# python implementation of particle swarm optimization (PSO)

# minimizing rastrigin and sphere function

import random

import math # cos() for Rastrigin import copy # array-copying convenience

import sys # max float

#-------fitness functions---------

# rastrigin function def fitness\_rastrigin(position):

fitnessVal = 0.0 for i in range(len(position)):

xi = position[i]

fitnessVal += (xi \* xi) - (10 \* math.cos(2 \* math.pi \* xi)) + 10 return fitnessVal

#sphere function def fitness\_sphere(position): fitnessVal = 0.0 for i in range(len(position)):

xi = position[i] fitnessVal += (xi\*xi);

return fitnessVal;

#-------------------------

#particle class class Particle:

def \_\_init\_\_(self, fitness, dim, minx, maxx, seed):

self.rnd = random.Random(seed)

# initialize position of the particle with 0.0 value self.position = [0.0 for i in range(dim)]

# initialize velocity of the particle with 0.0 value self.velocity = [0.0 for i in range(dim)]

# initialize best particle position of the particle with 0.0 value self.best\_part\_pos = [0.0 for i in range(dim)]

# loop dim times to calculate random position and velocity # range of position and velocity is [minx, max] for i in range(dim):

self.position[i] = ((maxx - minx) \* self.rnd.random() + minx) self.velocity[i] = ((maxx - minx) \*

self.rnd.random() + minx)

# compute fitness of particle

self.fitness = fitness(self.position) # curr fitness

# initialize best position and fitness of this particle self.best\_part\_pos = copy.copy(self.position) self.best\_part\_fitnessVal = self.fitness # best fitness

# particle swarm optimization function def pso(fitness, max\_iter, n, dim, minx, maxx):

# hyper parameters w = 0.729 # inertia

c1 = 1.49445 # cognitive (particle)

c2 = 1.49445 # social (swarm)

rnd = random.Random(0)

# create n random particles

swarm = [Particle(fitness, dim, minx, maxx, i) for i in range(n)]

# compute the value of best\_position and best\_fitness in swarm best\_swarm\_pos = [0.0 for i in range(dim)]

best\_swarm\_fitnessVal = sys.float\_info.max # swarm best

# computer best particle of swarm and it's fitness for i in range(n): # check each particle if swarm[i].fitness < best\_swarm\_fitnessVal: best\_swarm\_fitnessVal = swarm[i].fitness

best\_swarm\_pos = copy.copy(swarm[i].position)

# main loop of pso Iter = 0 while Iter < max\_iter:

# after every 10 iterations

# print iteration number and best fitness value so far if Iter % 10 == 0 and Iter > 1:

print("Iter = " + str(Iter) + " best fitness = %.3f" % best\_swarm\_fitnessVal)

for i in range(n): # process each particle

# compute new velocity of curr particle for k in range(dim): r1 = rnd.random() # randomizations r2 = rnd.random()

swarm[i].velocity[k] = (

(w \* swarm[i].velocity[k]) +

(c1 \* r1 \* (swarm[i].best\_part\_pos[k] - swarm[i].position[k])) + (c2 \* r2 \* (best\_swarm\_pos[k] -swarm[i].position[k]))

)

# if velocity[k] is not in [minx, max]

# then clip it if swarm[i].velocity[k] < minx: swarm[i].velocity[k] = minx elif swarm[i].velocity[k] > maxx: swarm[i].velocity[k] = maxx

# compute new position using new velocity for k in range(dim):

swarm[i].position[k] += swarm[i].velocity[k]

# compute fitness of new position

swarm[i].fitness = fitness(swarm[i].position)

# is new position a new best for the particle? if swarm[i].fitness < swarm[i].best\_part\_fitnessVal: swarm[i].best\_part\_fitnessVal = swarm[i].fitness swarm[i].best\_part\_pos = copy.copy(swarm[i].position)

# is new position a new best overall?

if swarm[i].fitness < best\_swarm\_fitnessVal: best\_swarm\_fitnessVal = swarm[i].fitness

best\_swarm\_pos = copy.copy(swarm[i].position)

# for-each particle

Iter += 1 #end\_while

return best\_swarm\_pos

# end pso

#----------------------------

# Driver code for rastrigin function

print("\nBegin particle swarm optimization on rastrigin function\n") dim = 3

fitness = fitness\_rastrigin

print("Goal is to minimize Rastrigin's function in " + str(dim) + " variables") print("Function has known min = 0.0 at (", end="") for i in range(dim-1): print("0, ", end="")

print("0)")

num\_particles = 50

max\_iter = 100

print("Setting num\_particles = " + str(num\_particles)) print("Setting max\_iter = " + str(max\_iter))

print("\nStarting PSO algorithm\n")

best\_position = pso(fitness, max\_iter, num\_particles, dim, -10.0, 10.0)

print("\nPSO completed\n") print("\nBest solution found:")

print(["%.6f"%best\_position[k] for k in range(dim)]) fitnessVal = fitness(best\_position)

print("fitness of best solution = %.6f" % fitnessVal)

print("\nEnd particle swarm for rastrigin function\n")

print() print()

# Driver code for Sphere function

print("\nBegin particle swarm optimization on sphere function\n") dim = 3

fitness = fitness\_sphere

print("Goal is to minimize sphere function in " + str(dim) + " variables") print("Function has known min = 0.0 at (", end="") for i in range(dim-1): print("0, ", end="")

print("0)")

num\_particles = 50

max\_iter = 100

print("Setting num\_particles = " + str(num\_particles)) print("Setting max\_iter = " + str(max\_iter))

print("\nStarting PSO algorithm\n")

best\_position = pso(fitness, max\_iter, num\_particles, dim, -10.0, 10.0)

print("\nPSO completed\n") print("\nBest solution found:")

print(["%.6f"%best\_position[k] for k in range(dim)]) fitnessVal = fitness(best\_position)

print("fitness of best solution = %.6f" % fitnessVal)

print("\nEnd particle swarm for sphere function\n")