Masayu Leylia Khodra

KK IF – Teknik Informatika – STEI ITB

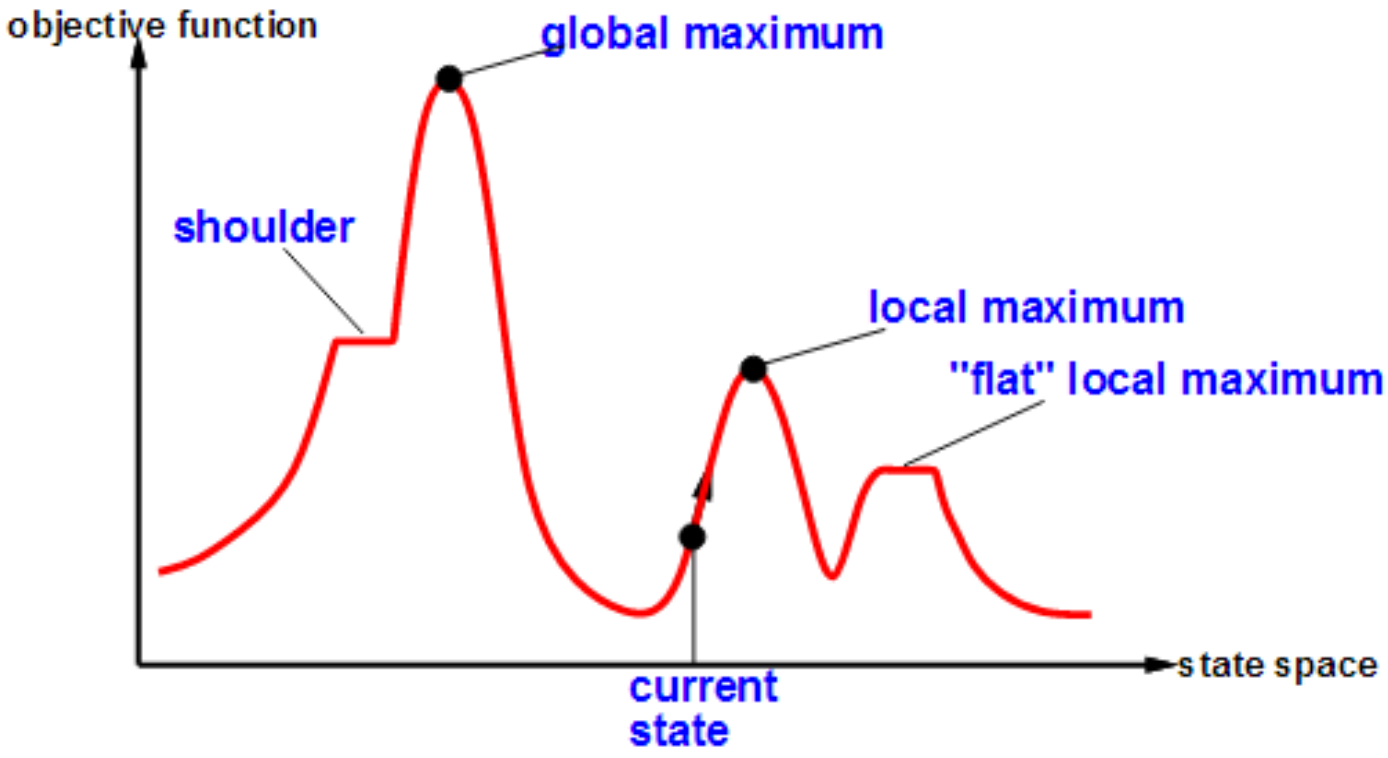
Modul 3: Beyond Classical Search

Hill-climbing Search

Inteligensi Buatan
(*Artificial Intelligence*)

Hill-climbing Search

"Like climbing Everest in thick fog with amnesia"



Starting from a randomly generated initial state

Loop that continually moves in the direction of increasing value (objective) or decreasing value (cost)

Terminates when it reaches a "peak" where no neighbor has a higher value

Hill-climbing Search: Steepest Ascent

(Russel & Norvig, 2010)

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

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
  neighbor ← a highest-valued successor of *current*
  if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
  current ← *neighbor*

