ARTIFICIAL INTELLIGENCE (AI)
STATE SPACE AND SEARCH ALGORITHMS

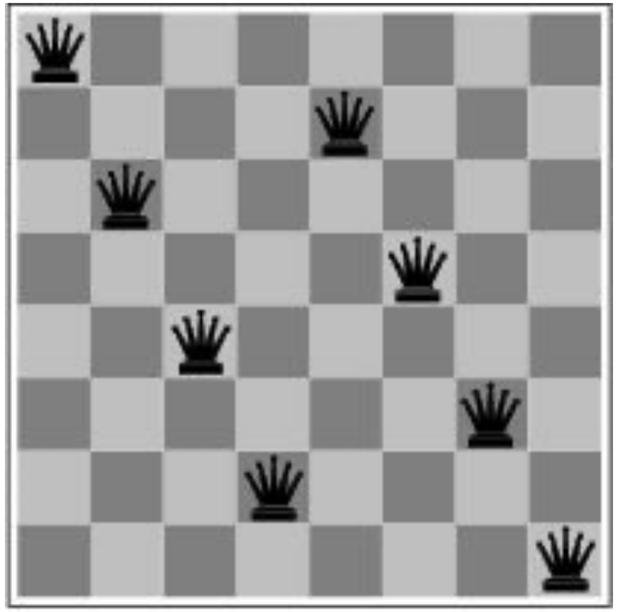
Shreya Ghosh
Assistant Professor, Computer Science and Engineering
IIT Bhubaneswar

LOCAL SEARCH AND OPTIMIZATION

- Previous lecture: path to goal is solution to problem
  - systematic exploration of search space.

- This lecture: a state is solution to problem
  - for some problems path is irrelevant.
  - E.g., 8-queens

- Different algorithms can be used
  - Depth First Branch and Bound
  - Local search



Goal Satisfaction

reach the goal node
Constraint satisfaction

Optimization

optimize(objective fn)
Constraint Optimization

You can go back and forth between the two problems
Typically in the same complexity class

