Outline

- Hill-Climbing Search.
- Simulated Annealing Search.
- Local Beam Search.
- Genetic Algorithms.

Classical Search vs. Local Search

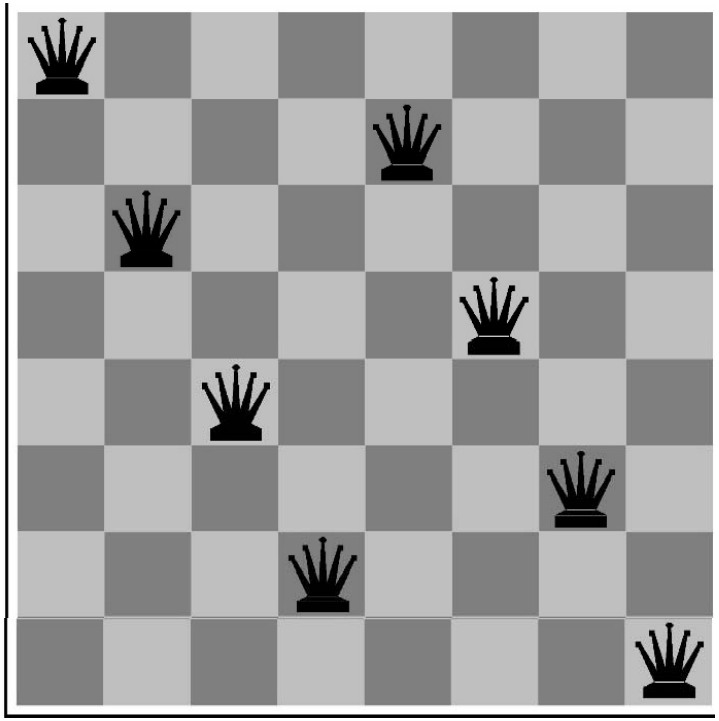
Classical search	Local Search
• systematic exploration of search space. • Keeps one or more paths in memory. • Records which alternatives have been explored at each point along the path. • The path to the goal is a solution to the problem.	• In many optimization problems, the path to the goal is irrelevant; the goal state itself is the solution. • State space = set of "complete" configurations. • Find configuration satisfying constraints, Find best state according to some objective function h(s). e.g., n-queens, h(s)= number of attacking queens. In such cases, we can use Local Search Algorithms.

Local Search Algorithms

- Local Search Algorithms keep a **single "current"** state and move to neighboring states to try to **improve** it.
- Solution path needs not to be maintained.
- Hence, the search is "local".
- Local search **suitable** for problems in which path is not important; the goal state itself is the solution.
- It is an **optimization** search

Example: n-queens

- Put n queens on an n × n board with no two queens on the same row, column, or diagonal.
- In the 8-queens problem, what matters is the final configuration of queens, not the order in which they are added.



Local Search: Key Idea

Key idea:

1. Select (**random**) initial state (generate an initial guess).
2. Make local modification to improve current state

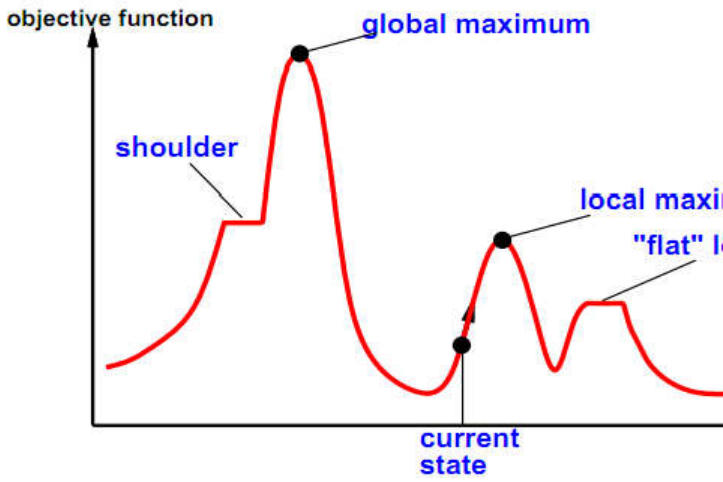
   (evaluate current state and move to other states).
3. Repeat Step 2 until goal state found (or out of time).

Local Search: Key Idea

Advantages	Drawback:
• Use very little memory – usually a constant amount. • Can often find reasonable solutions in large or infinite state spaces (e.g., continuous). For which systematic search is unsuitable.	• Local Search can get stuck in local maxima and not find the optimal solution.

State-Space Landscape

- A state space landscape: is a graph of states associated with their costs.
- State-space landscape
  - Location (defined by state)
  - Elevation (defined by the value of the heuristic cost function or objective function)
  - If elevation = cost, aim to find the lowest valley (a global minimum)
  - If elevation = objective function, find the highest peak (a global maximum)
  - A *complete local search algorithm always find a goal* if one exists
  - An **optimal** algorithm always find a **global minimum/maximum**



