

REPORTE TAREA #1

INTELIGENCIA

COMPUTACIONAL

línea horizontal

Diego Arnoldo Azuela Rosas - A01208345



**CONTENIDO**

# 

[**1. DESCRIPCIÓN DEL PROBLEMA**](#_1d3blirkvx7b) **3**

[Instrucciones](#_albjccy15t9v) 3

[Interpretación del Problema](#_ciwlzli16y57) 3

[**2. DESCRIPCIÓN DE LA REPRESENTACIÓN**](#_tnnn92xp2u3i) **3**

[Describir el formato en el que se representan las soluciones del problema](#_wjw3hbkbzoul) 3

[Se presenta en el reporte los resultados obtenidos por los algoritmos.](#_h0oay9airwd1) 3

[**3. CONCLUSIÓN Y RETOS ENCONTRADOS**](#_94de9u6grl8a) **10**

# 

# 1. DESCRIPCIÓN DEL PROBLEMA

## Instrucciones

1. Instala Python 3.6.
2. Programa los algoritmos de búsqueda de un padre-varios hijos y varios padres-varios hijos con y sin traslape generacional.
3. Utiliza los algoritmos para encontrar el menor valor de la ecuación y=x^3−2x^2+1 en el intervalo 0≤x≤5.

## Interpretación del Problema

* Se busca que el alumno configure los algoritmos vistos en clase conocidos como algoritmos de búsqueda padre-varios hijos y varios padres-varios hijos. Ambos con y sin traslape generacional. Lo que esto significa es que el alumno deberá trabajar con funciones, para realizar la mutación y la evaluación.

Lo que se busca es crear un valor que se evalúe dentro de una función. A partir de eso se muta el valor utilizado dentro de la función y se vuelve a iniciar el proceso, evaluando el nuevo valor dentro de la función.

# 2. DESCRIPCIÓN DE LA REPRESENTACIÓN

## Describir el formato en el que se representan las soluciones del problema

* El IDE utilizado es Colab de Google. Formato similar al presentado por Jupyter Notebooks, pero con la versatilidad de Google que permite acceder a la información de diferentes puntos y con la seguridad de que todo se ha guardado en la nube. Se presentará el código junto con una foto de la gráfica que produce el código.

## Se presenta en el reporte los resultados obtenidos por los algoritmos.

* UN PADRE - VARIOS HIJOS SIN TRASLAPE

| **CÓDIGO** | |
| --- | --- |
| import random  #random.seed(10)  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #Crear la función a utilizar  def mutar(x):  mutation=x+round(random.triangular(-0.5,0.5),2)  return mutation  def evaluar(x):  valor=round(pow(x,3)-2\*(pow(x,2))+1,2)  return valor  #Crear un dataframe  df=pd.DataFrame()  #Establecer # de generaciones  generaciones = 10  #Generar listas para guardar los valores  padre\_list=[]  hijo\_list=[]  hijo\_evaluado\_list=[]  iter\_list=[]  #Generar los valores iniciales del padre  padre = random.randint(10,50)  #Iniciar ciclo for  for i in range(generaciones):  padre\_list.append(padre) #Agrega el valor del padre en la lista  hijo=mutar(padre)  hijo\_list.append(hijo) #Agrega el valor del hijo en la lista  hijo\_evaluado=evaluar(hijo) #Se evalua el hijo con la ecuacion x^3−2x^2+1 [donde x es el padre]  hijo\_evaluado\_list.append(hijo\_evaluado)    padre=min(hijo\_list) #El valor del padre se cambia por el del menor hijo  iter\_list.append(i+1) #Se lleva un track de la iteracion actual  df['Iteracion']=iter\_list  df['Padre']=padre\_list  df['Hijo']=hijo\_list  df['Hijo Evaluado']=hijo\_evaluado\_list  df.set\_index('Iteracion',inplace=True)  plt.plot(iter\_list,df['Padre'],df['Hijo'])  plt.legend(['Padre', 'Hijo','Hijo Evaluado'])  plt.show() | |
|  |  |

* UN PADRE - VARIOS HIJOS CON TRASLAPE

| **CÓDIGO** | |
| --- | --- |
| import random  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #Crear la función a utilizar  def mutar(x):  mutation=x+round(random.triangular(-0.5,0.5),2)  return mutation  def evaluar(x):  valor=round(pow(x,3)-2\*(pow(x,2))+1,2)  return valor  #Crear un dataframe  df=pd.DataFrame()  #Establecer # de generaciones  generaciones = 10  #Generar listas para guardar los valores  padre\_list=[]  hijo\_list=[]  hijo\_evaluado\_list=[]  iter\_list=[]  #Generar los valores iniciales del padre  padre = random.randint(10,50)  #Iniciar ciclo for  for i in range(generaciones):  padre\_list.append(padre) #Agrega el valor del padre en la lista  hijo\_list.append(padre)  hijo=mutar(padre)  hijo\_list.append(hijo) #Agrega el valor del hijo en la lista  hijo\_evaluado=evaluar(hijo) #Se evalua el hijo con la ecuacion x^3−2x^2+1 [donde x es el padre]  hijo\_evaluado\_list.append(hijo\_evaluado)    padre=min(hijo\_list) #El valor del padre se cambia por el del menor hijo  iter\_list.append(i+1) #Se lleva un track de la iteracion actual  df['Iteracion']=iter\_list  df['Padre']=padre\_list  df['Hijo']=hijo\_list[:-(generaciones)]  df['Hijo Evaluado']=hijo\_evaluado\_list  df.set\_index('Iteracion',inplace=True)  plt.plot(iter\_list,df['Padre'],df['Hijo'])  plt.legend(['Padre', 'Hijo','Hijo Evaluado'])  plt.show() | |
|  |  |