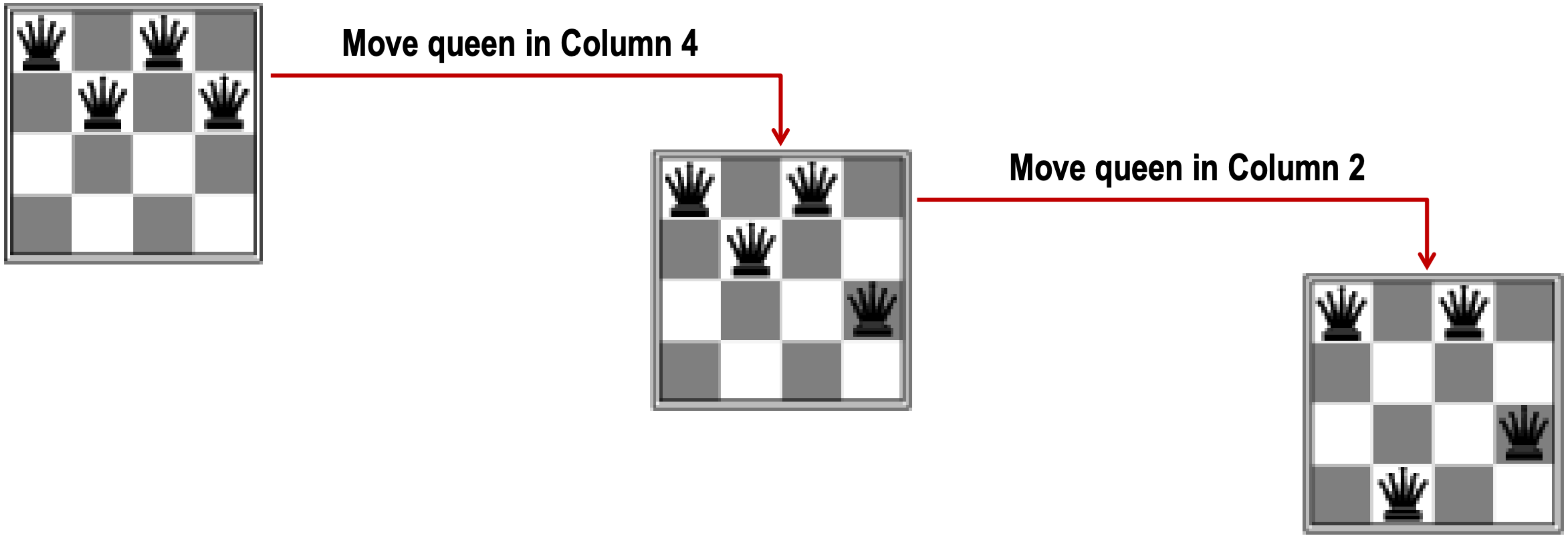
LOCAL SEARCH AND OPTIMIZATION

- Local search
  - Keep track of single current state
  - Move only to neighboring states
  - Ignore paths

- Advantages:
  - Use very little memory
  - Can often find reasonable solutions in large or infinite (continuous) state spaces.

- “Pure optimization” problems
  - All states have an objective function
  - Goal is to find state with max (or min) objective value
  - Does not quite fit into path-cost/goal-state formulation
  - Local search can do quite well on these problems.

4-queens Problem



HILL-CLIMBING (GREEDY LOCAL SEARCH) MAX VERSION

function HILL-CLIMBING(*problem*) **return** a state that is a local maximum

input: *problem*, a problem

local variables: *current*, **a node.**

neighbor, **a node.**

current ← MAKE-NODE(INITIAL-STATE[*problem*])

loop do

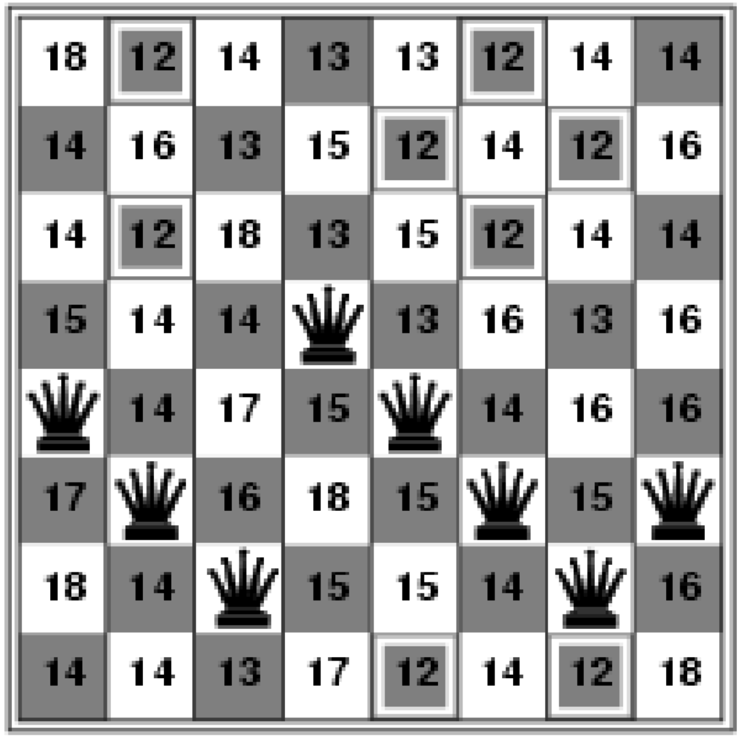
neighbor ← a highest valued successor of *current*

if VALUE [*neighbor*] ≤ VALUE[*current*] **then return** STATE[*current*]

