Mini-Project 2

Timothy Thatcher
tthatcher3@gatech.edu

***Abstract*—This report describes the design of a knowledge-based artificial intelligence (AI) agent that can be used to effectively solve the Blocks World toy problem.**

1 INTRODUCTION

The **Blocks World** (Russell and Norvig, 2010) problem is a toy NP-Hard planning problem in AI research. The problem can be succinctly described as:

- Only one block at a time may be moved onto another block or to the table.
- The goal is to move the initial arrangement to the goal arrangement with the minimal number of moves possible.

The toy problem is challenging because of the existence of the Sussman Anomaly (Russell and Norvig, 2010) which results in the direct pursuit of sub-goals being suboptimal in many cases (figure 1).

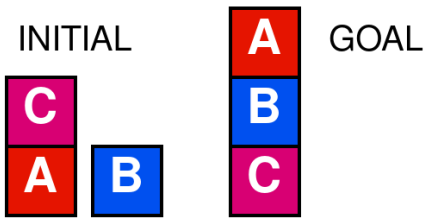

***Figure 1*—Visualization of the Sussman Anomaly. Although placing `B` onto `C` would solve the subgoal of `B` being stacked onto `C`, the stack would need to be disassembled to move `A` on top.**

2 AGENT DESIGN

The agent design was based on a variant of **A* Search** (Russell and Norvig, 2010) that combined elements of **generate & test** and **ends-means analysis**.

2.1 State Representation

A graph-based **semantic network** representation of the block arrangements was used to capture the puzzle state. Let $\mathcal{B}$ represent the set of unique blocks (ver-

tices). The state, S, was represented using a set of tuples of directed edges:

$$S = \left\{ (B_u, B_l) \;\middle|\; B_u \in \mathcal{B} \;\wedge\; B_l \in \mathcal{B} \cup \{\varnothing\} \;\wedge\; B_u \neq B_l \right\}$$

where B_u is the upper block and B_l is the lower block or the table (represented by $\varnothing$). In other words, each block $B_u \in \mathcal{B}$ was paired with the table or block it is located on top of.

2.2 State Comparison and A* Heuristic

For both the **generator** and the A* **heuristic** function, a **compare** function was required. The compare function would consume a goal state G with a current state S and return a set M of matched blocks and a mapping R of blocks and their required location to improve the matches in M. The matched blocks consisted *only* of blocks that formed a chain of correctly placed blocks at the base of a stack. Similarly, the mapping of required blocks consisted *only* unobstructed positions that would improve the matches. For example, given the following goal state G and initial state S (see figure 2):

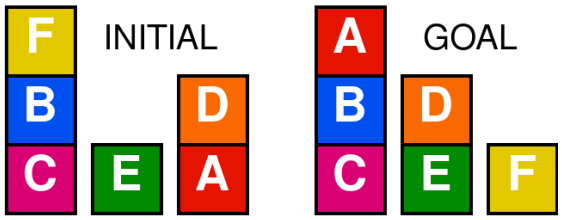

Figure 2—Visualization of the example initial and goal states. Blocks `B`, `C` and `E` form the matched set M. Mapping R consists of `F` onto $\varnothing$ (table) and `D` onto `E`; `A` onto `B` is obstructed by block `F` and is therefore excluded.

$$G = \{(A,B),(B,C),(C,\varnothing),(D,E),(E,\varnothing),(F,\varnothing)\}$$
$$S = \{(F,B),(B,C),(C,\varnothing),(E,\varnothing)(D,A),(A,\varnothing)\}$$

Then applying compare $(G,S) \mapsto (M,R)$ would output:

$$M = \{B,C,E\} \qquad R = \{D : E,\ F : \varnothing\}$$

In other words, block `D` should be moved on top of block `E` and block `F` should be moved to the table. Note that `A` : `B` is not in R since block `F` would need to be moved in order to open the position.