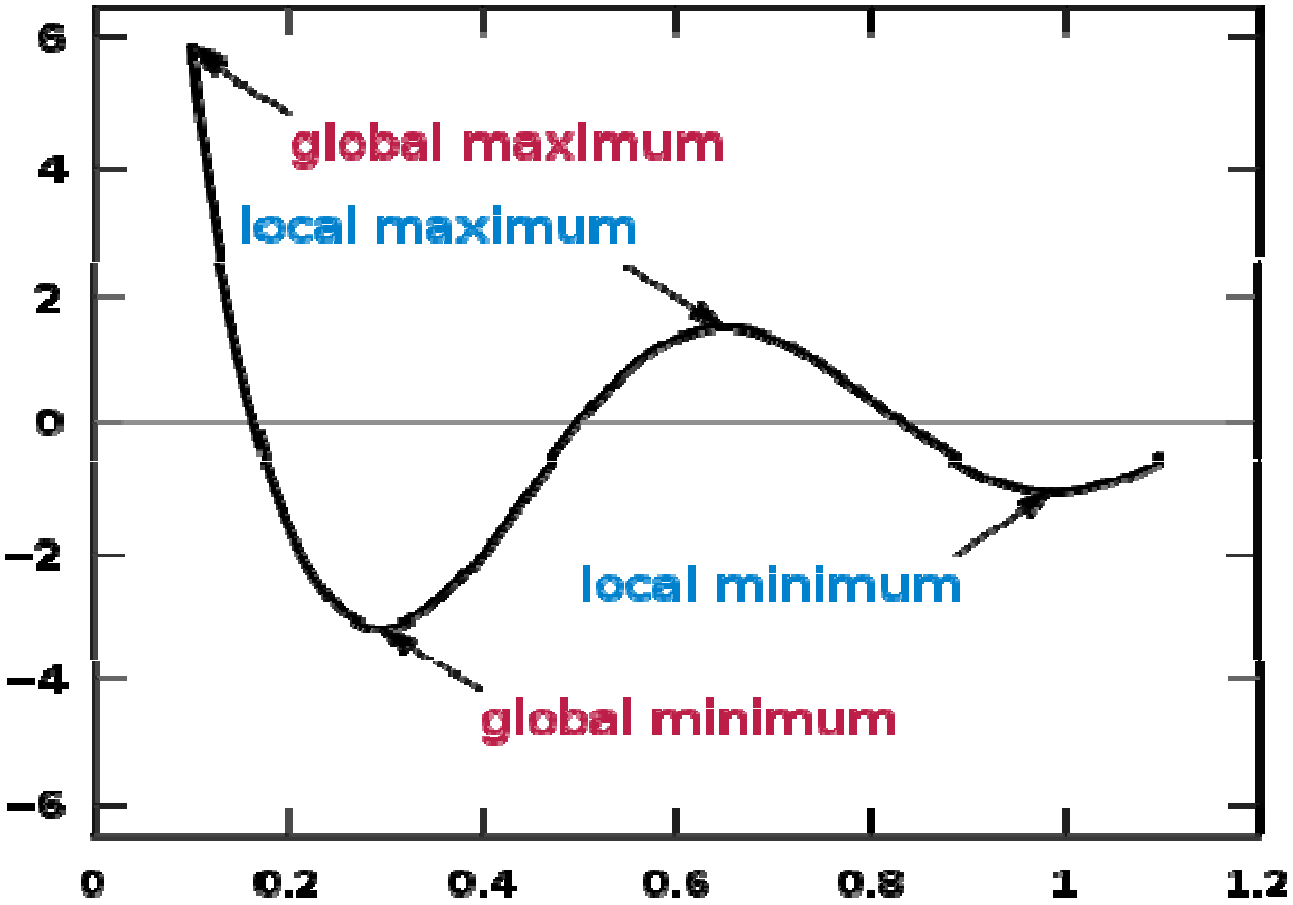
Local and Global Optima

- Global optimum
  - A solution that is better than all other solutions
  - Or no worse than any other solution
- Local optimum
  - A solution which is better than *nearby* solutions
  - A local optimum is not necessarily a global one

Global /Local(max/min)

- A local max/min is over a small area.
  - For instance, if a point is lower than the next nearest point on the left & right then it's a local min.
- There can be many local maxes and mins over an entire graph.
- A global max/min is the highest/lowest point on the entire graph.
- There can only be ONE global max and/or min on a graph and there may not be one at all.

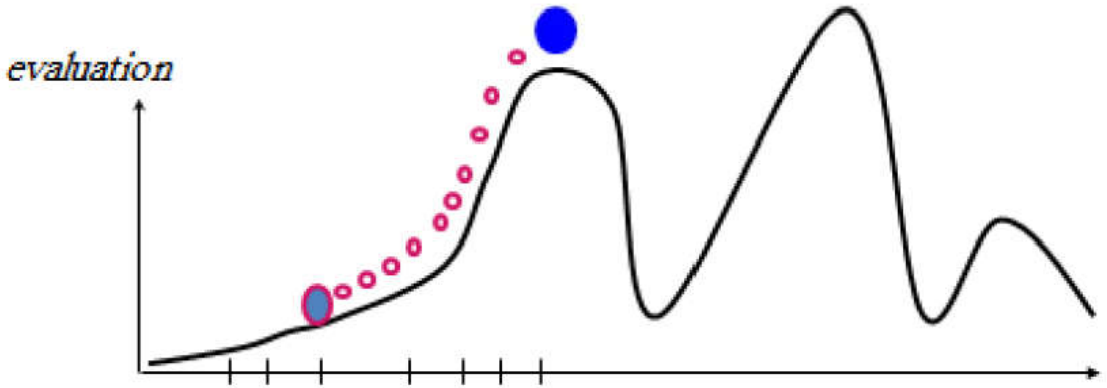
Global /Local(max/min)

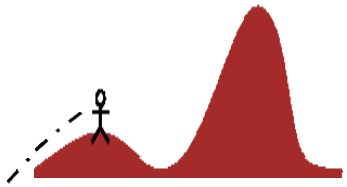


Hill-Climbing Search

Hill-Climbing Search

- **Main Idea**: Keep a single current node and move to a neighboring state to improve it.
- Uses a loop that continuously moves in the direction of increasing value (**uphill**):
  - Choose the best successor, choose randomly if there is more than one.
  - Terminate when a peak is reached where no neighbor has a higher value.
- It is also called **greedy local search**, steepest ascent/descent.





Hill-Climbing Search

- "Like climbing Everest in thick fog with amnesia"
- Only record the state and its evaluation instead of maintaining a search tree

