Artificial Intelligence

Local Search

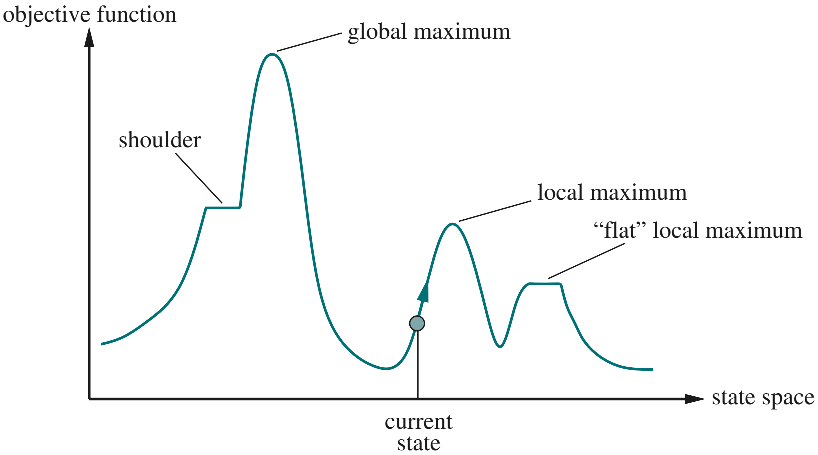
Christopher Simpkins

Local Search

If you don't care about the path to a goal state, you can use **local search**.

- Search neighbors of current state, moving to best neighbor.
- Track only current state.
- Uses very little memory.
- Can find reasonable solutions in large or infinite state spaces.
- Often used for **optmization** problems – finding states that maximize or minimize an **objective function**.

State Space Landscape



Hill-Climbing Search

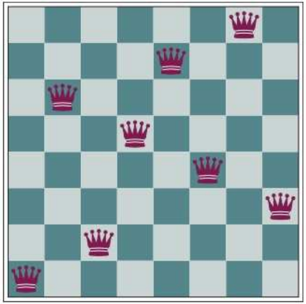
function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum
  current ← *problem*.INITIAL
  while *true* **do**
    neighbor ← a highest-valued successor state of *current*
    if VALUE(*neighbor*) $\leq$ VALUE(*current*) **then return** *current*
    current ← *neighbor*

- Also known as **greedy local search**

The 8 Queens Problem

Complete-state formulation: row position for each of 8 columns, e.g., (a) below is

`<1, 6, 2, 5, 7, 4, 8, 3>`



(a)

(b)

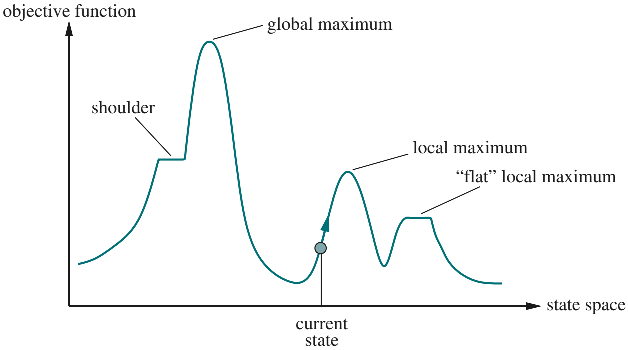
- Action: move a single queen to new row within column. Each state has $8\dot{7} = 56$ successor states.
- Possible heuristic: number of pairs of attacking queens (even if blocked). (b) above has $h = 17$.
  - Useful to remember: $\binom{n}{k} = \frac{n!}{k!(n-k)!}$

KENNESAW STATE UNIVERSITY

Disadvantages of Hill-Climbing

Susceptible to getting stuck in:

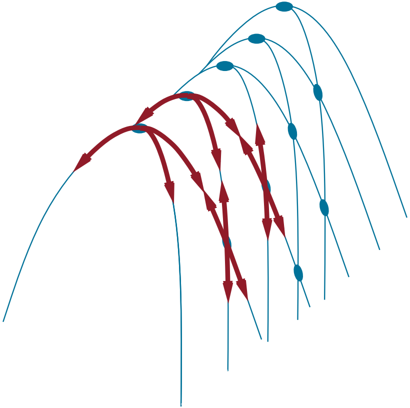
- local maxima
- ridges – sequences of local maxima
- plateaus, e.g., flat local maxima or shoulders.



