ALGORITMOS GENÉTICOS
TRABAJO PRÁCTICO 1

"Selección Natural"

Cutró, Gastón	Cabanellas, Ignacio	Benedetti, Juan	Fernández, Natalia	Gofiar, Cristian
44757	44987	45403	44758	44839

2020

Índice general

Capítulo 1

Introduccion

1.1. Introducción

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

En los años 1970, de la mano de John Henry Holland, surgió una de las líneas más prometedoras de la inteligencia artificial, la de los algoritmos genéticos. Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular.

Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Los algoritmos genéticos se enmarcan dentro de los algoritmos evolutivos, que incluyen también las estrategias evolutivas, la programación evolutiva y la programación genética.

Capítulo 2

Definicion del Sistema

2.1. Definición del Sistema

Desarrollamos un software con fines didácticos que encuentra el valor óptimo para la función objetivo dada:

$$f(x) = \left(\frac{x}{2^{30} - 1}\right)^2 \tag{2.1}$$

en el dominio $[0,\ 2^{30} - 1]$.

Se realiza la ejecucion con tipos de Selección, Crossover y Mutación definidos:

- Método de Selección: Ruleta.
- Método de Crossover: 1 Punto.
- Método de Mutación: Invertida.

los datos que proporciona la ejecución del software corresponde al cromosoma de máximo valor obtenido, gráficas del proceso evolutivo y una tabla donde se vuelcan todos los datos de los cromosomas con sus hijos y evoluciones.

Este programa le da libertad al usuario para elegir la cantidad de valores iniciales a generar, la cantidad de ciclos a realizar, la probabilidad de crossover y mutación expresada en porcentajes y si desea realizar la ejecucion con o sin elitismo.

Capítulo 3

Resultados obtenidos

3.1. Resultados

3.1.1. Resultados analíticos

Realizaremos la prueba 2 veces. La primera vez la hacermos **sin elitismo** y la segunda con **elitismo**.

Para los siguientes resultados en tabla se utilizaron los siguiente valores para los parámetros respectivos:

- Probabilidad de Crossover = 0.75
- Probabilidad de mutación = 0.05
- Población inicial = 10 individuos
- Ciclos del programa = 20

Ejecución sin elitismo

Cromosoma máximo obtenido: 1064476994

Generacion	Minimo FO	Maximo FO	Valor cromosoma dec (Maximo)	Valor cromosoma bin (Maximo)	Promedio FO
1	6,74002E-06	0,982817363	1064476994	11111101110010101000010100010	0,286779851
2	0,469974289	0,739938953	923629541	11011100001101011011111100101	0,712942315
3	0,739938696	0,739938964	923629548	11011100001101011011111101100	0,739938831
4	0,739938707	0,739938964	923629548	11011100001101011011111101100	0,739938783
5	0,739518749	0,739938965	923629549	11011100001101011011111101101	0,739896737
6	0,739518749	0,739938707	923629388	11011100001101011011101001100	0,739896712
7	0,739518749	0,739938707	923629388	11011100001101011011101001100	0,73981272
8	0,687152964	0,739938707	923629388	11011100001101011011101001100	0,734618137
9	0,687152964	0,739938707	923629388	11011100001101011011101001100	0,734660133
10	0,636320345	0,739938707	923629388	11011100001101011011101001100	0,729576871
11	0,636320345	0,739938707	923629388	11011100001101011011101001100	0,688129526
12	0,636320345	0,739938707	923629388	11011100001101011011101001100	0,688129526
13	0,636320345	0,739938707	923629388	11011100001101011011101001100	0,708853199
14	0,636320345	0,739938707	923629388	11011100001101011011101001100	0,708853199
15	0,636320345	0,739938707	923629388	11011100001101011011101001100	0,698491362
16	0,587440851	0,739938707	923629388	11011100001101011011101001100	0,683241577
17	0,587440851	0,739938707	923629388	11011100001101011011101001100	0,667601166
18	0,587440851	0,687152964	890074956	11010100001101011011101001100	0,636515657
19	0,636320345	0,687152964	890074956	11010100001101011011101001100	0,661736644
20	0,636320345	0,687152964	890074956	11010100001101011011101001100	0,65157013

