COL333/671: Introduction to AI
Semester I, 2022-23

Local Search Algorithms

Rohan Paul

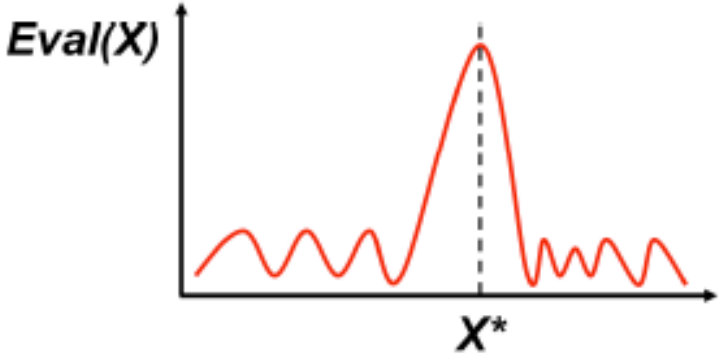
Outline

- Last Class
  - Informed Search
- This Class
  - Local Search Algorithms
- Reference Material
  - AIMA Ch. 4.1

Acknowledgement

These slides are intended for teaching purposes only. Some material has been used/adapted from web sources and from slides by Doina Precup, Dorsa Sadigh, Percy Liang, Mausam, Dan Klein, Nicholas Roy and others.

Search Methods for Discrete Optimization



- **Setting**
  - A set of discrete states, *X*.
  - An objective/evaluation function assigns a "goodness" value to a state, *Eval(X)*
  - Problem is to search the state space for the state, *X** that maximizes the objective.

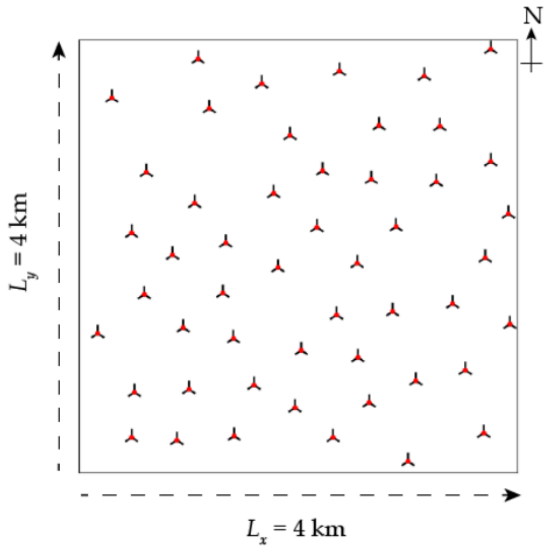
- **Searching for the optimal solution can be challenging. Why?**
  - The number of states is very large.
    - Cannot simply enumerate all states and find the optimal.
  - We can only evaluate the function.
    - Cannot write it down analytically and optimize it directly.

Key Idea
- **Searching for "the optimal" solution is very difficult.**
- **Question is whether we can search for a reasonably good solution.**

Example

Problem: Optimizing the locations of windmills in a wind farm

- An area to place windmills.
- Location of windmills affects the others. Reduced efficiency for those in the wake of others.
- Grid the area into bins.
- A large number of configurations of windmills possible.
- Given a configuration we can evaluate the total efficiency of the farm.
- Can neither enumerate all configurations nor optimize the power efficiency function analytically.
- **Goal is to <u>search</u> for the <u>configuration</u> that maximizes the efficiency.**



