Informed search algorithms

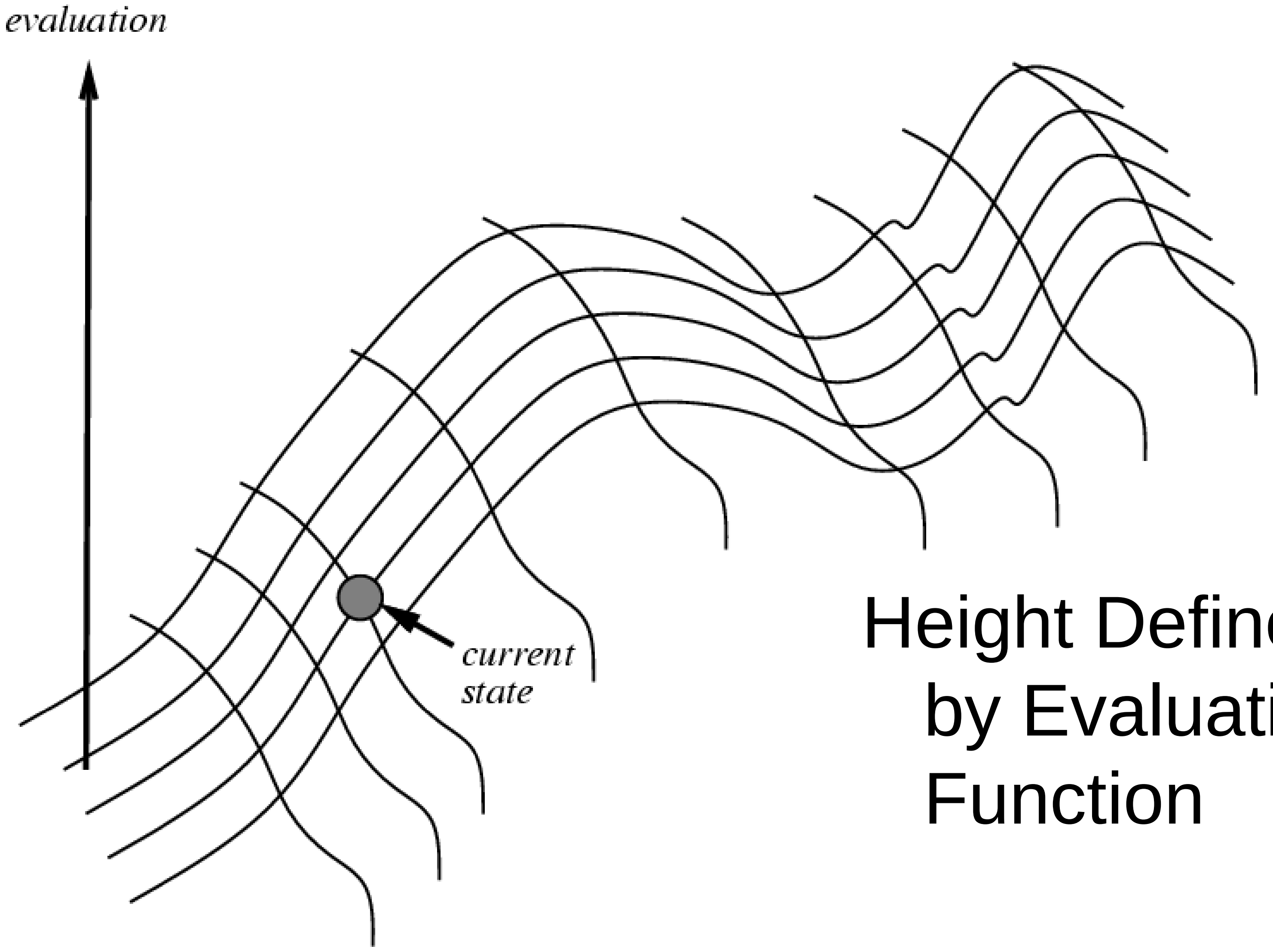
Local search algorithms

- In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution
  - State space = set of "complete" configurations
  - Find configuration satisfying constraints, e.g., n-queens
- In such cases, we can use local search algorithms
  - keep a single "current" state, tries to improve it

Iterative Improvement Search

- Another approach to search involves starting with an initial guess at a solution and gradually improving it until it is a legal/optimal one.
- Some examples:
  - Hill climbing
  - Simulated annealing
  - Constraint satisfaction

Hill Climbing on a Surface of States



Height Defined by Evaluation Function

Hill Climbing Search

- If there exists a successor s for the current state n such that
  - h(s) < h(n)
  - h(s) ≤ h(t) for all the successors t of n,

  then move from n to s. Otherwise, halt at n.
- Looks one step ahead to determine if any successor is better than the current state; if there is, move to the best successor.
- Similar to Greedy search in that it uses h, but does not allow backtracking or jumping to an alternative path since it doesn't “remember” where it has been.
- Corresponds to Beam search with a beam width of 1 (i.e., the maximum size of the nodes list is 1).
- Not complete since the search will terminate at "local minima," "plateaus," and "ridges."

