Hill Climbing Search

Introduction

- It is simply a loop that continually moves in the direction of increasing value—that is, uphill
- It terminates when it reaches a “peak” where no neighbor has a higher value
- The algorithm does not maintain a search tree, so the data structure for the current node need only record the state and the value of the objective function
- Hill climbing does not look ahead beyond the immediate neighbors of the current state

Hill Climbing Algorithm (Steepest Ascent version)

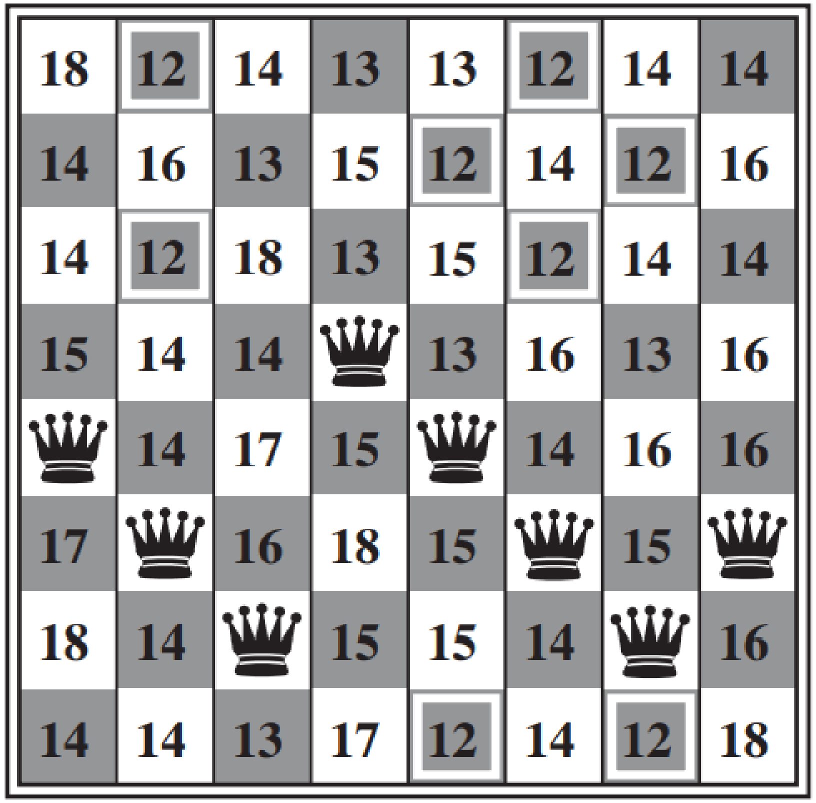
```
function HILL-CLIMBING(problem) returns a state that is a local maximum

  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
```

8-queens problem

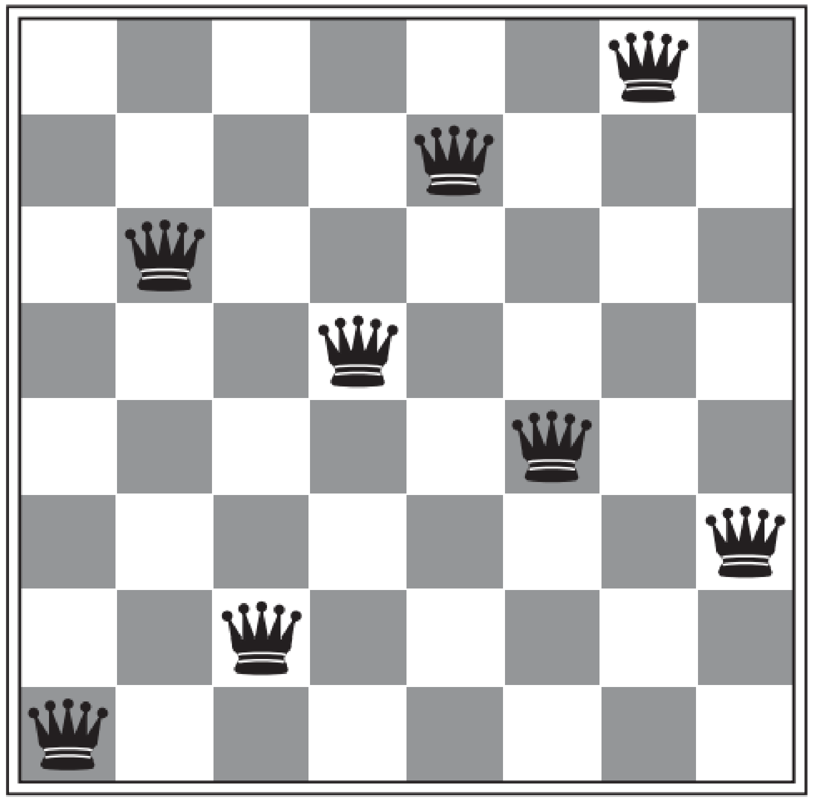
- Suppose each state has 8 queens on the board, one per column
- The successors of a state are all possible states generated by moving a single queen to another square in the same column
- So, each state has $8 \times 7 = 56$ successors
- The heuristic cost function h is the number of pairs of queens that are attacking each other, either directly or indirectly
- The global minimum of this function is zero, which occurs only at perfect solutions

- Hill climbing is greedy local search because it grabs a good neighbor state without thinking ahead about where to go next
- Hill climbing often makes rapid progress toward a solution because it is usually quite easy to improve a bad state



(a)

An 8-queens state with heuristic cost estimate h = 17, showing the value of h for each possible successor obtained by moving a queen within its column. The best moves are marked.



