Mini Project 2

Bennett Buchanan
bbuchanan9@gatech.edu

1 MINI PROJECT 2

1.1 How does your agent work?

My agent uses a combination of techniques including Generate and Test, and Means-Ends Analysis. Additionally it uses breadth-first search and a class-based approach for easier debugging and management of the generated states and moves. At a high-level, the agent worked in the following manner:

1. Initiate a generator using the initial arrangement and goal arrangement.
2. From there, pass the generator into a breadth-first search function. For each iteration, the generator generates all possible next states, each with an associated move that is used to move into that state. The generated states are stored for use later by the tester.
3. Next, generate all the successive states of the current generated states, i.e. one level below the current depth in the tree of states (see Figure 1).
4. At this stage, we have a history of states that have occurred and have also generated all possible states for the current level. The tester is then used to eliminate any generated states that have occurred before to avoid loops in the state traversal (and we know, in any case, this is a non-productive move).
5. Finally, the agent uses Means-End Analysis to compare the generated state with the goal state and find the differences between them: the delta between the generated state and the goal state is calculated. If some generated states have a smaller delta than others, those states are chosen as the next states to traverse on.
6. Additionally, the states that were generated for the level below the current level are compared. From those states, the state that has the smallest delta is chosen. This effectively allows for choosing a path from a generator set where the delta does not move us closer to the final goal—by traversing one below and then comparing all successive states, the agent uses problem reduction (see Figure 2).

7. When a state is arrived at where the delta is zero, then the breadth-first search is ended and the path is returned as the optimal solution.

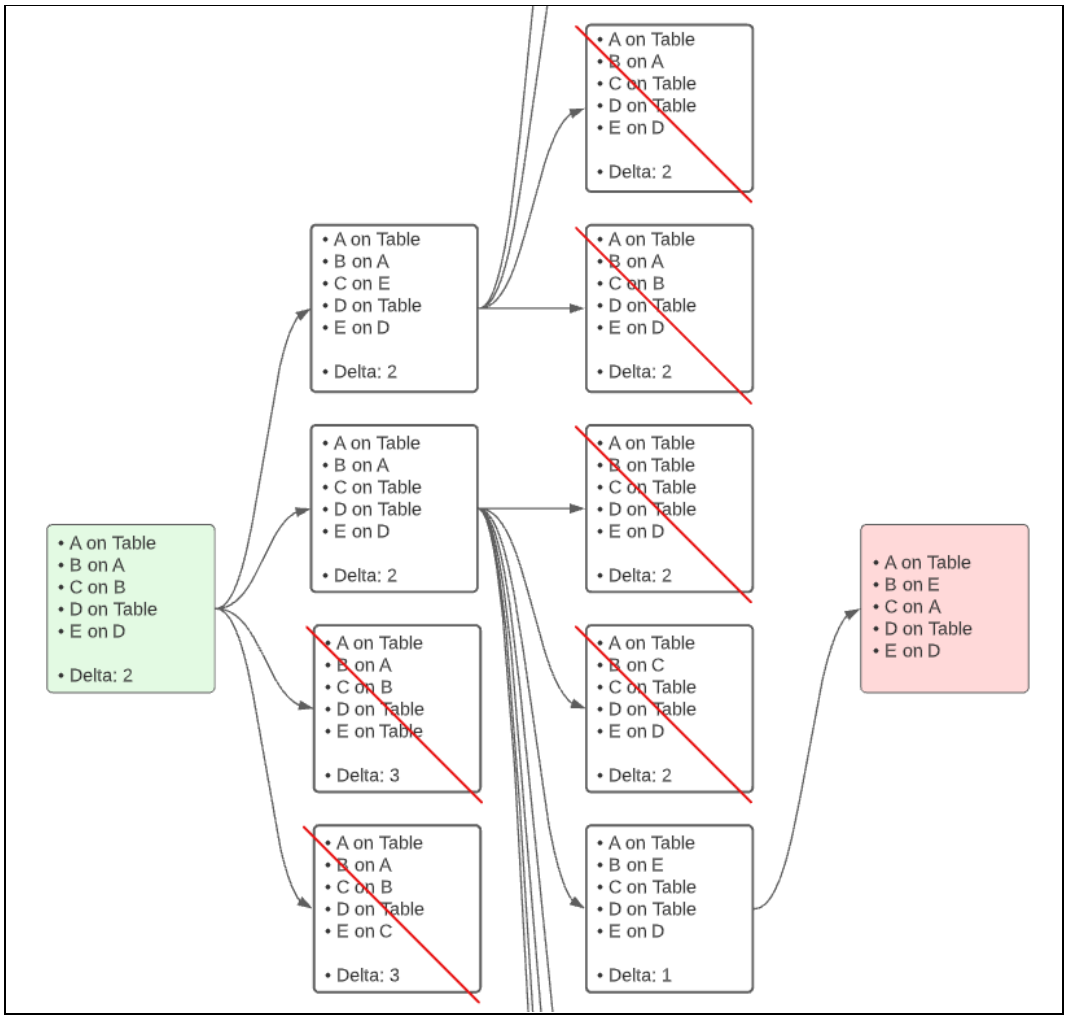

Figure 1—Depiction of the generated and tested states when the agent traverses at a depth of 4. States are eliminated either if they have occurred before or if the delta does not move the agent towards the goal state relative to the other states on that level.

