Scientific Computing

March 24, 2025

Announcements

→ Homework 4 assigned, see D2L

Due Wednesday, April 2, 11:59pm

Today

→ Hill Climbing

Office Hours:

Mon + Fri

9:30am - 10:30am

Cudahy 307

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

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

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

probably doesn't make total sense in cns space

OR

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

one

"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 — finite search space
* score = cost of tour, want to minimize
* nbhd(x): Suppose $x = c_1 \to c_2 \to \dots \to c_n \to c_1$

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

  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 ~ infinite
* score = value of the function
* nbhd(x) = all points within some fixed distance δ of x — small #
* tweak(x) = a random point in nbhd(x)

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

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

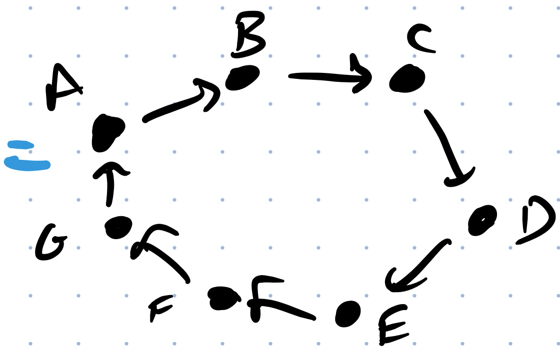
Scoring a tour with 300 cities is not horrible — 300 distance calculations (two subtractions, two squarings, one addition, one square root)

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!

How can we speed up scoring? Think about our tweak function. Suppose we have a tour:



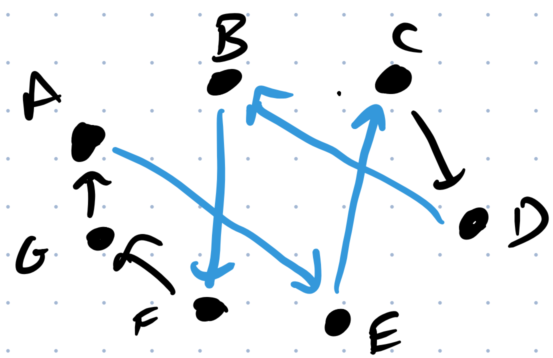
Let d = distance function.

$$\text{Score} = d(A,B) + d(B,C) + d(C,D) + d(D,E) + d(E,F) + d(F,G) + d(G,A)$$

Swap B and E:

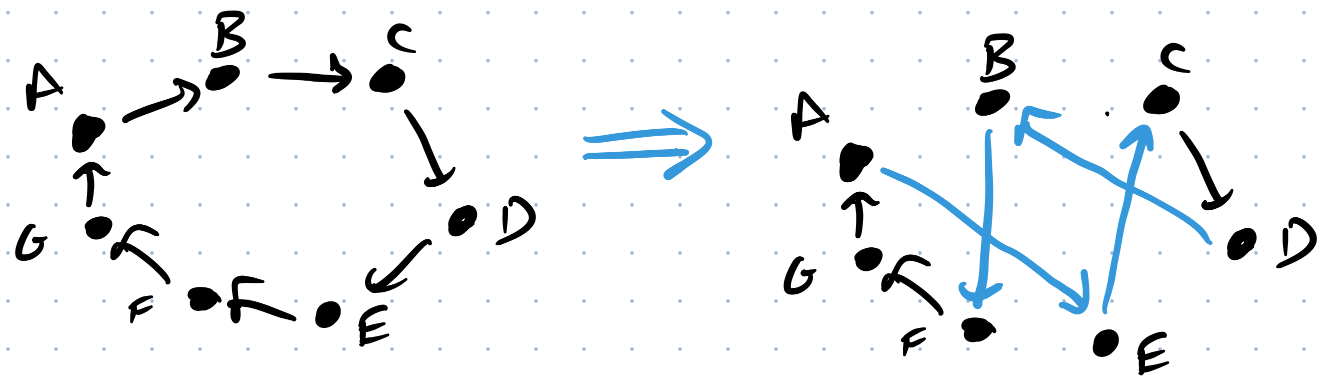
$A \to B \to C \to D \to E \to F \to G \to A$

$A \to E \to C \to D \to B \to F \to G \to A$



A → B → C → D → E → F → G → A

A → E → C → D → B → F → G → A



Four edges change.

$$\text{Score} = \cancel{d(A,B)} + \cancel{d(B,C)} + d(C,D) + \cancel{d(D,E)} + \cancel{d(E,F)} + d(F,G) + d(G,A)$$

$d(A,E)$ $d(E,C)$ $d(D,B)$ $d(B,F)$

If you have 300 cities, still only 4 edges change:

new score = old score − 4 edges + 4 edges.

