Mini Project 2 – Block World Problem

Michael Lukacsko
mlukacsko3@gatech.edu

1 AGENT EVALUATION

The agent in this problem employs a Greedy Search algorithm to solve the Block World problem of transforming an initial arrangement of blocks into a goal arrangement. This approach works by selecting moves that are most immediately beneficial based on a heuristic function, specifically the number of blocks that are already in the correct position relative to the goal state. The key principle behind Greedy Search is to make locally optimal decisions at each step, continuously choosing the move that most reduces the difference between the current and desired arrangements. For this problem, the agent is designed to minimize the "difference" by focusing on how many blocks are already stacked correctly and moving blocks that appear out of place to more appropriate positions, either on to other stacks or onto the table. To accomplish this, at each iteration, the agent generates potential moves by evaluating two key options: moving a block from one stack to another or moving a block to or from the table. This approach allows the agent to explore a variety of paths to the goal without fully committing to an exhaustive search of all possibilities, as some other algorithms might. Instead, it relies on the heuristic function to guide it, ensuring that each move incrementally brings the current state closer to the goal.

The choice of using Greedy Search to solve the Block World problem is particularly well-suited because the problem has a straightforward structure, where the goal is clearly defined by the arrangement of blocks. The heuristic function used (the count of blocks already in the correct position) is simple, effective, and easy to compute. This allows the agent to avoid unnecessary computation and focus on progressively making progress toward the goal. The problem itself benefits from this heuristic-driven approach, as it doesn't require backtracking or reconsidering moves, which occurs often when employing Generate & Test and/or Means-Ends Analysis.

2 AGENT EVALUATION – PERFORMANCE

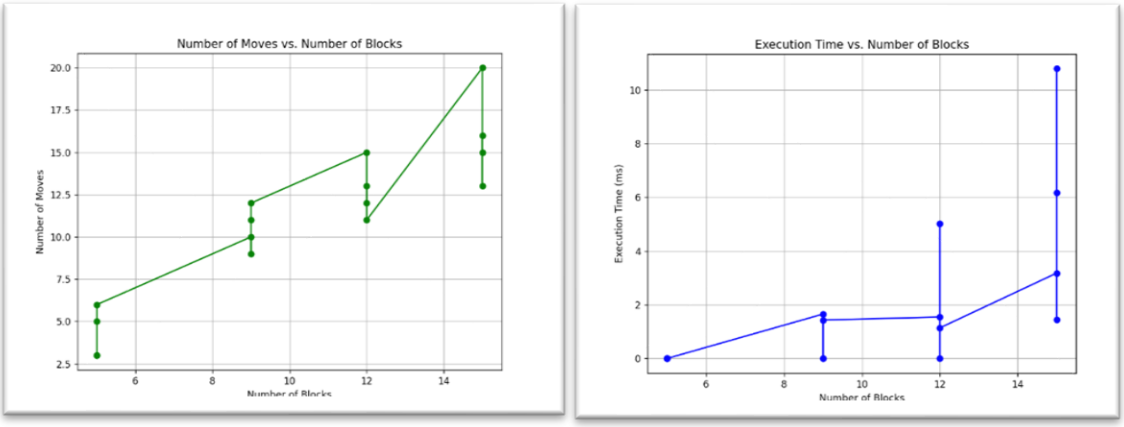

Figure 1 — Agent evaluation. Left -Number of Moves vs Number of Blocks. Right – Execution Time vs Number of Blocks