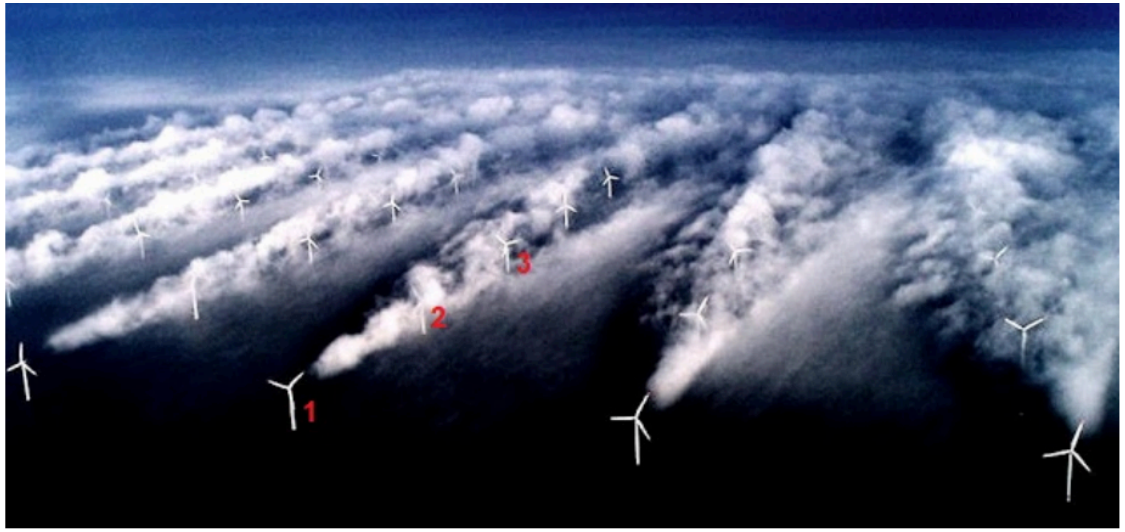


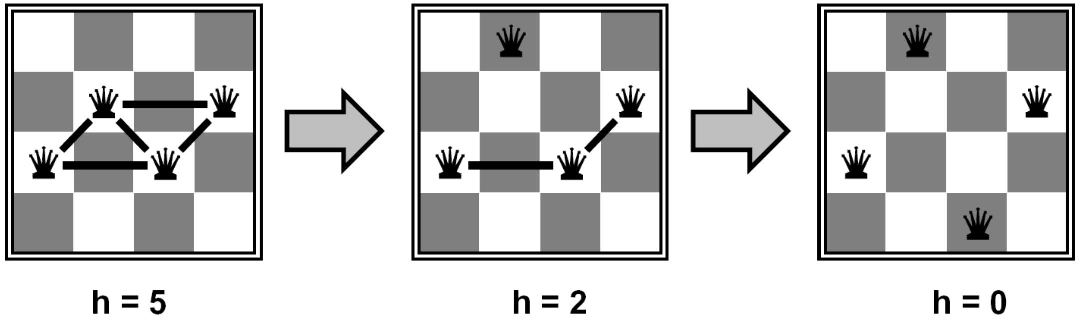
Figure 5: Turbines experiencing multiple wakes. As an example, turbine 3 is experiencing wake effects from both turbine 1 and 2. Image adopted from [4].

Inspired from this example: https://www.shell.in/energy-and-innovation/ai-hackathon/_jcr_content/par/textimage_1834506119_1619963074.stream/1612943059963/4b0a86b7cc0fe7179148284ffed9ef33524c2816/windfarm-layout-optimisation-challenge.pdf

Example

4-Queens Problem

- Discrete set of states: 4 queens in 4 columns (4^4 = 256 states)
- Goal is to find a configuration such that there are no attacks.
  - Moving a piece will change the configuration.
- Any configuration can be evaluated using a function
  - h(x) = number of attacks (number of violated binary constraints)
- Search for the configuration that is optimal such that h = 0.



Slide adapted from from Dan Klein and Anca Dragan

Local Search Methods

- **Keep track of a single "current" state**
  - We need a principled way to search/explore the state space hoping to find the state with the optimal evaluation.
  - Do not maintain a search tree as we need the solution not the path that led to the solution.
  - Only maintain a single current state.

- **Perform local improvements**
  - Look for alternatives in the vicinity of that solution
  - Try to move towards more better solutions.

Hill-climbing Search

Let S be the start node and let G be the goal node.

