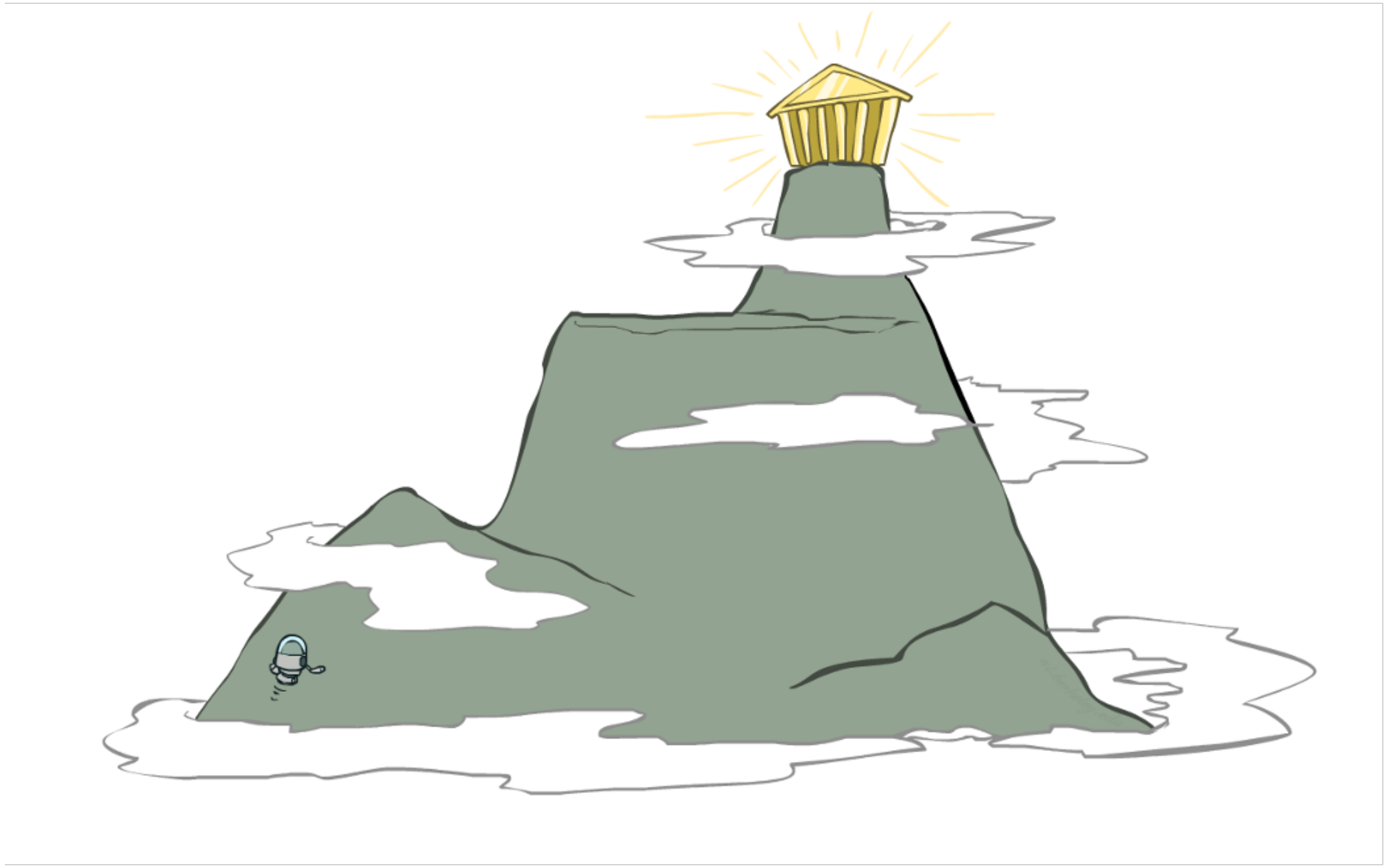
COMPSCI 761: ADVANCED TOPICS IN ARTIFICIAL INTELLIGENCE

LOCAL SEARCH

Anna Trofimova, August 2022

TODAY

- CSP Definition recap
- CSP Backtracking
- CSP Inference



SYSTEMATIC VS LOCAL SEARCH

- Systematic search strategies
  - Frontier maintains all unexpanded successors of expanded nodes.
  - Traverse the search space of a problem instance in a systematic manner.
  - Guarantee completeness, i.e., that eventually either a solution is found, or determine that the solution does not exist.

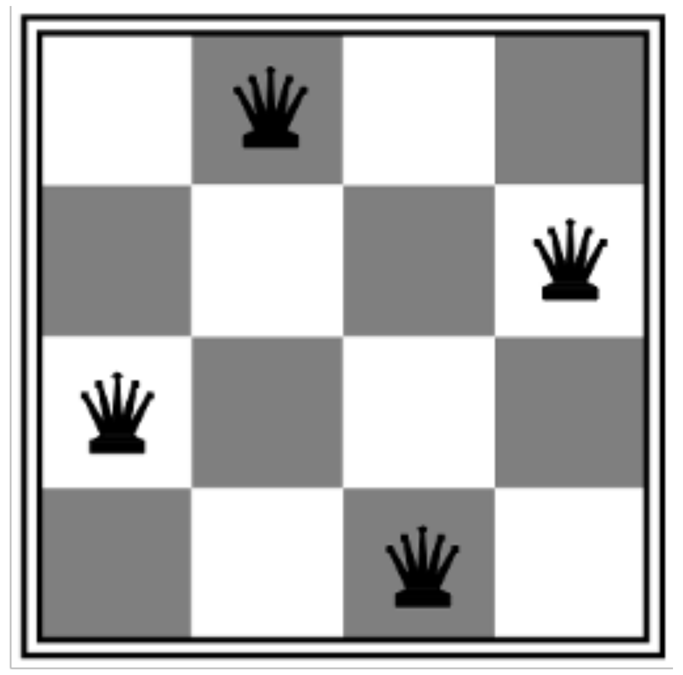
- Local search strategies (also called Iterative Improvement):
  - Frontier maintains some unexpanded successors of expanded nodes.
  - Start at some state and “move” from present location(s) to neighbouring location(s). The moves are determined by the present location(s).
  - Do not guaranteed completeness.