Referencing the left image in figure 1 above, the agent's performance generally follows an increasing trend as the number of blocks grows, indicating that more moves are required for larger block configurations. To evaluate the agent's performance, testing was performed using initial arrangements of 5 blocks, 9 blocks, 12 blocks, and 15 blocks, with each initial arrangement tested against four goal arrangements of varying complexity. For smaller block setups, such as the 5-block configuration, the agent is highly efficient, needing only a few moves to reach the goal. However, as the number of blocks increases, especially beyond 10, the number of moves required grows significantly. This suggests that the complexity of solving the problem increases non-linearly with the number of blocks. As such, it can be concluded that the greedy search algorithm employed by the agent performs well for smaller and moderately sized configurations, as it is able to quickly identify promising moves that reduce the difference between the current and goal states. However, for very large initial arrangements, such as those with 25 or more blocks, it is possible the greedy search may encounter challenges, as it can select locally optimal moves that seem beneficial in the short term but lead to suboptimal solutions in the long run due to local maxima. In such cases, the greedy approach might struggle to find the globally optimal solution, resulting in more moves than necessary or even failure to reach the optimal solution.

3 AGENT EVALUATION – EFFICIENCY

The image on the right in figure 1 above shows the relationship between execution time and the number of blocks, highlighting how the agent's efficiency changes as the problem size increases. For smaller block configurations, such as 5 or 9 blocks, the agent performs very efficiently, with execution times close to 0 milliseconds. As the number of blocks increases to 12 and 15, the execution time starts to rise for more complex goal arrangements, though remaining within acceptable limits for moderately sized configurations. This indicates that the agent scales reasonably well with moderate block sizes, maintaining quick runtimes for most cases.

However, the non-linear increase in execution time suggests that the algorithm's complexity grows as the number of blocks increases. The observed increase in execution time can be attributed to the fact that the agent must consider more possible moves and configurations as the number of blocks increases. The algorithm follows a greedy search, which typically has a time complexity of **O(n^2)** in this context, as each block can potentially be moved to multiple locations, requiring repeated evaluations of the block positions to find the optimal moves. While this approach works efficiently for smaller problem sizes, the execution time increases more significantly as the number of blocks grows, suggesting that for even larger block configurations, the agent's efficiency may degrade further. This non-linear growth implies that the agent could struggle with inputs greater than 15 blocks, and more optimized algorithms or heuristics might be necessary to maintain efficiency in such cases.

4 SEARCH STRATEGY AND HEURISTIC OPTIMIZATION

By employing a greedy search algorithm, the agent makes locally optimal decisions at each step, aiming to quickly reduce the difference between the current and goal arrangements. This approach minimizes backtracking and avoids redundant evaluations of the same block positions, allowing the agent to efficiently reach a solution. Three clever optimizations enhance the efficiency of this greedy search:

1. The agent uses a difference function to measure the distance between the current state and the goal state. This function prioritizes blocks that are already correctly placed and uses this information to guide the search,

significantly narrowing the search space and consequently improving efficiency.

2. The agent avoids expensive deep copies of states by utilizing shallow copies and directly manipulating block positions within stacks. This approach reduces the overhead associated with managing large data structures, especially when dealing with higher block counts.

3. The agent's heuristic focuses on reducing the number of misplaced blocks. It evaluates multiple possible moves, whether placing a block on another stack or moving it to the table, and selects the move that results in the greatest improvement. This ensures that the agent consistently makes progress and avoids moves that do not directly contribute to solving the problem.

5 AGENT VS HUMAN COMPARISON

In comparison to a human, the agent follows a more systematic and consistent approach, leveraging a greedy search algorithm to guide its decisions. While a human might rely on intuition or experience to solve the block-world problem, the agent methodically evaluates each move based on a heuristic that calculates the number of misplaced blocks and attempts to reduce this number at every step. This means that the agent might approach the problem with a focus on efficiency, aiming to minimize the number of moves, but it lacks the flexible reasoning or strategic foresight that a human could apply, especially in complex scenarios.

As an example, a human may look ahead and foresee moves that might create problems later in the process, avoiding certain short-term decisions that might lead to more complex situations down the line. The agent, however, being guided by a greedy algorithm, makes locally optimal moves without this foresight, which can sometimes result in getting stuck in suboptimal configurations or local maxima. In this regard, the agent does not entirely solve the problem the same way a human would, as it focuses strictly on immediate gains rather than long-term strategy.