Memoria

Grupo 10

Integrantes

- Xiao Peng Ye
- Zheng Yu Ye
- Juan Diego Valencia Marin

Índice

Introducción

Los algoritmos genéticos son métodos de optimización heurística que se emplean para hallar un valor o valores que consiguen maximizar o minimizar una función, esto depende del fitness a utilizar, ya que puede ser de maximización o de minimización. El funcionamiento de este algoritmo está inspirado en la teoría evolutiva de la selección natural.

Los algoritmos genéticos surgieron en los años 1970 gracias a John Henry Holland, (también es conocido como el padre del Algoritmo genético), siendo una de las ideas más prometedoras en la inteligencia artificial.

John Holland estaba consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla en un programa de computadora. Su objetivo era lograr que las computadoras aprendieran por sí mismas. Aunque la técnica que inventó Holland se le llamó originalmente "planes reproductivos", se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro (Adaptation in Natural and Artificial Systems) en 1975.

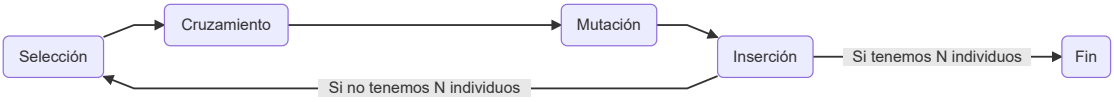
Algoritmo implementado

En nuestro caso, hemos implementado el **algoritmo genético (GA)** con una configuración específica, el *intercambio genético uniforme*, *mutación uniforme*, *selección competitiva (tournament)* y *reemplazo en estado estacionario (Steady-state)*, la cual se comentará más tarde.

El funcionamiento de este algoritmo es el siguiente:

1. Crear una población inicial aleatoria con **N** individuos.
2. Calcular el fitness de cada individuo de la población.
3. Crear una nueva población aplicando los siguientes pasos:
    1. Seleccionar dos individuos de la población ya creada en función del método de selección realizado.

2. Cruzar los dos individuos seleccionados para generar un nuevo individuo en función del intercambio genético realizado.
3. Realizar el método de mutación sobre el individuo.
4. Añadirlo en la nueva población.
5. Repetir todos estos pasos hasta obtener el mismo número de individuos que la antigua población.



4. Reemplazar la nueva población por la antigua.

Funcionamiento específico

Intercambio genético uniforme (Uniform crossover)

En esta etapa se consigue generar nuevos individuos a partir de los anteriores, sustituyendo los anteriores por los nuevos.

El intercambio genético uniforme se realiza intercambiando un único individuo, con una determinada probabilidad, por uno de los individuos seleccionados previamente.

Mutación uniforme (Uniform mutation)

Este paso es importante para obtener diversidad y evitar que el algoritmo caiga en mínimos locales debido a que todos los individuos sean demasiado semejantes.
Una vez obtenido el nuevo individuo, este se somete a un proceso de mutación el cual puede verse modificado con una determina probabilidad.

La mutación uniforme se consigue sumando a la posición i un valor extraído de una distribución uniforme.

Selección competitiva (Tournament selection)

En general la selección de los individuos se realiza favoreciendo a aquellos con mejor fitness, nuestro caso no es la excepción, debido a que se selecciona el mejor individuo con mejor fitness de una forma competitiva.

Este tipo de selección se llama competitiva porque se comprueba el fitness de dos individuos aleatorios un número determinado de veces hasta conseguir el individuo con mejor fitness.

Reemplazo (Steady-state replacement)

Esta etapa es la encargada de cambiar la población de individuos y creando así, una nueva generación de individuos. Además, evita la convergencia prematura y promociona a los individuos más aptos.

El reemplazo estacionario sustituye el peor individuo de la población por el nuevo descendiente.

Fitness utilizado

El fitness (aptitud o capacidad en español) es la forma de evaluar cada individuo, ver cuan bueno es como solución del problema.

