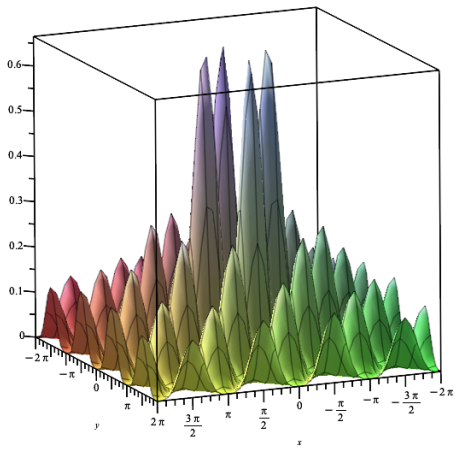
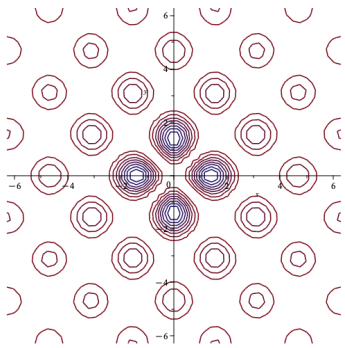
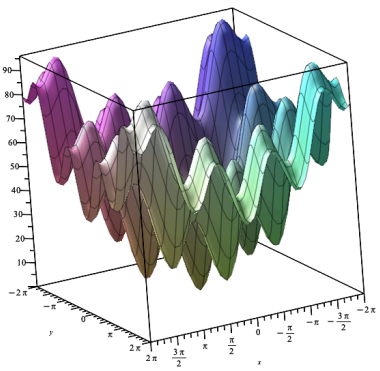
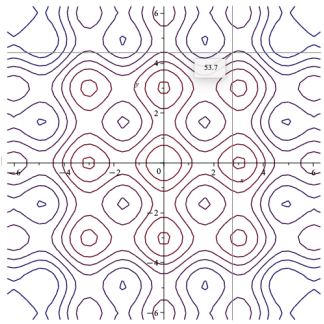
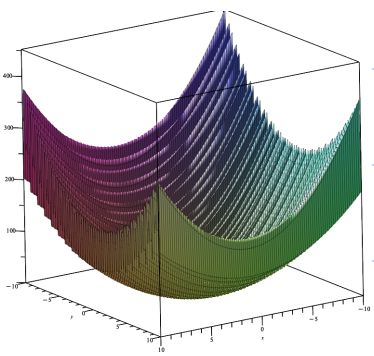
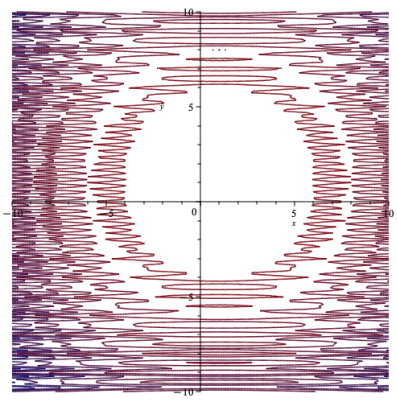


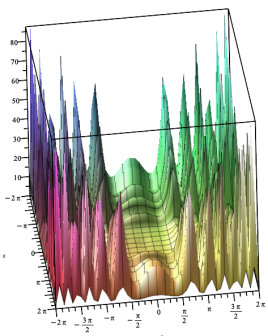


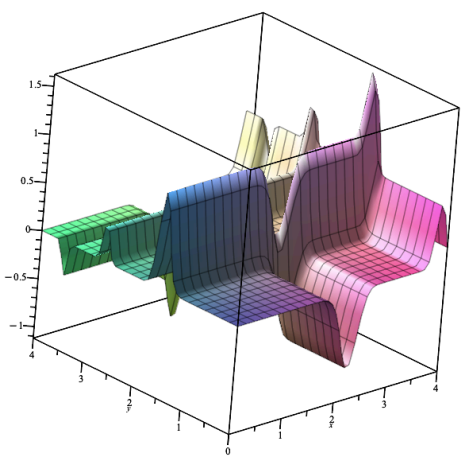












Wed, Mar. 27, 2024
Scientific Computing

Announcements

→ No OH Th. or Mon
→ No lecture Fri or Mon
→ Monday is a home work day to work on HW 4
→ HW 4 due next Fri, but extension requests will be granted
→ Office Hours today, 2pm - 3pm

