RPM Milestone 1 Journal

James Moore
jmoore429@gatech.edu

***Abstract*—This document presents an algorithm for solving the sheep and wolves problem, also known as the cannibals and missionaries, or prisoners and guards problem using a priority search algorithm using Means-End Analysis. Performance characteristics, Computational and space complexity are scrutinized along with a comparison against a human solver (the author).**

1 PROBLEM

The basic problem statement for the sheep and wolf problem is that N sheep and M wolves must be moved from one shore to a second shore using a boat that can only contain two animals. The wolves can never outnumber the sheep on either coast. The goal of our agent is to produce the smallest set of moves that result in all animals being moved to the opposite shore. Alternately the agent can determine that there is no path to moving all animals to the opposite shore.

2 ALGORITHM

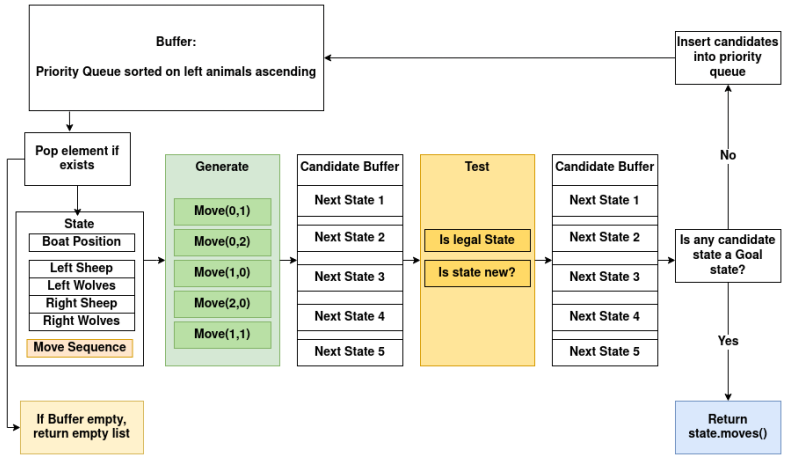

***Figure 1*—A flow chart depicting the priority search algorithm. The moves required to reach the state are stored in a hidden variable on State**

The algorithm depicted in Figure 1 represents a state as a 4-dimensional array combined with a boolean flag to indicate boat position. The state representation

also stores a list of moves required to reach the state internally for tracking purposes. At each step in the search, the searcher selects a state from the buffer and generates a list of candidates for all valid moves. A valid move is any move where there are a sufficient number of animals on the shore for the move to occur. All candidate states are then filtered to ensure that wolves do not outnumber sheep on either coast and that the state has not been previously visited by the searcher using a hash set. All remaining candidate states are added to the buffer.

The primary consideration for the performance and accuracy of the search algorithm is the selection operation on the buffer. Selecting the oldest entry in the buffer results in an exhaustive Breadth-First Search which is guaranteed to produce the optimal path but is impractical on large state spaces. Leveraging Means-End analysis, we implement a priority selection. The search algorithm selects the next search state by finding the candidate state with the smallest delta to the goal state of having all animals on the right river bank given by the sum (left-sheep + left-wolves). To efficiently maintain this ordering of candidate states, we store the candidate states within a priority queue.

2.1 Accuracy

The priority search algorithm successfully passes all 40 test cases on the CS7367 rubric. The algorithm is guaranteed to identify the optimal move ordering given that there does not exist any state **S** with the following properties. The state **S** would need to increase the total number of animals on the left coast *and* lead to a shorter path to the goal state than any non-cyclical paths from states with lower animal counts on the left coast. It can be shown that such a state **S** does not exist as increasing the number of animals on the left coast is equivalent to solving a larger sheep and wolves problem. Therefore the shortest route to move all sheep from the left coast to the right coast must always be the path that increases the number of animals on the left the least.

2.2 Performance and Complexity

The searcher depicted in figure 1 features the following computational complexities on it's components.

1. $\mathbf{O}(\mathbf{BufferStates} * \mathbf{Log}(\mathbf{BufferStates}))$ Complexity of maintaining the buffer.
2. $\mathbf{O}(\mathbf{Moves}^5)$ Worst case increase in Buffer States to be explored
3. $\mathbf{O}(\mathbf{BufferStates})$ Cost to score the states according to Means End Analysis.

4. **O**(**sheep** + **wolves**) Number of moves required to move the sheep and wolves to the other side - or identify lack of paths.