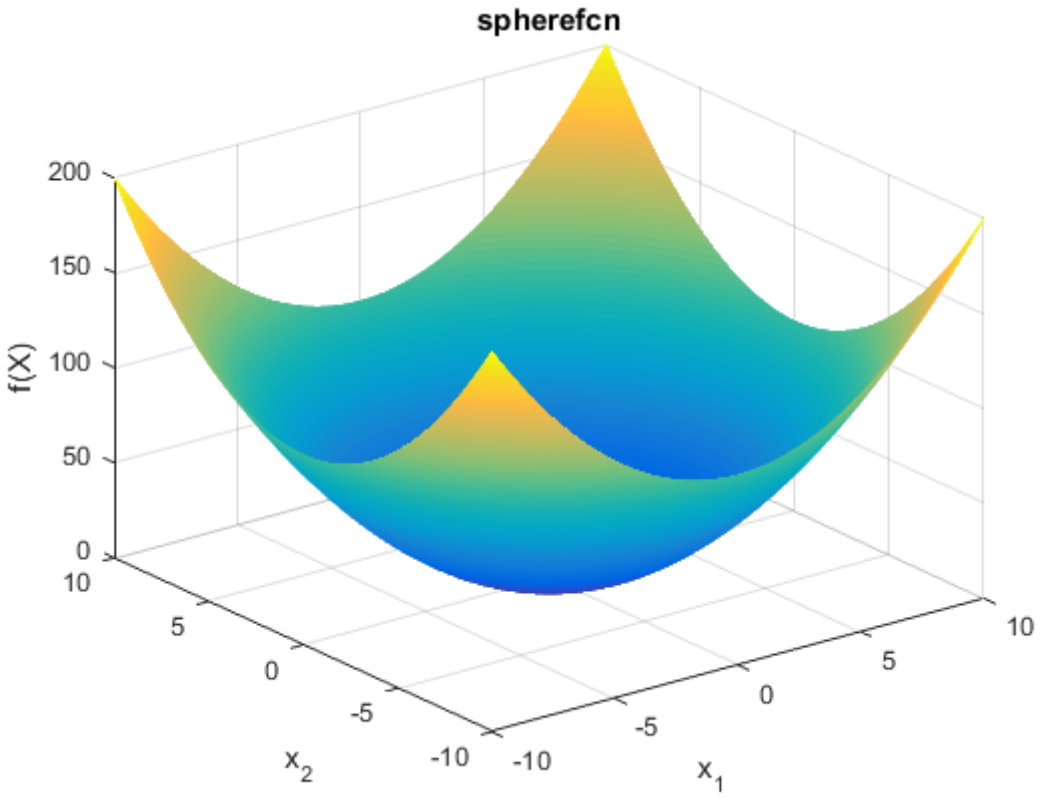
En nuestro caso hemos realizado la función de fitness para adaptar problemas de minimización de una función, que significa que cuanto menor valor de la función objetivo tiente el individuo, mayor fitness se obtiene.

Una forma de calcularlo es realizando la función fitness del individuo y después invertirla - $f(\mathbf{individuo})$ o $\frac{1}{1-f(\mathbf{individuo})}$.

Pruebas realizadas

Todas las soluciones siguientes son uno de los ejemplos diversos al ejecutar el programa con la función correspondiente alrededor de 20000 iteraciones sobre una población de 50 genomas con longitud 10 variables.

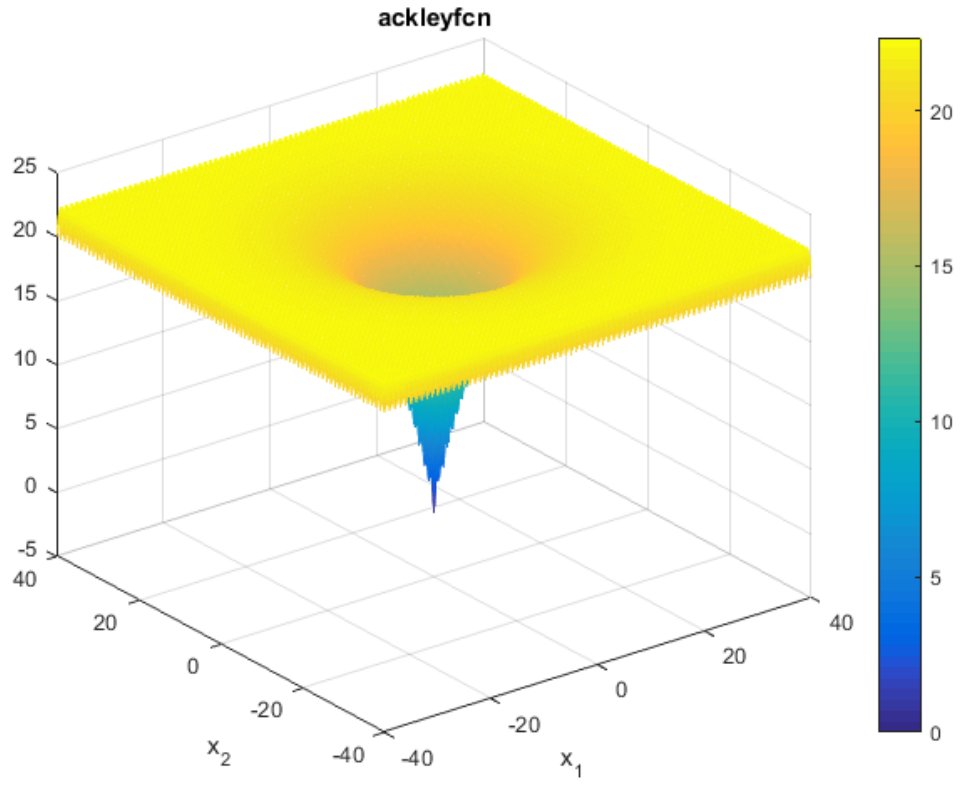
- **Function Sphere**
  - $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i^2$.
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, \ldots, 0)$



  - Bounds: $x_i \in [-5.12, 5.12]$ for $i = 1..n$
  - Una posible salida con nuestro programa:

```
[-0.00818072  0.11111143  0.05332875 -0.0282895  0.01149842 -0.02333086
-0.01925091 -0.00604656 -0.00959379  0.04176625]
Fitness: -0.018977087586905483
```

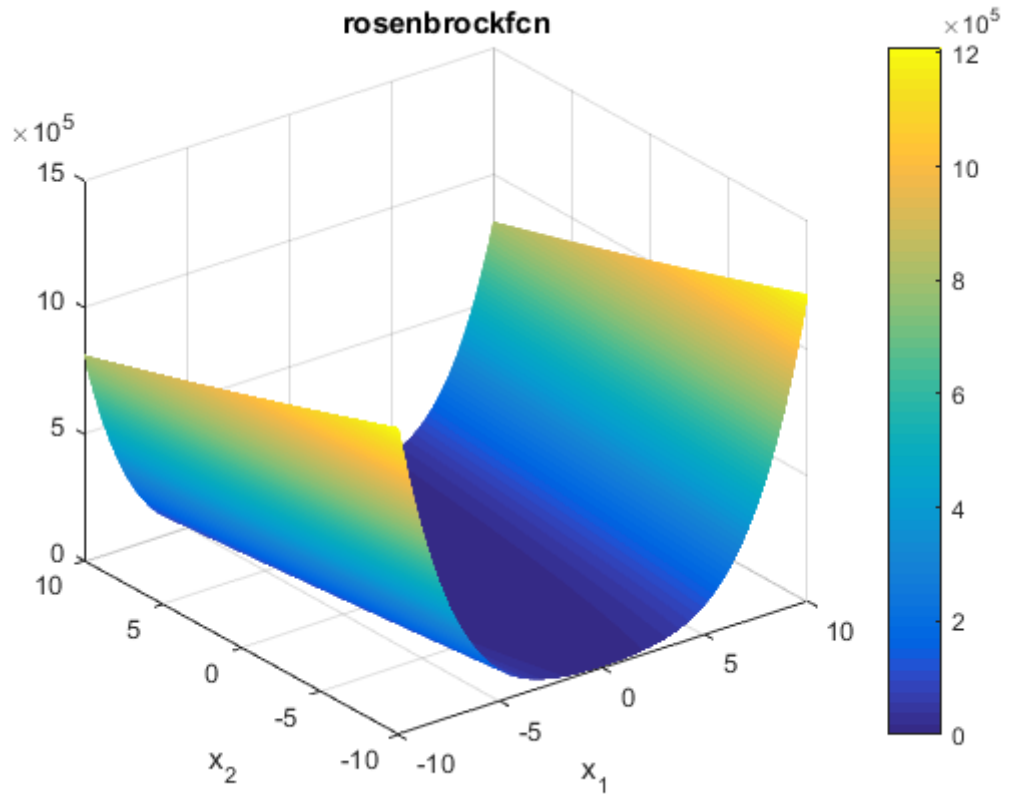
- **Function Ackley**
  - $f(\mathbf{x}) = f(x_1, \ldots, x_n) = -a.\, exp(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)) + a + exp(1)$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, \ldots, 0)$



