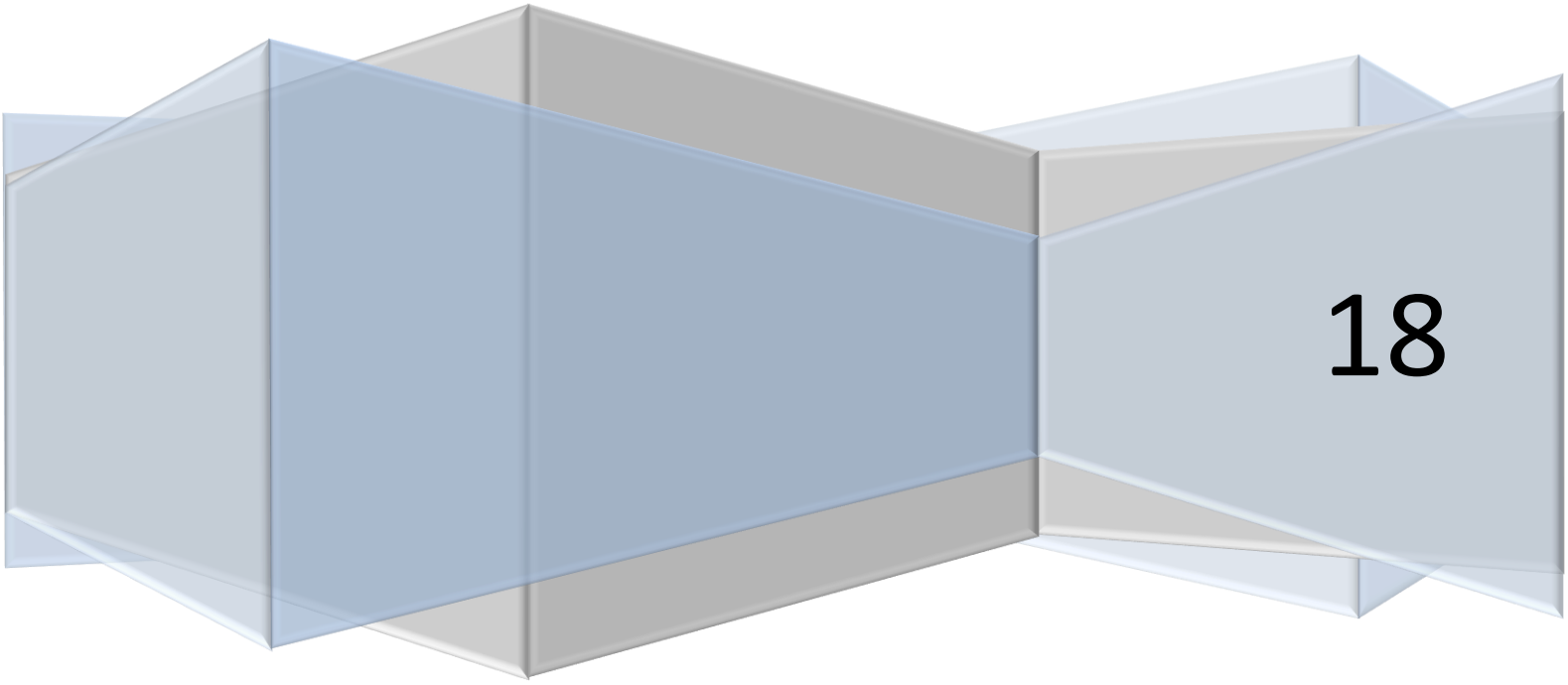
UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL ROSARIO

Trabajo Práctico 1

Algoritmos Geneticos

Ciriaci Santiago, Botello Andrés y Corsetti Ornela



18

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL ROSARIO

Contenido

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL ROSARIO

Enunciado

Hacer un programa que utilice un Algoritmo Genético Canónico para buscar un máximo de la función: $f(x) = (\frac{x}{coef})^2$ en el dominio $[0; 2^{30} - 1]$; donde coef = $2^{30} - 1$.