Given the above, the algorithm has an extreme worst-case performance if all moves need to be explored and added into buffer states of $\mathbf{O}(\mathbf{N}^5)$, assuming a constant linear relationship between the number of sheep and wolves and the number of moves required for them to reach the other island. There is a best case complexity of O(sheep+wolves) if the algorithm correctly selects the next state to be explored after every move. Based on the convergence argument from Section 2.1 we would expect the means-end analysis to only explore a fixed number of paths as it only searches the moves that allow it to make forward progress. Empirical evaluation of the algorithm's performance for problems of size (i+1, i), (i,0), (i,i), and (0,i) for ranges of i between 0 and 150 in figure 2 which clearly demonstrate a greater than linear runtime complexity.

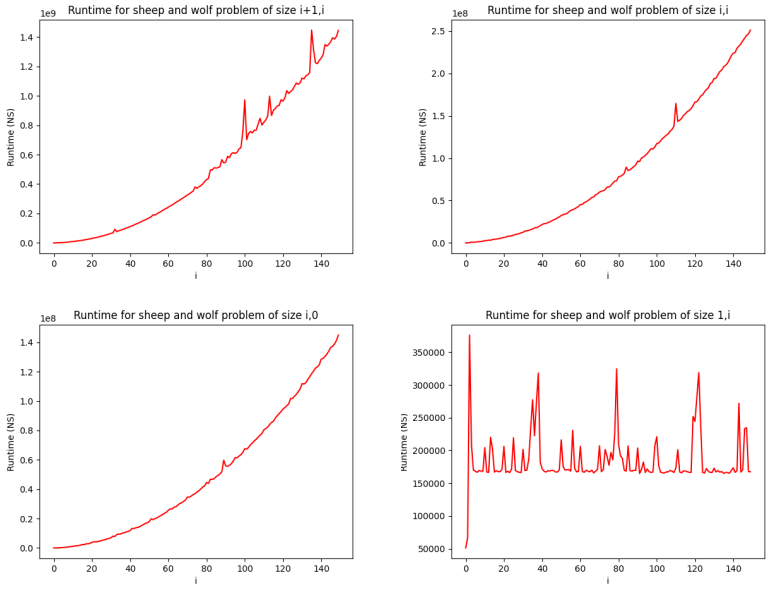

Figure 2—Runtime for sheep and wolf problems of sizes (i+1,i), (1,i), (i,0), and (i,i)

The performance on (i,i) indicates that the algorithm must search a substantial portion of the state space to reject the problem. Direct inspection of the searched states for this case indicates that the algorithm attempts to move all wolves to the opposite shore then attempts and fails to move the sheep.

2.3 Performance Improvements

The search algorithm presented in Section 1 requires a super-linear time search to find the optimal moves to solve the sheep and wolves problem. Inspection of the generated moves reveals that the move set for any problem of shape (i+1,i) is equivalent to concatenating the moves from an (i,i-1) agent except for (3,3), (2,2), and (1,1) problems. The moves required to generate an (i,0) agent are equivalent to the moves needed to solve the problem for (i-1,0) sheep concatenated with the solution (1,0). Problems of size (i,i) greater than i=3 never have a solution. This can be seen explicitly in the sequences summarized in table 1. Using these patterns, a dynamic programming agent could automatically generate the optimum path or reject the problem. This algorithm would achieve linear time performance for solvable problems and O(1) performance for unsolvable problems

Table 1—Sample Sequences from problem of shape 5,4 and 6,5. Note the bolded begin and end sequences are identical and the mid sequence consists of a repeating pattern of (1,0),(1,1),(0,1),(1,1)

(Sheeps, Wolves)	Move Sequence
(5,4)	**(1, 1), (1, 0), (2, 0)**, (1, 0), (1, 1), (0, 1), (1, 1), (1, 0), (1, 1), (0, 1), (1, 1), (1, 0), **(2, 0), (0, 1), (0, 2)**
(6,5)	**(1, 1), (1, 0), (2, 0)**, (1, 0), (1, 1), (0, 1), (1, 1), (1, 0), (1, 1), (0, 1), (1, 1), (1, 0), (1, 1), (0, 1), (1, 1), (1, 0), **(2, 0), (0, 1), (0, 2)**

3 HUMAN COMPARISON

The search algorithm presented in Section 1 requires a super-linear time search to find the optimal moves to solve the sheep and wolves problem. The Search algorithm presented in Section 2 follows a similar approach to how the author solved the problem on pen and paper. Following the process of 1) Searching the most productive states, and 2) excluding states that I've already visited. On larger animal count problems, the author experienced a process similar to the dynamic programming exercise in section 2.3, where a pattern of productive moves is repeated until it is no longer productive.