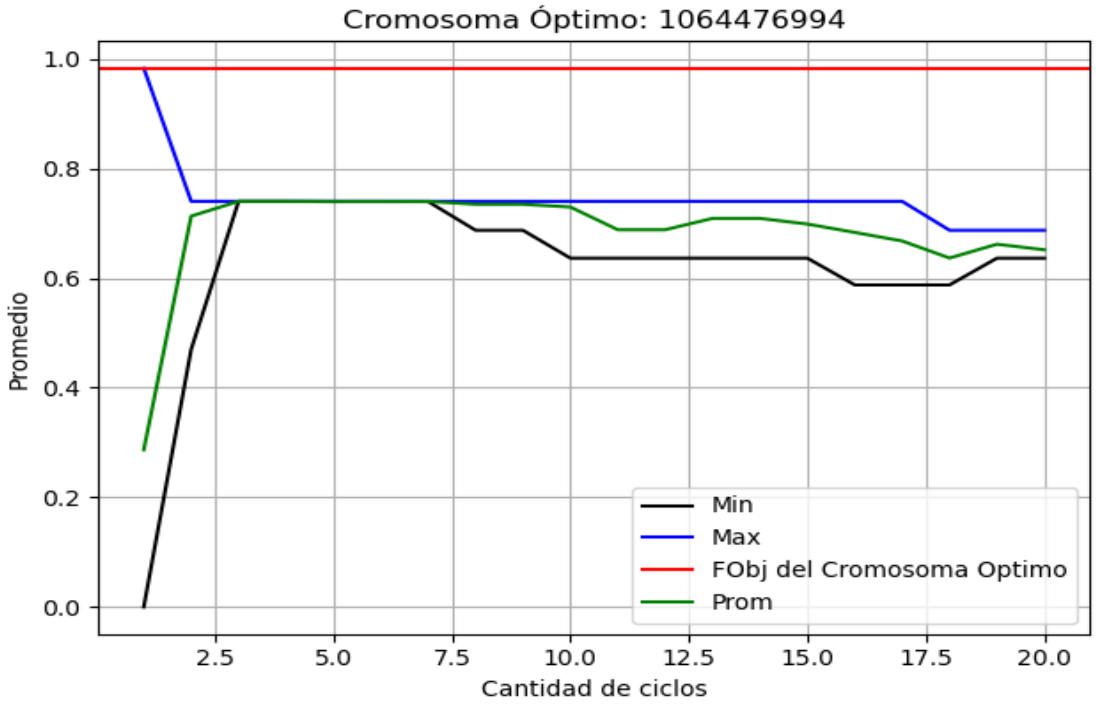
Ejecución con elitismo

Cromosoma máximo obtenido: 1054730763

Generacion	Minimo FO	Maximo FO	Valor cromosoma dec (Maximo)	Valor cromosoma bin (Maximo)	Promedio FO
1	0,005039476	0,881645149	1008200203	11110000010111111010100001011	0,332486882
2	0,198655429	0,881645149	1008200203	11110000010111111010100001011	0,540815994
3	0,199916881	0,881645149	1008200203	11110000010111111010100001011	0,655149207
4	0,199916881	0,881645149	1008200203	11110000010111111010100001011	0,723558483
5	0,199916474	0,881645149	1008200203	11110000010111111010100001011	0,587212711
6	0,199916474	0,882298957	1008573963	11110000011101100111100001011	0,709340859
7	0,192669631	0,897036675	1016962571	11110010011101100111100001011	0,660028929
8	0,199915621	0,897036675	1016962571	11110010011101100111100001011	0,610822653
9	0,192991256	0,897042094	1016965643	11110010011101101010100001011	0,613858748
10	0,192993769	0,904456752	1021159947	11110011011101101010100001011	0,61616423
11	0,192993769	0,904456752	1021159947	11110011011101101010100001011	0,757715361
12	0,192993769	0,964872646	1054714379	11111011011101101010100001011	0,766722813
13	0,192993769	0,964872646	1054714379	11111011011101101010100001011	0,774247334
14	0,192993769	0,964872646	1054714379	11111011011101101010100001011	0,715136258
15	0,192993769	0,964872646	1054714379	11111011011101101010100001011	0,881631179
16	0,232593334	0,964902624	1054730763	11111011011101111010100001011	0,883390881
17	0,232593334	0,964902624	1054730763	11111011011101111010100001011	0,891647713
18	0,232534466	0,964902624	1054730763	11111011011101111010100001011	0,745185964
19	0,232593334	0,964902624	1054730763	11111011011101111010100001011	0,745197846
20	0,232593334	0,964902624	1054730763	11111011011101111010100001011	0,818422592

