**SOURCE CODE**

from collections import Counter

import math

import random

from operator import xor

rule\_gen=[]

v=[]

value=[]

c1=[]

c2 = []

c3 = []

ls1=[]

ls2=[]

count=0

count2=0

new\_lst=[]

lst=[]

lst2=[]

lst3=[]

lst4=[]

lst5=[]

rand1=[]

rand2=[]

rand3=[]

final\_lst=[[],[],[]]

sum=0

sum2=0

v\_lst=[]

v\_lst2=[]

v\_lst3=[]

s=0

random\_weight=[]

generated\_weights=[]

sum\_weights=[]

new\_five\_sets=[]

sum=0

sum\_val=0

sum2\_val=0

output\_of\_phase2=[]

sum\_weights\_newset=[]

new\_five\_sets\_phase3=[]

vi\_lst2=[]

sum\_weights\_newset2=[]

index\_val=0

outputs\_of\_phases=[]

outputs\_of\_phase3=[]

h1\_weights=[]

h2\_weights=[]

mare=0

diff=0

cluster1=[[3,4,4,85,1,805,40,60,100,18,83,1],[5,0,0,86,4,2149,140,94,234,24,208,1],[6,0,0,86,4,2821,97,89,186,38,192,1]]

for i in range(0,12):

value.append(cluster1[0][i])

value.append(cluster1[1][i])

value.append(cluster1[2][i])

value.sort()

min\_val=value[0]

max\_val=value[2]

range\_val=(max\_val-min\_val)/3.0

print "range\_val", range\_val

if cluster1[0][i] >= 0 and cluster1[0][i] <= range\_val:

c1.append("L")

if cluster1[0][i] > range\_val and cluster1[0][i] <= (2 \* range\_val):

c1.append("M")

if cluster1[0][i] > (2 \* range\_val):

c1.append("H")

if cluster1[1][i] >= 0 and cluster1[1][i] <= range\_val:

c2.append("L")

if cluster1[1][i] > range\_val and cluster1[1][i] <= (2 \* range\_val):

c2.append("M")

if cluster1[1][i] > (2 \* range\_val):

c2.append("H")

if cluster1[2][i] >= 0 and cluster1[2][i] <= range\_val:

c3.append("L")

if cluster1[2][i] > range\_val and cluster1[2][i] <= (2 \* range\_val):

c3.append("M")

if cluster1[2][i] > (2 \* range\_val):

c3.append("H")

value \*=0

rule\_gen.append(c1)

rule\_gen.append(c2)

rule\_gen.append(c3)

print rule\_gen

c4=0

for i in range(0,12):

ls2.append([x[i] for x in rule\_gen])

print "\nls2",ls2

for i in range(0,12):

c1=ls2[i].count('H')

c2=ls2[i].count('L')

c3=ls2[i].count('M')

if (c1>1 or c2>1 or c3>1):

print i

for j in range(0,12):

new\_lst =[]

k=j+1

#print "k",k

for i in range(k,12): # i=j+1

if (ls2[j][0]==ls2[j][1] and ls2[i][0]==ls2[i][1]) or (ls2[j][0]==ls2[j][2] and ls2[i][0]==ls2[i][2]):

new\_lst.append(ls2[j][0])

new\_lst.append(ls2[i][0])

#print "lst",new\_lst

elif (ls2[j][1]==ls2[j][2] and ls2[i][1]==ls2[i][2]):

new\_lst.append(ls2[j][1])

new\_lst.append(ls2[i][2])

lst.append(new\_lst)

print lst

lst2.append(lst[0][0])

for i in range(0,10):

lst2.append(lst[i][1])

lst3.append(lst[0][2])

lst3.append(lst[1][2])

for i in range(0,9):

lst3.append(lst[i][3])

lst4.append(lst[0][4])

lst4.append(lst[1][4])

lst4.append(lst[2][4])

for i in range(0,8):

lst4.append(lst[i][5])

#lst2.append(lst[][1])

#print lst2

#print lst3

#print lst4

lst2=[i.replace("H",'3')for i in lst2]

lst2=[i.replace("L",'1')for i in lst2]

lst2=[i.replace("M",'2')for i in lst2]

#print lst2

lst3=[i.replace("H",'3')for i in lst3]

lst3=[i.replace("L",'1')for i in lst3]

lst3=[i.replace("M",'2')for i in lst3]

#print lst3

lst4=[i.replace("H",'3')for i in lst4]

lst4=[i.replace("L",'1')for i in lst4]

lst4=[i.replace("M",'2')for i in lst4]

#print lst4

lst5.append(lst2)

lst5.append(lst3)

lst5.append(lst4)

print "\n"

print lst5

for i in range(0,12):

#b=random.uniform(0,1)

rand1.append(format(random.uniform(0,1), '.1f'))

print "rand1",rand1

for i in range(0,12):

#b=random.uniform(0,1)

rand2.append(format(random.uniform(0,1), '.1f'))

print "rand2",rand2

print "lst2",lst2

for i in range(0,11):

mul= float(lst2[i]) \* float(rand1[i])

#print "mul",mul

sum +=mul

print "sum",sum

for i in range(0,11):

mul2= float(lst2[i]) \* float(rand2[i])

