Local Search

Stephen G. Ware

CSCI 4525 / 5525

Classical vs. Local Search

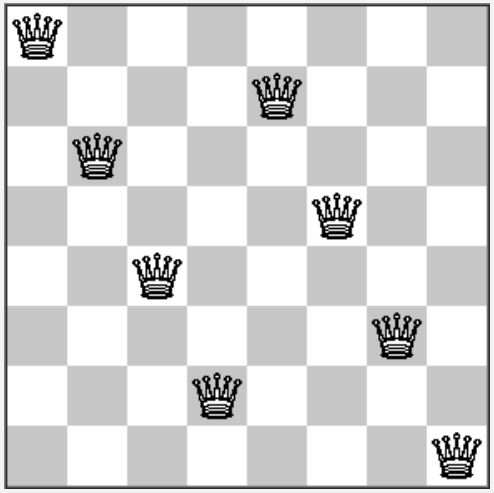
Classical Search:

- Start with an empty path.
- Build the path piece by piece.
- Stop when you find a solution.

Local Search:

- Start with a bad solution.
- Move to a slightly different solution.
- Stop when you find a valid or optimal solution.

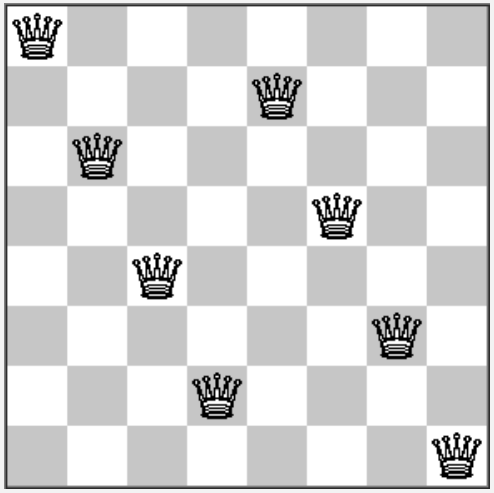
8 Queen Problem



Classical Search:

- Start with an empty board.
- At each step, add a new queen to the board.
- If any pair of queens is attacking, backtrack.
- Stop when 8 queens have been placed.

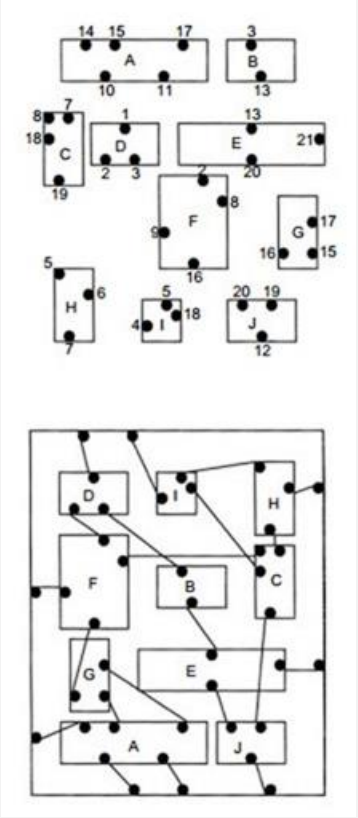
8 Queen Problem



Local Search:

- Start with 8 queens on the board, even if some are attacking.
- At each step, move one of the queens to somewhere else in her column.
- Stop when no pairs are attacking.

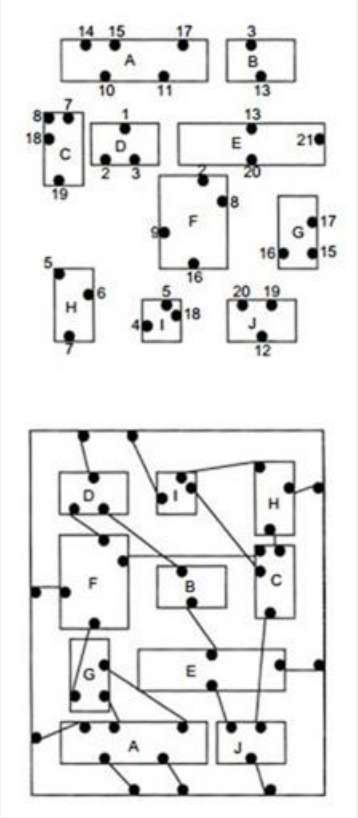
Circuit Layout



Classical Search:

- Start with an empty circuit.
- At each step, add a new piece of the circuit.
- If the new piece violates the constraints, backtrack.
- Stop when all pieces have been added to the circuit.

Circuit Layout



Local Search:

- Start with all piece on the circuit, even if layout is bad.
- At each step, move a piece of the circuit somewhere else.
- Stop when no constraints are violated or an optimal layout has been found.

Classical vs. Local Search

Classical Search:

- Keep all paths in memory.
- At each step, extend one path.
- Uses a lot of memory, and is **systematic**.

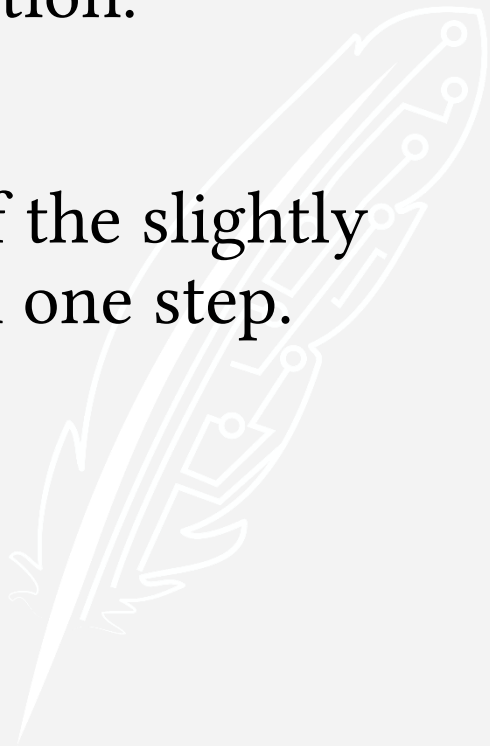
Local Search:

- Keep only the current solution in memory.
- At each step, move to a neighbor of the current solution.
- Uses little memory, but is **not systematic**.

