Artificial Intelligence

Lec 6: Local Search

Pratik Mazumder

Content taken from various sources including
CS188 Intro to AI at UC Berkeley

Satisfaction and Optimization

- In our search algorithms, we were solving some satisfaction and some optimization problems
  - Satisfaction: Find a path to the goal
  - Optimization:
    - Find the optimal path to the goal
      or
    - Optimize some objective function in general

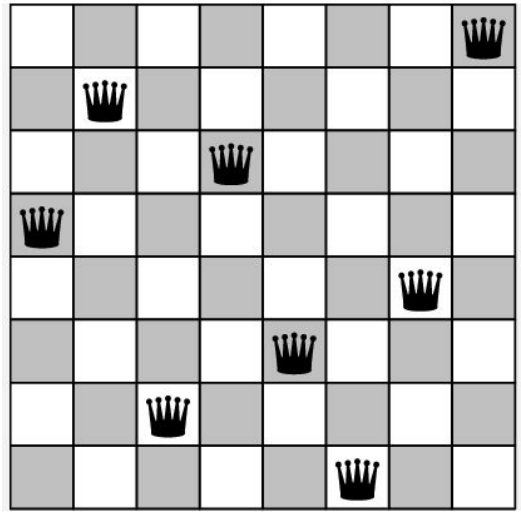
- In the context of optimization problems, the terms "objective function" and "cost function" are often used interchangeably, but they can have slightly different meanings depending on the context.

- In general, an objective function is a measure of the quality of a state or a solution, while a cost function is a measure of the resources or effort required to reach a state or a solution.

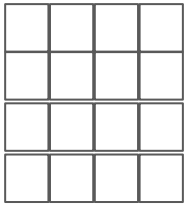
- The objective function is used to evaluate the quality of a solution, while the cost function is used to evaluate the efficiency of a search algorithm.

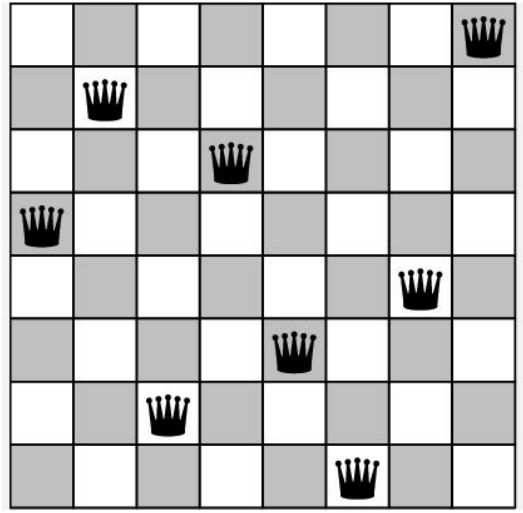
N-queens problem

- Goal of the N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.
- Queen attacks any piece in the same row, column or diagonal
- No two or more queens on the same row, column or diagonal
- Although efficient special-purpose algorithms exist for this problem and for the whole n-queens family, it remains a useful test problem for search algorithms.

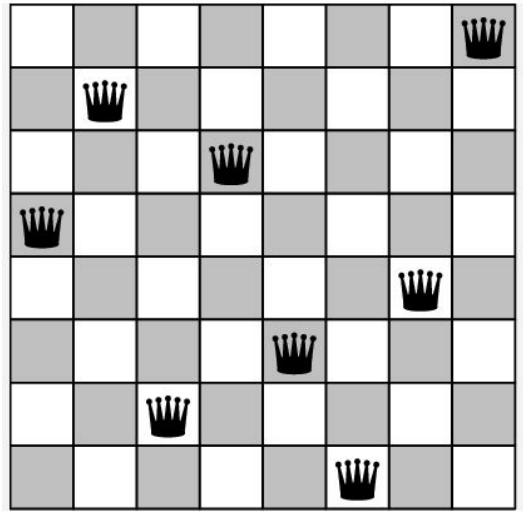


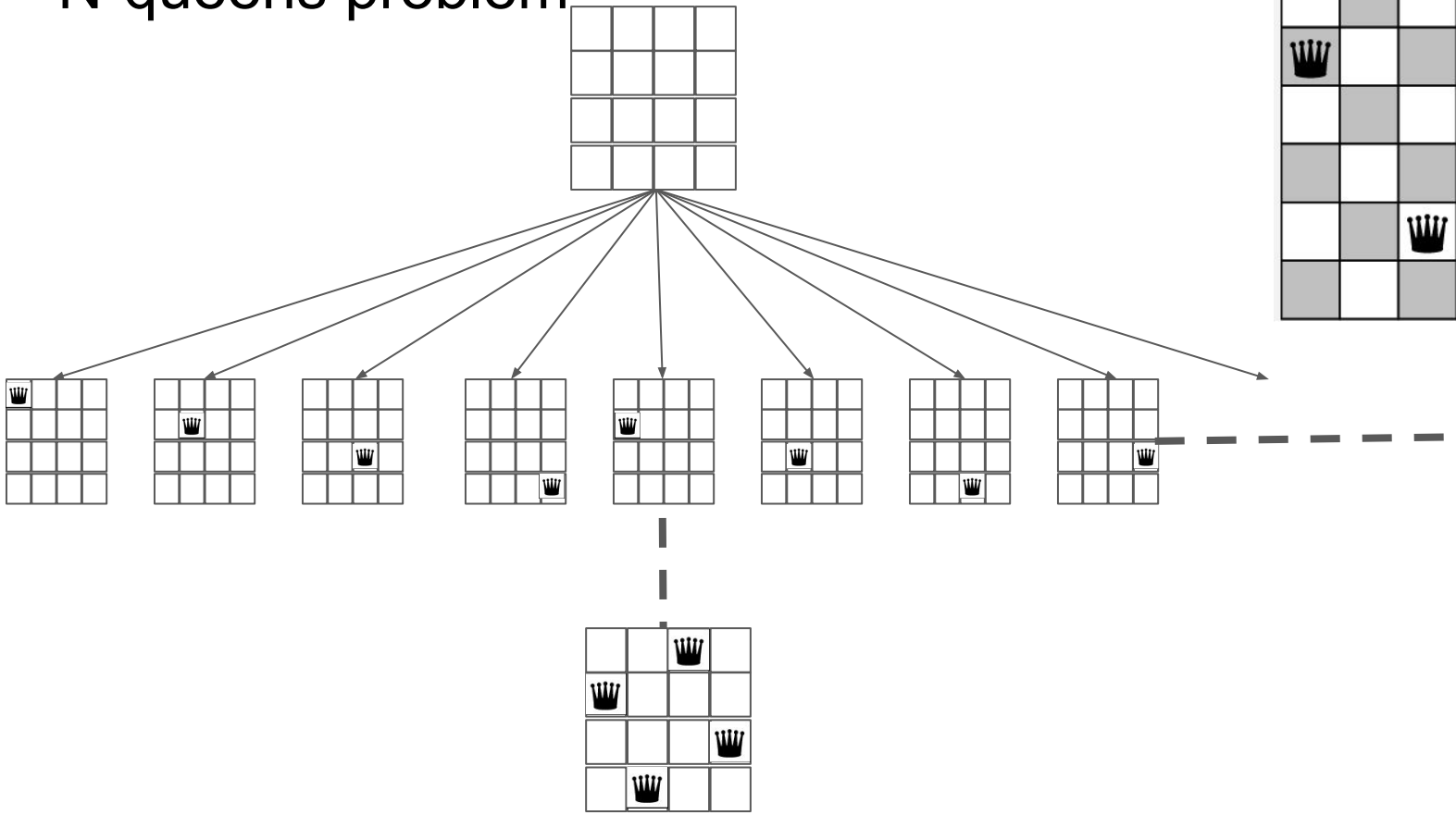
N-queens problem





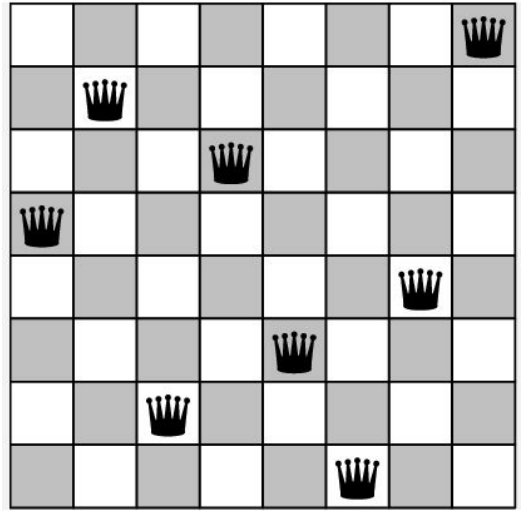
N-queens problem

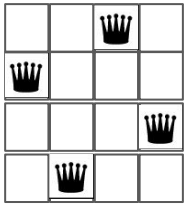




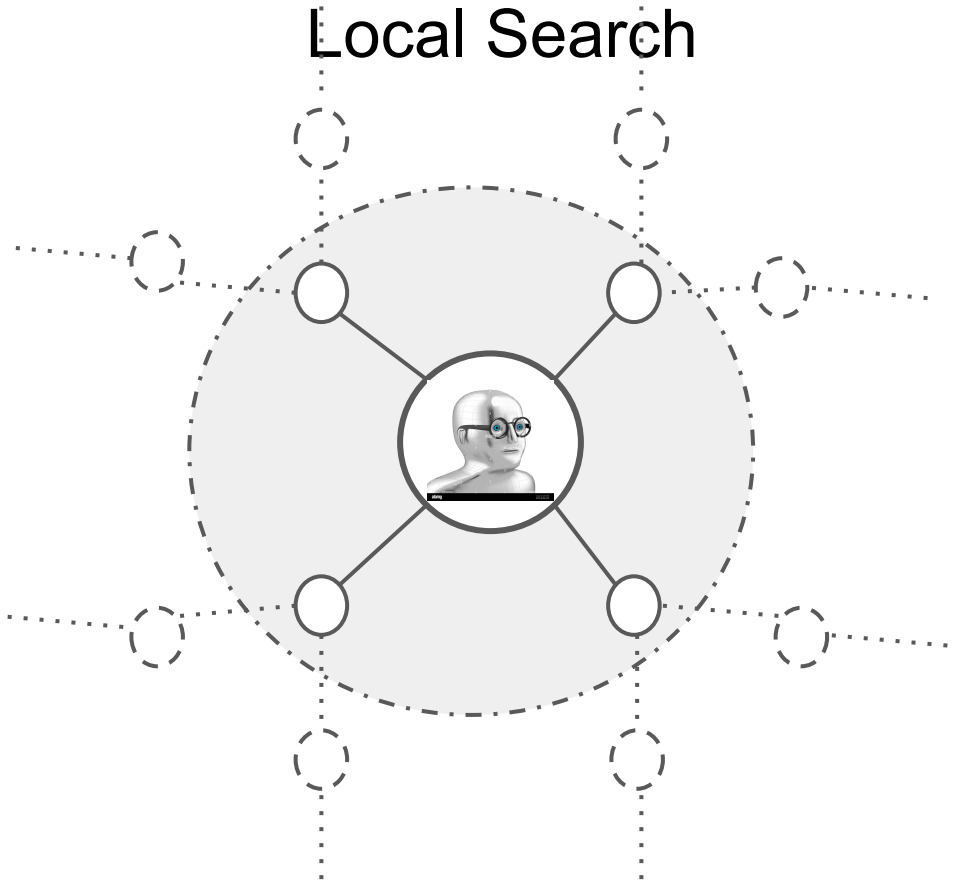
N-queens problem

- Goal of the N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.
- Queen attacks any piece in the same row, column or diagonal
- No two or more queens on the same row, column or diagonal
- Although efficient special-purpose algorithms exist for this problem and for the whole n-queens family, it remains a useful test problem for search algorithms.
- What is the maximum size of the solution using a search tree?
- What is the minimum size of the solution using a search tree?
- Does the sequence of steps matter or the final arrangement?





Local Search



Different way to think about Search: Local Search

- Class of algorithms that operate using a single current node and **generally move only to neighbors of that node**.

- Search Algorithms **Upto Now**: **path to goal** is the **solution** to the problem.

    - Roughly systematic exploration of the search space
    - This systematicity is achieved by **keeping one or more paths in memory** and by recording which alternatives have been explored at each point along the path.
    - When a goal is found, the **path to that goal constitutes a solution** to the problem.

- For some problems the **sequence of action leading to the goal state may be irrelevant.**
    - E.g. 8-queens problem - Place 8 queens on a chess board such that no queen is attacking the other.

- We will consider a setting where the **state or node itself will be a solution to the problem**. **HOW????**

- **Local search** algorithms are useful/efficient in such cases.

- Unlike systematic search algorithms, local search algorithms **do not retain the paths** followed during the search.

Local Search

- Local Search
  - Keep track of single current state
  - Move to only neighboring states
  - Ignore paths

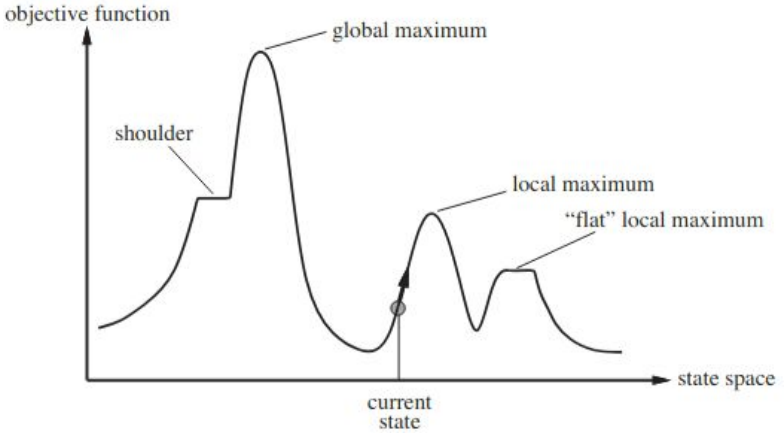
- Advantages
  - Use very little memory
  - Can often find reasonable solutions in large or infinite (continuous) state spaces

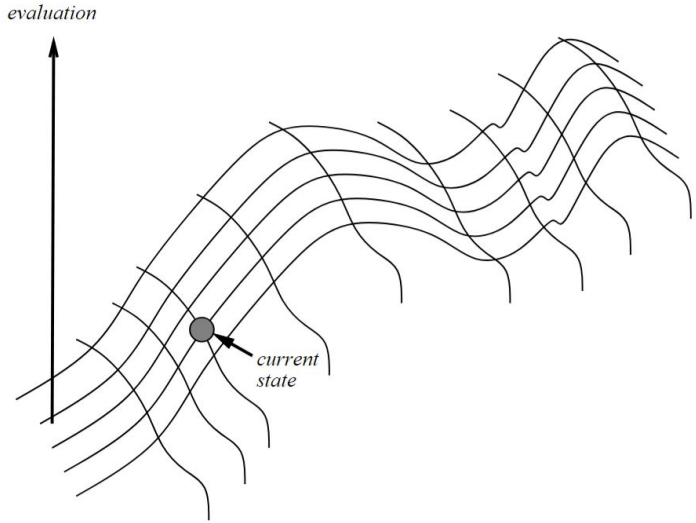
- Suitable problems
  - All states have an objective function
  - Goal is to find a state with the max (or min) objective value.
  - Does not quite fit into the path-cost formulation
  - Local search can do quite well on these problems

- Any state is a potential solution – is there any advantage??