Using the match output of the comparison function, the A* heuristic was constructed as:

$$\text{heuristic}(S, G) = |\mathcal{B}| - |M|$$

In plain terms, the heuristic is the number of blocks that are not correctly arranged at the bottom of a stack. This corresponds to the best-case minimal number of moves to arrange the current state to the goal state: moving each out-of-place block once. This was an important consideration since in order for a heuristic to be **admissible** for A*, it must never over-estimate the number of moves.

2.3 Algorithm

The A* algorithm was selected because it is guaranteed to find the optimal path to the goal given an admissible heuristic. The **generator**, used by A* to explore the neighbouring states, made use of the the following two important observations:

1. A block should only be stacked if its goal position is open and it would never be required to move again (eg. all lower blocks in the stack are correct).
2. A blocks should only be moved to the table if its goal position is not open.

The conditions above were verified using the mapping R from the compare function. This had the advantage of **pruning** a large number of neighbour branches in the search. The heuristic function served as an **ends-means analysis** to prioritize the states in the **priority queue** of the A* search. Each iteration of A* performed a **test** of the current state in the queue to determine if the goal had been reached.

3 AGENT EFFECTIVENESS

The agent consistently finds an optimal solution to the Blocks World problem. Although many arrangements exist that present the Sussman Anomaly, the backtracking nature of A* enables the agent to return to earlier states and continue the search. Further, the design of the generator prevents the pursuit of subgoals where an arrangement would need to be undone. Similarly, the heuristic function also does not reward subgoals that would result in arrangements being reverted. This robust design prevents the agent from becoming stuck in edge cases.

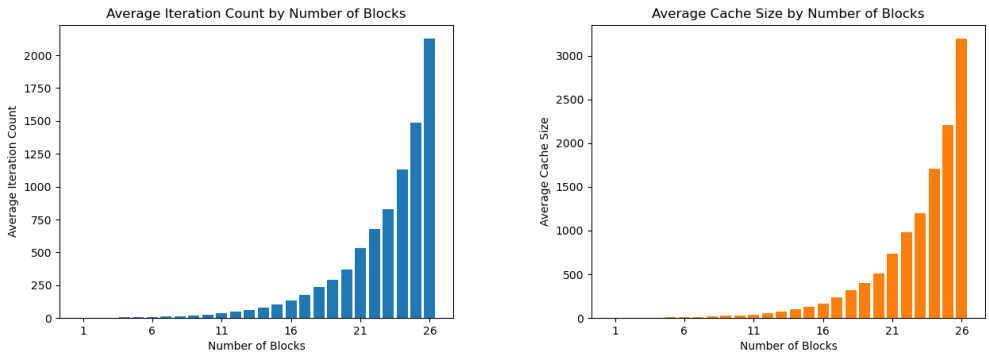

Figure 3—The plot on the left tracks the number of A* iterations (empirical performance complexity) and the plot on the right tracks the size of the search cache (empirical space complexity).

To demonstrate the empirical computational efficiency of the agent, 1000 randomly generated initial states and goal states were created for problems up to size 26 to be solved by the agent. The number iterations and the search cache size upon completion of A* was tracked for each of the test cases. Figure 3 shows a plot of the agent performance based on the two metrics. The A* algorithm is known to be exponential with the branch factor as the base and the distance as the exponent (Russell and Norvig, 2010). Consistent with the theory, the empirical performance and space complexity grow exponentially as the size of the problem grows.

4 AGENT COMPARISON TO HUMANS

The generator and heuristic function of the were designed based on the strategy the author took to solving the Blocks World problem. As a result, the behavior of the agent closely aligns with that of the author (though, the agent is far less error-prone at the execution).

5 CONCLUSION

This paper demonstrates how the principles of knowledge-based AI can be used to design an agent that effectively solves the Blocks World problem.

6 REFERENCES

[1] Russell, Stuart and Norvig, Peter (2010). *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall.