Scientific Computing

March 19, 2025

Announcements

→ None!

Today

→ Object-Oriented Programming

→ Introduction to Metaheuristics

Office Hours:

Mon + Fri

9:30am – 10:30am

Cudahy 307

Topic 10 - Introduction to Metaheuristics

We have mostly focused on ways to find an optimal solution.

These techniques can be hard, and aren't always applicable to real-world problems, or are way too slow.

Metaheuristics:
- General problem solving paradigms that can be <u>easily</u> adapted to many problems
- Look for <u>good</u> solutions, and rarely find an actual optimal one
- Pretty fast

Similar setup:

evolutionary biology

* Search space of candidates / solutions
* Every candidate has a score (/ fitness / quality)

* Goal: Find a candidate with a good score
[in the abstract we will always talk about maximizing, but in some applications we want to minimize]

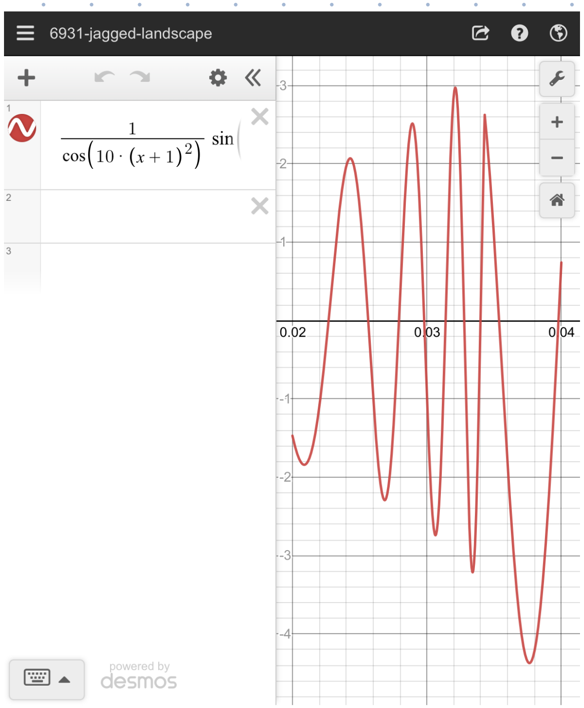
Many of our problems will be discrete (finite search space), but some will be continuous.

[traveling salesman demos]

Ex: Find the maximum value of

$$f(x) = \frac{1}{\cos\left(10(x+1)^2\right)} \cdot \sin\left(\min\left((x+1)^{100}, \frac{1}{x}\right)\right)$$

on the interval $0.02 \le x \le 0.04$.



Maybe we could do this one with calculus, but we'll usually have functions that are more implicit, like solutions of ODEs.

Most of the spaces we'll work in are not 1-D.

Traveling Salesman or Knapsack:

Finite search space, but huge

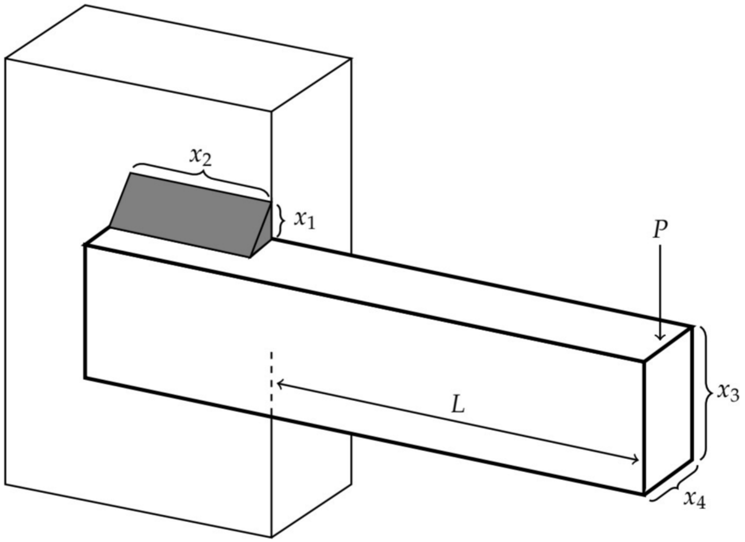
Technically 0-dimensional

2. Figure 2 below shows a long, horizontal beam welded to a taller wall. The gray shaded triangular prism is the weld that fixes the long beam to the wall.

   The beam will bear weight and therefore must be strong enough to withstand a downward force P on the far end without risk of deflecting, buckling, or the weld breaking. At the same time, you want to produce the beam as cheaply as possible.