Let h(c) be a heuristic function giving the value of a node

Let c be the start node

Loop

    Let c' = the highest valued neighbor of c

    If h(c) ≥ h(c') then return c

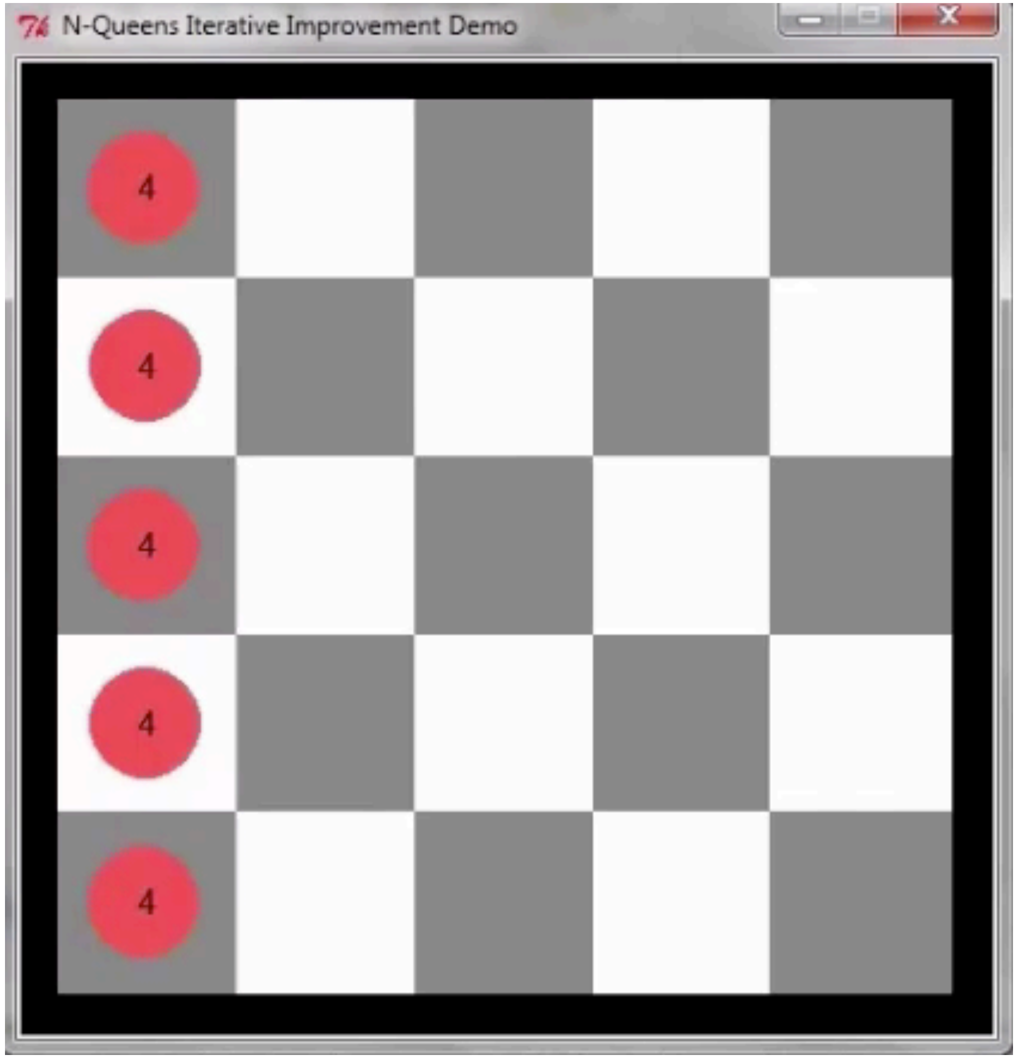
    c = c'

Hill climbing

Start at a configuration. Evaluate the neighbors. Move to the highest valued neighbor if its value is higher than the current state. Else stay.