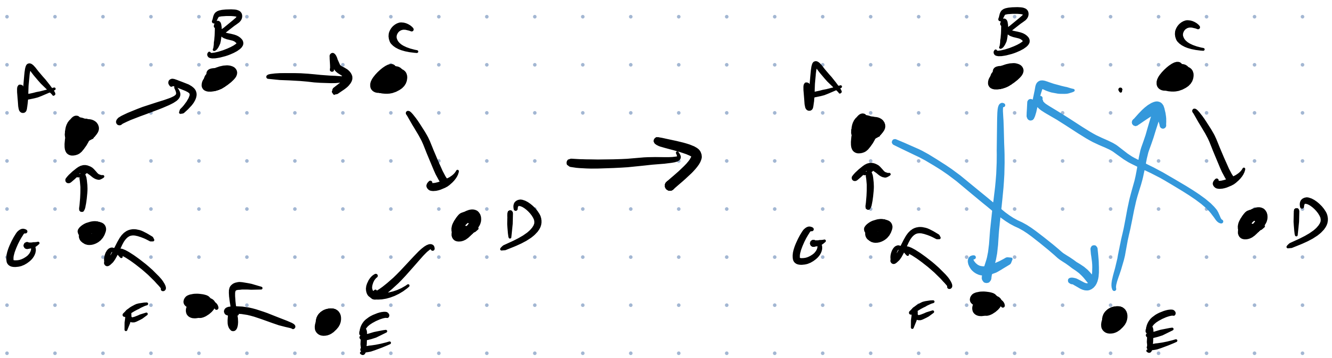
8 distances instead of 300 = $\frac{300}{8}$ = 37.5x faster!

Demo 05

Second question: Is this a good tweak?

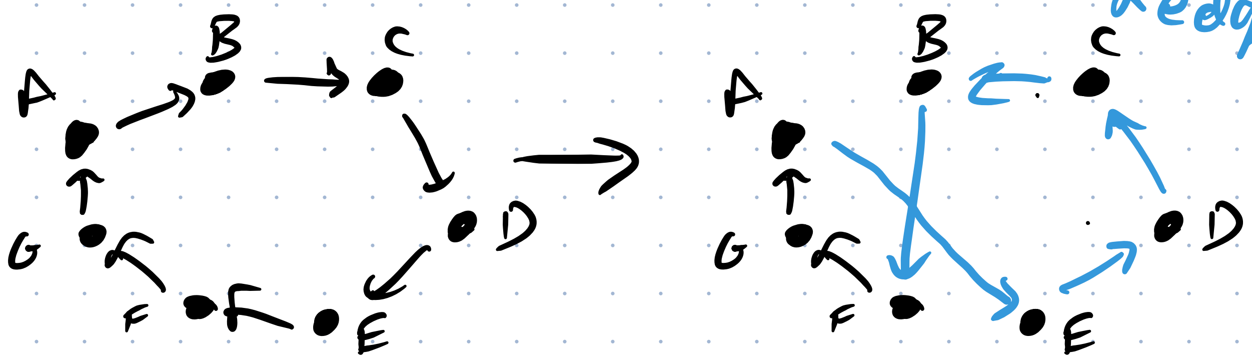
Our demos suggest maybe not.

Why not? Small tweaks are better (at least for now).



4 edges changed... can we change just 1?
No. How about 2?

4 edges: swap 2 cities
2 edges: reverse block



Assuming distance is symmetric yes.
This is picking two cities and reversing the whole block.

A → B → C → D → E → F → G → A

A → E → D → C → B → F → G → A

(faster scoring too!)

Demos: 06 - SA RB 50
07 - SA RB 300

08 / Fast Score

The big theme of MHs is that they are super <u>flexible</u>. You can and should tweak them in all kinds of ways for particular problems. Always experiment.

How can we adapt this for continuous spaces?

How can we adapt this for continuous spaces?

Check n things in the neighborhood and take the best.

MH #3 n-Trial Steepest Ascent

```
x = random element of S
while True:
    temp = x
    repeat n times:
        s = tweak(x)
        if score(s) > score(temp)
            temp = s
    x = temp
```

one step

(if nothing beats x, it stays the same)

temp is the best neighbor out of the n we tried

Later we will see good ways to tweak for continuous spaces.

When n=1, this is just called "Hill Climbing"

MH #4: Hill Climbing

```
x = random element of S
while True:
    s = tweak(x)
    if score(s) > score(x):
        x = s
```

Repeat:

- Take a (small) step.
- If better, stay.
  - If worse, go back.
- Repeat

HC = Hill Climbing

SA = Steepest Ascent

Demos:

09 -	TSP	HC	Swap	2	50
10 -	TSP	HC	Swap	2	300
11 -	TSP	HC	RB		50
12 -	TSP	HC	RB		300

RB = reverse a whole block of cities

Swap 2 = swap just 2 cities

50 cities		300 cities	
SA Swap 2	9.878	SA Swap 2	32.828
SA RB	6.487	SA RB	14.362
HC Swap 2	8.428	HC Swap 2	29.439
HC RB	6.412	HC RB	14.252

Regular HC beat Steepest Ascent

RB very much beat Swap 2