   The cost of producing the beam and weld in dollars, including materials and labor, is

$$1.10471x_1^2x_2 + 0.04811x_3x_4(10 + x_2).$$



This is the quantity you want to minimize.

In order to make sure that the beam functions safely, the following constraints must be satisfied.

$$\begin{aligned}
T(x_1, x_2, x_3, x_4) &\leq 13600, \\
S(x_1, x_2, x_3, x_4) &\leq 30000, \\
x_1 &\leq x_4, \\
0.10471x_1^2 + 0.04811x_3x_4(10 + x_2) &\leq 5, \\
D(x_1, x_2, x_3, x_4) &\leq 0.25, \\
6000 &\leq Q(x_1, x_2, x_3, x_4), \\
0.125 \leq x_1 &\leq 2, \\
0.1 \leq x_2 &\leq 10, \\
0.1 \leq x_3 &\leq 10, \\
0.1 \leq x_4 &\leq 2,
\end{aligned}$$

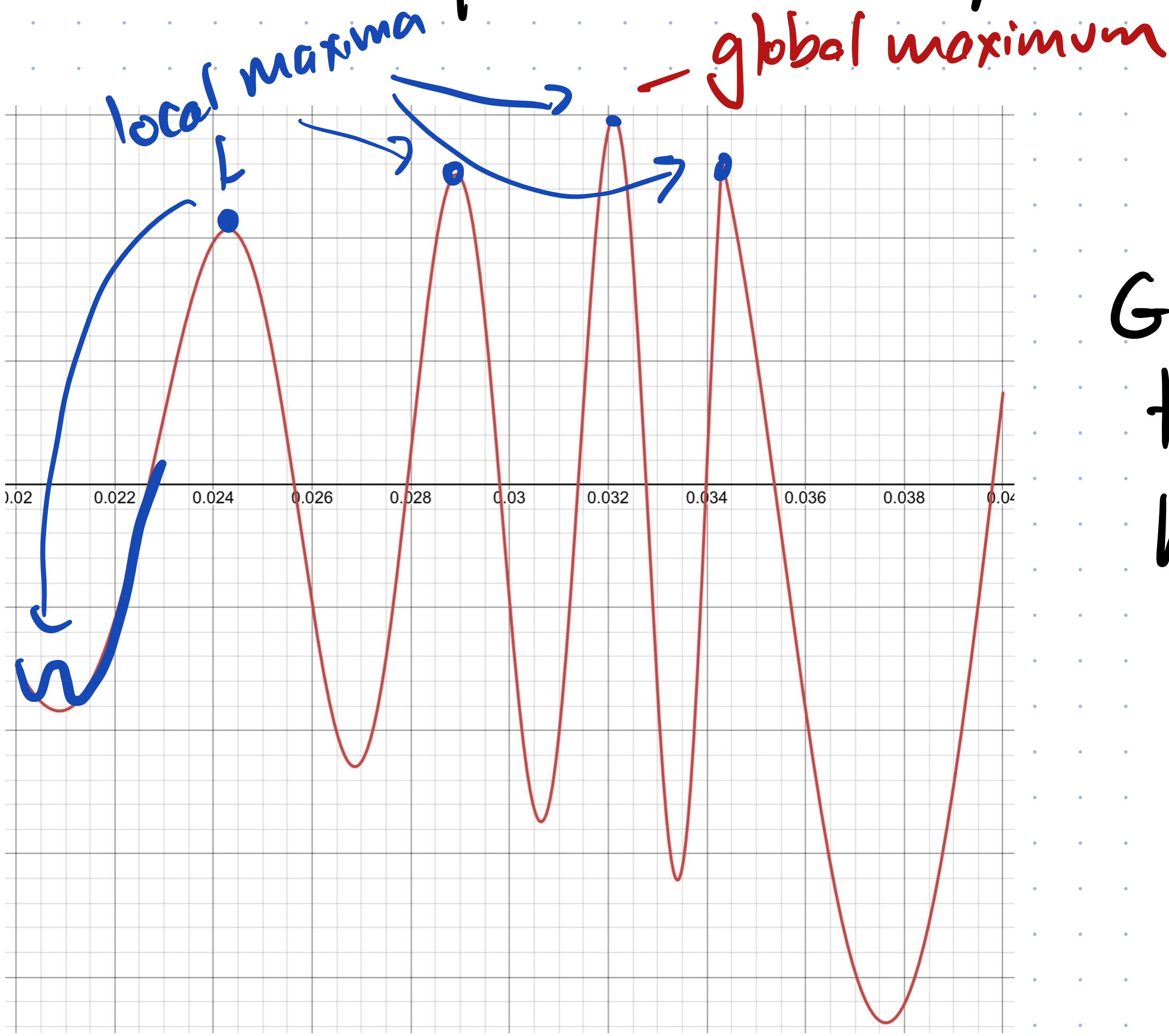
where the functions T, S, D, and Q are defined by:

$$\begin{aligned}
R(x_1, x_2, x_3, x_4) &= \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)} \\
J(x_1, x_2, x_3, x_4) &= \sqrt{2} \cdot x_1x_2 \left(\frac{x_2^2}{12} + \left(\right.\right. \\
S(x_1, x_2, x_3, x_4) &= \frac{6PL}{x_3^2x_4} \\
D(x_1, x_2, x_3, x_4) &= \frac{4PL^3}{Ex_3^3x_4} \\
Q(x_1, x_2, x_3, x_4) &= \frac{4.013x_3x_4^3\sqrt{EG}}{6L^2} \cdot \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \\
E &= 30 \cdot 10^6 \\
G &= 12 \cdot 10^6 \\
L &= 10 \\
P &= 6000
\end{aligned}$$

$$\begin{aligned}
T(x_1, x_2, x_3, x_4) &= \sqrt{(T')^2 + 2 \cdot T' \cdot T'' \cdot \frac{x_2}{2R} + (T'')^2} \\
T'(x_1, x_2, x_3, x_4) &= \frac{P}{\sqrt{2} \cdot x_1 \cdot x_2} \\
T''(x_1, x_2, x_3, x_4) &= \frac{M \cdot R}{J} \\
M(x_1, x_2, x_3, x_4) &= P \cdot \left(L + \frac{x_2}{2}\right)
\end{aligned}$$

4D problem!

