Introduction to AI

Lecture 7: Constraint Satisfaction & Optimization

Mona Taghavi

LaSalle College
Montréal

Informed search - optimization

- Constraint Satisfaction Problem & Backtracking
- Hill climbing
- Simulated annealing
- Genetic algorithm
- Bounce n bound
- Monte Carlo
- Etc

Constraint Satisfaction Problem (CSP)

- In CSPs, the problem is to search for a set of values for the variables so that the values satisfy some conditions (constraints).

- More formally, a CSP consists of
  - A set of variables V1, …, Vn
  - For each variable a domain of possible values Dom[Vi].
  - A set of constraints C1,…, Cm.

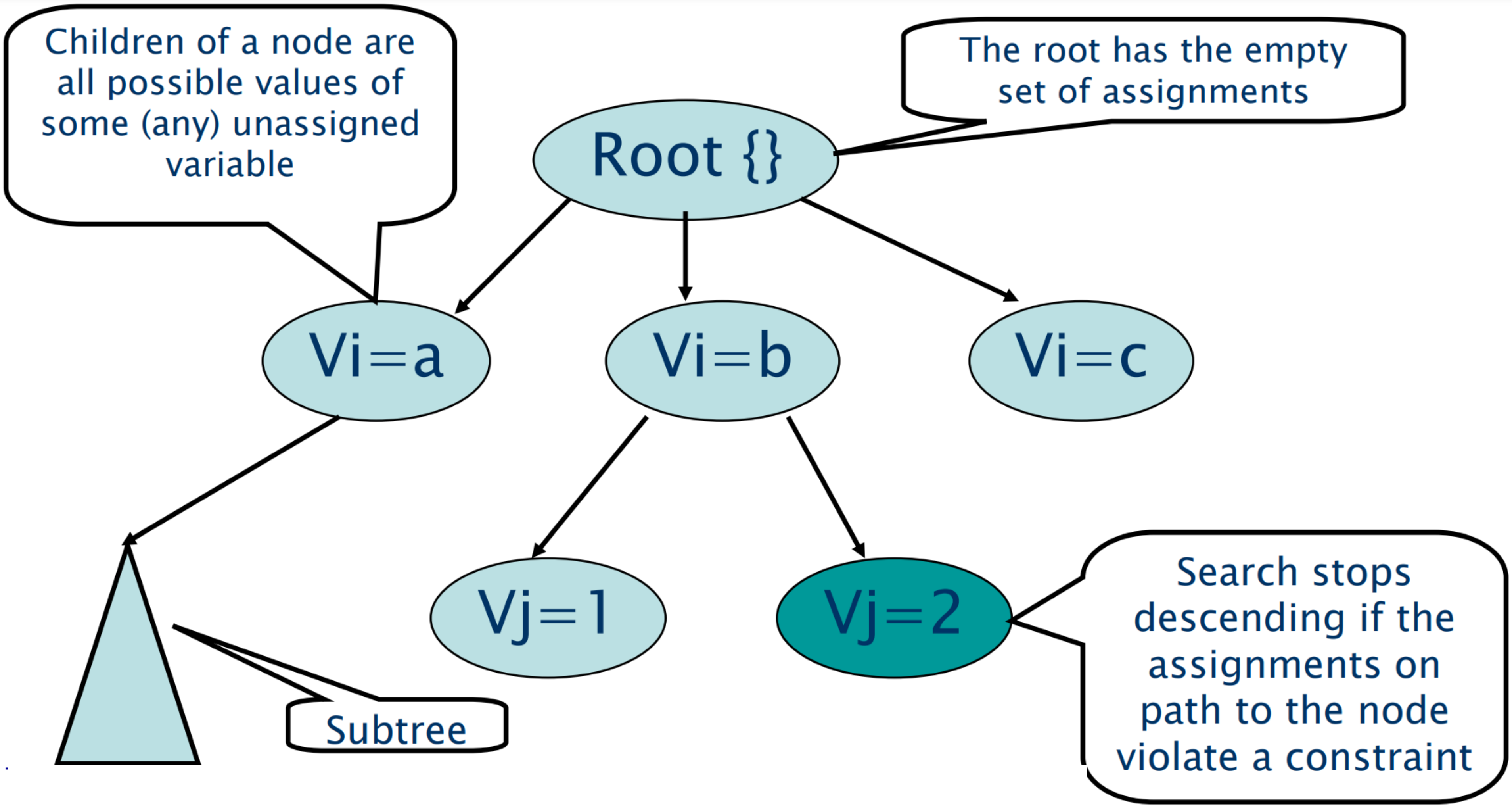
- A solution to a CSP is an assignment of a value to all of the variables such that every constraint is satisfied.

CSP as a Search Problem

- Initial state: empty assignment
- Successor function: a value is assigned to any unassigned variable, which does not conflict with the currently assigned variables
- Goal test: the assignment is complete

Solving CSPs – Backtracking Search

- The algorithm searches a tree of partial assignments.



