LOCAL SEARCH

CSE 511A: Introduction to Artificial Intelligence

LOCAL SEARCH

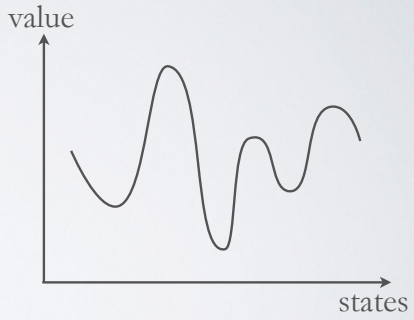
- In many applications, the path to the goal is irrelevant. Only the goal state is the solution.
- Examples: TSP, n-queens

- Use iterative improvement algorithms:
  - Keep a current state and try to improve it

HILL CLIMBING

Hill climbing idea:

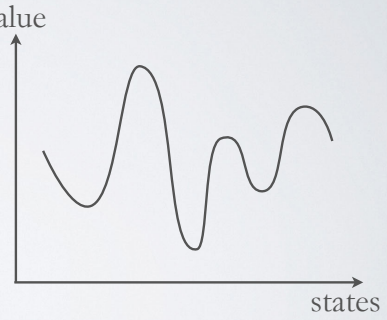
- Keep on improving your current state until you cannot improve it further
- Keep on climbing the hill you are on until you reach the peak



HILL CLIMBING

Hill climbing pseudo-code:

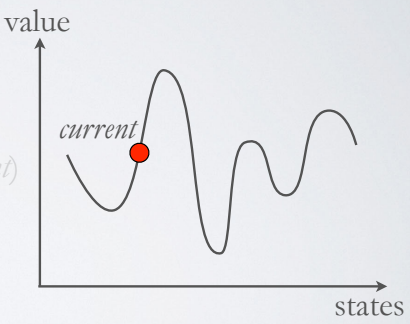
(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)



HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

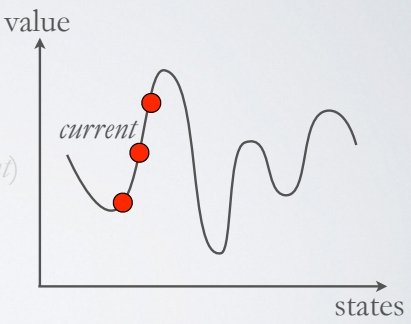


5

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

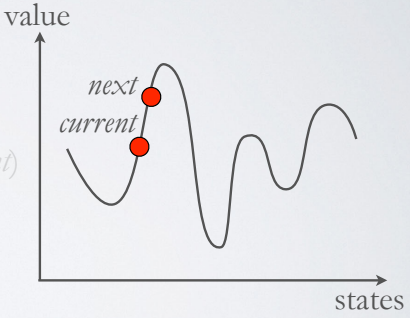


6

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

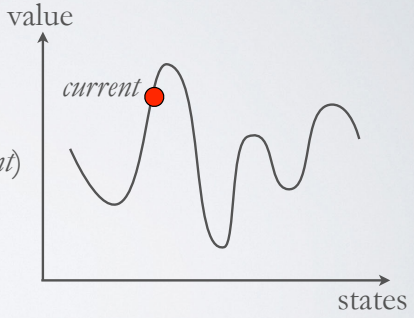


7

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

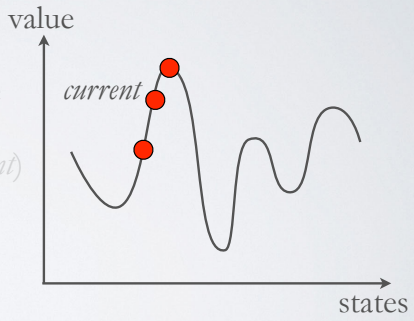


8

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

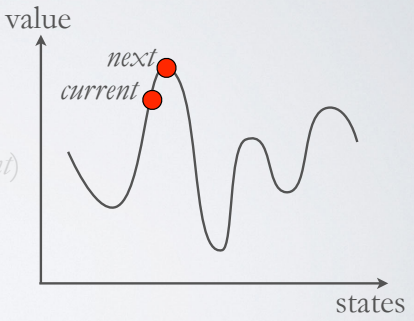


9

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

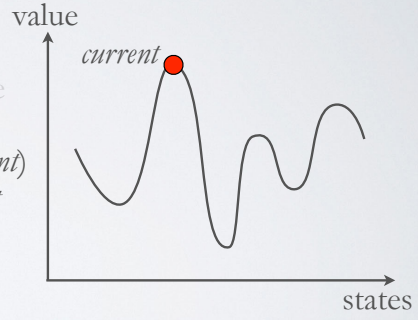


10

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

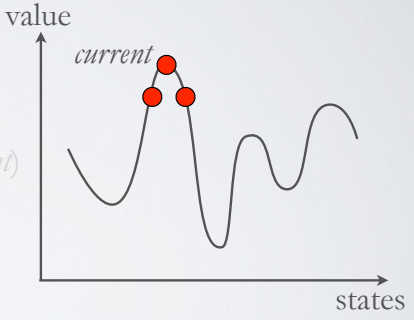


11

HILL CLIMBING

Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)