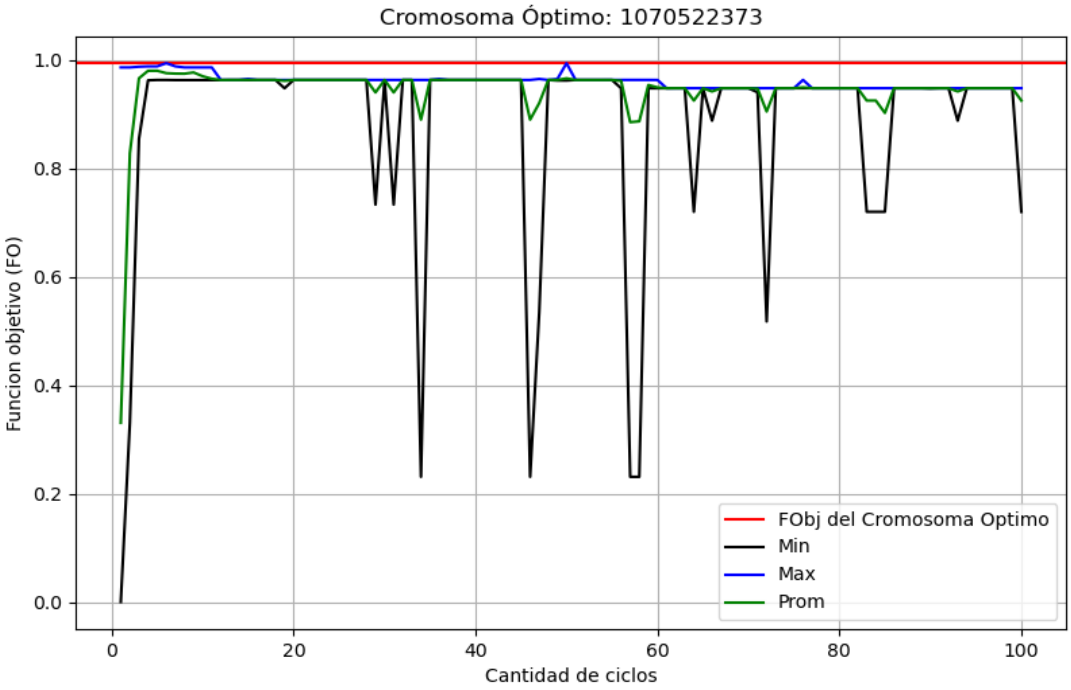
3.1.2. Resultados gráficos

Ejecución sin elitismo



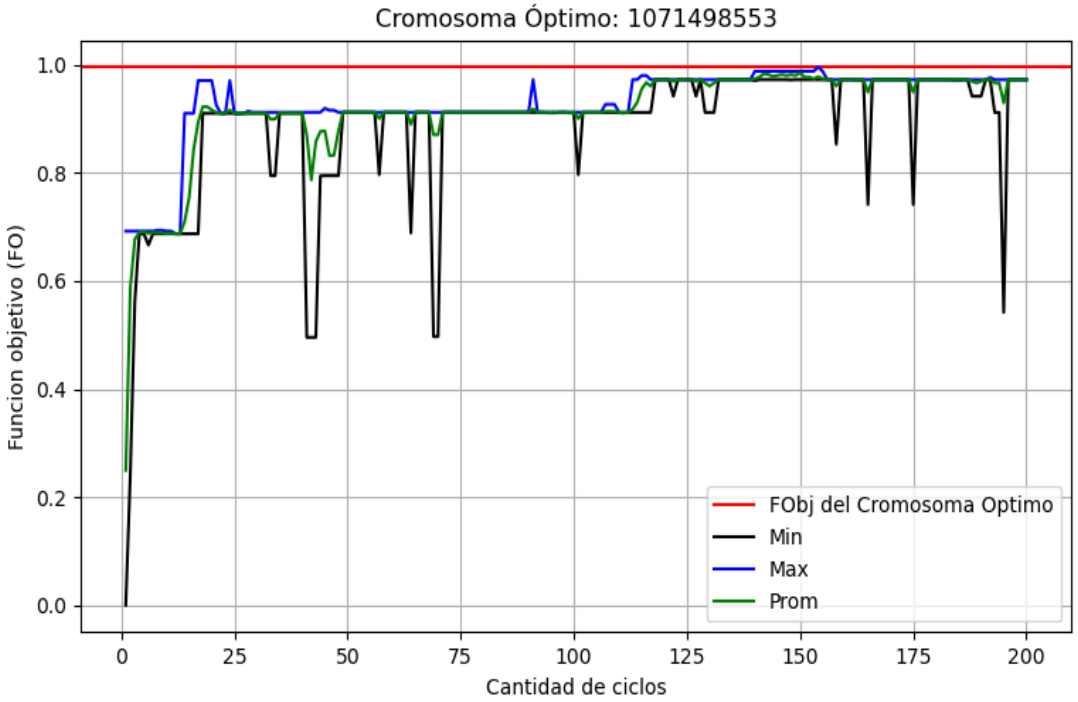
Parámetros:
Probabilidad de Crossover = 0.75
Probabilidad de mutación = 0.05
Población inicial = 10 individuos
Ciclos del programa = 20

En esta gráfica podemos notar que, al ser pocos ciclos y tener una tasa de mutación muy baja, el cromosoma máximo obtenido no se acerca al óptimo esperado. Sin embargo, el resultado es un cromosoma óptimo debido a que al comienzo del programa se generó una población incicial que lo contenía. Pero si se vuelve a correr el programa es muy probable que no ocurra esto. También puede verse que si la tasa de mutación hubiese sido mayor, podrían haberse obtenido mejores resultados.



Parámetros:
Probabilidad de Crossover = 0.75
Probabilidad de mutación = 0.05
Población inicial = 10 individuos
Ciclos del programa = 100

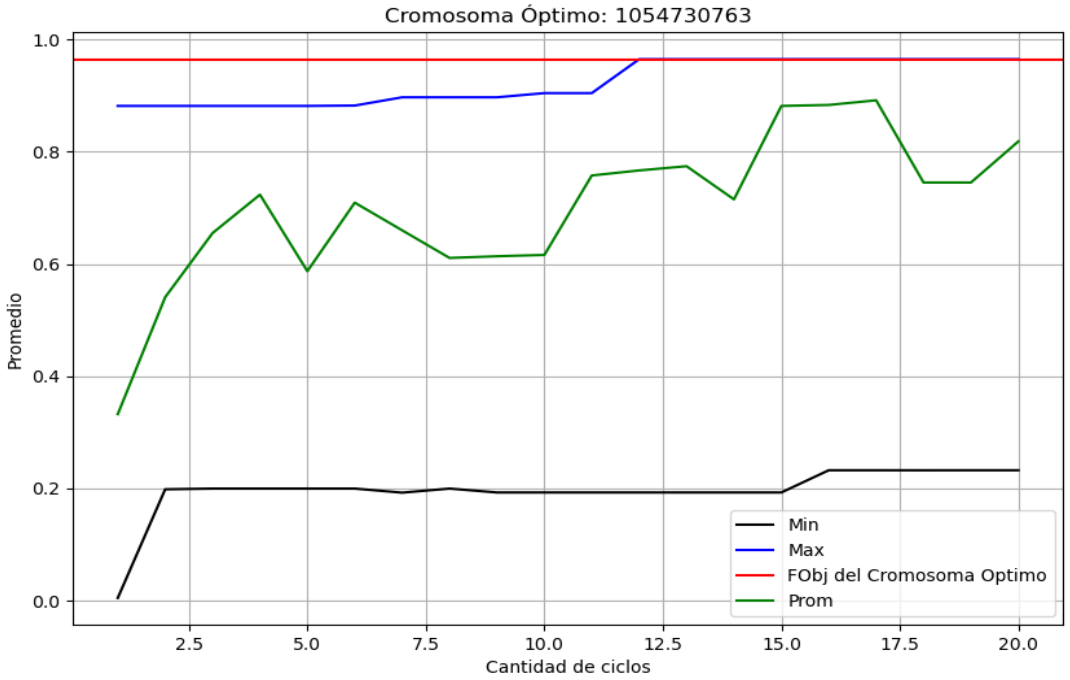
En esta gráfica podemos notar que la población inicial tiene valores muy alejados, pero a medida que van ejecutando los ciclos del programa los valores se vuelven cada vez mas homogéneos y el promedio se acerca al valor óptimo. Al ser la cantidad de ciclos relativamente grande obtenemos algunas mutaciones las cuales se ven reflejadas en la caída de los valores mínimos representadas en la gráfica.