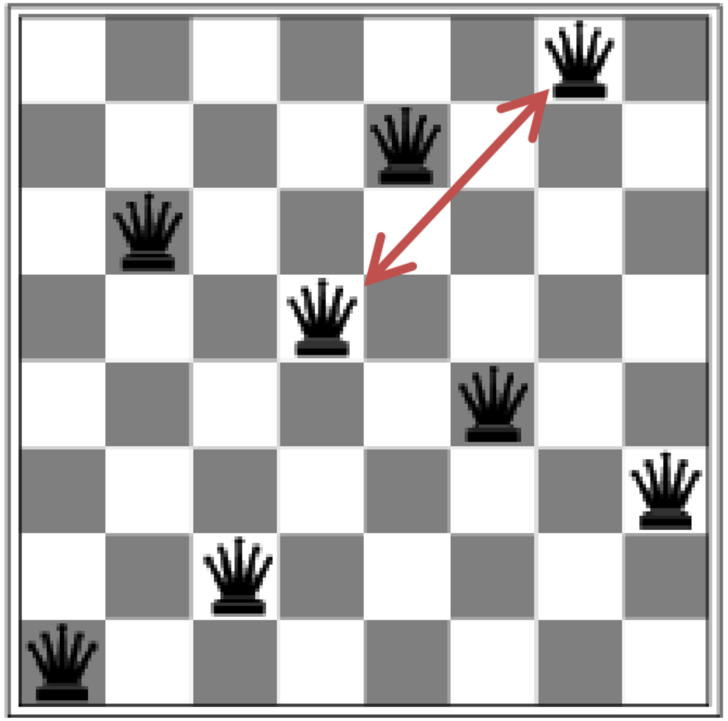
current ← *neighbor*

min version will reverse inequalities and look for lowest valued successor

GRADIENT DESCENT IN 8-QUEENS



- A local minimum with only one conflict
- All one-step neighbors have more than one conflict



How to get out of local minima?

HILL-CLIMBING SEARCH

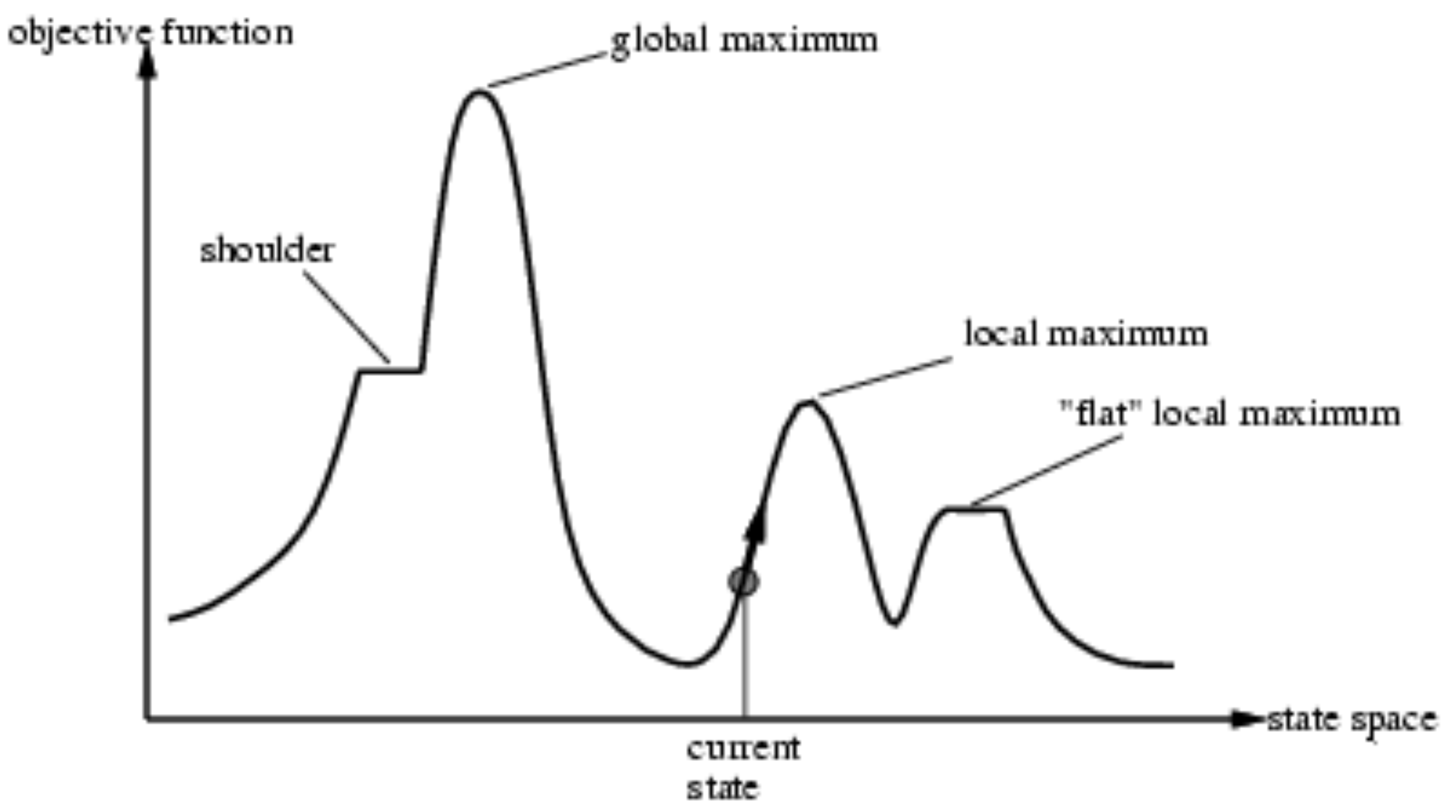
- “a loop that continuously moves towards increasing value”
  - terminates when a peak is reached
  - Aka greedy local search