State Space Landscape

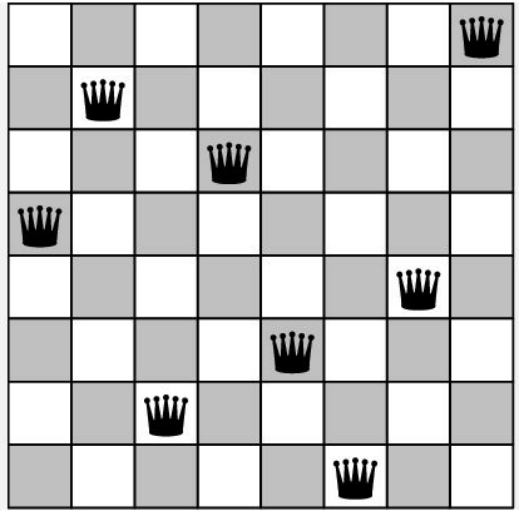
- A landscape has both “**location**” (defined by the **state**) and “**elevation**” (defined by the **value** of the heuristic cost function or objective function).
- If elevation corresponds to **cost**, then the **aim** is to find the **lowest valley**—a global minimum.
- If elevation corresponds to an **objective function**, then the **aim** is to find the **highest peak**—a global maximum.
- Local search algorithms explore this landscape.
- Ideally, the evaluation function h should be monotonic: the **closer** a state to an **optimal goal state** the **better** its **h-value.**
- Each state can be seen as a point on a surface.
- The search consists of moving on the surface, looking for its highest peaks (or lowest valleys): the optimal solutions.





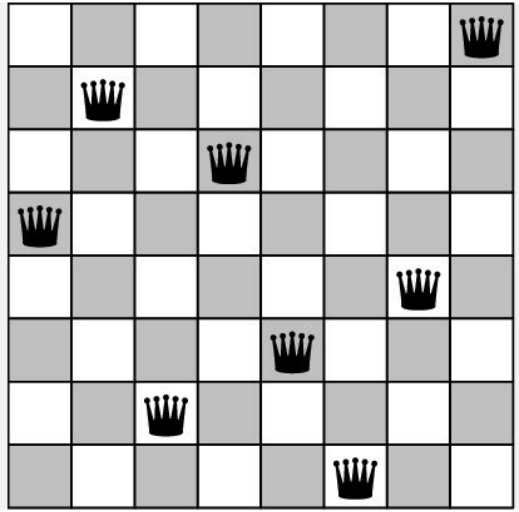
Local Search: N-queens problem

- Goal of the N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.
- Queen attacks any piece in the same row, column or diagonal
- No two or more queens on the same row, column or diagonal
- Although efficient special-purpose algorithms exist for this problem and for the whole n-queens family, it remains a useful test problem for search algorithms.
- Does the sequence of steps matter or the final arrangement?
- Since the order does not matter, we can use local search for this problem.



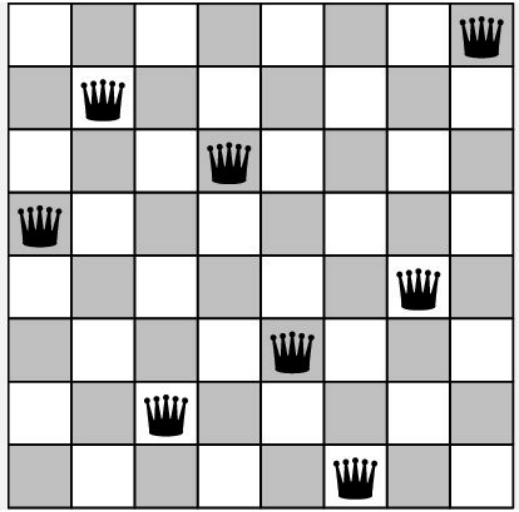
Local Search: N-queens problem

- We have to formulate it as an optimization problem and a state will be the solution.
- We need an objective function h for this
- h = number of pair of queens attacking each other
- Objective function will be minimized in this case
    - Min = 0, Max= 28 for 8 Queen problem - 8C2
- States are possible solutions
- What should a state look like in terms of the queens on the board?