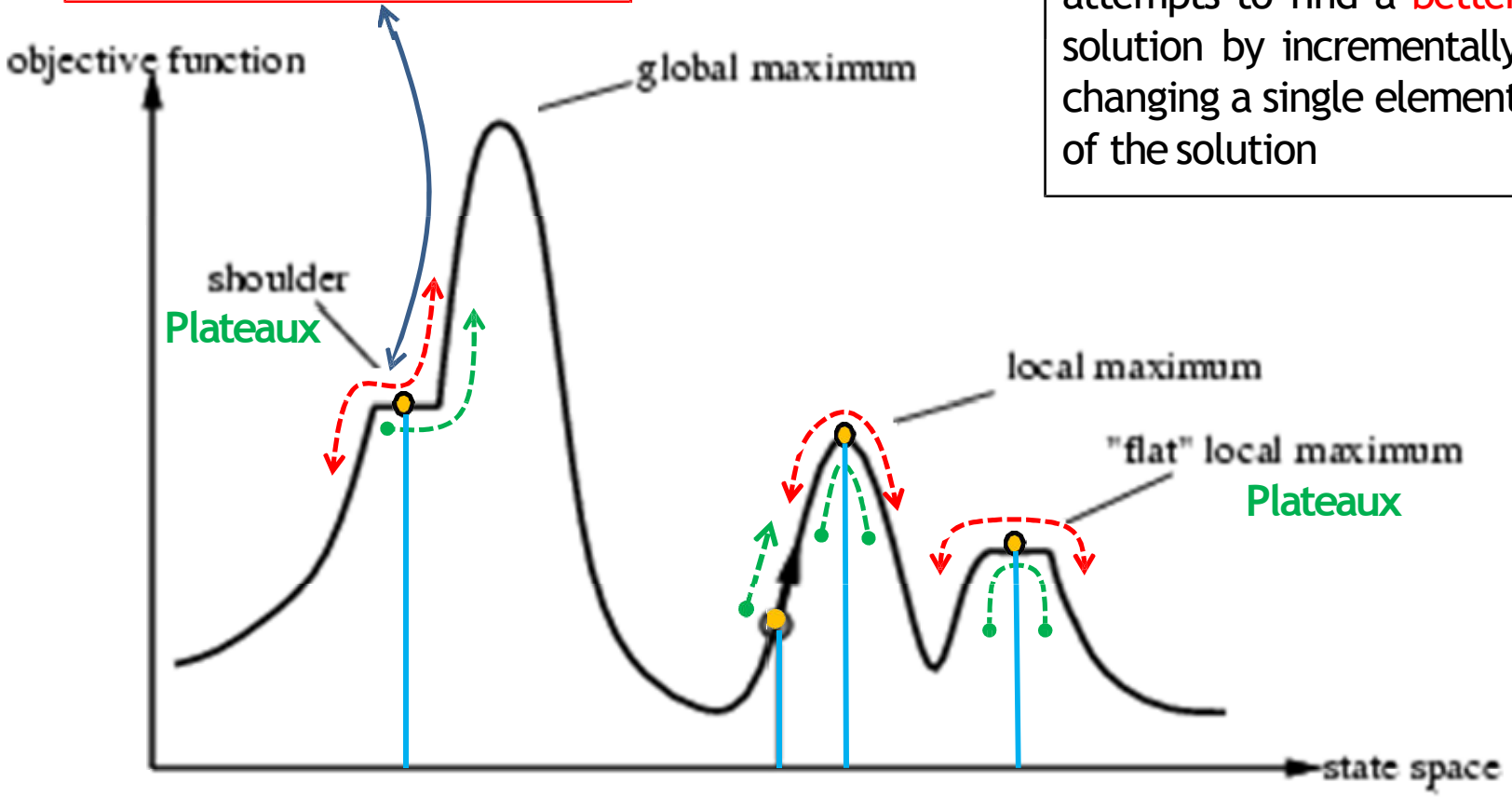
```
function HILL-CLIMBING ( problem ) returns a state that is a local maximum
  inputs: problem, a problem
  local variables:    current, a node
                      neighbor, a node

  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 Search

it is possible to makeprogress

attempts to find a better solution by incrementally changing a single element of the solution



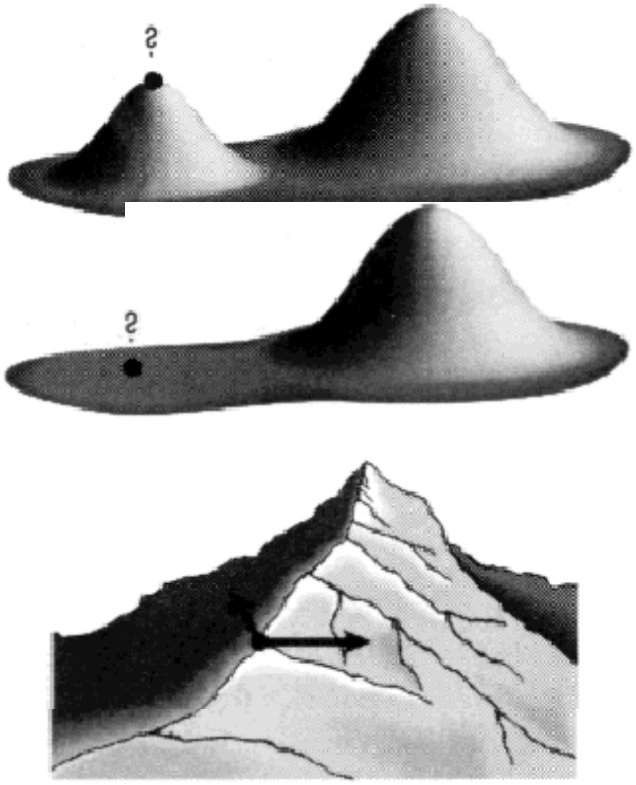
Current state

Hill-Climbing Search

ocal maxima: a local maximum is a peak that is higher than each f its neighboring states, but lower than the global maximum. ill-climbing algorithms that reach the vicinity of a local maximum ill be drawn upwards towards the peak, but will then be stuck ith nowhere else to go.

