Workspace for 'chapter04b-1'

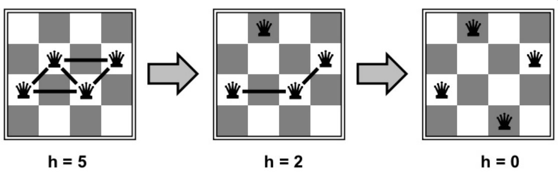
Page 1 (row 2, column 1)

Iterative improvement algorithms

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

In such cases, can use iterative improvement algorithms; keep a single "current" state, try to improve it

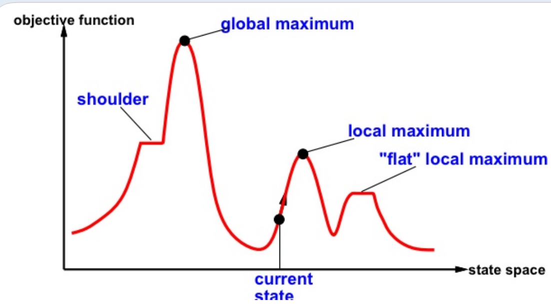
N Queen



Hill-climbing (or gradient ascent/descent)

"Like climbing Everest in thick fog with amnesia"

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node
    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
    end
```



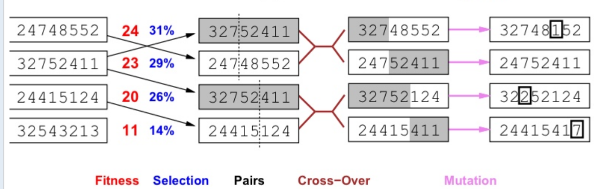
Simulated annealing

Idea: escape local maxima by allowing some "bad" moves
but gradually decrease their size and frequency

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
    inputs: problem, a problem
            schedule, a mapping from time to "temperature"
    local variables: current, a node
                     next, a node
                     T, a "temperature" controlling prob. of downward steps
    current ← MAKE-NODE(INITIAL-STATE[problem])
    for t ← 1 to ∞ do
        T ← schedule[t]
        if T = 0 then return current
        next ← a randomly selected successor of current
        ΔE ← VALUE[next] − VALUE[current]
        if ΔE > 0 then current ← next
        else current ← next only with probability e^{Δ E/T}
```

Genetic algorithms

= stochastic local beam search + generate successors from **pairs** of states



Particle Swarm Optimization

- A particle (individual) is composed of:
  - Three vectors:
    - The **x-vector** records the current position (location) of the particle in the search space,
    - The **p-vector** records the location of the best solution found so far by the particle, and
    - The **v-vector** contains a gradient (direction) for which particle will travel in if undisturbed.
  - Two fitness values:
    - The **x-fitness** records the fitness of the x-vector, and
    - The **p-fitness** records the fitness of the p-vector.

Velocity calculation Swarm search

$$v_{id(t)} = \omega v_{id(t-1)} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id})$$

Position update

$$x_{id(t)} = x_{id(t-1)} + v_{id(t)}$$

xid – current value of the dimension "d" of the dividual "i"
vid – current velocity of the dimension "d" of the individual "i".
Pid – optimal value of the dimension "d" of the individual "i" so far.
Pgd – current optimal value of the dimension "d" of the swarm

Particle Swarm Optimization
The algorithm

1. Initialise particles in the search space at random.
2. Assign random initial velocities for each particle.
3. Evaluate the fitness of each particle according a user defined objective function.
4. Calculate the new velocities for each particle.
5. Move the particles.
6. Repeat steps 3 to 5 until a predefined stopping criterion is satisfied.