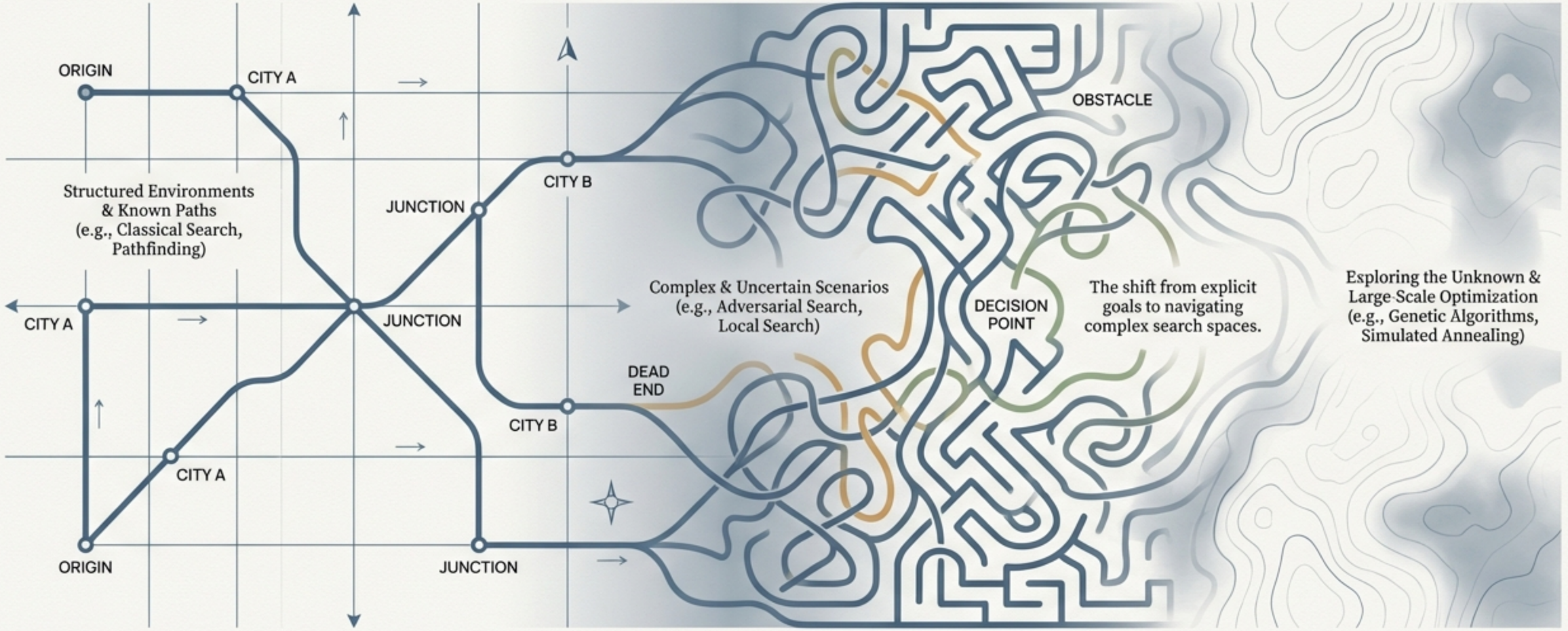
AI Problem Solving: From Maps to Mazes

A journey through advanced search and optimisation strategies.



An Agent’s Journey Through an Expanding Universe

We will follow an AI agent as its world becomes progressively more complex and uncertain. Its problem-solving strategies must evolve to survive. Each new challenge is defined by the relationship between the agent and its environment.

World vs. Agent

The World

Describes the rules of the environment.
(e.g., Deterministic, Fully Observable)

The Agent's Strategy

Describes the algorithms used to succeed.
(e.g., Use heuristics to find the optimal path)

Part 1: The Known World

Navigating with a Guide: Informed Search and Heuristics

The World vs. The Agent Frame

The World	The Agent's Strategy
Deterministic, Fully Observable, Known Goal.	Use heuristics to find the optimal path.

Core Concept

Informed search algorithms use problem-specific knowledge beyond the problem definition itself. This knowledge is encapsulated in a heuristic function.

Heuristic Function, h(n)