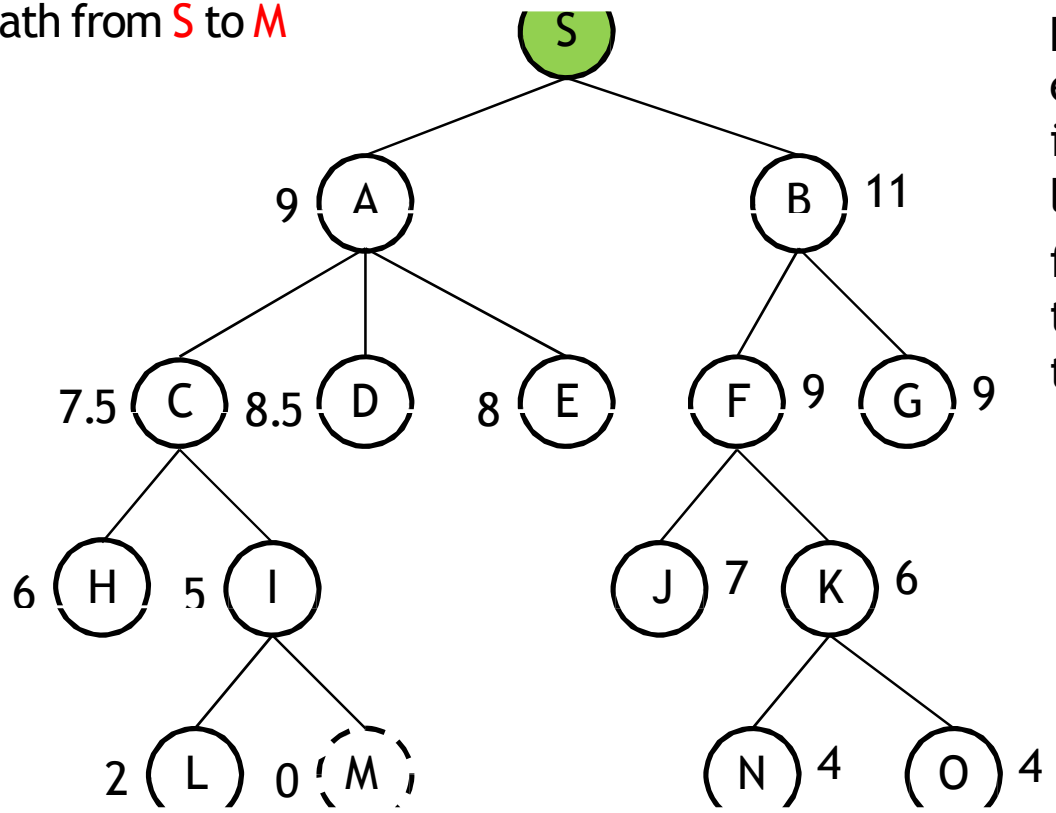
lateaux: a plateau is an area of the state space landscape where he evaluation function is flat. It can be a flat local maximum, from hich no uphill exit exists, or a **shoulder,** from which it is possible to ake progress.

idges: Ridges result in a sequence of local maxima that is very ifficult for greedy algorithms to navigate. (the search direction is ot towards the top but theside)

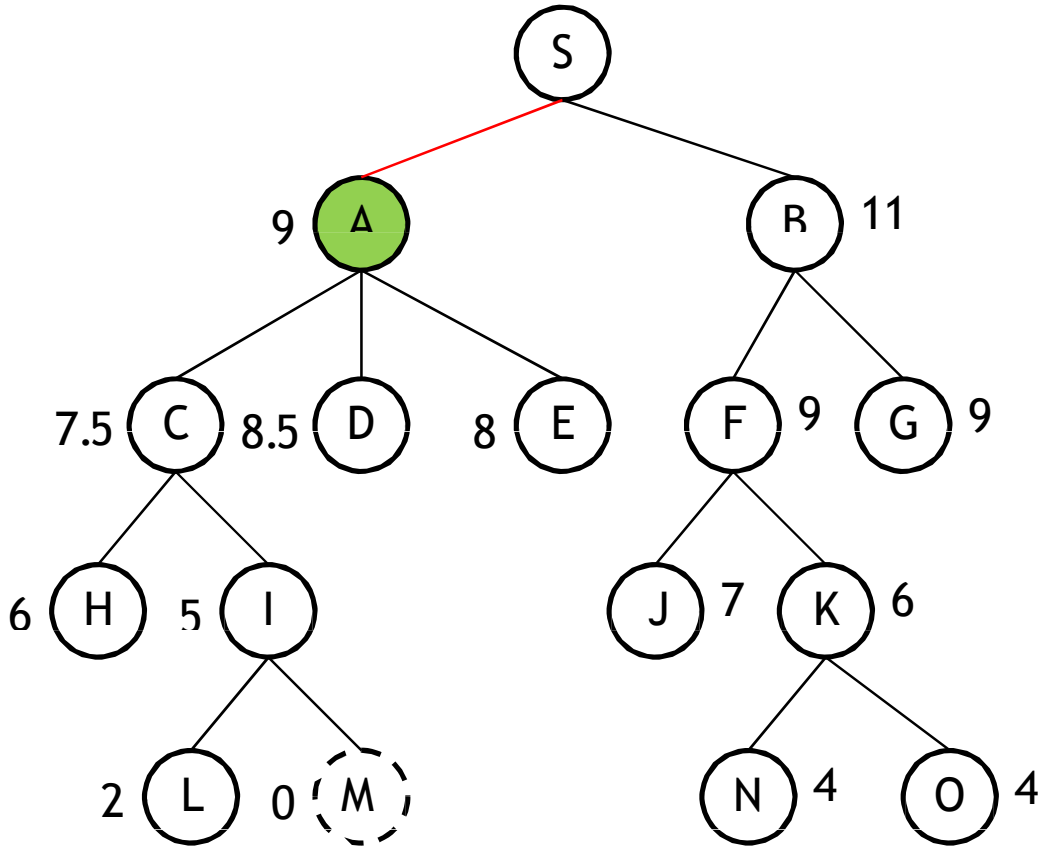


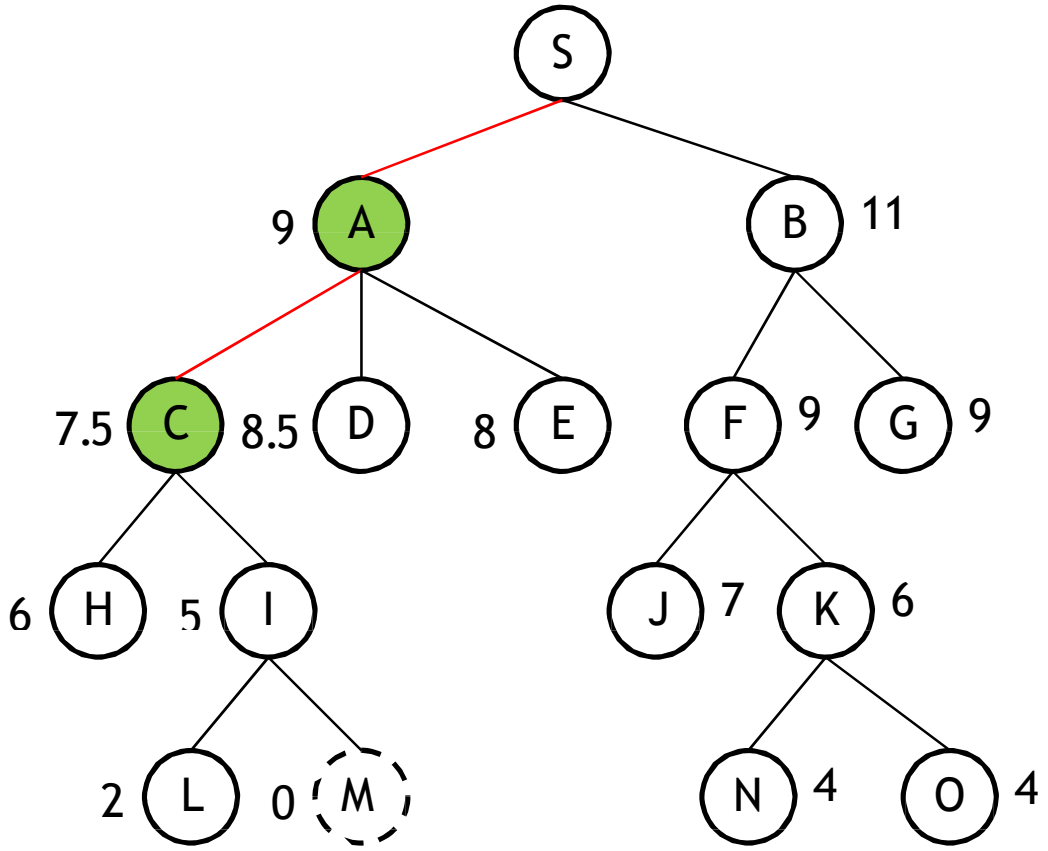
Hill-Climbing Search Example

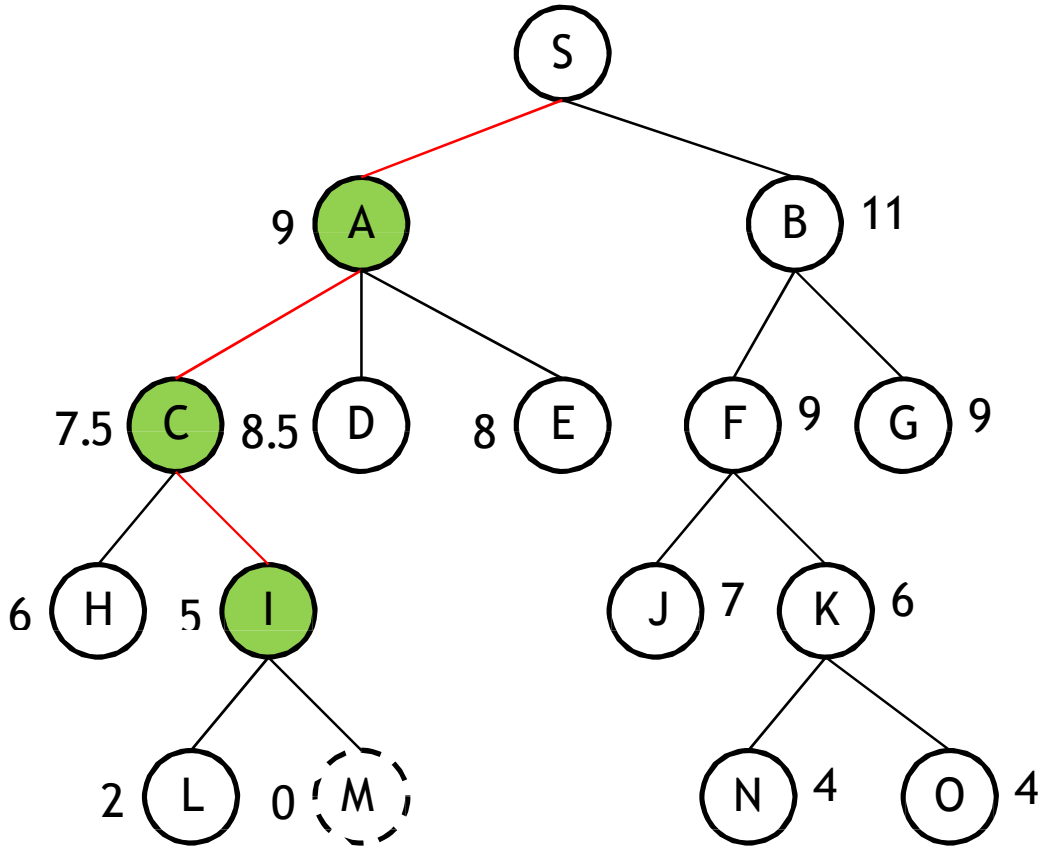
our aim is to find a path from S to M