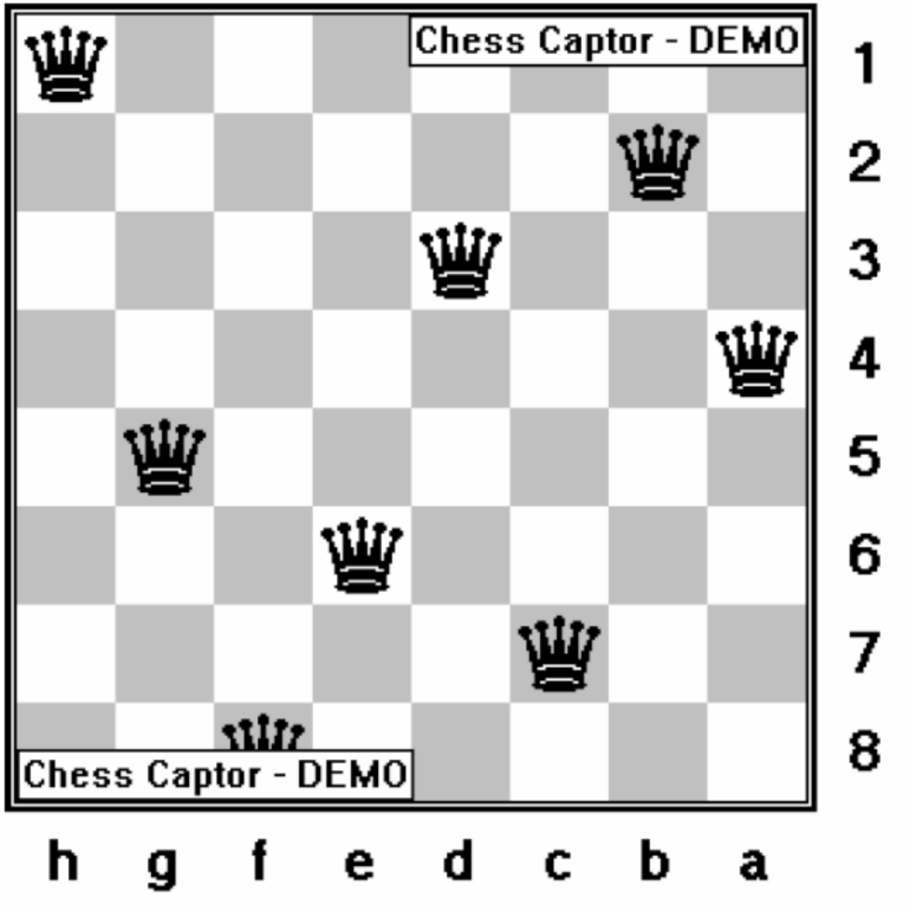
Backtracking Search

- Heuristics are used to determine
  - the order in which variables are assigned: PickUnassignedVariable()
  - the order of values tried for each variable.
- The choice of the next variable can vary from branch to branch,
  - e.g., under the assignment V1=a we might choose to assign V4 next, while under V1=b we might choose to assign V5 next.
- This "dynamically" chosen variable ordering has a tremendous impact on performance.

- Real-world CSPs

Example: N-Queens

- Place N Queens on an N X N chess board so that no Queen can attack any other Queen.