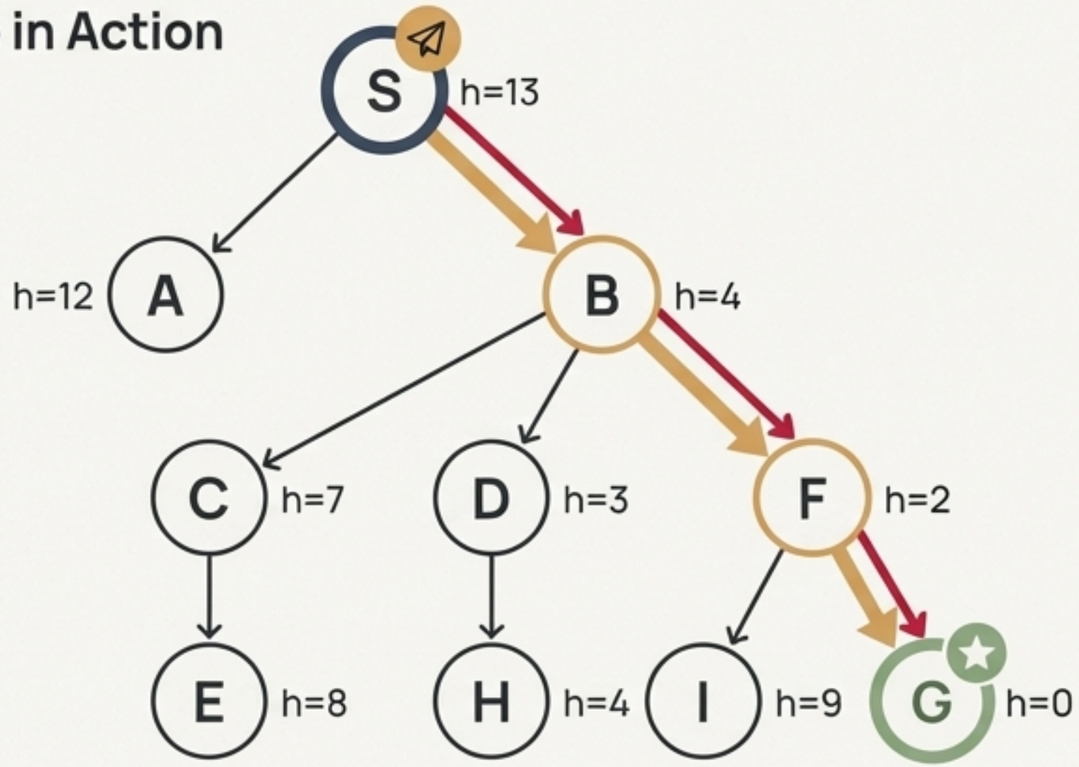
- **Definition:** An estimate of the cost of the cheapest path from the state at node n to a goal state.
- **Purpose:** It guides the search toward the most promising paths, drastically reducing the search space compared to uninformed methods.
- **Admissibility:** A key property. A heuristic is admissible if h(n) <= h*(n), where h*(n) is the true cost. An admissible heuristic is optimistic; it never overestimates the cost to reach the goal.

The Direct Route: Greedy Best-First Search

Core Idea: ‘Always select the path which appears best at that moment.’ It expands the node that is estimated to be closest to the goal.

The Logic: The evaluation function is simply the heuristic: $f(n) = h(n)$.

Example in Action



The Verdict

Strengths:

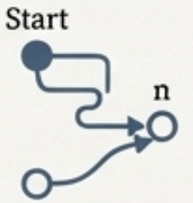
- More efficient than uninformed search.

Weaknesses:

- **Not Optimal:** It can be drawn down long, dead-end paths.
- **Incomplete:** Can get stuck in loops.
- **Complexity:** Worst-case time and space complexity is $O(b^m)$.

The Smart Route: A* Search

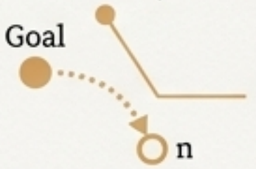
A* combines the strengths of **Uniform Cost Search** (favouring low-cost paths) and **Greedy Best-First Search** (favouring paths that seem close to the goal).



Cost of the path from the start node to `n`. (The journey so far)

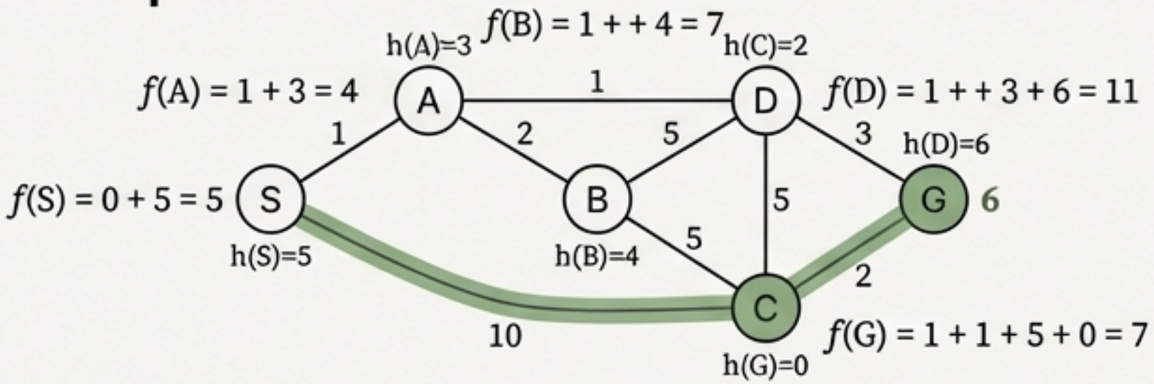
$$f(n) = g(n) + h(n)$$

Estimated cost of the cheapest solution *through* `n`.



Estimated cost from `n` to the goal. (The journey ahead)

Example Revisited



The Verdict

Properties: Optimal and complete, provided $h(n)$ is admissible (for tree search) and consistent (for graph search).

Complexity: Time and space complexity is $O(b^d)$.

Drawback: Memory requirement is its main weakness, as it keeps all generated nodes in memory.