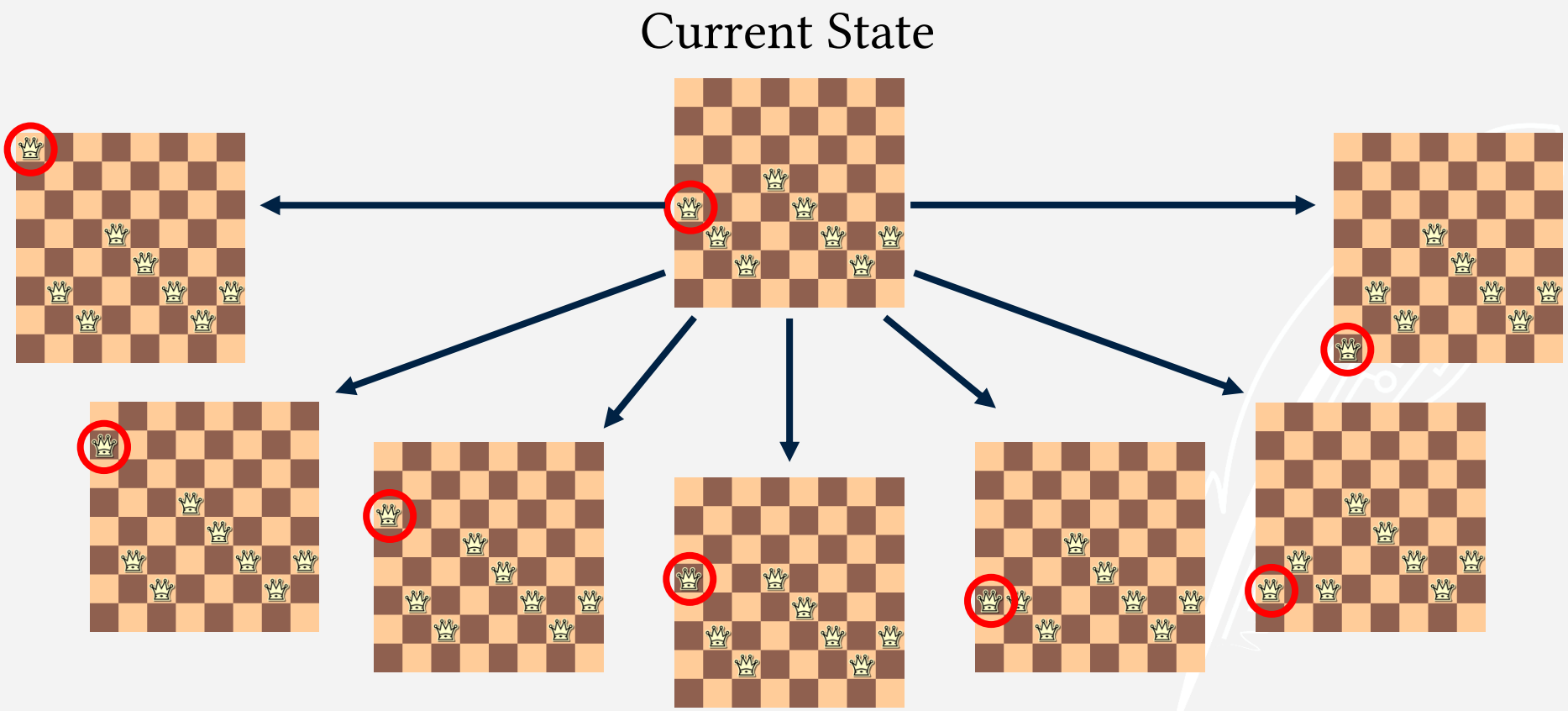
Solution Neighborhood

To use local search, we need to define a way to turn one solution into a slightly different solution.

The **neighborhood** of a solution is all of the slightly different solutions that can be reached in one step.

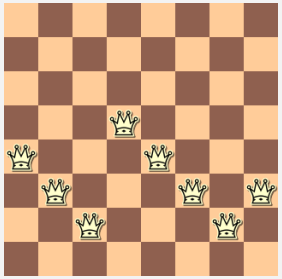


8 Queens Neighborhood



8 Queens Neighborhood

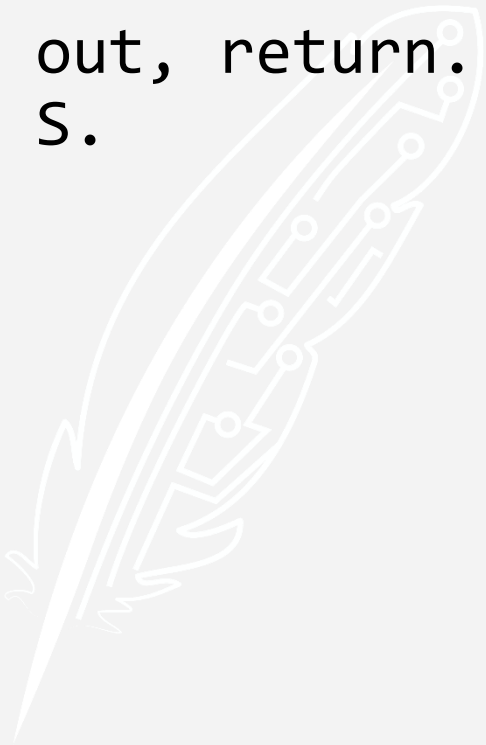
Current State



Each 8 Queens solution always has 56 neighbors.