(b)

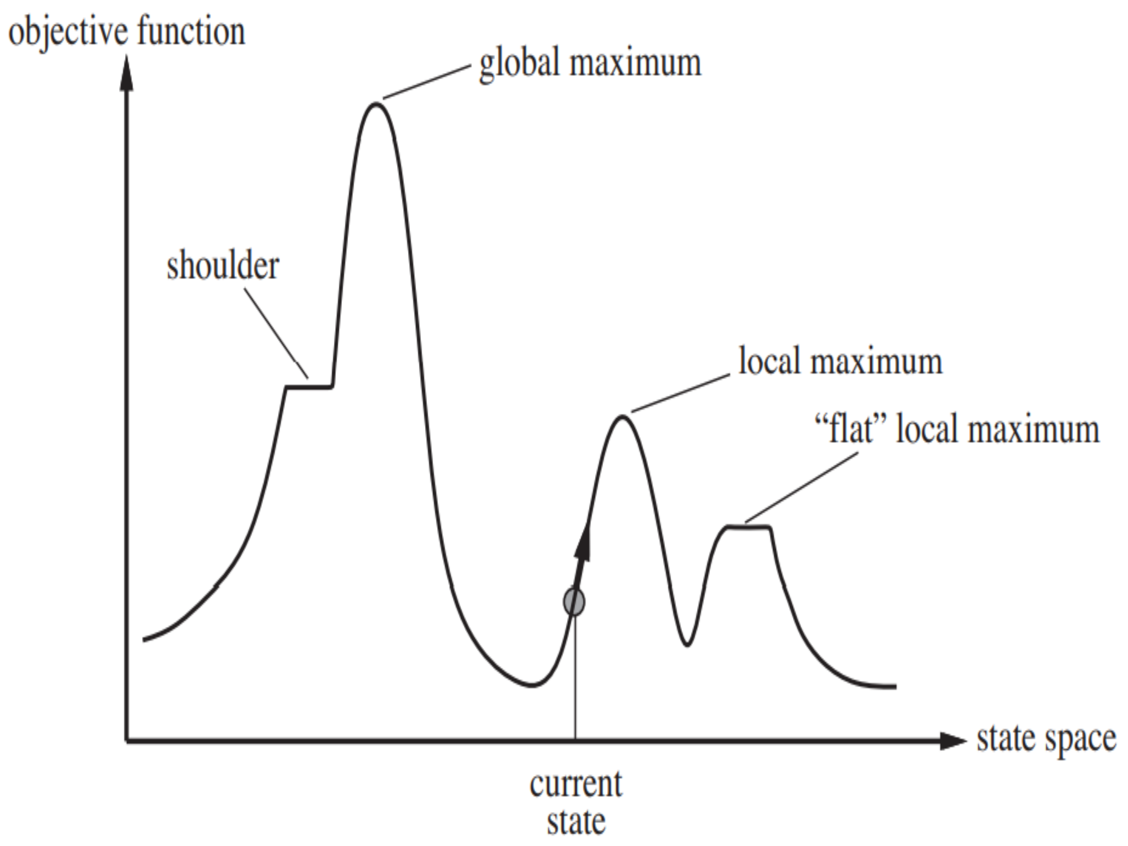
A local minimum in the 8-queens state space; the state has h = 1 but every successor has a higher cost.

- For example, from the state in Figure (a), it takes just five steps to reach the state in Figure (b), which has h = 1 and is very nearly a solution