Part 2: The Search for the Peak

When the Destination is All That Matters: Local Search

The World vs. The Agent

The World

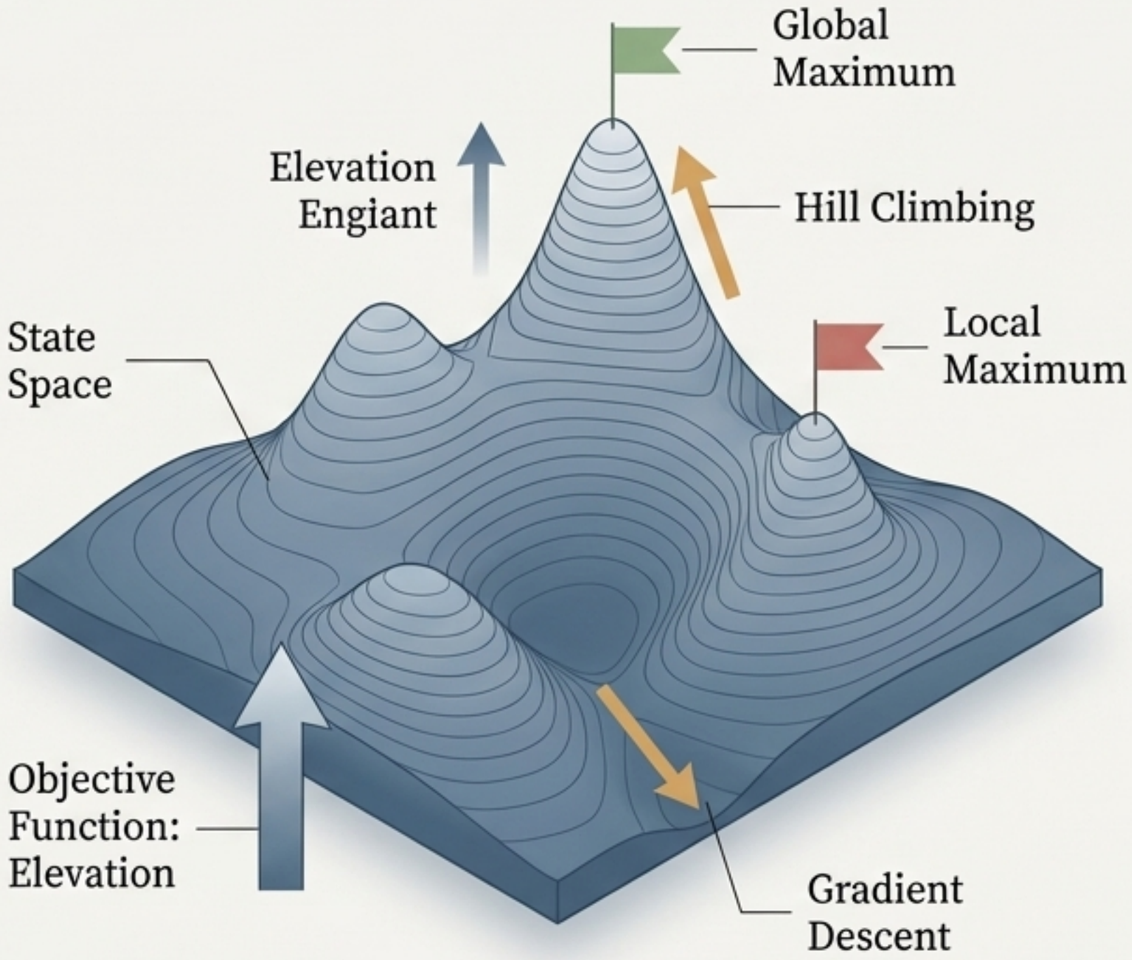
The path is irrelevant; only the final state's value matters.

The Agent's Strategy

Iteratively improve a single state to find an optimal configuration.

Core Concept

Local search algorithms operate on a single current state, moving to neighbouring states. They are not systematic but use very little memory and can find reasonable solutions in vast state spaces.



The Simplest Ascent: Hill-Climbing Search

It keeps track of one current state and on each iteration moves to the neighbouring state with the highest value. It terminates when it reaches a peak where no neighbour has a higher value.

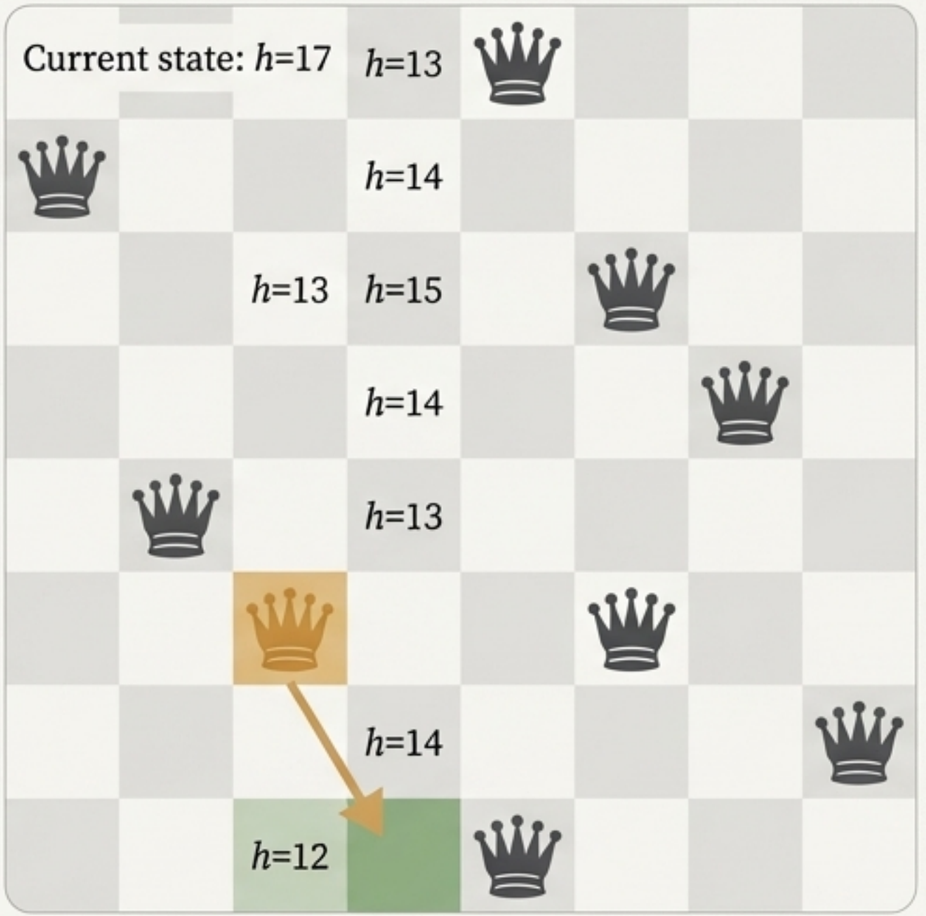
The Logic & Example

```
function HILL-CLIMBING(problem) returns a state
    current ← problem.INITIAL
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE(neighbor) <= VALUE(current) then return current
        current ← neighbor
```

Example: The 8-Queens Problem

Objective: Place 8 queens on a chessboard so that no queen attacks another.