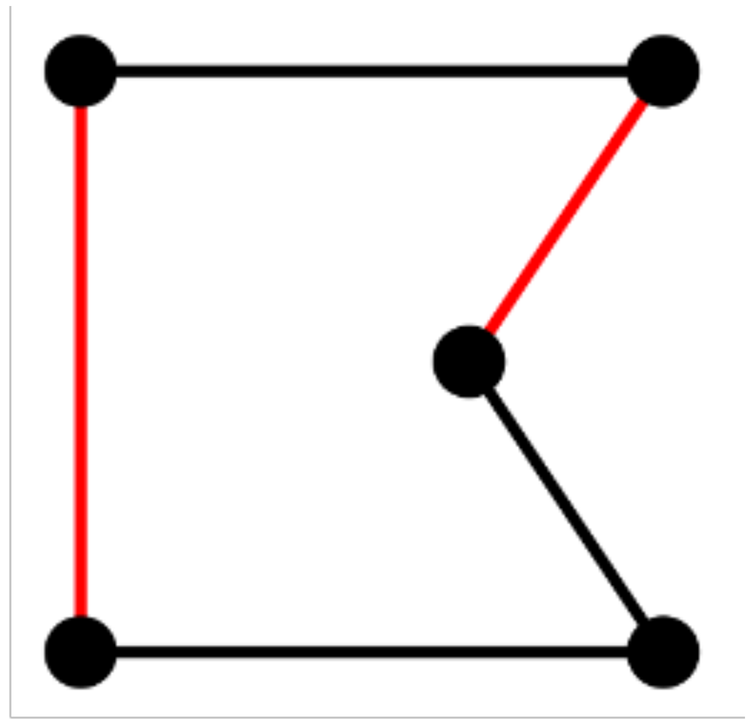
WHAT DOES SEARCH RETURNS?

- In Systematic Search a solution can be a path or a state!
- In Local Search a state is a solution!
  - Since it doesn’t maintain a path (“only a current state”) it can’t return a path
  - Can use a goal test or partial goal state
    - (but not a complete goal state – why?)

LOCAL SEARCH ALGORITHMS

- In many optimisation problems, ***path*** is irrelevant; the goal state ***is*** the solution
- Then state space = set of "complete" configurations;
  - find ***configuration satisfying constraints***, e.g., n-queens problem; or, find ***optimal configuration***, e.g., travelling salesperson problem





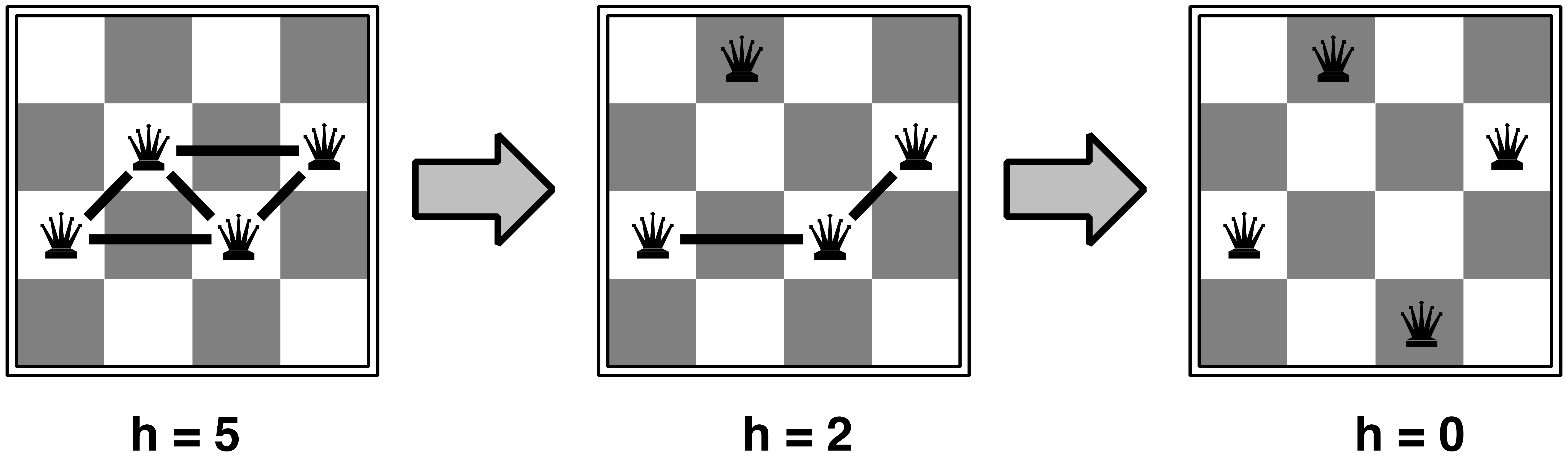
- In such cases, can use ***iterative improvement*** algorithms: keep a single "current" state, try to improve it
- Constant space, suitable for online as well as offline search

A GENERIC LOCAL SEARCH ALGORITHM

```
The Generic LocalSearch Procedure
INPUT: A search problem with features X1, . . . , Xk
OUTPUT: A solution (x1, . . . , xk)
   Choose (x1, . . . , xk) using initialisation function
 while stopping criterion is not met do
    Select a successor (y1, . . . , yk)
    Update (x1, . . . , xk) to (y1, . . . , yk)
   end while
   return (x1, . . . , xk)
```

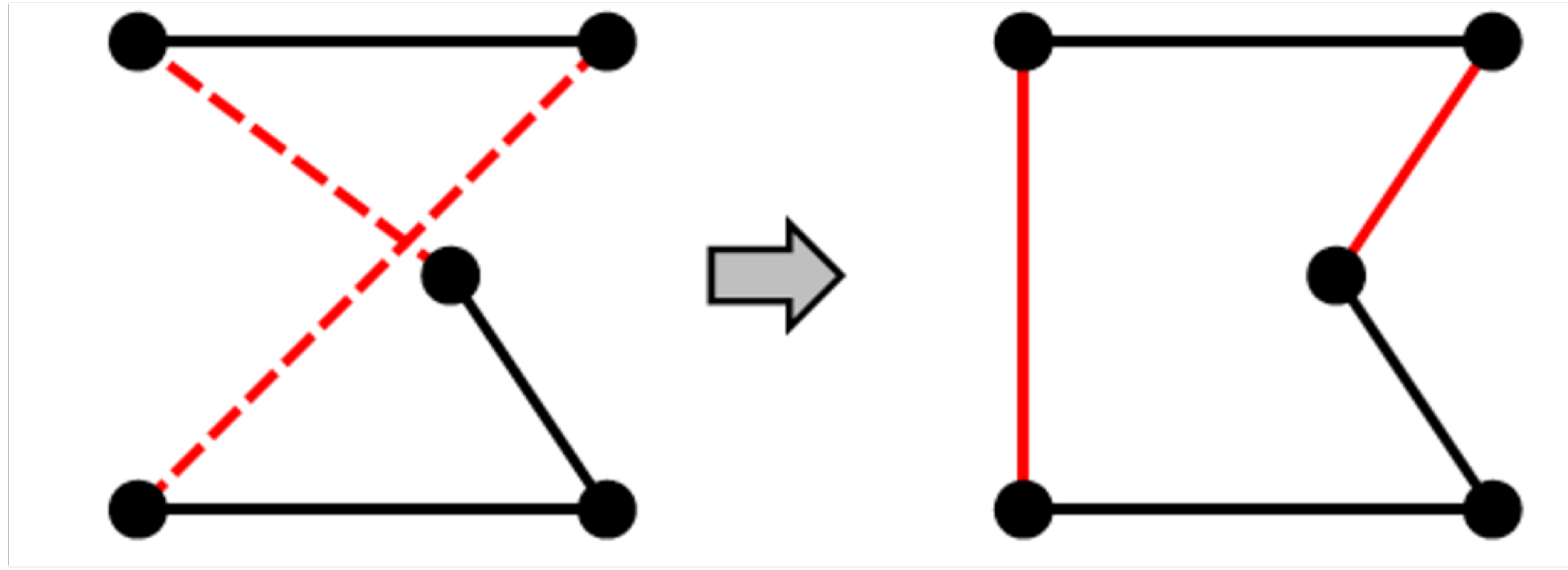
EXAMPLE: N-QUEENS

- Iterative Improvement
  - assign all variables randomly in the beginning (thus violating several constraints),
  - change one variable at a time, trying to reduce the number of violations at each step.
  - Greedy Search with h = number of constraints violated



