Project 1: Moving Blocks

Markos Meytarjyan, David Sawires, Omar Al-Taie
St.Olaf College
Meytar1@stolaf.edu, Sawire1@stolaf.edu, Altaie2@stolaf.edu

Abstract

The Blocks World is a classic problem in artificial intelligence involving a set of blocks and a set of actions that can be performed on these blocks. In this project, we propose an AI system that tackles the Blocks World problem by generating a sequence of scenes (states) that transition from an initial configuration of blocks to a desired goal configuration. Our algorithm is designed to handle any combination of five blocks (A, B, C, D, E) and their relations (AIR, ON, CLEAR, TABLE), allowing for a flexible and versatile approach to solving the problem. Our algorithm also attempts to solve the case if there is a triangle instead of a block shape. Attempt to solve the problem with efficient time and space complexity.

Introduction

"Blocks world is a model domain used in artificial intelligence to explore different approaches to automated reasoning—especially heuristic search and planning" (Chenowet, 1991). The Blocks World is a simplified representation of a physical world where blocks can be stacked, unstacked, and moved. This problem serves as an ideal testing ground for automated planning and problem-solving in AI. The key objective of this project is to develop an algorithm that can autonomously plan and execute a sequence of actions to transition from an arbitrary initial state to a user-specified goal state.

Related Work

The work of Slaney J, Thiébaux S, Blocks World Revisited (Slaney, Thiébaux, 2001). Their work also toggles the famous AI problem of Blocks World, yet the problem is solved in a different approach. The authors explore elements of uncertainty, sensing, and action costs, while exploring how such elements could be integrated with Blocks World problem. The work discusses optimal plans, misplaced and constructive blocks, deadlocks, as well as strategies for Near-Optimal Planning (Unstack-Stack, GN1, GN2). The paper provides valuable insights into the Blocks World planning domain and the development of efficient planning algorithms within this context.

Approach

The algorithm for automated planning in the Blocks World problem comprises several key components and steps, as described below

State Representation

The algorithm begins by representing the state of the Blocks World using a logical representation. Each block in the world is associated with specific attributes, including its current location, the block it is resting on, and its clearance status. Relations among the blocks are represented using predicates. The initial state is provided as input, containing the current configuration of blocks.

Operator Definitions

The algorithm defines a set of operators, representing actions that can be performed on the blocks. These operators include pickup(block1), putdown(block1), unstack(block1, block2), stack(block1, block2), move().

Main Algorithm

The algorithm uses dictionaries and a list to manipulate the boxes and achieve the change from the initial form to the goal form. The 'table' is initialized in the dictionary with an empty value, then using a for loop, it iterates through the elements of the goal state. For each block, it builds the dictionaries representing the bottom up relations of the goal state blocks and another dictionary representing top down relations of the goal state blocks. The code enters a **while** loop to move blocks to the table. The loop continues as long as there is any movement (i.e., **moved** is **True**). Inside the

loop, it iterates through the blocks in the initial state. For each block, it checks if the block is clear (not supporting any other block) and if it is not on a block of type 3 (The Table). If the conditions are met, it generates a set of actions, such as "unstack" and "putdown," and performs these actions on the blocks. It also updates the dictionary and prints the state of the state after each action. The **moved** flag is set to **True** if any block was moved during the iteration. The loop continues until no more blocks can be moved onto the table. The next part of the algorithm uses the other dictionary to build each tower in the goal state. It starts from the list of blocks on the table in the goal states and then iterates through each block above it, picking up the block from the table and stacking it on top of the current tower. Once there are no more blocks to stack on the tower, it moves onto the next block on the table, repeating the previous steps until all “towers” are built. The algorithm displays the generated sequence of scenes to provide a visual representation of the planning process. This visualization allows users to observe the step-by-step progress from the initial state to the goal state.

Results

The outcome of this project is a search algorithm that can effectively solve the Blocks World problem by generating and displaying a visually interpretable sequence of scenes that represents the transition from the initial state to the goal state. This project shows one way of how to solve the problem, which uses dictionaries, lists, and swapping the blocks using Air, Table, Clear, and On.

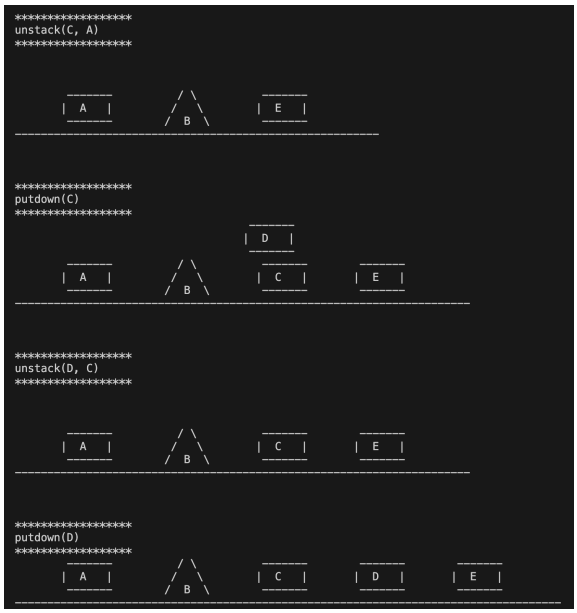

Figure: Printed Terminal Output

The algorithm essentially uses series of unstack and putdown operations to put down all the blocks on the table. In the worst-case scenario this has O(n) time complexity where n is the number of blocks since we may have to move n-1 blocks on the table. Afterwards we use dictionaries to stack the blocks on top of each other as it is in the goal state. In the worst case we must move n-1 the blocks to their right position making the total time complexity of the algorithm O(n) + O(n) which makes up for total of O(n) time complexity. A drawback to this algorithm is the fact that there is no assessment of how close is the initial state to the goal state. For instance, in a particular scenario where only one move could solve the problem this algorithm might go over that fact and unstack all the block on the table. This is by far not the most optimal algorithm to solve the problem, however the implementation of two dictionaries makes the process of stacking all the blocks to achieve the goal state quite optimal.

Conclusion

The search algorithm proved to work with a few different inputs of initial and goal blocks. However, due to the nature of the algorithm, where it stacks the blocks one by one after looping and adding each of the blocks in a table. This method would be more time complicated and inefficient with more inputs. An example of other research which utilizes better time complex algorithms could be seen in Slaney’s and Thiébaux’s work of Artificial Intelligence: Blocks World Revisited (Slaney’s and Thiébaux’s, 2001). Their work revolves around modifying The GN1 algorithm to handle the Blocks World problem by incorporating the minimal set of deadlocks as a parameter. The algorithm is efficient for most of the sub-problems. Hence, the algorithm used in this research can have improvements regarding time complexity as the number of inputs increase.

References

Chenoweth SV (1991)In: ON THE NP-HARDNESS OF BLOCKS WORLD . https://cdn.aaai.org/AAAI/1991/AAAI91-097.pdf. Accessed: 2023-October-30th

Slaney J, Thiébaux S (2001)In: Blocks World revisited. https://www.sciencedirect.com/science/article/pii/S0004370200000795