Teniendo en cuenta los siguientes datos:

1. Probabilidad de Crossover = 0,75
2. Probabilidad de Mutación = 0,05
3. Población Inicial: 10 individuos
4. Ciclos del programa: 20
5. Método de Selección: Ruleta
6. Método de Crossover: 1 Punto
7. Método de Mutación: invertida

El programa debe mostrar, finalmente, el Cromosoma correspondiente al valor máximo obtenido y gráficas, usando EXCEL, de Máx, Mín y Promedio de la función objetivo por cada generación.

Desarrollo de la codificación

Con el objetivo de resolver el enunciado mencionado anteriormente, hemos implementado el lenguaje de programación *"Python"* en su versión 3.6. La decisión de utilizar este lenguaje en particular se debe principalmente a que posee una sintaxis simple y ordenada, con una gran variedad de funciones y librerías que nos resultaron de utilidad en este paradigma de programación. A continuación describiremos brevemente algunas de ellas.

Librería "xlwt"

Es una librería diseñada específicamente para hojas de cálculo de Excel. Esta nos permite mediante su conjunto de funciones, volcar el/los valor/es de las variables que contienen nuestros resultados en una hoja de cálculo creada por el programa.

Posteriormente utilizaremos esta hoja para generar las gráficas solicitadas.

Librería "random"

Esta librería nos permite, dependiendo de la función, generar números aleatorios de diferentes maneras. Utilizamos, principalmente, dos de sus funciones:

1. random(): genera un número aleatorio entre 0 y 1.
2. randint(min, max): genera un entero aleatorio entre los parámetros definidos.

