CS 540 Introduction to Artificial Intelligence
Advanced Search

Yingyu Liang
University of Wisconsin-Madison

Nov 18, 2021

Based on slides by Fred Sala

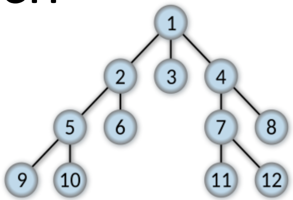
Outline

- Advanced Search & Hill-climbing
  - More difficult problems, basics, local optima, variations
- Simulated Annealing
  - Basic algorithm, temperature, tradeoffs
- Genetic Algorithms
  - Basics of evolution, fitness, natural selection

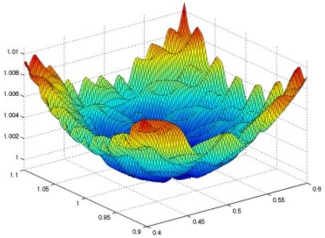
Search vs. Optimization

Before: wanted a **path** from start state to goal state

- Uninformed search, informed search



Wiki



TuringFin

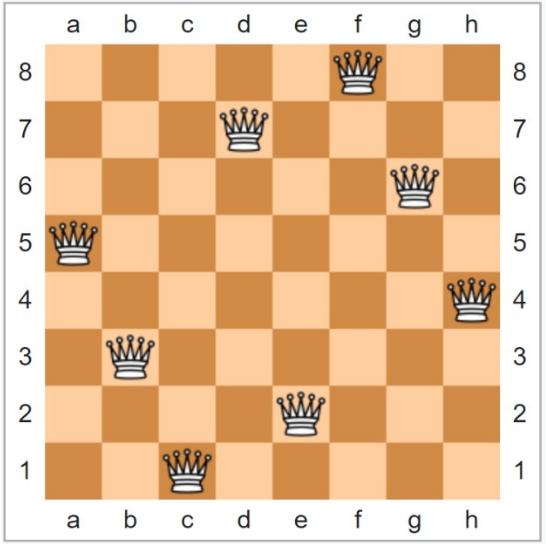
New setting: optimization

- States *s* have values ***f***(*s*)
- Want: *s* with optimal value ***f***(*s*) (i.e, **optimize** over states)
- Challenging setting: **too many states** for previous search approaches, but maybe not a continuous function for SGD.

Examples: n Queens

A classic puzzle:

- Place 8 queens on 8 x 8 chessboard so that no two have same row, column, or diagonal.
- Can generalize to n x n chessboard.
- What are states *s*? Values ***f***(*s*)?
  - State: configuration of the board
  - ***f***(*s*): # of non-conflicting queens



Wiki

Hill Climbing

One approach to such optimization problems

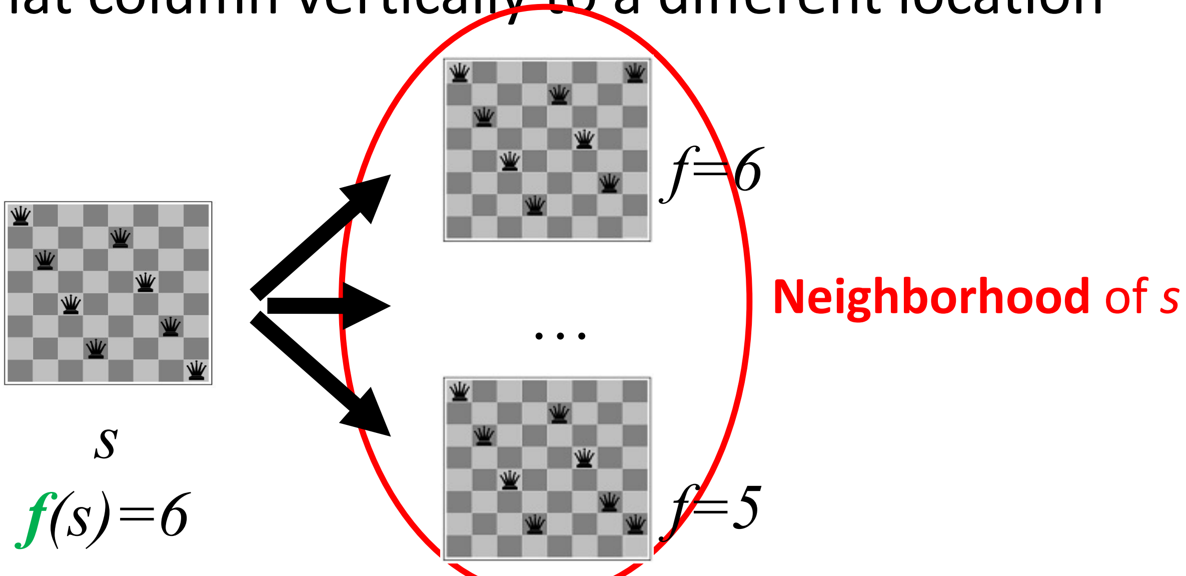
- Basic idea: move to a neighbor with a better $f(s)$