* VARIOS PADRES - VARIOS HIJOS SIN TRASLAPE

| **CÓDIGO** | |
| --- | --- |
| import random  #random.seed(10)  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #Crear la función a utilizar  def mutar(x):  mutation=x+round(random.triangular(-0.5,0.5),2)  return mutation  def evaluar(x):  valor=round(pow(x,3)-2\*(pow(x,2))+1,2)  return valor  #Crear un dataframe  df=pd.DataFrame()  #Establecer # de generaciones  generaciones = 10  #Generar listas para guardar los valores  padre\_list=[]  hijo\_list=[]  hijo\_evaluado\_list=[]  iter\_list=[]  #Iniciar ciclo for  for i in range(generaciones):  #Generar los valores iniciales del padre  padre = random.randint(10,50)  padre\_list.append(padre) #Agrega el valor del padre en la lista  hijo=mutar(padre)  hijo\_list.append(hijo) #Agrega el valor del hijo en la lista  hijo\_evaluado=evaluar(hijo) #Se evalua el hijo con la ecuacion x^3−2x^2+1 [donde x es el padre]  hijo\_evaluado\_list.append(hijo\_evaluado)  for j in range(len(padre\_list)):  padre=min(hijo\_list) #El valor del padre se cambia por el del menor hijo    iter\_list.append(i+1) #Se lleva un track de la iteracion actual  df['Iteracion']=iter\_list  df['Padre']=padre\_list#[:-(generaciones)]  df['Hijo']=hijo\_list#[:-(generaciones)]  df['Hijo Evaluado']=hijo\_evaluado\_list#[:-(generaciones)]  df.set\_index('Iteracion',inplace=True)  plt.plot(iter\_list,df['Padre'],df['Hijo Evaluado'])  plt.legend(['Padre', 'Hijo','Hijo Evaluado'])  plt.show() | |
|  |  |

* VARIOS PADRES - VARIOS HIJOS CON TRASLAPE

| **CÓDIGO** | |
| --- | --- |
| import random  #random.seed(10)  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  #Crear la función a utilizar  def mutar(x):  mutation=x+round(random.triangular(-0.5,0.5),2)  return mutation  def evaluar(x):  valor=round(pow(x,3)-2\*(pow(x,2))+1,2)  return valor  #Crear un dataframe  df=pd.DataFrame()  #Establecer # de generaciones  generaciones = 10  #Generar listas para guardar los valores  padre\_list=[]  hijo\_list=[]  hijo\_evaluado\_list=[]  iter\_list=[]  #Iniciar ciclo for  for i in range(generaciones):  padre\_list.append(random.randint(10,50))  padre\_list.append(padre) #Agrega el valor del padre en la lista  hijo=mutar(padre)  hijo\_list.append(hijo) #Agrega el valor del hijo en la lista  hijo\_evaluado=evaluar(hijo) #Se evalua el hijo con la ecuacion x^3−2x^2+1 [donde x es el padre]  hijo\_evaluado\_list.append(hijo\_evaluado)  padre=min(hijo\_list) #El valor del padre se cambia por el del menor hijo  iter\_list.append(i+1) #Se lleva un track de la iteracion actual  df['Iteracion']=iter\_list  df['Padre']=padre\_list[:-(generaciones)]  df['Hijo']=hijo\_list  df['Hijo Evaluado']=hijo\_evaluado\_list  df.set\_index('Iteracion',inplace=True)  plt.plot(iter\_list,df['Padre'],df['Hijo'])  plt.legend(['Padre', 'Hijo','Hijo Evaluado'])  plt.show() | |
|  |  |

# 3. CONCLUSIÓN Y RETOS ENCONTRADOS

A través del código realizado fue interesante trabajar con funciones. La creación de funciones ha sido algo con lo cual el alumno no se encontraba tan familiarizado. Para el alumno este fue uno de los retos, para hacer un código más eficiente es necesario utilizar funciones. Una vez superado esta barrera quedan los temas alternos que se enfocan en la lógica del programa. Lo que se busca es crear un valor que se evalúe dentro de una función. A partir de eso se muta el valor utilizado dentro de la función y se vuelve a iniciar el proceso, evaluando el nuevo valor dentro de la función.

Uno de los valores presentados en la gráfica y el data frame es conocido en el código como “padre”. Este valor corresponde al valor óptimo encontrado, es decir, aquel que al ser evaluado en la función arroja el menor valor. Una vez que se entiende la asignación se entienden las complicaciones que conlleva.

El alumno se debe asegurar que el valor que se evalúa en es el menor, asegurarse que agarra el valor y no el valor evaluado y pasarlo al padre para ser desplegado. En el caso del alumno por las funciones que se estaban utilizando para evaluar los números más chicos, se complicaba el hecho de evaluar las dos listas y extraer el menor valor, por eso mismo se tomó una sola lista en donde se creaban todos los valores. Sin embargo esto despliega al final inconsistencias. Si se hubiera contado con más tiempo estas inconsistencias no hubieran sido inconveniente alguno.