Código

```
#máximo de la función 1073741823
from random import randint
from random import random
import xlwt
wb=xlwt.Workbook() #Crea archivo de excel
ws=wb.add_sheet('AG') #Agregamos hoja
pm=0.05
pc=0.75
n=int(input('Ingrese Numero de corridas.'))
cr=[['' for j in range(10)]for i in range(n+1)]
for j in range(30):        #Crea poblacion inicial
    for i in range(10):
        cr[0][i]+=str(randint(0,1))

tab=[[0 for j in range(3)]for i in range(n+1)]  #Creacion Tabla de Resultados
for c in range(n):
    print(cr[c])
    dec=[] #Tabla con números en decimal
    for i in range(10):
        num= int(str(cr[c][i]),2)
        dec.append(num)
    fit=[[0 for j in range(4)]for i in range(10)] #"Creacion Tabla Fitness"
    sum=0
    for i in range(10): #"Llenado tabla fitness"
        fit[i][0]=str(bin(dec[i])[2:].zfill(30)) #binario
        fit[i][1]=dec[i] #decimal
        fit[i][2]=(fit[i][1]/((2**30)-1))**2 #el decimal evaluado en la función
        sum+=fit[i][2]
    for i in range(10): #"Calculo Probabilidades"
        if round(fit[i][2]*100/sum)==0:
            fit[i][3]=1
        else:
            fit[i][3]=round(fit[i][2]*100/sum)
    print(fit)
    #Llena Tabla
    tab[c][0]=((max(dec))/((2**30)-1))**2 #MÁXIMO
    tab[c][1]=((min(dec))/((2**30)-1))**2 #MÍNIMO
    tab[c][2]=sum/10                  #PROMEDIO
    ws.write(c,0,str(bin(max(dec))[2:].zfill(30)))
    ws.write(c,1,tab[c][0]) #Llenamos filas y columnas de la hoja excel
    ws.write(c,2,tab[c][1])
    ws.write(c,3,tab[c][2])
```

```
   print(tab[c])
   decord=[] #Tabla para Decimales Ordenados
   decord=sorted(dec,reverse=True) #Ordena de mayor a menor dec
#Asigna a la siguiente generacion los dos valores mas altos
   cr[c+1][0]=str(bin(decord[0])[2:].zfill(30))
   cr[c+1][1]=str(bin(decord[1])[2:].zfill(30))
   ws.write(c,4," ")
   ws.write(c,5,cr[c+1][0])
   ws.write(c,6,cr[c+1][1])
   ru=[] #Ruleta
   for i in range(10): #"Carga Ruleta"
      for j in range (fit[i][3]):
         ru.append(fit[i][1])
   for i in range (4): #crossover
      cr1=randint(0, len(ru)-1) #"Selecciona Candidatos"
      cr2=randint(0, len(ru)-1)
      cr1h=str(bin(ru[cr1])[2:].zfill(30)) #Transforma contenido de ruleta en dicha posición a binario
con 30 dígitos
      cr2h=str(bin(ru[cr2])[2:].zfill(30))
      pc1= random() #Consulta probabilidad de Crossover
      cr1h=list(map(int,str(cr1h)))  #Separa los elementos de la lista como un elemento c/u
      cr2h=list(map(int,str(cr2h)))
      if pc1<=pc: #"Crossover"
         gen= randint (0,29)
         aux1=cr1h[gen:30]
         aux2=cr2h[gen:30]
         cr1hijo=cr1h[0:gen]+aux2
         cr2hijo=cr2h[0:gen]+aux1
         cr1h=cr1hijo
         cr2h=cr2hijo
         cr1hijo=("".join(map(str,cr1hijo))) #Junta todos los elementos en un binario de 30
         cr2hijo=("".join(map(str,cr2hijo)))
         cr[c+1][(i*2)+2]=cr1hijo #Le asigna a la prox generacion los hijos
         cr[c+1][(i*2)+3]=cr2hijo
      else:
         cr[c+1][(i*2)+2]=("".join(map(str,cr1h)))    #Si no entra en crossover, se le asignan los padres
         cr[c+1][(i*2)+3]=("".join(map(str,cr2h)))
      pm1=random() #Consulta probabilidad de Mutación
      if pm1<=pm:
         gen= randint(0,29)
         cr1hijo=cr1h
         cr2hijo=cr2h
         if(cr1hijo[gen]==0):
            cr1hijo[gen]=1   #Inserta el elemento y luego borra el que estaba
```

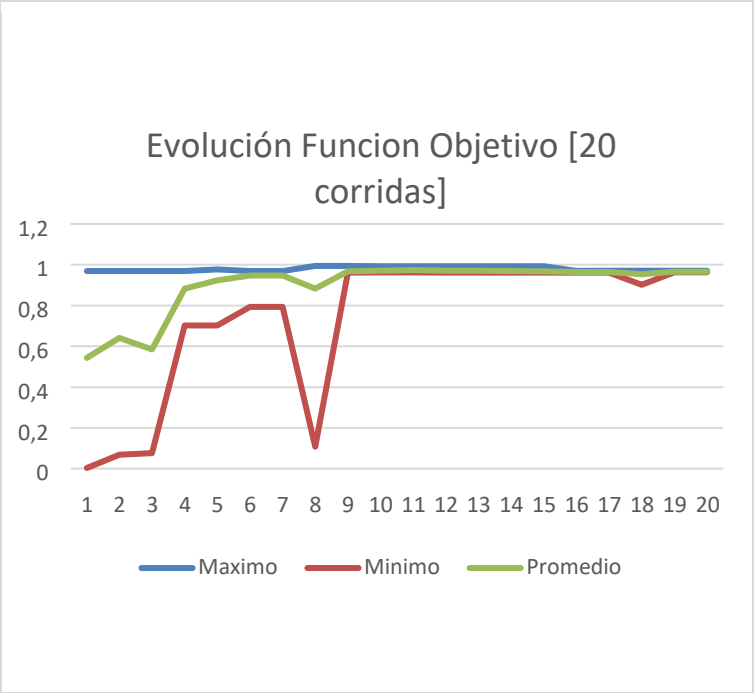
```
        else:
            cr1hijo[gen]=0
        cr1hijo=("".join(map(str,cr1hijo)))    #Junta todos los elementos en un binario de 30
        cr[c+1][(i*2)+2]=cr1hijo
        if(cr2hijo[gen]==0):
            cr2hijo[gen]=1
        else:
            cr2hijo[gen]=0
        cr2hijo=("".join(map(str,cr2hijo)))    #Junta todos los elementos en un binario de 30
        cr[c+1][(i*2)+3]=cr2hijo

wb.save(r'C:\Users\Public\Documents\AG.xls')
```