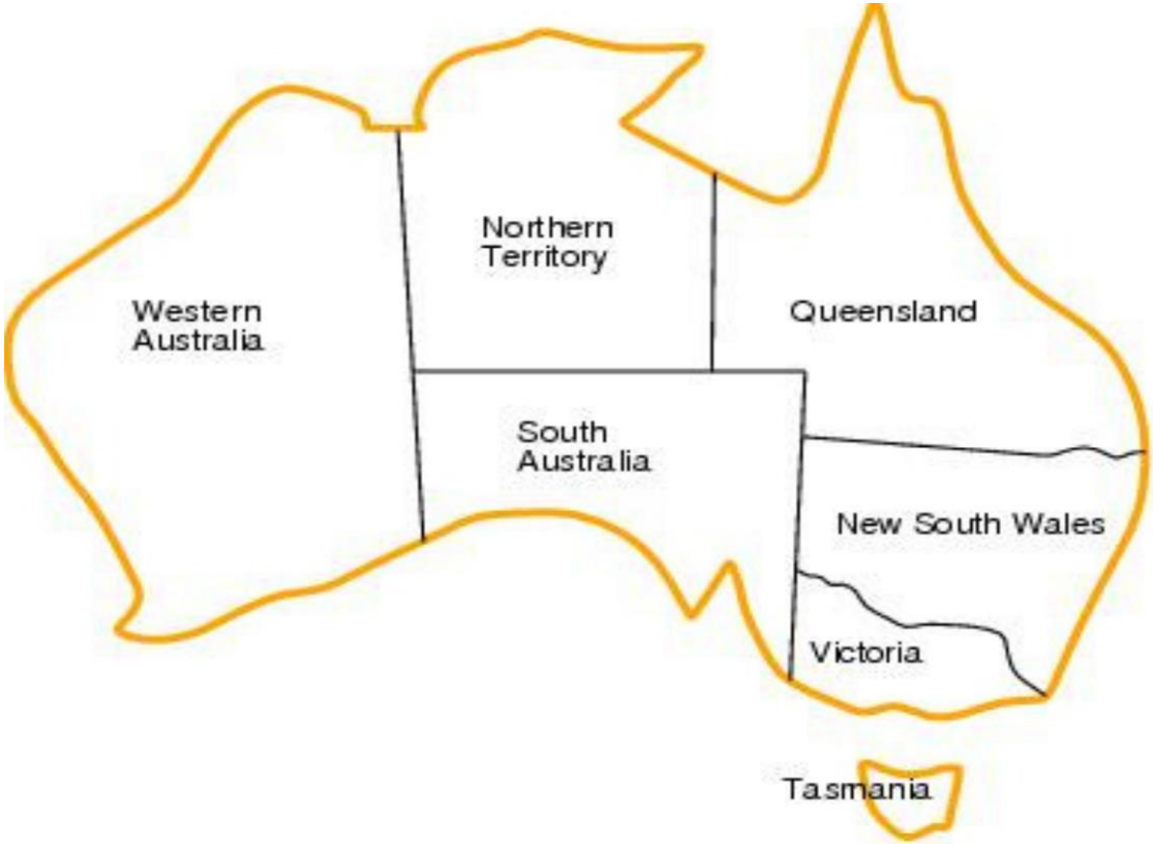


Example: N-Queens

- Place N Queens on an N X N chess board so that no Queen can attack any other Queen.
- Constraints:
  - Can't put two Queens in same column $Q_i \neq Q_j$ for all $i \neq j$
  - Diagonal constraints $|Q_i - Q_j| \neq i-j$
  - i.e., the difference in the values assigned to Qi and Qj can't be equal to the difference between i and j.

Example – Map Colouring

- Color the following map using red, green, and blue such that adjacent regions have different colors.



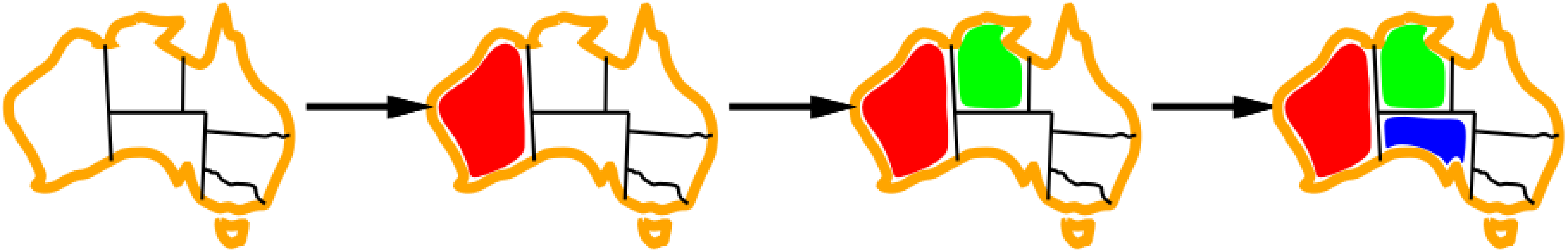
Improving backtracking efficiency

- 1. Which variable should be assigned next?
- 2. In what order should its values be tried?
- 3. Can we detect inevitable failure early?

- Forward Checking: keep track of remaining legal values for unassigned variables to guide us which variables to try next

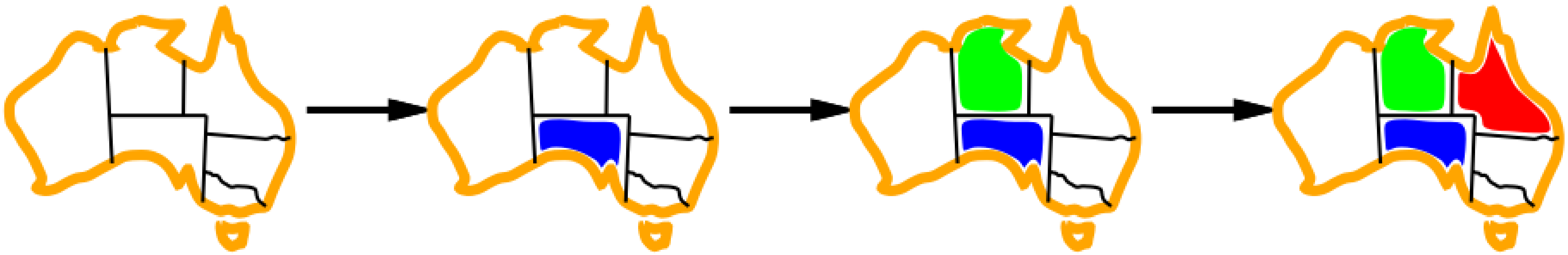
Minimum remaining values

- Minimum remaining values (MRV): choose the variable with the fewest legal values