TRAVELING SALESPERSON PROBLEM

Start with any complete tour, perform pairwise exchanges



Variants of this approach get within 1% of optimal very quickly with thousands of cities

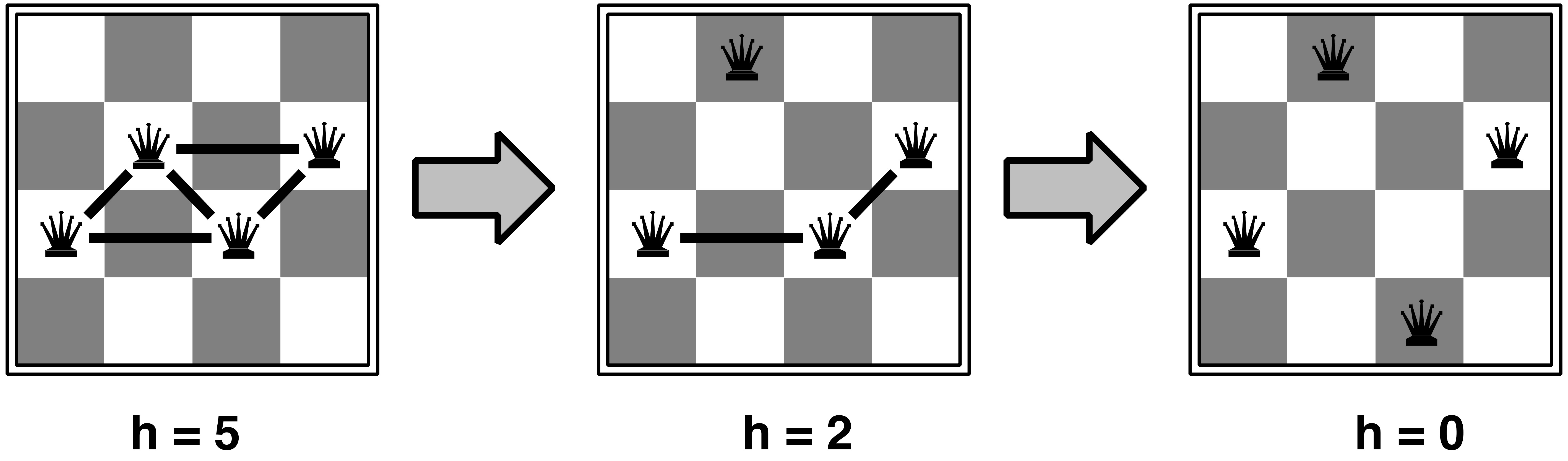
.

STOPPING CRITERIA

- Heuristic value – You can use the heuristic value as a termination test in some problems. (e.g., n-queens)
- Goal test – You can use a goal test (like in Systematic Search). This will work in Sudoku or in N-queens.
    - You cannot have a “complete goal state” because then the problem will already be solved
- Number of runs – Sometimes you just need to specify when to stop. For instance, in the Traveling Salesperson Problem.

LOCAL SEARCH HEURISTICS

- A local search heuristic, $h: S \rightarrow R$.
- This allows the algorithm to make moves maximising or minimising the value.
- It does not have to be admissible or consistent.
- It can be involved in the goal test or termination condition, or not.



ITERATIVE BEST IMPROVEMENT

Iterative best improvement (IBI) is a local search strategy that always selects a successor that minimises (or maximises) a heuristic function which is typically the loss (or gain), i.e., it performs greedy choice.

Successor A

A′=argmin h(A′) or A′=argmax h(A′)

where $h: S \rightarrow R$ is a heuristic function

Characteristic:

Tie break is typically random.

.

HILL CLIMBING OR GREEDY DESCENT

For minimisation the method is called greedy descent (similar to gradient descent – we will see later)

.

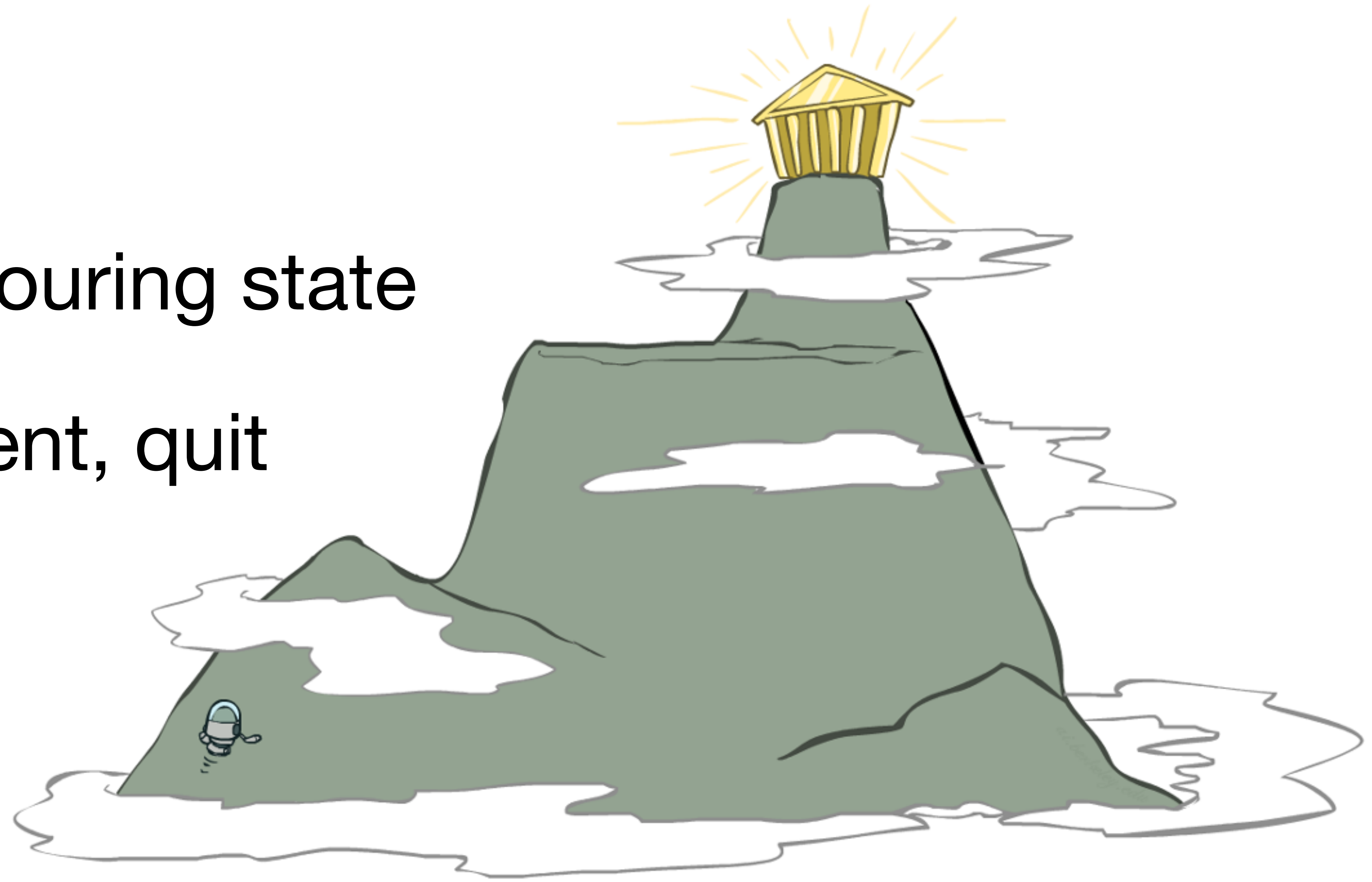
For maximisation the method is called hill-climbing

