Usage:

1) create subclasses
2) import in main
3) instantiate Specific Swarm
4) Call Specific Config

Set:

- dim · weights · cost
- DHW · # of particles

New usage:

1) Define params: max_it, pos_lims, vel_lims
   weights, dimensions, population
2) instantiate Swarm
   a) instantiate particles

Swarm Particle:

Props: Pos :: Position
velocity :: Velocity
cost :: num

Pbest_cost :: num
Pbest_pos :: Position
lbest_cost :: num
lbest_pos :: Position
dim :: num
neighbors :: [Particle HANDLES]

Methods: update Velocity
update Position()
get Neighbors(n)
update Lbest()
evaluate

Swarm Class:

Props: Swarm_arr [Particles]
Config :: Swarm Config

Methods: updateWeights()
☆optimize
update GlobalBest

Classes:

- Particle
- Swarm
- Position
- Velocity
- SwarmConfig (?)
- Sol'n Space (?)
- Problem
  - to define Consts of cost_func

4D continuous PSO OOP

Noes:

- Weights class?

Particle.m

Props:
- Cost
- position
- velocity
- pbest_cost
- pbest-position
- lbest_cost
- lbest-position
- neighbors = [Particles]
- velocity_limits[min, max] (const)
- position_limits(min, max)(const)

Swarm:

Props: population
max_it
iteration
weights
swarm_arr[Particles]
gbest_cot
gbest-pos
cost_func

Weights.m:

prop: ω
w_1
w_2
w_3

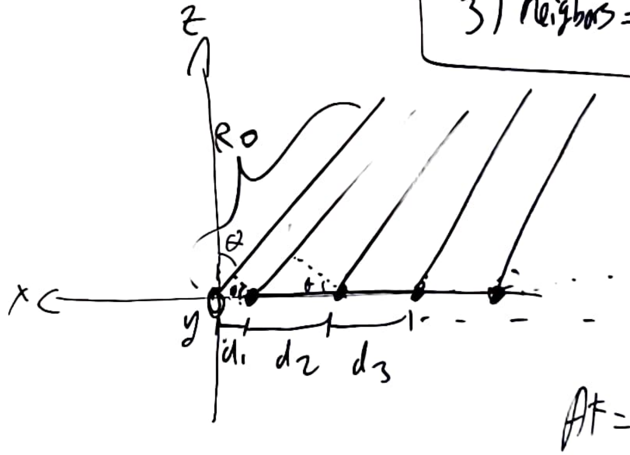
method: get.prop(obj, n)
$ use Dependent

OR

method: updateWeights

Find n nearest:

1) $[\vec{d}] = |[\vec{x}_{j\neq i}] - \vec{x}_i|$
2) $\vec{d} = \text{Sort}(\vec{d})$ low to high
3) neighbors $= \vec{d}(1:n)$



$$R_i = R_{i-1} - d_i \sin\theta$$

$$R_i = R_0 - \sum_{n=1}^{i} d_i \sin\theta$$

$$A_T = \sum A_i e^{jk\sum_{n=1}^{i} d_i \sin\theta}$$

Usage:

1) Create specific subclasses
2) import in main
3) instantiate specific Swarm
4) call specific config

Set

- dim
- weights
- cost
- DHN
- # of particles

[illegible] mize:

```
while (!converged || i < max_it)
    for (all Particles in Swarm)
        Particle.evaluate(cost_func)   <  return = cost_func(Particle.position)
    end
    Swarm.updateGlobalBests
```

SwarmParticle Class:

Props: position :: Position
velocity :: Velocity
cost :: num
pbest_cost :: num
pbest_pos :: Position
lbest_cost :: num
lbest_position :: num
dim :: num (? different class)

neighbors :: array[Particle Handles]

methods: updateVelocity()
UpdatePosition()

calls { getNeighbors(n) [get closest n particles]
UpdateLocalBest
evaluate(cost_func)

Swarm Class:

Props: Swarm_arr[Particles]
Config: SwarmConfig

methods: update Weights()
★ Optimize()
update GlobalBests()

Classes:

- Swarm
- Particle
- SwarmConfig (?)
- Solution Space (?)
- Problem
- cost func wrapper
- allow constants

- DHN as subclass of Config

- How to handle different dims?
- May not need SwarmConfig
- Must validate dim(cost_func) = dim(particle)
- Swarm ↔ Particle access

Notes:

- Vectorize Particle props?
- always minimize?
- Particle (Composes) Swarm aggregates
- Swarm holds gbest