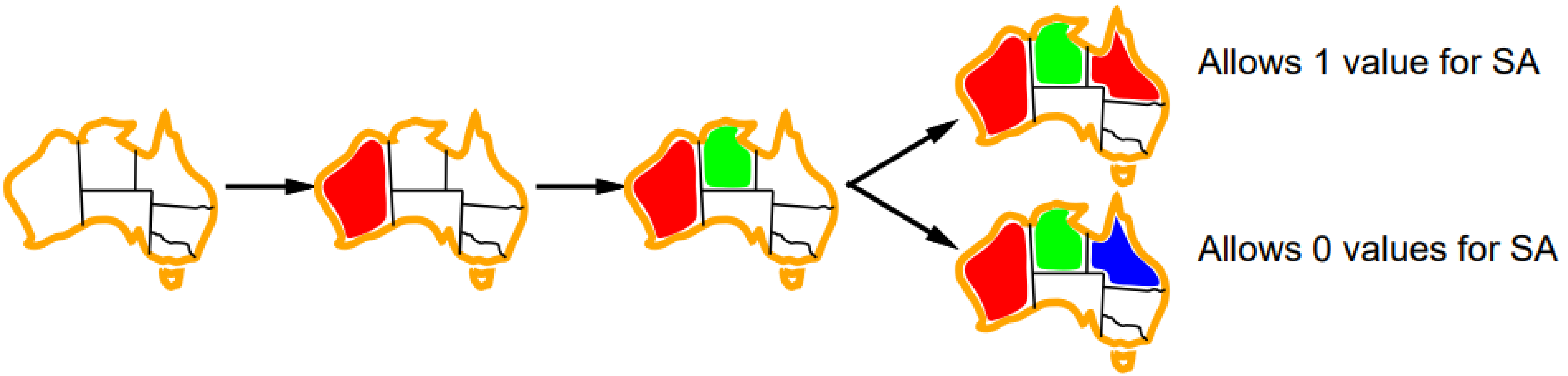
Degree heuristic

- Degree heuristic: choose the variable with the most constraints on remaining variables



Least constraining value

- Given a variable, choose the least constraining value: the one that rules out the fewest values in the remaining variables



Optimization problems

- In many optimization problems, path is irrelevant; the goal state itself is the solution.

  Algorithm goal:

  - find optimal configuration (e.g., TSP), or,
  - find configuration satisfying constraints (e.g., n-queens)

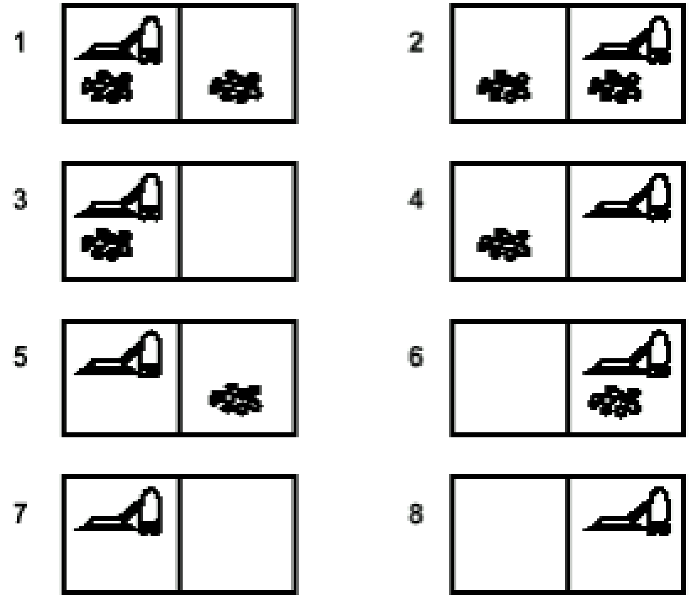
- In such cases, can use keep a single "current" state, and try to improve it → iterative improvement

Example: vacuum world

Simplified world: 2 locations, each may or not contain dirt, each may or not contain vacuuming agent.

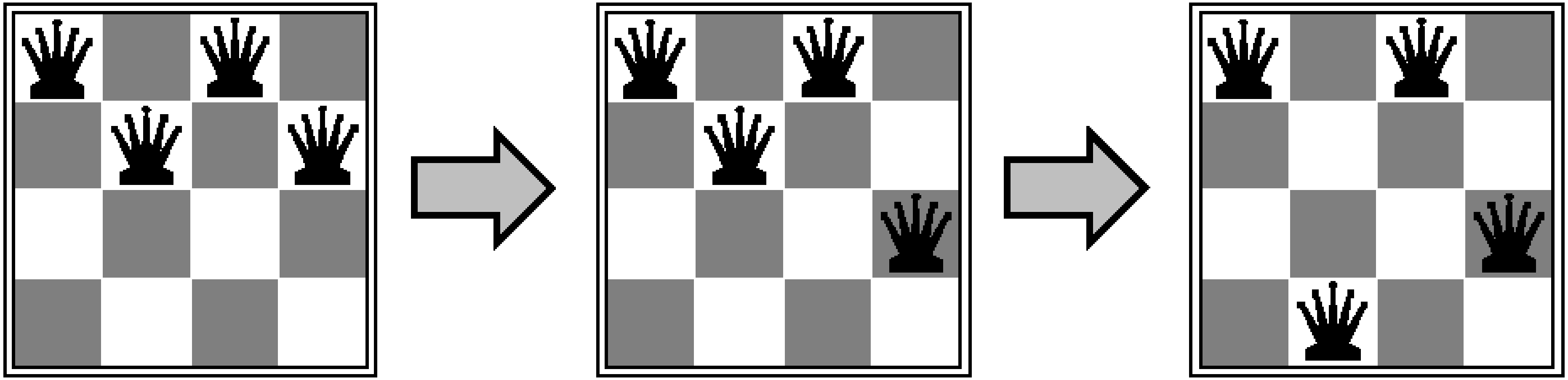
Goal of agent: clean up the dirt.