HILL-CLIMBING

"Like climbing Everest in thick fog with amnesia"

- Simple, general idea:
  - Start wherever
  - Repeat: move to the best neighbouring state
  - If no neighbours better than current, quit

.



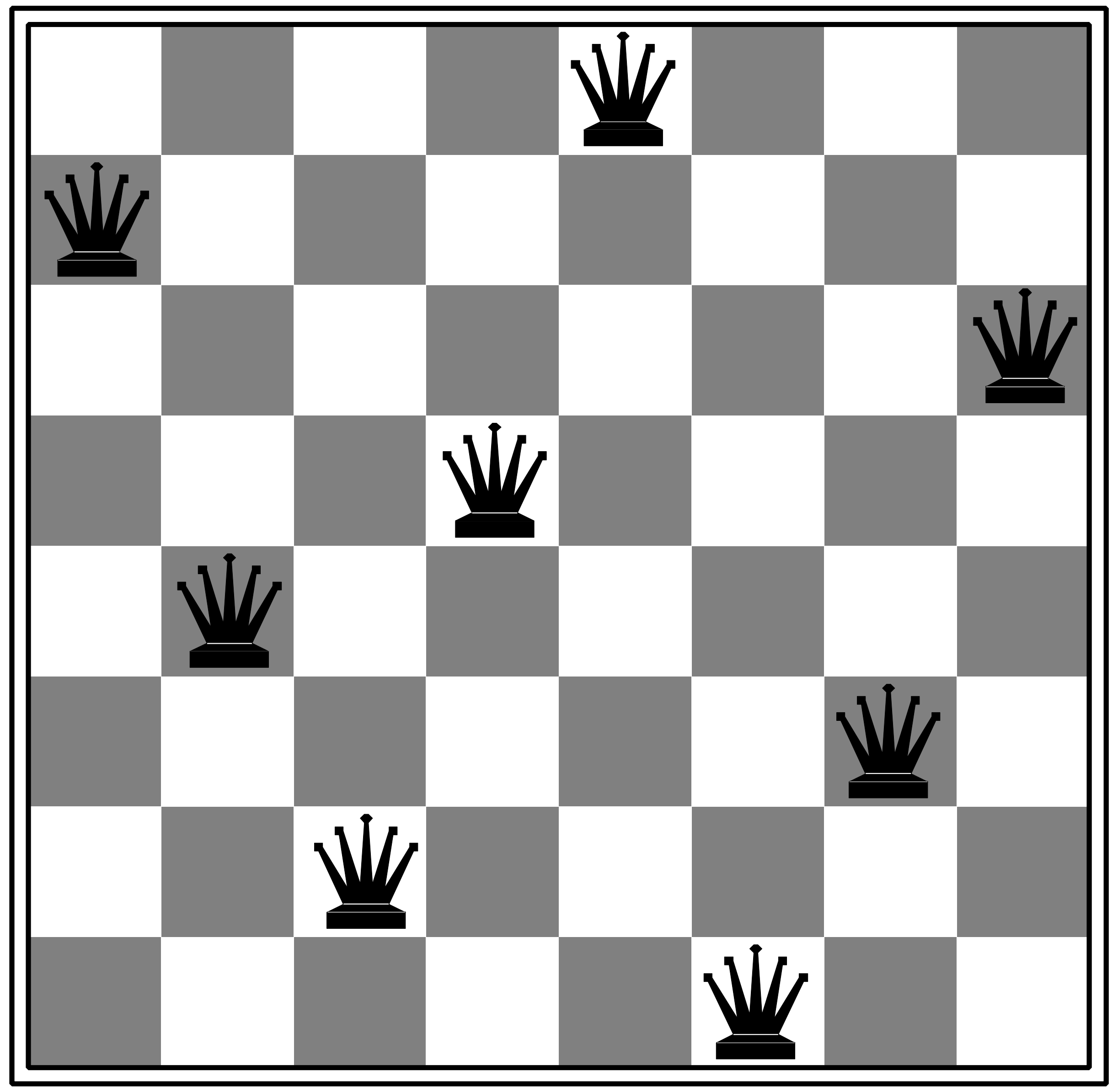
HILL-CLIMBING SEARCH

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
    inputs: 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
```

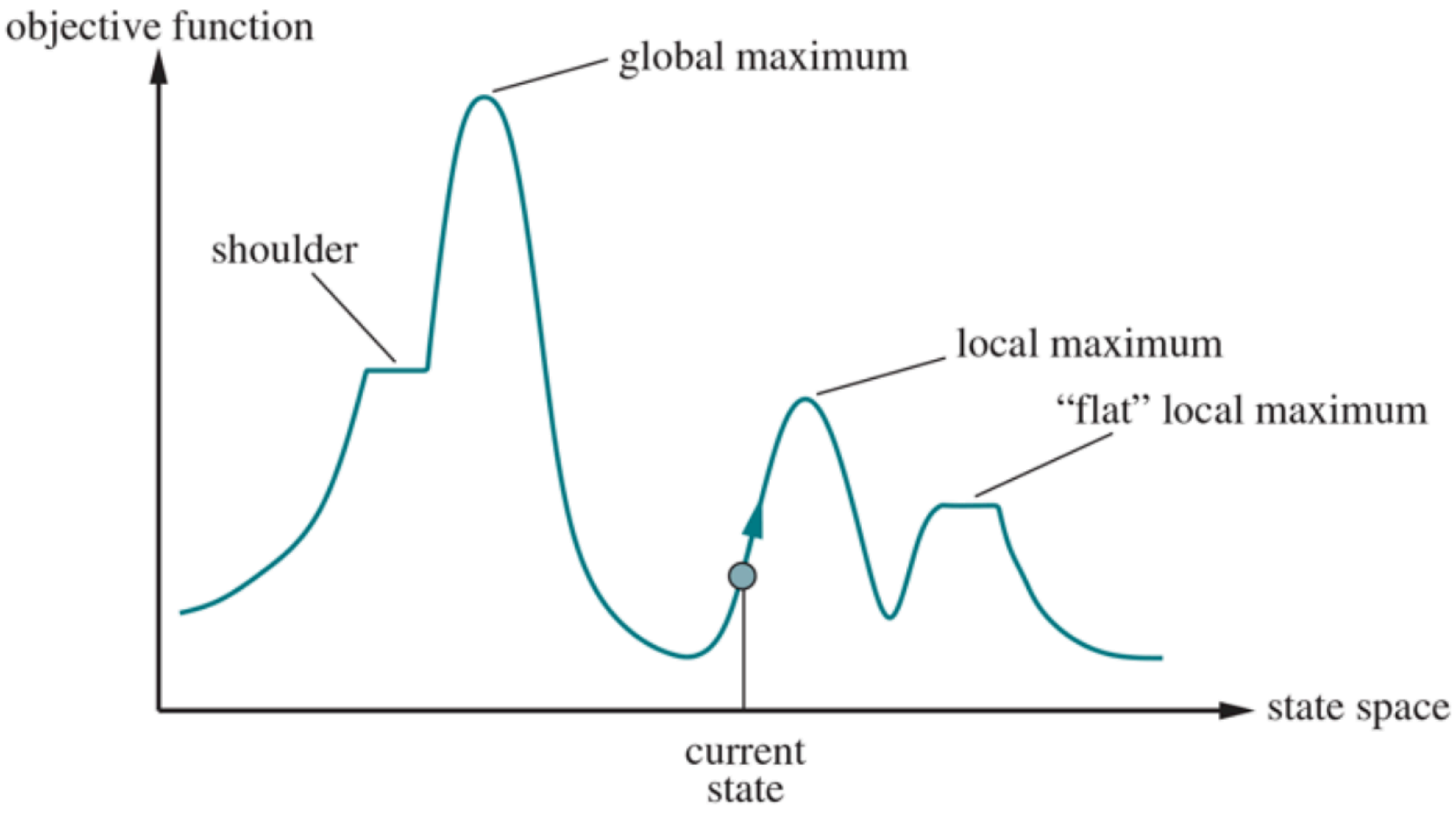
HILL-CLIMBING WITH 8-QUEENS

- Randomly generated 8-queens starting states…
    - Gets stuck 86% of the time
    - Solves only 14% of problem instances
- However
    - 4 steps on average when it succeeds
    - 3 steps on average when it gets stuck
    - (for a state space with 8^8 =~17 million states)