Análisis del algoritmo habiendo realizado 20 corridas

Probabilidades definidas en el enunciado

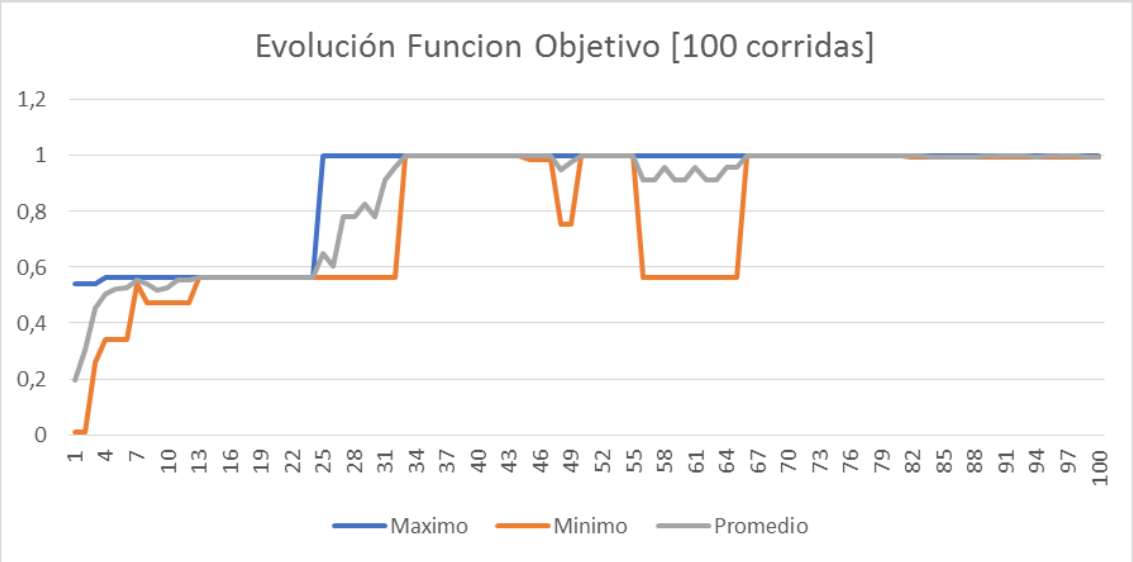
Con estas probabilidades el promedio de las ejecuciones se vio afectada por un promedio de 2 mutaciones, y el promedio final del valor de la función objetivo varió en el intervalo de [0,92;0,97].

MÁXIMO	MÍNIMO	PROMEDIO
0,96977364	0,00308521	0,542975
0,96977364	0,06836317	0,64163776
0,96977394	0,0757333	0,58376245
0,96978896	0,70183048	0,88312886
0,97672314	0,70180161	0,92392286
0,96977394	0,79388646	0,94643394
0,96977394	0,79388646	0,94727301
0,99402654	0,10767139	0,88191885
0,99402654	0,96133532	0,96735707
0,99295596	0,96205358	0,9712493
0,99295596	0,96205358	0,97349382
0,99299279	0,96133532	0,97039636
0,99299279	0,96133532	0,97032454
0,99297948	0,96133532	0,96968456
0,99297948	0,96133532	0,96814763
0,96973168	0,96133532	0,96421294
0,97069358	0,96133532	0,96440539
0,97069264	0,90172676	0,95322693
0,97069358	0,96205264	0,96550844
0,97069358	0,96205264	0,96531663