If path does not matter, do not need to keep track of it.



Example: n-queens

- Goal: Put n chess-game queens on an n x n board, with no two queens on the same row, column, or diagonal.



- Here, goal state is initially unknown but is specified by constraints that it must satisfy.

Hill climbing (or gradient descent)

- Very simple idea: Start from some state s, Move to a neighbor t with better score. Repeat.
- Iteratively maximize "value" of current state, by replacing it by successor state that has highest value, as long as possible.
- Evaluating and modifying current state/s rather than systematically exploring paths from an initial state.
  - Less memory

Hill climbing algorithm

- Define the current state as an initial state
- Loop until the goal state is achieved or no more operators can be applied on the current state:
  - Apply an operation to current state and **get a new state**
  - **Compare** the new state with the goal
  - **Quit** if the goal state is achieved
  - Evaluate new state with heuristic function and **compare it with the current state**
  - **If the newer state is closer** to the goal compared to current state, **update the current state**

8 puzzle problem using hill climbing

1	2	4
5		7
3	6	8

Initial state

1	4	7
2	5	8
3	6	

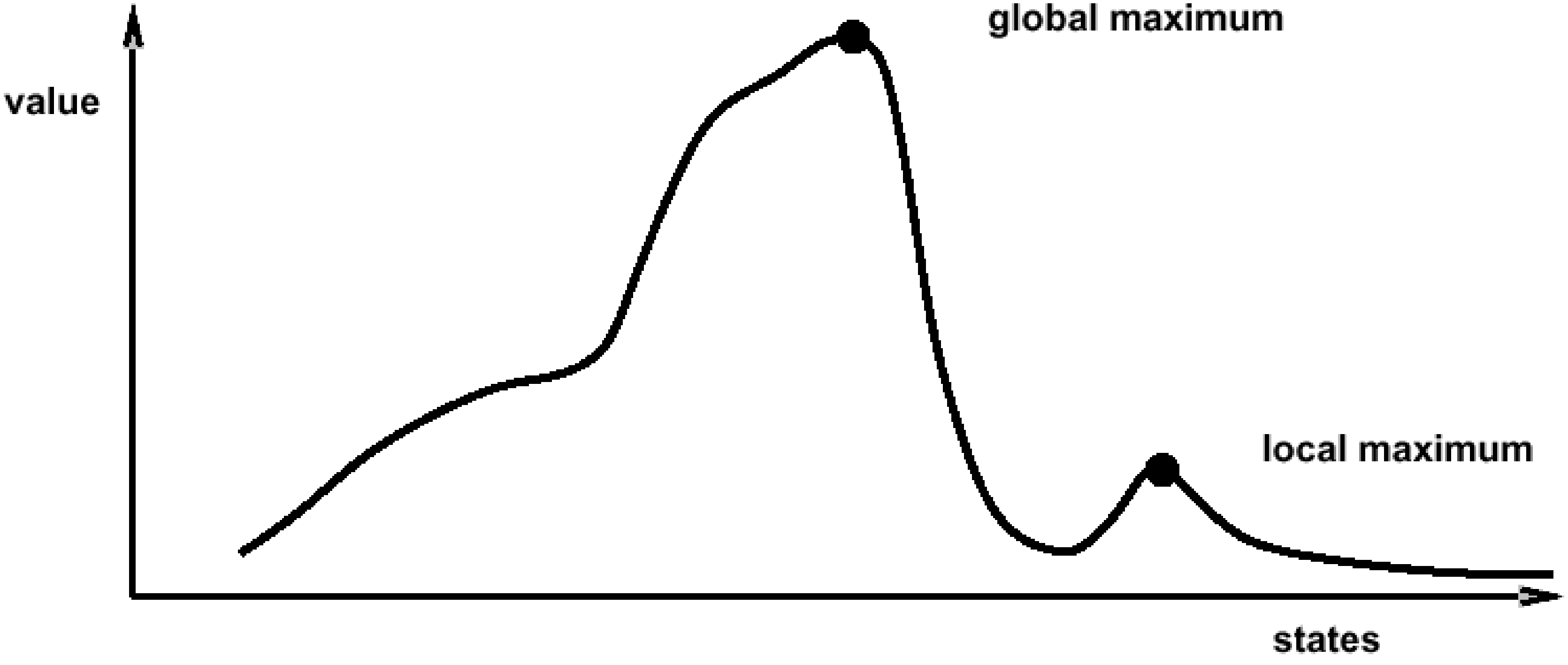
final state

h1 = the number of misplaced tiles.