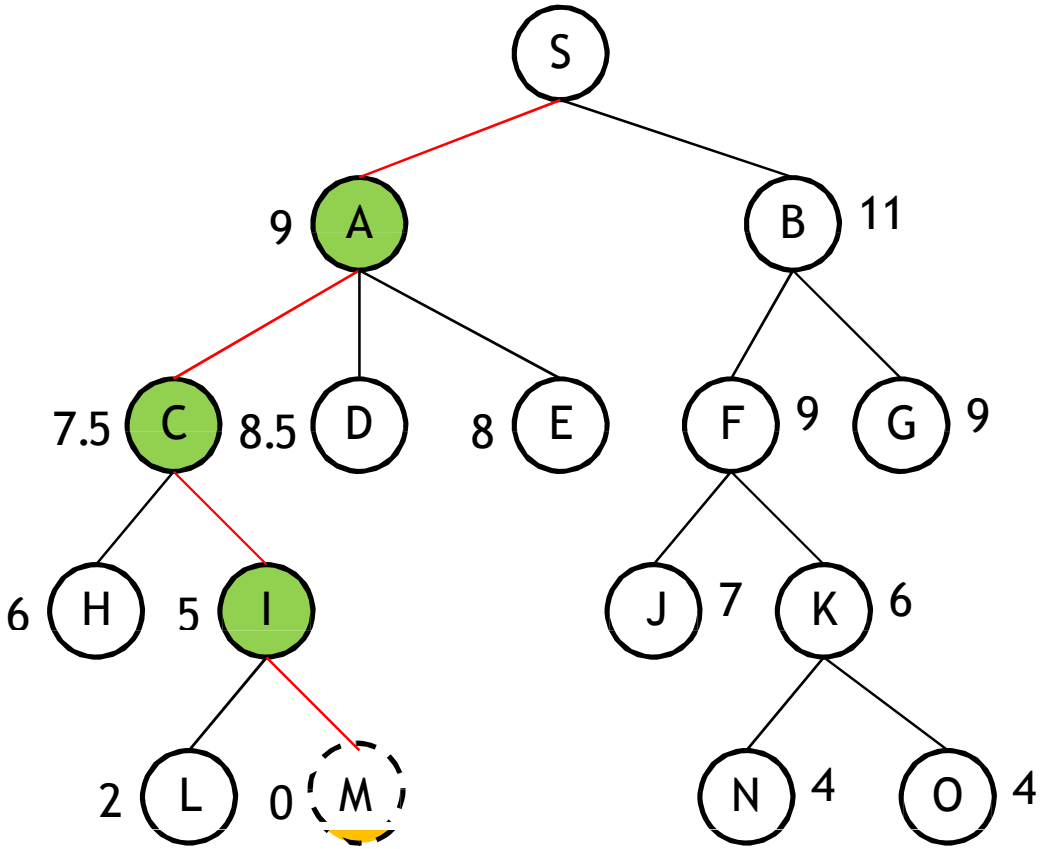


associate heuristics with every node, that is the straight line distance from the path terminating city to the goal city


S
9
A
B
11
7.5
C
8.5
D
8
E
F
9
G
9
6
H
5
I
J
7
K
6
2
L
0
M
N
4
O
4

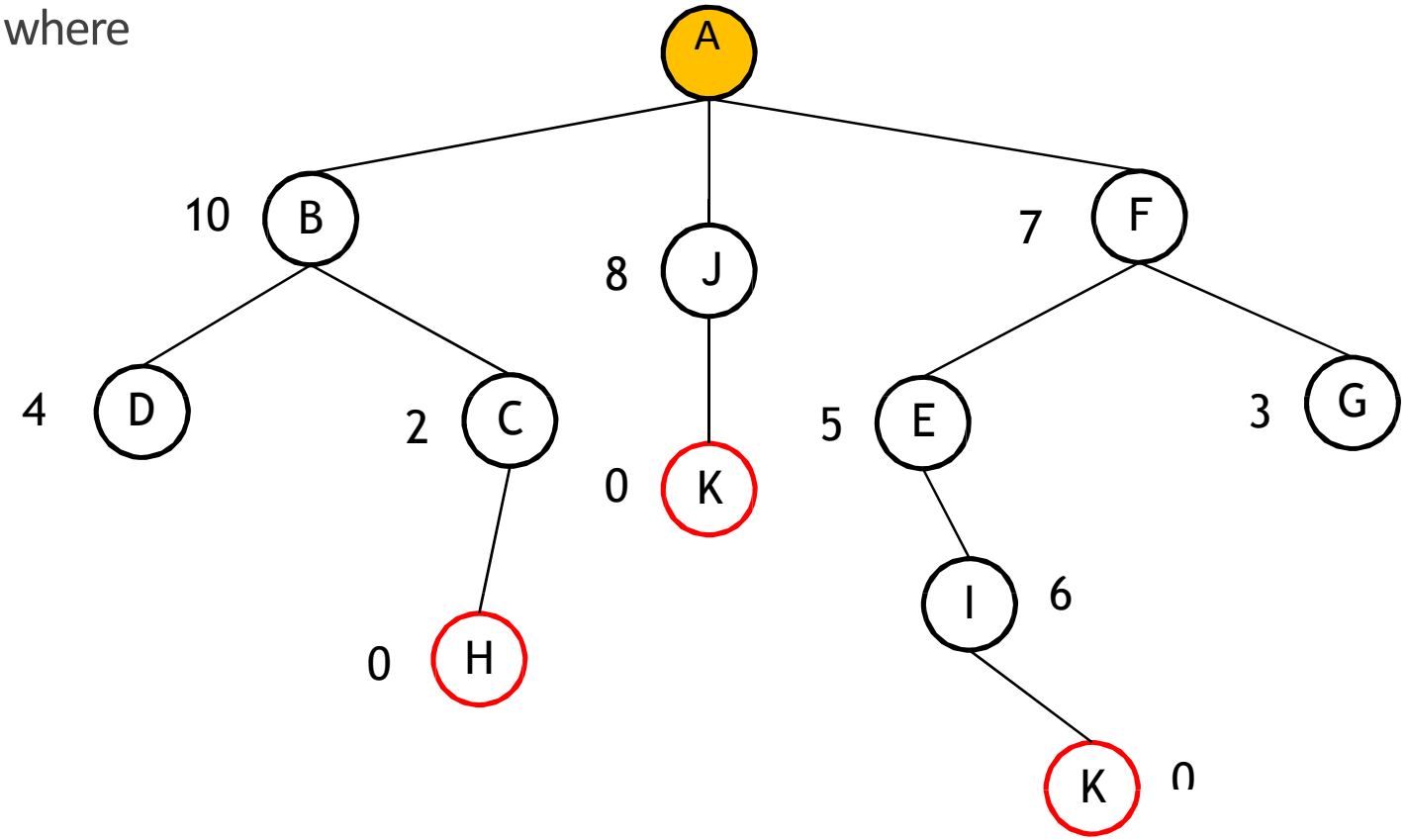

S
9
A
B
11
7.5
C
8.5
D
8
E
F
9
G
9
6
H
5
I
J
7
K
6
2
L
0
M
N
4
O
4


S
9
A
B
11
7.5
C
8.5
D
8
E
F
9
G
9
6
H
5
I
J
7
K
6
2
L
0
M
N
4
O
4

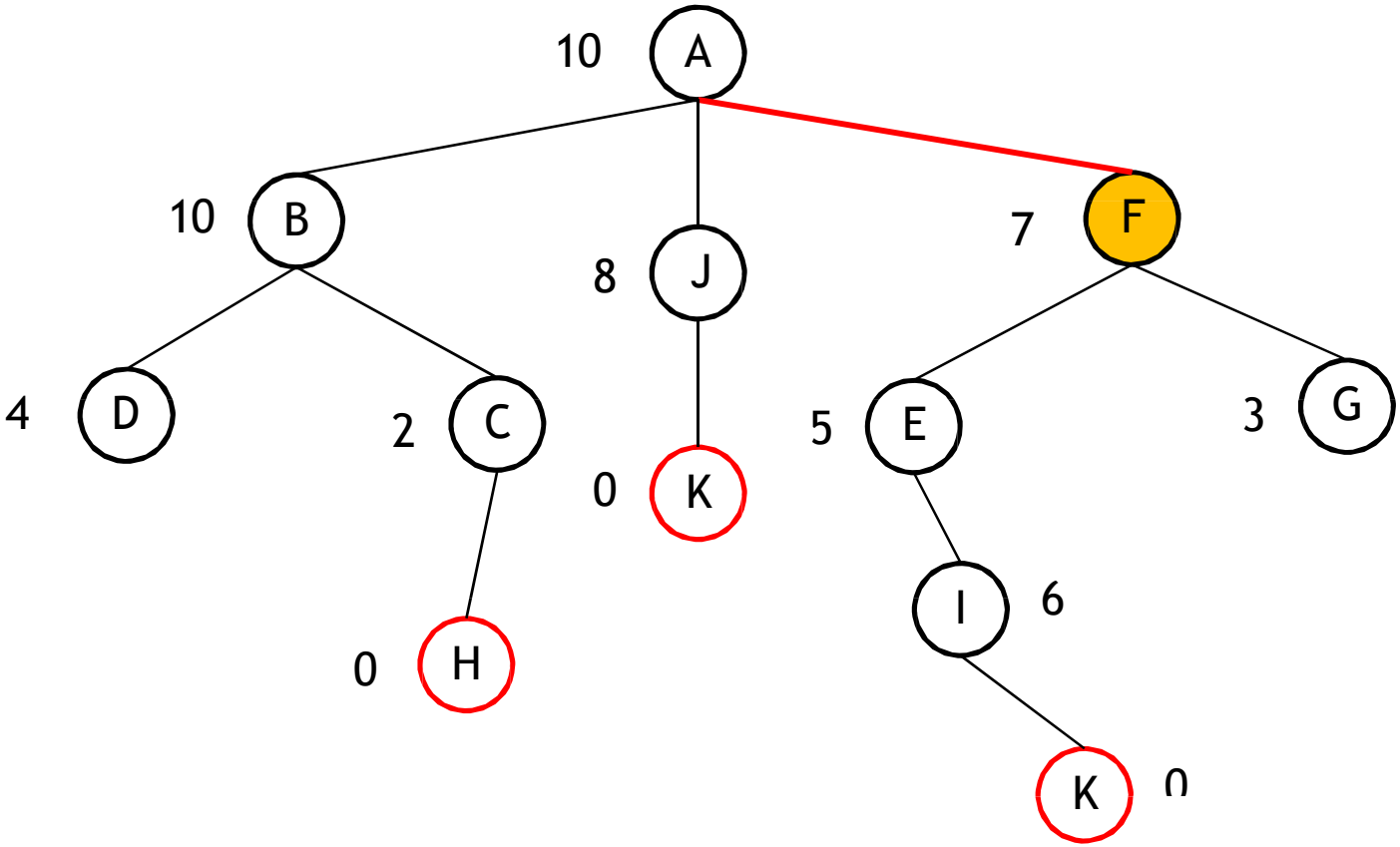

S
9
A
B
11
7.5
C
8.5
D
8
E
F
9
G
9
6
H
5
I
J
7
K
6
2
L
0
M
N
4
O
4

Hill-Climbing Search Example

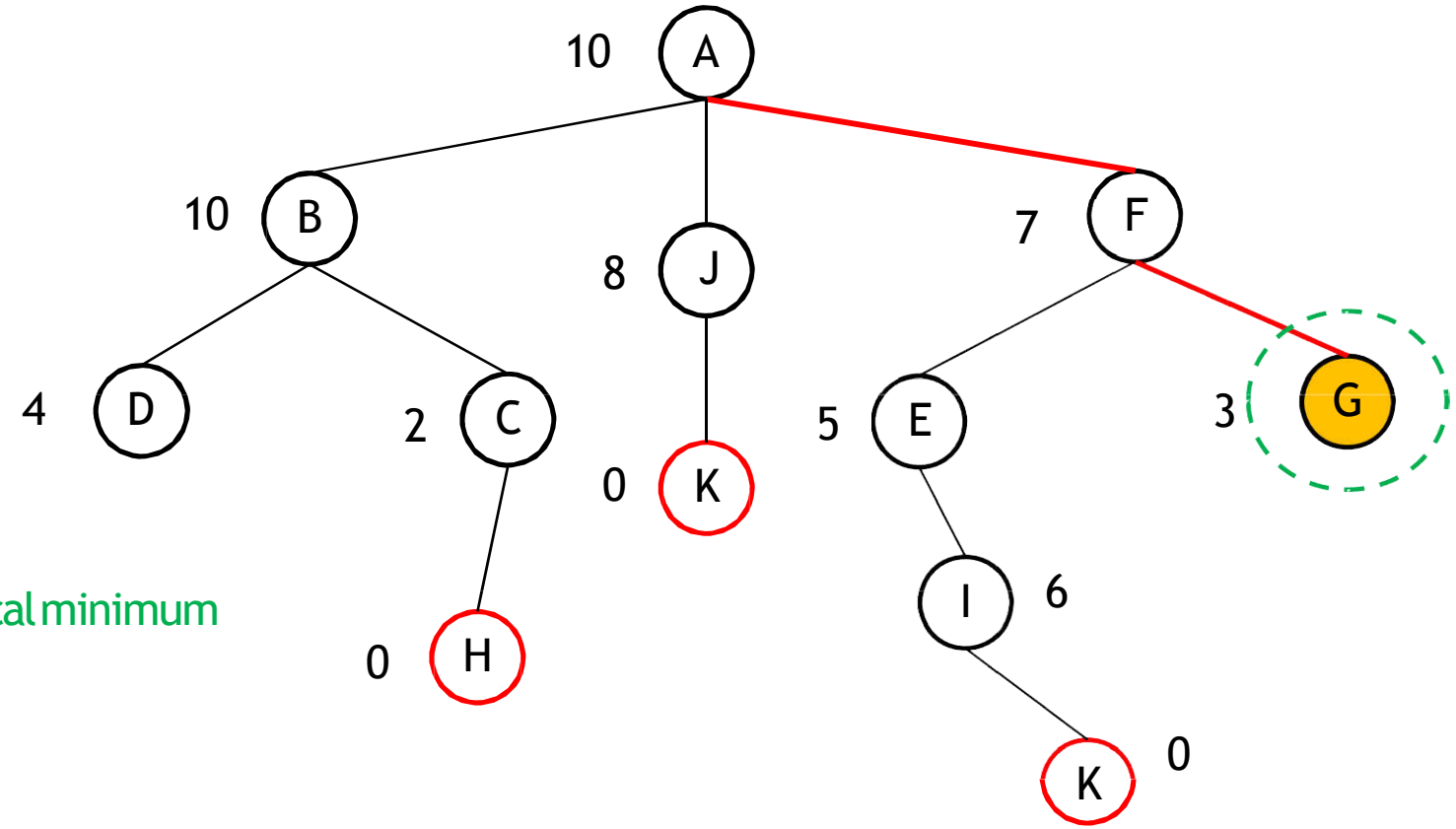
om A find a solution where
and K are final states



Local Maximum



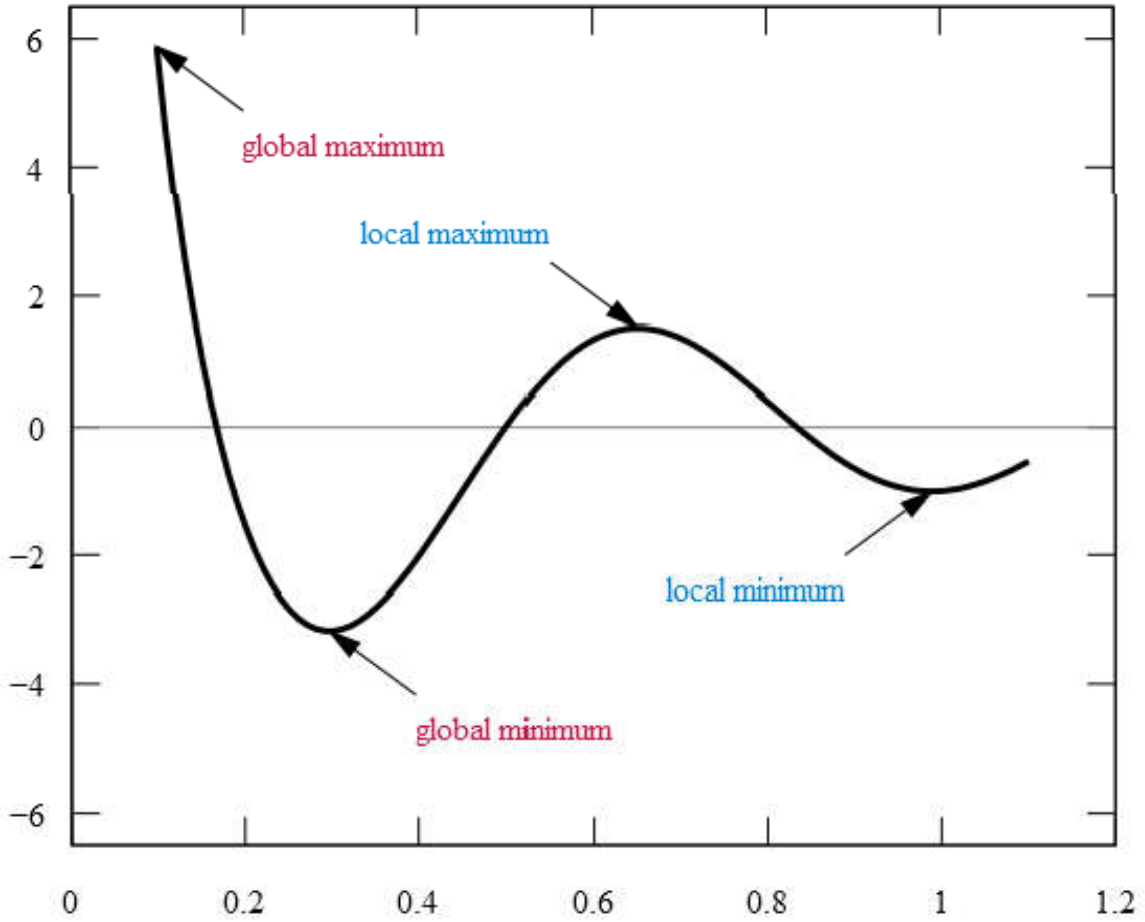
Local Minimum



