Mini-Project 1 Journal

Juejing Han
jhan446@gatech.edu

1 QUESTION 1

How does your agent work? How does it generate and test new states?

My agent solves the Sheep and Wolves problem by searching the state space with the Breadth-First Search (BFS) algorithm. The boat starts on the left bank and alternates sides after every move. My agent tracks the left bank state as:

$$State_{left} = (S_{left,}W_{left,}B)$$

Where S_{left} is the number of sheep on the left bank, W_{left} is the number of wolves on the left bank, and B indicates on which bank (left or right) the boat is.

Meanwhile, my agent can compute the corresponding right bank state by:

$$S_{right} = S_{initial} - S_{left}, W_{right} = W_{initial} - W_{left}$$

Where $S_{initial}$ and $W_{initial}$ are the initial numbers of sheep and wolves, and S_{right} and W_{right} are the numbers of sheep and wolves on the right bank, respectively.

My agent begins from its initial state on the left bank as:

$$State_{left,initial} = (S_{initial,}W_{initial,}left)$$

And it reaches its goal state as:

$$State_{left,goal} = (0, 0, right)$$

In each iteration, my agent applies all valid moves to the current valid state to generate new states. The valid moves are [(1, 0), (0, 1), (1, 1), (2, 0), (0, 2)], representing how many sheep and wolves are on a boat trip.

A new state is then tested as valid if 1) the count of each group (sheep or wolves) on each side is **non-negative**; 2) the **Outnumber Restriction holds** on each side – wolves never outnumber sheep when sheep are present; and 3) the state **has not been visited** before. After applying these rules (collectively referred to as **the Validation Rule** in this journal) to test the new state, my agent adds the valid new state to its state space.

With the BFS algorithm, my agent explores states in layers[1], ensuring all states at the current layer are visited before moving to the next layer. In this Sheep and Wolves problem, the first layer contains the initial state, the second layer covers all states reachable in one move from the initial state, the third layer includes all states reachable in two moves from the initial state, and so on. My agent utilizes a queue with the FIFO (First In, First Out) rule to maintain the layer order.

My agent also tests whether the state it is processing is the goal state. In the BFS setting, when my agent reaches its goal, it yields the solution with the minimum number of moves from the initial state to the goal state, i.e., the optimal solution. If my agent cannot reach its goal state after exhausting the state space, it returns an empty list to indicate the problem is unsolvable.

2 QUESTION 2

How well does your agent perform? Does it struggle with any particular cases?

My agent successfully passes 20 out of 20 test cases during the performance test, yielding optimal solutions to 18 solvable cases and correct returns for 2 unsolvable ones. It receives full credit and does not struggle with any particular cases during the performance test.

As explained in Section 1, BFS explores states layer by layer, ensuring that the shortest path is found once the goal is reached. Utilizing the BFS algorithm, my agent guarantees the optimal solution if one exists. Although larger configurations will expand the state space, my agent prunes the state space to remain efficient within the performance test settings.

3 QUESTION 3

How efficient is your agent? How does its performance change as the number of animals rises?

In the performance test, my agent performs efficiently by passing 20 out of 20 test cases within the allowed runtime and receives full credit.

1 https://www.appliedaicourse.com/blog/bfs-in-ai/.

The time complexity of BFS is $O(V + E)$, where V is the number of states and E is the number of transitions[2]. It is linear in the graph size[3]. In this Sheep and Wolves problem, each state has up to 5 possible moves, so $E \leq 5V$. Therefore, the theoretical time complexity of my agent is $O(V + E) = O(V + 5V) = O(V)$, which is linear in the number of states.

Table 1 — Agent performance by problem size.

Test Case (Sheep, Wolves)	Optimal Solution Found	Number of Moves	Runtime (s)
(10, 9)	*True*	35	0.0002
(100, 99)	*True*	395	0.0024
(1000, 999)	*True*	3995	0.0459
(10000, 9999)	*True*	39995	2.6405

Table 1 shows that as the problem size (i.e., number of animals) increases from 10^1 to 10^4, my agent always returns the optimal solution, while the runtime increases from 0.0002 to 2.6 s. It implies that my agent scales efficiently as the problem size increases. The reason is that my agent only explores the valid state space, filtering out invalid states that violate the Validation Rule. This demonstrates that pruning keeps the state space search manageable as the problem size grows.

4 QUESTION 4

Does your agent do anything clever to yield an answer more efficiently?

My agent does two things to yield answers efficiently. As explained in Section 1, my agent utilizes BFS, which guarantees the optimal solution (if one exists) because my agent explores the state space in layers. In addition, my agent also uses the Validation Rule defined in Section 1 to filter the invalid states. Therefore, my agent effectively reduces the search space, prevents the state space from growing excessively, and yields the optimal solution with less computational cost.

2 https://en.wikipedia.org/wiki/Breadth-first_search.

3 https://web.engr.oregonstate.edu/~huanlian/algorithms_course/3-graph/bfsdfs.html.

5 QUESTION 5

How does your agent compare to a human? Does it solve the problem the same way you would?

My agent solves the problem the same way I would. We both use the same strategy described in Section 1. I would process valid states in layers to find the optimal solution, manually generating and testing new states under the Validation Rule. My agent applies a similar way to human reasoning.

Generally, my agent outperforms a human. For simple cases such as problems (1, 1) or (2, 1), a human may arrive at the optimal solution as efficiently and accurately as my agent. However, as the problem size increases, my agent demonstrates consistent efficiency and accuracy (as shown in Sections 2 and 3). In contrast, as a human, I need considerable time to process the problem, sort out valid states, track possible moves, and progress step by step toward the goal. When mistakes occur in this manual process, it is difficult to debug and often requires starting over and repeating the same reasoning at each layer.

Therefore, my agent is more efficient and accurate at solving the Sheep and Wolves problem than a human. In particular, for larger problem sizes, my agent undoubtedly outperforms a human.