Parámetros:
Probabilidad de Crossover = 0.75
Probabilidad de mutación = 0.05
Población inicial = 10 individuos
Ciclos del programa = 200

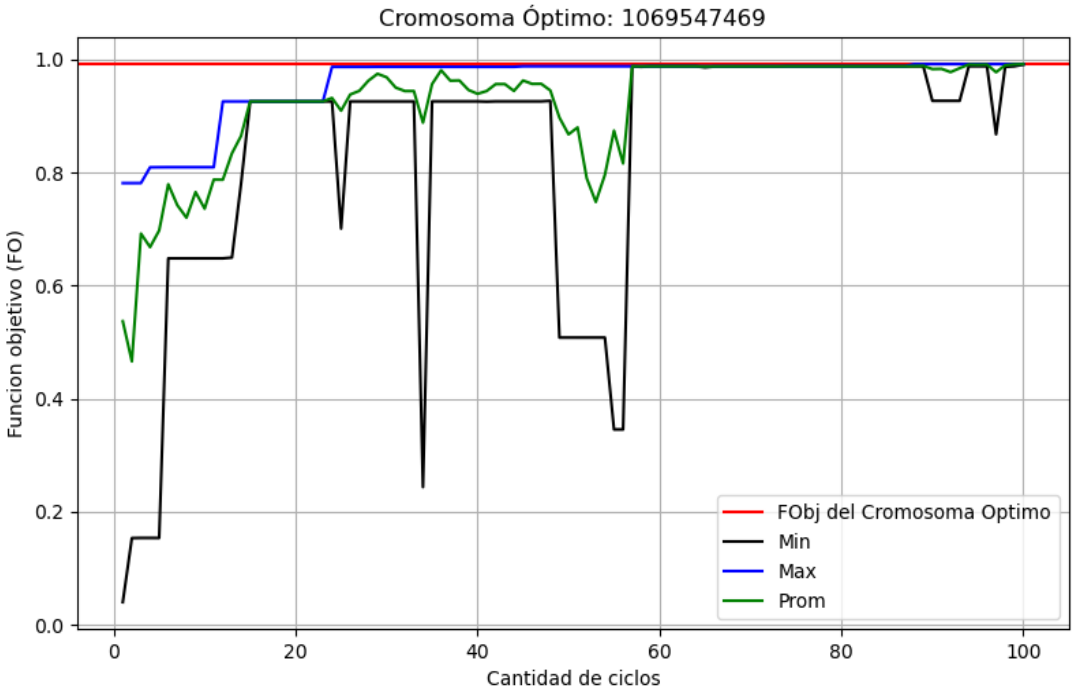
En esta gráfica no obtuvimos una buena población inicial sin embargo el mayor valor obtenido se acerca mas al optimo en comparación de otras gráficas debido al aumento de la cantidad de ciclos y las mutaciones, gracias a esto los cromosomas no se estancaron en un valor fijo sino que a lo largo de toda la ejecución los valores fueron variando y en este caso nos ayudaron a obtener un valor muy cerca del óptimo.

Ejecución con elitismo



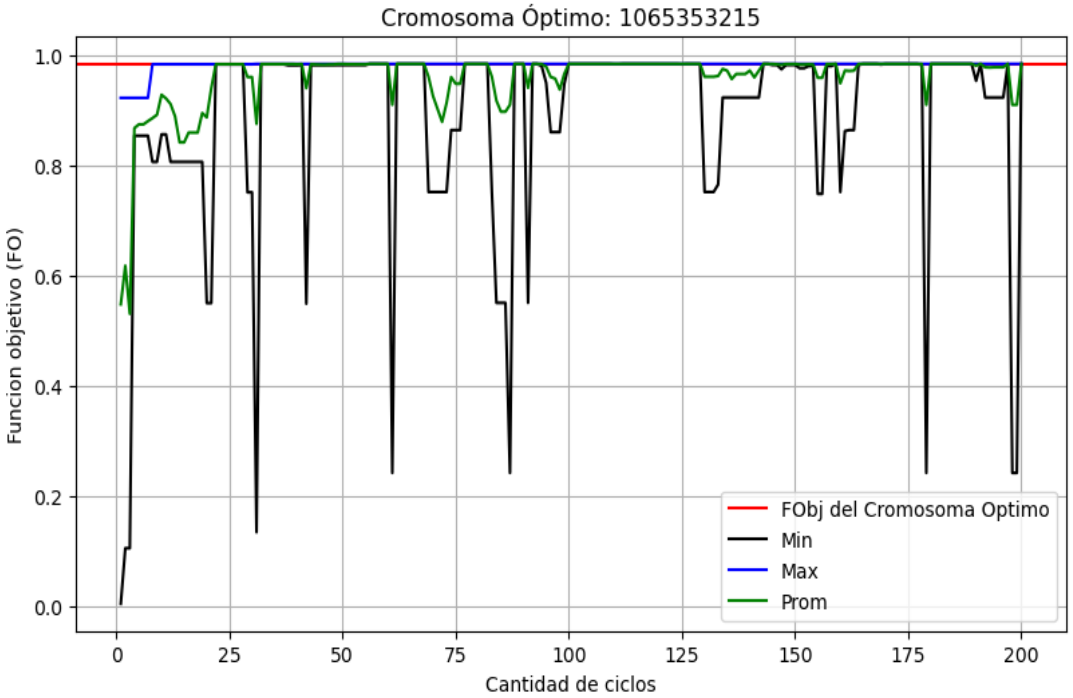
Parámetros:
Probabilidad de Crossover = 0.75
Probabilidad de mutación = 0.05
Población inicial = 10 individuos
Ciclos del programa = 20

En esta gráfica no se tuvieron mutaciones debido a la baja probabilidad y la poca cantidad de ciclos en la ejecución, por lo que la grafica no cambió drasticamente. En la población inicial se obtuvieron valores muy alejados y a lo largo de la ejecución de los ciclos, el cromosoma máximo y mínimo no se acercaron, pero sus valores aumentaron, lo que provocó que el valor promedio aumente.