G is local minimum

ing is sometimes called **greedy local search** because it grabs a good neighbor state without thinking ahead about

Local Maximum, Local Minimum



Alternative hill climbing

- Stochastic hill climbing
  - chooses at random from among the uphill moves (neighbors); the probability of selection can vary with steepness of the uphill move.
  - This usually converges more slowly than the steepest ascent, but in some state landscapes, it finds better solutions.
- First-choice hill climbing
  - implements stochastic hill-climbing by generating successors randomly until one is generated that is bet than the current state.
  - This is a good strategy when a state has many (e.g., thousands) of successors.
- Random-restart hill climbing
  - adopts the well-known adage(proverb), "If at first, you don't succeed, try, try again." It conducts a serie hill-climbing searches from the randomly generated initial state, stopping when a goal is found. t is cor with probability approaching 1, for the trivial reason that it will eventually generate a goal state as the state. (It iteratively does hill-climbing, each time with a random initial condition)

Criteria	Stochastic Hill Climbing	First-Choice Hill Climbing	Random Restart Hill Climbing
Selection of Neighbors	Randomly chooses a neighbor at each step	Selects the first better neighbor found	Examines multiple neighbors but selects only one
Determinism	Non-deterministic	Deterministic	Non-deterministic (due to random restarts)
Efficiency	Less efficient compared to deterministic approaches	Efficient when there are many neighbors to consider	Can be computationally expensive due to multiple restarts
Escape from Local Optima	Less prone to getting stuck in local optima due to randomness	Can get stuck in local optima if the first better neighbor is not the global optimum	Effective in escaping from local optima by exploring different regions
Computational Complexity	Moderate	Low to Moderate	Moderate to High
Application	Suitable for noisy or uncertain environments	Suitable for problems with many neighbors	Suitable for complex search spaces with multiple local optima