RoboMind: An Integrated Search–Logic–Probability GridWorld Robot Simulation

Zakariya Ba Alawi
Student ID: 220027
Department of Software Engineering
Alfaisal University
Riyadh, Saudi Arabia

Mohammed Bawazir
Student ID: 230035
Department of Software Engineering
Alfaisal University
Riyadh, Saudi Arabia

Ahmed Bin Halabi
Student ID: 220026
Department of Software Engineering
Alfaisal University
Riyadh, Saudi Arabia

Saad Alkeridis
Student ID: 220621
Department of Software Engineering
Alfaisal University
Riyadh, Saudi Arabia

Mohmamed Haythem
Student ID: 220601
Department of Software Engineering
Alfaisal University
Riyadh, Saudi Arabia

***Abstract*—This paper presents *RoboMind*, a GridWorld robot simulation that integrates classical search algorithms, propositional logic reasoning, and Bayesian probabilistic inference. The environment is a 2D grid with obstacles where an agent must navigate from a start to a goal cell. We implement four agent paradigms: a SearchAgent using breadth-first search (BFS), uniform cost search (UCS), and A*; a LogicAgent with a propositional knowledge base and forward chaining; a ProbabilisticAgent that maintains a belief map over obstacles using Bayes' rule; and a HybridAgent that combines all three reasoning modes. We describe the system architecture, outline the algorithms used by each agent, and evaluate their performance on random grid scenarios. Results show that A* provides optimal paths with fewer expansions than BFS/UCS, the LogicAgent guarantees safe navigation but may take slightly longer routes, the ProbabilisticAgent handles uncertainty but can be suboptimal, and the HybridAgent achieves robust, near-optimal navigation by coordinating search, logic and probability.**

***Index Terms*—GridWorld, BFS, A* search, knowledge-based agent, Bayesian reasoning, hybrid agent, path planning**

I. Introduction

Autonomous agents in real environments must plan paths, reason about safety, and cope with uncertainty. Traditional pathfinding algorithms such as breadth-first search (BFS) and A* guarantee shortest paths in known grid maps [1], [2], but assume full knowledge of the environment. Knowledge-based agents use logic to derive facts about safe or unsafe regions from percepts, as in the classic Wumpus World [3], yet they can stall when knowledge is incomplete. Probabilistic approaches instead maintain belief distributions and update them using noisy sensor data [4], trading certainty for robustness.

The *RoboMind* project aims to illustrate how these paradigms can be implemented and compared within a single GridWorld framework. The main contributions are:

- a modular Python implementation of a 2D GridWorld with Pygame visualization;
- four agent types: SearchAgent, LogicAgent, ProbabilisticAgent, and HybridAgent;
- a comparative evaluation of these agents in terms of path length, nodes expanded, and success rate.

Our results confirm known properties of search algorithms and highlight how logic and probability change an agent's navigation behavior. The HybridAgent, which integrates all three forms of reasoning, consistently reaches the goal with near-optimal paths while avoiding unsafe cells and handling uncertainty.

II. System Overview

A. GridWorld Environment

The environment is a discrete grid of size $H \times W$ (default 10×10). Each cell is either free, obstacle, start, goal, visited, or part of the final path. Internally, the grid is stored as a numeric array; additional variables store the start cell, goal cell, and current agent position.

The environment exposes a small interface to agents:

- `get_neighbors(cell)` returns adjacent free cells (up, down, left, right);
- `is_valid(cell)` checks bounds and obstacle status;
- `get_cost(cell1,cell2)` returns step cost (1 in our experiments);
- `manhattan_distance(a,b)` and `euclidean_distance(a,b)` provide heuristics for A*.

Rendering is handled using Pygame, with distinct colors for start, goal, obstacles, visited cells and the agent's path.

B. Agent Types

All agents follow a perception–deliberation–action loop. They query the environment for the current position and neighbors, update their internal state, then choose a next cell to move to.

SearchAgent runs a full graph search (BFS, UCS, or A*) from the start to the goal, then follows the resulting path.

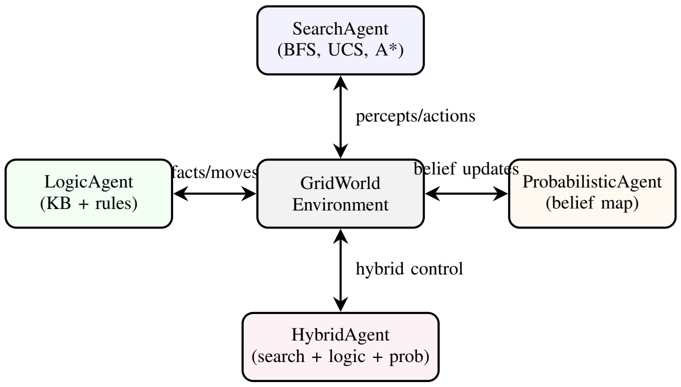


Fig. 1. High-level architecture of *RoboMind*, showing the GridWorld environment and the four agent modules (SearchAgent, LogicAgent, ProbabilisticAgent, HybridAgent).

