Artificial Intelligence Foundation – JC3001

Lecture 8: Search III: Local Search I

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

September 2025

Material adapted from:

Russell and Norvig (AIMA Book): Chapter 4 (4.1–4.4)

Dana Nau (University of Maryland)

Course Progression

- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. Search 2: Heuristic Search ✓
  3. **Search 3: Local Search**
  4. Search 4: Adversarial Search
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction
  2. Reasoning 2: Logic and Inference
  3. Probabilistic Reasoning 1: BNs
  4. Probabilistic Reasoning 2: HMMs
- Part 4: Planning
  1. Planning 1: Intro and Formalism
  2. Planning 2: Algos and Heuristics
  3. Planning 3: Hierarchical Planning
  4. Planning 4: Stochastic Planning
- Part 5: Learning
  1. Learning 1: Intro to ML
  2. Learning 2: Regression
  3. Learning 3: Neural Networks
  4. Learning 4: Reinforcement Learning
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

Objectives

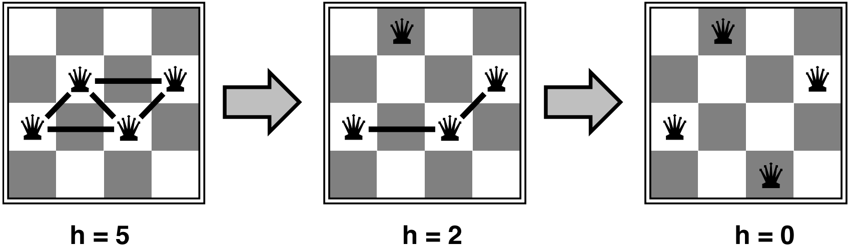
- Single-state problems
- Local Search and Optimization
  - Hill Climbing
  - Simulated Annealing
  - Beam Search
  - Genetic Algorithms

Iterative Improvement Algorithms

- In many optimization problems, the **path** to a goal is irrelevant
  - the goal state itself is the solution
- State space = a set of goal states
  - find one that satisfies constraints (e.g., no two classes at same time)
  - or find **optimal** one (e.g., highest possible value, least possible cost)
- Iterative improvement algorithms – keep a single “current“ state, try to improve it
  - Constant space
  - Suitable for both offline and online search

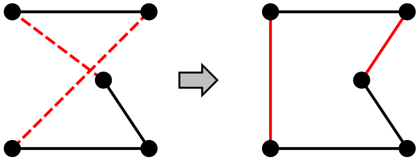
Example: the n-Queens Problem

- Put n queens on an $n \times n$ chessboard — No two queens on the same row, column, or diagonal
- Iterative improvement:
  - Start with one queen in each column
  - Move a queen to reduce number of conflicts
- Even for very large n (e.g., n = 1 million), this usually finds a solution almost instantly



Example: Travelling Salesperson Problem

- Given a complete graph (edges between all pairs of nodes)
- Find a least-cost tour (simple cycle that visits each city exactly once)
- Iterative improvement:
  - Start with any tour, perform pairwise exchanges
  - Variants of this approach get within 1% of optimal very quickly with thousands of cities



Example: University Timetabling

- Given sets of lecturers, courses, lecture halls, and time slots
  - In practice, a lot more complex than that
- Find an assignment of courses to lecturers, to lecture halls to time slots
- Iterative improvement:
  - Start with a random assignment, swap resources
- International Timetabling competition (`https://www.itc2019.org`)

- Hill Climbing
- Simulated Annealing

Hill-climbing (or gradient ascent/descent)

Like climbing Everest in thick fog with amnesia

function Hill-Climbing($problem$)
    $current \leftarrow$ new node containing $problem$'s initial state
    loop
        $next \leftarrow$ a highest-valued neighbour of $current$
        if $Value[next] \leq Value[current]$ **then return** $State[current]$
        $current \leftarrow next$

- $Value[x]$ is x's objective-function value – how good we consider x to be
- At each step, move to a neighbour of higher value
  in hopes of getting to a solution having the highest possible value
- Can easily modify this for problems where we want to minimize rather than maximize

- Thinking of a 5-bit sequence
- $f(x) = \#$ correct bits
- $N(x)$: one bit differences from X

X : 5-bit sequences

- Thinking of a 5-bit sequence
- $f(x) = \#$ correct bits
- $N(x)$: one bit differences from X

$$
\begin{aligned}
x :& 00000 - 2 \\
& 10000 - 3 \\
& 01000 - 1 \\
& 00100 - 3 \\
& 00010 - 3 \\
& 00001 - 1
\end{aligned}
$$

X : 5-bit sequences

- Thinking of a 5-bit sequence
- $f(x) = \#$ correct bits
- $N(x)$: one bit differences from X

$$
\begin{aligned}
x : & 00000 - 2 \\
& 10000 - 3 \\
& 01000 - 1 \\
& 00100 - 3 \\
& 00010 - 3 \\
& 00001 - 1
\end{aligned}
\qquad
\begin{aligned}
x : & 10000 - 3 \\
& 11000 - 2 \\
& 10100 - 4 \\
& 10010 - 4 \\
& 10001 - 2
\end{aligned}
$$

X : 5-bit sequences

- Thinking of a 5-bit sequence
- $f(x) = \#$ correct bits
- $N(x)$: one bit differences from X