#print "mul",mul2

sum2 +=mul2

print "sum",sum2

rand3.append(format(random.uniform(0,1), '.1f'))

rand3.append(format(random.uniform(0,1), '.1f'))

for i in range (0,12):

random\_weight.append(rand1[i])

for i in range (0,12):

random\_weight.append(rand2[i])

for i in range (0,2):

random\_weight.append(rand3[i]) #random\_weights denotes 1st set of weights

mul3= sum \* float(rand3[0])

mul4= sum \* float(rand3[1])

print "H1",mul3

print "H2",mul4

final\_val= mul3 + mul4

print "final\_val",final\_val # estimated effort value

generated\_weights.append(random\_weight)

for sets in range(0,4):

v\_lst=[]

for i in range(0,26):

v=[]

si=format(random.uniform(-1,1), '.1f')

l=random.choice(lst2)

d=float(random\_weight[i]) - float(l)

v=format(float(random\_weight[i])+ (float(si)\* float(d)),'.1f')

v\_lst.append(v)

generated\_weights.append(v\_lst)

print "random weight",random\_weight

print "\n\nweight sets",generated\_weights #output of phase1

for setno in range(0,5):

s=0

for i in range(0,25):

s+=float(generated\_weights[setno][i])

sum\_weights.append(format(s,'.1f' ))

print "tot",sum\_weights

for newlst in range(0,5):

v\_lst2=[]

for i in range(0,26):

f\_src=format(float(generated\_weights[newlst][i])/float(sum\_weights[newlst]),'.1f') #ABC-phase2

v\_lst2.append(f\_src)

new\_five\_sets.append(v\_lst2)

print "\nanother five set of weights: ",new\_five\_sets #additional five sets generated in phase2

for setno in range(0,5):

s=0

for i in range(0,26):

s+=float(new\_five\_sets[setno][i])

sum\_weights\_newset.append(format(s,'.1f' ))

print "tot2",sum\_weights\_newset

for setno in range(0,5):

if (sum\_weights[setno] < sum\_weights\_newset[setno]):

output\_of\_phase2.append(generated\_weights[setno])

else:

output\_of\_phase2.append(new\_five\_sets[setno])

outputs\_of\_phases.append(new\_five\_sets[setno])

print "\noutput\_of\_phase2\n",output\_of\_phase2 #phase2 output

for newlst in range(0,5):

vi\_lst2=[]

for i in range(0,25):

sigma=format(random.uniform(0,1), '.1f')

Mu=format(random.uniform(0,1), '.1f')

k=1

alpha=1

pow\_value=float((Mu\*k))

sigma\_s = float(sigma)\*\*pow\_value

result\_xor=xor(bool(alpha), bool(sigma\_s))

mit= float(output\_of\_phase2[newlst][i]) + float(result\_xor) #MCS-phase3

vi\_lst2.append(mit)

new\_five\_sets\_phase3.append(vi\_lst2)

outputs\_of\_phases.append(vi\_lst2)

print "\nanother five set of weights: ",new\_five\_sets #additional five sets generated in phase3

for setno in range(0,5):

s=0

for i in range(0,26):

s+=float(output\_of\_phase2[setno][i])

sum\_weights\_newset2.append(format(s,'.1f' ))

#print "tot2",sum\_weights\_newset2

for setno in range(0,5):

s=0

for i in range(0,26):

s+=float(new\_five\_sets[setno][i])

sum\_weights\_newset2.append(format(s,'.1f' ))

print "tot2",sum\_weights\_newset2

min\_val= min(sum\_weights\_newset2)

print min\_val

for position in range(0,10):

if (sum\_weights\_newset2[position]==min\_val):

index\_val=position

break

print index\_val

outputs\_of\_phase3=outputs\_of\_phases[index\_val]

print outputs\_of\_phase3

#Sigma(s) = Sigma(0)^(-Mu\*k)

'''Where, Sigma(0), (-Mu) - represents the constants [Hint: Random values between 0 and 1]

K - symbolizes the current generation

i.e., value of k is starting from 1 and it increments by 1 for every iteration.'''

for i in range(0,12):

h1\_weights.append(outputs\_of\_phase3[i])

for i in range(12,24):

h2\_weights.append(outputs\_of\_phase3[i])

print h1\_weights

print h2\_weights

for i in range(0,11):

mul\_val= float(lst2[i]) \* float(h1\_weights[i])

#print "mul",mul

sum\_val +=mul\_val

#print "sum",sum\_val

for i in range(0,11):

mul2\_val= float(lst2[i]) \* float(h2\_weights[i])

#print "mul",mul2

sum2\_val +=mul2

#print "sum",sum2\_val

while (diff<3.0):

h1\_process=float(outputs\_of\_phase3[25])\*float(sum\_val)

h2\_process=float(outputs\_of\_phase3[25])\*float(sum2\_val)

estimated\_effort=float(h1\_process)+float(h2\_process)

estimated\_effort\*=1000

diff=abs(estimated\_effort-5152)

print "\nEstimated Effort",abs(estimated\_effort)

actual\_effort=5152

mare=(abs(estimated\_effort-actual\_effort))/actual\_effort

print "MARE",float(mare)