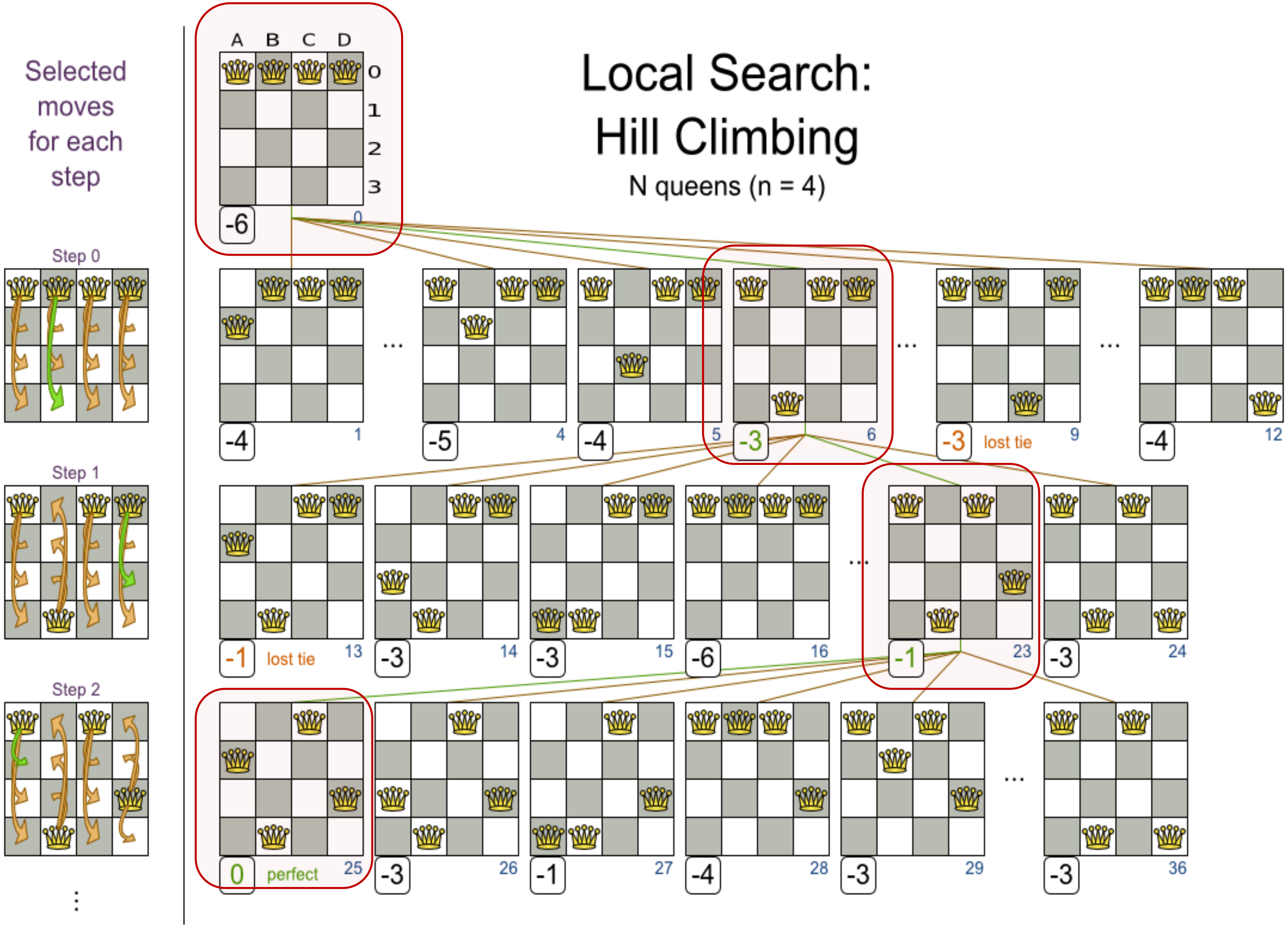
Hill climbing for 4 -queens

- Select a column and move the queen to the square with the fewest conflicts.
- Perform local modifications to the state by changing the position of one piece till the evaluation is minimum.
- Evaluate the possibilities from a state and then jump to that state.



Example

- Local search looks at a state and its local neighborhood.
- Not constructing the entire search tree.
- Consider local modifications to the state. Immediately jump to the next promising neighbor state. Then start again.
- *Highly scalable.*