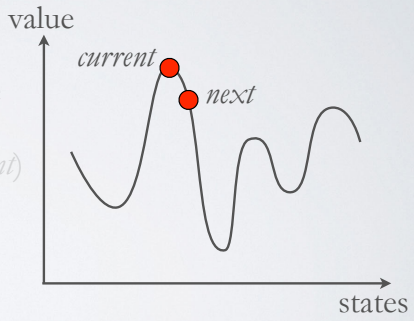


12

HILL CLIMBING

Hill climbing pseudo-code:

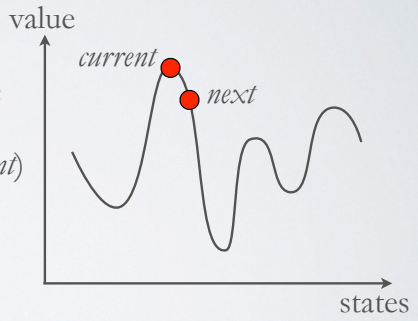
(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)



HILL CLIMBING

Hill climbing pseudo-code:

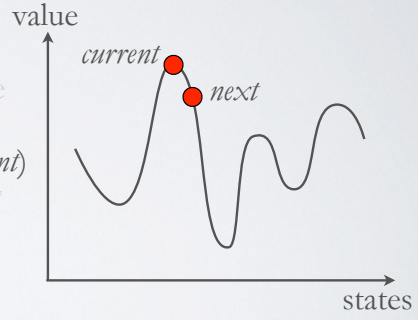
(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)



HILL CLIMBING

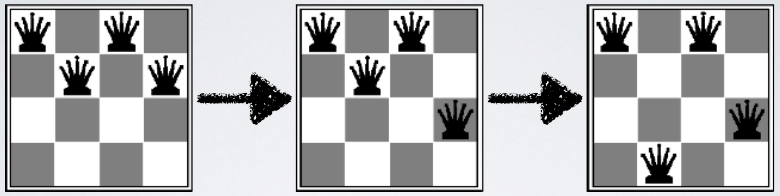
Hill climbing pseudo-code:

(1) *current* = any "complete" state
(2) *next* = best neighboring state
(3) ΔE = value(*next*) - value(*current*)
(4) If $\Delta E > 0$, then *current* = *next*
(5) Else, return *current*
(6) Go to (2)



Problem: Get stuck in local maxima

HILL CLIMBING



n-queens problem:
put *n* queens on an *n* x *n* board, with no two queens on the same row, column or diagonal

Can solve very large *n*-queens problems ($n \approx 1$ million) very quickly!

SIMULATED ANNEALING

Simulated annealing idea:

- Accept "bad moves" once in a while in order to escape local maximum
- Accept them with decreasing probability over time

SIMULATED ANNEALING

Simulated annealing pseudo-code:

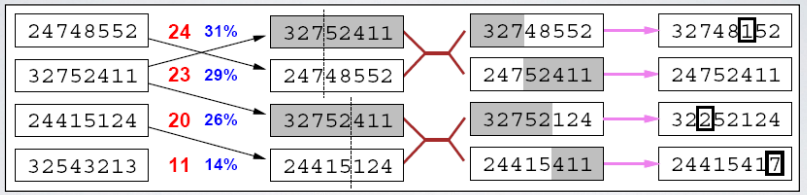
(1) T = a high temperature
(2) *current* = any "complete" state
(3) decrement T; if $T = 0$, then return *current*
(4) *next* = random neighboring state
(5) ΔE = value(*next*) - value(*current*)
(6) If $\Delta E > 0$, then *current* = *next*
(7) Else, *current* = *next* with probability $e^{\Delta E/T}$
(8) Go to (3)

SIMULATED ANNEALING

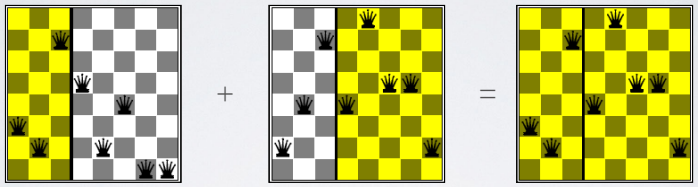
Behavior:

- If temperature is high:
  - accept bad move with high probability
  - similar to random walk
- If temperature is low:
  - accept bad move with low probability
  - similar to hill climbing

GENETIC ALGORITHM



GENETIC ALGORITHM



Example crossover with the *n*-queens problem