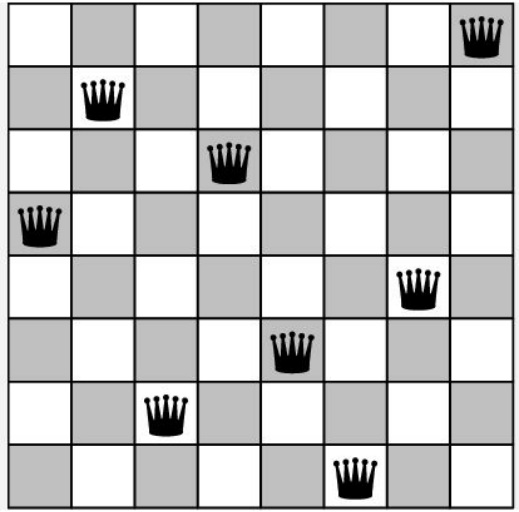
Local Search: N-queens problem

- We have to formulate it as an optimization problem and a state will be the solution.
- We need an objective function h for this
- h = number of pair of queens attacking each other
- Objective function will be minimized in this case
    - Min = 0, Max= 28 for 8 Queen problem - 8C2
- States are possible solutions
- What should a state look like in terms of the queens on the board?
    - All N=8 queens on the board
    - Can be many such states but all of them will not be solutions since the objective function value has to be lowest for that state
    - Can introduce **some conditions to reduce the number of states**: Exactly One queen per column



Local Search: N-queens problem

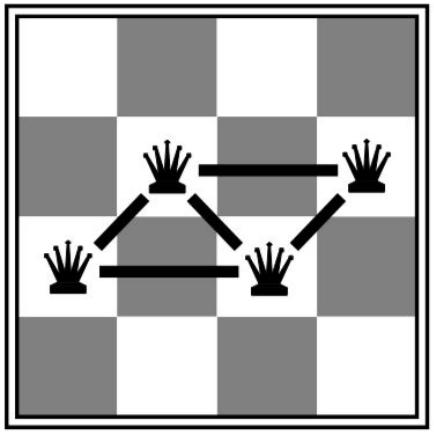
- We have to formulate it as an optimization problem and a state will be the solution.
- We need an objective function h for this
- h = number of pair of queens attacking each other
- Objective function will be minimized in this case
    - Min = 0, Max= 28 for 8 Queen problem - 8C2
- States are possible solutions
- What should a state look like in terms of the queens on the board?
    - All N=8 queens on the board
    - Can be many such states but all of them will not be solutions since the objective function value has to be lowest for that state
    - Can introduce **some conditions to reduce the number of states**: Exactly One queen per column

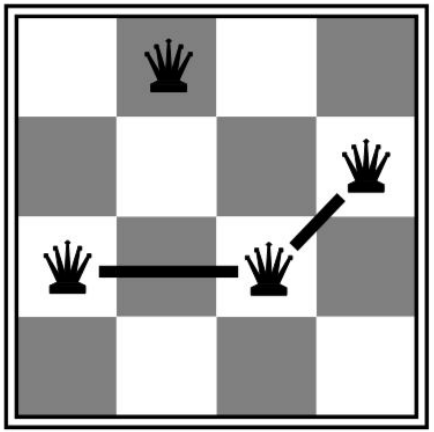


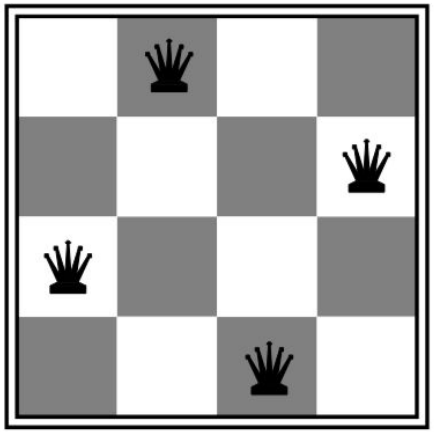
No state can be a Partial Solution as opposed to the state space in Regular Search problems

Local Search: N-queens problem

h = number of pairs of queens attacking each other

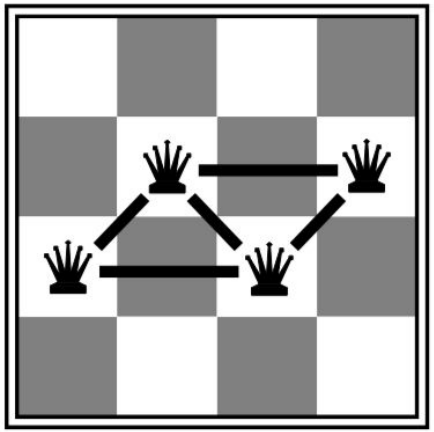




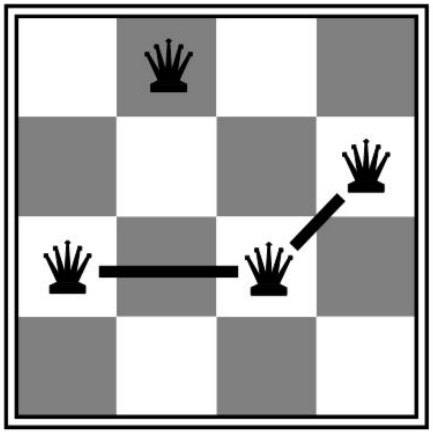


Local Search: N-queens problem

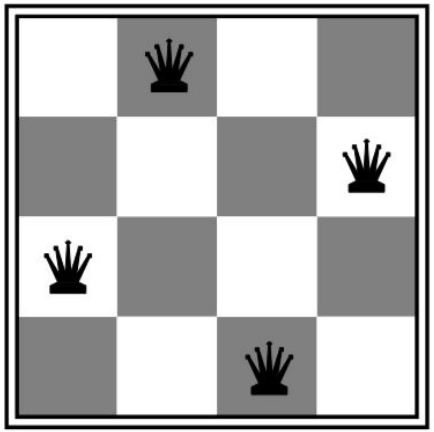
h = number of pairs of queens attacking each other