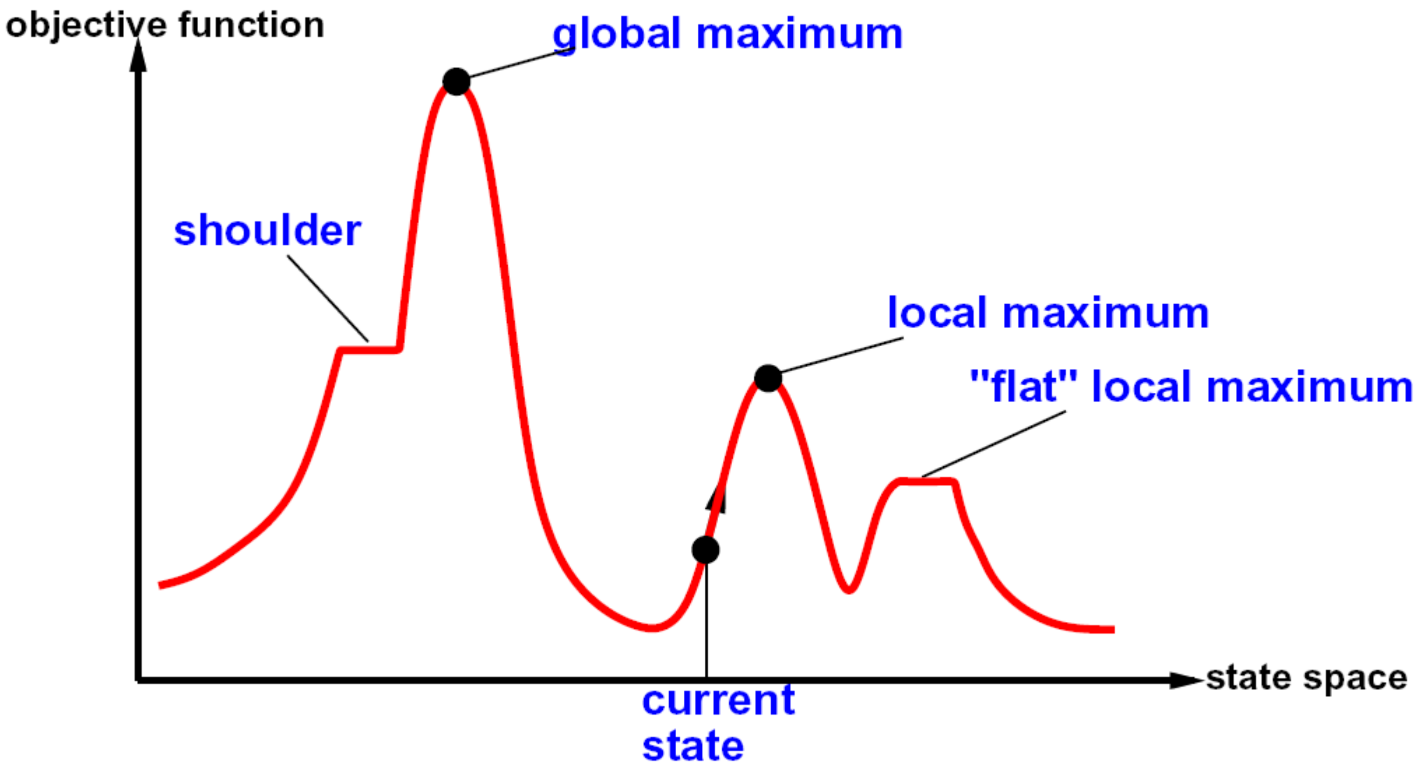


Problem with hill climbing

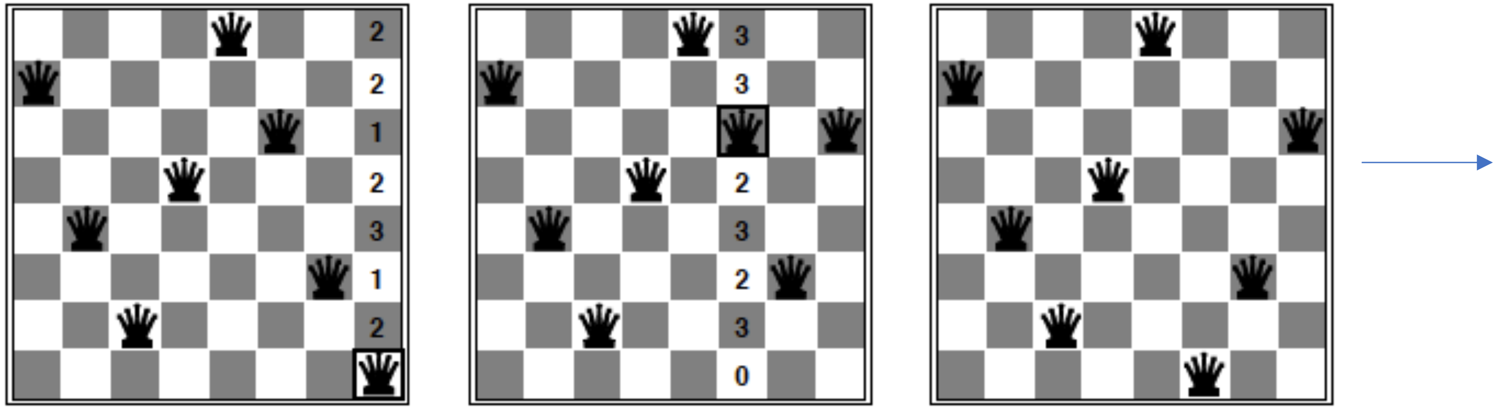
Hill climbing can get stuck in the local maxima.