How to fix:

- Allow "sideways" moves
- Stochastic hill climbing chooses randomly from uphill moves.
- Random restart hill climbing restarts from multiple initial states.

Grid of states superimposed on ridge rising from left to right.



KENNESAW STATE UNIVERSITY

Simulated Annealing

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  current ← problem.INITIAL
  for t = 1 to ∞ do
     T ← schedule(t)
     if T = 0 then return current
     next ← a randomly selected successor of current
     ΔE ← VALUE(current) – VALUE(next)
     if ΔE > 0 then current ← next
     else current ← next only with probability e^(ΔE/T)
```

- Based on annealing in metallurgy – gradually cooling metals or glass to reach a low-energy crystalline state.
- Think in terms of gradient descent.
- Similar to hill climbing, but picks a random move and
  - accepts it if its better,
  - if not better, accept with probability < 1.
- Probability of accepting a worse move depends on:
  - how much worse the move is, ΔE, and
  - the current "temperature," T.

If T decreases sufficiently slowly, then the Boltzman distribution, $e^{\frac{\Delta E}{T}}$, ensures that all the probability is concentrated on the global maxima, so the algorithm finds a global maximum with probability approaching 1.

Genetic Algorithms

A kind of **local beam search**: tracking k states instead of just one.

Elements of genetic algorithms:

- Fitness function.
- Population size.
- Candidate representation:
  - Typically a string (vector) over a finite alphabet.
  - **Evolution strategies**: sequence of real numbers.
  - **Genetic programming**: computer programs.
- Mixing number, ρ: number of "parents" from which to generate new candidates. When $\rho = 1$, stochastic beam search.
- Selection process for choosing "parents."
- Recombination procedure.
- Mutation rate.
- Composition of next generation.
  - Elitism: choose top-scoring candidates.
  - Culling: eliminate bottom-scoring candidates.

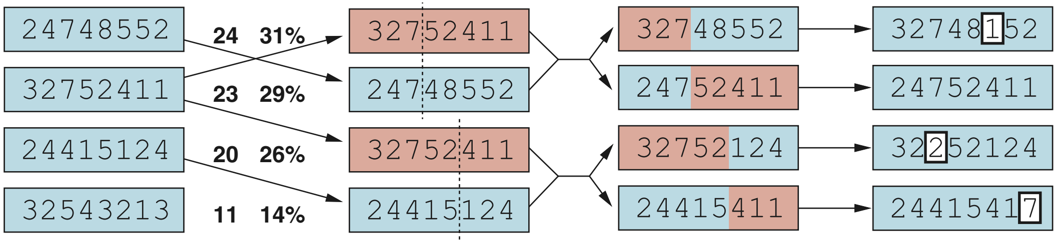
KENNESAW STATE UNIVERSITY

A Genetic Algorithm

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
  repeat
      weights ← WEIGHTED-BY(population, fitness)
      population2 ← empty list
      for i = 1 to SIZE(population) do
          parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(population, weights, 2)
          child ← REPRODUCE(parent1, parent2)
          if (small random probability) then child ← MUTATE(child)
          add child to population2
      population ← population2
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to fitness

function REPRODUCE(parent1, parent2) returns an individual
  n ← LENGTH(parent1)
  c ← random number from 1 to n
  return APPEND(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c + 1, n))
```

Genetic Algorithm on 8-Queens Problem



(a) Initial Population (b) Fitness Function (c) Selection (d) Crossover (e) Mutation

Crossover in the 8-queens Problem