HILL-CLIMBING DIFFICULTIES

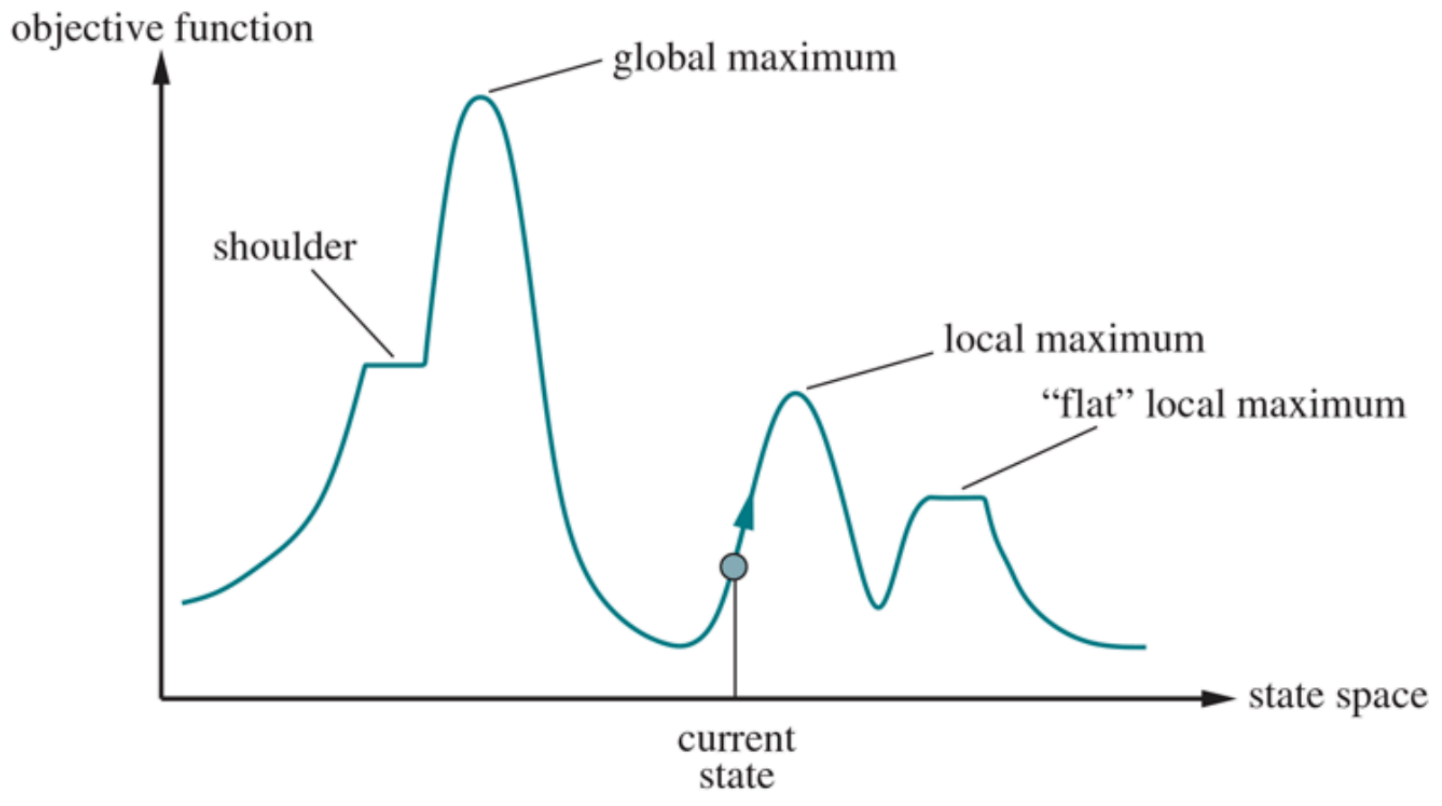
Problem: depending on initial state, can get stuck in local maxima

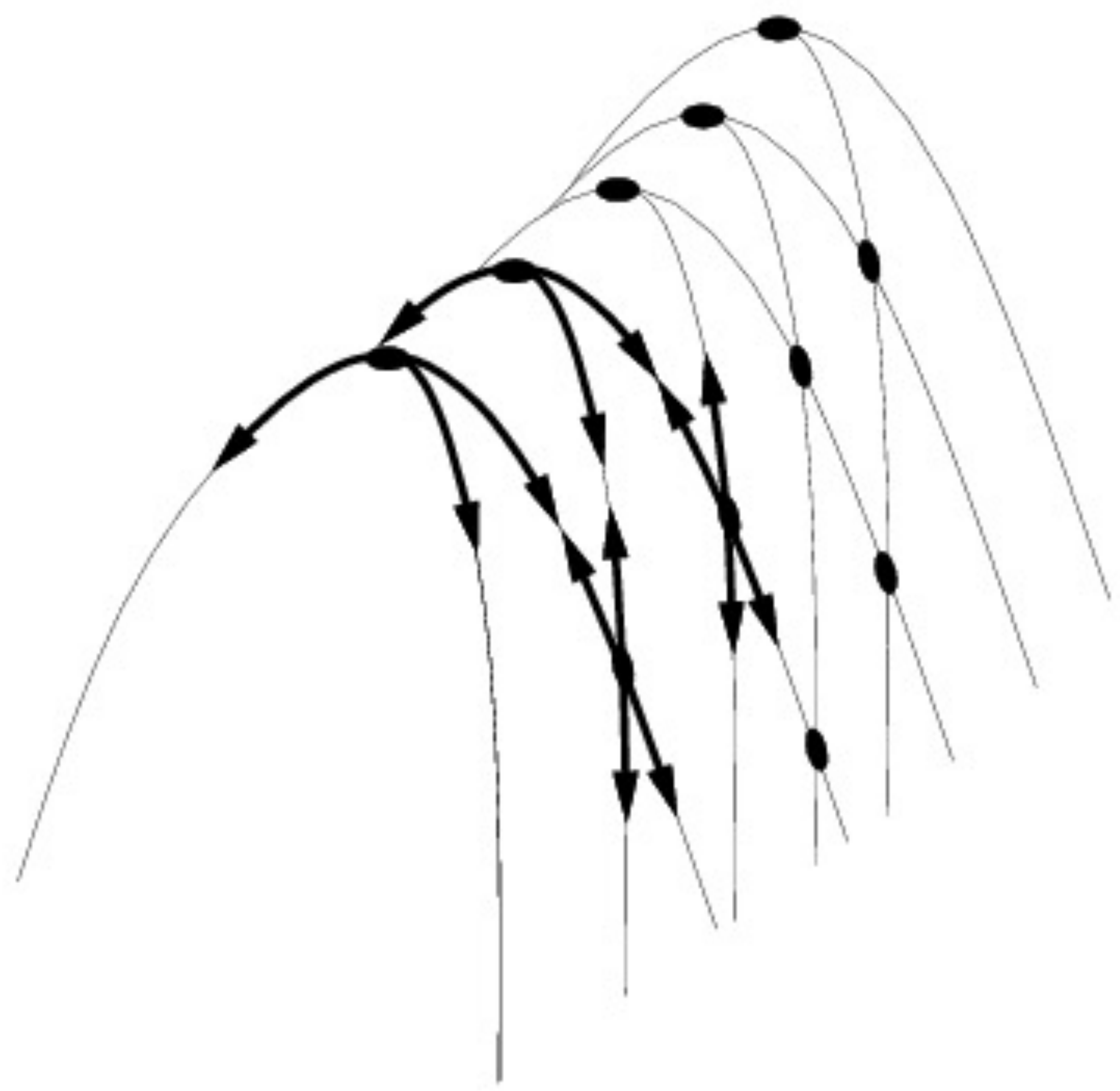


Note: these difficulties apply to all local search algorithms, and usually become much worse as the search space becomes higher dimensional

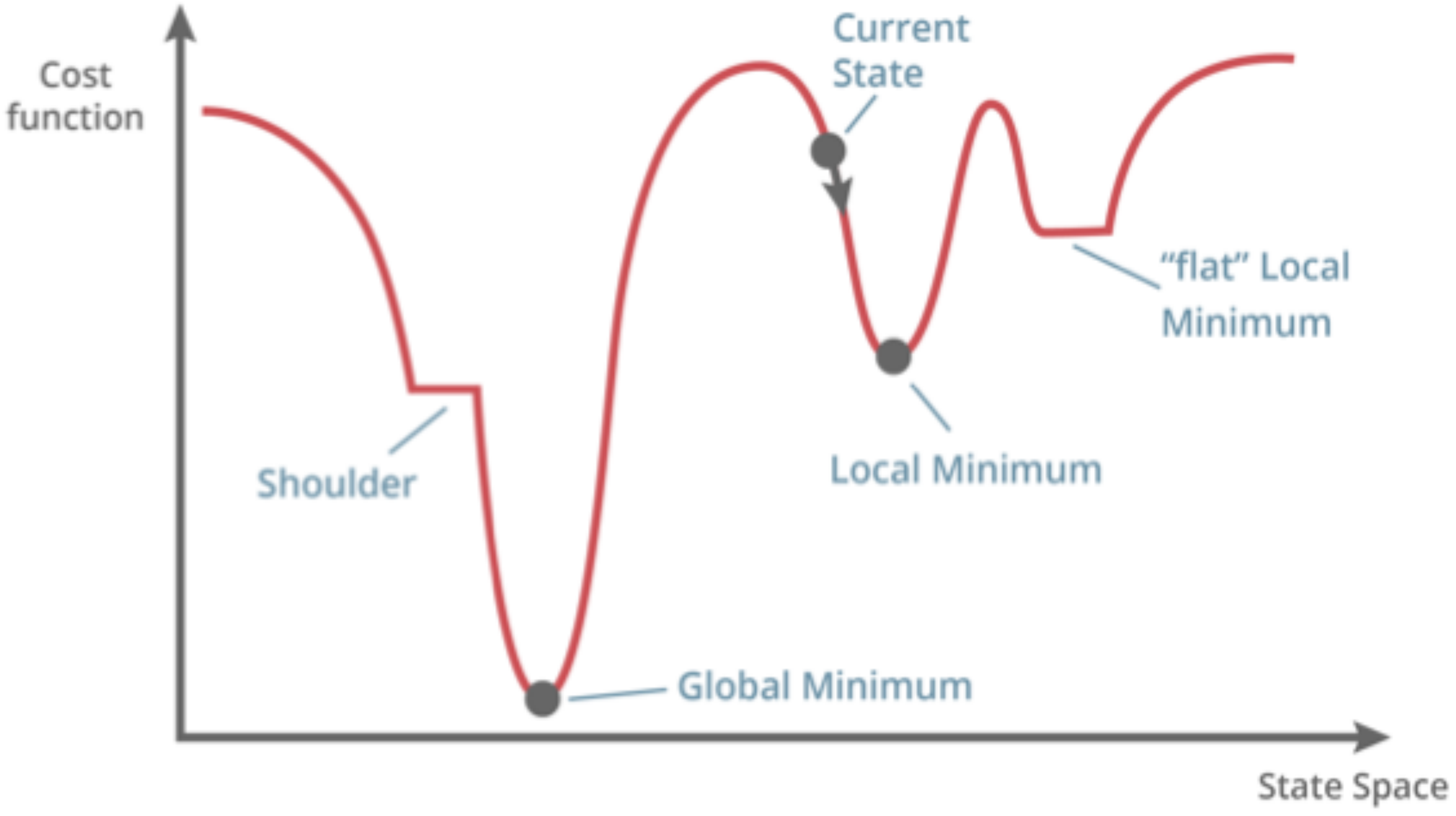
HILL-CLIMBING PROBLEMS

- Local maxima
- Plateaus
- Diagonal ridges





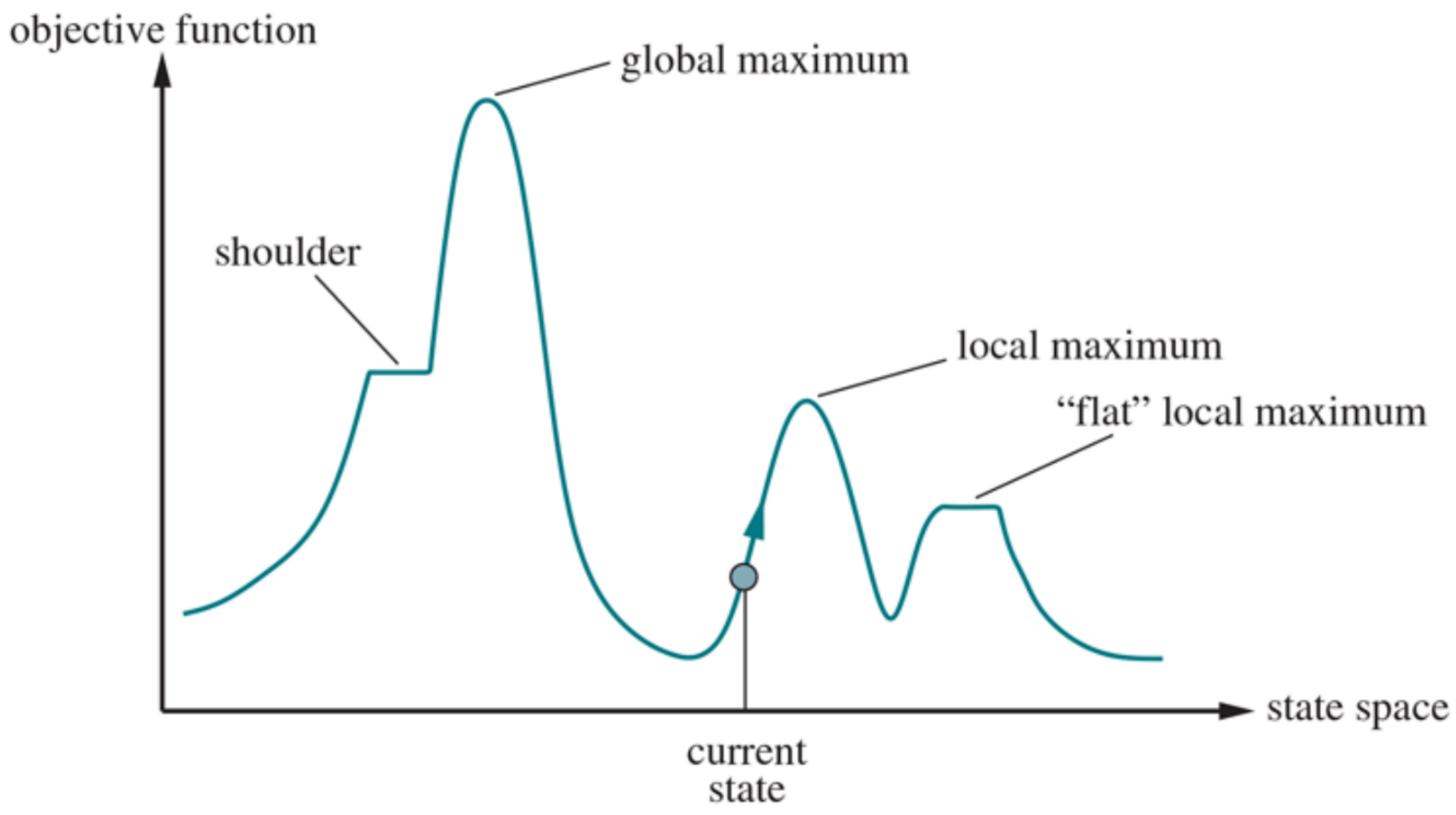
INVERTED VIEW FOR GREEDY DESCENT



- Sometimes, have to go sideways or even backwards in order to make progress towards the actual solution.

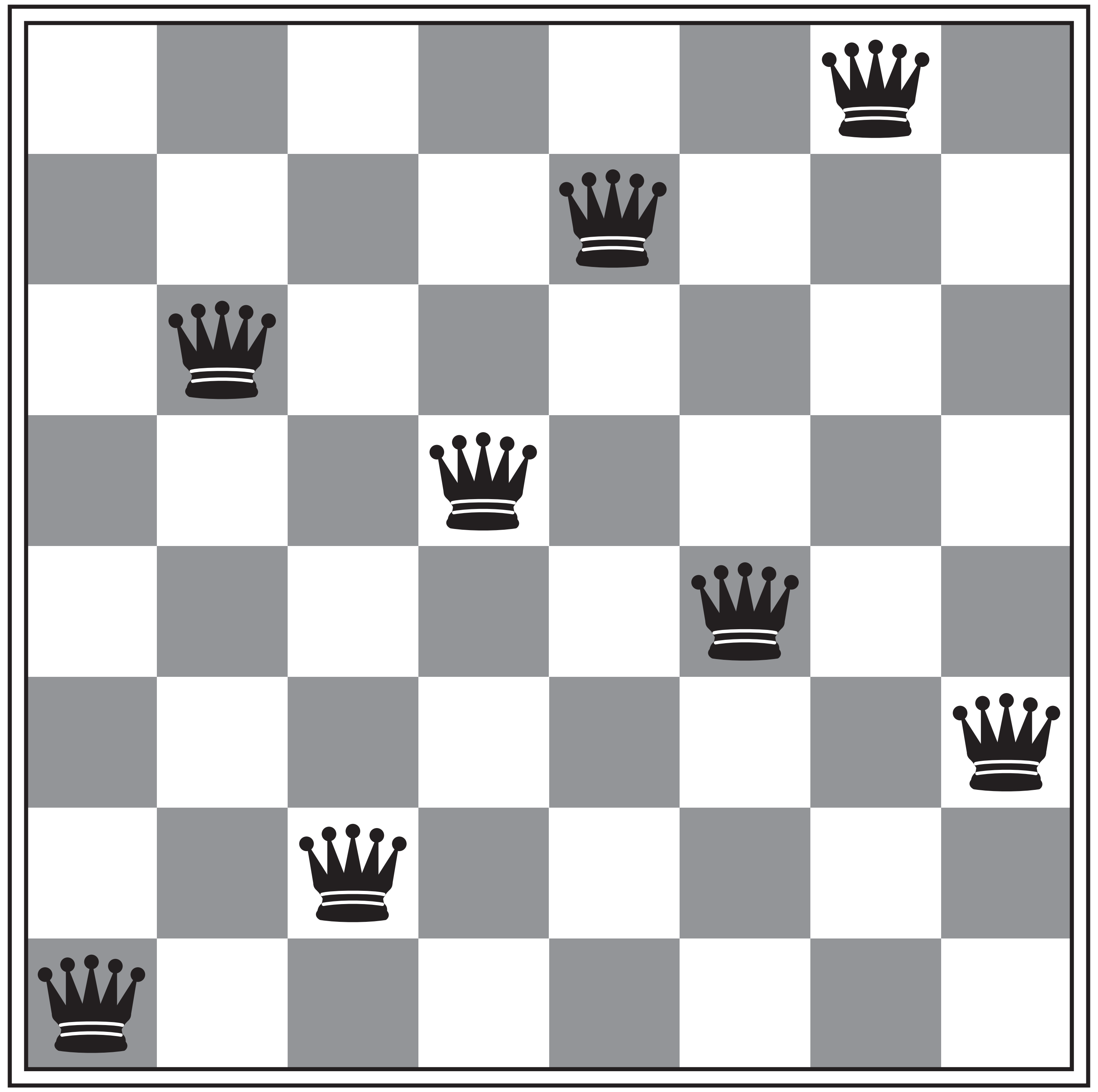
HILL-CLIMBING WITH SIDE-WAY MOVES

