CS7637 Mini Project 2: Block World

Alyssa Woo
awoo34@gatech.edu

***Abstract*—The "Block World" problem starts with an arbitrary initial arrangement of unique blocks and returns a list of moves required to transform the initial arrangement to a goal arrangement. An AI Agent was designed to make use of a Generate and Test method, along with a Means Ends Analysis (MEA) with Problem Reduction approach to help vastly decrease the state space. The generator determines valid next states that could result from the current state, and the tester makes use of a breadth-first search (BFS) algorithm to find the first instance of reaching the target state to find the optimal solution.**

1 THE AI AGENT: HOW IT WORKS

The AI Agent makes use of the MEA and Problem Reduction to help reduce the state space as it uses a Generate and Test approach.

1.1 Problem Reduction

The Agent reduces the large given problem into smaller mini goal states to work towards before moving on to the next larger goal. The Agent works one level at a time across all stacks in the goal arrangement, adding one more block at a time with each mini goal state. Figure 1 displays an example of how Problem Reduction was used to determine what the mini goal states should be as the Agent solved the problem.

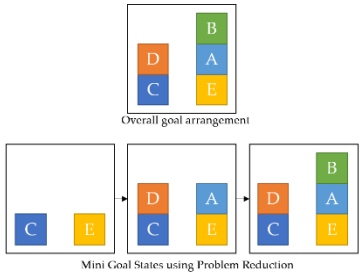

***Figure 1*—Agent used Problem Reduction to work on one level of blocks at a time across all stacks in goal arrangement**

1.2 Breadth First Search

The Agent creates its initial Graph node with the state of the initial arrangement. For each mini goal state generated from the Problem Reduction section, the Agent performs BFS on the current node in the graph to achieve that mini goal state. The Agent generates its graph of states during the BFS. At the Agent's current state, if the current state matches the target state, then it stops the BFS. Otherwise, the Agent calls its Generator on the current state, which determines the valid next states that could result (in other words, the Generator is generating the child nodes of the current node in the graph). Details regarding how the Generator determines the valid next states are described in section 1.3. Then, the next states that resulted from the Generator are strategically added to the BFS state queue. The Tester checks to see if the generated state has a smaller distance towards the goal compared to the current state. If it makes progress, the Agent adds that state to the front of the BFS state queue. As soon as the Agent finds a state that makes forward progress, it clears the queue except for the generated state, so that the BFS visits that state next. Otherwise, if no progress is made in terms of reducing distance to the goal state, the Agent does one of two actions:

1. The Agent keeps track of *goal blocks*, meaning the last block in the stack of the mini goal, so that it knows which blocks to try to free up. For example, in Figure 1, with the first mini goal state, the goal blocks were C and E; in the second mini goal state, the goal blocks were D and A, and so on. If in the generated state, a block is moved to access a block in a stack containing a goal block, the Agent adds this state to the front of the BFS state queue to prioritize it.
2. Otherwise, if no progress is made and the state does not help free up a goal block, the Agent adds the state to the back of the BFS state queue.

The Agent must also keep track of a list of moves, which are tuples (Block, Move Position). This list of moves is initialized to be empty. When the Agent finds the goal state, by the time it exits the BFS, the current node in the graph should be the first instance of the goal state that the Agent saw. The Agent then traverses backwards to determine what move was required to get from the previous state to the current state.

1.3 Generator

The Generator generates the list of possible states based on the current state and adds those next states to the list of children in the current state's graph node. The possible blocks that could move are the blocks at the top of each stack in the arrangement. If a block is on the table and not one of the goal blocks, the Generator leaves that block alone for now. There are two types of moves the Generator could do when generating the next state:

1. Move the block on top of another block. This is done only if positive progress is made (reduced distance) towards achieving the mini goal.
2. Move the block to the floor. This can be done if positive progress is made or no change in progress is made towards achieving the mini goal.

2 AGENT PERFORMANCE

The Agent performs well in that it often finds the optimal solution but is not guaranteed to find the minimum number of moves, depending on the problem. An example of a problem where the Agent does not find the optimal solution is in Figure 2. The Agent does not make use of knowledge of future goal states, making all its decisions based on making progress towards the current mini goal state. As a result, the Agent makes an extra move of placing B on the table (highlighted in red) rather than directly on E while solving Mini Goal State 2 because it has no knowledge of where B is supposed to go in the future states.

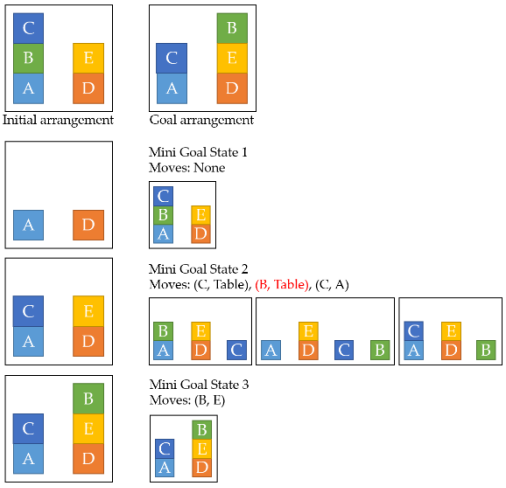

Figure 2 — Example of a problem where Agent does not find optimal solution. The Agent prefers to place blocks on the table if not immediately relevant to the current mini goal state.

3 AGENT EFFICIENCY

When generating states, for each block at the top of a stack, the Agent must check through S stacks to see if placing that block on top of another stack would make progress. Therefore, the generate function operates in $O(S^2)$ time where S is the number of stacks in the goal arrangement. In the worst case, the generate function may need to be called B times where B is the number of blocks in a stack (if the block the Agent searching for is at the bottom of the stack). Lastly, the Agent must do this for N blocks where N is the total number of blocks in the arrangement. Therefore, the overall time complexity for finding a solution is $O(NBS^2)$.

4 OPTIMIZING THE AGENT

The Agent has some intelligence in the Generator, as described in section 1.3. In general, the Agent places blocks on the table to get them out of the way, unless placing the block on top of another block makes progress towards the current mini goal state. By doing this, the Agent never moves a block more than two times. Moving any one block more than twice would be sub-optimal, and the Agent avoids this.

5 AGENT COMPARISON TO HUMAN PERFORMANCE

The Agent generally uses Problem Reduction to define mini goals from the bottom of the stacks upwards just as a human would. The Agent also prefers to move blocks to the table unless moving the block on top of another block achieves progress towards the mini goal, just as a human would. However, the Agent an solve the problems much more quickly than a human can in most cases. In some cases, with the maximum number of blocks and multiple stacks, a human may be able to solve the problem more quickly than the Agent, as the state space could get relatively large.