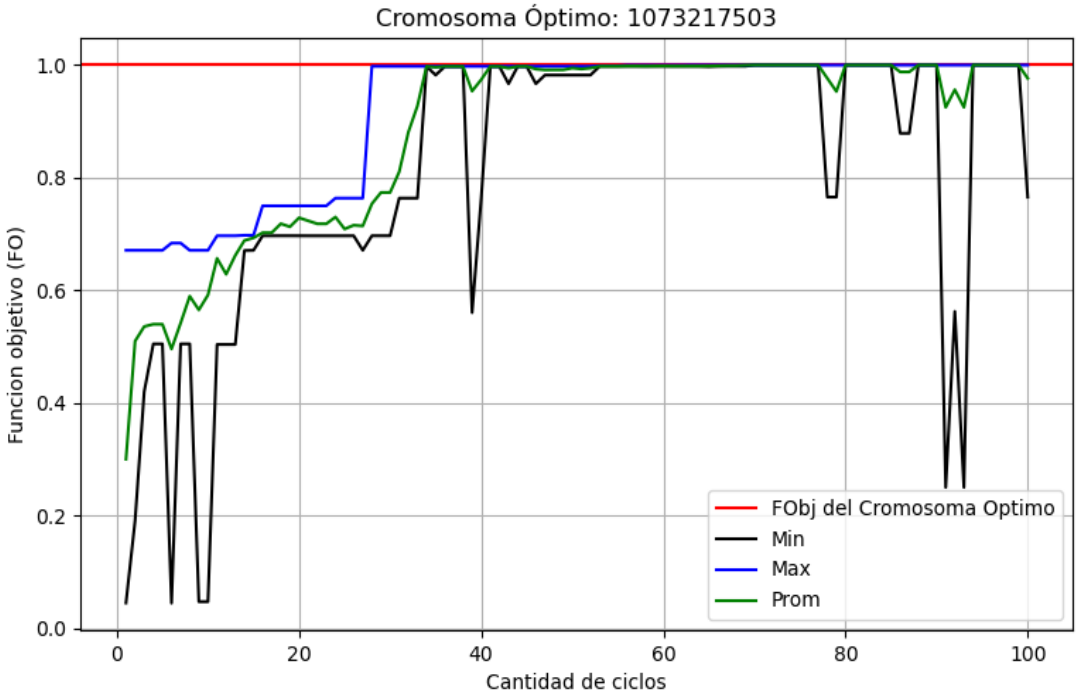
Parámetros:
Probabilidad de Crossover = 0.75
Probabilidad de mutación = 0.05
Población inicial = 10 individuos
Ciclos del programa = 100

Notamos nuevamente como el elitismo hace su trabajo y mantiene los máximos siempre al alza, mientras que los mínimos sufren varias mutaciones muy significativas. Gracias a estas mutaciones notamos las fluctuaciones del promedio.



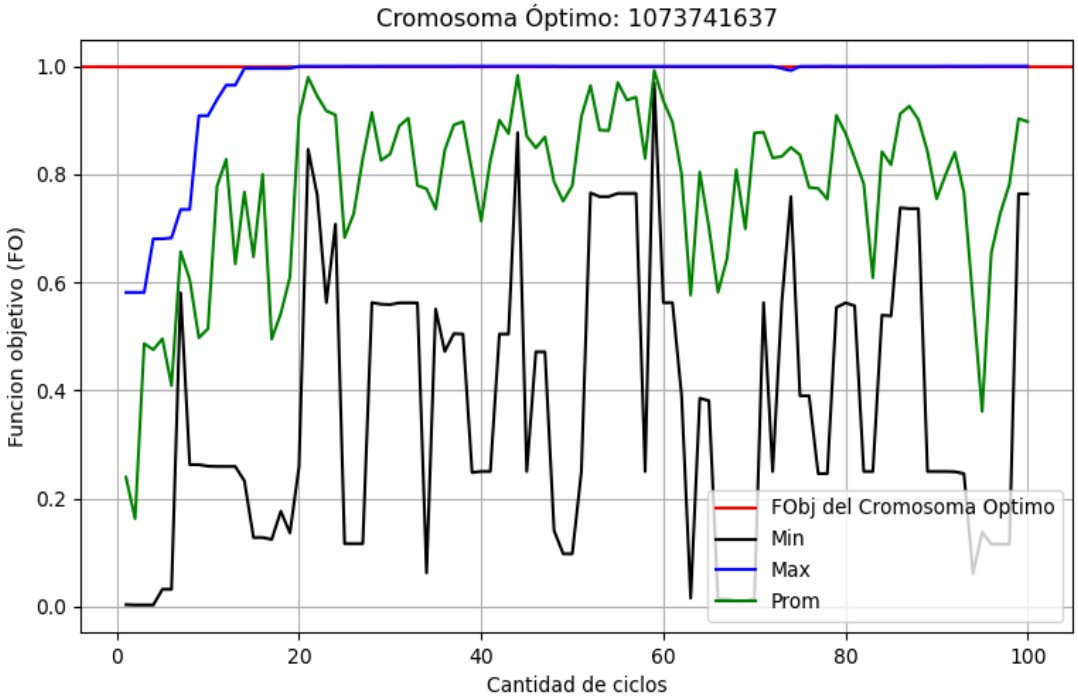
Parámetros:
Probabilidad de Crossover = 0.75
Probabilidad de mutación = 0.05
Población inicial = 10 individuos
Ciclos del programa = 200

En esta gráfica podemos ver que debido al elitismo, si bien encontramos varias mutaciones debido a la cantidad de ciclos del programa, los valores promedio tienden al óptimo sin descender demasiado.