- **Q**: how do we define **neighbor**?
  - Not as obvious as our successors in search
  - Problem-specific
  - As we'll see, needs a careful choice

Defining Neighbors: n Queens

In n Queens, a simple possibility:

- Look at the **most-conflicting column** (ties? right-most one)
- Move queen in that column vertically to a different location



Hill Climbing Neighbors

Q: What's a **neighbor?**

- **Vague definition**. For a given problem structure, neighbors are states that can be produced by a small change
- **Tradeoff**!
  - Too small? Will get struck.
  - Too big? Not very efficient

- **Q**: how to pick a neighbor? Greedy
- **Q**: terminate? When no neighbor has better value

Hill Climbing Algorithm

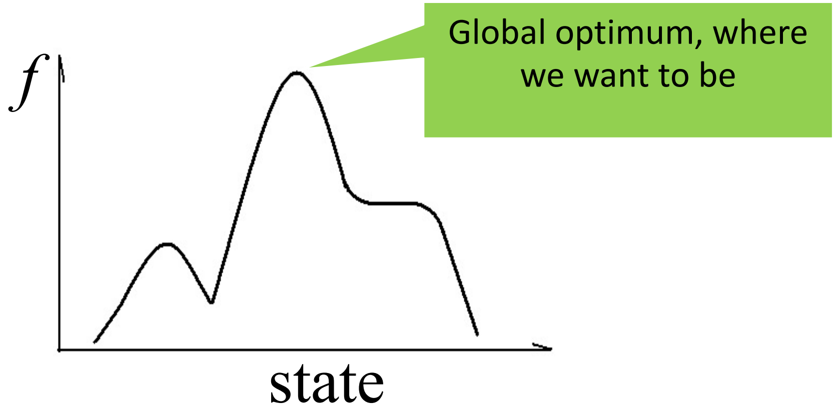
Pseudocode:

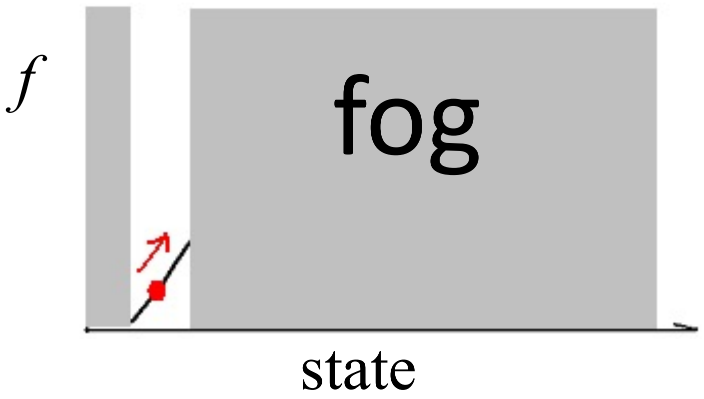
1. Pick initial state *s*
2. Pick *t* in **neighbors**(*s*) with the best ***f***(*t*)
3. if ***f***(*t*) is not better than ***f***(*s*) THEN stop, return *s*
4. *s* ← *t*. goto 2.

What could happen? **Local optima!**

Hill Climbing: Local Optima

Q: Why is it called hill climbing?