Topic 9 - Introduction to Metaheuristics (continued)

HW 4, question 2

```
class Vector:

    def __init__(self, x, y, z):
        :

    def dot(self, other):
        do stuff ...
        return ans
```

Vector object

(b)

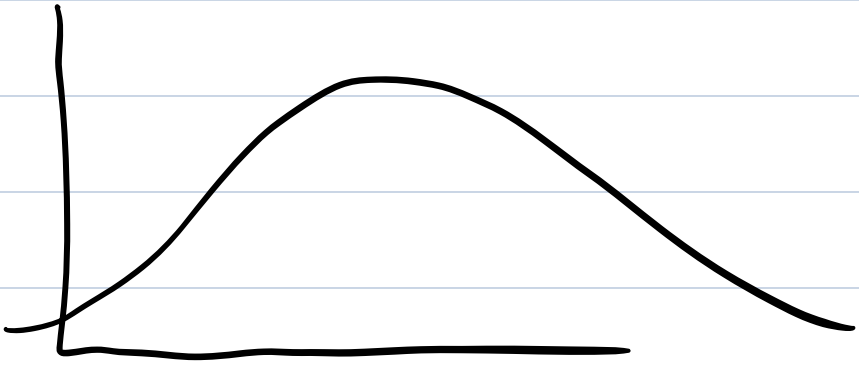
```
def test_dot_product():
    v1 = Vector(-3, 1, 2)
    v2 = Vector(5, 0, 7)
    dot_product = v1.dot(v2)
    assert dot_product == -1

:

test_dot_product()
:
```

```
v = Vector(3,2,1)
assert v.x == 3
```

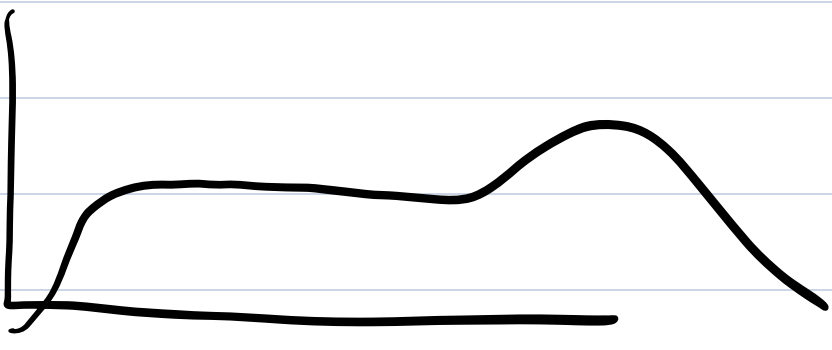
Topic 9 - Intro to metaheuristics.

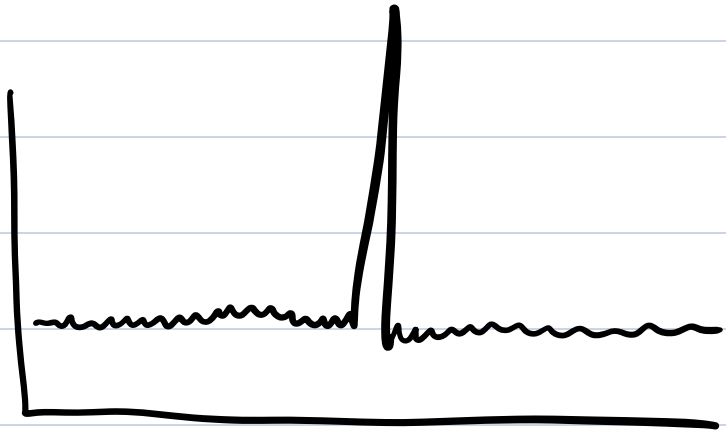
What makes a landscape difficult?