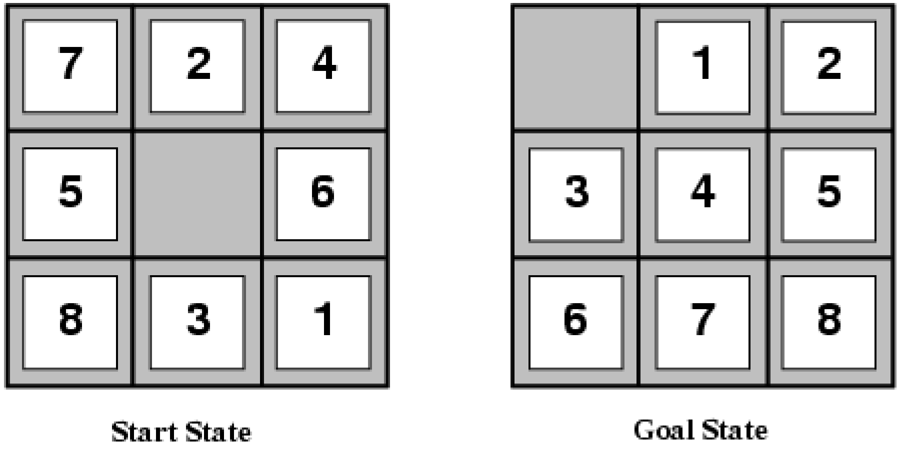
Hill-climbing search

- "Like climbing Everest in thick fog with amnesia"

```
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
```

Example: The 8-puzzle



- states? locations of tiles
- actions? move blank left, right, up, down
- goal test? = goal state (given)
- path cost? 1 per move

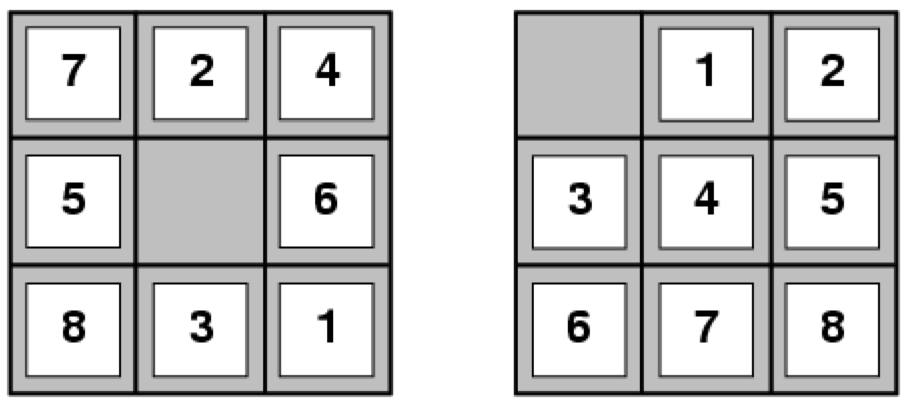
[Note: optimal solution of *n*-Puzzle family is NP-hard]

