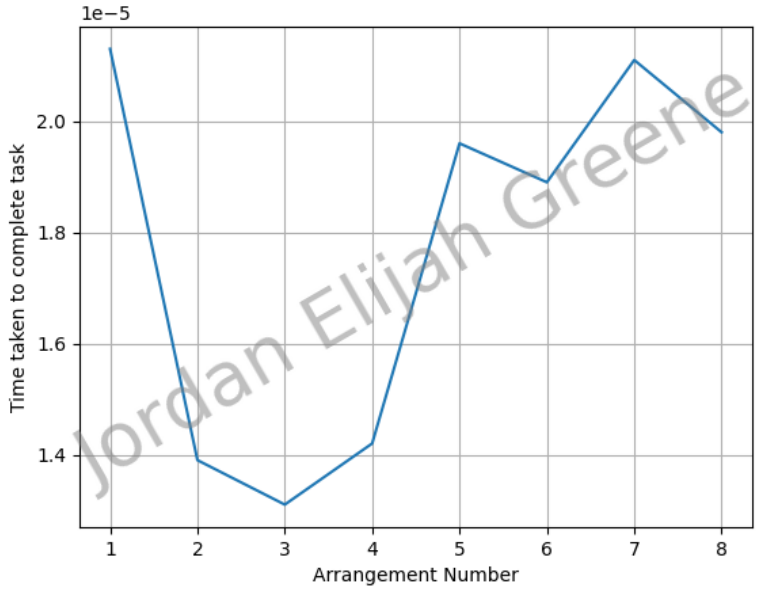

Figure 1— A Graph of the performance of the agent. The y-axis represents time in, what is close to, microsecond. The x-axis represents the arrangement number numbers as provided in the base code for the min-project.

1.2 Performance Analysis

The graph, as one can see, fluctuates greatly throughout. However, this is primarily due to how the blocks are arranged. If we look at the arrangement numbers where the goal arrangement is just one stack (arrangement numbers 2, 3, 5, and 6), they have close to the same execution times when compared to its nearest neighbor.

Arrangement numbers 2 and 3 have small differences in execution times because their goal states are similar: one stack of blocks the length of which is 5 elements long. Arrangement numbers 5 and 6 also have similar execution times but take longer than arrangements 2 and 3. This is due in part to there being only one stack in the goal arrangement with a length of 9 elements. In both cases where the blocks are stacked in reverse order (arrangement numbers 3 and 6) it takes less time to execute it when compared to its neighbors (arrangement numbers 2 and 5) which have a block stack in regular order.

The noticeable jumps in the graph for time taken to execute (arrangement numbers 1, 7, and 8), come from the fact that there is more than one goal stack. The agent must loop through each individual stack to disassemble them and then again to reassemble them. Interestingly, arrangement number 4 has a much better execution time than arrangement number 1 despite both having the same number of goal stacks. What makes this truly bizarre is that the ordering of the blocks should have no impact on the execution time. Because of this, I decided to graph another aspect of the agent's performance: the number of moves it made in each arrangement number.

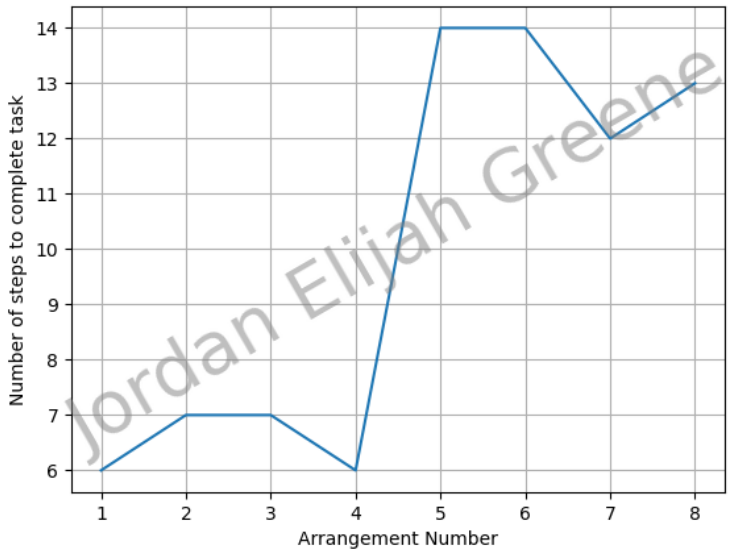

Figure 2 — A Graph of the number of steps taken to complete each arrangement.

This, sadly, did not shed any new light onto this little issue. Paradoxically, the number of moves needed to perform the task did not consistently correlate with the time taken to execute it. Using a previous example, arrangement number 4 performed the same number of steps as arrangement 1 yet took more time to execute.

2 THE AGENT AND HUMAN COMPARISON

The approach of unstacking all the blocks and then re-stacking them is something a human would do if the number of blocks involved became arbitrarily large

because we humans can only keep track of so many things. However, given a small number of blocks to work with, a human would be able to keep track of available pieces and be able to make a new stack while disassembling the initial arrangement. This agent does not do that. The agent unstacks all blocks regardless of the number of blocks it's working with. In truth, this was my first instinct when I was tackling this problem.

However, after reading the instructions I thought that my approach was not sufficient for the task at hand. I believed the agent needed to be more human-like in its approach. I divided the Solve method into two parts. I then created sets and dictionaries to keep track of the positions of the blocks. But all of this was for not. For every test it would pass it would run into many problems. The more problems it ran into, the more solutions it required. Implementing those solutions became arduous due to the complexity and depth of the code I made. In the end, I implemented my original approach of unstacking the entire initial arrangement and then building it back up from bottom to top. In the end, this was the cleverest approach the agent could do since this makes every arrangement solvable in the most optimal number of moves.

The lesson I learned from this min-project is to go with your gut instincts first for problems like these. Your gut instinct may be right.