h = 5



h = 2

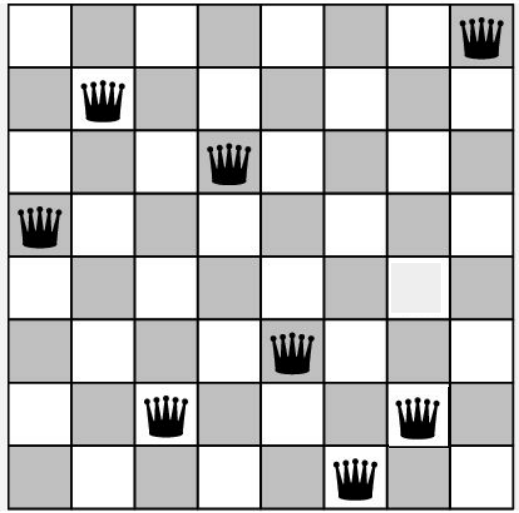


h = 0

Local Search: N-queens problem

- We have defined the states and objective functions.

- We need to define the successor function/neighbourhood function
  a. Option 1: One queen can move to any position inside its column
     - Max Number of neighbours: 56
  b. Option 2: One queen can move 1 step inside its column
     - Max Number of neighbours: 16
  c. Option 3: Move any number of queens anywhere, but in their respective column only
     - Max Number of neighbours: 8^8 - 1 (All possible states)
  d. Option 4: …….

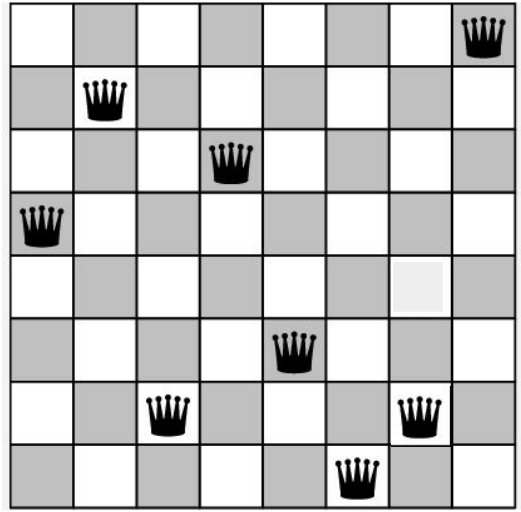


- Idea behind neighbourhood function: Two states in the neighbourhood of each other are similar solutions.

- A **good neighborhood function** is one that results in a good balance between the number of states in the immediate neighborhood and the length of the path to the solution state.

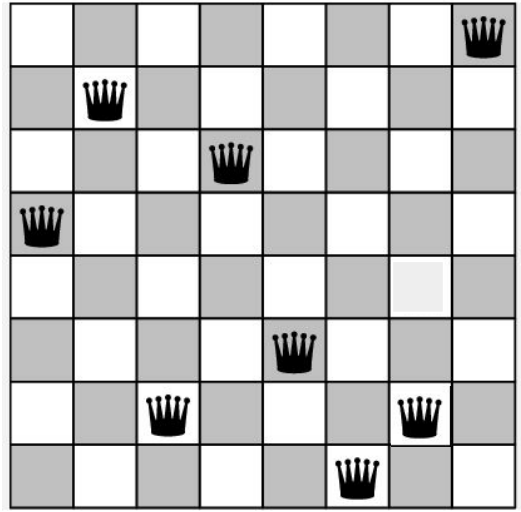
Local Search: N-queens problem

- States: All 8 queens placed on the 8x8 board in some configuration.
- Successor function: Move a single queen to another square in the same column.
- Objective function: The number of pairs of queens that are attacking each other.
    - In this case, the objective function has to be minimized.
- Find a state that minimizes this objective function.



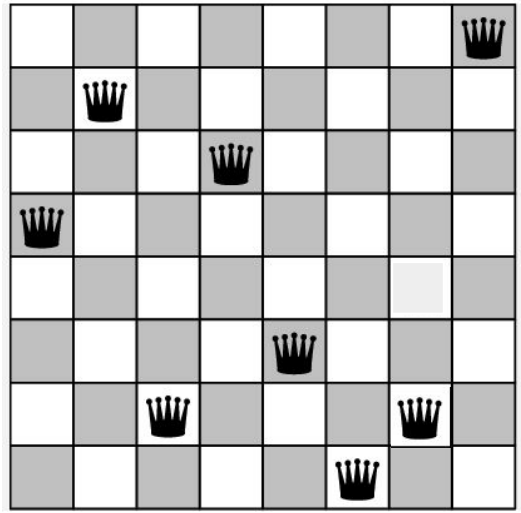
N-queens problem - Trivial Solutions

- Random Sampling
  - Generate a state randomly.
- Random Walk
  - Randomly pick a neighbor of the current state.
- Ignoring memory and time restrictions, these are complete.
- Too slow to be useful in practice.
- Can we make some changes to make a more practical algorithm?