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



Start State Goal State

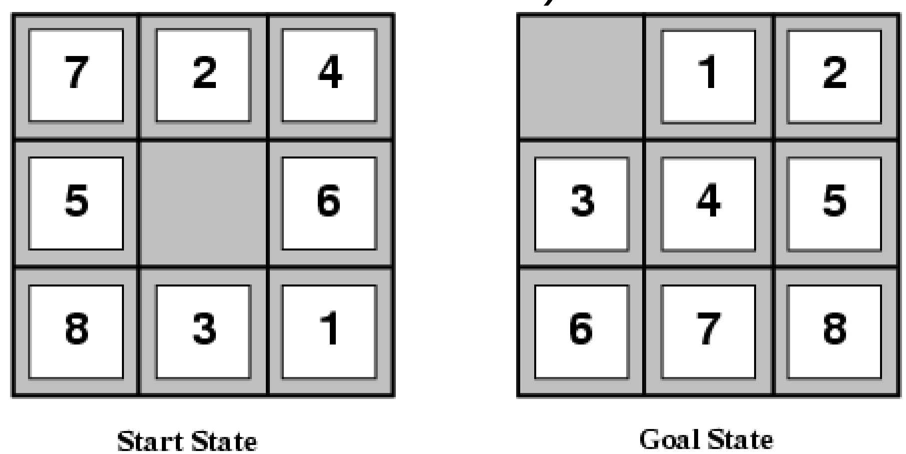
- $h_1(S) = ?$
- $h_2(S) = ?$

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



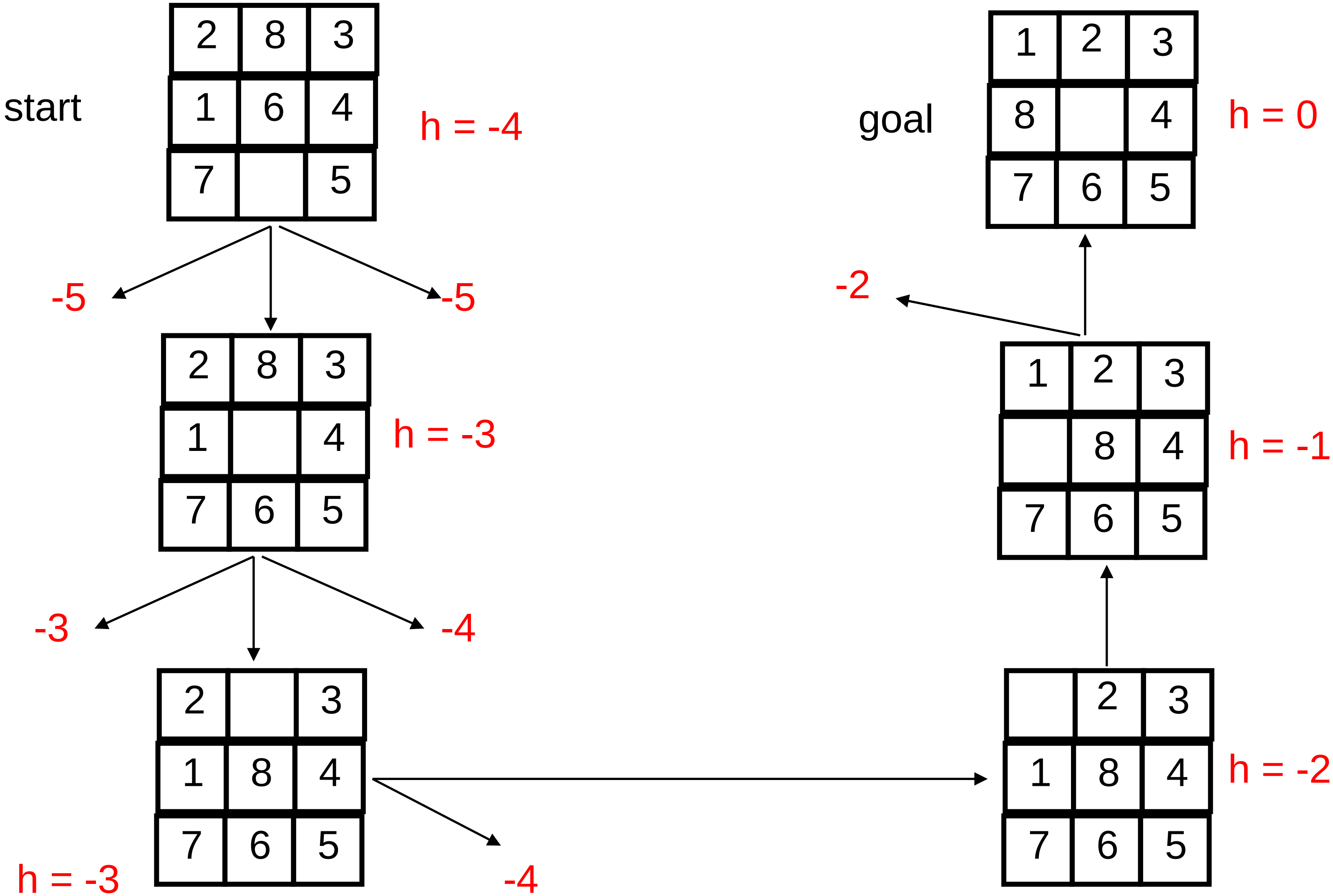
- $h_1(S) = ?$ 8
- $h_2(S) = ?$ 3+1+2+2+2+3+3+2 = 18

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible)

  then h_2 dominates h_1
- h_2 is better for search
- Typical search costs (average number of nodes expanded):
- $d=12$ IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- $d=24$ IDS = too many nodes
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

Hill Climbing Example



f(n) = -(number of tiles out of place)

