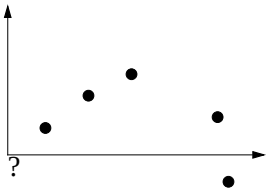
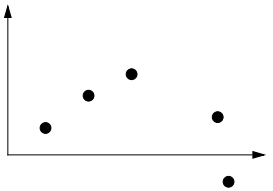
Artificial Intelligence
Other Search and Optimization Methods

Russell and Norvig
Chapter 4

prediction
predict f(x) for a given x



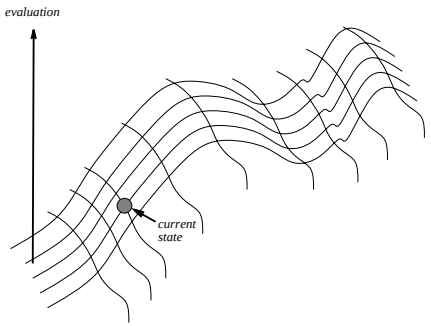
optimization
find an x so that f(x) is maximal



is this search?
is this reinforcement learning?

playground example

gradient ascent (hillclimbing)



gradient ascent (hillclimbing)

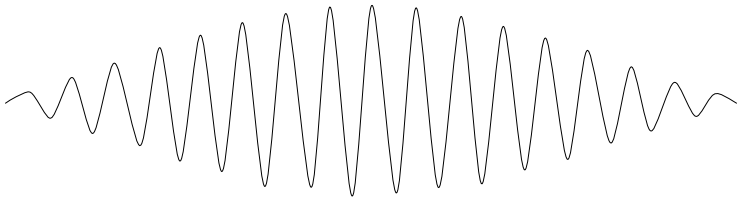
```
function HILL-CLIMBING(problem) returns a solution state
  inputs: problem, a problem
  static: current, a node
          next, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
     next ← a highest-valued successor of current
     if VALUE[next] < VALUE[current] then return current
     current ← next
  end
```

$$x_i = x_i + \alpha \, dVALUE/dx_i$$

learning rate (small positive constant)

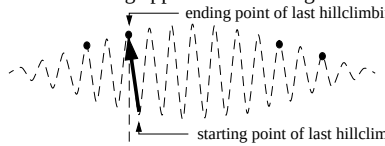
problems with hillclimbing

problems:
- local maxima -> random restarts
- plateaus -> random restarts
- ridges



problems with hillclimbing
one solution: STAGE [Boyan and Moore]

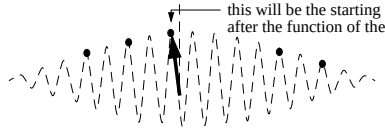
1. remember the maxima of all hillclimbing applications to the given function



2. estimate a function of the maxima

3. Stage 1: use the ending point as a starting point for hillclimbing on the function of the maxima

4. Stage 2: use the ending point as a starting point for hillclimbing on the given function

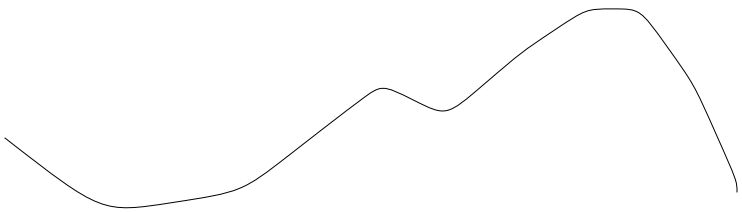


5. go to 2

simulated annealing

annealing = the process of gradually cooling a liquid until it freezes
if the temperature is lowered sufficiently slowly,
the material will attain a lowest-energy (perfectly ordered) configuration

hillclimbing with going downhill (from time to time)



VLSI layout and lots of other applications

simulated annealing

VALUE[] = total energy of the atoms in the material
T = temperature

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"
  static: *current*, a node
    next, a node
    T, a "temperature" controlling the probability 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*
    Δ*E* ← VALUE[*next*] – VALUE[*current*]
    if Δ*E* > 0 **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

genetic algorithms and evolutionary programming

hillclimbing with going downhill and parallel search

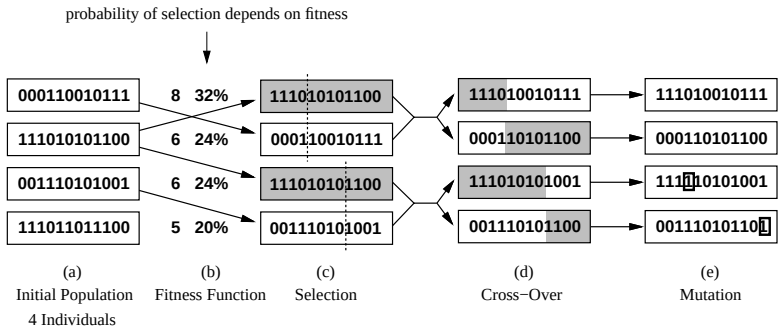
have a group of individuals

offsprings are “genetic mixtures” of their parents

some random mutations occur

if the performance of an individual is bad,
it dies early and thus doesn’t reproduce very often

genetic algorithms and evolutionary programming



? genetic algorithms are the third best way of doing just about anything ?