Lecture 8: Optimization and hill-climbing

Artificial Intelligence
CS-GY-6613

Julian Togelius
julian.togelius@nyu.edu

On the menu:

- Optimization versus tree search
- Hill-climbing
- Simulated annealing
- Evolutionary algorithms

Tree search versus optimization

Local search algorithms

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

- State space = set of "complete" configurations

- Find configuration satisfying constraints, e.g., n-queens

- In such cases, we can use local search algorithms: keep a single "current" state, try to improve it

n-queens

- Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal

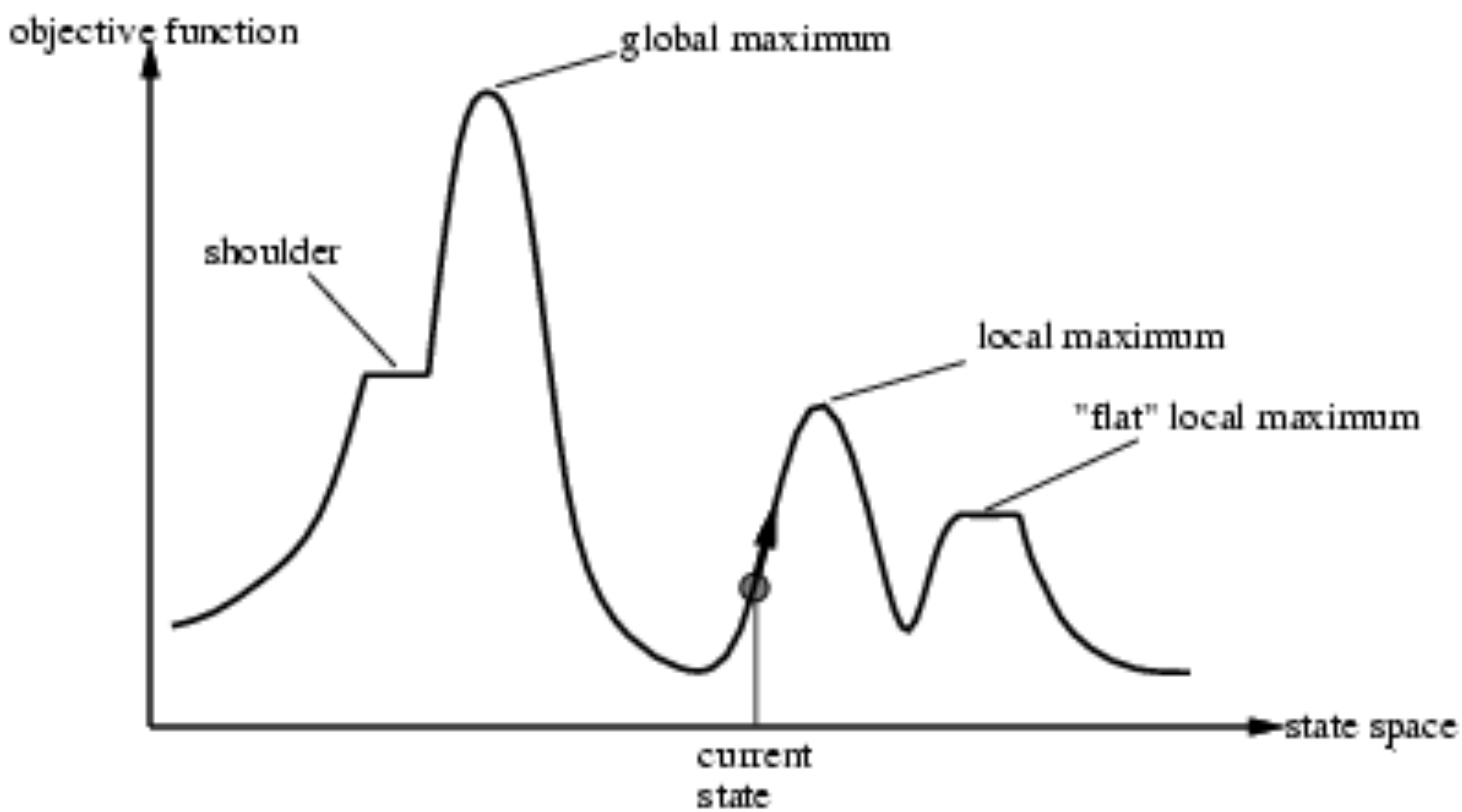
Hill-climbing

```
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
```