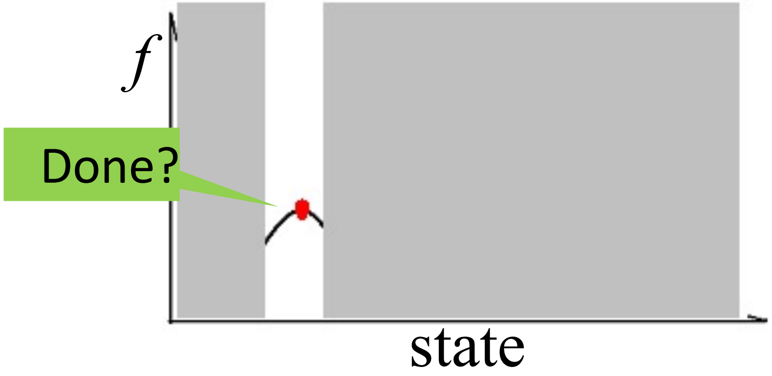


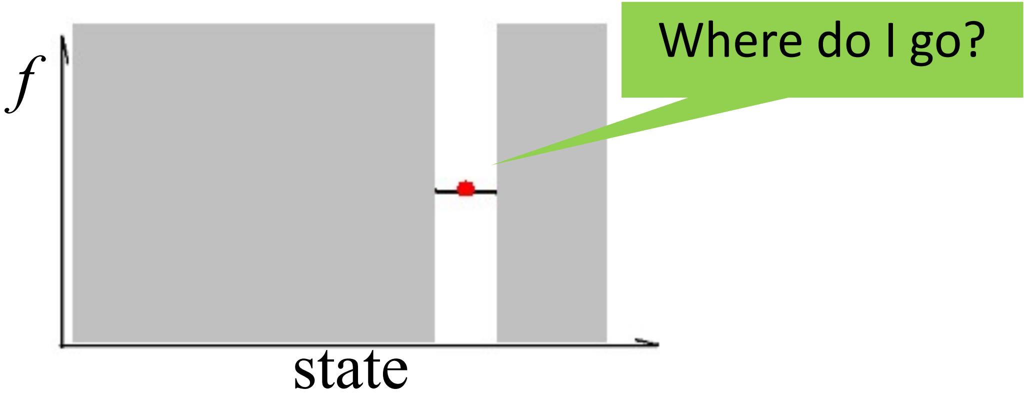
L: What's actually going on.

R: What we get to see.

Hill Climbing: Local Optima

Note the **local optima**. How do we handle them?





Escaping Local Optima

Simple idea 1: random restarts

- Stuck: pick a random new starting point, re-run.
- Do *k* times, return best of the *k* runs

Simple idea 2: reduce greed

- “Stochastic” hill climbing: randomly select between neighbors
- Probability proportional to the value of neighbors

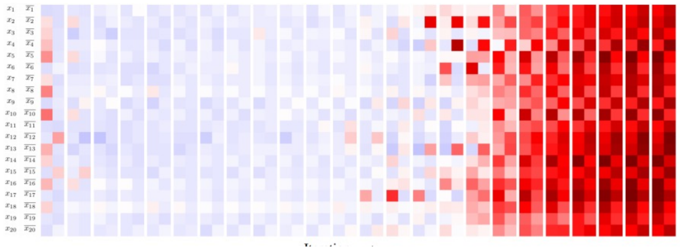
Hill Climbing: Variations

Q: neighborhood too large?

- Generate random neighbors, **one at a time**. Take the better one.

Q: relax requirement to always go up?

