Wolf-Goat-Cabbage (WGC) — Combined Report

fractal13

2025-09-01

Contents

Github Repository: https://github.com/fractal13/ut-cs4300-202540-simple-search

This document consolidates the PEAS analysis, problem formulation, summarized results table, search analysis, and raw results for the Wolf-Goat-Cabbage puzzle as used in this repository.

PEAS Assessment

This section summarizes the PEAS (Performance, Environment, Actuators, Sensors) evaluation for the WGC problem.

- Performance measure
  - Success: all three items and the farmer end on the goal bank.
  - Safety: avoid any state where an item is eaten (wolf with goat alone, goat with cabbage alone).
  - Efficiency: minimize number of crossings (steps).
  - Robustness: handle invalid actions gracefully and avoid repeated unnecessary moves.
- Environment
  - Deterministic and fully observable.
  - Discrete and finite state space (positions for farmer, wolf, goat, cabbage).
  - Episodic; static between actions; single-agent.
- Actuators

  - Farmer crosses alone or with one item (wolf, goat, or cabbage). Actions toggle banks accordingly.
- Sensors
  - Observe banks (left/right) of farmer, wolf, goat, cabbage and detect unsafe configurations.

Problem model

This section describes how the problem is modeled for search.

- State representation
  - Tuple: (farmer, wolf, goat, cabbage) with each value in {L, R}.
  - Unsafe states (predator-prey alone without farmer) are excluded.
- Initial state
  - By convention: (L, L, L, L) — all on the left bank.
- Actions
  - Cross alone, cross with wolf, cross with goat, cross with cabbage.
  - Preconditions: farmer and any transported item must be on same bank; boat capacity <= 1 item.
- Transition model
  - Deterministic toggle of farmer and transported item; successors that are unsafe are discarded.
- Goal test
  - All entities on the goal bank: (R, R, R, R).
- Cost function
  - Unit step cost per action (classical formulation). Other costings possible but not used here.

Results table (summary)

Domain	Algorithm	Solution Cost	Depth	Nodes Generated	Nodes Expanded	Max Frontier
WGC	BFS	7.0	7	26	9	2
WGC	BFS	4.0	4	26	9	3
WGC	BFS	6.0	6	26	9	3
WGC	IDS	7.0	7	304	211	10
WGC	IDS	4.0	4	90	69	7
WGC	IDS	6.0	6	275	203	10

Note: these statistics are copied from the raw results. See Discussion below about frontier measurement methodology.

Analysis: BFS vs IDS

- Expected behavior

  - BFS finds optimal-depth solutions for unit-cost steps; requires more frontier memory.
  - IDS finds optimal-depth solutions with lower memory (stack depth) but generates more nodes due to repeated searches.
- Observed behavior
  - Both algorithms found optimal-depth solutions in the canonical problems.
  - BFS generated fewer total node visits and ran slightly faster in these small instances; IDS generated many more nodes due to repeated depth-limited iterations.
  - Reported "Max Frontier" values show IDS with larger peak frontiers than BFS in these logs; this is likely due to inconsistent measurement (see caveat below).
- Notable points
  - Ensure consistent duplicate-state handling across algorithms when comparing node counts.
  - When measuring peak memory, define whether you count only frontier, frontier+explored, recursion stack, or aggregate per-iteration metrics; inconsistent definitions lead to misleading comparisons.

Measurement caveat

The reported "Max Frontier" values were copied from the raw outputs. If you plan to use peak memory as a metric, ensure a consistent definition across algorithms (frontier only vs frontier+explored vs recursion stack). The current logs suggest IDS reports higher frontier peaks than BFS, which may reflect differences in measurement rather than true memory use.

Conclusion

This consolidated report brings together the PEAS assessment, problem model, summarized stats, analysis of algorithmic behavior, and the raw run outputs. For further work, consider standardizing measurement code for frontier/peak memory and rerunning experiments to ensure apples-to-apples comparisons.

Addendum: Raw results

The following are the original raw run outputs used to build the summary table.

BFS Reports

```
Domain: WGC | Algorithm: BFS
Solution cost: 7.0 | Depth: 7
Nodes generated: 26 | Nodes expanded: 9 | Max frontier: 2
Path:
  1) Move Goat       (L,L,L,L) -> (R,L,R,L)
```

```
  2) Return 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) Return alone    (R,R,L,R) -> (L,R,L,R)
  7) Move Goat       (L,R,L,R) -> (R,R,R,R)

Domain: WGC | Algorithm: BFS
Solution cost: 4.0 | Depth: 4
Nodes generated: 26 | Nodes expanded: 9 | Max frontier: 3
Path:
  1) Move Goat       (R,R,R,L) -> (L,R,L,L)
  2) Move Cabbage    (L,R,L,L) -> (R,R,L,R)
  3) Return alone    (R,R,L,R) -> (L,R,L,R)
  4) Move Goat       (L,R,L,R) -> (R,R,R,R)

Domain: WGC | Algorithm: BFS
Solution cost: 6.0 | Depth: 6
Nodes generated: 26 | Nodes expanded: 9 | Max frontier: 3
Path:
  1) Return alone    (R,L,R,L) -> (L,L,R,L)
  2) Move Wolf       (L,L,R,L) -> (R,R,R,L)
  3) Move Goat       (R,R,R,L) -> (L,R,L,L)
  4) Move Cabbage    (L,R,L,L) -> (R,R,L,R)
  5) Return alone    (R,R,L,R) -> (L,R,L,R)
  6) Move Goat       (L,R,L,R) -> (R,R,R,R)
```

IDS Reports

```
Domain: WGC | Algorithm: IDS
Solution cost: 7.0 | Depth: 7
Nodes generated: 304 | Nodes expanded: 211 | Max frontier: 10
Path:
  1) Move Goat       (L,L,L,L) -> (R,L,R,L)
  2) Return 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) Return alone    (R,R,L,R) -> (L,R,L,R)
  7) Move Goat       (L,R,L,R) -> (R,R,R,R)

Domain: WGC | Algorithm: IDS
Solution cost: 4.0 | Depth: 4
Nodes generated: 90 | Nodes expanded: 69 | Max frontier: 7
Path:
  1) Move Goat       (R,R,R,L) -> (L,R,L,L)
  2) Move Cabbage    (L,R,L,L) -> (R,R,L,R)
```

```
  3) Return alone    (R,R,L,R) -> (L,R,L,R)
  4) Move Goat       (L,R,L,R) -> (R,R,R,R)

Domain: WGC | Algorithm: IDS
Solution cost: 6.0 | Depth: 6
Nodes generated: 275 | Nodes expanded: 203 | Max frontier: 10
Path:
  1) Return alone    (R,L,R,L) -> (L,L,R,L)
  2) Move Wolf       (L,L,R,L) -> (R,R,R,L)
  3) Move Goat       (R,R,R,L) -> (L,R,L,L)
  4) Move Cabbage    (L,R,L,L) -> (R,R,L,R)
  5) Return alone    (R,R,L,R) -> (L,R,L,R)
  6) Move Goat       (L,R,L,R) -> (R,R,R,R)
```