Objective Function: h = number of attacking pairs (to be minimised).



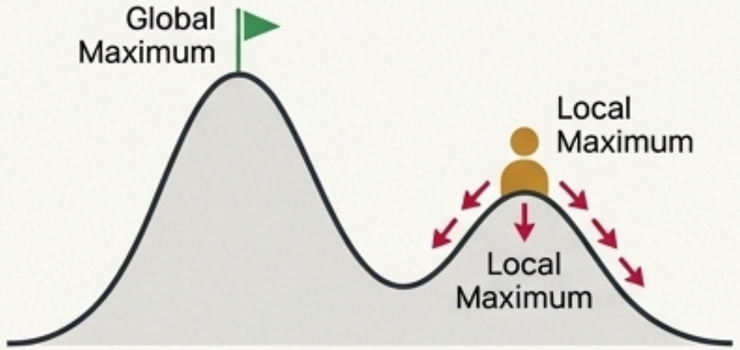
Getting Stuck: The Perils of a Greedy Ascent

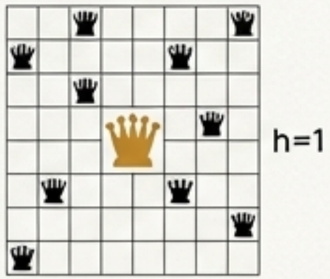
Hill climbing can get stuck because it only looks at its immediate neighbours and never accepts a “downhill” move.

The Three Traps

1. Local Maxima

A peak that is higher than its neighbours but lower than the global maximum. The algorithm is drawn to the peak and then has nowhere to go.

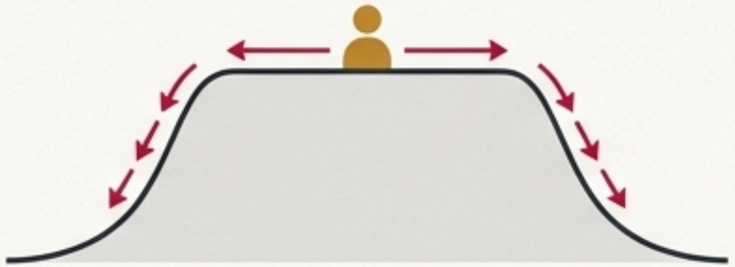




Local Maximum

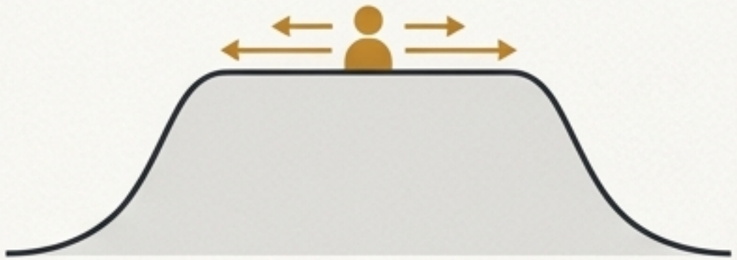
2. Ridges

A sequence of local maxima that is difficult to navigate. All available actions from the ridge-top point downhill.



3. Plateaus

A flat area of the state space. The search can wander aimlessly. It can be a “flat local maximum” or a “shoulder”.



For the 8-queens problem, standard steepest-ascent hill climbing gets stuck 86% of the time, solving only 14% of instances.

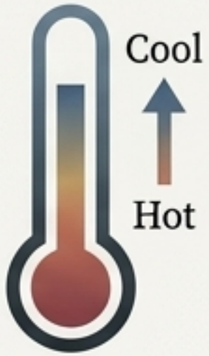
The Toolkit for Escaping Plateaus

To overcome the limitations of simple hill climbing, we need more sophisticated strategies that can navigate complex landscapes.

Random-Restart Hill Climbing

If at first you don't succeed, try, try again.

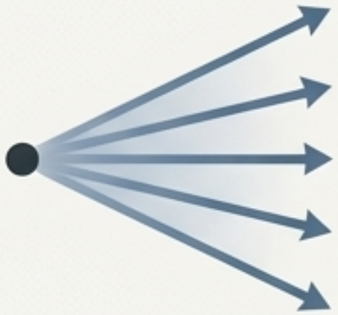
Conducts a series of searches from random initial states. Solves 94% of 8-queens problems.



Simulated Annealing

Starts "hot" (allowing bad moves), then gradually "cools".

Accepts "downhill" moves with a probability $e^{-\Delta E/T}$, which decreases as temperature `$T$` lowers.



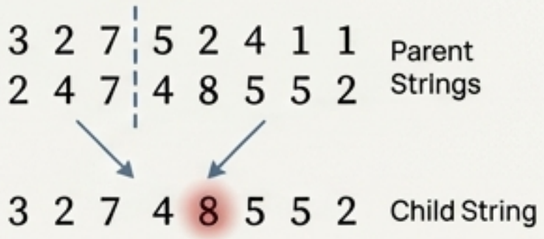
Local Beam Search

Keeps track of `$k$` states instead of just one.