- Value can be either
  - Objective function value
  - Heuristic function value (minimized)

- Hill climbing does not look ahead of the immediate neighbors
- Can randomly choose among the set of best successors
  - if multiple have the best value

“climbing Mount Everest in a thick fog”

“LANDSCAPE” OF SEARCH



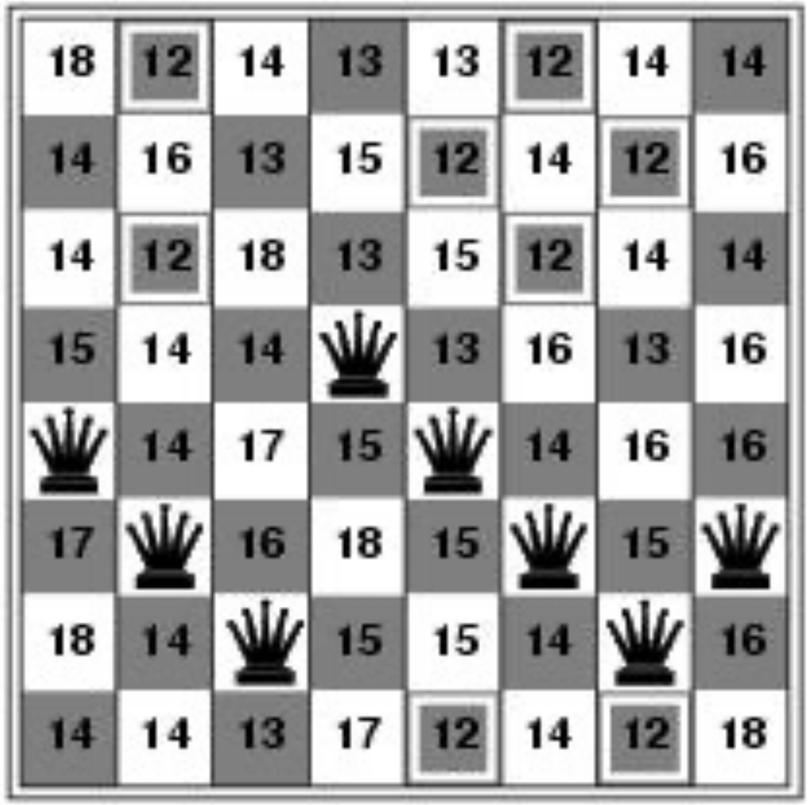
Hill Climbing gets stuck in local minima

depending on?