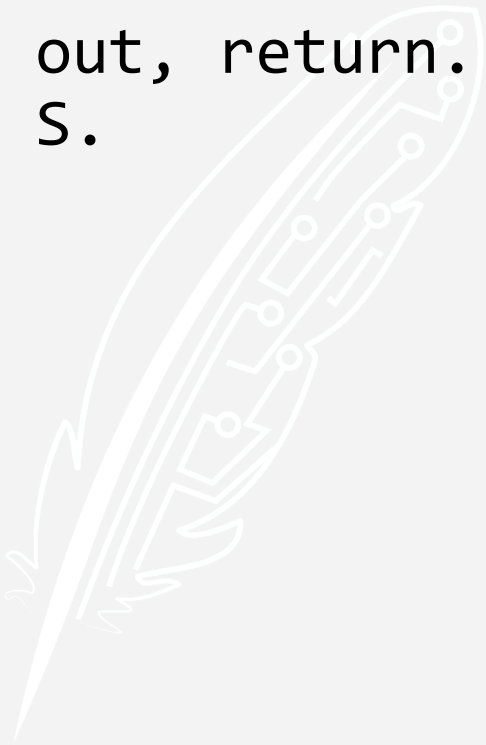
Local Search

```
1. Let S be some initial solution.
2. Loop:
3.   If S is valid or time has run out, return.
4.   Generate all the neighbors of S.
5.   Choose a neighbor N of S.
6.   Replace S with N.
```



Local Search

1. Let S be some initial solution.
2. Loop:
3. If S is valid or time has run out, return.
4. Generate all the neighbors of S.
5. **Choose a neighbor N of S.**
6. Replace S with N.