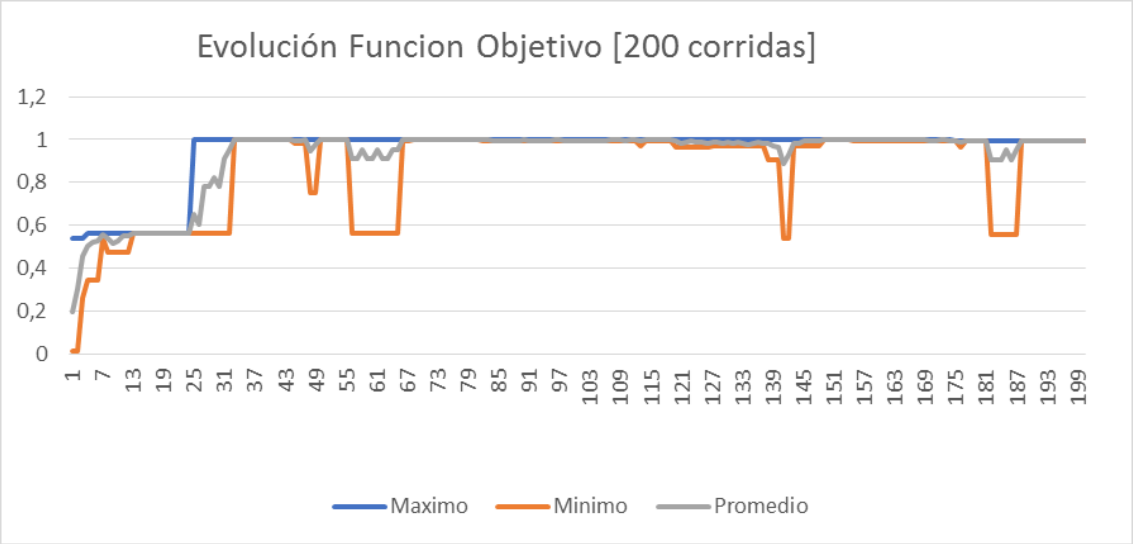
Análisis del algoritmo habiendo realizado 100 corridas

Con estas probabilidades el promedio de las ejecuciones se vio afectada por un promedio de 3 mutaciones, y el promedio final del valor de la función objetivo varió en el intervalo de [0,94;0,98].



Análisis del algoritmo habiendo realizado 200 corridas

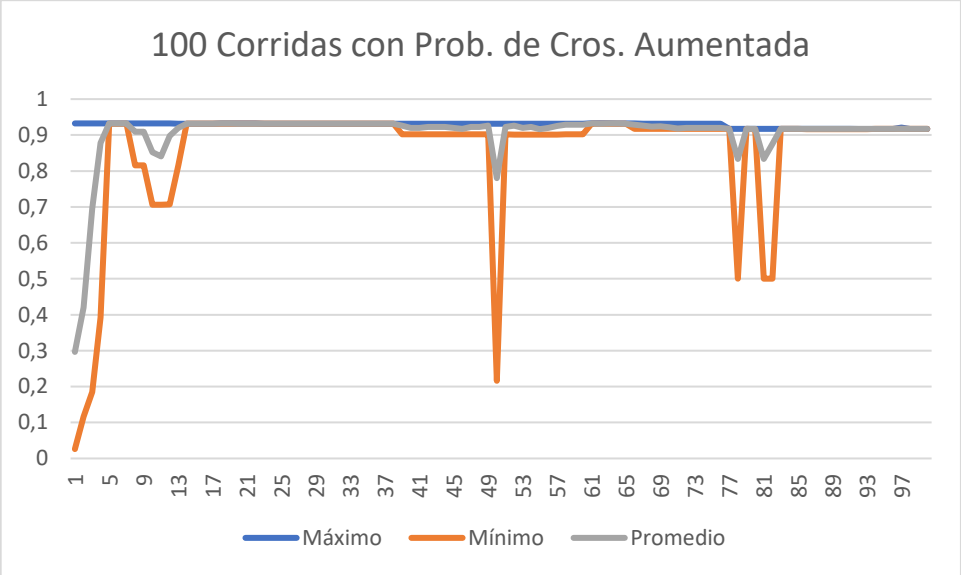
Con estas probabilidades el promedio de las ejecuciones se vio afectada por un promedio de 5 mutaciones, y el promedio final del valor de la función objetivo varió en el intervalo de [0,97;0,99].



