# Scientific Computing

March 21, 2025

## Announcements

→ Homework 4 assigned, see D2L
Due Wednesday, April 2, 11:59pm

## Today

→ Introduction to Metaheuristics

→ Hill Climbing

**Office Hours:**

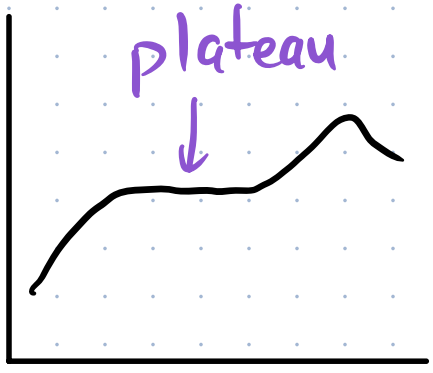
Mon + Fri

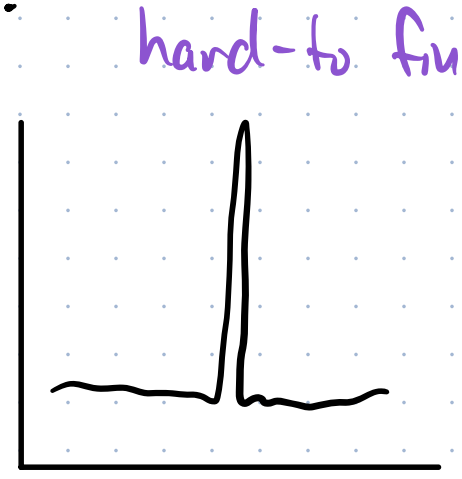
9:30am - 10:30am

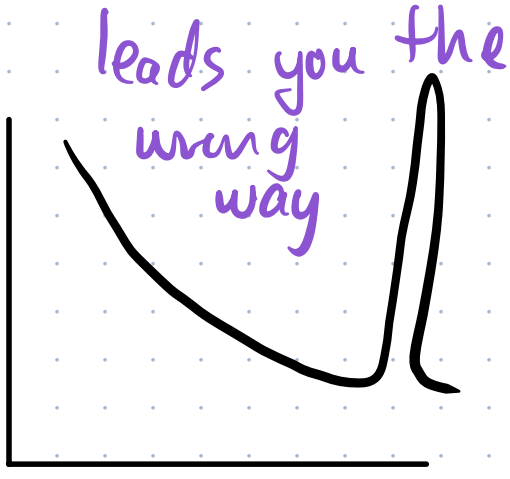
Cudahy 307

Metaheuristics are all about exploring the search space in clever ways in the hopes of finding the global optimum.

Typical obstacles:





We can't see the landscape when we're on the mountain!

# Topic 11 - Hill Climbing

With Gradient Ascent as our inspiration, we want to think about ways to search for global optima in cases where our search space is (1) discrete
(2) cts, but can't compute a gradient

# Problem Setup:

* Search space $S$ full of candidates / possible solutions

* Scoring function: score(x), $x \in S$
  (also called "fitness" w/ biological inspiration, or "quality")

* A way to generate either

x = current candidate = where you're standing in the mountains

- all the candidates "near" a candidate, the "neighborhood" nbhd(x)

probably doesn't make total sense in cts space

OR

one

- a random candidate near a candidate (sometimes called a "tweak") tweak(x).

"nearby" is up for you to define, and different definitions can totally change how a metaheuristic behaves.

Two running examples in this section.

(1) TSP:

* discrete
* score = cost of tour, want to minimize
* nbhd(x): suppose $x = c_1 \to c_2 \to \dots \to c_n \to c_1$

"n-1 choose 2" Define the neighborhood to be all ways of picking two cities and swapping them (excluding $c_1$).

size $= \binom{n-1}{2} = \frac{1}{2}(n-1)(n-2)$ (big!) $\approx n^2$

* tweak(x): a random thing in the nbhd of x

(2) optimizing a continuous function in two variables $f(x,y)$.

* continuous
* score = value of the function
* nbhd(x) = all points within some fixed distance $\delta$ of x
* tweak(x) = a random point in nbhd(x)

$\delta$: small #

infinite

# MH #1: Random Search

```
best = random element of S
while True:                    (quit whenever you want)
    x = random element of S
    if score(x) > score(best):
        best = x
```

Possible stopping conditions:

* best score does not improve for N iters
* preset number of iters
* you get impatient

This is not a good metaheuristic in most cases! It doesn't use any old information to guide future choices.

[2 Demos]
- 01: TSP random
- 02: Contour 1- random

Gradient Ascent inspires this next one.

MH #2: Steepest Ascent Hill-Climbing (Discrete only)

```
x = random element of S
while True:
    N = nbhd(x)
    s = element of N with the best score
    if score(s) > score(x):
        x = s
    else:
        quit
```

Stopping conditions:
* run out of time
* no further improvement

What does this do? Climbs right up the hill you start on.

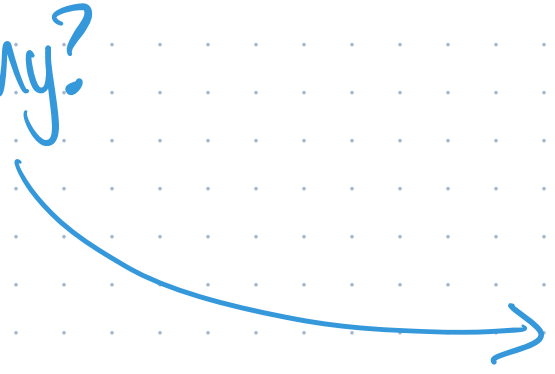
**Pros**

* Finds a local optimum

**Cons**

* Unlikely to find global optimum except in very nice spaces
* very slow, especially if nbhds are big, like TSP.

why?



Only really doing two things:

(1) generating the neighborhood

(2) scoring each element of it

Scoring a tour with 300 cities is not horrible — 300 distance calculations (two subtractions, two squarings, one addition, one square root)

But bad when you do it $\binom{299}{2} = 44{,}551$ times.

Often, you don't have to restore a solution from scratch because it only changes a little bit. More on this in a bit.

Demos: 03 - TSP St. Asc. 50
04 - TSP St. Asc. 300 slow!