Fundamentals of Artificial Intelligence: Definitions and Algorithms

This study guide provides highly condensed notes focusing on the core definitions, algorithms, performance metrics, and key comparisons required to tackle MCQ and fill-in-the-blank questions efficiently.

--------------------------------------------------------------------------------

I. AI Foundations and Rational Agents

1. Defining AI [1]

|  |  |  |
| --- | --- | --- |
| Approach | Focus | Key Concept |
| **Acting Humanly** | The **Turing Test** (Imitation Game) [2]. Requires knowledge, reasoning, language understanding, and learning [3]. | Problem: Test is biased, not reproducible, or mathematically amenable [3]. |
| **Thinking Humanly** | Cognitive Science/Neuroscience [4]. Based on predicting behavior (top-down) or neurological data (bottom-up) [5]. | Requires scientific theories of internal brain activities [4]. |
| **Thinking Rationally** | Laws of Thought (Logic) [6]. Focuses on **correct argument/thought processes** [6]. | Problem: Not all intelligent behavior is mediated by logical deliberation [6]. |
| **Acting Rationally** | **Rational Agents**. This is the focus of the course [1, 2]. | Agent chooses action maximizing expected **goal achievement** [1, 7, 8]. |

2. Rationality and Agents

• **Rational Agent:** Chooses the action maximizing the expected value of the **performance measure** [7, 8].

• **Rationality $\ne$ Omniscient/Clairvoyant:** Agents may not have all relevant information (**percepts**) or action outcomes may not be expected [8, 9].

• **Rationality $\implies$ Exploration, learning, and autonomy** [8, 9].

• **Agent Definition:** Anything that perceives (via **sensors**) and acts (via **actuators**) [10, 11].

• **Agent Function:** Abstractly, $f : P^\* \to A$ (maps **percept histories** to **actions**) [2, 11, 12].

3. PEAS Framework

Used to specify the **task environment** [10, 13].

|  |  |
| --- | --- |
| Component | Example (Automated Taxi) [14, 15] |
| **P**erformance measure | Safety, profits, comfort, legality. |
| **E**nvironment | US streets/freeways, traffic, weather. |
| **A**ctuators | Steering, accelerator, brake, horn. |
| **S**ensors | Video, accelerometers, GPS. |

4. Agent Types (in order of increasing generality) [16]:

1. **Simple reflex agents**

2. **Goal-based agents**

3. **Utility-based agents**

4. **Learning agents**

--------------------------------------------------------------------------------

II. Uninformed and Informed Search

1. Search Problem Fundamentals

• **Goal:** Find a sequence of actions (path) in a fully observable, deterministic, discrete, static environment [17].

• **Components:** Initial state, Actions (transition model), Goal state, and **Path Cost** (sum of nonnegative step costs; optimal solution minimizes this) [18, 19].

• **State vs. Node:** A **State** is a description of the world. A **Node** is a data structure in the search tree (containing state ID, parent, and path cost) [20, 21].

• **Complexity Metrics:** $b$ (branching factor), $d$ (depth of optimal solution), $m$ (maximum path length) [22].

2. Uninformed Search Strategies [19, 23, 24]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Strategy | Complete? | Optimal? | Time/Space Complexity | Frontier (Data Structure) |
| **Breadth-First Search (BFS)** | **Yes** | **Yes** (if step costs are equal) | $O(b^d)$ (Exponential space is the issue) | **Queue (FIFO)** [25] |
| **Depth-First Search (DFS)** | No (Fails in loops/infinite depth) | No | $O(b^m)$ (Worst Case) | **Stack (LIFO)** [26] |
| **Uniform Cost Search (UCS)** | **Yes** | **Yes** (Lowest Cost Path) | $O(\text{Nodes with } g(n) \le C^\*)$ | **Priority Queue** (sorted by $g(n)$ or $V\_n$—cost to arrive) [27] |

• **UCS/Dijkstra's:** They evaluate the same nodes in the same order and yield the same minimum-cost path [28].

3. Informed Search (Heuristics)

• **Heuristic ($h(n)$):** An estimate of the distance-to-goal that is **cheap** to compute (i.e., less than $O(b^d)$) [29]. E.g., Manhattan distance, Euclidean distance [30, 31].

• **Greedy Best-First Search:** Expands node that appears **closest to the goal** using $h(n)$ only [32, 33]. Neither complete nor optimal [24].

• **A\* Search:** Traverses the graph based on the **lowest expected total cost** [33, 34].

    ◦ **Evaluation Function:** $f(n) = g(n) + h(n)$ [31]. \* $g(n)$: Past path cost (known cost from start) [34]. \* $h(n)$: Future path cost (heuristic estimate) [34].