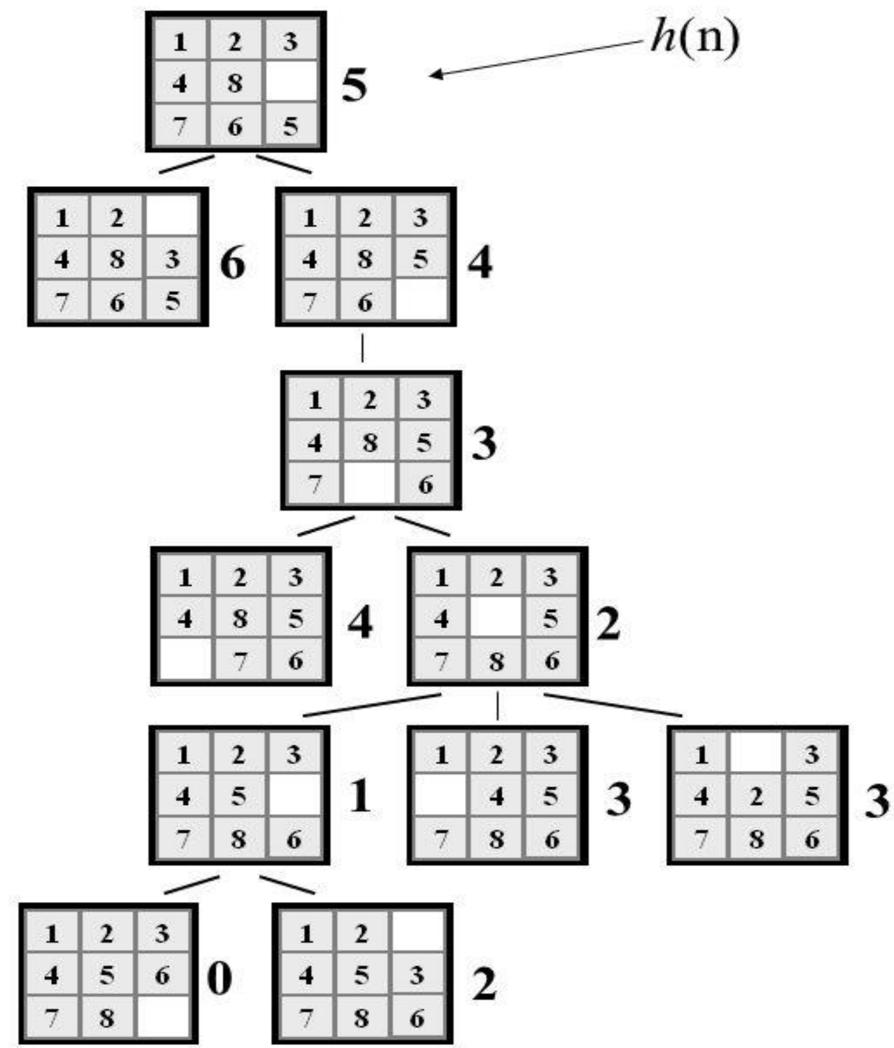
Hill Climbing Example

Hill climbing with minimization goal:
Here, the objective function is
min f Where, $f = h(n)$=(manhattan distance)

We can use heuristics to guide "hill climbing" search.

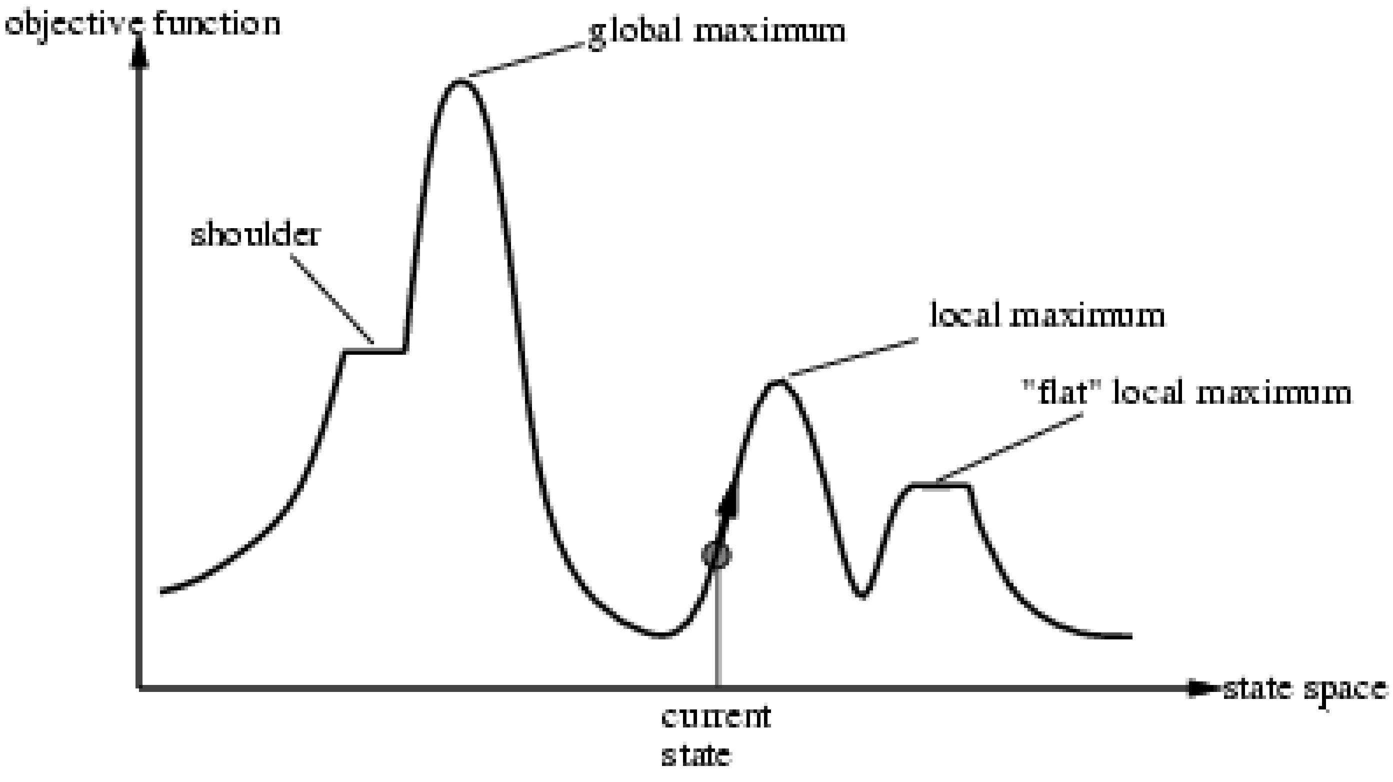
In this example, the Manhattan Distance heuristic helps us quickly find a solution to the 8-puzzle.