Begins with `$k$` random states. At each step, generates all successors and selects the best `k, allowing parallel threads to share information.

Evolutionary Algorithms

Uses natural selection to evolve a population of solutions.



Part 3: An Uncertain World

When Actions Have a Mind of Their Own: Nondeterministic Search

The World

Nondeterministic; an action can lead to one of several possible outcomes.

The Agent's Strategy

Create a conditional (contingency) plan that handles every possible outcome.

The Problem & Example

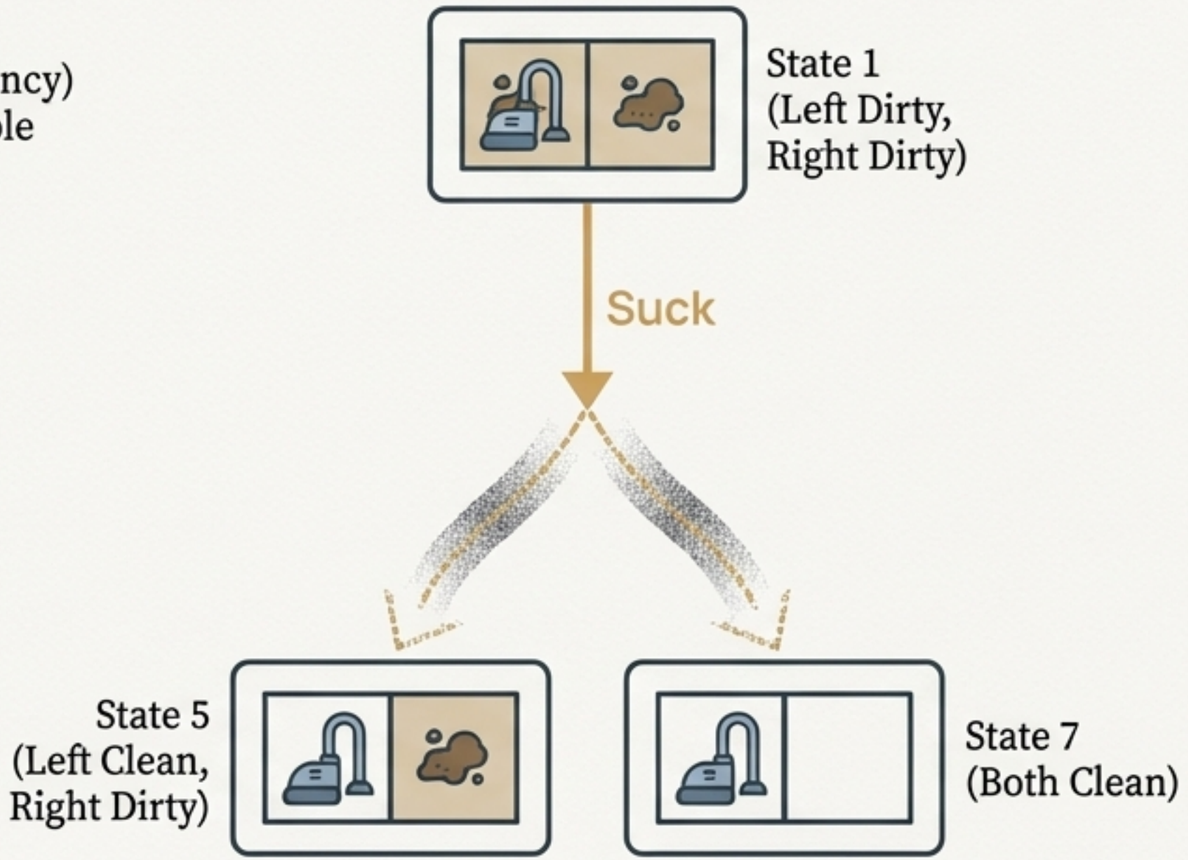
A single sequence of actions is no longer a solution. The agent must anticipate the environment's choices.

Example: The Erratic Vacuum World

The `Suck` action in a dirty square not only cleans the current square but *sometimes* cleans an adjacent one too.

`RESULTS(state, action)` now returns a *set* of outcomes.
e.g., `RESULTS(1, Suck) = {5, 7}`.

The solution is a tree, not a line:
`[Suck, if State = 5 then [Right, Suck] else []]`.



