Mini-Project 2:
Spring 2022

Casey Hirschmann
chirschmann3@gatech.edu

1 HOW IT WORKS

This agent uses a combination of breadth-first search (BFS), generate and test, and means-ends analysis (MEA).

To initiate BFS, a class of nodes is created which contains each node's block configuration, action taken to achieve that configuration, parent node, and delta to the goal similar to what is accomplished in the lecture by Yu & Malan (Yu & Malan, n.d.). Next, a frontier was initiated which these nodes could be added to. Node states would be plucked from the frontier in the order they were added, then checked against the goal arrangement, added to a set of explored nodes, and sent through the generator. The generator would perform all possible moves, as well as calculate the delta between each generated state and the current subgoal. The deltas were compared to the current minimum subgoal delta, and anything with a delta higher than the minimum was deleted from the outputs. Finally, the states were checked to ensure they weren't already in the explored states or in the queue in the frontier before being added to the frontier.

The generator takes in the current node state and current subgoal arrangement and outputs all possible new configurations, their associated moves, and the calculated deltas for each state and the inputted subgoal. Note "possible configurations" include putting something on the floor if not already there or stacking on another block only if it furthers the goal (i.e. matches any of the configurations for the overall goal). The "test" portion happens after this where anything with a delta too high or any repeat states are removed.

MEA is performed by using subgoals. The subgoals start from the blocks on the floor and move up each layer until all subgoals are satisfied. Based on the nature of BFS, a node may generate states that accomplish the goal state, however, there are still nodes in the queue that need to have states generated and tested against the subgoal before moving to the next subgoal as seen in Figure 1. To account for this, the subgoal wouldn't be updated until the agent recognized a node with a

delta = 0. This would indicate that any future nodes would have a delta = 0 since states with a delta less than that wouldn't be added to the frontier. From here, it would update the subgoal and reset all deltas to begin the comparison again.

Once all subgoals are met, the frontier is explored until a node matching the goal arrangement is found. From here, the move is stored, and the parent node is retrieved to gather the move to accomplish that node. The process is repeated until the root node is reached to return the full sequence of moves.

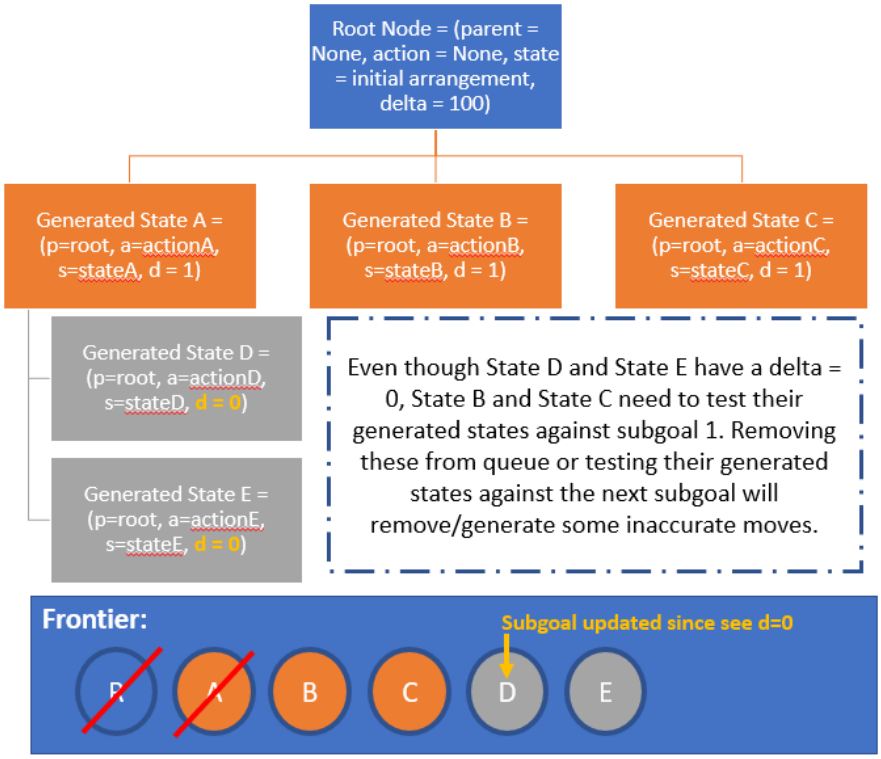

Figure 1—Visual of how nodes are generated and appear in the queue. One can see how there are still nodes from the same level in the queue that must be checked against subgoal 1 before proceeding to subgoal 2.