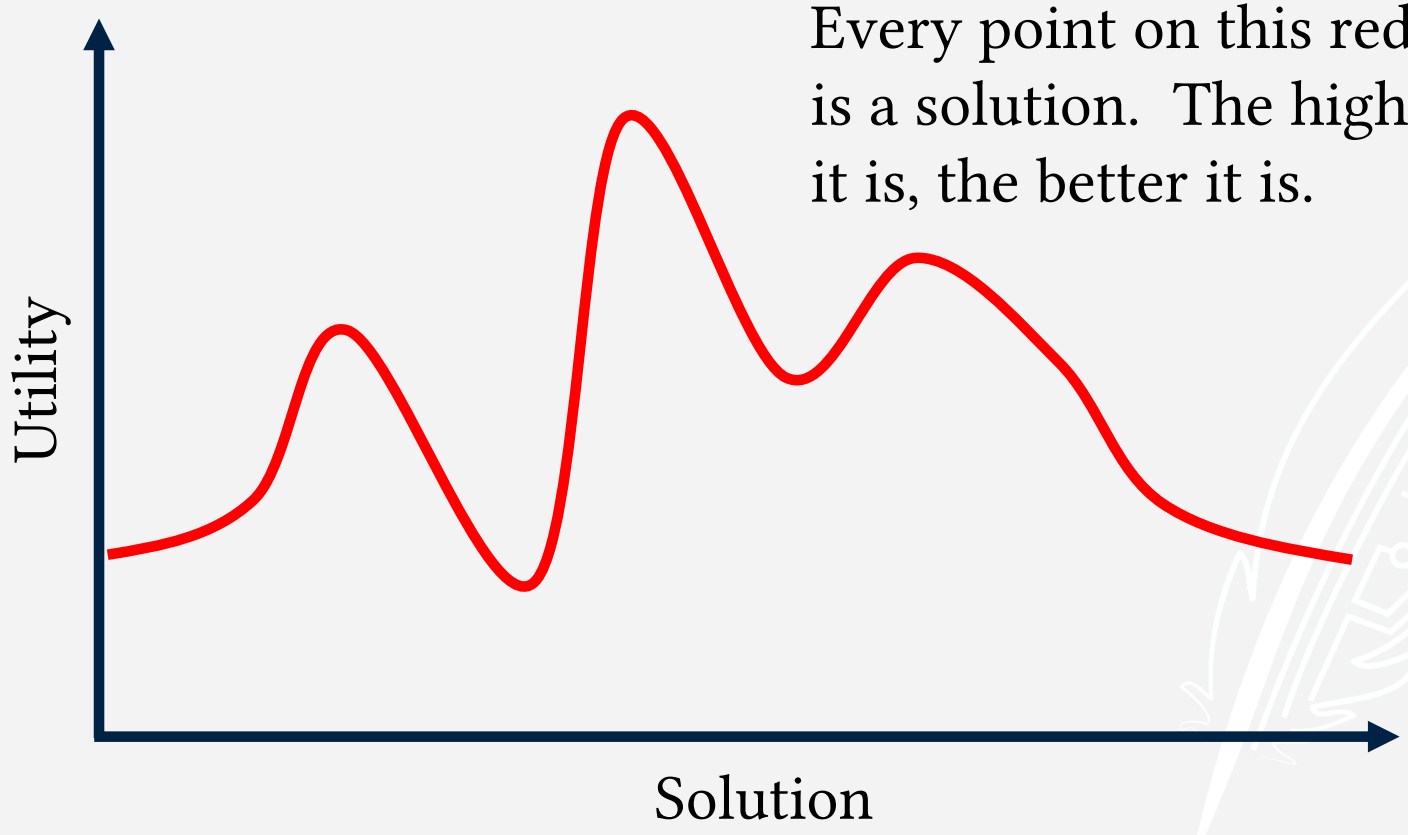
Types of Local Search

- Greedy Local Search
- Random Restarts
- Simulated Annealing

Greedy Local Search

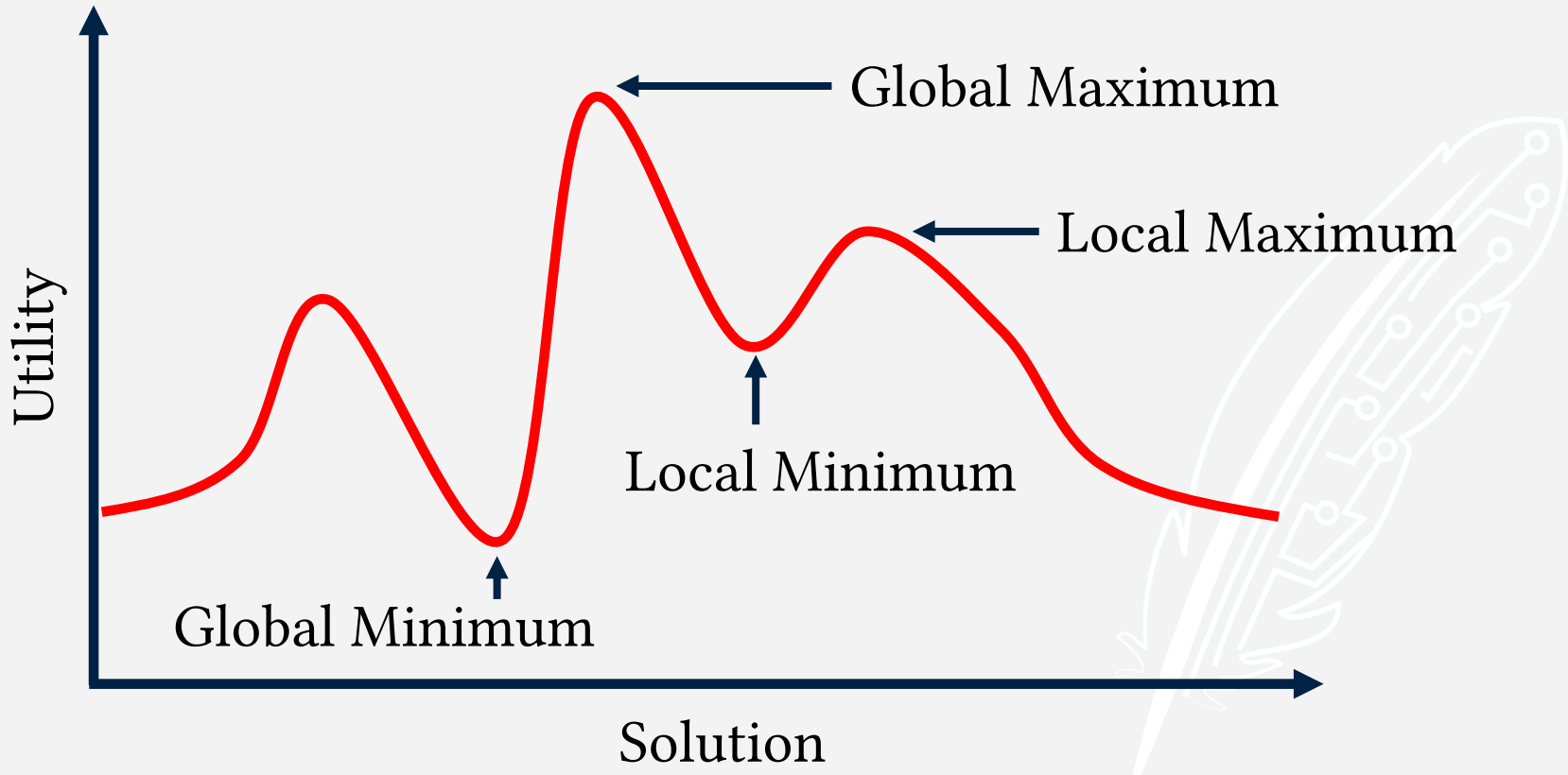
When choosing a neighbor, always choose the neighbor which has the highest utility.