- Often useful for harder problems

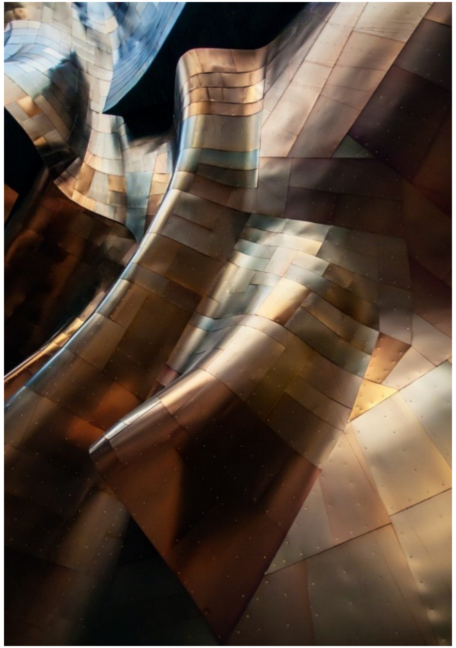


D. Selsam

Simulated Annealing

A more sophisticated optimization approach

- **Idea**: move quickly at first, then slow down
- Pseudocode:



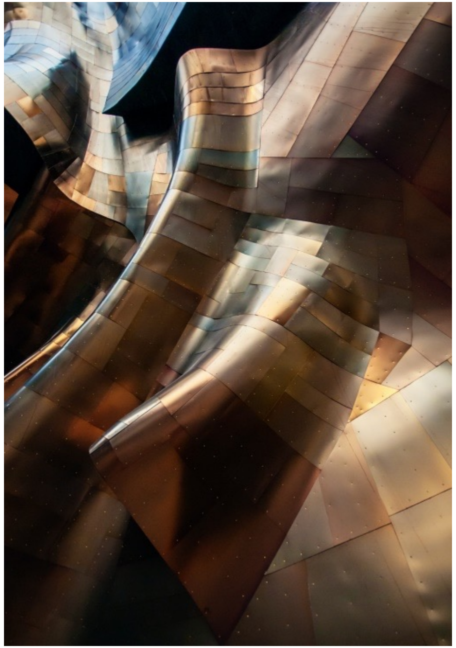
Pick initial state s
For k = 0 through k_{max}:
    $\boldsymbol{T}$ ← temperature($(k+1)/k_{max}$)
    Pick a random neighbor, t ← neighbor(s)
    If $\boldsymbol{f}$(t) better than $\boldsymbol{f}$(s), then s ← t
    Else, with prob. P($\boldsymbol{f}$(s), $\boldsymbol{f}$ (t), $\boldsymbol{T}$) then s ← t
Output: the final state s

The interesting bit

Simulated Annealing: Picking Probability

How do we pick probability P? Note 3 parameters.

- Decrease with time
- Decrease with gap $|f(s) - f(t)|$



Pick initial state s
For $k = 0$ through k_{max}:
    $T \leftarrow$ temperature($(k+1)/k_{max}$)
    Pick a random neighbour, $t \leftarrow$ neighbor(s)
    If $f(t)$ better than $f(s)$, then $s \leftarrow t$
    Else, with prob. $P(f(s), f(t), T)$ then $s \leftarrow t$
Output: the final state s

Simulated Annealing: Picking Probability

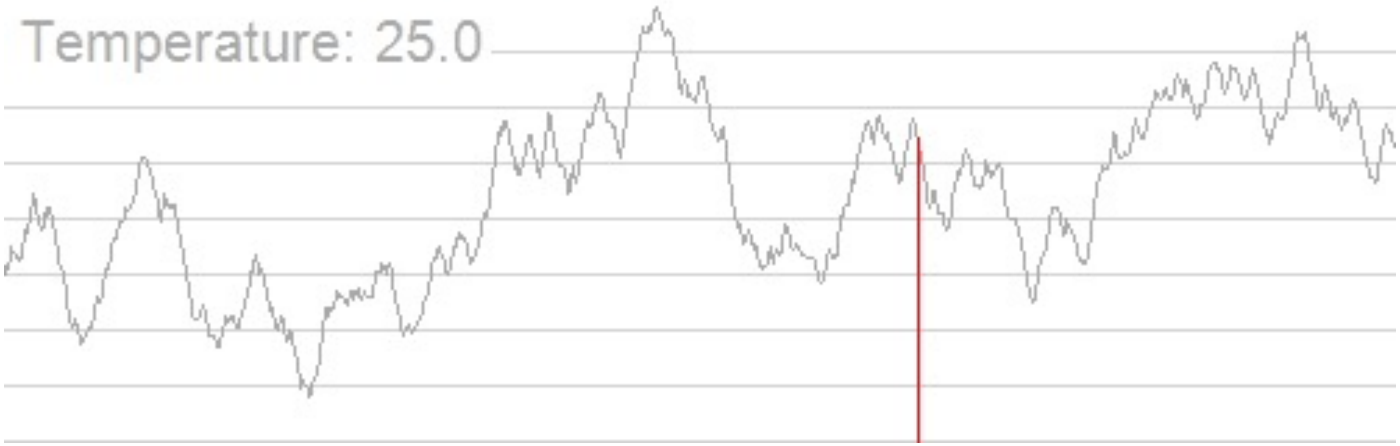
How do we pick probability P? Note 3 parameters.