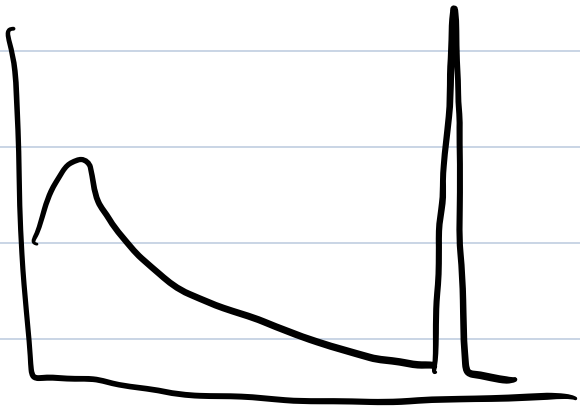
Easy



Obstacles:







Topic 10 - Hill Climbing

↳ Metaheuristic version of gradient ascent

Gradient Ascent doesn't work for discrete search spaces or non-diff. functions.

Hill-Climbing works for everything.

↓

Problem Setup:

* Search space S of candidates
* Scoring function: score(x) for $x \in S$
  (also called fitness or quality)
* A way to generate either:
  - all the candidates "near" a particular candidate

the set of all candidates "near x" is called the "neighborhood of x"

Notation: nbhd(x)

OR

- a random candidate "near" another one, a "tweak"

tweak(x) = an element of the search space that is "near" x.

→ "nearby" is up for you to decide
→ different definitions can give better or worse solutions

Two running examples in this lecture:

(1) TSP:

* discrete
* search space = all tours of the cities
* score = sum of the distances traveled, we are minimizing
* Let $x = C_1 \to C_2 \to \cdots \to C_n \to C_1$

Define nbhd(x) to be all tours you can get by swapping two cities.

How big is nbhd(x)?

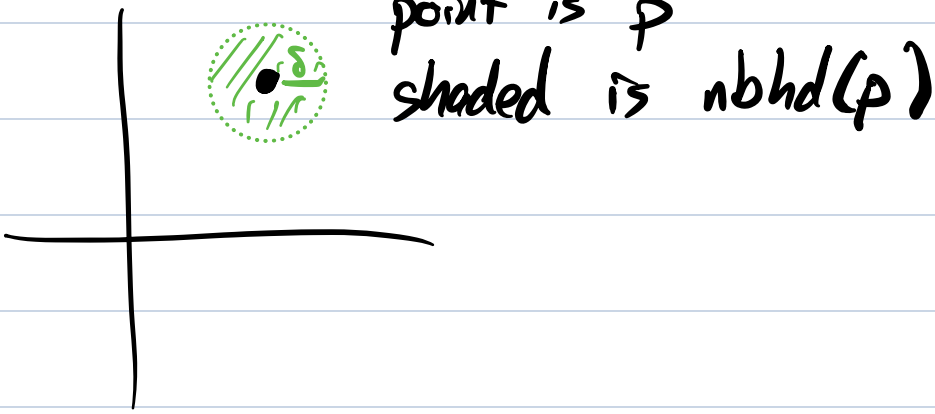
$$\binom{n-1}{2} = \frac{(n-1)\cdot(n-2)}{2}$$

$$\approx n^2/2$$

* tweak(x): randomly pick two cities and swap the

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

* continuous search space containing all (x,y) points, maybe in some bounds
* score = the value of the function at that point
* nbhd(p) = all points within some fixed distance δ of p.
  ↑ Euclidean



point is p
shaded is nbhd(p)

* tweak(p) = one random point in nbhd(p)