Hillclimbing Problem

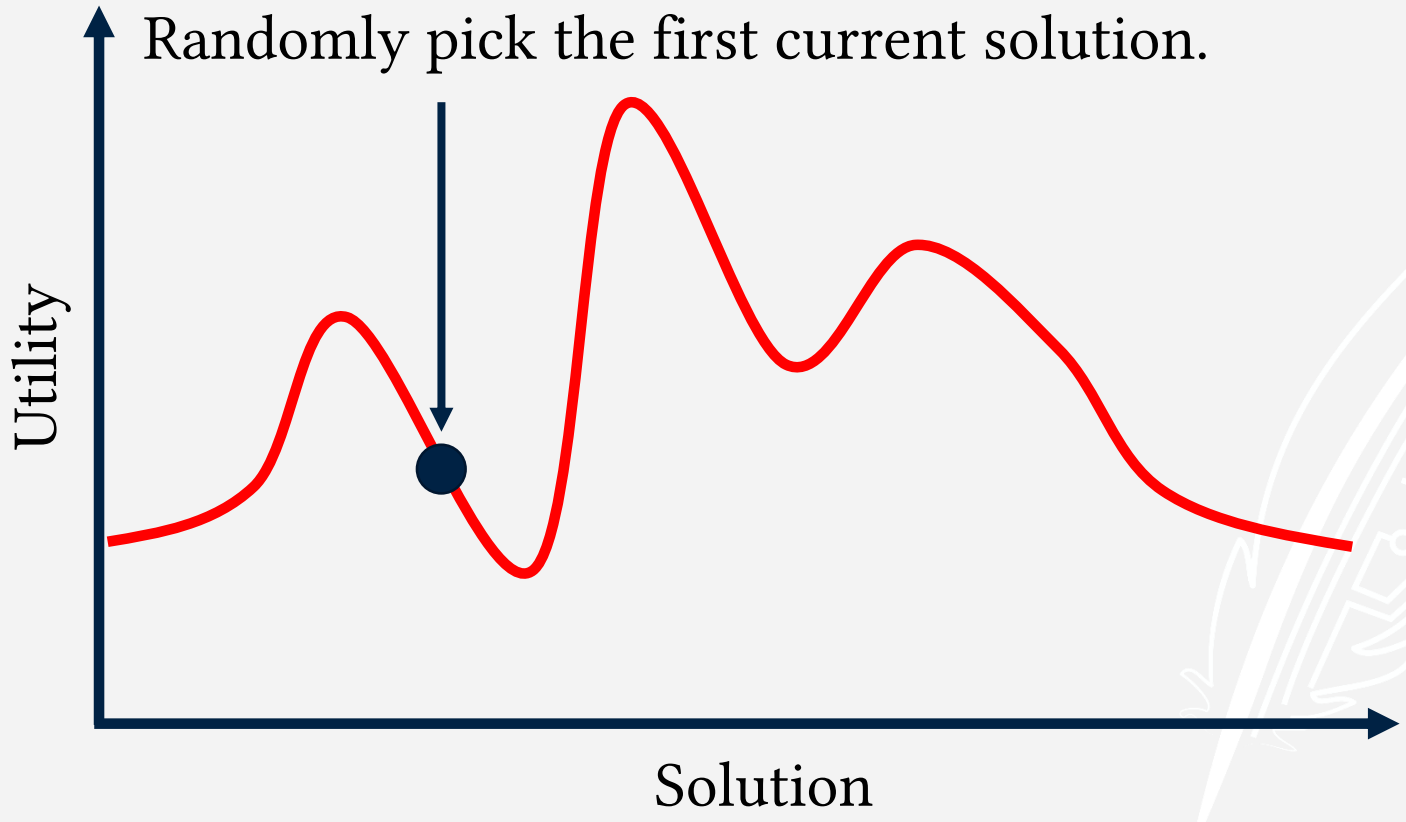


Every point on this red line is a solution. The higher up it is, the better it is.