- Hill-Climbing with sideways moves uses the same procedures as hill-climbing with the following differences:
    - When no uphill successors (ones that increase loss strictly) are present, move “sideways” (successors that have the same value).
    - Introduce a parameter *m* such that only ≤ *m* such sideways moves can be performed consecutively.
- Loop on at maxima
    - Limit number of consecutive sideways steps
- Escape from shoulders



HILL-CLIMBING WITH SIDE WAY MOVES ON THE 8-QUEENS PROBLEM

- No sideways moves:
  - Succeeds w/ prob. p = 0.14
  - Average number of moves per trial:
    - 4 when succeeding, 3 when getting stuck
  - Expected total number of moves needed:
    - 3(1-p)/p + 4 =~ 22 moves



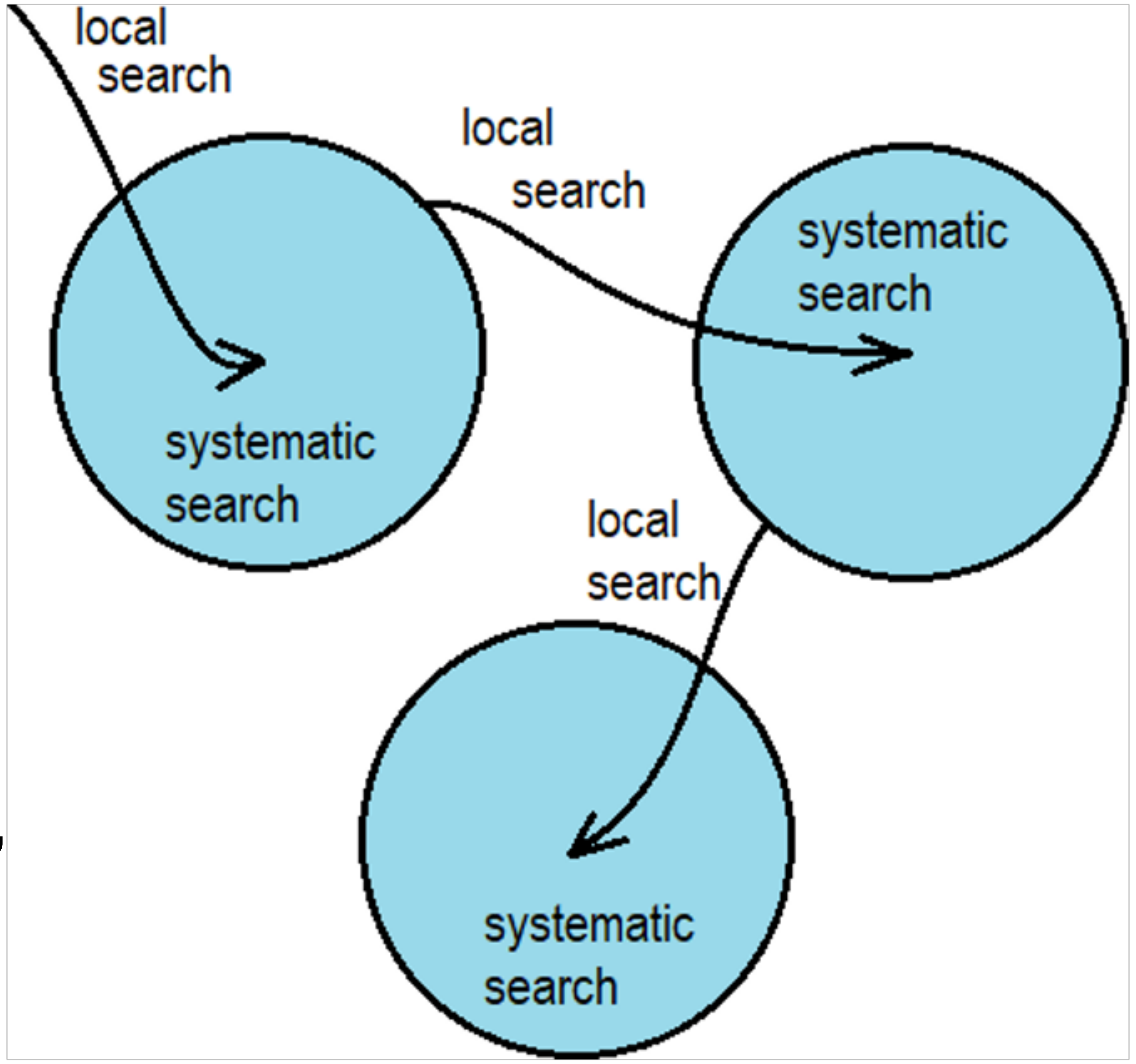
- Allowing 100 sideways moves:
  - Succeeds w/ prob. p = 0.94
  - Average number of moves per trial:
    - 21 when succeeding, 65 when getting stuck
  - Expected total number of moves needed:
    - 65(1-p)/p + 21 =~ 25 moves

ENFORCED HILL-CLIMBING

- Perform breadth first search from a local optima
    - to find the next state with better h function
    - good At Escaping Shoulders/local Optima

- Typically,
    - prolonged periods of exhaustive search
    - bridged by relatively quick periods of hill-climbing
    - Sometimes called “Variable Neighbourhood Descent”

- Middle ground b/w local and systematic search



.

TABU SEARCH

- Prevent returning quickly to the same state
  - Keep fixed length queue (“tabu list”)
  - Add most recent state to queue; drop oldest
  - Never make the step that is currently tabu’ed
- Properties:
  - As the size of the tabu list grows, hill-climbing will asymptotically become “non-redundant” (won’t look at the same state twice)
  - In practice, a reasonable sized tabu list (say 100 or so) improves the performance of hill climbing in many problems
  - The list can allow the algorithm to escape some of the local minima

SUMMARY

Hill Climbing – chose best child

Hill Climbing with Sideways Moves – chose best child or equal child

Tabu Search – keep list of nodes you have been to and don't go back

Enforced Hill Climbing – use breadth first search until you find a "better" node