Mini-Project 1: Sheep and Wolves Journal

Natalie Olivo
nolivo3@gatech.edu

Abstract—This Mini-project challenges us to build an agent which can efficiently solve Wolves and Sheep crossing a river, where wolves cannot out-number sheep on either side.

1 AGENT DESCRIPTION

My agent utilizes the Generate and Test methodology, constructed by a smart generator and smart tester. The agent first generates all possible moves, ie: 1) only moves that do not undo the last move, 2) moves that transfer beings that are actually there. Next, the agent uses a smart tester to only follow paths which satisfy the sheep outnumbering wolves requirement. The agent then uses Depth-First search to pursue each path from its initial state to its desired state. When the agent finds itself revisiting the initial state, 100 moves, or there are no moves that create a legal state, it immediately discards this path, and presumes the next path could be the answer. My agent has trouble with animals in odd quantities (5,5) and (13, 13).

1.1 Smart Generator

The smart generator starts by normalizing the data. The side of the river with the boat is known as a_state and the destination of boat passengers is known as b_state. This normalization is also useful in the Smart Tester.

From the list of all possible moves, or [(1, 0), (2, 0), (0, 1), (0, 2), (1, 1)], the Smart Generator then only preserves moves that do not match the most recent move (as that would undo what had just occured), and it makes sure all animals moving over on the boat actually exist, by checking the number of animals proposed to leave on the boat is less than or equal to the number of animals on the shore.

1.2 Smart Tester

The smart tester makes sure the state created by a proposed move is legal, ie: it fulfills the constraints placed upon the number of wolves and sheep on each side of the shore. These tests are able to substantially reduce the problem space making the agent more efficient.

2 AGENT EFFICIENCY

2.1 Reducing problem spaces with Recursion

Given the smart Generator, the smart Tester, the problem is substantially reduced. My agent is able to use depth-frist search for finding the the most efficient path to the desired state. If at any point the path begins repeating itself, or cannot proceed without undoing its last step, the entire path is discarded and the next possibility is pursued. Despite this efficiency, my agent is prone to be greedy - or choose the first path to the desired state over the quickest.

3 AGENT PERFORMANCE

My agent achieved 18/20 challenge questions correctly and struggled with (5 sheep, 5 wolves) and (13 sheep, 13 wolves.)

4 HUMAN COMPARISON

As a human, I'm quickly overwhelmed by problems prone to combinatorial explosion. Although I may catch inefficient paths quicker than some, my intuition towards the most efficient path can be flawed. I quickly look to white-board or pen-and-paper it out. It's easy to get disorganized in a hurry, and honestly, I loved imagining the correct knowledge representation for this problem.