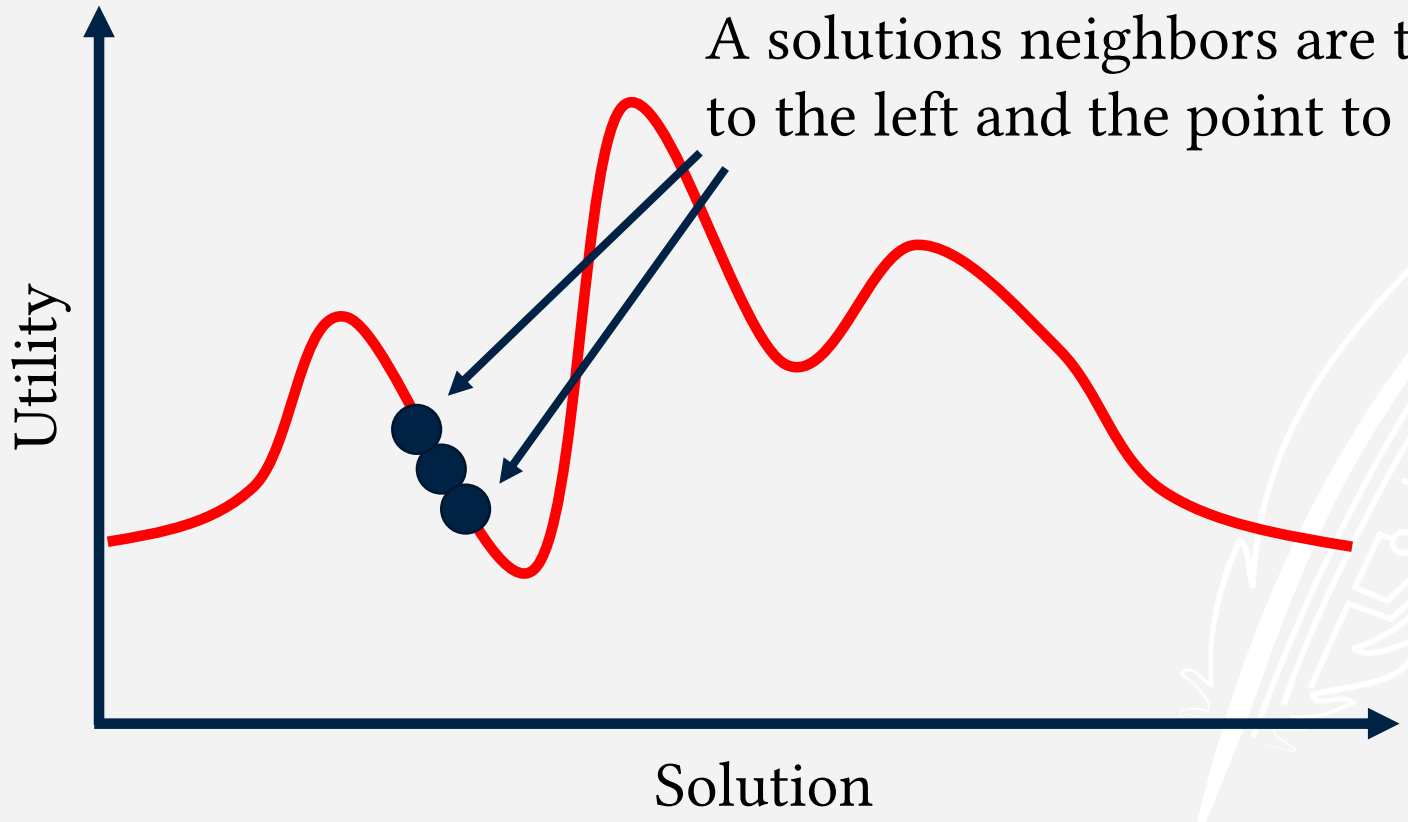
Hillclimbing Problem



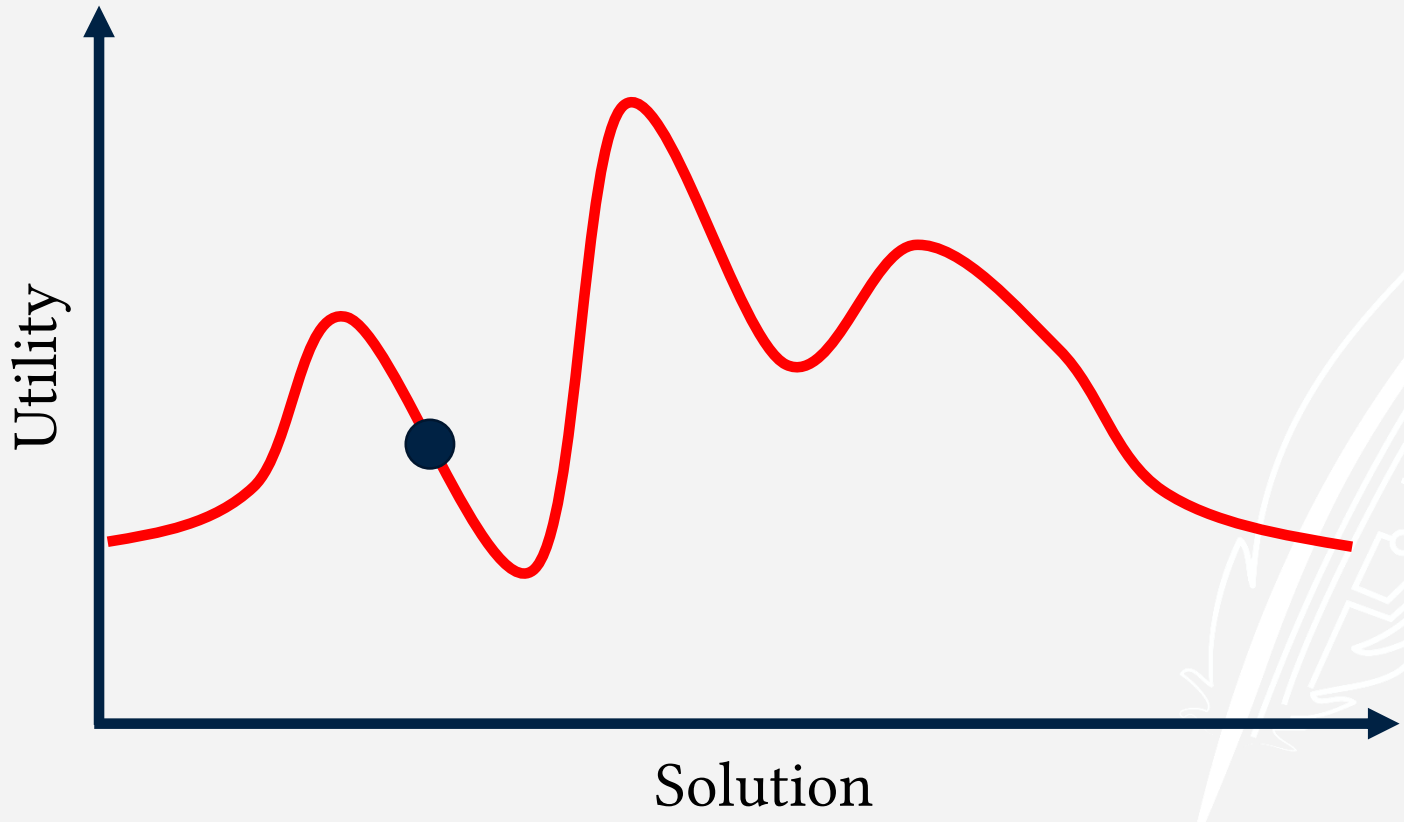
Hillclimbing Problem



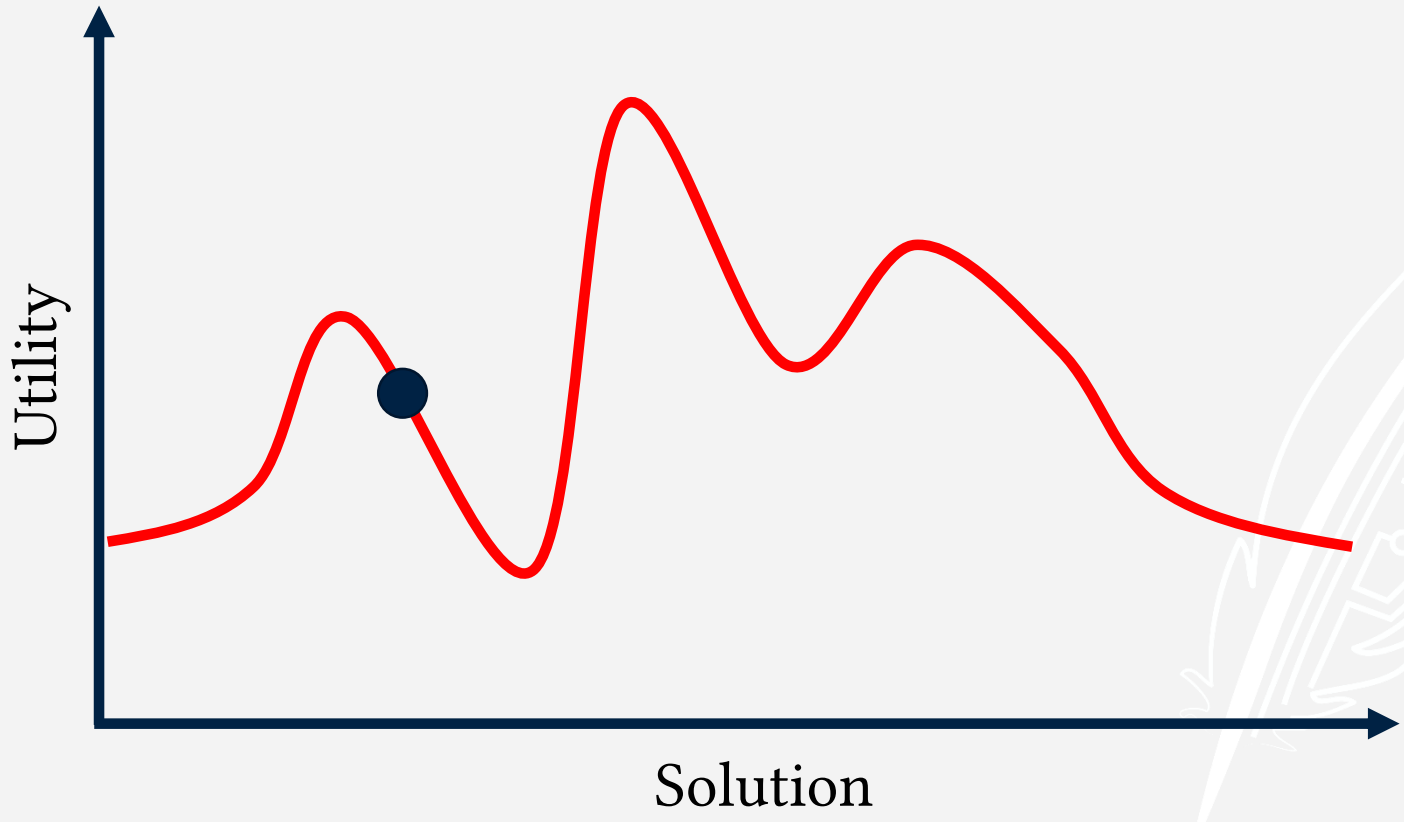
Hillclimbing Problem



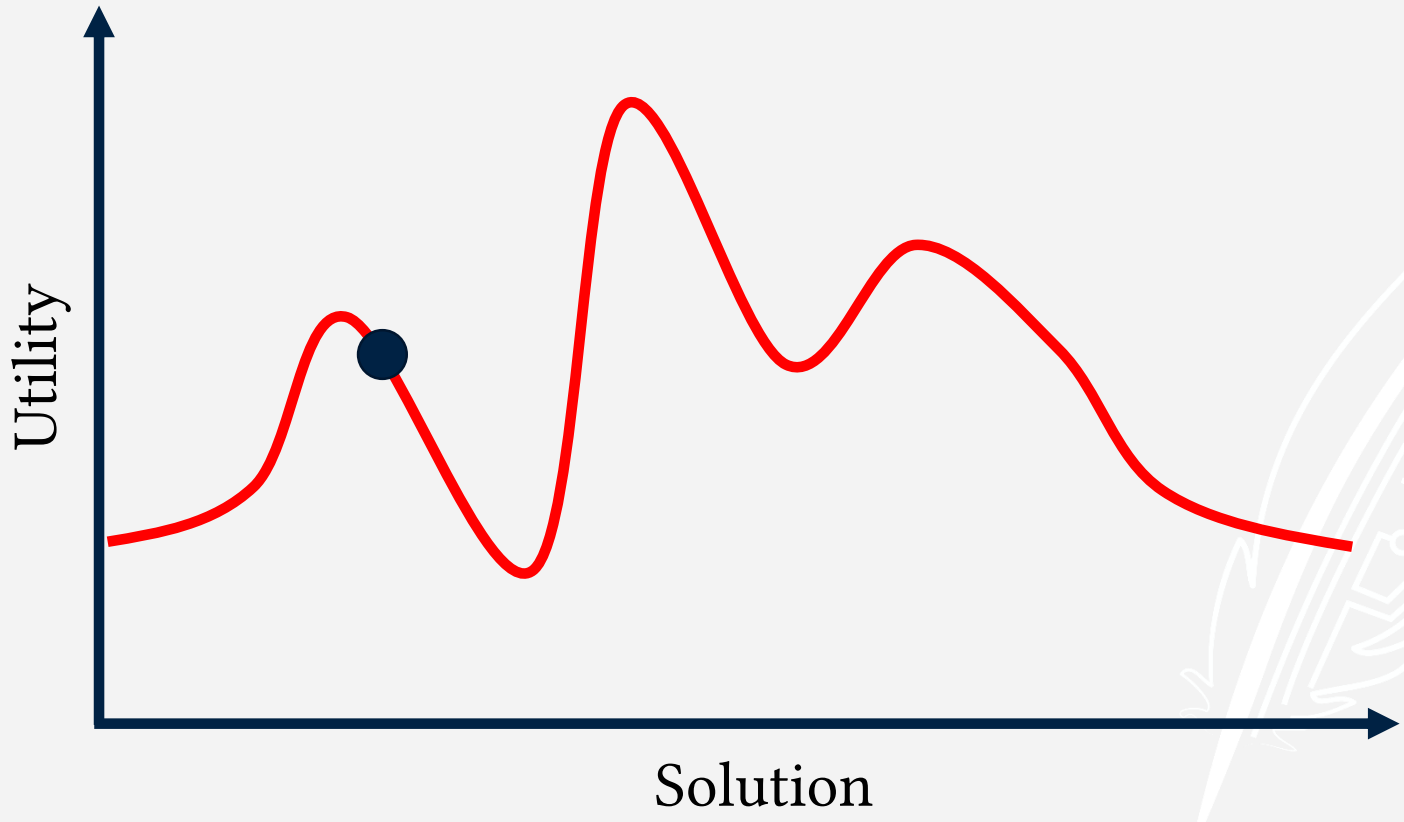
Hillclimbing Problem



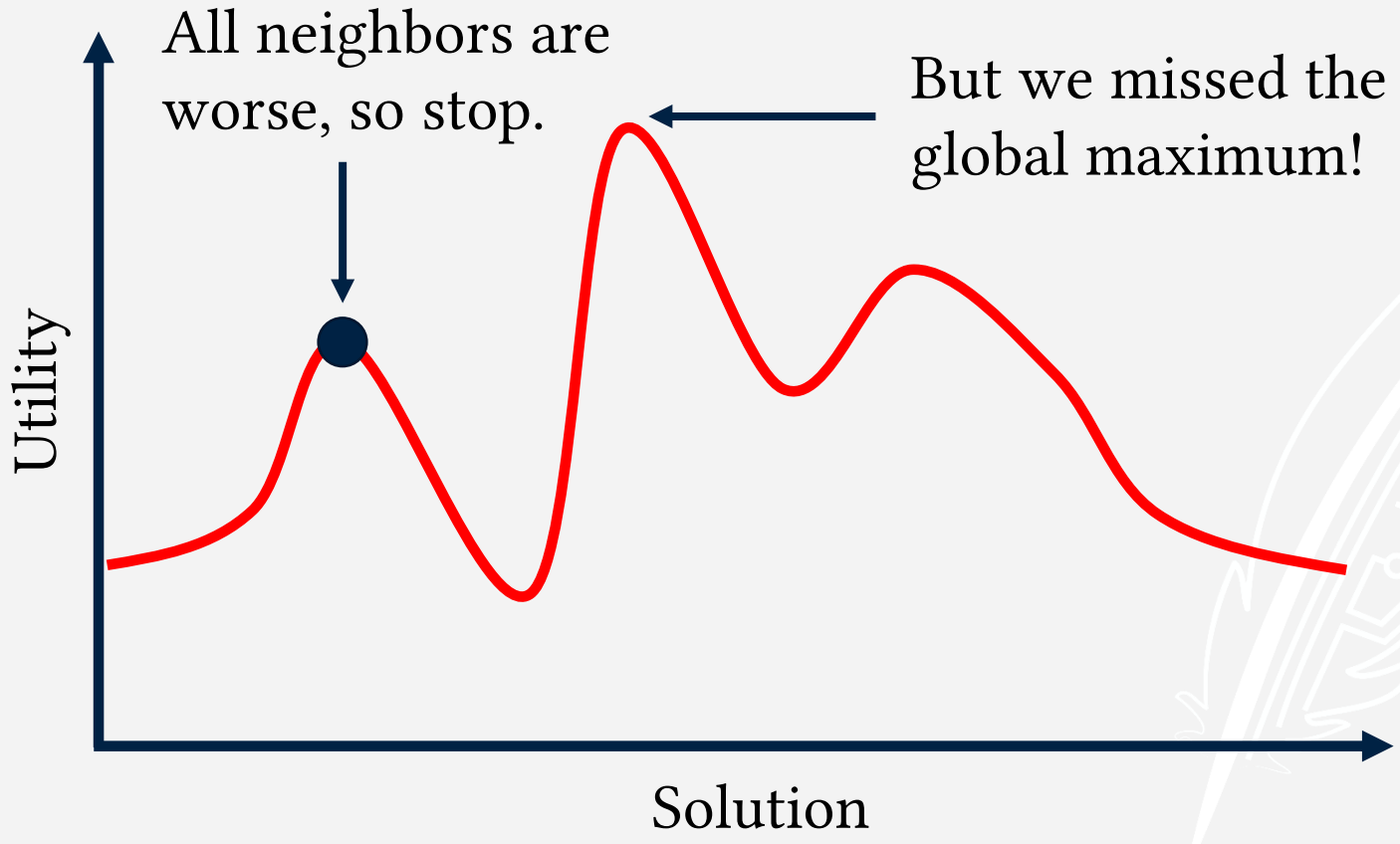
Hillclimbing Problem



Hillclimbing Problem



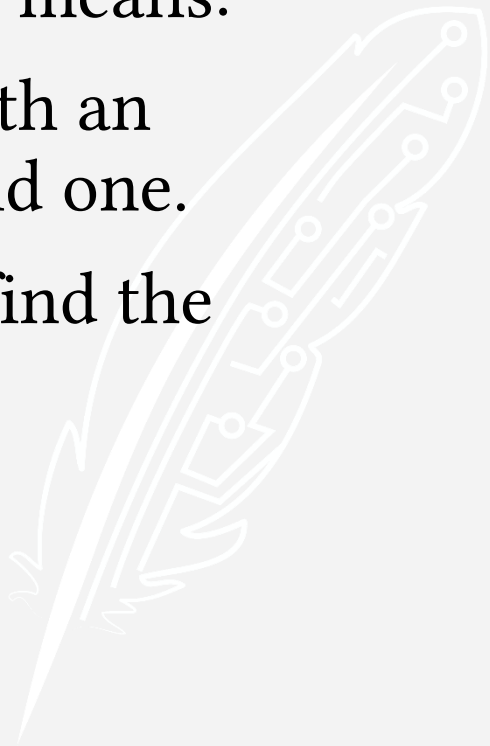
Hillclimbing Problem



Hillclimbing Problem

When local search only chooses better neighbors, it can get stuck in a local maximum, which means:

- It is **incomplete** because if we start with an invalid solution it may never find a valid one.
- It is **suboptimal** because it might not find the global maximum.



Random Restarts

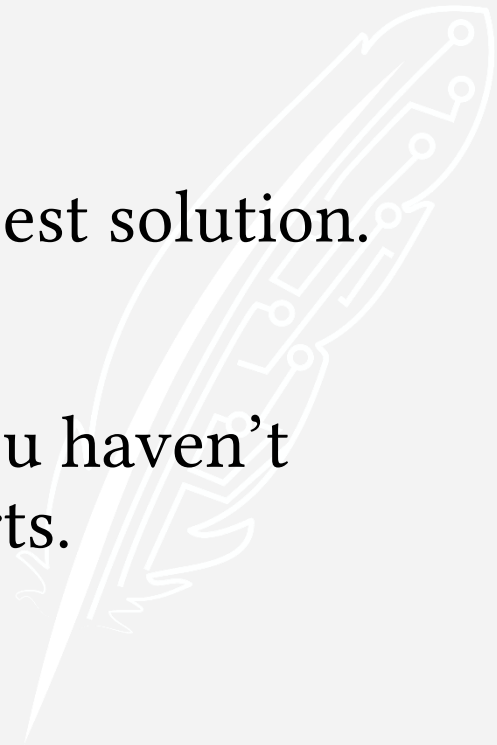
Once you find the first solution, record it as the “best solution so far.”

Restart the search from a random place.

If a better solution is found, update the best solution.

Each time a solution is found, restart.

Keep restarting until time runs out or you haven’t found a better solution after many restarts.



Annealing

Metal is made up of tiny crystals. When these crystals are in a low energy state, the metal is more ductile and easier to work with.

To anneal a metal, you heat it up until the crystals break down. Then you let it cool slowly, allowing the crystals to reform in a low energy state.

Simulated Annealing

Don't always choose the best next solution.

Choose a random next solution based on the current *temperature*.

Temperature starts high and gets low over time. That means you have a higher chance to make a suboptimal move early on, and a lower chance to make a suboptimal move later on.