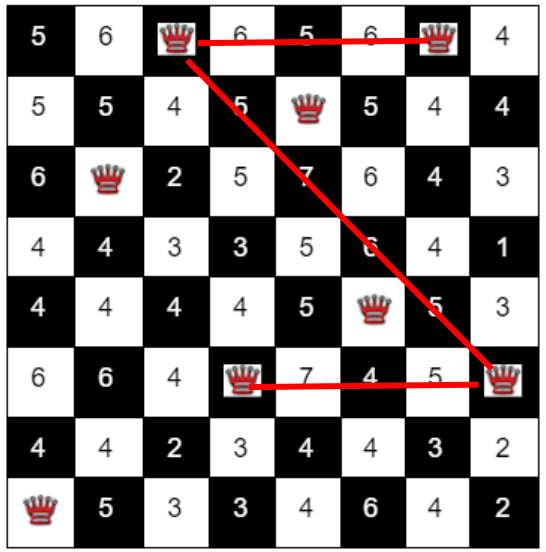
Starting from a randomly generated initial state

Loop that continually moves in the direction of increasing value (objective) or decreasing value (cost)

Terminates when it reaches a **"peak"** including **"flat"** where no neighbor has a higher value

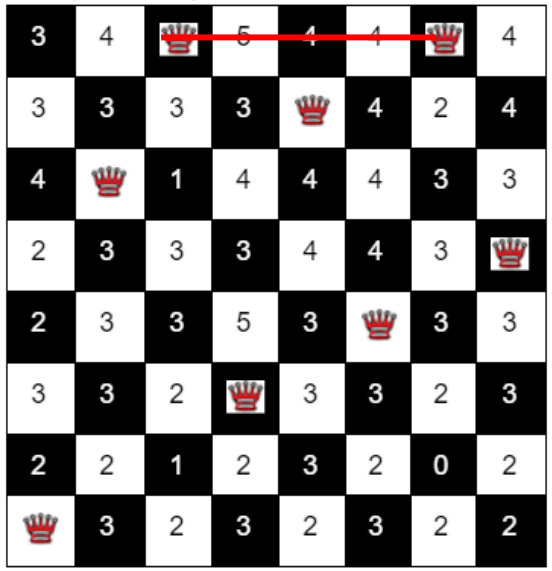
Hill-climbing: Illustration

h=-3



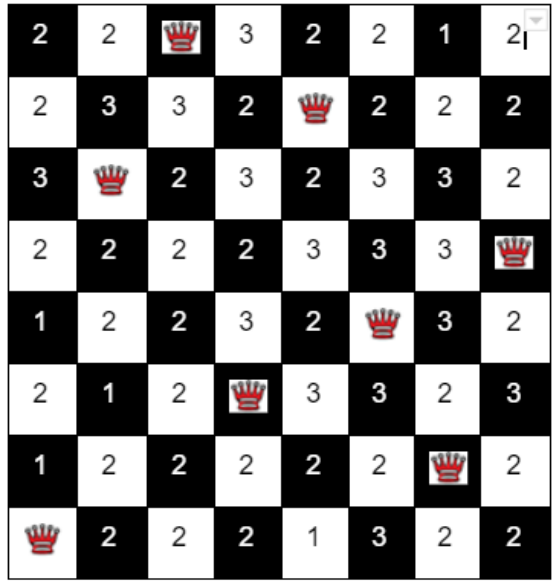
Q8: 3→5

h=-1 (Q8: 3 →5)



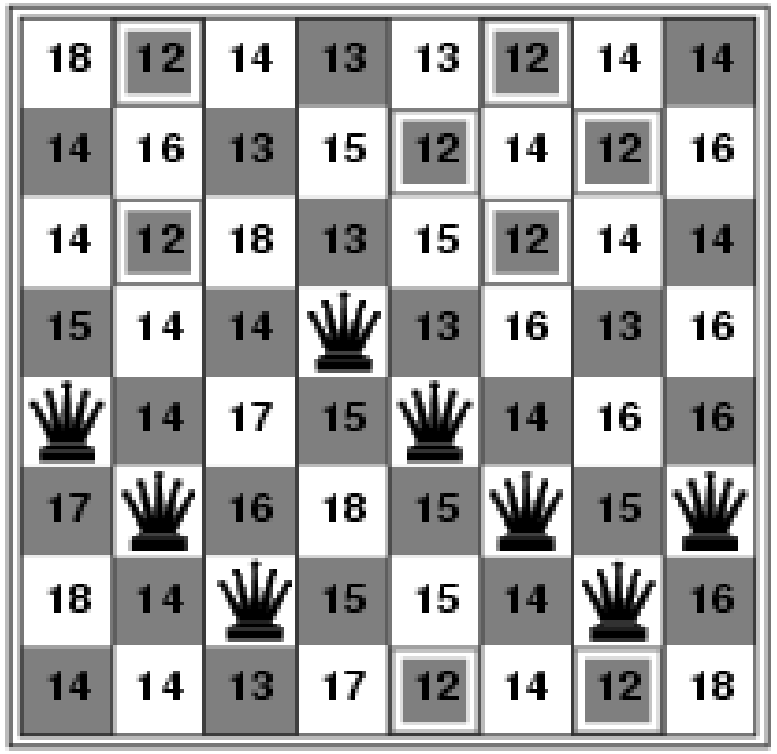
Q7: 8→2

h=0 (Q7: 8 →2)