2 How well does your agent perform? Does it struggle in any particular cases?

The agent performs well when the delta is a simple heuristic to select paths to take in the traversal. The tester eliminates a large number of states based on that heuristic and the heuristic for eliminating states that have occurred before. Due to these eliminated states, the traversal does not need to search across many states and backtrack. It struggles, however, in cases where the smallest delta

states are also those that have occurred before. In this case, those generated states are not productive so the smallest delta heuristic does not always work. To solve this the agent needs to iteratively test states that have an increasingly greater delta until a state is found that allows for progress.

1.3 How efficient is your agent?

The agent is efficient because it is able to eliminate many states. Thus even with the number of blocks increasing, for each traversal the number of states that have been generated also increases rapidly such that the eliminated states also grows to help prevent a combinatorial explosion of state growth. The process by which the agent compares successive states against each other also avoids unproductive traversals.

1.4 Does your agent do anything clever to arrive at an answer more efficiently?

At each level in the traversal the agent looks at all immediate children of the generated states and compares the delta of all those states to help eliminate unproductive moves. In this way, the agent performs problem reduction at each level of the traversal to avoid traversal on many non-productive paths.

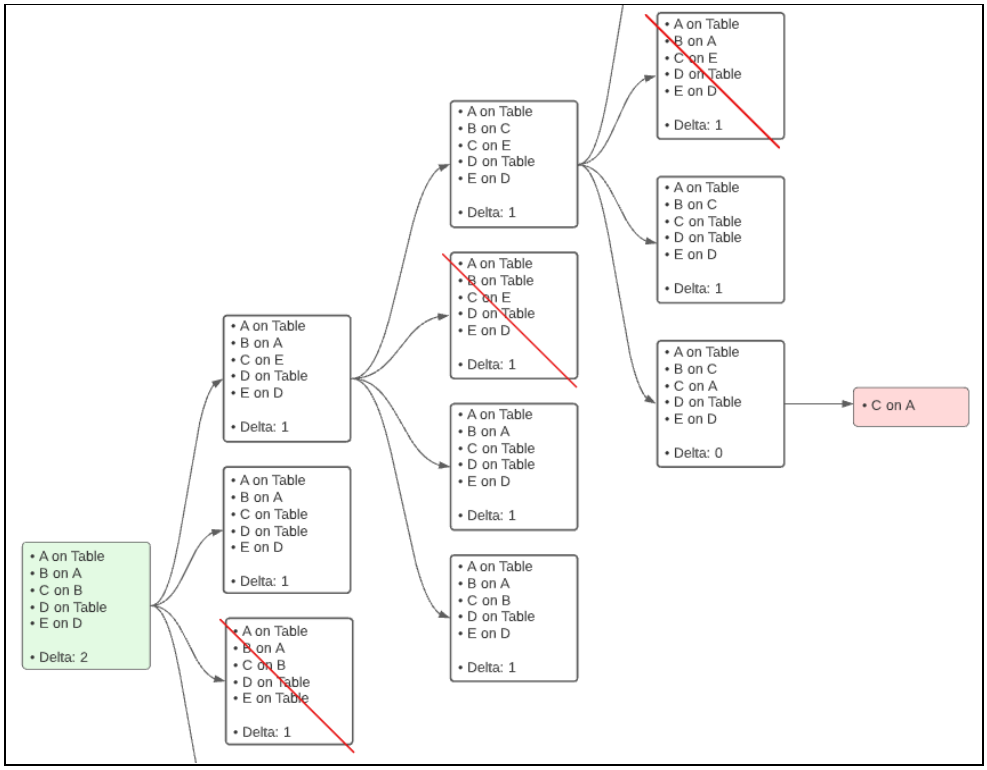

Figure 2—Depiction of the generated and tested states when the agent traverses using a subproblem as a goal.

1.5 How does your agent compare to a human? Does your agent solve the problem the same way you would?

I do think the agent solves the problem in the same way that I would. Previously, in the previous mini project I would not have said this about the AI agent. However, for this mini project, the problem solving methods that the agent uses was derived directly from my own attempts to solve the problem and understand how Means-End Analysis and Generate and Test can be used to effectively arrive at an optimal traversal. I could only implement the AI agent after I had understood the problem solving method clearly—from there it was just a matter of translating that into the AI agent's code. In fact, I'm starting to think the key to solving these algorithms is to closely observe how a human mind thinks and translate that precisely into an algorithmic solution. Once the logic that operates in the human mind is identified and implemented as an algorithm, the AI agent performs that problem solving technique but much more rapidly and with greater capacity to scale to larger problem sets. Thus the agent solves the problem in the same way that I would but, whereas a human cannot effectively manage tracking and comparing so many states, the agent is able to do this easily.