CS7637: Knowledge-Based AI: Mini-Project 1

Yaxin Yang
Yyang894@gatech.edu

Abstract— The SemanticNetsAgent is designed to solve a classic river crossing problem involving sheep and wolves. The agent aims to move all the sheep and wolves from one side of the river to the other while following specific constraints.

1 HOW DOES THE AGENT WORK

The SemanticNetsAgent is constructed around a Breadth-First Search (BFS) algorithm to explore possible states and navigate through them to find a solution to the problem. Each state is represented as an instance of the SemanticNetsAgent class, capturing the number of sheep and wolves on both sides of the river, the side on which the boat currently resides, and other meta-information like previous states and actions taken.

1.1 State Generation:

The agent generates new states using the generate_next_states method. This method loops through potential combinations of sheep and wolves that can be moved across the river. Specifically, it does so in a nested for-loop that considers moving between 0 and 2 sheep and between 0 and 2 wolves.

The method then simulates these moves to produce a new state. It uses deep copy to clone the current state and then modifies the clone based on the action being considered (moving num_of_sheep sheep and num_of_wolf wolves).

1.2 State Testing

Each newly generated state undergoes a validity check through the is_valid_state method. The method ensures that the following conditions are met:

1. The number of wolves never exceeds the number of sheep on either side of the river. This prevents the wolves from eating the sheep.
2. None of the counts of sheep or wolves is negative. This ensures physical feasibility.

Only states that pass this validity check are added to the BFS queue for further exploration.

By iterating through possible moves and checking each resulting state's validity, the agent builds a tree where each node represents a possible configuration of sheep, wolves, and boat positions. The BFS algorithm traverses this tree to find the shortest path from the initial state to the goal state.

2 PERFORMANCES

The agent performs reasonably well for a majority of test cases, particularly for those with smaller numbers of animals. Given that it employs a Breadth-First Search (BFS) algorithm, the agent is guaranteed to find the shortest path to the solution if one exists. This makes it highly reliable for most configurations.

The agent may run into performance bottlenecks in the following situations:

1. Large Number of Animals: The time and space complexity grow considerably as the number of animals increases. This is a direct consequence of the combinatorial nature of the problem and the BFS algorithm employed.

2. Resource Constraints: In environments with limited memory or CPU capabilities, the agent may struggle due to its requirement to store all visited states and the BFS queue.

3. Not Optimal Solution: Interestingly, after submission, in some cases the agent made valid movements, but the solution is optimal.

3 EFFICIENCIES

The efficiency of the agent decreases as the number of animals increases. The time complexity of BFS is O (V + E), where V is the number of states, and E is the number of transitions. Thus, as the number of animals increases, the state space grows exponentially, making the algorithm slower.

As the number of animals (sheep and wolves) rises, the number of possible states and edges grows exponentially. This not only increases the time needed to find a solution but also the memory requirements to keep track of visited states.

The BFS queue may become extremely large, causing significant memory usage, and possibly leading to memory overflow errors in systems with limited resources. In addition, storing visited states also becomes computationally expensive as the state space grows, adding to both time and memory requirements.

4 CLEVER

The most straightforward optimization in the agent is the "early exit" strategy. As soon as a goal state is encountered, the BFS algorithm terminates, avoiding unnecessary exploration of the remaining state space. This helps in reaching a solution faster when the goal state is reachable before exploring all possible states. The agent tests for state validity as soon as it generates a new state. This ensures that only valid states are appended to the BFS queue and reduces the number of states that need to be processed.

5 COMPARISONS TO HUMAN

5.1 Speed and Efficiency

The most obvious advantage of the agent is speed. Once the algorithm is initiated, it can examine states and find a solution much faster than a human can manually analyze the problem. A computer can process the BFS algorithm in a matter of seconds, whereas a human might take minutes or even longer to manually solve the same problem, especially as the number of animals increases.

5.2 Algorithmic Versus Intuitive

The agent takes a systematic, algorithmic approach to solve the problem, explore all possible states in a breadth-first manner. In contrast, humans typically employ a more intuitive, heuristic-based strategy, using their understanding of the problem to eliminate unlikely moves and focusing on the most promising paths. This might lead to faster solutions in some cases but could also result in missing the optimal path if the intuition is incorrect.

5.3 Error-prone

Humans are more likely to make errors in complex and large tasks, especially when they involve a lot of state tracking and mental computation. The agent, once programmed correctly, is not susceptible to such errors and will consistently produce the correct output for any given input. Just image when sheep and wolf are both in 3 digits. Can humans provide the number of movements in minutes or seconds?

5.4 Insight and Understanding

While the agent can find a solution, it doesn't understand the problem in the way a human does. A person can often provide a rationale for why a particular solution is more efficient or why a certain strategy won't work, which can be important for teaching or explaining the problem to others.