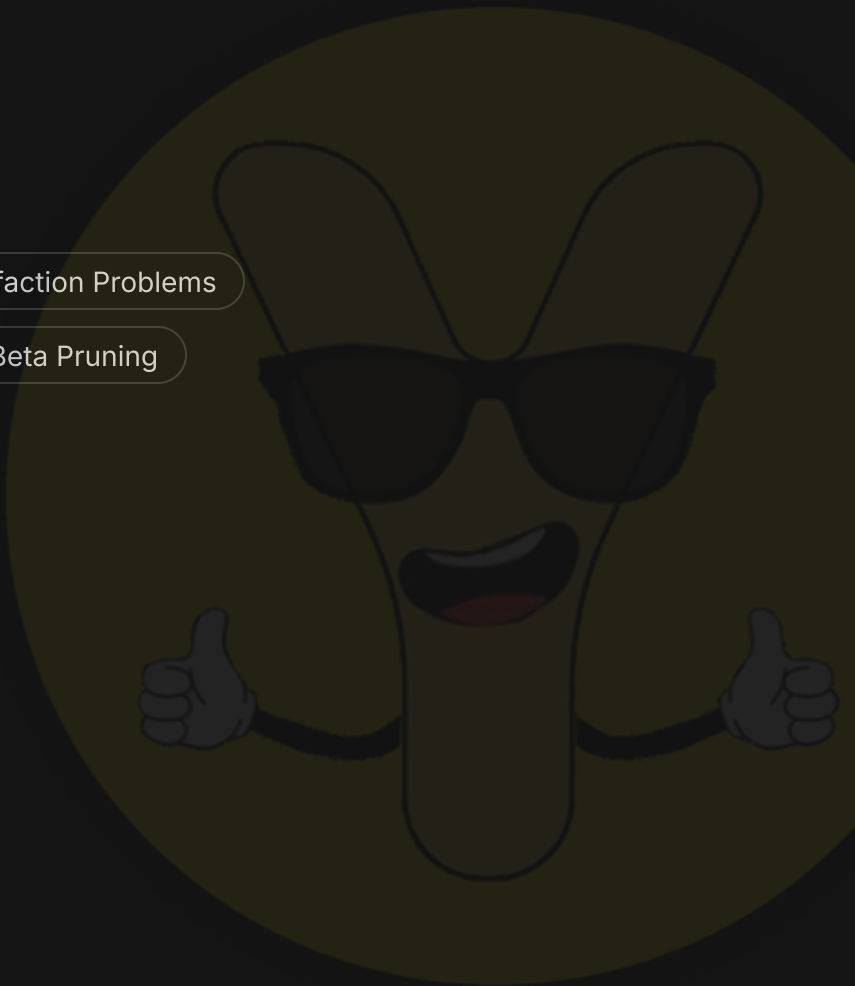
AKTU · 3RD YEAR · SEMESTER 5 · BCAI501

Artificial Intelligence

Unit 2 — Problem Solving Methods

- Problem Solving Methods
- Search Strategies
- Uninformed Search
- Informed Search
- Heuristics
- Local Search Algorithms and Optimization Problems
- Searching with Partial Observations
- Constraint Satisfaction Problems
- Constraint Propagation
- Backtracking Search
- Game Playing
- Optimal Decisions in Games
- Alpha-Beta Pruning
- Stochastic Games



DECODED BY

Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

Problem Solving Methods in AI

Problem solving in AI involves goal-formulated search through a state space.

- State Space: Set of all possible states reachable from the initial state by any sequence of actions.
- Initial State: The state the agent starts in to begin the problem-solving process.
- Actions & Transition Model: Actions(s) returns legal actions; Result(s, a) returns the outcome state.
- Goal Test: Test that determines whether a given state is a goal state.
- Path Cost: Numeric cost assigned to a path, denoted as $c(s, a, s')$.