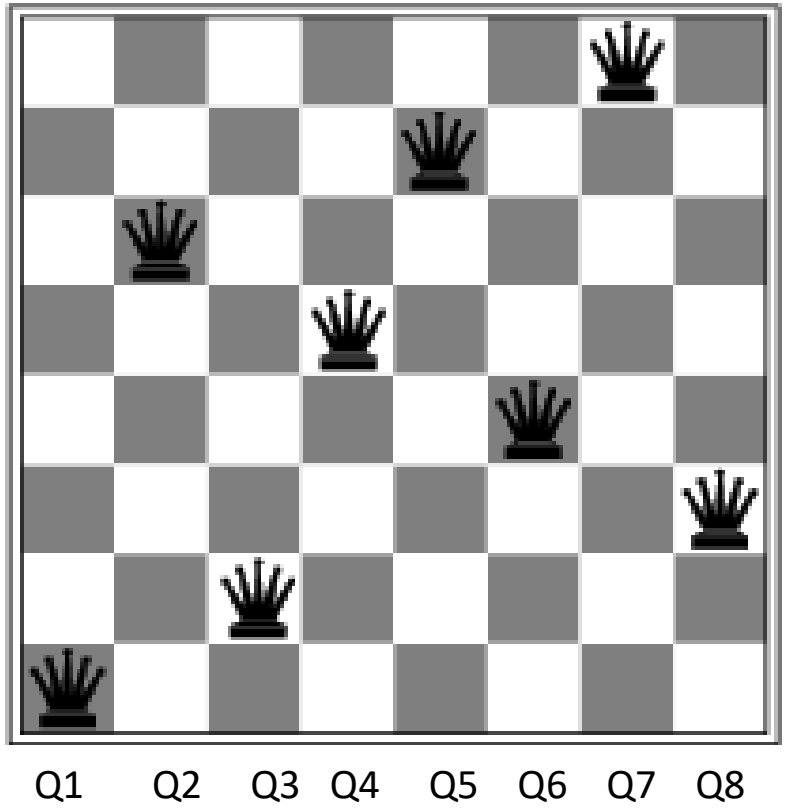
stop in global optimum (solution)

Hill-climbing: Stuck In Local Optimum



6 step

Step 1: h=-17 → h=-12
Step 2: h=-12 → h=-7
Step 3: h=-7 → h=-4
Step 4: h=-4 → h=-3
Step 5: h=-3 → h=-1
Step 6: h=-1 stop



Hill-climbing for 8-Queen Problem

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

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
neighbor ← a highest-valued successor of *current*
if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
current ← *neighbor*