Planning for Every Possibility: AND-OR Search Trees

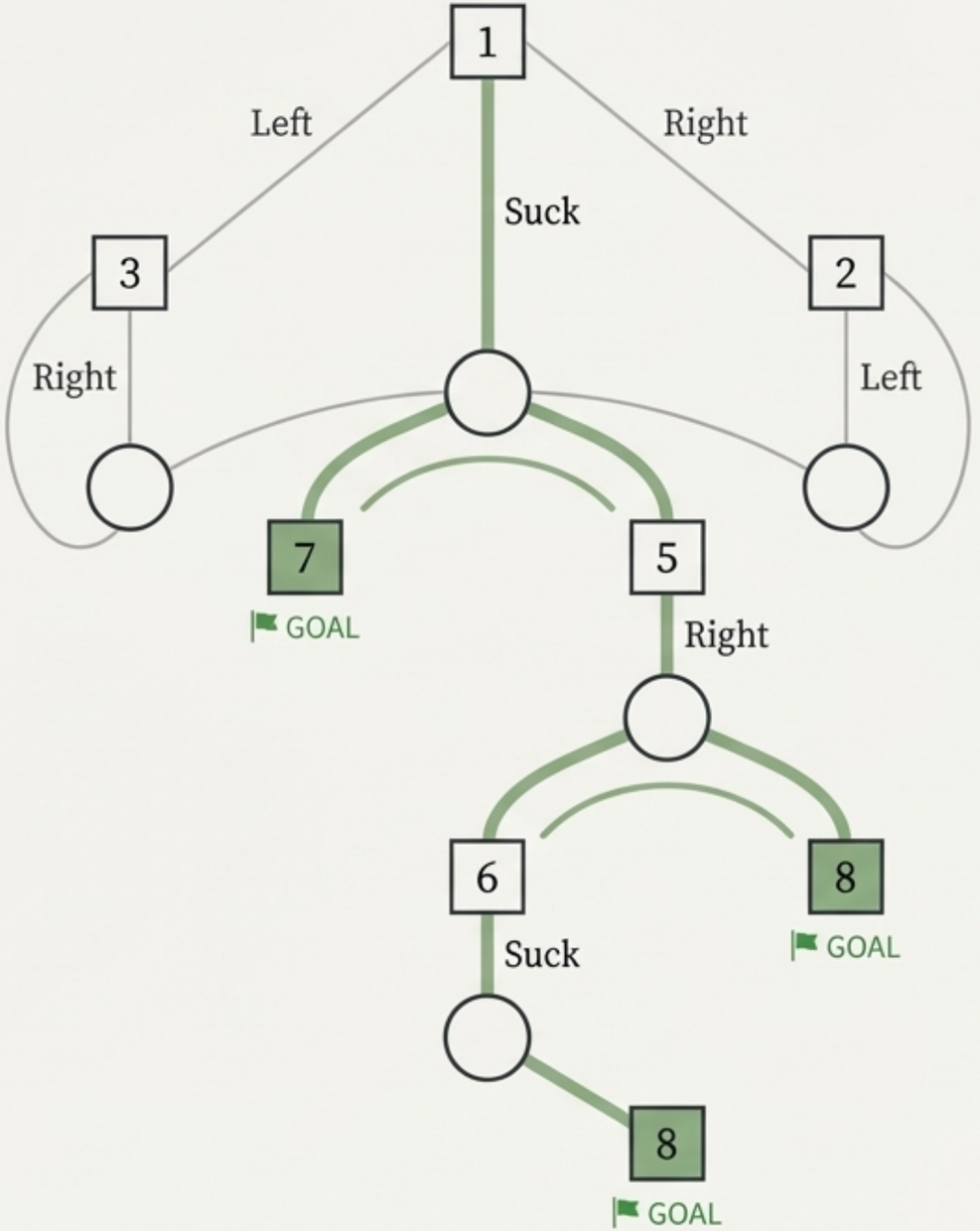
The search tree must now represent both the agent's choices and the environment's responses.

The Two Node Types

- **OR Nodes (State Nodes):** Represent points where the agent must choose an action. Shown as squares.
- **AND Nodes (Outcome Nodes):** Represent the set of possible outcomes for an action. The plan must handle *all* of them. Shown as circles.

What a Solution Looks Like

1. Has a goal node at every leaf.
2. Specifies one action at each of its OR nodes.
3. Includes every outcome branch at each of its AND nodes.



Piercing the Fog of War: Search in Partially Observable Environments

The World vs. The Agent

The World	The Agent's Strategy
Partially Observable; the agent's percepts are not enough to uniquely identify its state.	Search in the space of *belief states* rather than physical states.