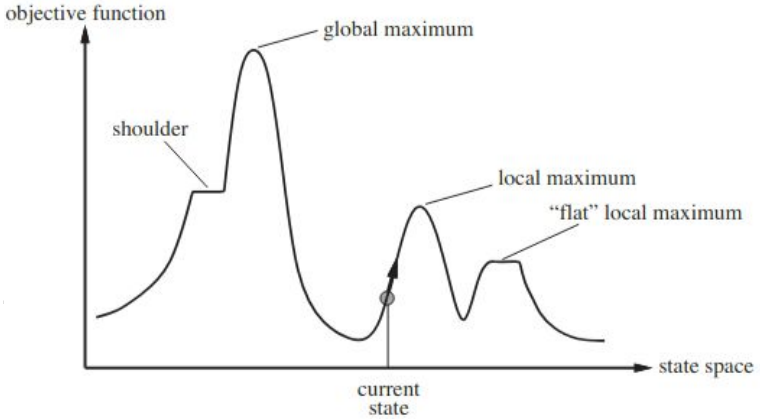


Local Search: Hill-climbing (Greedy Local Search)

- Suppose you are at a state with multiple neighbors.
- For each neighbor, you have the objective function value.
- Strategy:
  - You can pick the neighbor with the best objective function and repeat
  - But then, when will you stop?
    - If **none** of the neighbours have a **lower objective function value** then stop [Case: when objective function has to be minimized]
    - If **none** of the neighbours have a **higher objective function value** then stop [Case: when objective function has to be maximized]
- Hill-climbing or Greedy Local Search



Hill-climbing (Greedy Local Search)



- "A **loop** that **continuously moves towards increasing value**"
  - Terminates when a peak is reached.
- The objective function value has to be maximized or minimized
- Hill-climbing does not look ahead of the immediate neighbours.
- Can randomly choose among the best set of successors, if multiple states have the same objective function value.
- "trying to find the top of Mount Everest in a thick fog while suffering from amnesia" – Russel & Norvig
  - Thick Fog - You cannot look beyond your immediate neighbourhood.
  - Amnesia - No memory of how you reached the current state.
  - Find the Top- Because you are choosing the best neighbor at every step.
  - Strict contrast to DFS, BFS, etc.

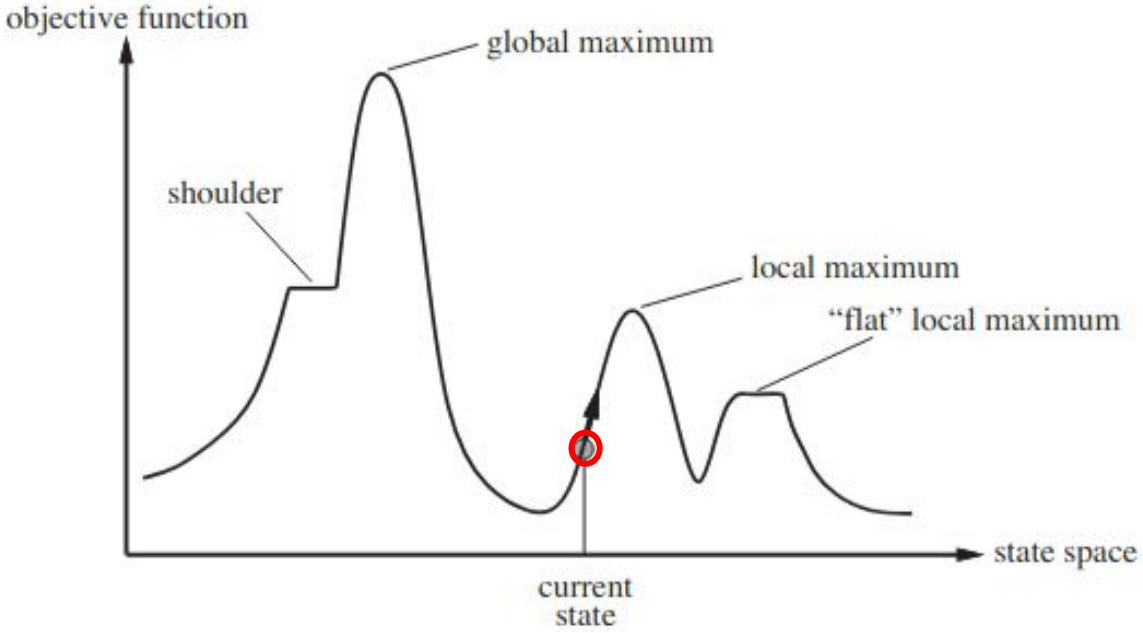
Hill-climbing (Greedy Local Search)

```
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
```