But "hill climbing has a problem..."



Drawbacks of Hill-climbing search

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



Exploring the Landscape

- **Local Maxima**: peaks that aren't the highest point in the space
- **Plateaus:** the space has a broad flat region that gives the search algorithm no direction (random walk)
- **Ridges:** flat like a plateau, but with drop-offs to the sides; steps to the North, East, South and West may go down, but a step to the NW may go up.

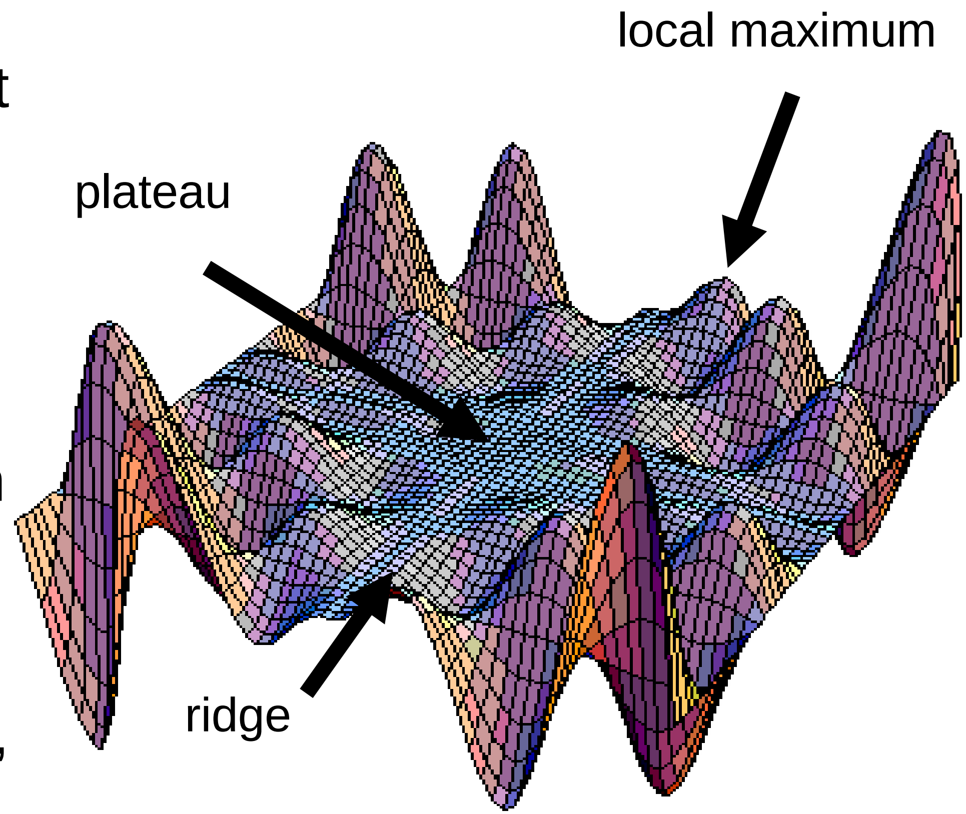
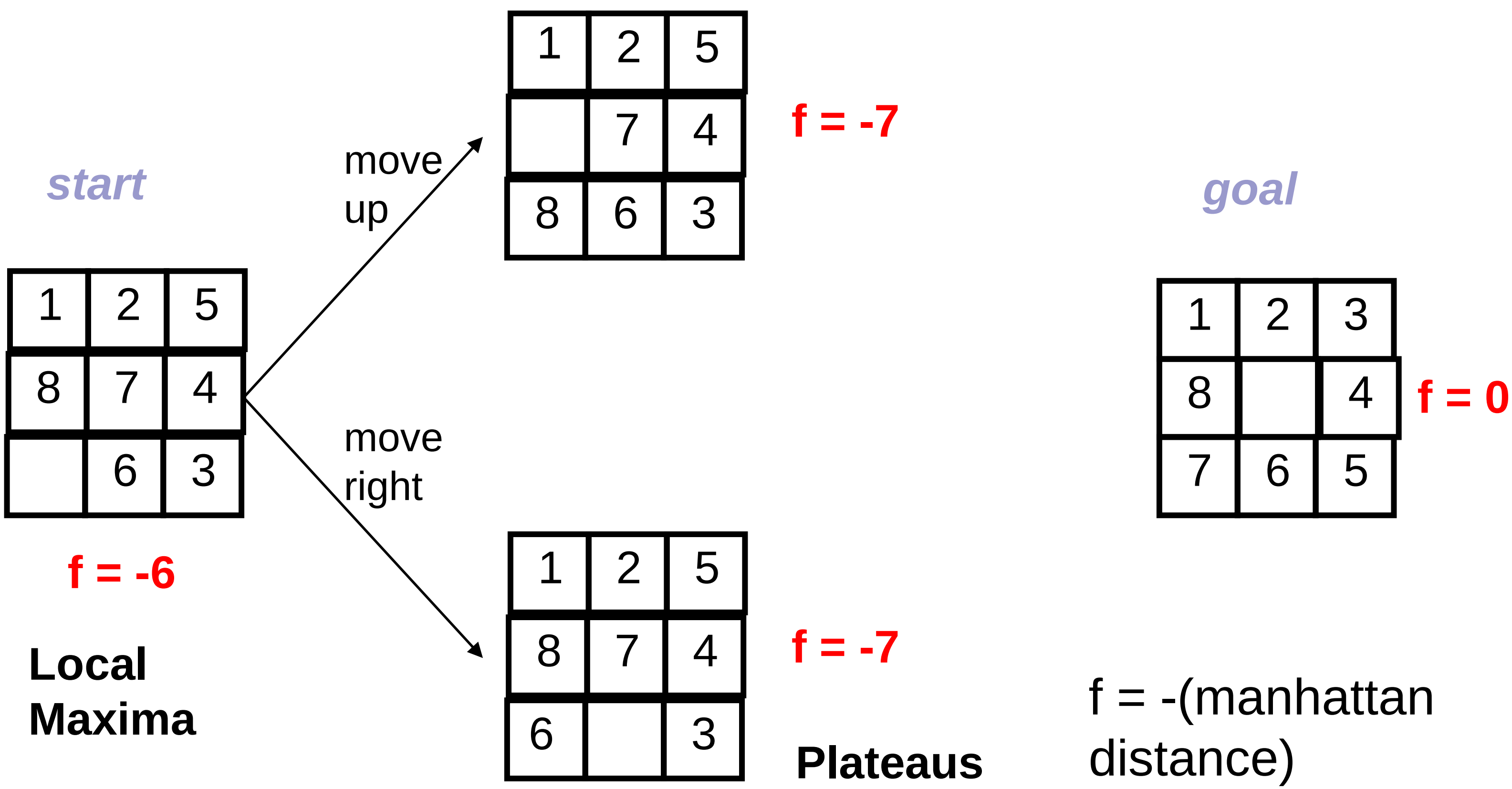