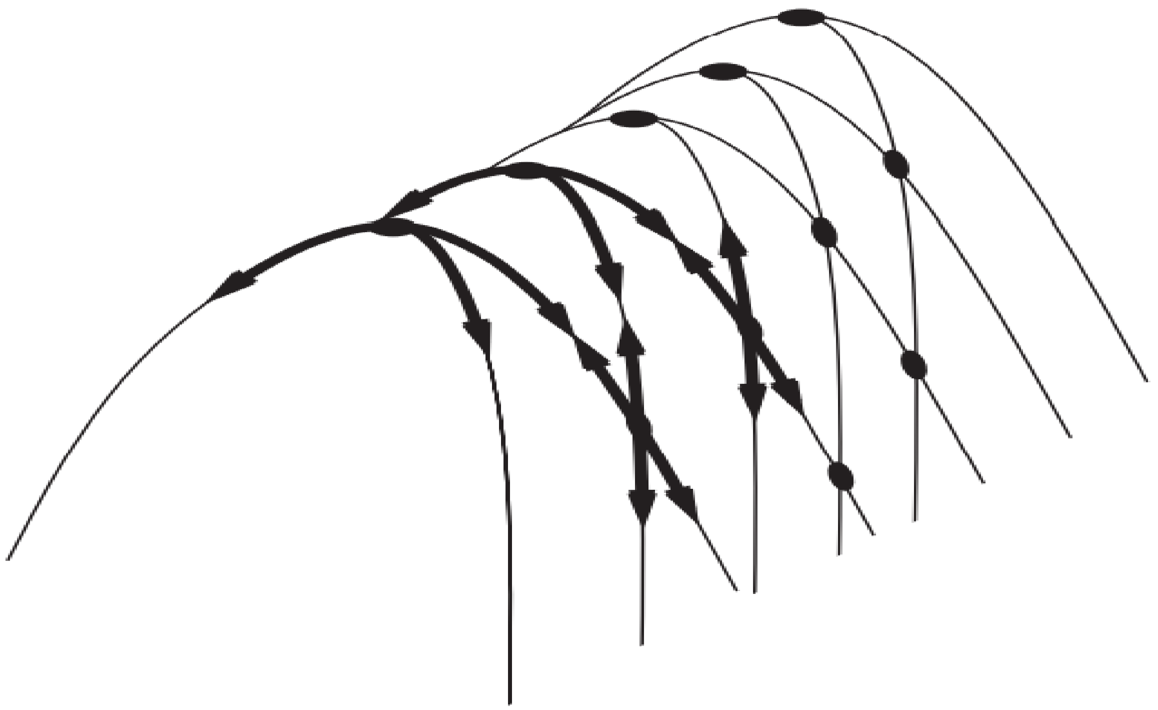
Problem with Hill Climbing

Often get stuck due to local maxima, plateau, and ridge

- **Local maxima:** Hill-climbing algorithms that reach the vicinity of a local maximum will be drawn upward toward the peak but will then be stuck with nowhere else to go
- For example, the state in Figure (b) is a local maximum (i.e., a local minimum for the cost h); every move of a single queen makes the situation worse
- **Plateaux:** a plateau is a flat area of the state-space landscape. It can be a flat local maximum, from which no uphill exit exists, or a shoulder, from which progress is possible. A hill-climbing search might get lost on the plateau



Ridges: result in a sequence of local maxima that is very difficult for greedy algorithms to navigate



The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill

Discussions

- Starting from a randomly generated 8-queens state, steepest-ascent hill climbing gets stuck 86% of the time, solving only 14% of problem instances
- It works quickly, taking just 4 steps on average when it succeeds and 3 when it gets stuck—not bad for a state space with 8^8 ≈ 17 million states
- The algorithm halts if it reaches a plateau where the best successor has the same value as the current state
- Is it a good idea to keep going—to allow a sideways move in the hope that the plateau is really a shoulder? The answer is usually yes, but we must take care

- If we always allow sideways moves when there are no uphill moves, an infinite loop will occur whenever the algorithm reaches a flat local maximum that is not a shoulder
- One common solution is to put a limit on the number of consecutive sideways moves allowed
- For example, we could allow up to, say, 100 consecutive sideways moves in the 8-queens problem
- This raises the percentage of problem instances solved by hill climbing from 14% to 94%
- Success comes at a cost: the algorithm averages roughly 21 steps for each successful instance and 64 for each failure

- Many variants of hill climbing have been invented
- Stochastic hill climbing chooses at random from among the uphill moves; the probability of selection can vary with the steepness of the uphill move

  - This usually converges more slowly than 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 better than the current state

  - This is a good strategy when a state has many (e.g., thousands) of successors

- The hill-climbing algorithms described so far are incomplete—they often fail to find a goal when one exists because they can get stuck on local maxima
- Random-restart hill climbing conducts a series of hill-climbing searches from randomly generated initial states, until a goal is found

  - It is trivially complete with probability approaching 1, because it will eventually generate a goal state as the initial state
- If each hill-climbing search has a probability p of success, then the expected number of restarts required is 1/p

- For 8-queens instances with no sideways moves allowed, $p \approx 0.14$, so we need roughly 7 iterations to find a goal (6 failures and 1 success)
- When we allow sideways moves, $1/0.94 \approx 1.06$ iterations are needed on average
- For 8-queens, then, random-restart hill climbing is very effective indeed
- Even for three million queens, the approach can find solutions in under a minute

Conclusion

- The success of hill climbing depends very much on the shape of the state-space landscape
- If there are few local maxima and plateaux, random-restart hill climbing will find a good solution very quickly