Probabilidades alteradas

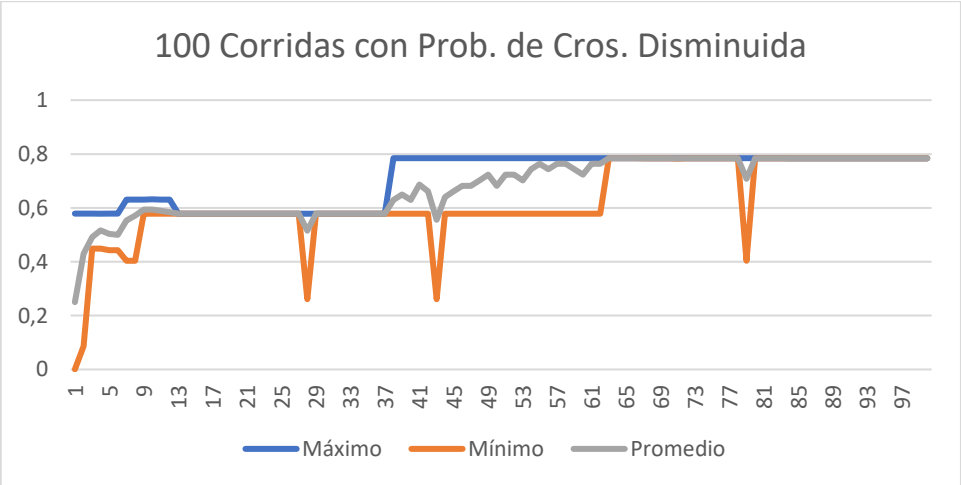
Probabilidad de Crossover aumentada a 0,90

Podemos notar que luego de haberlo ejecutado una cierta cantidad de veces, que los mínimos y máximos convergen a valores similares entre sí en una menor cantidad de corridas.



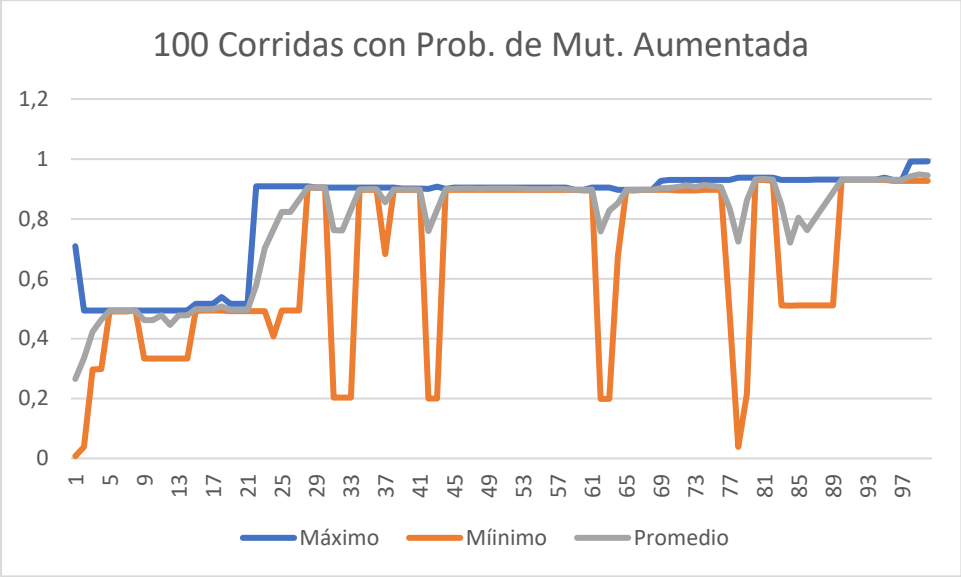
Probabilidad de Crossover disminuida a 0,50

