Algoritmo genético - Problema de la mochila

Tatiana Guzmán
Saira Salazar
Hernan Iglesias

Partiendo del código dado se establecen los siguientes errores por los cuales no estaría realizando de forma correcta la implementación de algoritmo genético:

- Cruce. Como se puede observar siempre corta en el mismo punto de corte, ya que está predefinida. Solo usa la probabilidad de cruce para una condición, pero no se verifica el número aleatorio para validar el punto de cruce, además está planteando que tienen 7 variables.

```
def cruce(a1,p1,p2):
    if a1<Pcruce:
      print("Mas grande", Pcruce, "que ", a1, "-> Si Cruzan")
      #Punto de cruce no es aleatorio
      temp1=p1[0:3] #[i:j] corta desde [i a j)
      temp2=p1[3:6]
      print(temp1,temp2)
      temp3=p2[0:3]
      temp4=p2[3:6]
      print(temp3,temp4)
      hijo1 = list(temp1)
      hijo1.extend(list(temp4))
      hijo2 = list(temp3)
      hijo2.extend(list(temp2))

    else:
      print("Menor", Pcruce, "que ", a1, "-> NO Cruzan")
      hijo1=p1
      hijo2=p2

    return hijo1,hijo2
```

Realizamos las siguientes modificaciones para cruce:

Cruce recibe 4 valores, a1 que es la probabilidad de corte, a2 que es el punto de cruce, p1, y p2, que son los dos hijos recibidos para verificar y realizar el corte, y x que es el tamaño del cromosoma. Primero se saca el punto de cruce teniendo en cuenta la cantidad de cromosomas que puede ser cualquiera, luego se verifica si se realiza el cruce o no, si se realiza entra a un ciclo while, que se repite las veces que sea necesario hasta que a2 que es el punto de cruce, sea menor que el valor donde debe realizarse el cruce, de ahí se toma la lista con los valores desde el punto 0 hasta el i, que es dependiendo donde se genere el cruce, de i a x (tamaño del cromosoma) con ambos hijos, y realiza el cruce y los vuelve a unir en una misma lista, o en caso de que

no haya cruce entre ambos cromosomas, se informa que no hay corte y se retornan los hijos sin cambios, como se ve en la imagen inferior.

```
def cruce(a1,p1,p2,a2,x):
    cruces = 1/(n-1)
    #print("Cruces: ", cruces)
    print("a2 es: ", a2)
    if a1<Pcruce:
        print("Mas grande", Pcruce, "que ", a1, "-> Si Cruzan")
        #Error en cruce siempre fijo
        print("_____________________________________________________")
        print("ENTRO AL CRUCE ")
      #Reemplace por x para que sea mas general en caso que nos
      #den un valor mayor a 4
        s = 0
        i = 0
        while s == 0: #0,89
          i += 1
          if a2<cruces: #0,33
            # print("Cruce entre ", i, "y ", i+1)
            temp1=p1[0:i]
            temp2=p1[i:x]
            temp3=p2[0:i]
            temp4=p2[i:x]
            hijo1 = list(temp1)
            hijo1.extend(list(temp4))
            hijo2 = list(temp3)
            hijo2.extend(list(temp2))
            print("YA LOS CRUZO Y EL RESULTADO ES")
            print(hijo1)
            print(hijo2)
            i = x
            return hijo1,hijo2
          else:
              cruces += 1/(n-1)
```

- Mutación. No se encuentra definido el método mutación, solo presenta el parámetro de mutación. Por tanto, no se realiza la suma de pesos para que cumpla con la condición.

```
89    Pmuta=0.1   #Probabilidad de Mutación
90
```

Realizamos las siguientes modificaciones para mutación:

Se define la función mutación que recibe 3 valores, x (tamaño del cromosoma), p (hijo a mutar), Pmuta (probabilidad de mutación), se genera una lista con los valores de mutación para cada uno de los genes, luego se realiza un ciclo for en el cual se

verifica para cada uno de los genes la mutación y se realiza si es él caso, y por último se retorna el hijo.

```
#En esta función se realiza la mutación
def mutacion (x,p,Pmuta):
    #print("ENTRO A MUTAR")
    valores=[]
    for i in range(0,x):
      valores.append(np.random.rand())
    for i in range(0,x):
      if valores[i]<Pmuta:
        if p[i]==1:
          p[i]=0
        else:
          p[i]=1
    return p
```

- Validación de peso. Se verifica que el sistema no contaba con una función que calculara el peso y verificara si el hijo es válido para tomar o si se le deba dar pena de muerte.
  Realizamos las siguientes modificaciones para validar el hijo:

  Se define la función verificacromosoma que recibe dos valores, el hijo a evaluar y la lista de pesos, esta función verifica si el hijo sirve o no sirve teniendo en cuenta que su peso no debe superar el peso máximo dado. Para esta función primero se pone un contador, luego se recorre el hijo sumando sus pesos y por último si este no supera el peso se retorna True, que el hijo cumple con la condición de peso, o se devuelve False y un mensaje que informa que se debe dar pena de muerte.

```
#En esta función se realiza la verificación de la restricción de peso
def verificacromosoma(hijo, pesos):
  contador=0
  for i in range(0,x):
    contador+=hijo[i]*pesos[i]
  if contador <=pesmax:
    return True
  else:
    print("Se da pena de muerte a: ",hijo)
    return False

def calpeso(hijo, pesos):
  contador=0
  for i in range(0,x):
    contador+=hijo[i]*pesos[i]
  return contador
```

- Hijo con pena de muerte. Se determina que como no se realizaba el calculo de si un hijo merecia o no pena de muerte tampoco se consideraba si de debía recalcular.

```
# Crear vector de 5x2 vacio  a = numpy.zeros(shape=(5,2))
for iter in range(1):
  print("\n","Iteración ", iter+1)

  for i in [0,2]:  ## Para el bloque de 2 hijos cada vez
    papa1=seleccion(acumulado) # Padre 1
    print("padre 1:", papa1)
    papa2=seleccion(acumulado) # Padre 2
    print("padre 2:", papa2)

    hijoA,hijoB=cruce(np.random.rand(),papa1,papa2)
    print("hijo1: ", hijoA)
    poblIt[i]=hijoA
    print("hijo2: ", hijoB)
    poblIt[i+1]=hijoB
```

Realizamos los siguientes cambios para las iteraciones:

Para las iteraciones se cambia el ciclo for, y se pone un ciclo while que se va a realizar las veces que el usuario desee pero que además, cada vez que un hijo merece pena de muerte este se va incrementar una vez más, repitiendo el proceso para asegurar que sesustituira el hijo al que se le dio pena de muerte, además de tener una variable adicional que contará la cantidad de iteraciones realizadas en su totalidad teniendo encuenta todas las veces que se debe repetir para sacar un hijo nuevamente.

```
iter = 0
niter = 0
while iter < (1):
  niter += 1
  print("\n","Iteración ", iter+1)
  iter += 1
  for i in [1]:  ## Para el bloque de 2 hijos cada vez
    # Para la prueba habiamos puesto un valor fijo de escoje
    # lo volvi a poner aletorio par que cada vez se escoja un padre diferente
    papa1=seleccion(acumulado) #escoje = 0.0404) # Padre 1
    print("padre 1:", papa1)
    papa2=seleccion(acumulado) #escoje = 0.5121) # Padre 2
    print("padre 2:", papa2)

    hijoA,hijoB=cruce(np.random.rand(),papa1,papa2, np.random.rand(),x)
    print("hijo1: ", hijoA)
    #poblIt[i]=hijoA
    print("hijo2: ", hijoB)
```

```
valor1=mutacion(x,hijoA,Pmuta)
hijoA=valor1
#print("ESTO ES VALOR1: ", valor1,x)
poblIt[i]=hijoA
#print("ESTO ES POBLIT I: ", poblIt[i])
valor2=mutacion(x,hijoB,Pmuta)
hijoB=valor2
poblIt[i+1]=hijoB
if verificacromosoma(hijoA,pesos) and verificacromosoma(hijoB,pesos):
    print("El peso de A es: ",calpeso(hijoA, pesos))
    print("El peso de B es: ",calpeso(hijoB, pesos))
elif verificacromosoma(hijoA,pesos) == False and verificacromosoma(hijoB,pesos) == False:
  print("Se da pena de muerte al hijo A: ",hijoA)
  print("Se da pena de muerte al hijo B: ",hijoB)
  iter -= 1
elif verificacromosoma(hijoB,pesos) == False or verificacromosoma(hijoA,pesos) == False:
    if verificacromosoma(hijoA,pesos) == False:
      print("Se da pena de muerte al hijo A: ",hijoA)
      print("El peso de B es: ",calpeso(hijoB, pesos))
      iter -= 1
    else:
      print("El peso de A es: ",calpeso(hijoA, pesos))
      print("Se da pena de muerte al hijo B: ",hijoB)
      iter -= 1
```

Realizamos estos cambio para el tamaño de la población:

Para la creación de la población teniendo en cuenta que el algoritmo puede funcionar para cualquier tamaño de población, se piden los datos del tamaño de la población, además de que como es lógico, según el tamaño el peso también podría variar, además de la posibilidad de ingresar el número de iteraciones requeridas.

```
print("Ingrese el tamaño de la población: ")
n = int(input())
print("Ingrese el tamaño de los cromosomas: ")
x = int(input())
print("Ingrese el peso máximo de la mochila: ")
pesmax = int(input())
print("Ingrese el número de iteraciones: ")
iteraciones = int(input())
```