- Decrease with time
- Decrease with gap $|f(s) - f(t)|$: $\exp\left(-\frac{|f(s)-f(t)|}{Temp}\right)$

- Temperature cools over time.
  - So: high temperature, accept any *t*
  - But, low temperature, behaves like hill-climbing
  - Still, $|f(s) - f(t)|$ plays a role: if big, replacement probability low.

Simulated Annealing: Visualization

What does it look like in practice?



Wiki

Simulated Annealing: Picking Parameters

- Have to balance the various parts., e.g., cooling schedule.
  - Too fast: becomes hill climbing, stuck in local optima
  - Too slow: takes too long.
- Combines with variations (e.g., with random restarts)
  - Probably should try hill-climbing first though.
- Inspired by cooling of metals
  - We’ll see one more alg. inspired by nature

Genetic Algorithms

Another optimization approach based on nature

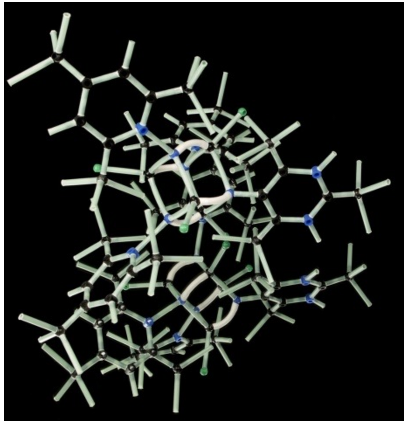
- Survival of the fittest!

Evolution Review

Encode genetic information in DNA (four bases)

- A/C/T/G: nucleobases acting as symbols

- Two types of changes
  - Crossover: exchange between parents' codes
  - Mutation: rarer random process
    - Happens at individual level



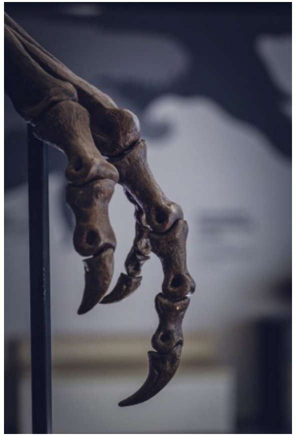
Natural Selection

Competition for resources

- Organisms better fit ➔ better probability of reproducing
- Repeated process: fit become larger proportion of population

Goal: use these principles for optimization

- New terminology: state is ‘**individual**’
- Value $f(s)$ is now the ‘**fitness**’

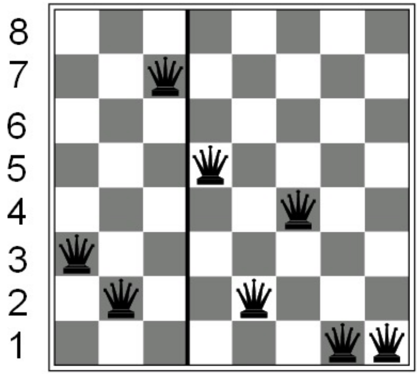


Genetic Algorithms Setup I

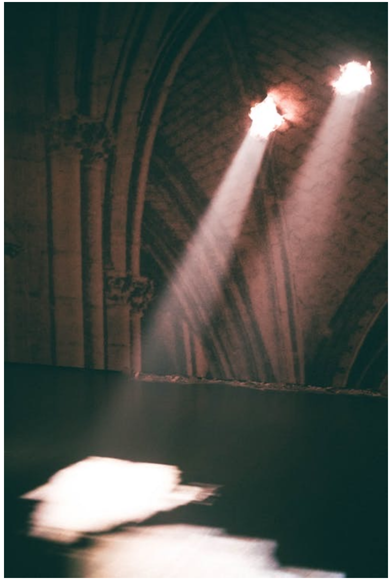
Keep around a fixed number of states/individuals