Why? The neighbors may not be of higher value. The search will stay at the current state for a long time.

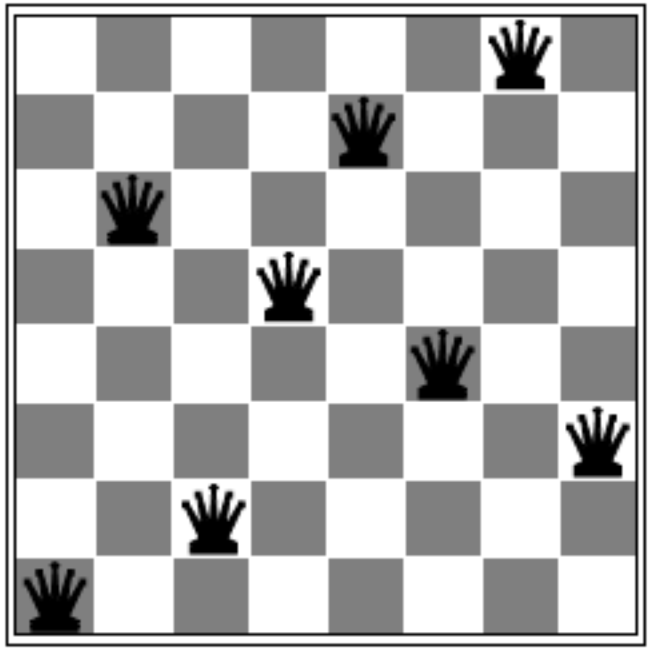
Optimization Landscape



Example: 8 Queens Problem



Local minima (h = 1). Every successor has a higher cost.



Improvements

- Random Re-start
  - A series of searches from randomly generated initial states.
  - Escape a plateau or local minimum.
- *Stochastic* Hill Climbing
  - Instead of picking the *best move,* pick *any* move that produces an *improvement*.
  - Probability: steepness of the uphill move.
  - Introduce randomness.

How to escape local minima?
- One way is to pick “bad” moves.

Simulated Annealing

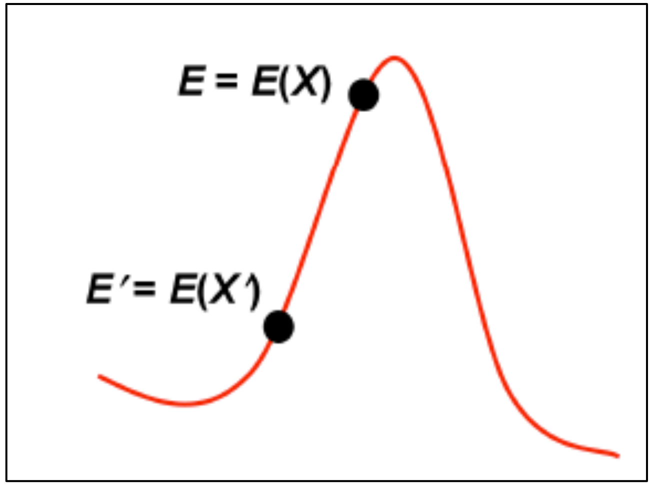
- Allows some apparently **bad moves** - to escape local maxima.
- **Decrease** the size and the frequency of bad moves over time.

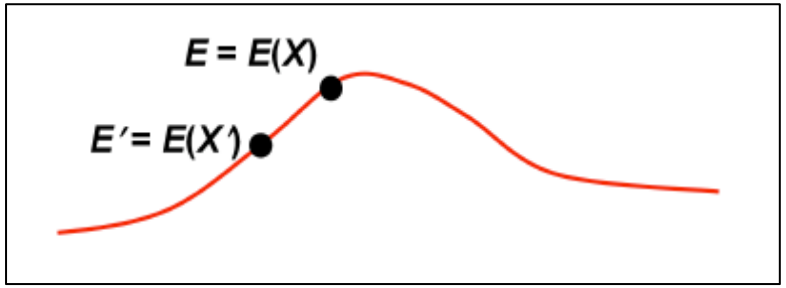
- Algorithm sketch
  1. Start at initial configuration X of value E (high is good)
  2. Repeat:
     (a) Let X_i be a *random neighbor* of X and E_i be its value
     (b) If $E < E_i$ then let $X \leftarrow X_i$ and $E \leftarrow E_i$
     (c) Else, *with some probability* p, still accept the move: $X \leftarrow X_i$ and $E \leftarrow E_i$
- Best solution ever found is always remembered

A form of Monte-Carlo Search. Move around the environment to explore it instead of systematically sweeping. Powerful technique for large domains.