EXAMPLE: *N*-QUEENS

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

HILL-CLIMBING SEARCH: 8-QUEENS PROBLEM

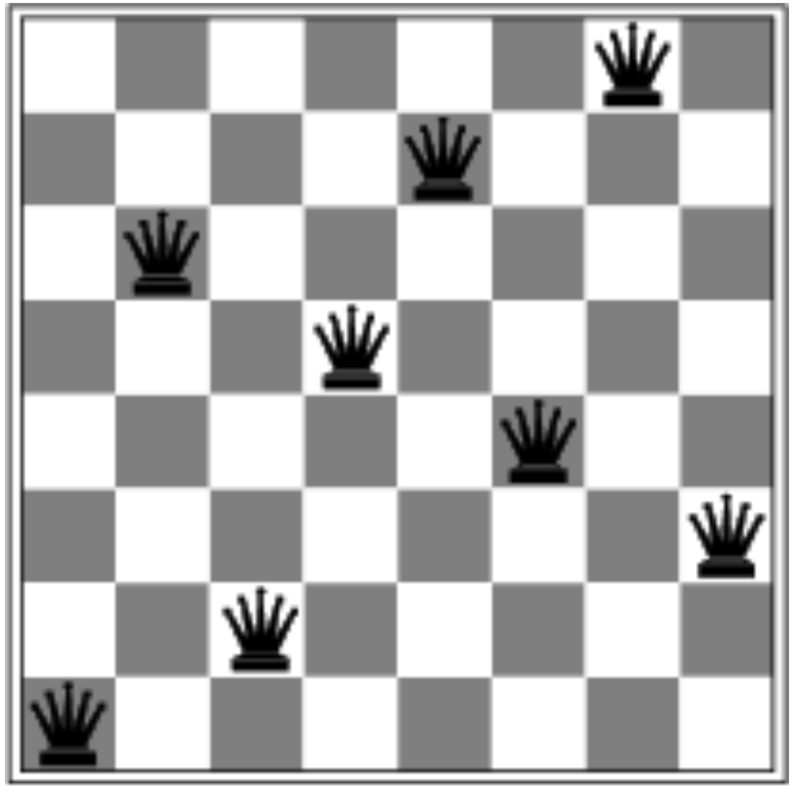


- Need to convert to an optimization problem
- h = number of pairs of queens that are attacking each other
- $h = 17$ for the above state

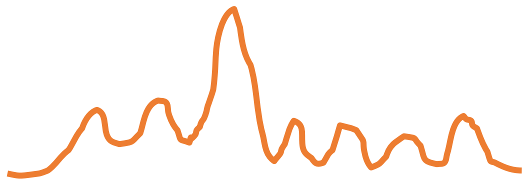
SEARCH SPACE

- State
  - All 8 queens on the board in some configuration
- Successor function
  - move a single queen to another square in the same column.
- Example of a heuristic function $h(n)$:
  - the number of pairs of queens that are attacking each other
  - (so we want to minimize this)