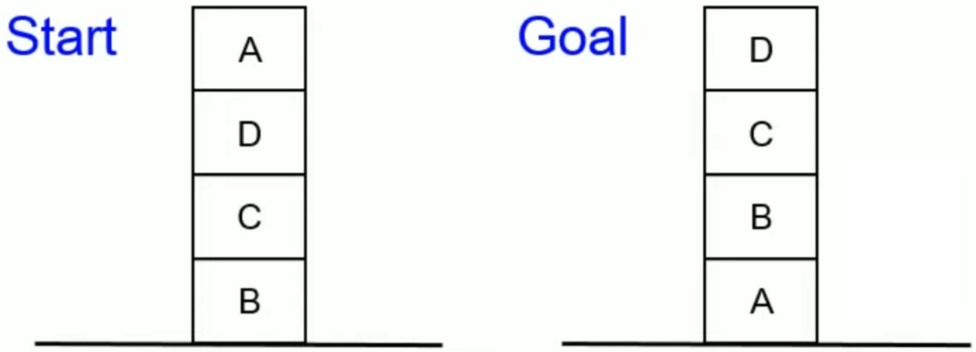
h1 is an admissible heuristic because it is clear that any tile that is out of place must be moved at least once.

Hill climbing

- Problem: depending on initial state, may get stuck in local extremum.