Simulated Annealing: How to decide *p*?

- Considering a move from state of value E to a lower valued state of E’. *That is considering a sub-optimal move (E is higher than E’)*.
- If (E – E’) is large:
  - Likely to be close to a promising maximum.
  - Less inclined to to go downhill.
- If (E – E’) is small:
  - The closest maximum may be shallow
  - More inclined to go downhill is not as bad.





Simulated Annealing: Selecting Moves

- If the new value E_i is ***better*** than the old value E, move to X_i

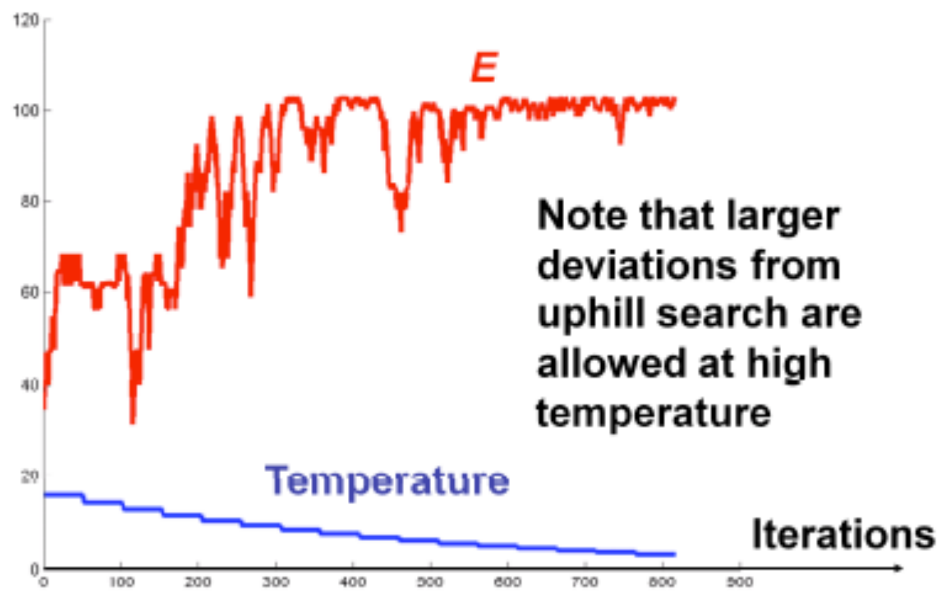
- If the new value is ***worse*** ($E_i < E$) then move to the neighboring solution as per *Boltzmann* distribution.