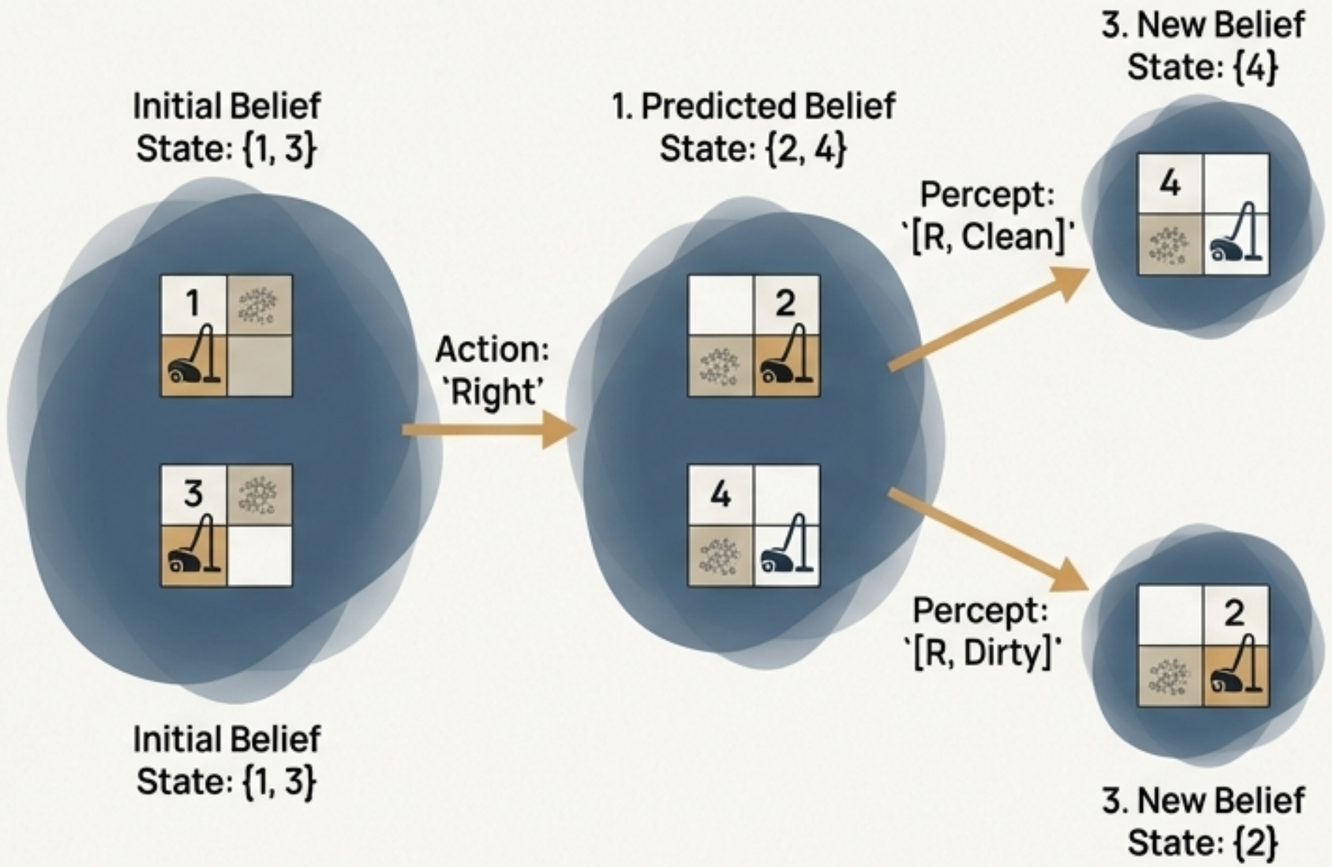
Core Concept: The Belief State

Definition

A set of physical states that the agent believes are currently possible.

Example

In the "local-sensing" vacuum world, the percept [L, Dirty] could correspond to state 1 or state 3. The initial belief state is {1, 3}.



Prediction -> Possible Percepts -> Update

Part 4: The Uncharted Territory

Drawing the Map as You Go: Online Search

The World vs. The Agent Frame	
The World Unknown; the agent does not know the states or the results of its actions in advance.	**The Agent's Strategy** Interleave computation and action. Use actions as experiments to learn a map of the environment.

Core Idea

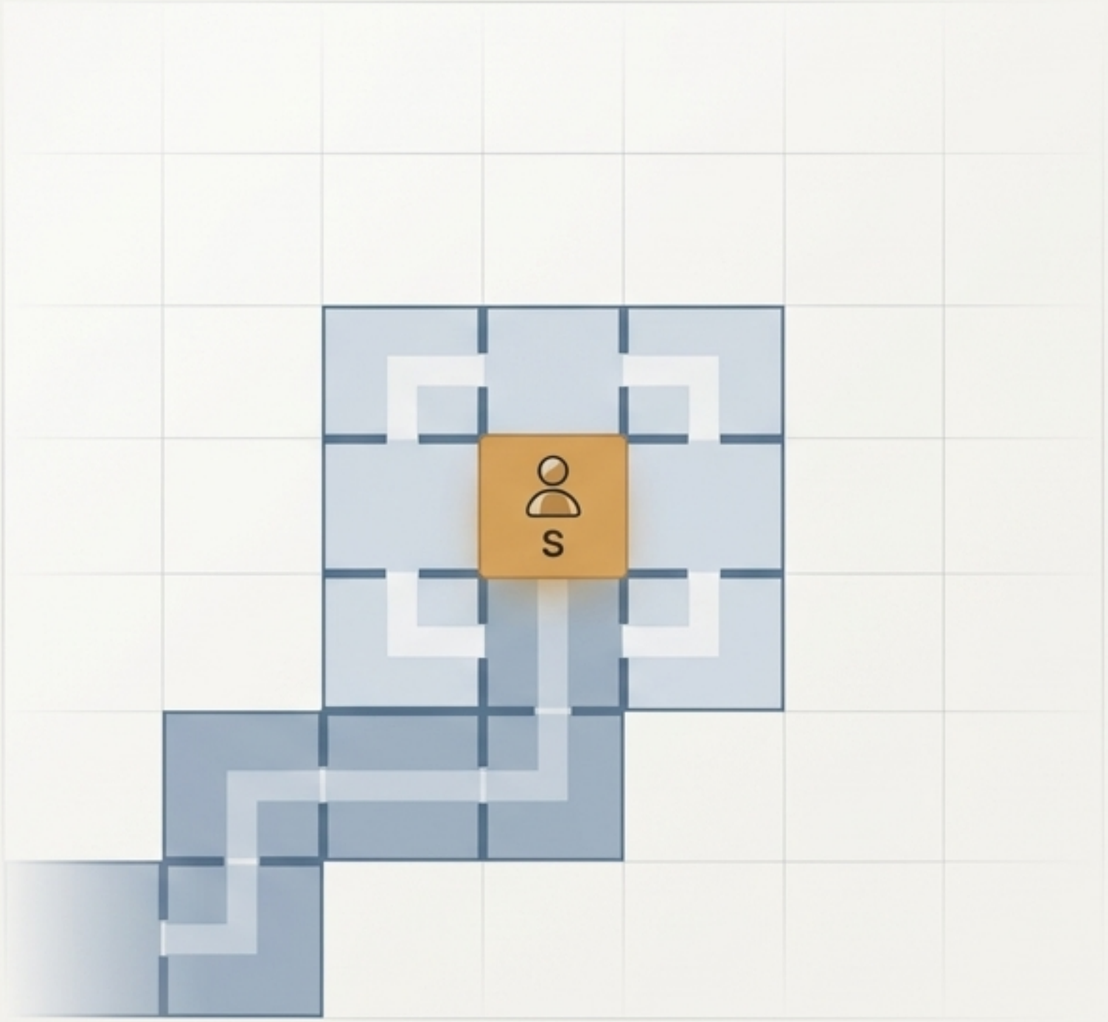
In contrast to offline search (compute a full solution, then act), an online search agent takes an action, observes the result, and then computes the next action.