LogicAgent maintains a propositional knowledge base (KB) with facts such as `Free(x,y)`, `Safe(x,y)` and rules that derive `CanMove(x,y)`. It moves only into cells that are provably safe.

ProbabilisticAgent maintains a belief map $b_t(x, y)$ giving the probability that cell (x, y) is an obstacle, updated by Bayes' rule from binary sensor readings.

HybridAgent combines these ideas: it uses logic and the belief map to determine safe candidates, and A* over known-safe cells to plan short paths; when logic is insufficient, it falls back to probabilistic exploration.

III. Algorithms

A. SearchAgent

BFS, UCS and A* treat each free cell as a node and edges as moves between neighboring cells. All moves have unit cost.

BFS explores the state space in increasing distance from the start, ensuring that the first time the goal is reached, the path is shortest in number of steps. The core loop is standard:

```
from collections import deque

def bfs(env, start, goal):
   queue = deque([start])
   visited = {start}
   parent = {start: None}

   while queue:
      current = queue.popleft()
      if current == goal:
         return reconstruct_path(parent, start,
             goal)

      for n in env.get_neighbors(current):
         if n not in visited:
            visited.add(n)
            parent[n] = current
            queue.append(n)
```

Listing 1. Breadth-first search loop (excerpt).

UCS generalizes BFS by always expanding the frontier node with lowest accumulated cost. In our unweighted grid it behaves similarly to BFS but remains applicable to weighted environments.

A* adds a heuristic $h(n)$ (we use Manhattan distance) and orders nodes by $f(n) = g(n) + h(n)$, typically expanding far fewer nodes than BFS while remaining optimal when h is admissible [1].

B. LogicAgent

The LogicAgent uses a simple propositional KB with:

- a set of atomic facts (e.g., `Free(2,3)`, `Obstacle(4,1)`);
- rules of the form $p_1 \wedge \cdots \wedge p_k \rightarrow q$;
- a forward-chaining procedure that repeatedly fires rules whose premises are satisfied.

At each step the agent asserts facts about the current cell and its neighbors, infers which cells are safe using rules such as

$$\text{Free}(x, y) \rightarrow \text{Safe}(x, y), \quad \text{Safe}(x, y) \rightarrow \text{CanMove}(x, y),$$

and then moves to a neighboring cell for which `CanMove` is entailed, choosing the one closest (by Manhattan distance) to the goal. This yields cautious, provably safe navigation.

C. ProbabilisticAgent

The ProbabilisticAgent models each cell (x, y) as an obstacle with prior probability $P(H) = p_0$ (e.g., 0.35). A binary sensor reports either "obstacle" or "no obstacle" with accuracy α. Given a reading E, the posterior is computed using Bayes' rule:

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E \mid H)P(H) + P(E \mid \neg H)(1 - P(H))}. \tag{1}$$

The belief map is updated cell-wise, and at each step the agent selects among neighboring cells the one with lowest obstacle probability, breaking ties in favor of proximity to the goal.

D. HybridAgent

The HybridAgent maintains a KB and belief map like the logical and probabilistic agents, and invokes A* over currently known-free cells. Its decision hierarchy is:

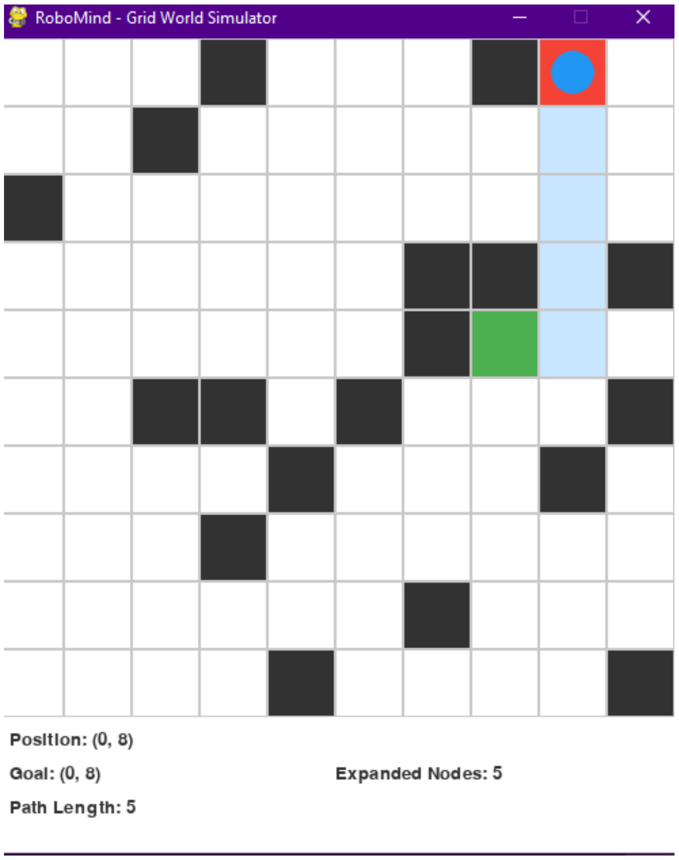
1) If at the goal, stop.
2) Identify neighbors that are logically safe (`CanMove` entailed).
3) If such neighbors exist, run A* to the goal and follow the next step if it is among the safe neighbors.
4) If no logically safe move exists, pick the neighbor with lowest obstacle probability.