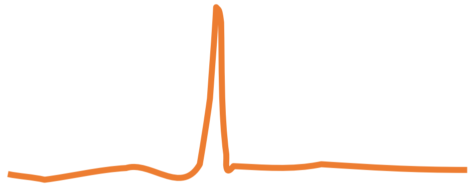
HILL-CLIMBING SEARCH: 8-QUEENS PROBLEM



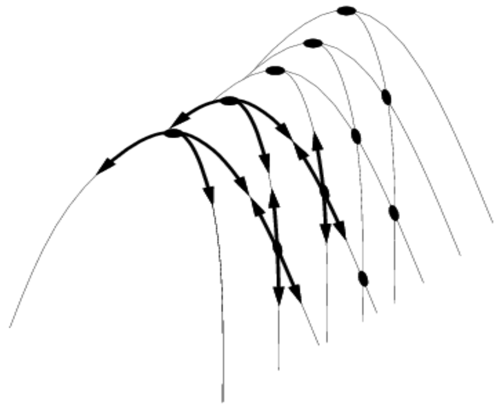
- Local maxima



- Plateaus



- Diagonal ridges



HILL-CLIMBING WITH RANDOM RESTARTS

- If at first you don't succeed, try, try again!
- Different variations
  - For each restart: run until termination vs. run for a fixed time
  - Run a fixed number of restarts or run indefinitely
- Analysis
  - Say each search has probability p of success
    - E.g., for 8-queens, $p = 0.14$ with no sideways moves
  - Expected number of restarts?
  - Expected number of steps taken?
- If you want to pick one local search algorithm, learn this one!!

HILL-CLIMBING WITH RANDOM WALK (ADDITIONAL)

- At each step do one of the two
  - Greedy: With prob p move to the neighbor with largest value
  - Random: With prob 1-p move to a random neighbor

Hill-climbing with both

- At each step do one of the three
  - Greedy: move to the neighbor with largest value
  - Random Walk: move to a random neighbor
  - Random Restart: Resample a new current state

SIMULATED ANNEALING

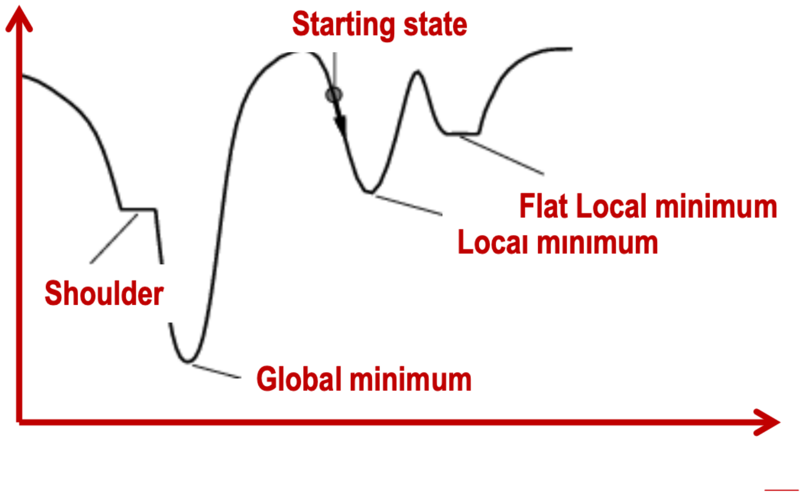
- Simulated Annealing = physics inspired twist on random walk
- Basic ideas:
  - like hill-climbing identify the quality of the local improvements
  - instead of picking the best move, pick one randomly
  - say the change in objective function is δ
  - if δ is positive, then move to that state
  - otherwise:
    - move to this state with probability proportional to δ
    - thus: worse moves (very large negative δ) are executed less often
  - however, there is always a chance of escaping from local maxima
  - over time, make it less likely to accept locally bad moves
  - (Can also make the size of the move random as well, i.e., allow “large” steps in state space)

Simulated annealing search

IDEA: Escape local maxima by allowing some "bad" moves but **gradually decrease** their probability