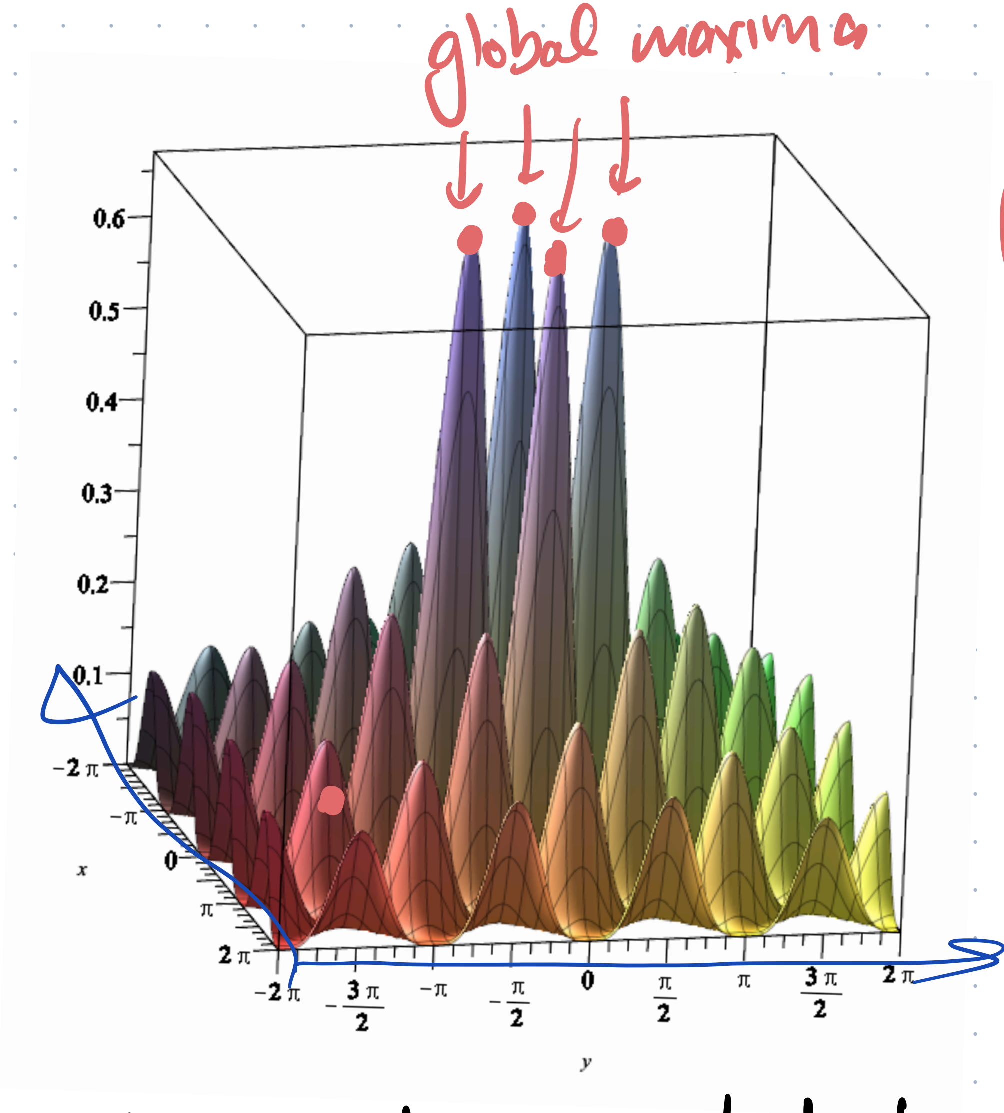
Useful mental picture: landscapes



Goal: Climb to the top of the tallest hill

1D input (x) 1D output (y)

2D input (x,y)   1D output (z)



$$\frac{\sin^2(x-y)\sin^2(x+y)}{\sqrt{x^2+y^2}}$$

Goal: find the top of the tallest hill but don't get stuck on the wrong hilltop!

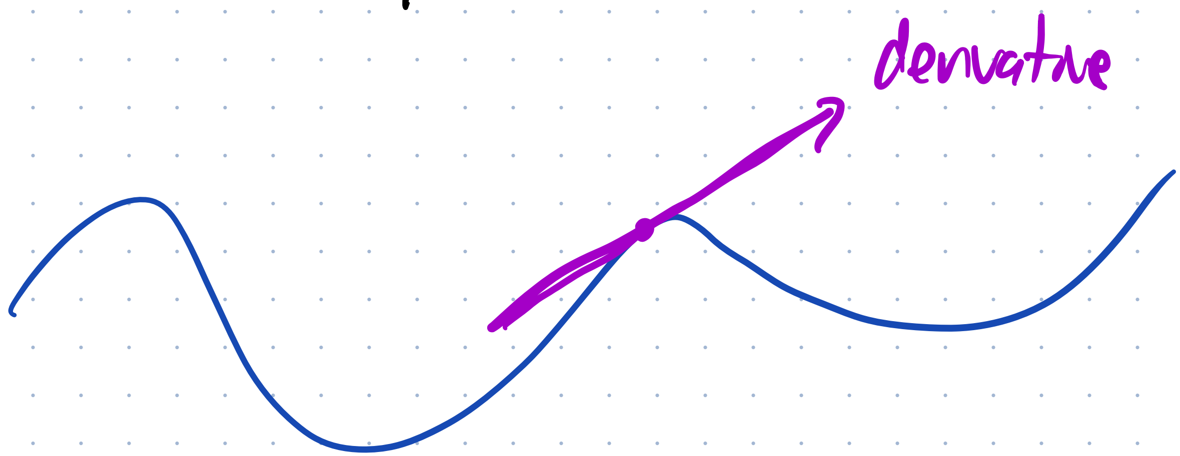
x/y is the ground location
z is the altitude

[demos]

Gradient Ascent / Descent

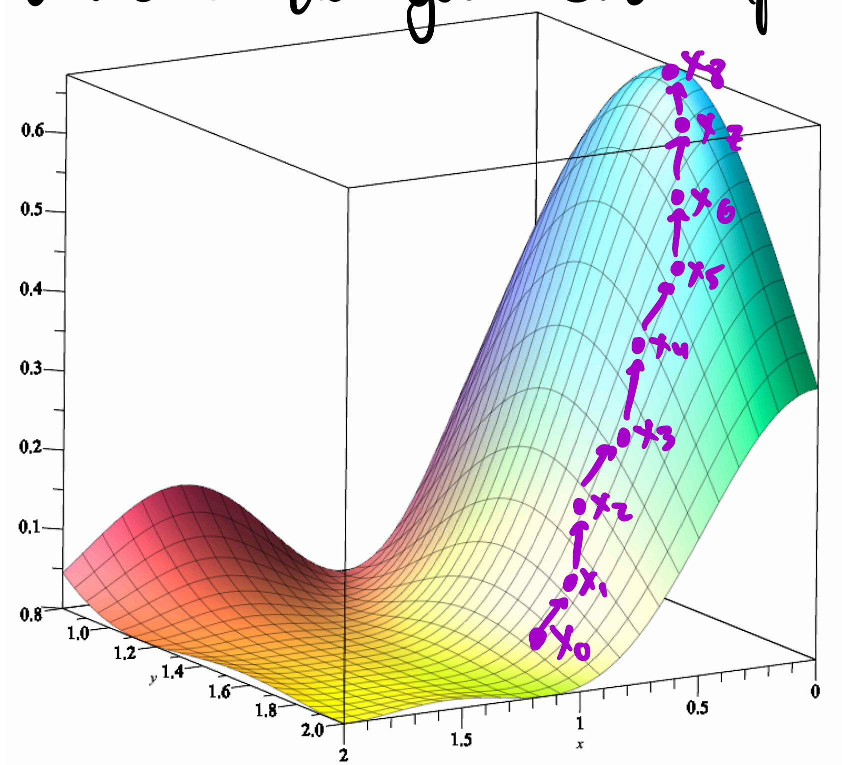
* Optimization method you learn in other classes

