Q2 Final version (R code)

#decimal: generate a series of 14 binary numbers

decimal=function(x){

a=0

for (i in 1:length(x)){

a=a+(x[length(x)+1-i]\*(2^(i-1)))}

a}

#gfunction: convert binary numbers into decimal numbers with 4 decimal places

gfunction=function(x){

a=decimal(x)

(b=round(a\*10/(2^14-1),4))

}

#ppcode: generate an initial population

ppcode=function(x){

B=matrix(0,10\*x,14)

for(i in 1:x){

A=matrix(0,10,14)

while (prod(apply(A,1,gfunction))<0.75){

A=matrix(round(runif(140)),10,14)}

B[(10\*i-9):(10\*i),]=A}

B}

#fitness: calculate the fitness

fitness=function(x){

a=abs((sum(cos(x)^4))-2\*prod(cos(x)^2))/sqrt(sum(c(1:10)\*x^2))

}

#rangroup & rangroup6: tournament selection

rangroup=function(y){

a=c()

x=sample(y)

for (i in 1:7){

a=c(a,max(x[(3\*i-2):(3\*i)]))}

a}

rangroup6=function(y){

a=c()

for (i in 1:6){

a=c(rangroup(y),a)}

a}

#Nextgen: crossover

Nextgen=function(y,Population){

B=matrix(0,210,14)

for (i in 0:20){

x=y[(2\*i+1):(2\*i+2)]

for(j in 1:10){

B[10\*i+j,]=c(Population[(x[1]-1)\*10+j,][1:4],Population[(x[2]-1)\*10+j,][5:10],Population[(x[1]-1)\*10+j,][11:14])

}

}

B}

#Nextgen3: Roulette wheel selection

Nextgen3=function(r,Population){

B=matrix(0,10,14)

x=sample(c(1:21),size=2,prob=2\*c(1:21)/(21\*22),replace=F)

y=match(x,floor(rank(r)))

for(j in 1:10){

B[j,]=c(Population[(y[1]-1)\*10+j,][1:4],Population[(y[2]-1)\*10+j,][5:10],Population[(y[1]-1)\*10+j,][11:14])

}

B}

#Mutation: mutation rate=0.02

Mutation=function(matrix){

for (i in 1:nrow(matrix)){

for (j in 1:ncol(matrix)) {

a=runif(1)

if (a<0.02) {

if (matrix[i,j]==1){

matrix[i,j]=0}

else {if (matrix[i,j]==0){

matrix[i,j]=1}}

}

}

}

matrix}

#check: check the next generation has missing value or not

check=function(Q,W){

ma=apply(Q,1,gfunction)

F=matrix(0,10,14)

while(is.na(sum(ma))){

y=match(NA,ma)

F=W[y:(y+9),]

F=Mutation(F)

Q[c(y:(y+9)),]=F

ma=apply(Q,1,gfunction)}

Q}

#Finstep: check the product of x1 to x10 of the next generation is larger than or equal to 0.75 or not

Finstep=function(matrix1,matrix2,r,matrix3){

matris=check(matrix1,matrix2)

mat=apply(matris,1,gfunction)

for (i in 1:21){

while (prod(mat[(10\*i-9):(10\*i)])-0.75<0){

matris[(10\*i-9):(10\*i),]=Nextgen3(r,matrix3)

matris=check(matris,matrix2)

mat=apply(matris,1,gfunction)

}

}

matris}

#Retur: find out the maximum result and its corresponding x1 to x10

Retur=function(matrix){

x=apply(matrix,1,fitness)

b=match(max(x),x)

a=matrix[b,]

a}

#GA: huge loop to repeat the procedures for n times

GA=function(x){

matr=matrix(0,x,10)

Population=ppcode(21)

for (i in 1:x){

PREAL=apply(Population,1,gfunction)

PREALMATRIX=matrix(PREAL,21,byrow=T)

r=apply(PREALMATRIX,1,fitness)

B=sample(rangroup6(floor(rank(r))))

A=match(B,floor(rank(r)))

D=Nextgen(A,Population)

S=D

E=Mutation(D)

Population=Finstep(E,S,r,Population)

PREAL=apply(Population,1,gfunction)

PREALMATRIX=matrix(PREAL,21,byrow=T)

matr[i,]=Retur(PREALMATRIX)}

Fit=apply(matr,1,fitness)

(outcome=cbind(matr,Fit))}

#Result: return the maximum and its corresponding x1 to x10 among n generations

Result=function(matrix){

D=max(matrix[,ncol(matrix)])

E=match(D,matrix[,ncol(matrix)])

(F=matrix[E,])

}

Result(GA(1000))