Parámetros:
Probabilidad de Crossover = 0.90
Probabilidad de mutación = 0.10
Población inicial = 10 individuos
Ciclos del programa = 100

En esta grafica se usa una probabilidad de crossover y mutacion un poco mas alta de la recomendada, y se observan cambios continuos en los valores maximos y minimo para reflejar esto, hasta que se aproxima al valor optimo donde se estabiliza por usar el metodo elitista, aun asi se siguen viendo mutaciones ya que no dependen del metodo elitista



Parámetros:
Probabilidad de Crossover = 0.10
Probabilidad de mutación = 0.90
Población inicial = 10 individuos
Ciclos del programa = 100

En esta corrida decidimos probar con una probabilidad de crossover muy baja y puede verse como los valores maximos y minimos no se tornan todos hacia el mismo valor. Esto es debido a que, al ser la probabilidad de mutacion muy alta, los cromosomas se mantendran variando muy rapido a lo largo de las generaciones no tendiendo a estabilizarse nunca. Sin embargo, como en esta grafica aplicamos elitismo, se llego al cromosoma optimo gracias a la gran cantidad de variabilidad que presenta esta corrida.

Capítulo 4

Metodología de desarrollo

4.1. Procedimientos llevados a cabo

Como se dijo en la sección anterior, en este trabajo tendremos en cuenta que el comportamiento del algoritmo genético es altamente dependiente de los parámetros iniciales (tamaño de la población, porcentaje de cruce, porcentaje de mutación, número de generaciones, etc.), por lo que ajustaremos esos parámetros para tratar de mejorar la solución para los objetivos del problema propuesto.

A cada iteración de este proceso se le denomina una generación. Un algoritmo genético típicamente se itera de 50 a 500 o incluso más generaciones. El conjunto entero de generaciones se denomina una ejecución. Al final de una ejecución existen a menudo uno o varios cromosomas altamente adecuados en la población, y que pueden ser elegidos como solución al problema, a los cuales llamaremos "cromosomas óptimos".

Por lo tanto, en nuestro trabajo implementaremos la ejecución de un algoritmo genético simple para resolver el problema de optimización planteado en el enunciado. El diseño del cromosoma representará una solución factible que satisface la ecuación, y cuyo espacio de solución es $\{ x \mid x \epsilon Z \text{ y } 0 <= x <= 2^{30-1} \}$. Es por esto que la longitud de cada cromosoma será de 30 bits, o lo que es lo mismo, de 30 genes.

Implementamos diferentes funciones y procedimientos que están directamente relacionados con la teoría que sustenta estos algoritmos. A continuación describiremos dichas funciones y procedimientos en forma de pasos, que son los que fueron llevados a cabo para la correcta implementación del algoritmo:

1. **Inicio**: Una vez definidos los diferentes parámetros que se ingresan por consola, se crea la población inicial, la cual es generada de manera aleatoria, mediante una función. En esta instancia se generan los cromosomas en binario y en decimal.

2. **Calcula-f-obj()**: Luego procedemos a calcular el valor de la función objetivo para cada cromosoma de la generación. Para realizar esto se utiliza el cromosoma en decimal y se reemplaza en la función dada en el enunciado.

3. **Calcula-fitness()**: Se invoca a un método que calcula el fitness (aptitud) para cada cro-

mosoma. Este método lo que hace es ejecutar la siguiente función para cada cromosoma$_i$:

$$fitness_i = \frac{FO_i}{\sum_{j=0}^{n} FO_j} \tag{4.1}$$

4. **Selec-cross(elit)**: Aplicamos el operador de selección con base en el fitness de la población. Luego se procede a calcular la frecuencia acumulada de cada cromosoma, basado en el fitness de cada uno. Una vez hecho esto se "tira la ruleta" dos veces, una para cada padre, y se eligen los cromosomas que tengan la frecuencia acumulada proporcional a dicho numero.

5. Mediante la utilización del método anterior, consultamos la probabilidad de Crossover y, en caso afirmativo, aplicamos el operador genético de reproducción denominado "Crossover de un corte" a cada uno de los padres.

6. **mutación()**: Aplicamos el operador de mutación invertida. Previamente consultamos la probabilidad de mutación y en caso afirmativo lo aplicamos. Se aplican estos operadores a la población actual para generar a la población de la siguiente generación.

7. Ir al paso 2 hasta que la condición de parada se satisfaga.

Como se ha podido ver en el algoritmo, en cada generación se selecciona a un conjunto de los mejores individuos (paso 4) y se les modifica para generar la siguiente generación mediante los llamados operadores genéticos (paso 5 y 6). Estos operadores son tres: reproducción, cruce y mutación. El operador de cruce intenta simular la reproducción sexual, de tal manera que los individuos resultantes del cruce contendrán información de varios individuos. El operador de mutación simula la mutación biológica, de tal manera que los individuos mutados serán ligeramente diferentes de los individuos originales. Así en la siguiente generación, algunos individuos simplemente habrán sido copiados, y otros cruzados y/o mutados.