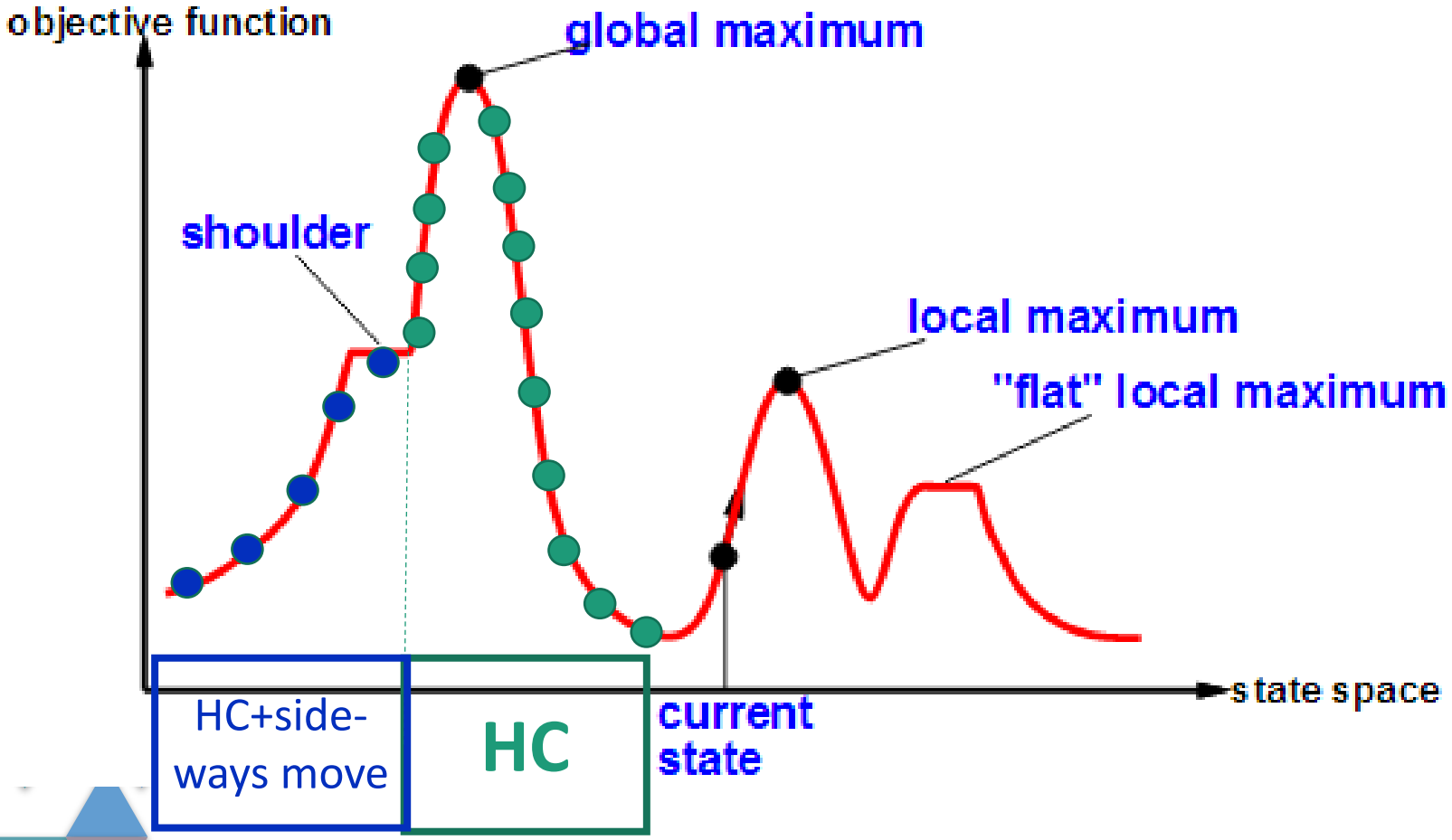
State space:
$8^8 \approx 16.8$ million states

Average case: works quickly
when **success**: avg 4 steps
when **stuck**: avg 3 steps

Best case:
initial state = goal state
Prob: $92 / 8^8 = 0.00054\%$

Get **stuck** 86%,
solving only 14% of
problem instances

Hill-climbing Search: Final State



Success: global maximum

Stuck: 1) local maximum, 2) flat local maximum, 3) shoulder

Shoulder: still possible to global max.
Variant HC: + **sideways move** with limit on number of consecutive ways

Variant 1: Hill-climbing with Sideways Move

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

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
neighbor ← a highest-valued successor of *current*
if neighbor.VALUE < current.VALUE **then return** *current*.STATE
current ← *neighbor*

Terminates when it reaches a "peak" ~~including "flat"~~

Increase success for 8-queens problem.
Limit=100:
14% ➔ 94% success

Works **slower**:
when **success**: avg 4 ➔ 21 steps
when **stuck**: avg 3 ➔ 64 steps

Variant 2: Random Restart Hill-climbing

If at first you don't succeed, try, try again. It conducts **a series of hill climbing** searches (random initial states, until a goal is found)

Expected nb of restarts = `1/p` → p=0.14: 7 restart
Expected nb of steps = `s+f(1-p)/p` → p=0.14, s=4, f=3: 22 steps.

Very effective for n-queens problem: solving 3 million queens in under a minute (Luby et al., 1993)

Variant 3: Stochastic Hill-climbing

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

current ← MAKE-NODE(*problem*.INITIAL-STATE)
repeat nmax times
  neighbor ← a **random** successor of *current*
  if neighbor.VALUE **>** current.VALUE **then** *current* ← *neighbor*

Terminates when it reaches **nmax iteration**

Generating a successor randomly (not all successor)

Move to neighbor if it is better than current state.

Works **slower** (more steps)

Summary: Hill-climbing

continually moves in the direction of increasing value (objective) or decreasing value (cost)

Depending on initial state, can get stuck in local maxima.

Variant: steepest ascent HC, HC with sideways move, stochastic HC, random restart HC

Next:

- Simulated annealing

THANK YOU