Search Strategy Evaluation Metrics

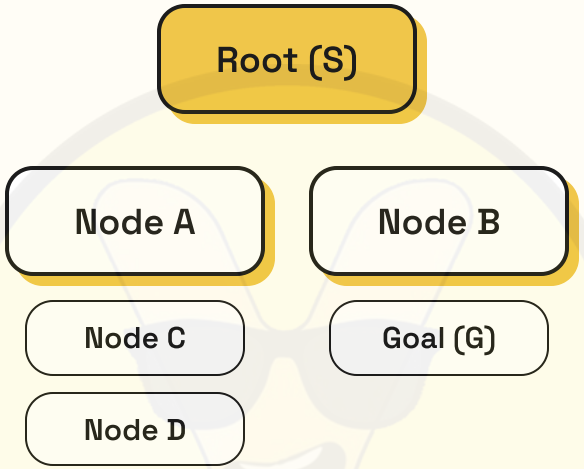
Search algorithms are evaluated using four primary performance metrics.

- Completeness: Does the algorithm guarantee finding a solution if one exists?
- Time Complexity: How long (number of nodes generated) does it take to find a solution?
- Space Complexity: How much memory is needed to perform the search?
- Optimality: Does the strategy find the highest-quality (lowest path cost) solution?
- Complexity Parameters: Expressed using branching factor b, depth d, and max path length m.

Comparison of Uninformed Search Strategies

Algorithm	Complete?	Time Complexity	Space Complexity	Optimal?
BFS	Yes (if b is finite)	$O(b^d)$	$O(b^d)$	Yes (if step costs equal)
DFS	No (in infinite depth)	$O(b^m)$	$O(bm)$	No
Uniform Cost (UCS)	Yes	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	Yes
Depth-Limited (DLS)	No (if cutoff < d)	$O(b^l)$	$O(bl)$	No
Iterative Deepening	Yes	$O(b^d)$	$O(bd)$	Yes (if step costs equal)

Search Tree Structure



Hierarchical expansion of nodes during state space search.

Informed (Heuristic) Search Strategies

INFORMED SEARCH

A search strategy that uses domain-specific knowledge via a heuristic function $h(n)$ to estimate the cost from node n to the goal.

- Greedy Best-First Search: Expands node with lowest $h(n)$; fast but neither complete nor optimal.
- A* Search: Evaluates nodes using $f(n) = g(n) + h(n)$, where $g(n)$ is path cost from start to n.
- A* Optimality: Guaranteed to be optimal if $h(n)$ is admissible for tree search or consistent for graph search.

A* Search Computation Example

Find the optimal path from Start (S) to Goal (G) using A* evaluation function $f(n) = g(n) + h(n)$.

1. Start state S: $g(S) = 0, h(S) = 6 \implies f(S) = 6$. Expand S.
2. S connects to A and B. Path S-A: $g(A) = 2, h(A) = 4 \implies f(A) = 6$. Path S-B: $g(B) = 3, h(B) = 2 \implies f(B) = 5$.
3. Select node B (lowest f value = 5). Expand B to G with step cost 4: $g(G) = 7, h(G) = 0 \implies f(G) = 7$.
4. Select node A ($f(A) = 6$). Expand A to G with step cost 3: $g(G) = 5, h(G) = 0 \implies f(G) = 5$.

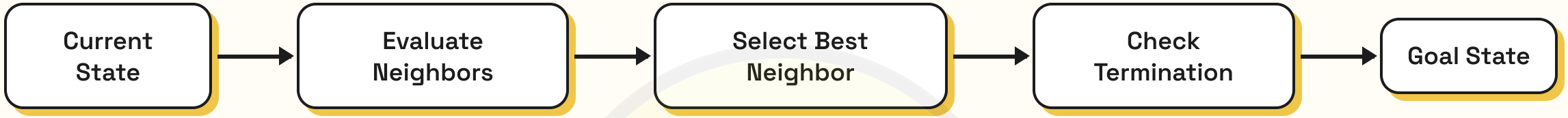
Optimal Path: S → A → G with total path cost $g(G) = 5$.

Heuristic Functions Properties

The performance and optimality of informed search algorithms depend on properties of heuristic functions.

- Admissibility: A heuristic $h(n)$ is admissible if it never overestimates the true cost to reach the goal: $h(n) \leq h^*(n)$.
- Consistency (Monotonicity): $h(n) \leq c(n, a, n') + h(n')$ for every node n and successor n'.
- Dominance: If $h_2(n) \geq h_1(n)$ for all n, then h_2 dominates h_1 and expands fewer nodes.
- Heuristic Construction: Derived by relaxing problem constraints (e.g., Manhattan distance for 8-puzzle).