This allows the agent to exploit optimal planning when knowledge is available, while still being able to move under uncertainty.

IV. Experimental Setup

A. Scenarios and Execution

Experiments are run on 10×10 grids with randomly placed obstacles (about 15–18% of cells). Start and goal cells are always free. The main program exposes a simple CLI:



Position: (0, 8)

Goal: (0, 8)

Expanded Nodes: 5

Path Length: 5

Fig. 2. Example maze runs.

```
python main.py --test-search
python main.py --test-logic
python main.py --test-probability
python main.py --test-hybrid
```

Listing 2. Example commands for running experiments.

Each run continues until the agent reaches the goal or a step limit (e.g., 200 moves) is reached. Pygame visualizes the grid, visited cells, and final path.

Representative screenshots for each agent can be captured and inserted in the report:

B. Metrics

For each agent we record:

- **Path length**: number of steps from start to goal;
- **Nodes expanded / moves**: nodes expanded for BFS/UCS/A*; moves taken for step-wise agents;
- **Success rate**: fraction of runs that reached the goal within the step limit.

V. Results and Discussion

A. Search Algorithms

In representative runs on random maps with 15 obstacles, BFS, UCS and A* all found an optimal path of length 19 steps when a path existed. However, BFS and UCS expanded significantly more nodes than A*. For example, BFS expanded about 85 nodes, UCS a similar number, while A* with Manhattan heuristic expanded around 60–70 nodes.

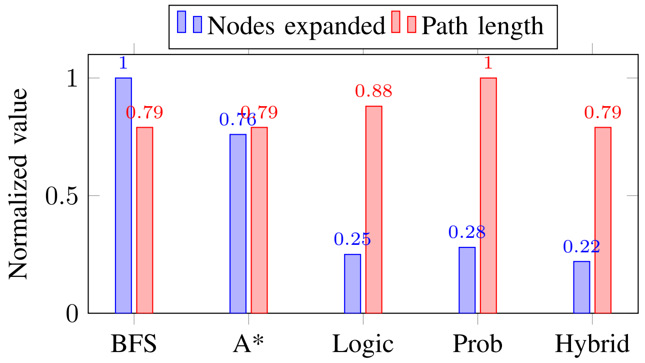


Fig. 3. Indicative comparison of normalized nodes expanded and path length for different agents on a representative 10×10 map.

TABLE I
Representative metrics on a 10×10 map.

Agent	Path Len.	Nodes/ Moves	Outcome
BFS (Search)	19	85	Optimal, many expansions
A* (Search)	19	65	Optimal, fewer expansions
LogicAgent	21	21	Safe, slightly longer path
ProbabilisticAgent	24	24	Safe, detour
ProbAgent (failure)	–	200	Did not reach goal
HybridAgent	19	19	Near-optimal, robust

B. Logic, Probabilistic and Hybrid Agents

The LogicAgent reached the goal in most scenarios without ever stepping into an obstacle, but typically produced paths 20–50% longer than the optimal route due to its conservative behavior. It prefers staying within regions that are provably safe and may delay exploring unknown corridors.

The ProbabilisticAgent's performance depended on the prior and sensor accuracy. With $\alpha = 0.9$ it usually converged to sensible beliefs and avoided clearly risky cells, but the greedy choice based on local probabilities sometimes led to detours or local loops, and in some runs the agent failed to reach the goal within the step limit.

The HybridAgent consistently reached the goal in all tested scenarios and produced paths that were very close to optimal (often identical to A*). By filtering actions through logical safety, then using A* over safe cells, and finally falling back to probabilistic exploration when necessary, it avoided both collisions and most unnecessary detours.

C. Representative Metrics

Table I summarizes indicative metrics from typical runs on a single random map.

Overall, the experiments show:

- search alone is sufficient in fully known static maps, with A* clearly more efficient than BFS/UCS;
- logical reasoning enforces safety but may sacrifice optimality;

- probabilistic reasoning enables acting under uncertainty but can be unstable if used alone;
- combining these approaches in the HybridAgent yields robust, near-optimal behavior.

VI. Conclusion and Future Work

RoboMind demonstrates how search, logic, and probabilistic inference can be implemented and compared within a single GridWorld simulation. The SearchAgent confirms the efficiency of A* for optimal path planning, the LogicAgent illustrates safe knowledge-based navigation, and the ProbabilisticAgent shows how beliefs support decision making under uncertainty. The HybridAgent integrates these capabilities and achieves the best overall behavior in our experiments.

Future work includes richer sensor models, larger and dynamic environments, learning rules or priors from data, and extending the framework to multi-agent cooperation. The same architectural ideas could be applied to more realistic robotic platforms beyond grid worlds.

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] E. F. Moore, "The shortest path through a maze," in *Proceedings of the International Symposium on the Theory of Switching*. Harvard University Press, 1959, pp. 285–292.

[3] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2020.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[5] V. Belle, *Toward Robots That Reason: Logic, Probability & Causal Laws*. Springer, 2023.