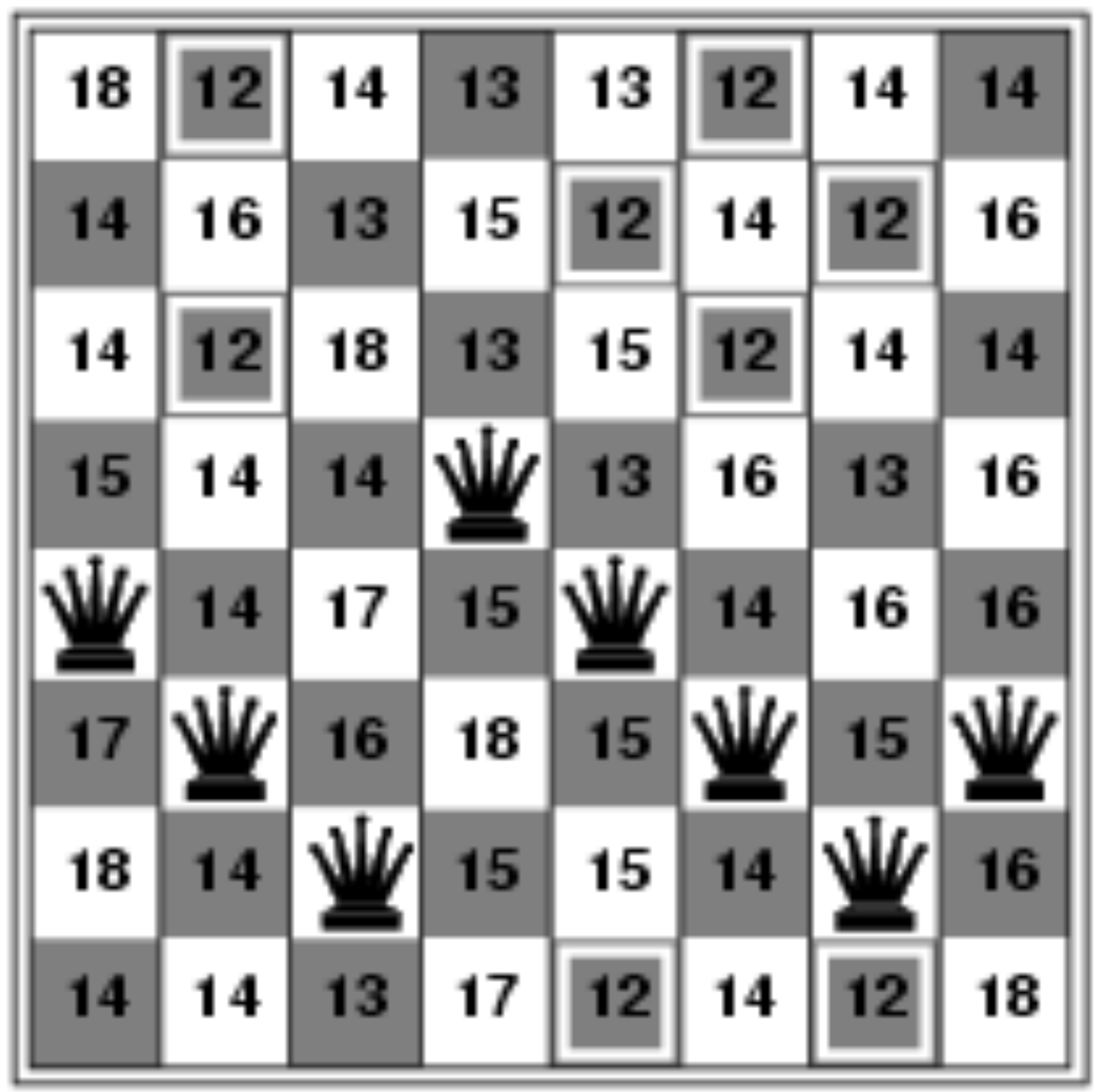
Hill-climbing

- Can get stuck in local maxima/minima

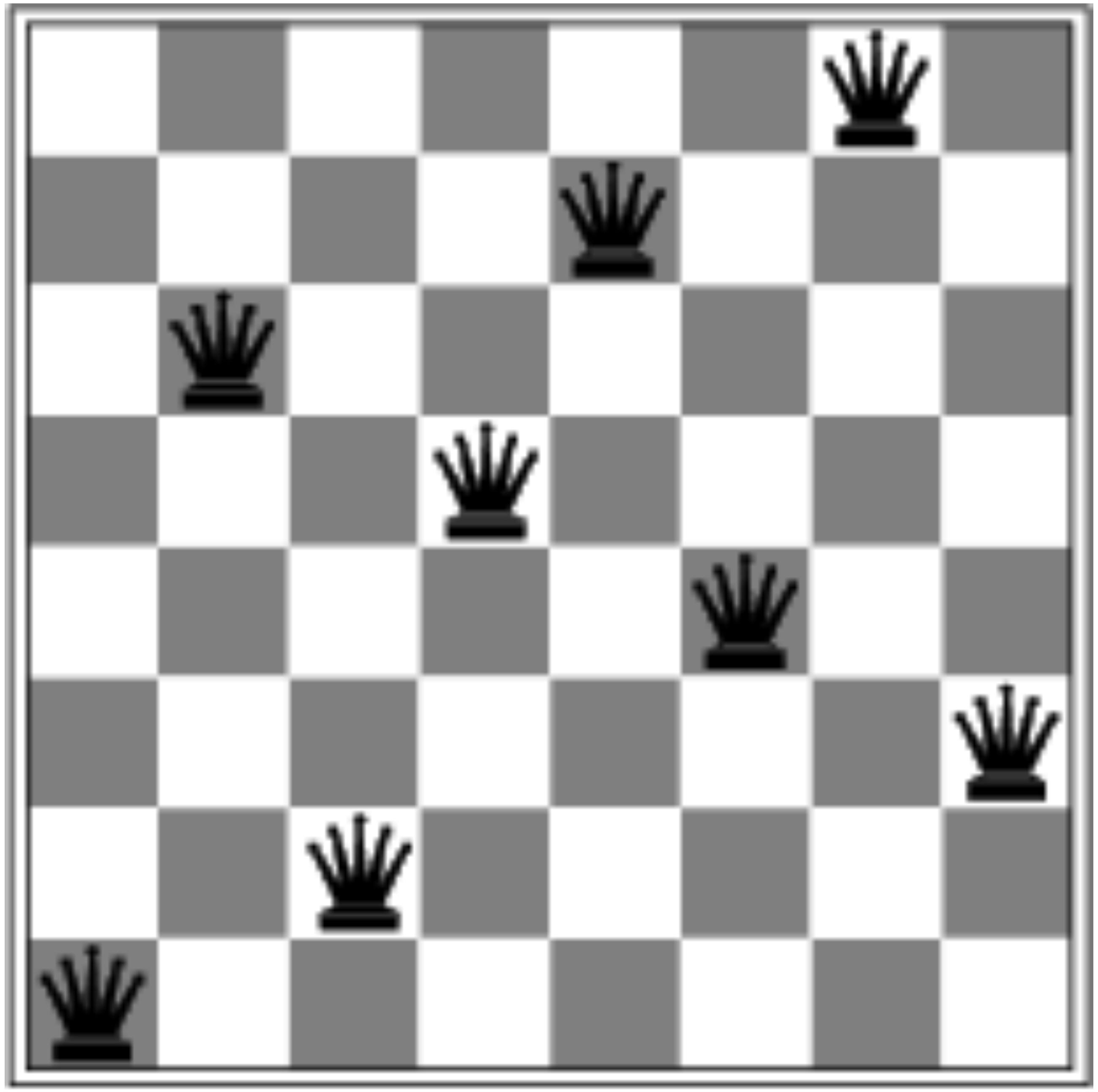


Hill-climbing 8-queens

- *h* = number of pairs of queens that are attacking each other, either directly or indirectly. Here, *h=17*.