Local Search Execution Flow



Iterative execution flow of local search optimization algorithms.

Local Search Algorithms and Optimization

Local search operates on full-state representations to find optimal or near-optimal configurations.

- Hill-Climbing Search: Continuously moves in direction of increasing value (steepest ascent); stops at local peak.
- Local Maxima: Peaks higher than surrounding states but lower than the global maximum.
- Plateaus & Ridges: Flat areas or narrow climbing sequences causing hill-climbing search to stall.
- Simulated Annealing: Escapes local maxima by allowing occasional downhill moves with probability $e^{\Delta E/T}$.
- Genetic Algorithms: Population-based search using selection, crossover, and mutation operations.

Searching with Partial Observations

Handling environments with incomplete information requires reasoning over sets of possible states.

- Belief State: Set of all possible actual states the agent could be in given its observation history.
- Sensorless (Conformant) Problems: Agent has no sensors; must find an action sequence reaching goal from any state.
- Partially Observable Problems: Agent receives percepts providing partial info; updates belief state after each action.
- Belief State Space: Searching in space of belief states; size is 2^N where N is the number of physical states.

Constraint Satisfaction Problems (CSP)

CONSTRAINT SATISFACTION PROBLEM

A problem defined by a set of variables, domains of possible values, and constraints restricting valid variable combinations.

- Variables $X = \{X_1, X_2, ..., X_n\}$: Features requiring values.
- Domains $D = \{D_1, D_2, ..., D_n\}$: Sets of allowable values for each variable.
- Constraints $C = \{C_1, C_2, ..., C_m\}$: Rules specifying legal value combinations.
- Goal: Find a complete and consistent assignment satisfying all constraints.

Constraint Propagation & Arc Consistency

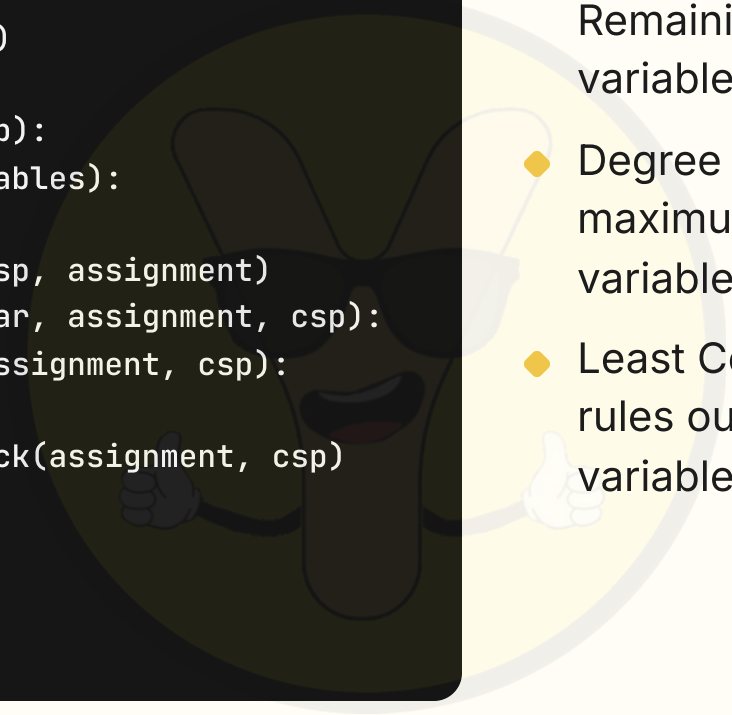
Constraint propagation reduces search space size by enforcing consistency rules before or during search.

- Arc Consistency (AC-3): Variable X_i is arc-consistent with X_j if for every value in D_i, there exists a valid value in D_j.
- Forward Checking: Tracks remaining valid values for unassigned variables when a variable is assigned.
- Node Consistency: Eliminates domain values that violate unary constraints.
- Path Consistency: Enforces constraints on triplets of variables to further prune search space.