2 HOW IT PERFORMS

Originally, the agent was generating states where blocks are stacked on each stack. This performed well until you passed a solution that requires greater than about 11 moves to solve. From that point, the performance degraded exponentially.

To deal with this, a "Hail Mary" code was written. A counter was added that tracked the number of nodes explored. When that passed 2000 nodes, the current approach was diverted to the Hail Mary approach which would resort to creating a list of moves that placed all of the blocks on the floor and stacked them into the goal arrangement. This ensured an accurate solution was reached, albeit, the solution was not the optimal one.

However, after further contemplation of the problem, it was determined that generating moves stacking a block on each other stack was often counterintuitive to the problem and led to an exponentially increasing frontier, thus the generator was altered to only stack blocks that matched the goal arrangement.

3 EFFICIENCY

The efficiency certainly decreases as the problems increase in complexity.

As seen in Figure 2, the performance is related to but not solely based on the number of blocks as some cases see a decreased time requirement when more blocks are added.

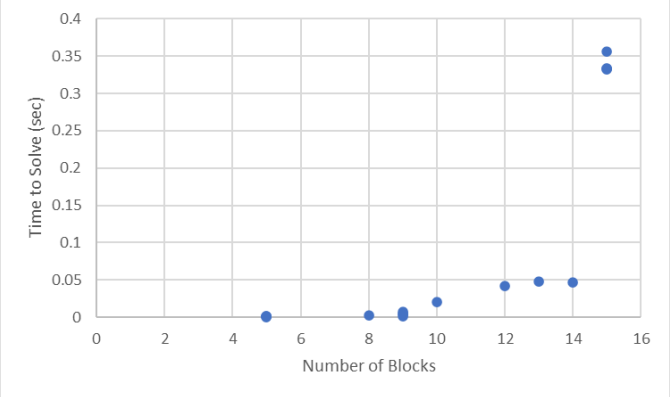

Figure 2—Plot of no. of blocks vs time to solve random cases of various initial block numbers, configurations, and goal configurations.

Upon investigating a number of things, the efficiency relationship can be expressed as follows:

$$efficiency = f(no.\ blocks,\ no.\ initial\ stacks,\ no.\ goal\ stacks)$$

Another big impact for this agent is how "buried" in a stack a "floor"block is. For example, a stack ['A','O','D','B','Z','Y','J'] that has a goal of ['O','B',Z'], ['A'], ['Y',

'D','J'] will perform much worse since essentially any state generated will have the same delta until the blocks are unstacked to 'O'. This leads to an exponential growth of the frontier. One way to improve this is not to test the subgoals as a unit (['O'], ['A'], ['Y'] from the example above) but rather as individual stacks. For this example, this means 'D' being stacked on 'Y' would decrease the delta even though the ['O'] portion of the goal hasn't yet been satisfied. However, this would still only have a minor impact on the number of cases on the frontier depending on how "deep" the floor blocks are in a stack.

4 CLEVER ASPECTS

The "cleverest" thing this agent does is the BFS storing parent information in the node similar to a Russian doll. This associates each node with its own sequence at arriving there without having to maintain a list of moves and states associated with each generated state.

The other clever element is the delta minimum test. This quickly prunes any nodes that aren't "competitive" when trying to accomplish a subgoal. Without this pruning, the frontier would exponentially increase. By maintaining this delta minimum, subsequent nodes on the same level of the tree are still judged against the best performing generated node rather than just the nodes that were generated from that parent.

5 HOW IT COMPARES TO HUMANS

The agent solves the problem very similarly to how a human would. A human would likely follow a BFS approach where an initial arrangement generates possible arrangements and then each of those arrangements will have their potential arrangements generated before moving forward. For each "level" of the tree, the arrangements will have a delta calculated and then compared. Any non-competitive states (above minimum delta) or repeat states would be eliminated. The agent follows this same idea.

The only thing a human might do differently is perceive what states head in the goal direction and start chasing those paths rather than continuing to generate the nodes in the BFS order.

6 APPENDICES

1. Yu, B., & Malan, D. J. (n.d.). *Week 0*. CS50's Introduction to Artificial Intelligence with Python. Retrieved February 9, 2022, from https://cs50.harvard.edu/ai/2020/weeks/0/