**Pseudo-Code**

**Function definitions:**

1. f1 = absolute value function

g(x) = sum(x)<=-50

2. f10 = quadratic function

3. f12 = rastrigin function

h(x) = f12(x)-1=0

4. f22= spherical function

g(x) = sum(x) <=-2

5. f24 = vincent function

h(x) = f24(x) +1=0

Note: In the results documents, the true constrained objective function minimum is given in the ‘Min’ column

**PSO initializatiuon:**

All PSOs were initialized with:

D=30 #objective function dimsensionality

swarmsize =20

w=0.7

c1=1.4, c2=1.4

maxiter=2000

minstep=1e-12

minfunc=1e-12

x=np.random.uniform(low=func.lb[0],high=func.ub[0],size=(S,D)) #S=swarmsize, D=dimensions

v = np.zeros\_like(x) # particle velocities

fx = np.zeros(S) # current particle function values

fs = np.ones(S, dtype=bool) # feasibility of each particle

fp = np.ones(S)\*np.inf # best particle function values

p = np.ones\_like(x)\*np.inf # best particle values

g = np.inf # best swarm position starting value

return = returns global best, number of constraints violated by the global best, magnitude of those violations, and feasibility (derived from whether any violations occured)

**Hard**

1. Find objective function values for each particle (fx)

2. Get all the feasible particles whose function vales are less than the personal bests. Update the new personal bests accordingly

3. Check if the minimum of the personal bests is less than the global best, if it is, update the global best.

4. While iterations < max number of iterations (2000):

rp,rg = random.uniform #shape =SxD

update the particles velocities -> v = w\*v + c1\*rp\*(p - x) + c2\*rg\*(g - x)

update the particles' positions -> x = x + v

find the function value of each particle

find the feasibility of each particle

update the new personal best (criteria is the smallest feasible function value less than the previous personal best)

compare the new personal best to the global best

if the personal best < global best:

stepsize = np.sqrt(np.sum((g – p\_min)\*\*2))

if abs(fg-fp\_min) <= minfunc:

print('Stopping search: Swarm best objective change less than minfunc’)

return

if stepsize <= minstep:

print('Stopping search: Swarm best position change less than minstep’)

return

else:

update global best

iterations +1

5. return

**Dynamic**

1. find the particle fitness values

where: fitness = fx + penalty

penalty =(q\*\*a)\*s\*np.abs(np.sum(constraint(x)))\*\*b

a,b,s =2 and b =#iterations

2. Update the new personal bests (all new function values < current personal bests).

3. Compare the smallest personal best to the global best, and if its less, update the global best

4. While iterations < max number of iterations (2000):

rp,rg = random.uniform #shape =SxD

update the particles velocities -> v = w\*v + c1\*rp\*(p - x) + c2\*rg\*(g - x)

update the particles' positions -> x = x + v

find the fitness of each particle ->function value+ penalty

update the new personal bests as all fitness values < previous personal bests

find the particle with the smallest fitness fp\_min and compare it to the global best fg

if fp\_min<fg:

stepsize = np.sqrt(np.sum((g – p\_min)\*\*2))

if abs(fg-fp\_min)<=minfunc:

print('Stopping search: Swarm best objective change less than minfunc)

return

elif stepsize <= minstep:

print('Stopping search: Swarm best position change less than minstep)

return

else:

update global best

iterations +1

5. return

**Soft\_C**

1. Find objective function values for each particle (fx) and the number of constraint violations for each.

2. find all the particles whose function vales are less than the personal bests, and whose number of constraint violations are less than the personal best violation counts . Update the new personal bests accordingly

3. Check if the minimum of the personal bests is less than the global best, and if it violates less constraints. If it does, update the global best.

4. While iterations < max number of iterations (2000):

rp,rg = random.uniform #shape =SxD

update the particles velocities -> v = w\*v + c1\*rp\*(p - x) + c2\*rg\*(g - x)

update the particles' positions -> x = x + v

Find function values for each particle and the number of constraint violations.