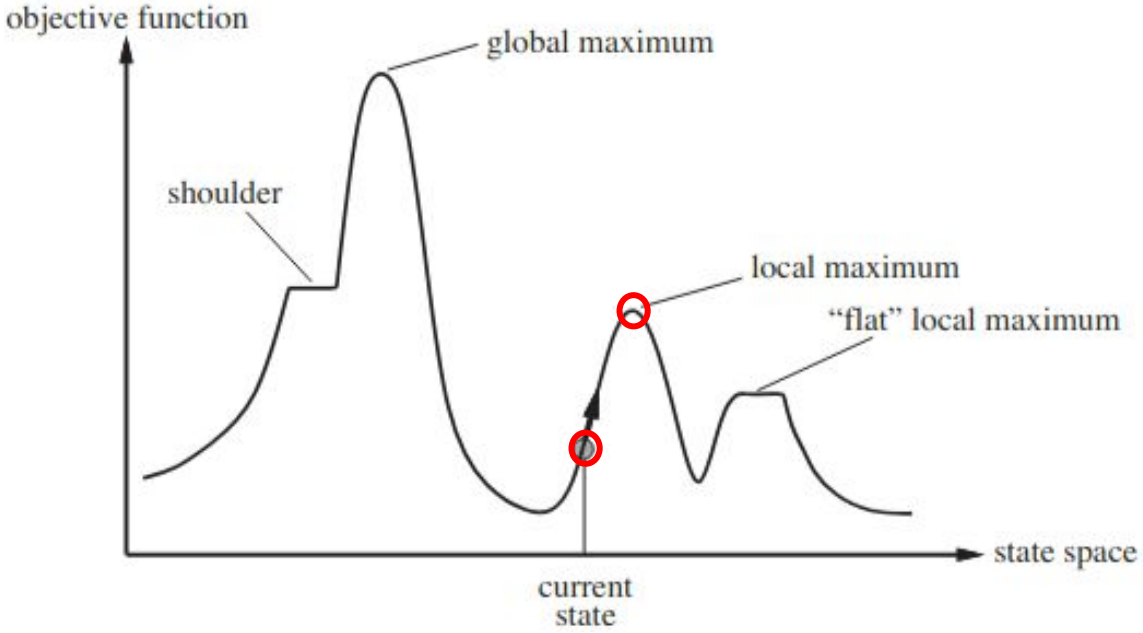
Hill-climbing (Greedy Local Search)

- Is it always possible to reach the global maximum?
    - Depends on where you start.



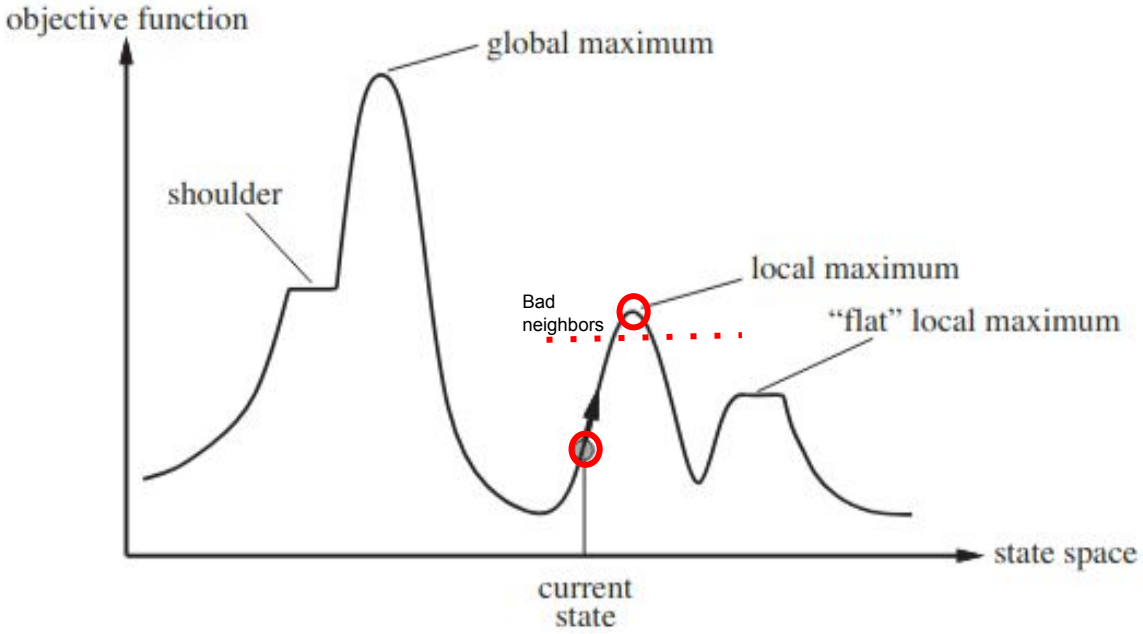
Hill-climbing (Greedy Local Search)

- Is it always possible to reach the global maximum?
    - Depends on where you start.



Hill-climbing (Greedy Local Search)

- Is it always possible to reach the global maximum?
  - Depends on where you start.



Hill Climbing gets stuck at a local maxima (or local minima)

Hill-climbing on 8-queens

- Randomly generated 8-queens starting states…
  - 14% cases the problem is solved.
  - 86% cases get stuck at a local maximum/minimum (depending on whether maximising/minimising)
- Good Aspects:
  - Requires 4 steps on average when it succeeds.
  - And 3 steps on average when it gets stuck.
  - This is significant considering a state space with 8^8 ~ 17 million states.