Sheep & Wolves Problem Report

Xinghao Chen
xchen785@gatech.edu

***Abstract*—**This document reports the solution to the Sheep & Wolves problem. The state of the problem is defined, and all the states that can occur are stored in a state tree without repetition. With a method similar to breadth-first search, a new layer of the state tree is explored iteratively to find the solution. This method ensures the optimal solution, at a relatively heavy consumption of time and memory.

1 STRATEGIES

The most important strategy used in the AI agent is just to generate and test.

1.1 State

In this problem I define the state as the number of sheep and wolves at both sides, and whether the shepard is on the left or the right side of the river.

Definition for state

- number of wolves and sheep on the left side
- number of wolves and sheep on the right side
- the side where the shepard is on

1.2 Generating new states and testing with constraints

We can generate all possible moves (2 or 1 sheep, 2 or 1 wolf, 1 sheep and 1 wolf) and check whether wolves are more than sheeps on each side of the river. Besides, the number of wolves and sheeps on each side should always be greater than 0.

State generation

- move 2 or 1 sheep
- move 2 or 1 wolf
- move 1 sheep and 1 wolf

Checking constraints

- no more wolves than sheeps on both sides, unless there is no sheep on a side

- number of wolves and sheep are no less than 0 on both sides

1.3 State Tree

A state tree is built in my AI agent to record every state that has been generated and tested. The depth of the tree refers to the number of operations. The tree is initialized with only 1 layer containing the input state, and every time, by trying all possible moves, a new layer is generated from all the states in the previous layer. States that are illegal or the same as previous states are dropped.

State tree structure

- root layer: 1 node recording the input state
- next layer: all possible states that can be generated from the previous layer
- requirement: All the nodes must be legal and productive.

Productivity check

- A new state cannot be inserted into the tree if there has been a same state existing in the tree.

1.4 Confirming the final answer

Given the input state, we can immediately find the state of the solution. Therefore, when exploring a new layer of states in the tree, we can compare each new legal state with the solution. If, in an epoch of exploration, no legal and productive state can be generated in the new layer, we can confirm that there is no solution to the problem.

Final answer confirmation

- The answer is found if a state that is the same as the solution is generated.
- There is no solution if no new legal and productive state can be generated in a new layer.

2 EFFICIENCY

My agent, by exploring a new layer of states in the tree each time, performs a process similar to breadth-first searching. This ensures the optimal solution with respect to steps the shepard has to move. If no state was dropped, each state from the previous layer would exponentially generate 5 new states in the new layer. In this way, though fortunately many states are dropped, the agent may

still consume up to $O(n^2)$ scale of memory and $O(n^3)$ scale of time (with n defined as the number of animals). But a breadth-first agent can always find the optimal solution. This method may be different from what humans use, beacuse it can be difficult for humans to record large numbers of states.

If the state tree would be overwhelmingly large according to some inputs, we may have to give up the breadth-first search. There may be some methods similar to heuristic depth-first searching, which is more human-like, and may give an answer more quickly than breadth-first search. We may choose to explore from "seemingly more productive" states preferentially.