Capítulo 5

Herramientas utilizadas

5.1. Herramientas informáticas

5.1.1. Programación

Para el desarrollo del algoritmo genético utilizamos el lenguaje de programación Python gracias a su eficiente generador de números random y fácil sintaxis.
También incorporamos distintas librerías:

- numpy
- random
- matplotlib.pyplot
- math
- pandas
- os
- xlsxwriter

Destacaremos los más utilizados, NumPy y MatPlotLib.

NumPy

NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. El ancestro de NumPy, Numeric, fue creado originalmente por Jim Hugunin con algunas contribuciones de otros desarrolladores. En 2005, Travis Oliphant creó NumPy incorporando características de Numarray en NumPy con algunas modificaciones. NumPy es open source.

MatPlotLib

Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática NumPy. Proporciona una API, pylab, diseñada para recordar a la de MATLAB.

5.1.2. Comunicación y trabajo en equipo

Utilizamos 4 herramientas para comunicarnos, Whatsapp, Discord, Zoom y Visual Studio Code con la extensión "Live Share".
Discord es una aplicación muy similar a Zoom utilizada para realizar llamadas grupales y compartir pantalla.
Live Share de Visual Studio Code se utiliza para poder escribir código todos los integrantes en tiempo real y ver los cambios de los integrantes. Además, también tiene una función para realizar llamadas grupales creando salas de voz desde la misma aplicación.

La metodología de trabajo constaba en:

1) Luego de acordar por Whatsapp un horario, realizar una llamada en alguna de las distintas aplicaciones mencionadas anteriormente.
2) Leer cuál era el próximo punto en la consigna a resolver.
3) Analizar el punto desde un punto teórico de algoritmos genéticos.
4) Trasladar el algoritmo teórico charlado a un algoritmo programado en python(Generalmente buscando por internet como escribir el código).
5) Probar que el código funcione y en caso de que no, todo el grupo busca el causante y trata de solucionarlo.

Después de eso varios integrantes podían decidir continuar depurando el código individualmente, y luego comunicar los cambios al código mediante Whatsapp.

Capítulo 6

Software Desarrollado

```
import numpy as np
import random
import matplotlib.pyplot as plt
import math
import pandas as pd
import os
import xlsxwriter

######## GENERA CRmOMOSOMAS EN BINARIO ########
def genera_cromo_bin():
    cromosoma = []
    for i in range(30):
        #Genero cada gen (en binario) de forma aleatoria
        x = np.random.randint(0,2)
        x = str(x)
        cromosoma.append(x)
    cromosoma = "".join(cromosoma)
    return cromosoma

######## GENERACION DE CROMOSOMAS ########
def big_bang():
    for i in range(p):
        #Genero cromosomas en binario y en decimal
        cromosomas_bin.append(genera_cromo_bin())
        cromosomas.append(int(cromosomas_bin[i],2))

######## FUNCION OBJETIVO ########
def calcula_f_obj():
    for i in range(p):
        f_obj[i] = ((cromosomas[i]/((2**30)-1))**2)
```

```
######## FITNESS ########
def calcula_fitness():
    for i in range(p):
        fitness[i] = (f_obj[i]/sum(f_obj))

######## RULETA ########
def calcula_ruleta():
    #Calculo la frecuencia acumulada de cada cromosoma
    frec_acum = []
    frec_acum.append(fitness[0])
    for i in range(1,p):
        acumulado = frec_acum[i - 1] + fitness[i]
        frec_acum.append(acumulado)
    return frec_acum


######## TIRADA DE RULETA ########
def tiradas(ruleta):
    padres = []
    for m in range(2): #Ciclo de 2 porque necesito dos cromosomas
        frec = random.uniform(0,1)
        #Para hacer Crossover utilizamos la frecuencia acumulada,-
        # - basada en los fitness de los cromosomas
        for i in range(p):
            if(ruleta[i] > frec):
                padres.append(cromosomas_bin[i])
                break

    return padres[0], padres[1]

######## CROSSOVER ########
def crossover(c1, c2):
    c = np.random.randint(0,101)
    if c <= cr:
        #Ejecucion del Crossover, se corta en un punto random y -
        # - se cortan en ese punto los cromosomas de los padres
        num_corte = np.random.randint(0, 29)
        aux = c1[num_corte:]
        c1 = c1[:num_corte] + c2[num_corte:]
        c2 = c2[:num_corte] + aux
    return c1, c2

######## SELECCION Y CROSSOVER ########
def selec_cross(elit):
    ruleta = calcula_ruleta()

    aux = []
```

```
    if elit == 2:
        #cromosomas_bin.sort(reverse = 1)
        for i in cromosomas_bin:
            aux.append(i)
        c_max1 = max(aux)
        indice1 = aux.index(max(aux))
        aux.remove(max(aux))
        c_max2 = max(aux)
        indice2 = aux.index(max(aux))

    for i in range(0, len(cromosomas), 2):
        c1,c2 = tiradas(ruleta)
        c1,c2 = crossover(c1,c2)
        cromosomas_bin[i] = c1
        cromosomas_bin[i+1] = c2

    cromosomas_bin[indice1] = c_max1
    cromosomas_bin[indice2] = c_max2

######## MUTACION ########
def mutacion():
    for i in range(len(cromosomas_bin)):
        x = np.random.randint(0, 101)
        if x <= m:
            corte = np.random.randint(0, 30)

            #Defino el punto de corte segun el numero random
            cromo = list(cromosomas_bin[i])
            cromo[corte] = str(1 - int(cromo[corte]))
            cromosomas_bin[i] = "".join(cromo)

######## BINARIO A DECIMAL ########
def cromosomas_to_decimal():
    for i in range(len(cromosomas_bin)):
        cromosomas[i] = int(cromosomas_bin[i], 2)

######## ELITISMO ########
def esElitismo():
    rta = int(input("¿Desea implementar elitismo? (Si: 1, No:0): "))
    if rta == 1:
        return 2
    else:
        return 0

######## Programa Principal ########

#Ingreso de parametros
```

```
p = int(input("Ingrese tamaño de poblacion:"))
g = int(input("Ingrese cantidad de generaciones:"))
m = int(input("Ingrese tasa de mutacion:"))
cr = int(input("Ingrese tasa de crossover:"))
elit = esElitismo()

cromosomas = []
cromosomas_bin = []
f_obj = list(np.zeros(p))
fitness = list(np.zeros(p))

#Declaracion de listas
lista_min = []
lista_max = []
lista_prom = []
lista_cromosomas_dec_max = []
lista_cromosomas_bin_max = []

#Se crea la poblacion inicial
big_bang()

#Se calcula la funcion objetivo y el fitness de dicha poblacion
calcula_f_obj()
calcula_fitness()

for i in range(g):
    #Se llenan las listas con los diferentes resultados de la ejecucion

    lista_min.append(min(f_obj)) #Todas las funciones objetivo minimas
    lista_max.append(max(f_obj)) #Todas las funciones objetivo maximas
    lista_prom.append(np.mean(f_obj)) #Todos los promedios de las funciones objetivo
    lista_cromosomas_dec_max.append(max(cromosomas))
    lista_cromosomas_bin_max.append(max(cromosomas_bin))

    #Se hace la evaluacion, seleccion y crossover
    selec_cross(elit)
    mutacion()
    cromosomas_to_decimal()

    #Se vuelven a inicializar las listas para la nueva generacion
    f_obj = list(np.zeros(p))
    fitness = list(np.zeros(p))

    #Calculo de la funcion objetivo y el fitness de los cromosomas hijos
    calcula_f_obj()
    calcula_fitness()

######## Salida del sistema ########
```

```
#Lista con la cantidad de generaciones
generacion = np.arange(1, g + 1)

## TABLA DE EXCEL
Datos = pd.DataFrame({"Generacion": generacion,
                      "Minimo FO": lista_min, "Maximo FO": lista_max,
                      "Valor cromosoma dec (Maximo)": lista_cromosomas_dec_max,
                      "Valor cromosoma bin (Maximo)": lista_cromosomas_bin_max,
                      "Promedio FO": lista_prom})
Tabla = pd.ExcelWriter('C:/Users/Gofiar/Desktop/tabla.xlsx', engine='xlsxwriter')
Datos.to_excel(Tabla, sheet_name='Valores', index = False)

## DISEÑO TABLA
workbook = Tabla.book
worksheet = Tabla.sheets["Valores"]

formato = workbook.add_format({"align": "center"})

worksheet.set_column("A:C", 15, formato)
worksheet.set_column("D:D", 32, formato)
worksheet.set_column("E:E", 35, formato)
worksheet.set_column("F:F", 15, formato)
worksheet.conditional_format("F1:F"+str(len(lista_prom)+1), {"type": "3_color_scale"})

Tabla.save()

## GRÁFICAS
plt.subplots()
plt.title("Cromosoma Óptimo: " + str(max(lista_cromosomas_dec_max)))
plt.axhline(y = max(lista_max), color = 'r', label = "FObj del Cromosoma Optimo")
plt.plot(generacion, lista_min, color = 'k', label = "Min")
plt.plot(generacion, lista_max, color = 'b', label = "Max")
plt.plot(generacion, lista_prom, color = 'g', label = "Prom")
plt.grid(True)
plt.xlabel("Cantidad de ciclos")
plt.ylabel("Funcion objetivo (FO)")
plt.legend(loc = "lower right")
plt.tight_layout()
plt.show()

#Se borra la tabla de EXCEL
input()
os.remove('C:/Users/Gofiar/Desktop/tabla.xlsx')
print("Tabla borrada. Fin de programa")
```