Update the personal bests to all particles whose function values are less than the current personal best’s and who violates less constraints

find the minimum of these, fp\_min, and compare it to the global best fg

if fp\_min < fg, and fp\_min has fewer constraint violations than fg:

stepsize = np.sqrt(np.sum((g - p\_min)\*\*2))

if np.abs(fg - fp[i\_min]) <= minfunc:

print('Stopping search: Swarm best objective change less than minfunc)

return

elif stepsize <= minstep:

print('Stopping search: Swarm best position change less than minstep)

return

else:

update the global best and its number of constraint violations to fp\_min

iterations +1

5. return

**Soft\_M**

1. Find objective function values for each particle (fx) and the magnitude of violations.

2. find all the particles whose function vales are less than the personal bests, and whose magnitude of constraint violations are less than the personal best violation counts . Update the new personal bests accordingly

3. Check if the minimum of the personal bests is less than the global best, and if its magnitude of constraint violations is less. If it does, update the global best.

4. While iterations < max number of iterations (2000):

rp,rg = random.uniform #shape =SxD

update the particles velocities -> v = w\*v + c1\*rp\*(p - x) + c2\*rg\*(g - x)

update the particles' positions -> x = x + v

Find function values for each particle and the magnitude of constraint violations.

Update the personal bests to all particles whose function values are less than the current personal best’s and who violates less constraints

find the minimum of these, fp\_min, and compare it to the global best fg

if fp\_min < fg, and fp\_min has fewer constraint violations than fg:

stepsize = np.sqrt(np.sum((g - p\_min)\*\*2))

if np.abs(fg - fp[i\_min]) <= minfunc:

print('Stopping search: Swarm best objective change less than minfunc)

return

elif stepsize <= minstep:

print('Stopping search: Swarm best position change less than minstep)

return

else:

update the global best and its number of constraint violations to fp\_min

iterations +1

5. return

**Coello**

1.Calculate objective functions, and feasibility, number of constraints violated, and magnitude of the violations for each particle

2. check if:

any feasible particles less than the personal bests, and if there are, update the personal bests to these.

Check if the minimum personal best fp\_min is less than the global best, fg. It it is, update fg to fp\_min.

else:

check the number of constraints violated by each particle.

If there is one single particle with the fewest constraints:

update fp\_min to that particle’s objective function value

else:

find all the particles with a shared least number of constraint violations. Then check the magnitude of their violations. The particles with the lowest magnitude of constraint violations are updated to the personal bests, if their magnitude of constraint violations is less than that of the current personal bests. Update the personal best with the minimum objective function value to fp\_min

3. while the number of iterations < max number of iterations:

rp,rg = random.uniform #shape =SxD

update the particles velocities -> v = w\*v + c1\*rp\*(p - x) + c2\*rg\*(g - x)

update the particles' positions -> x = x + v

repeat steps 1,2

4. if fp\_min < fg and the magnitude of its constraints < that of the global best:

stepsize = np.sqrt(np.sum((g - p\_min)\*\*2))

if np.abs(fg - fp[i\_min]) <= minfunc:

print('Stopping search: Swarm best objective change less than minfunc)

return

elif stepsize <= minstep:

print('Stopping search: Swarm best position change less than minstep)

return

else:

update the global best and its number of constraint violations to fp\_min

iterations +1

5. return

**Static**

1. find the particle fitness values

where: fitness = fx + penalty

penalty =l\*magnitude of constraint violations

2. Update the new personal bests (all new function values < current personal bests).

3. Compare the smallest personal best to the global best, and if its less, update the global best

4. While iterations < max number of iterations (2000):

rp,rg = random.uniform #shape =SxD

update the particles velocities -> v = w\*v + c1\*rp\*(p - x) + c2\*rg\*(g - x)

update the particles' positions -> x = x + v

find the fitness of each particle ->function value+ penalty

update the new personal bests as all fitness values < previous personal bests

find the particle with the smallest fitness fp\_min and compare it to the global best fg

if fp\_min<fg:

stepsize = np.sqrt(np.sum((g – p\_min)\*\*2))

if abs(fg-fp\_min)<=minfunc:

print('Stopping search: Swarm best objective change less than minfunc)

return

elif stepsize <= minstep:

print('Stopping search: Swarm best position change less than minstep)

return

else:

update global best

iterations +1

5. return