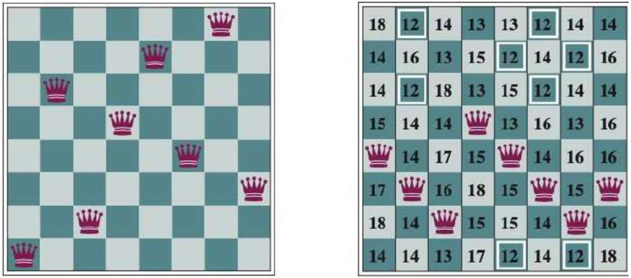
$$\begin{aligned} x :& 00000 - 2 \\ & 10000 - 3 \\ & 01000 - 1 \\ & 00100 - 3 \\ & 00010 - 3 \\ & 00001 - 1 \end{aligned}$$

$$\begin{aligned} x :& 10000 - 3 \\ & 11000 - 2 \\ & 10100 - 4 \\ & 10010 - 4 \\ & 10001 - 2 \\ x :& 10110 - 5 \end{aligned}$$

Hill-climbing example

- n-queens problem
- Successor function: move a single queen to another position in the same column
- Heuristic function $h(n)$: number of queen-pairs attacking each other (directly or indirectly)

Hill-climbing example

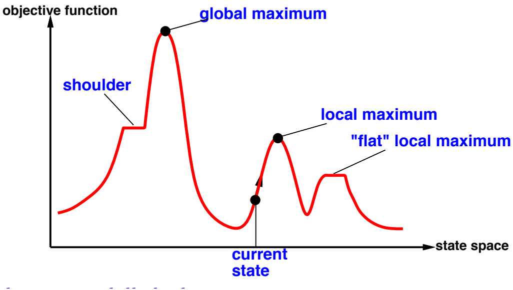


(a) (b)

a) a local minimum in the state-space of the 8-queens problems ($h = 1$)

b) state with $h = 17$ and h-value for each possible successor

State space "landscape":

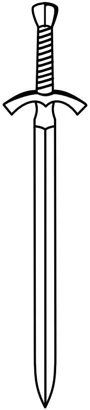


- Random-restart hill climbing:
  - repeat with randomly chosen starting points
- If finitely many local maxima, then $\lim_{restarts \to \infty} \mathbb{P}\left[complete\right] = 1$

- Local maxima — peak that is lower than the highest peak in the state-space
- Peak — search can oscillate
- Plateaux — state-space region in which evaluation function is flat

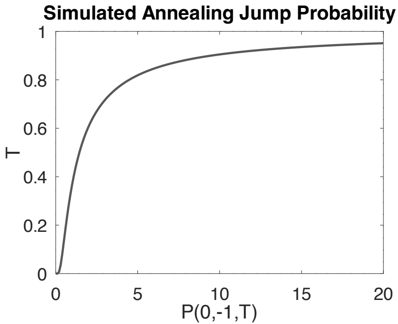
- Hill Climbing
- Simulated Annealing

- Even with random restarts
  - Optimization does not always improve (exploit)
  - Sometimes you need to search (explore)
- Origin: Metallurgy
- Repeated heating and cooling strengthens the blade

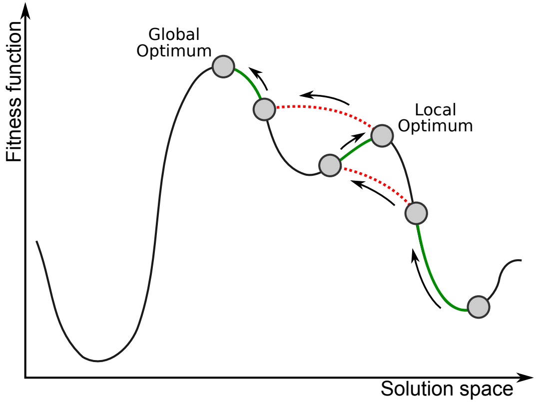


1. For a finite set of iterations
   1. Sample a new point $x_t \in N(x)$
   2. Jump to a new sample with probability given by an acceptance function $\mathbb{P}\left[x, x_t, T\right]$
   3. Decrease temperature T

$$\mathbb{P}\left[x, x_t, T\right] = \begin{cases} 1 & \mathit{if}\, f(x_t) \leq f(x) \\ \mathrm{e}^{\frac{f(x_t)-f(x)}{T}} & \mathit{otherwise} \end{cases}$$



- Green lines represent gradient-following optimizations
- Red lines represent jumps from simulated annealing
  - Possibly avoiding local optima



Properties of Simulated Annealing

- $T \to 0$: like Hill climbing
- $T \to \infty$: random walk
  - decrease T slowly

function Simulated-Annealing($problem$, $schedule$)
    $current \leftarrow problem$.Initial
    for $i \leftarrow 1$ to ∞ **do**
        $T \leftarrow schedule(t)$
        if $T = 0$ **then return** $current$
        $next \leftarrow$ a randomly selected successor of $current$
        $\Delta E \leftarrow$ Value$(current)$ $-$ Value$(next)$ $\triangleright$ *difference in desirability*
        if $DeltaE > 0$ **then** $current \leftarrow next$
        else $node \leftarrow next$ with probability $\mathrm{e}^{\frac{-\Delta E}{T}}$

Idea: escape local maxima by allowing some “bad” moves
but gradually decrease their frequency

Properties of Simulated Annealing

- At fixed "temperature" T, probability of being in any given state x approaches a Boltzman distribution

$$\mathbb{P}\left[x\right] = \alpha \mathrm{e}^{\frac{E(x)}{kT}}$$

- For every state x other than x^* and for small T

$$\mathbb{P}\left[x^*\right] / \mathbb{P}\left[x\right] = \mathrm{e}^{\frac{E(x^*)}{kT}} / \mathrm{e}^{\frac{E(x)}{kT}} = \mathrm{e}^{\frac{E(x^*) - E(x)}{kT}} \gg 1$$

- It can be shown that if we decrease T slowly enough, $\mathbb{P}\left[\text{reach } x^*\right]$ approaches 1
- Devised by Metropolis et al., 1953, for physical process modelling
  - Widely used in VLSI layout, airline scheduling, etc.

To continue in the next session.