Podemos notar que luego de haberlo ejecutado una cierta cantidad de veces, que los mínimos y máximos convergen a valores similares entre sí en una mayor cantidad de corridas.



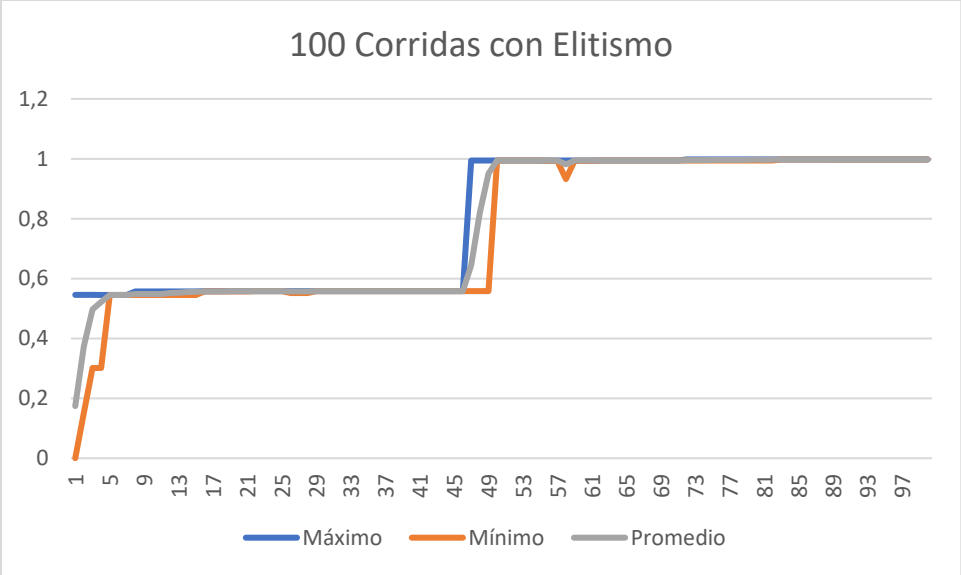
Probabilidad de Mutación aumentada a 0,25

Podemos notar que luego de haberlo ejecutado una cierta cantidad de veces, que el mínimo y el máximo no llegan a la convergencia durante demasiadas corridas.



Ejercicio Con Elitismo

Podemos notar que al agregar elitismo, como mantenemos siempre el valor máximo en la futura corrida, este no disminuye y por lo tanto el resultado suele ser muy próximo a 1 en la mayoría de las ejecuciones.



Conclusiones

Luego de haber ejecutado el algoritmo varias veces con distinta cantidad de corridas, podemos concluir que en la mayoría de los casos los resultados evolucionan desde un valor mínimo tendiendo al máximo valor que puede tomar la función objetivo (“1”).

Podemos notar como una vez ejecutadas muchas corridas si el valor mínimo de la función desciende notablemente en comparación a los anteriores (debido a una mutación) a este le toma muchas menos corridas retornar a los valores más óptimos de la función que cuando iniciamos el algoritmo. Deducimos que esto sucede debido a que el resto de la población, al tener valores óptimos, sustituye con sus hijos al valor mutado.