Mini Project 1:
The Sheep and Wolves Problem

Nicholas Gliserman
gliserma@gatech.edu

1 A LINEAR-TIME SOLUTION

In lecture 3.11, professor Goel refers to the received wisdom in the field of Artificial Intelligence that states: "If you have the right knowledge representation, problem solving becomes very easy." My experience with this project confirms this maxim. Honing my representation helped me to see a much simpler problem. Ultimately, I created an algorithm that provides an optimal solution to all configurations of the "sheep and wolves" problem in linear time (contingent on the number of sheep) with a trivial amount of code.

1.1 Knowledge Representation

Initially, I thought a graph would capture this problem well, with nodes representing a particular configuration of sheep and wolves on the two shores and with edges capturing the process of moving sheep and wolves back and forth. A breadth-first search (BFS) from the start state (i.e., everybody on one shore) to find the goal state (i.e., everybody on the other shore) would find an optimal solution, if it existed. I disregarded this approach, however, after completing a thought experiment where a table of Boolean values represented the wolves and sheep. The only 'true' value in the table indicates the number of wolves and sheep. Two such tables (fig. 1) would capture the configurations on both shores.

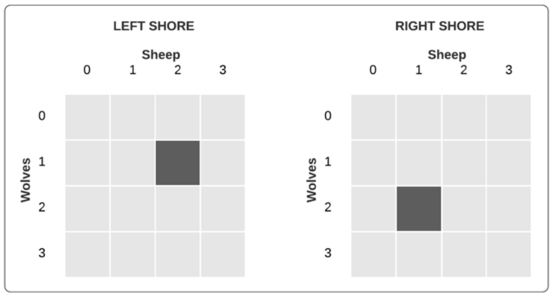

Figure 1—Two tables capturing a given state of the problem.

Soon, I realized that additional tables could encapsulate other information such as illegal sheep-wolf configurations (i.e., more wolves than sheep) or even the action of transporting sheep and wolves. My breakthrough came when I realized that the two tables, depicting right and left shores, had a dependency. If you know the configuration on the left shore, you can determine that of the right shore. Thinking about this in terms of transformations, to go from one shore to the other, you only need to rotate the table 180º. Superimposing the two tables together with illegal states marked (fig. 2) helped me to see the problem as a maze wherein the agent was constrained by how and where it could move.

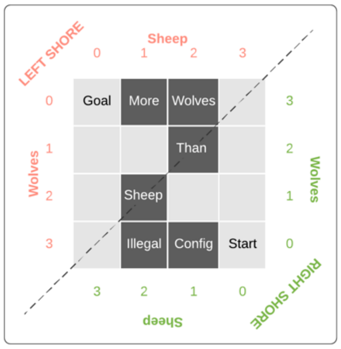

Figure 2 — The problem reconceptualized as a maze.

1.2 Movement

If this was an ordinary maze where the agent could move to any adjacent space, it would be trivial to find an optimal solution because 'walls' appear in predictable locations. If sheep and wolf numbers are equal, the agent only need move diagonally from start to goal. If sheep exceed wolves, the agent merely need move laterally after reaching the top of the maze. The caveat in this problem is that the movement patterns are not straightforward. From one space, an agent can move to five potential spaces (fig. 3). The five spaces then rotate 180º each turn as the boat reverses direction. Together, these choices provide up to twelve novel states reachable over two turns.

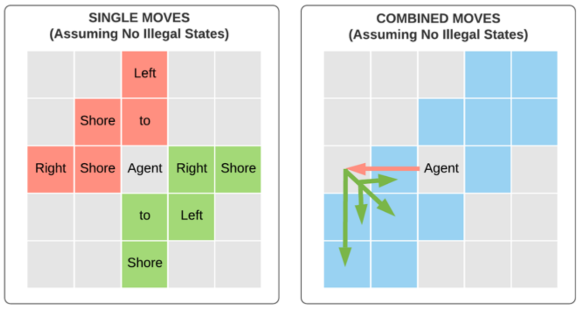

Figure 3 — Where the agent can move in one or two turns.

1.3 The Trivial Case: Sheep Outnumber Wolves

If we don't have to worry about obstacles, we can achieve the optimal diagonal path every four turns by moving left (up-left 1; down 1) and up (up-left 1, right 1). Our lateral path would involve moving left (left 2, right 1). This is the basis of the algorithm I developed. Whenever sheep outnumber wolves, no obstacles impede the agent's ability to follow this basic cycle of movement (fig. 4). The trickiest part of generating the list of moves is simply knowing A) if and when to transition from diagonal to lateral movement and B) when to stop. The processing time here is strictly related to the overall number of moves because the algorithm will never explore a suboptimal route.

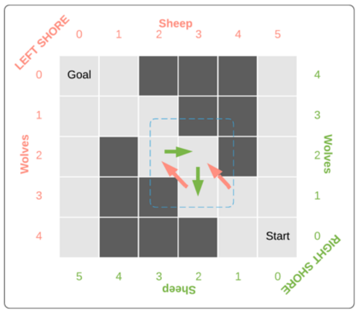

Figure 4—The vector cycle for net diagonal movement.

1.4 Corner Cases: Equal Number of Sheep and Wolves

When the number of sheep and wolves is the same, we do now have to worry about the obstacles. Observe in fig. 2, when the barriers close in, the diagonal path is 1-cell wide, meaning it is not possible to go left and then up. The agent would need to take a different approach to reaching the goal, namely jumping laterally over the barrier.

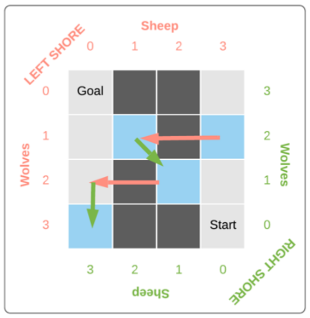

Figure 5—Crossing the divide with lateral movement.

This strategy only works if the lateral jump size is greater than half the number of sheep (rounded down). Why is this? When an agent is in any of the interior diagonal cells, they cannot move vertically—there are no 'legal' spaces to land on. So, the agent must be able to completely cross the divide by a lateral jump to the left, a slight rightwards diagonal backtrack, and then another leftward leap. Each lateral leap must carry the agent halfway across the board and a little bit extra to account for the backtracking. Consequently, if the largest lateral leap size is two, the problem is only solvable if there are three sheep or less.

With this insight, I could see there were really only three corner cases I would actually need to consider where: 0 < sheep == wolves <= 3. The case of 1 sheep/wolf is trivial. The case of 2 sheep/wolves follows a similar meta pattern as when sheep outnumber wolves with the order of up and left moves reversed.

The case of three sheep/wolves is the trickiest because it requires a different type of movement pattern but only one pathway (fig. 5) exists for crossing over the interior columns, guaranteeing it will be optimal. Said differently, once the agent initiates the lateral move, it only has one 'legal' option available to it at any time until it reaches the other side. When in the start and goal columns, it only needs to move upwards and it has only one way of making upwards progress (up 2; down 1). It might *seem* to require intelligence to know when to start moving laterally but when it reaches this position, it actually only has two options available: up 1, which returns it to a previous state; or left 2, the only novel option.

2 CONCLUSIONS

In the lectures, professors Goel and Joyner often discuss the role of human intelligence in designing AI agents, especially the question of which agent is ultimately intelligent—the computer or the human. In this case, human intelligence reduced the "wolves and sheep" problem into a very simple maze problem, solvable with very little recourse to deliberation. A human might visually intuit the solution while my AI agent simply follows a prescribed algorithm. In this sense, I created a state generator so good as to mitigate the need for the AI agent to do any testing.