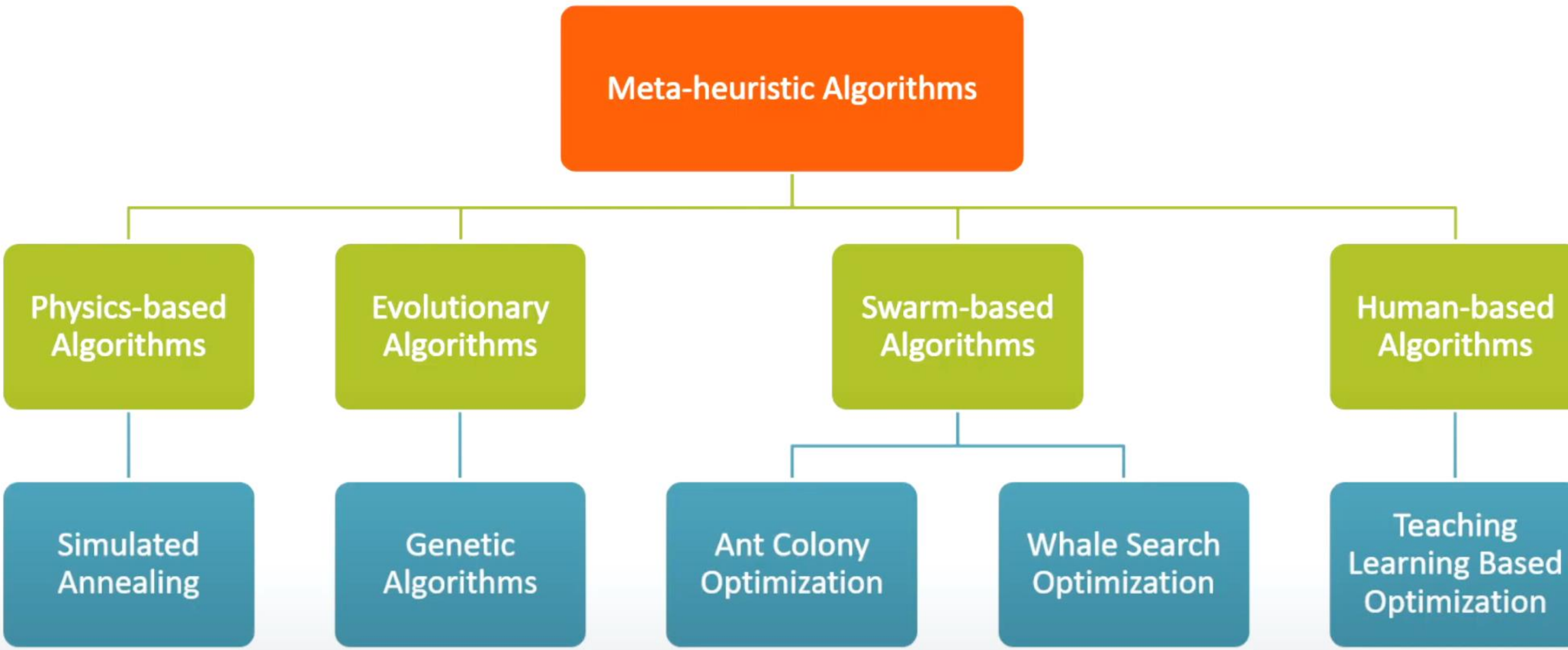
Hill Climbing: Local Heuristic function



Blocks World

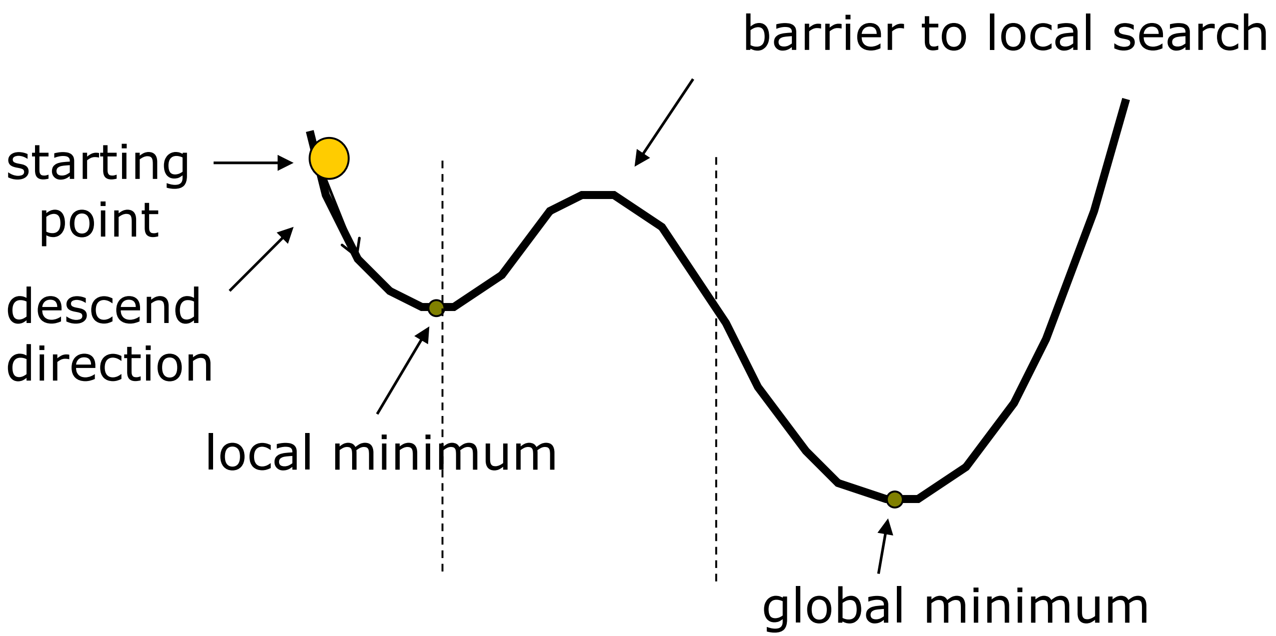
Bio-inspired algorithms

Meta-heuristic Algorithms



Local Minima Problem

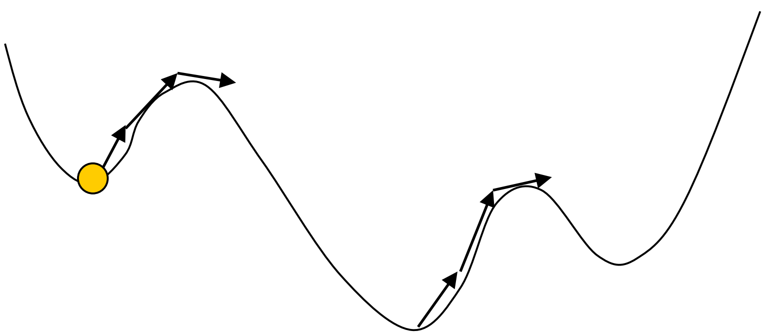
- Question: How do you avoid this local minimum?



Consequences of the Occasional Ascents

desired effect

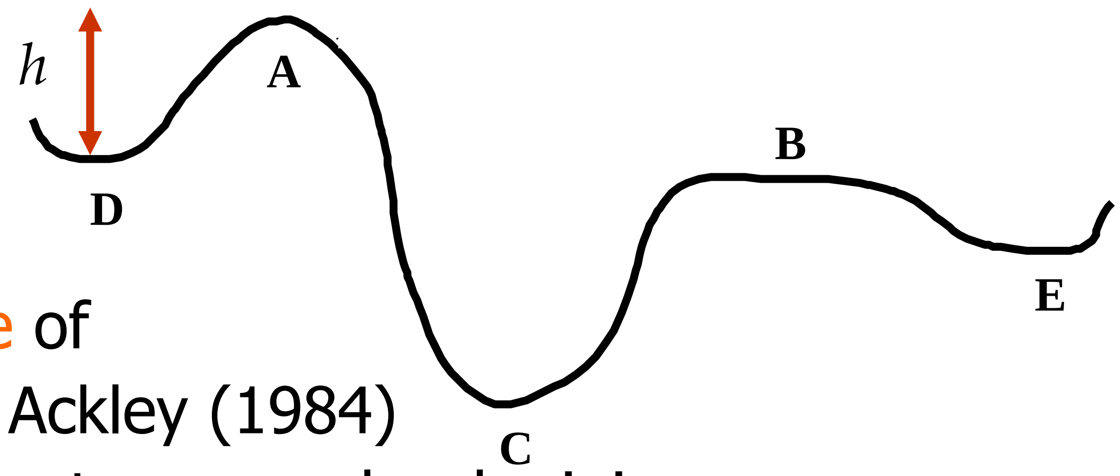
Help escaping the local optima.



adverse effect

Might pass global optima after reaching it

Boltzmann machines



The Boltzmann Machine of Hinton, Sejnowski, and Ackley (1984) uses simulated annealing to escape local minima.

To motivate their solution, consider how one might get a ball-bearing traveling along the curve to "probably end up" in the deepest minimum. The idea is to shake the box "about h hard" — then the ball is more likely to go from D to C than from C to D. So, on average, the ball should end up in C's valley.