When is it useful?

- Dynamic environments where thinking for too long is penalised.
- Nondeterministic domains, to focus on contingencies that actually occur.
- Completely unknown environments, like a robot mapping a new building.

The Challenge of Dead Ends

An agent can't know if an action is irreversible. We assume a "safely explorable" environment where a goal is reachable from every reachable state.



Strategies for the Explorer

Online Depth-First Search

A natural fit for online search. However, backtracking is now a physical action. The agent must physically travel back to a previous state to explore a different path.

Online Local Search

Basic hill-climbing gets stuck. Random walks can be exponentially slow. We need a smarter approach.

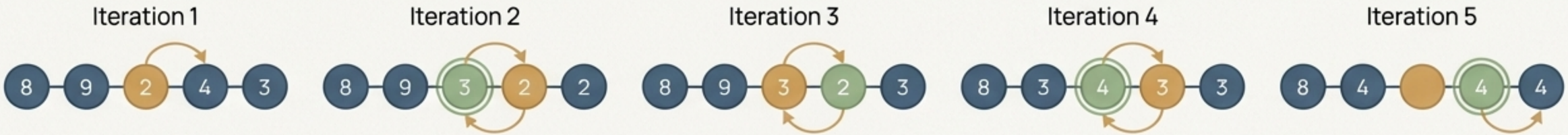
Learning Real-Time A* (LRTA*)

Core Idea: Augments hill climbing with memory. The agent stores and updates a cost estimate, H(s), for the cost to reach the goal from each visited state s.

Process:

1. Agent at state s calculates cost(s, a, s') + H(s') for its neighbours.
2. It moves to the neighbour with the minimum estimated total cost.
3. Crucially, it then updates the estimate for the state it just left, H(s), based on this new experience.

Effect: This process "flattens out" local minima, allowing the agent to eventually escape.

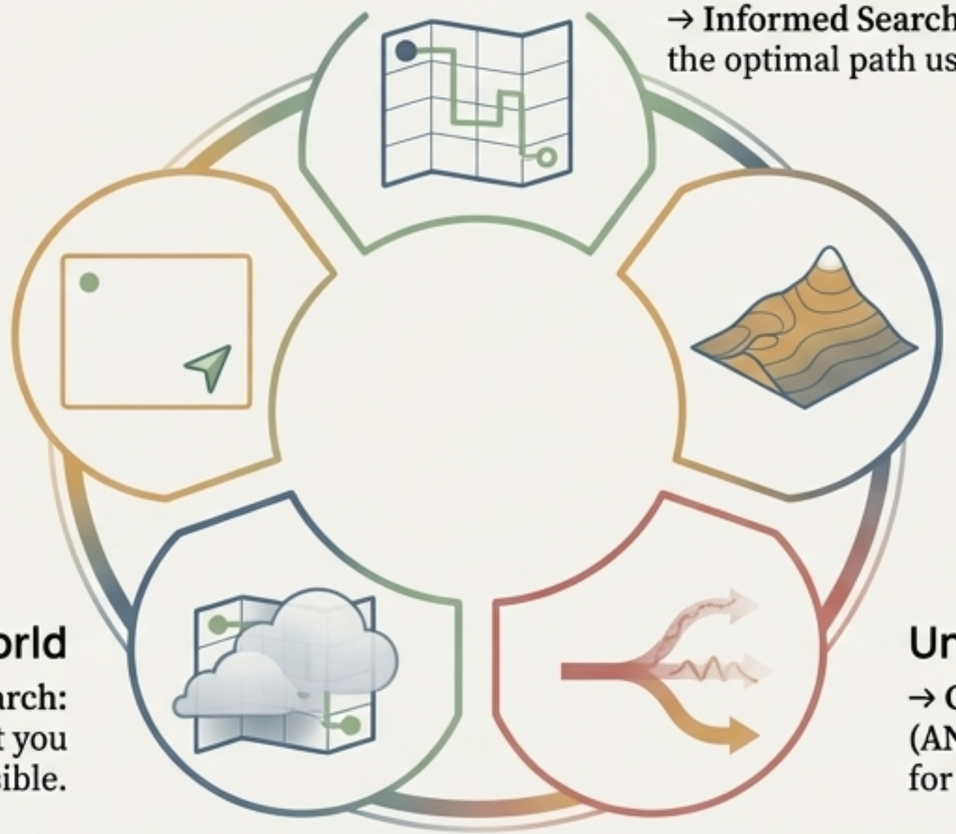


The Right Tool for the World

An intelligent agent's power comes from its ability to adapt its strategy to the nature of the problem. As the world transforms from a simple map to an unknown maze, the agent's toolkit must expand.

Known World

→ **Informed Search (A*):** Find the optimal path using a guide.



Unknown World

→ **Online Search (LRTA*):** Explore and learn the map as you go.

Optimisation Problem

→ **Local Search (Simulated Annealing):** Find the highest peak, avoiding local traps.

Obscured World

→ **Belief State Search:** Act based on what you believe is possible.

Uncertain World

→ **Contingency Planning (AND-OR Search):** Plan for every possibility.

"The essence of advanced problem-solving lies not in a single master algorithm, but in a deep understanding of the environment and the selection of a strategy to match."