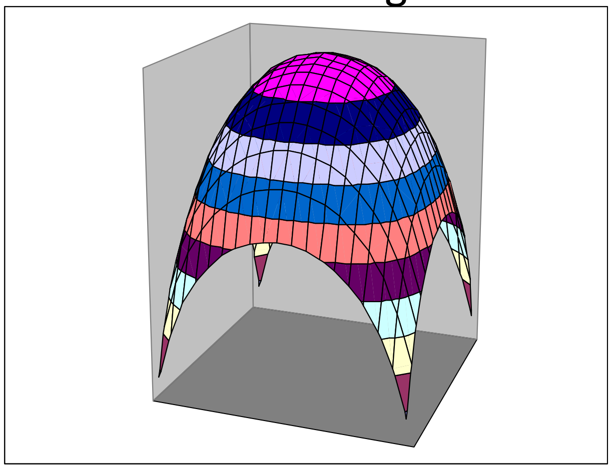
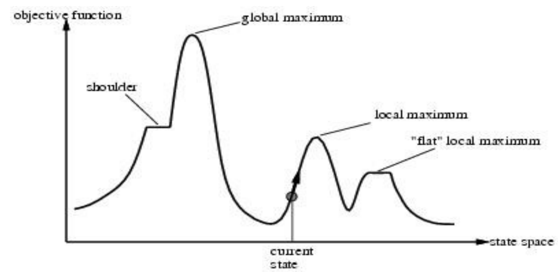
Hill Climbing



"Landscape" of search for max value



Hill Climbing - Algorithm

1. **Pick a random point in the search space**
2. **Consider all the neighbours of the current state**
3. **Choose the neighbour with the best quality and move to that state**
4. **Repeat 2 thru 4 until all the neighbouring states are of lower quality**
5. **Return the current state as the solution state**

Hill Climbing - Algorithm

Function HILL-CLIMBING(*Problem*) returns a solution state

Inputs:*Problem*, problem

Local variables:*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*

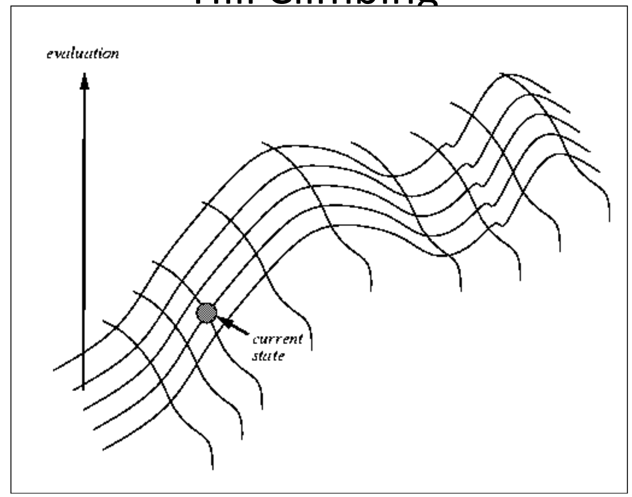
Current = *Next*

End

Hill-climbing search

- "a loop that continuously moves in the direction of 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 of the current state.
- Can randomly choose among the set of best successors, if multiple have the best value
- Characterized as "trying to find the top of Mount Everest while in a thick fog"

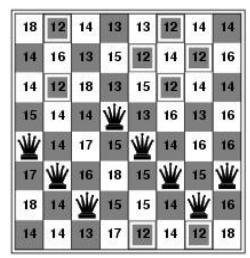
Hill Climbing



Hill-climbing example

- 8-queens problem, complete-state formulation
  - 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 (directly or indirectly)
  - (so we want to minimize this)

Hill-climbing example

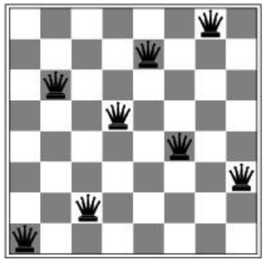


$(c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_8) = (5\ 6\ 7\ 4\ 5\ 6\ 7\ 6)$

Current state: h=17

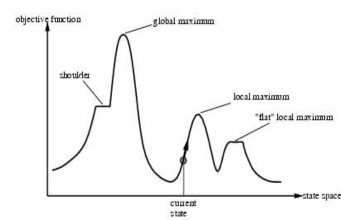
Shown is the h-value for each possible successor in each column

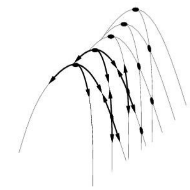
A local minimum for 8-queens



A local minimum in the 8-queens state space (h=1)

Other drawbacks





- Ridge = sequence of local maxima difficult for greedy algorithms to navigate
- Plateau = an area of the state space where the evaluation function is flat.

Performance of hill-climbing on 8-queens

- Randomly generated 8-queens starting states…
- 14% the time it solves the problem
- 86% of the time it get stuck at a local minimum
- However…
  – Takes only 4 steps on average when it succeeds
  – And 3 on average when it gets stuck
  – (for a state space with ~17 million states)

Possible solution…sideways moves

- If no downhill (uphill) moves, allow sideways moves in hope that algorithm can escape
  – Need to place a limit on the possible number of sideways moves to avoid infinite loops
- For 8-queens
  – Now allow sideways moves with a limit of 100
  – Raises percentage of problem instances solved from 14 to 94%
  – However….
    - 21 steps for every successful solution
    - 64 for each failure

Hill-climbing variations

- Stochastic hill-climbing
  - Random selection among the uphill moves.
  - The selection probability can vary with the steepness of the uphill move.
- First-choice hill-climbing
  - stochastic hill climbing by generating successors randomly until a better one is found
  - Useful when there are a very large number of successors
- Random-restart hill-climbing
  - Tries to avoid getting stuck in local maxima.

Hill-climbing with random restarts

- Different variations
  - For each restart: run until termination v. 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?

Local beam search

- Keep track of *k* states instead of one
  - Initially: *k* randomly selected states
  - Next: determine all successors of *k* states
  - If any of successors is goal → finished
  - Else select *k* best from successors and repeat.
- Major difference with random-restart search
  - Information is shared among *k* search threads.
- Can suffer from lack of diversity.
  - Stochastic beam search
    - choose k successors proportional to state quality.

Search using Simulated Annealing

- Simulated Annealing = hill-climbing with non-deterministic search

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)

Physical Interpretation of 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
    - get this by slowly reducing temperature T, which particles move around randomly

Simulated annealing

function SIMULATED-ANNEALING(*problem, schedule*) **return** a solution state
  input: *problem*, a problem
    schedule, a mapping from time to temperature
  local variables: *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*
    ∆E ← VALUE[*next*] - VALUE[*current*]
    if *∆E* > 0 **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E /T}$