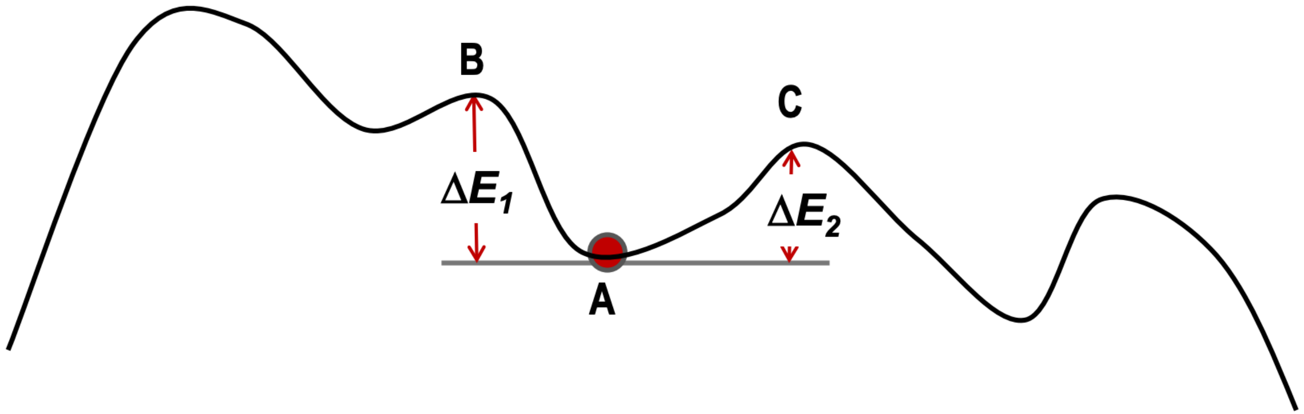
- The probability is controlled by a parameter called *temperature*
- *Higher temperatures allow more bad moves than lower temperatures*
- **Annealing:** Lowering the temperature gradually **Quenching:** Lowering the temperature rapidly

```
function SIMULATED-ANNEALING( problem, schedule )
    current ← 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 with probability e^(−ΔE/T)
```



How simulated annealing works

Probability of making a bad move = $e^{-\Delta E/T} = \dfrac{1}{e^{\Delta E/T}}$



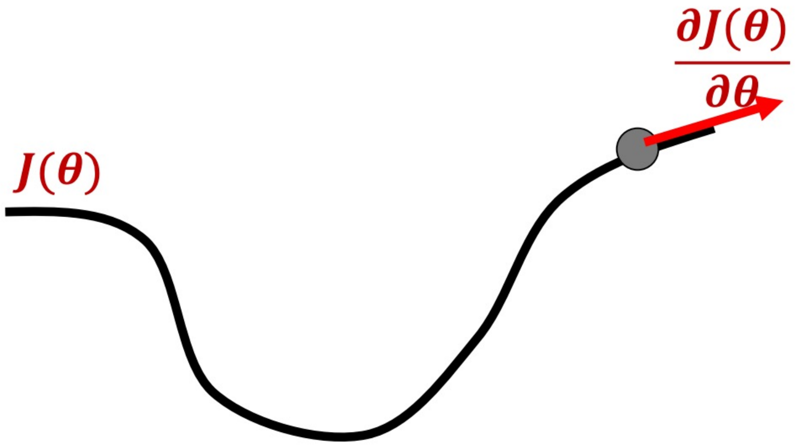
Since $\Delta E_1 > \Delta E_2$ moving from A to C is exponentially more probable than moving from A to B

Properties of Simulated Annealing

- It can be proven that:
  - If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1
  - Since this can take a long time, we typically use a temperature schedule which fits our time budget and settle for the sub-optimal solution
- Simulated annealing works very well in practice
- Widely used in VLSI layout, airline scheduling, etc.

Hill Climbing in Continuous Multi-variate State Spaces

Denote "state" as μ; cost as J(μ)



- Choose a direction in which J(μ) is decreasing
- Derivative: $\frac{\partial J(\theta)}{\partial \theta}$
  - Positive derivative means increasing
  - Negative derivative means decreasing
- Move: A short uphill step in the chosen direction

PHYSICAL INTERPRETATION OF SIMULATED ANNEALING

- A Physical Analogy:
  - imagine letting a ball roll downhill on the function surface
    - this is like hill-climbing (for minimization)
  - now imagine shaking the surface, while the ball rolls, gradually reducing the amount of shaking
    - this is like simulated annealing

- Annealing = physical process of cooling a liquid or metal until particles achieve a certain frozen crystal state
  - simulated annealing:
    - free variables are like particles
    - seek "low energy" (high quality) configuration
    - slowly reducing temp. T with particles moving around randomly