- Call this the **population**

For our n Queens game example, an individual:



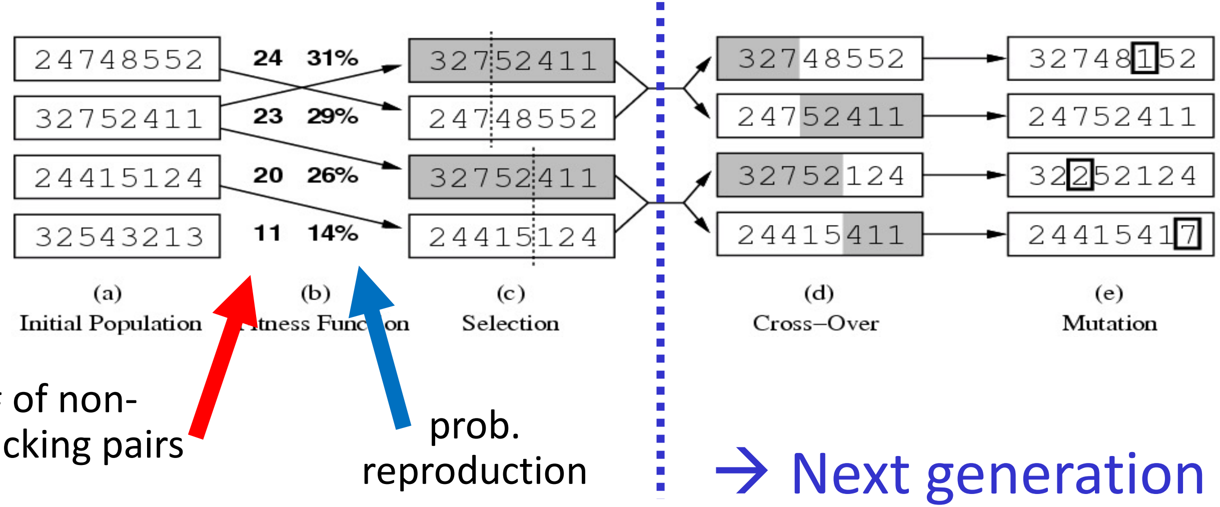
(3 2 7 5 2 4 1 1)



Genetic Algorithms Setup II

Goal of genetic algorithms: optimize using principles inspired by mechanism for evolution

- E.g., analogous to **natural selection, cross-over**, and **mutation**



# of non-attacking pairs

prob. reproduction ∝ fitness

→ Next generation

Genetic Algorithms Pseudocode

Just one variant:

1. Let $s_1, \ldots, s_N$ be the current population
2. Let $p_i = f(s_i) / \Sigma_j f(s_j)$ be the reproduction probability
3. for $k = 1$; $k<N$; k+=2
   - parent1 = randomly pick according to p
   - parent2 = randomly pick another
   - randomly select a crossover point, swap strings of parents 1, 2 to generate children $t[k]$, $t[k+1]$
4. for $k = 1$; $k<=N$; k++
   - Randomly mutate each position in $t[k]$ with a small probability (mutation rate)
5. The new generation replaces the old: $\{ s \} \leftarrow \{ t \}$. Repeat

Reproduction: Proportional Selection

Reproduction probability: $p_i = \boldsymbol{f}(s_i) \,/\, \Sigma_j \boldsymbol{f}(s_j)$

- **Example**: $\Sigma_j \boldsymbol{f}(s_j)$ = 5+20+11+8+6=50
- p_1=5/50=10%

Individual	Fitness	Prob.
A	5	10%
B	20	40%
C	11	22%
D	8	16%
E	6	12%

Acknowledgements: Adapted from materials by Jerry Zhu + Tony Gitter (University of Wisconsin), Andrew Moore