$$\exp\left(-\frac{E - E_i}{T}\right)$$

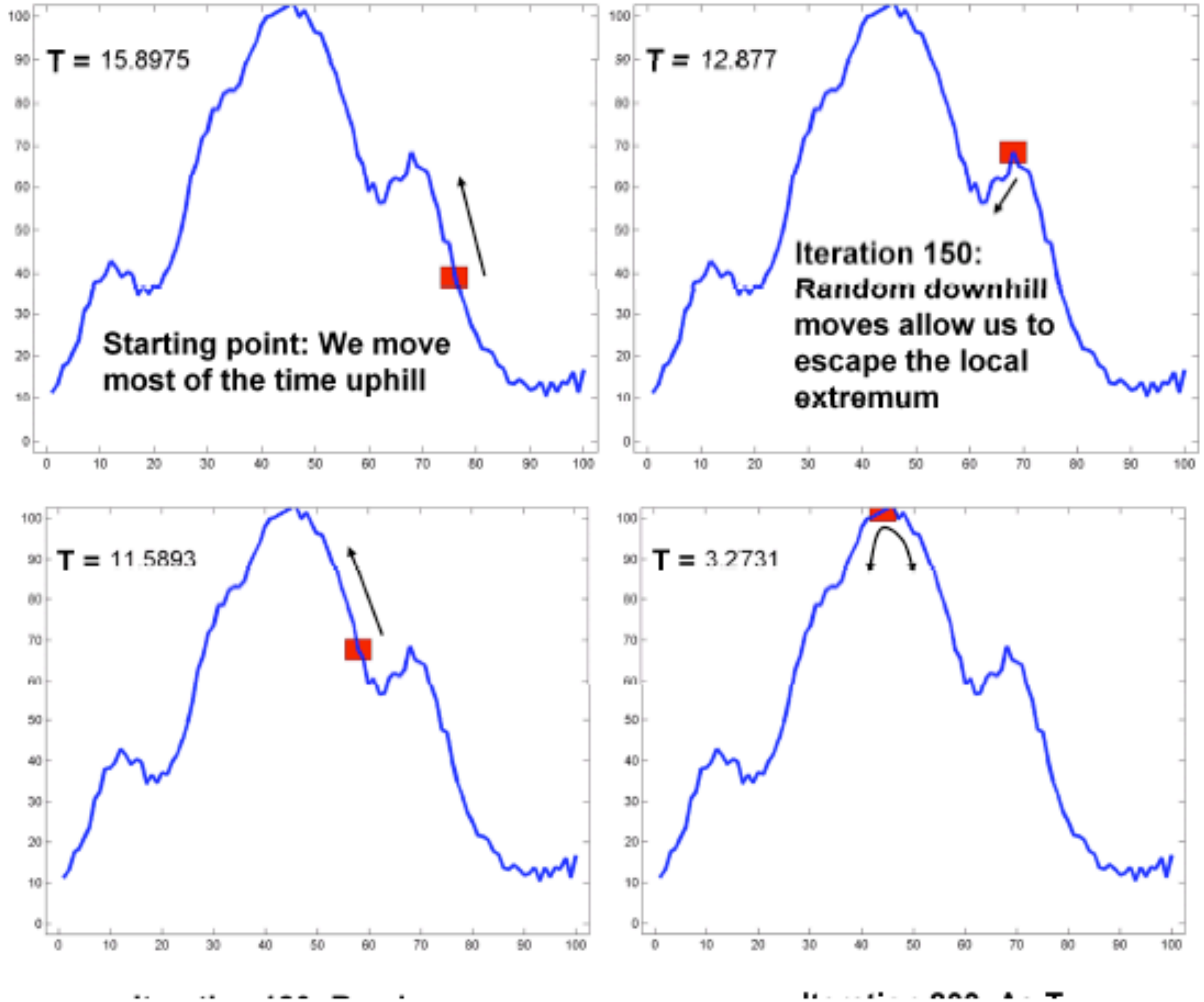
- Temperature (T>0)
  - **T is high**, exp is ~0, acceptance probability is ~1, high probability of acceptance of a worse solution.
  - **T is low**, the probability of moving to a worse solution is ~ 0, low probability of acceptance of a worse solution.
  - Schedule T to reduce over time.

Simulated Annealing: Properties

- **T is high**
  - The algorithm is in an *exploratory* phase
  - Even bad moves have a high chance of being picked
- **T is low**
  - The algorithm is in an *exploitation* phase
  - The “bad” moves have very low probability
- **If T is decreased slowly enough**
  - Simulated annealing is guaranteed to reach the best solution in the limit.



Simulated Annealing: Example



Able to escape local maxima.

Local Beam Search

- **Look for solutions from multiple points in parallel.**
- Algorithm
  - Track k states (rather than 1).
  - Begin with k randomly sampled states.
  - Loop
    - Generate successors of each of the k-states
    - If anyone has the goal, the algorithm halts
    - Otherwise, select only the k-best successors from the list and repeat.
  - Note:
    - Each run is not independent, information is passed between parallel search threads.
    - Promising states are propagated. Less promising states are not propagated.
    - Problem: states become concentrated in a small region of space.

Stochastic Beam Search

- Local beam search
  - Problem: states become concentrated in a small region of space
  - Search degenerates to hill climbing
- Stochastic beam search
  - Instead of taking the best k states
  - Sample k states from a distribution
  - Probability of selecting a state *increases* as the *value* of the state.

Summary

This Module

- Local Search

Next Module

- Variable-based models