Capítulo 7

Conclusión

7.1. Conclusión

Como hemos podido ver a lo largo del documento los algoritmos genéticos son actualmente una fuerte fuente de resolución de problemas pudiendo así obtener diferentes soluciones al problema. Sin embargo, como se ha podido observar, no hay ninguna estrategia que sea siempre invencible, sino que hay un conjunto de estrategias que suelen dar buenos resultados. Así pues, habrá que ajustar los parámetros de acción en función de cada problema a modelar para obtener una solución que se adapte mejor a unas determinadas condiciones. Sin embargo, en una situación real puede suceder que no se conozcan los parámetros iniciales o que no se sepa la duración del algoritmo. Por lo tanto habrá que elegir con completo cuidado los parámetros iniciales. Un crossover elevado es recomendable para la mayoría de los problemas aunque hay situaciones en las que un crossover menor dará mejores resultados. La mutación deberá ser baja, entorno al 5 %. Para una óptima resolución, el resto de parámetros habrá que determinarlos en función del problema. La población a elegir debe atender a un valor óptimo (no por tener una población mayor, se va a tener una solución mejor), la codificación dependerá del problema a resolver como se ha visto con anterioridad y por último el método de selección más utilizado es el de selección por rueda de ruleta, aunque como se ha comentado, dependerá de la cuestión que nos trate.

Bibliografía

[1] LaTeX2 Project Team *LaTeX2εforauthors*,2019.
https://www.latex-project.org/help/documentation/usrguide.pdf

[2] Ruben Ruiz *Cómo usar Overleaf editor de LaTeXonline -Tutorial-*.
https://www.youtube.com/watch?v=Uyjo6R5z2js

[3] *Algoritmo genético*.
https://es.wikipedia.org/wiki/Algoritmo_genético

[4] *NumPy*.
https://es.wikipedia.org/wiki/NumPy

[5] *NumPy*.
https://es.wikipedia.org/wiki/NumPy

[6] *Algoritmo genético*.
https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-