COMP3308

Table of Contents

1. Introduction to AI & Rational Agents

Introduction to AI

1. What is AI?

AI is defined by whether it focuses on Thinking vs. Acting and Human-like vs. Rational (ideal) behavior.

1. Think like humans (Cognitive approach): Models how the human brain works.
2. Act like humans (Behavioral approach): Passing the Turing Test.
3. Think rationally (Logical approach): Using formal logical / laws of thought (struggle with uncertainty).
4. Act rationally (Rational Agent approach).

2. The Rational Agent

- Intelligent Agent: Has knowledge and goals, perceives its environment, and acts to achieve goals.
- Rationality: Doing the action that maximizes a performance measure.
- Limited Rationality: Doing the best possible given computational limits (time/memory).

3. The Turing Test (Acting Humanly)

- Goal: Interrogator cannot tell if responses are from a human or machine.
- 4 Required Capabilities:
  1. NLP (Natural Language Processing): To communicate.
  2. Knowledge Representation: To store information.
  3. Automated Reasoning: To answer questions / draw conclusions.
  4. Machine Learning (ML): To adapt to new situations.
- Total Turing Test adds: Computer Vision (seeing) & Robotics (moving).
- Reverse Turing Test: CAPTCHA (blocks bots).

4. Main AI Subfields

- Search & Problem Solving: Finding optimal paths (ex. GPS navigation).
- Game Playing (Adversarial Search): AI guessing best moves under time limits.
- Machine Learning (ML):
  - Supervised: Learning from pre-classified data to predict labels.
  - Unsupervised: Grouping unlabelled data by similarity.
- Probabilistic Reasoning: Predicting outcomes under uncertainty.
- Deep Learning: Complex neural networks.

AI limitations & Cautions (GenAI / LLMs)

- Hallucinations: AI can give confident but completely incorrect answers.
- Bias: AI is only as smart / fair as its training data.
- Privacy Risks: Can expose personal data.

Search Strategies

1. Search Problem Formulation
A search problem requires 4 components:

1. Initial State: Where you start.
2. Goal State: Where you want to end up (can be explicit or a test).
3. Operators (Actions): Valid moves between states.
4. Path Cost: Numeric value for a path (ex. distance, time).

- Solution: A path from initial to goal state. Optimal = lowest cost.
- State vs. Node: A state is a configuration (ex. a chessboard). A node is a data structure in a search tree containing the state, parent, child, depth, and path cost $g(n)$.

2. Evaluating Search Strategies
We judge search algorithms using 4 criteria & 3 variables:

- Completeness: Is it guaranteed to find a solution if one exists?
- Optimality: Does it find the lowest-cost path?
- Time Complexity: How many nodes are generated?
- Space Complexity: Max memory required.
  (b = max branching factor, d = depth of optimal solution, m = max depth of state space)

3. Uninformed (Blind) Search
Uses NO domain knowledge / hints about how far the goal is.

- **Breadth-First Search (BFS)**
  - Expands the shallowest unexpanded node first (level by level).
  - Fringe: Insert children at the end.
  - Complete? Yes.
  - Optimal? Only if all step costs are equal.
  - Time & Space: $O(b^d)$ (Exponential). Space is a massive problem!
- **Uniform Cost Search (UCS)**
  - Expands nodes with the lowest path cost $g(n)$ from the root.
  - Fringe: Ordered by path cost.
  - Complete & Optimal? Yes to both.
  - Equivalent to BFS if step costs are constant.
- **Depth-First Search (DFS)**
  - Expands the deepest unexpanded node first.
  - Fringe: Insert children at the front.
  - Complete? No (can get stuck in infinite loops / paths).
  - Optimal? No.
  - Time: $O(b^m)$
  - Space: $O(bm)$ (Linear = Excellent memory efficiency!)
- **Depth-Limited Search (DLS)**
  - DFS with a strict depth limit (l). Prevents infinite loops, but you must guess l.
- **Iterative Deepening Search (IDS)** - Best of both worlds!
  - Runs DLS repeatedly, increasing the depth limit ($l = 0, 1, 2, \ldots$)
  - Combines benefits: Get the Linear Space of DFS $O(bd)$ with the Completeness / Optimality of BFS. Time is $O(b^d)$.
  - Overhead is tiny because most nodes are at the bottom of the tree anyway.

4. Informed (Heuristic) Search
Uses problem-specific knowledge to guess how close a node is to the goal.

- Heuristic $h(n)$: Estimated cost from node n to the goal. (For a goal node, $h(goal) = 0$). Ex. straight-line distance on a map.
- **Greedy Best-First Search**
  - Expands the node that appears closest to the goal (lowest $h(n)$).

- Complete? No (can get stuck in loops).
- Optimal? No (often takes sub-optimal shortcuts).
- Time & Space: $O(b^m)$ in the worst case, but a good heuristic dramatically improves real-world speed.

A* Search Algorithm (Informed Search)
A* combines Uniform Cost Search (UCS) and Greedy Search.

- Evaluation Function: $f(n) = g(n) + h(n)$
  - $g(n)$ = Cost so far (from start to node n)
  - $h(n)$ = Estimated cost to goal (heuristic)
  - $f(n)$ = Estimated total path cost
- Relation to Uninformed Search:
  - If $h(n) = 0 \rightarrow$ A* becomes UCS.
  - If $h(n) = 0$ AND step costs are equal ($g(n) = \text{depth}$) $\rightarrow$ A* becomes BFS.

Heuristics in A*

- **Admissible Heuristics (Optimistic)**
  - Never overestimates the true cost to the goal: $h(n) \leq h^*(n)$
  - Theorem: If $h(n)$ is admissible, A* is Complete and Optimal.
- **Consistent (Monotonic) Heuristics**
  - Satisfies the Triangle Inequality: $h(n_i) \leq cost(n_i, n_j) + h(n_j)$
  - $f(n)$ never decrease along any path.
  - Rule: Consistent $\Rightarrow$ Admissible (but Admissible does not imply Consistent).
  - Theorem: If $h(n)$ is consistent, A* is Optimally Efficient (no other optimal algorithm expands fewer nodes).
- **Dominant Heuristics**
  - If $h_2(n) \geq h_1(n)$ for all nodes, h_2 dominates h_1.
  - Dominant heuristics are better because they expand fewer nodes.
  - Combine heuristics using $h(n) = \max(h_1, h_2)$.
- **Inventing Heuristics**
  - Create a Relaxed Problem (remove restrictions from the rules). The exact solution to a relaxed problem is an admissible heuristic for the real problem.
- Space Complexity is exponential $O(b^d)$. It runs out of memory long before it runs out of time.