PEAS Assessment: The Wolf-Goat-Cabbage Problem

Connor Denney, August 30, 2025

Github link: https://github.com/Connor1031/ai-assignment-1-search

---

Performance Measure

The agent's performance is measured by its ability to transport the farmer, wolf, goat, and cabbage to the opposite river bank successfully and efficiently. The criteria are:

- **Goal:** The primary goal is to complete the task without any items being eaten. A solution is invalid if the wolf is left with the goat or the goat is left with the cabbage, unattended by the farmer.
- **Completeness:** The agent must successfully move all four things (farmer, wolf, goat, cabbage) to the other side/goal.
- **Efficiency:** A good solution minimizes the number of trips. Each trip from one bank to the other has the same cost, so the optimal solution is the one with the fewest steps.

Environment

The environment for this problem is the river and its two banks. Key properties include:

- **Observability:** The environment is **fully observable**. The agent knows the location of all entities at all times. There is no hidden information.
- **Determinism:** The environment is **deterministic**. The outcome of any action is completely predictable. For example, moving the goat from the left side to the right side will always result in the goat being on the right.
- **Sequential:** The environment is **sequential**. The outcome of a future action depends on the sequence of actions taken so far.
- **Static vs. Dynamic:** The environment is **static**. It does not change while the agent is deciding on its next action. The wolf, goat, and cabbage do not move on their own.
- **Discrete vs. Continuous:** The environment is **discrete**. There are a finite number of states and a finite set of actions to move between them.

Actuators

The agent (the farmer) acts upon the environment using the boat. The specific actions the agent can perform are:

- **Rowing the Boat:** The actuator is the farmer rowing the boat to move from one side to the other.
- **Carrying an Item:** The farmer can choose to carry one of the three items (wolf, goat, or cabbage) in the boat while crossing.
- **Returning Alone:** The farmer can choose to row back to the other side alone.

Sensors

In this problem, the sensors are able to provide complete information about the environment.

- **State Knowledge:** The agent can "sense" the location of the farmer, wolf, goat, and cabbage, knowing which bank each entity is on at any moment. This corresponds to the nature of the environment.

State-Space Model

Initial State

Everything starts on the first bank of the river. We'll use 0 for the starting bank and 1 for the destination bank. The state is a list of where everyone is: (Farmer, Wolf, Goat, Cabbage).

- **Starting State:** (0, 0, 0, 0)

Actions

The farmer is the only one who can row the boat. The available moves are:

- Move the Wolf across.
- Move the Goat across.
- Move the Cabbage across.
- Move back alone.

The farmer can only move an item if it's on the same bank as them.

Transition Model

The most important rule is that a move is only allowed if it's safe. A situation is unsafe if:

1. The wolf and goat are left alone on a bank (the wolf eats the goat).
2. The goat and cabbage are left alone on a bank (the goat eats the cabbage).

So, if the farmer makes a move that results in an unsafe situation, that move is forbidden. The program has to check this for every single move to make sure it's a valid one.

Goal Test

We win the game when everyone is on the destination bank.

- **Winning State:** (1, 1, 1, 1)

Path Cost

We want to solve the puzzle in the fewest steps possible. So, we'll say that every trip across the river costs 1 point.

Normal WGC

Metric	Breadth-First Search	Iterative-Deepening Search
Solution Cost	7	7
Solution Depth	7	7
Nodes generated	10	211
Nodes Expanded	9	100
Max Frontier Size	2	7

WGC + Sheep(wolf can eat sheep too)

Metric	Breadth-First Search	Iterative-Deepening Search
Solution Cost	No solution found	No solution found
Solution Depth	No solution found	No solution found
Nodes generated	No solution found	No solution found
Nodes Expanded	No solution found	No solution found
Max Frontier Size	No solution found	No solution found

Analysis and Insights
Finding the Best Solution: Both BFS and IDS found the best possible solution (the one with the fewest moves). This is expected. BFS checks all the closest possibilities first, so it's guaranteed to find the shortest path. IDS does the same thing, but by searching a little deeper each time until it stumbles upon the goal.

Memory vs. Time: BFS keeps track of every single possible move it could make next. The frontier can get huge. The benefit is that it never checks the same state twice, so it doesn't waste time re-doing work. IDS is memory efficient. It only has to remember the single path it's currently exploring. In the results, its max frontier size is tiny compared to BFS. The downside is that it's repetitive. It restarts its search over and over every time it deepens the search. This is why it is generating and expanding way more nodes.

For a puzzle like this, I think that BFS is better. If it were a bigger problem though, you should switch to IDS as to not have a problem with memory

Addendum: Raw Text Outputs

```
cboogie@ConnorLaptop:~/ai-assignment-1-search$ python3 -m run --domain wgc
--algo ids
------------------------------------------------------------------------
Domain: WGC | Algorithm: IDS
Solution cost: 7 | Depth: 7
Nodes generated: 211 | Nodes expanded: 100 | Max frontier: 7
Path:
  1) Move Goat      ('L', 'L', 'L', 'L') -> ('R', 'L', 'R', 'L')
  2) Move Alone     ('R', 'L', 'R', 'L') -> ('L', 'L', 'R', 'L')
  3) Move Wolf      ('L', 'L', 'R', 'L') -> ('R', 'R', 'R', 'L')
  4) Move Goat      ('R', 'R', 'R', 'L') -> ('L', 'R', 'L', 'L')
  5) Move Cabbage   ('L', 'R', 'L', 'L') -> ('R', 'R', 'L', 'R')
  6) Move Alone     ('R', 'R', 'L', 'R') -> ('L', 'R', 'L', 'R')
  7) Move Goat      ('L', 'R', 'L', 'R') -> ('R', 'R', 'R', 'R')
------------------------------------------------------------------------

cboogie@ConnorLaptop:~/ai-assignment-1-search$ python3 -m run --domain wgc
--algo bfs
------------------------------------------------------------------------
Domain: WGC | Algorithm: BFS
Solution cost: 7 | Depth: 7
Nodes generated: 10 | Nodes expanded: 9 | Max frontier: 2
Path:
  1) Move Goat      ('L', 'L', 'L', 'L') -> ('R', 'L', 'R', 'L')
  2) Move Alone     ('R', 'L', 'R', 'L') -> ('L', 'L', 'R', 'L')
  3) Move Wolf      ('L', 'L', 'R', 'L') -> ('R', 'R', 'R', 'L')
  4) Move Goat      ('R', 'R', 'R', 'L') -> ('L', 'R', 'L', 'L')
  5) Move Cabbage   ('L', 'R', 'L', 'L') -> ('R', 'R', 'L', 'R')
  6) Move Alone     ('R', 'R', 'L', 'R') -> ('L', 'R', 'L', 'R')
  7) Move Goat      ('L', 'R', 'L', 'R') -> ('R', 'R', 'R', 'R')
------------------------------------------------------------------------

cboogie@ConnorLaptop:~/ai-assignment-1-search$ python3 -m run --domain
wgc-sheep --algo bfs
------------------------------------------------------------------------
Domain: WGC-SHEEP | Algorithm: BFS
No solution found.
------------------------------------------------------------------------

cboogie@ConnorLaptop:~/ai-assignment-1-search$ python3 -m run --domain
wgc-sheep --algo ids
------------------------------------------------------------------------
Domain: WGC-SHEEP | Algorithm: IDS
No solution found.
------------------------------------------------------------------------
```