Backtracking Search Algorithm for CSP

```
def backtracking_search(csp):
    return recursive_backtrack({}, csp)

def recursive_backtrack(assignment, csp):
    if len(assignment) == len(csp.variables):
        return assignment
    var = select_unassigned_variable(csp, assignment)
    for value in order_domain_values(var, assignment, csp):
        if is_consistent(var, value, assignment, csp):
            assignment[var] = value
            result = recursive_backtrack(assignment, csp)
            if result ≠ None:
                return result
            del assignment[var]
    return None
```



- MRV Heuristic: Selects variable with Minimum Remaining Values ('most constrained variable').
- Degree Heuristic: Selects variable involved in maximum constraints with unassigned variables.
- Least Constraining Value: Chooses value that rules out fewest choices for neighboring variables.

Game Playing and Adversarial Search

Game playing models environments with competitive multi-agent interactions.

- Deterministic & Perfect Information: Zero-sum two-player games like Chess, Checkers, Tic-Tac-Toe.
- Minimax Strategy: Assumes optimal play by opponent; MAX maximizes utility while MIN minimizes MAX's payoff.
- Game Tree: Nodes represent game states; edges represent valid moves between states.
- Utility Function: Computes numeric payoff for terminal states (e.g., +1 for win, -1 for loss, 0 for draw).

Minimax Evaluation Example

Compute Minimax decision for root state MAX with terminal utilities at depth 2: Leaf values [3, 12, 8, 2, 4, 6, 14, 5].

1. Level 2 (Leaves): Grouped in pairs under MIN nodes: (3,12), (8,2), (4,6), (14,5).
2. Level 1 (MIN decisions): MIN picks minimum value of children.
3. MIN Node 1 = min(3, 12) = 3
4. MIN Node 2 = min(8, 2) = 2
5. MIN Node 3 = min(4, 6) = 4
6. MIN Node 4 = min(14, 5) = 5
7. Level 0 (MAX decision at root): MAX picks max(3, 2, 4, 5) = 5.

Optimal decision value at root = 5 (choosing branch leading to MIN Node 4).

Alpha-Beta Pruning Strategy

Alpha-Beta Pruning eliminates subtrees that cannot influence the final Minimax decision.

- Alpha (α): Highest-value choice found so far along path for MAX.
- Beta (β): Lowest-value choice found so far along path for MIN.
- Pruning Rule: Prune remaining subtrees when $\alpha \geq \beta$.
- Time Complexity: Reduces search time from $O(b^m)$ to $O(b^{m/2})$, doubling searchable depth.
- Move Ordering: Pruning efficiency depends heavily on exploring strong moves first.

Alpha-Beta Pruning Example

Demonstrate Alpha-Beta pruning on MIN node with children returning 3 and 2 when parent MAX has $\alpha = 5$.

1. Initial state: MAX node has already discovered a path with value $\alpha = 5$.
2. Search moves to a sibling MIN node with initial $\beta = +\infty$.
3. MIN node evaluates its first child, yielding value 3. MIN updates $\beta = \min(+\infty, 3) = 3$.
4. Compare parameters: $\alpha = 5$ and $\beta = 3$. Condition satisfied: $\alpha \geq \beta$ (5 >= 3).
5. Pruning triggered: Remaining children under this MIN node are pruned because MAX will never select this branch.

Branch pruned successfully without evaluating remaining child nodes.

Stochastic Games and Expectimax

Stochastic games incorporate randomness such as dice rolls or coin flips.

- Chance Nodes: Interspersed between MAX and MIN nodes in the game tree to represent random events.
- Expectimax Algorithm: Calculates Expected Value at chance nodes instead of pure Minimax values.
- Expected Value Formula: $Expectimax(n) = \sum_s P(s) \times Utility(s)$.
- Evaluation Function Sensitivity: Exact numeric values of evaluation functions directly affect Expectimax decisions.

Exam Corner - University Questions

- **Q1** Differentiate between uninformed and informed search techniques with suitable examples.
- **Q2** Define heuristic function. Explain admissibility and consistency conditions for A* search algorithm.
- **Q3** Trace A* search algorithm for an 8-puzzle problem and prove its optimality.
- **Q4** Describe local search algorithms. Explain local maxima, ridge, and plateau problems in Hill Climbing search.
- **Q5** Formulate Map Coloring or N-Queens problem as a CSP. Explain AC-3 algorithm for constraint propagation.
- **Q6** Explain Minimax algorithm for game playing. Trace Alpha-Beta pruning procedure with a numerical example.
- **Q7** Write short notes on Searching with Partial Observations and Belief States.

Unit 2 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 3 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