Simulated annealing: basic idea

- From current state, pick a **random** successor state;

- If it has better value than current state, then “accept the transition,” that is, use successor state as current state;

- Otherwise, do not give up, but instead flip a coin and accept the transition with a given probability (that is lower as the successor is worse).

- So we accept to sometimes “un-optimize” the value function a little with a non-zero probability.

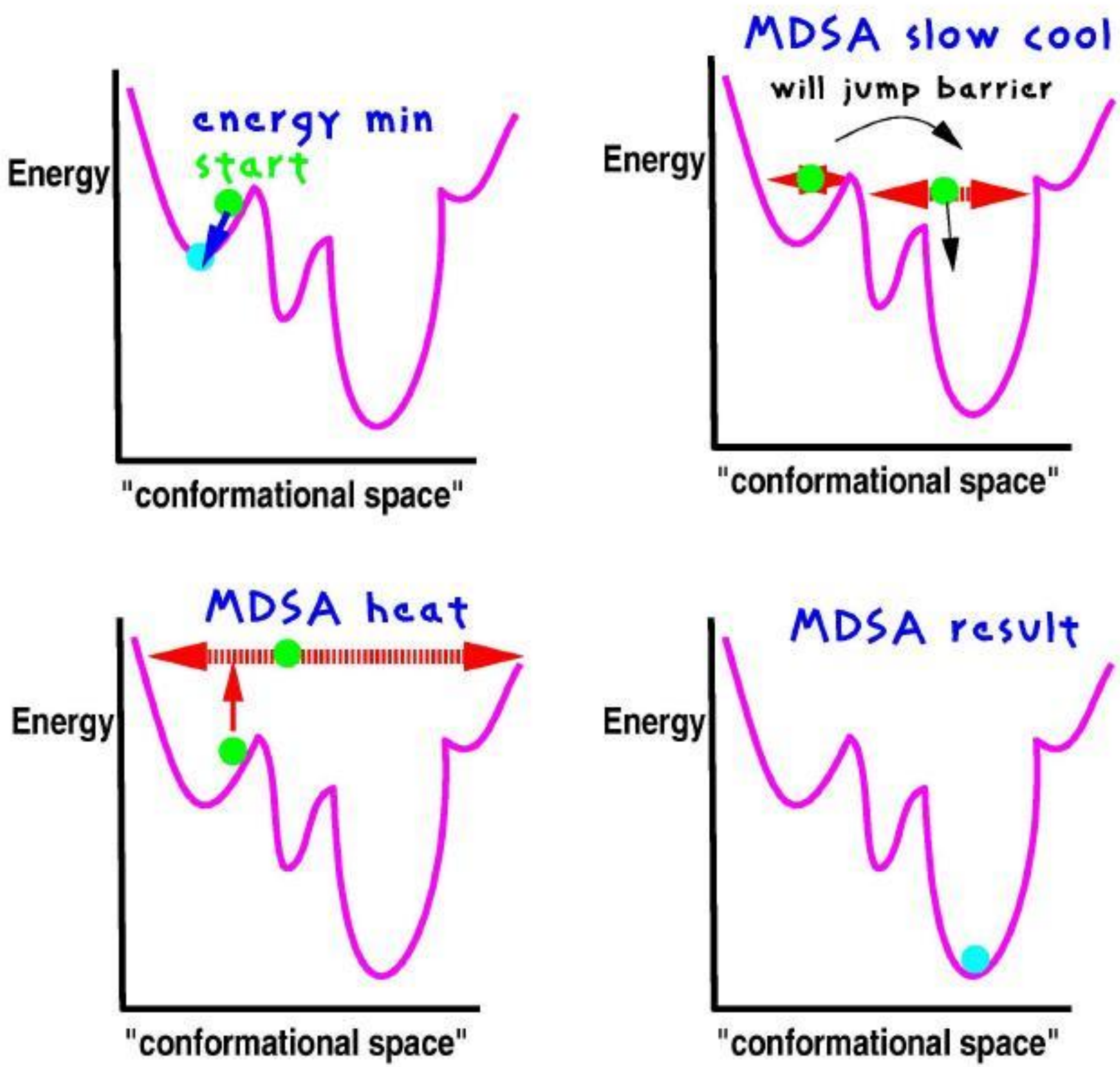
Simulated annealing

- Simulated annealing is a general method for making likely the escape from local minima by allowing jumps to other states with high heat.
- The analogy here is with the process of annealing used by a craftsman in forging a sword from an alloy.

Real annealing: Sword

- He heats the metal, then slowly cools it as he hammers the blade into shape.
  - If he cools the blade too quickly the metal will form patches of different composition;
  - If the metal is cooled slowly while it is shaped, the constituent metals will form a uniform alloy.

Simulated annealing in practice



MDSA: Molecular Dynamics Simulated Annealing

Simulated annealing

1. Pick initial state *s*
2. Randomly pick *t* in neighbors(*s*)
3. IF *f*(*t*) better THEN accept *s*←*t*.
4. ELSE /* *t* is worse than *s* */
5. accept *s*←*t* with a small probability
6. GOTO 2 until bored.

How to choose the small probability?

idea 1: $p = 0.1$

idea 2: p decreases with time

idea 3: p decreases with time, also as the 'badness' $|f(s)-f(t)|$ increases