Image from:
http://classes.yale.edu/fractals/CA/GA/Fitness/Fitness.html

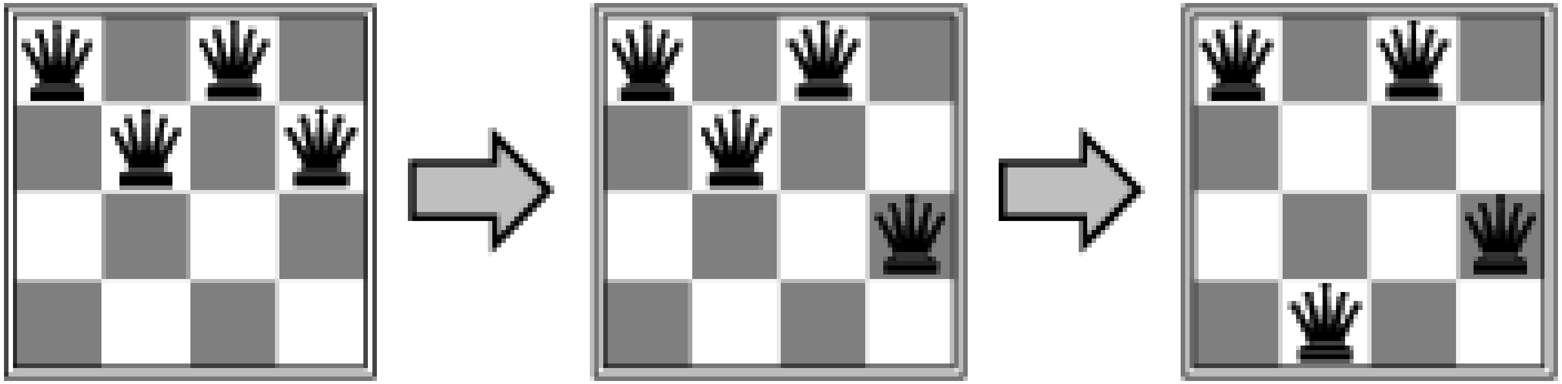
Example of a Local Optimum



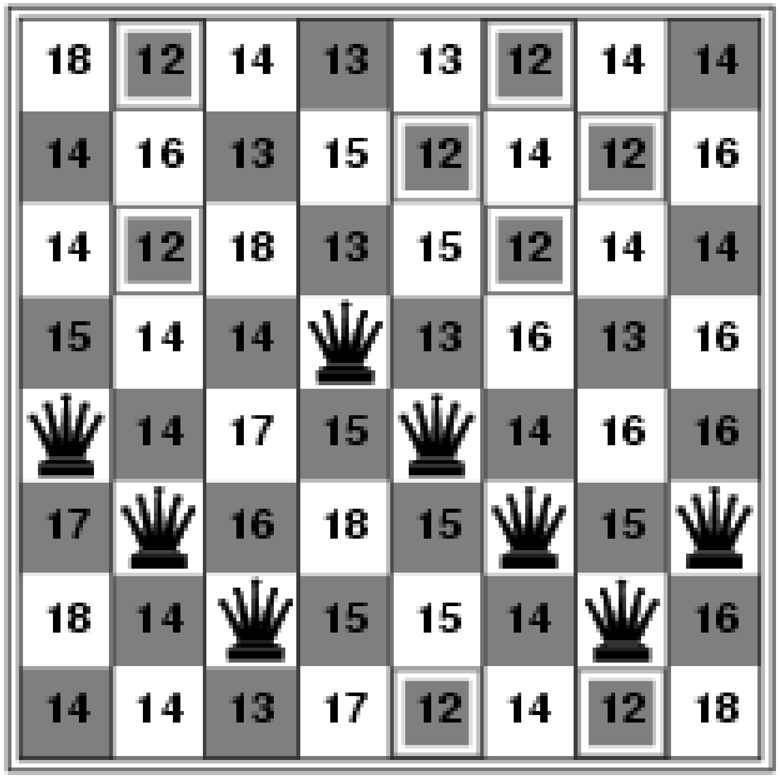
f = -(manhattan distance)

Example: *n*-queens

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

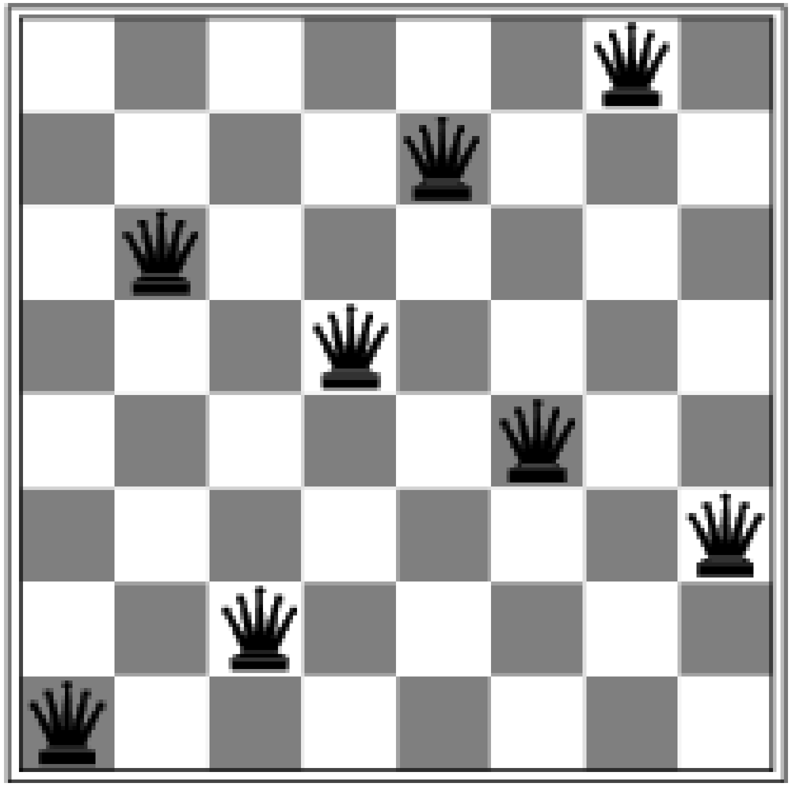


Hill-climbing search: 8-queens problem



- *h* = number of pairs of queens that are attacking each other, either directly or indirectly
- *h = 17* for the above state

Hill-climbing search: 8-queens problem



A local minimum with $h = 1$

Remedies of Hill Climbing Search

- Problems: local maxima, plateaus, ridges
- Remedies:
  - **Random restart:** keep restarting the search from random locations until a goal is found.
  - **Problem reformulation:** reformulate the search space to eliminate these problematic features
  - **Simulated Annealing**
- Some problem spaces are great for hill climbing and others are terrible.

Simulated Annealing

- Simulated annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum-energy crystalline structure (the annealing process) and the search for a minimum [or maximum] in a more general system.
- SA can avoid becoming trapped at local minima.
- SA uses a random search that accepts changes that increase objective function f, **as well as** some that **decrease** it.
- SA uses a control parameter T, which by analogy with the original application is known as the system "**temperature.**"
- T starts out high and gradually decreases toward 0.