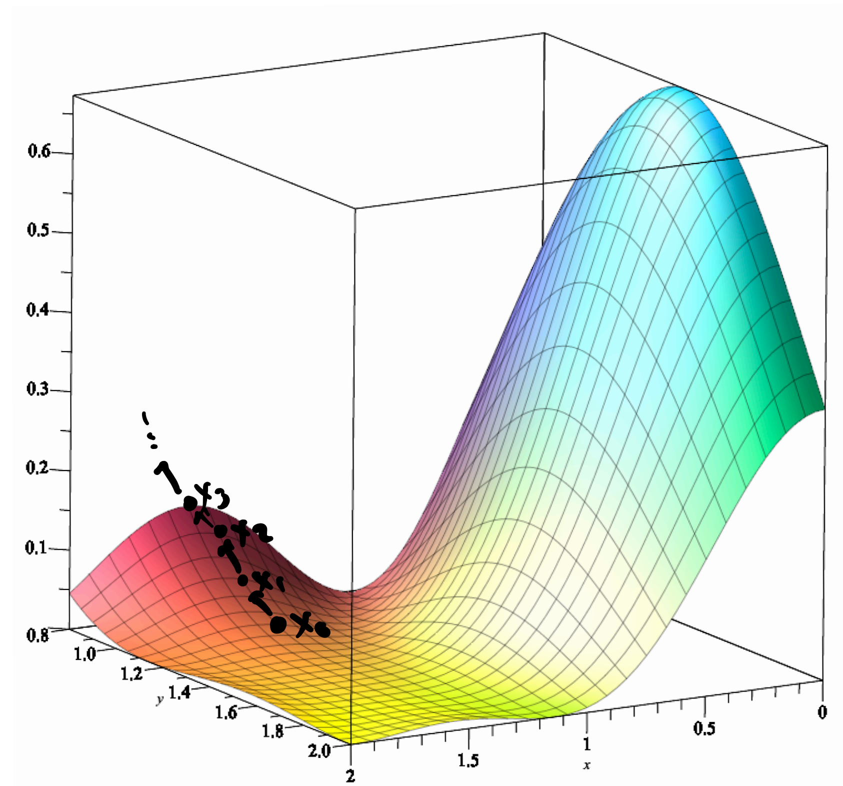
* If your function $f(x,y)$ differentiable, you can compute the gradient at a point, which is a vector that points you in the direction of steepest ascent.



* (1) Start at a point
  (2) compute gradient
  (3) move a little in that direction
  (4) repeat

Where do you end up?





might overshoot!

Always imagine yourself standing on the mountainside. You're just going in the steepest direction.

You usually end up at the top of the peak you start on (a local optimum).

We want a global optimum!

How could you do something like Gradient Ascent in a discrete search space (like TSP)?

[pretend you're in the mountains]

* look around you in a small "radius"
* find the point in your "radius" that is highest
* go there and repeat

Ex: TSP

- search space: all tours on the graph (these are the places on the mountain you could be standing)

- need a definition of "nearby" ("small radius")

cities 1, 2, $\overset{*}{3}$, 4, 5

what is nearby $3 \to 5 \to 2 \to 1 \to 4 \to 3$

up to you! many answers. one definition could be: swap any two cities

$3 \to 5 \to 2 \to 1 \to 4 \to 3$

$3 \to 1 \to 2 \to 5 \to 4 \to 3$

* start at a random tour
* calculate the score of all the nearby tours
* move to the cheapest one
* repeat

Of course, same problem - we just climb up a local optimum and never come down.

[demos]