Mini-Project One: Sheeps & Wolves

Allen Worthley

mworthley@gatech.edu

1 INTRODUCTION

This document discusses the Mini-Project One and the AI Agent code set that solves the Sheeps and Wolves problem. The problem follows the semantics of the Guards and Prisoners problem, however, applied to a shepherd in need of transporting animals across a river. To solve the problem, an AI agent needs to return a set of moves that optimally transports all animals to the opposite coast. Additionally, at any stage, wolves cannot outnumber sheep on either coast and the boat should only carry either one or two animals at a time. The proceeding sections demonstrate the design of the solution code, how it performs, and how it compares to human rationale.

2 DESIGN

The AI Agent follows a generate and test methodology. The generator attempts to minimize the unproductive states it produces to increase efficiency. The generator limits the number of states that produce negative values of animals on either coast and that are the result of boats that contained less than one or greater than two animals. The test portion of the methodology removes generated states that were previously generated and validated.

The following sections dive into the more granular level of design including the design of the overall data model, generator, and validator.

2.1 Data Model

There are four important data structures that organize the program: a nested list of branches with nodes (Master_Tree), a nested list with branches of moves

(Moves_Tree), a list of non-duplicated previous nodes (Memory) and the nested list representing the nodes. Figure 1 shows a high level representation of how the data is structured and related.

The structure of the Master_Tree and Moves_Tree align; meaning, for a given branch in Master_Tree, the same branch index for Moves_Tree yields the corresponding moves made by the nodes in Master_Tree. Additionally, the branch index is tracked at the node level in Branch_ID.

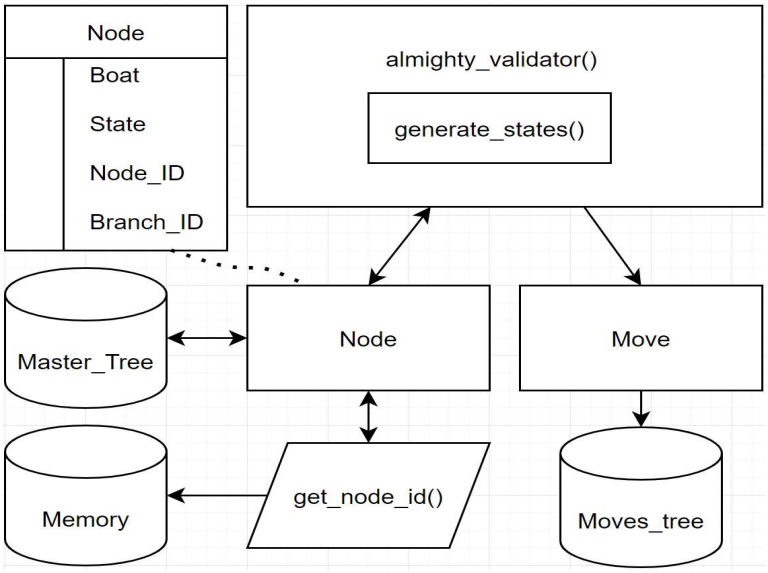

Figure 1— Data structure and relations.

2.2 Generating the Tree

The generate_states() function generates new possible nodes using a given node. The node houses information on the current Boat or coast and the matrix representation of the state. A loop iterates over changes in animal combinations then adds valid combinations multiplied by a boat modifier to the given node state to generate new nodes. The boat modifier inverts operations for either coast. The modifier takes the value of one when representing the left coast and negative one for the right coast. Once created, the generator assigns a Node_ID. The

Node_ID uniquely identifies the state and is a string concatenation of the boat modifier and state matrix cells.

2.3 Validating the Nodes

The almighty_validator() function checks the generated states for the following conditions: wolves do not outnumber the sheeps and either side, wolves can be zero but with no sheeps on a coast, the current Node_ID is unique in the Memory database. The validator function returns TRUE or FALSE to the generator function where it conditionally assigns the Node_ID to Memory database.

2.4 Wrapping it All Together

A While loop iterates on the condition that the number of new nodes generated is greater than zero and the branch containing the solution state (where all zeros are on the left coast) has not been found. While the condition is true, the AI Agent loops through all branches of the Master_Tree, calls the generate_states() function, and decrements the number of new states generated. Each iteration of the generate_states() function adds new branches to the Master_tree for each new node generated. Once the branch containing the solution is found, the program outputs the list of moves in the associated branch of the Moves_Tree.

3 PERFORMANCE

The cProfiler in python measures the performance in seconds of the AI Agent solving scenarios where Sheep and Wolves are the same at four different levels: one, ten, hundred, and one thousand. The agent performs fairly well and plateaus out at around 0.148 seconds as shown in figure 2. Luckily, the methodology only really writes data and does not sort or search. The methodology creates an algorithm that compares the length of the Memory

database to the length of the list as a set with the new node appended. If the set is longer than the original list with the new node, then the node is unique.

Figure 2— Mapping logic between moves_tree and master_tree.

However, the AI Agent can be improved. The current build uses repeating data for convenience. Parent nodes are repeated for each branch of both the Master_Tree and Move_Tree.

4 HUMAN COGNITION RELATION

Does the AI Agent solve the Sheeps and Wolves problem in a human like manner? No, the agent has far greater memory recall than a human and is far faster in iterating through all possible scenarios. However, the logic used by the agent is very similar to how an optimal human may solve the problem. To solve the problem, both a human and computer would apply structured rudimentary steps and searches to find the optimal output.