Local minimum



Hill-climbing variations

- Simple hill climber: generates one successor, accepts it if better than the current solution
- “Permissive” hill climber: accepts the successor, accepts it if *equal to or* better than the current solution
- Steepest ascent hill climber: generates *all* successors, chooses the best one

Optimization in general

- Optimal: no
- Complete: no
- Space: reasonable
- Time: who knows

Simulated annealing

- Do bad moves with decreasing probability

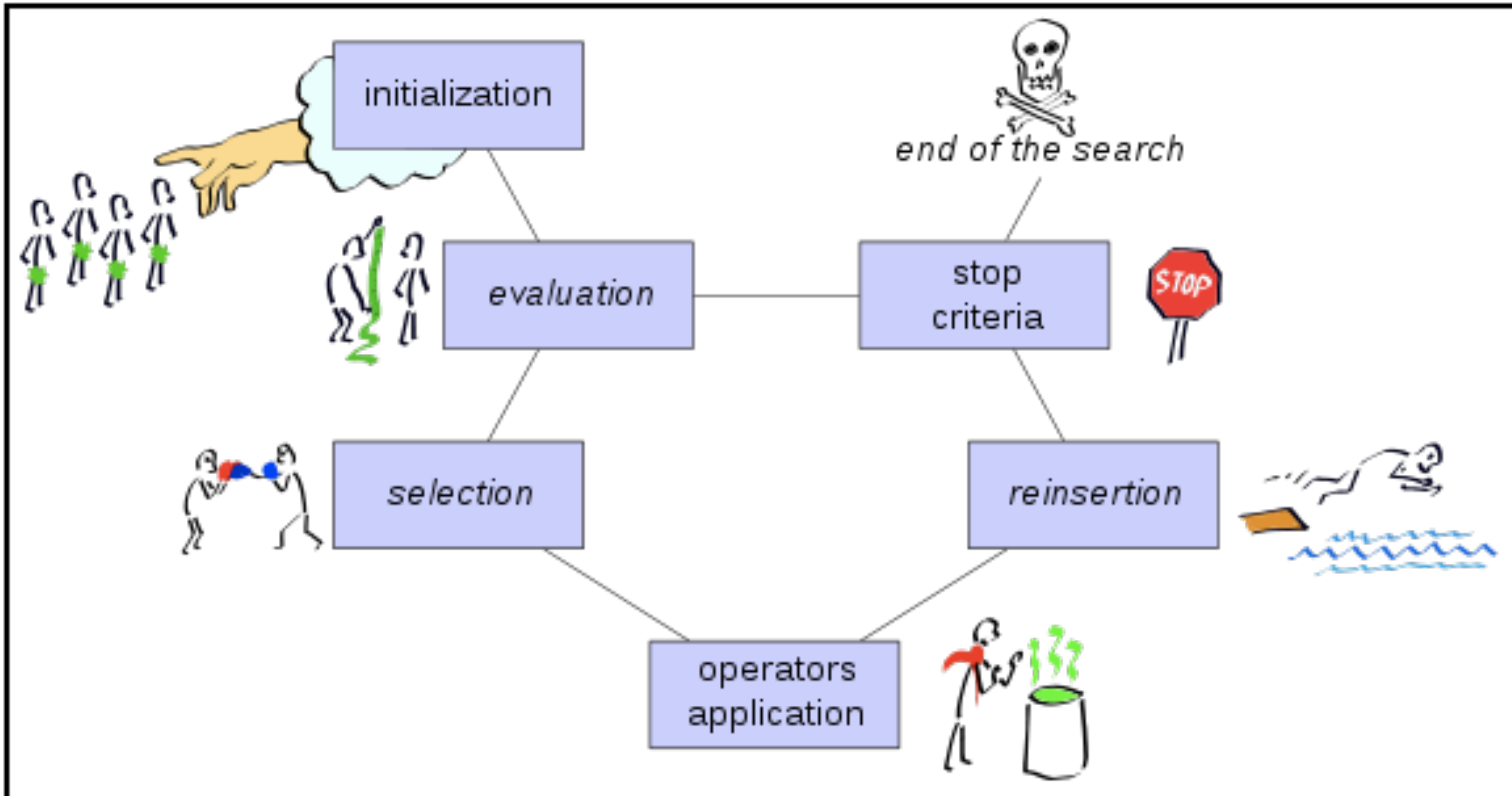
```
function SIMULATED-ANNEALING( problem, schedule) returns a solution state
   inputs: problem, a problem
           schedule, a mapping from time to "temperature"
   local variables: current, a node
                    next, a node
                    T, a "temperature" controlling prob. of downward 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^(ΔE/T)
```

Simulated annealing

- One can prove: If T decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

Evolutionary computation



General schema of an Evolutionary Algorithm (EA)

More about evolutionary computation next time!