Simulated annealing

- Idea: escape local maxima by allowing some "bad" moves but gradually decrease their frequency

1. $C = C_{init}$ // here, C is the current state and C_{init} is the initial state

2. For $T = T_{max}$ to T_{min} // here,T is the control temperature for annealing
    3. $E_C = E(C)$ // here, E_C is the Energy i.e. utility or goodness value of state C
    4. N = Next (C) // Here, N is next state of current state C
    5. $E_N = E(N)$ // here, E_N is the Energy i.e. utility or goodness value of state N
    6. $\Delta E = E_N - E_C$ //Here, ΔE is the Energy difference
    7. If ($\Delta E > 0$)
    8. C=N
    9. Else if ($e^{\Delta E / T} >$ rand(0,1)) // Suppose, $\Delta E = -1$, $T_{max} = 100$ and $T_{min} = 2$
    10. C=N // $e^{\Delta E / T} = 0.99$ for $T_{max} = 100$
11. End // $e^{\Delta E / T} = 0.60$ for $T_{min} = 2$

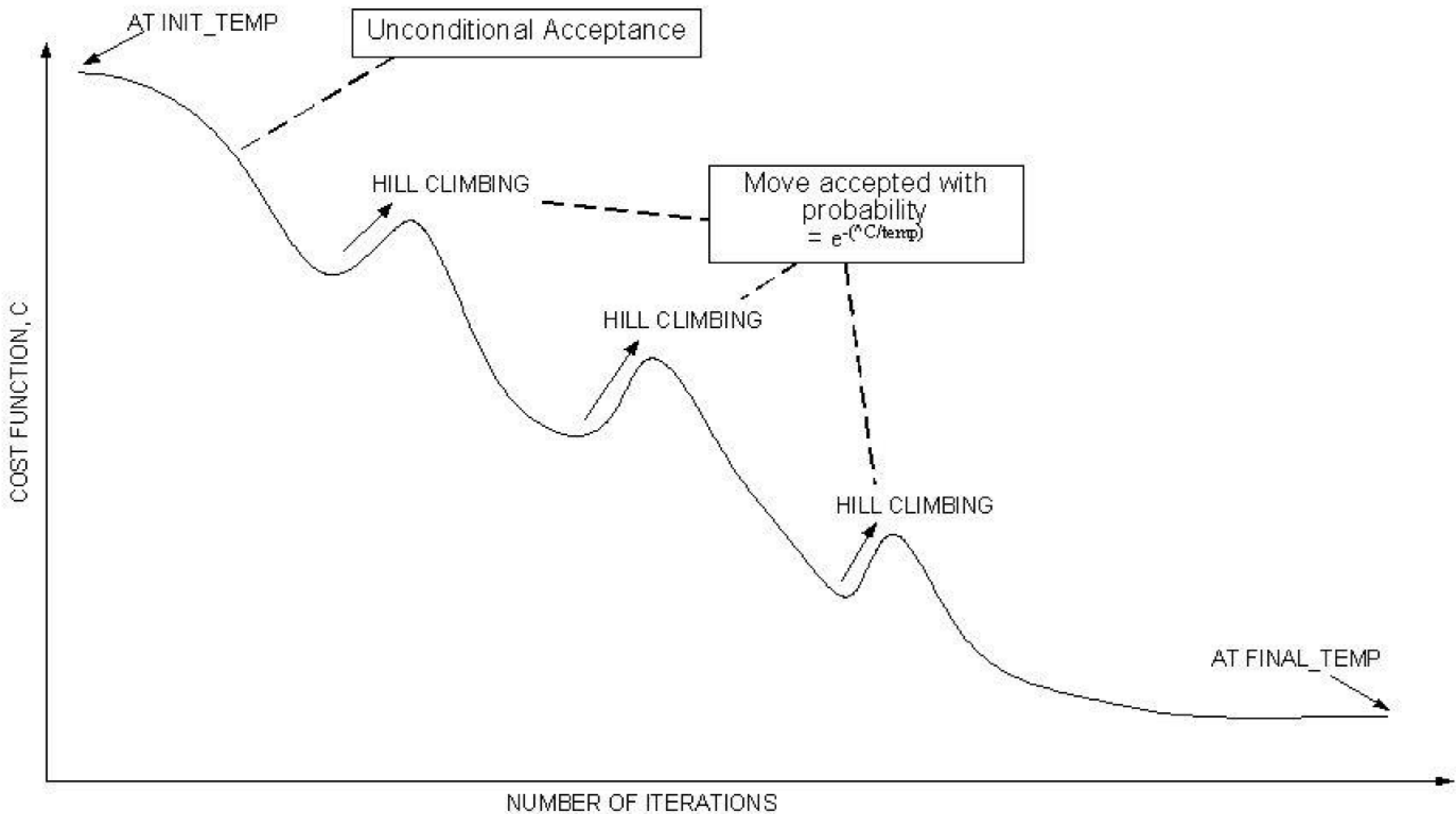
Simulated Annealing (cont.)

- f(s) represents the quality of state n (high is good)
- A "bad" move from A to B is accepted with a probability

  $P(move_{A \to B}) \approx e^{(f(B) - f(A)) / T}$

  - (Note that f(B) – f(A) will be negative, so bad moves always have a relatively probability less than one. Good moves, for which f(B) – f(A) is positive, have a relative probability greater than one.)
- The higher the temperature, the more likely it is that a bad move can be made.
- As T tends to zero, this probability tends to zero, and SA becomes more like hill climbing
- If T is lowered slowly enough, SA is complete and admissible.

Convergence of simulated annealing



Properties of simulated annealing search

- One can prove: If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

- Widely used in VLSI layout, airline scheduling, etc