    ◦ **Admissibility:** Required for optimality. $h(x)$ must **never overestimate** the true cost to the goal: **$h(x) \le d(x, goal)$** [34, 35].

    ◦ **Dominance:** If $h\_2(n) \ge h\_1(n)$ and both are admissible, $h\_2$ dominates $h\_1$ and requires A\* to expand **fewer nodes** [36].

    ◦ **Combining Admissible Heuristics:** Use the maximum: $h(n) = \max{h\_1(n), h\_2(n), \ldots}$ [37].

4. Local Search

• **Use Case:** Optimization problems (like TSP, N-Queens) where the **path is irrelevant**; the goal state itself is the solution [38].

• **Simulated Annealing (SA):** Allows for "bad" (downward) moves with a probability $P=e^{-\Delta E/kT}$ (where $T$ is temperature) to escape local maxima [39, 40]. Inspired by forging iron [41].

--------------------------------------------------------------------------------

III. Adversarial Search (Games)

1. Games vs. Search

• **Solution:** A **strategy or policy** (a mapping from state to the best move), not a fixed sequence of actions, because the opponent's actions are unknown [42].

• **Environment:** Typically Alternating two-player **Zero-sum games** (Max's gain is Min's loss) [42, 43].

• **Complexity:** The search depth is severely limited (e.g., Chess $\approx 10^{154}$ nodes total) [44].

2. Minimax and Pruning

• **Minimax Strategy:** Choose the move that gives the **best worst-case payoff** for Max, assuming the opponent (Min) plays optimally to minimize Max's utility [43, 45]. It is **optimal** against an optimal opponent [46].

• **Alpha-Beta Pruning:** Computes the exact minimax decision without expanding the whole tree [47].

    ◦ **Alpha ($\alpha$):** The highest value MAX knows how to force MIN to accept [48]. If a MIN node's value drops to $\le \alpha$, prune [49].

    ◦ **Beta ($\beta$):** The lowest value MIN knows how to force MAX to accept [50]. If a MAX node's value rises to $\ge \beta$, prune [51].

    ◦ **Efficiency:** With perfect ordering, reduces complexity from $O(b^m)$ to $O(b^{m/2})$, effectively **doubling the search depth** [52].

3. Limited Horizon and Evaluation

• **Limited Horizon:** Search is cut off at a certain depth [53].

• **Evaluation Function:** Used to estimate the value (or probability of winning) of a non-terminal state at the cutoff depth [54]. Often a **weighted sum of features** [54].

• **Horizon Effect:** Incorrectly estimating a state's value by overlooking a critical event just beyond the depth limit [55, 56].

• **Remedy:** **Quiescence Search**—do not cut off search at positions that are unstable (e.g., when a piece is about to be lost) [57].

• **Nondeterministic Games:** Solved using **Expectiminimax**, which handles chance nodes by calculating the average (expected) value of successors [58].

--------------------------------------------------------------------------------

IV. Constraint Satisfaction Problems (CSPs)

1. CSP Fundamentals

• **Purpose:** Search for **assignment** of values to variables while respecting constraints [59].

• **Solution:** A **complete and consistent assignment** [60, 61].

• **Examples:** Map coloring, Sudoku [61, 62].

• **Complexity:** CSPs and SAT are **NP-complete** [63].

• **Backtracking Search:** Depth-First Search tailored for CSPs [64]. We fix the order of assignments because variable assignments are **commutative** [65]. If $N$ variables have $D$ values, total possible paths is $D^N$ [64]. DFS is preferred over BFS for space management [66].

2. Backtracking Heuristics ($O{1}$)

|  |  |  |
| --- | --- | --- |
| Heuristic | Type | Goal/Rule |
| **Least Remaining Values (LRV)** | Variable Ordering | Choose the variable with the **fewest legal values** (minimizes branching factor) [67]. |
| **Most Constraining Variable (MCV)** | Variable Ordering | Choose the variable that imposes the **most constraints** on remaining variables (often a tie-breaker for LRV) [68]. |
| **Least Constraining Value (LCV)** | Value Ordering | Try the value that **rules out the fewest values** in the remaining variables (maximizes success probability) [69]. |

3. Consistency Checking (Early Failure Detection)

• **Forward Checking ($O{N}$):** Check that every unassigned variable still has at least **one possible assignment** remaining. Terminate if any variable domain becomes empty [70].

• **Arc Consistency ($O{N^2}$):** Check that every **pair of variables (arc)** still has a pairwise assignment that satisfies all constraints [71]. **Detects failure earlier than forward checking** [72].