- Bounds: $x_i \in [-32, 32]$ for $i = 1..n$
- Una posible salida con nuestro programa:

```
[-0.01337104  0.00378951  0.00671465  0.00461864 -0.01431343  0.01159774
-0.00041528  0.00478414  0.00542893  0.00308027]
Fitness: -0.03603606025219763
```

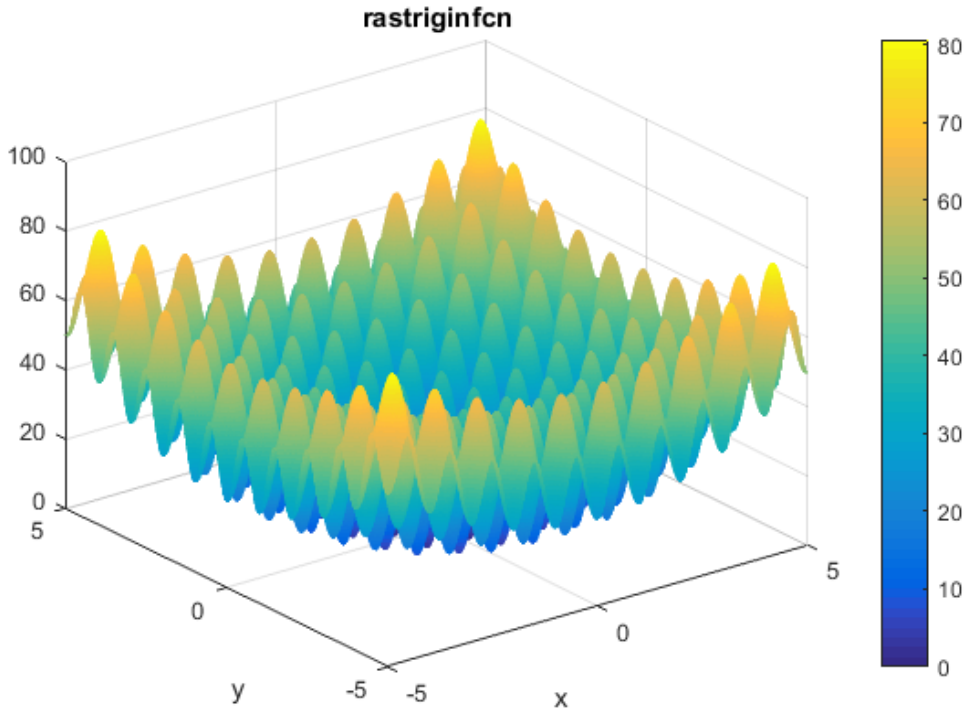
- **Function Rosenbrock**
  - $f(x, y) = \sum_{i=1}^{n} [b(x_{i+1} - x_i^2)^2 + (a - x_i)^2]$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (1, \ldots, 1)$



- Bounds: $x_i \in [-5, 10]$ for $i = 1..n$
- Una posible salida con nuestro programa:

```
[-2.01295544e+00  -2.00318632e+00  1.44479825e-03  -2.02127134e+00
-2.04204708e+00  8.07145638e-02  1.63757233e-01  -2.35384226e+00
8.39914142e-01  2.38744252e+00]
Fitness: -0.9649353905764941
```

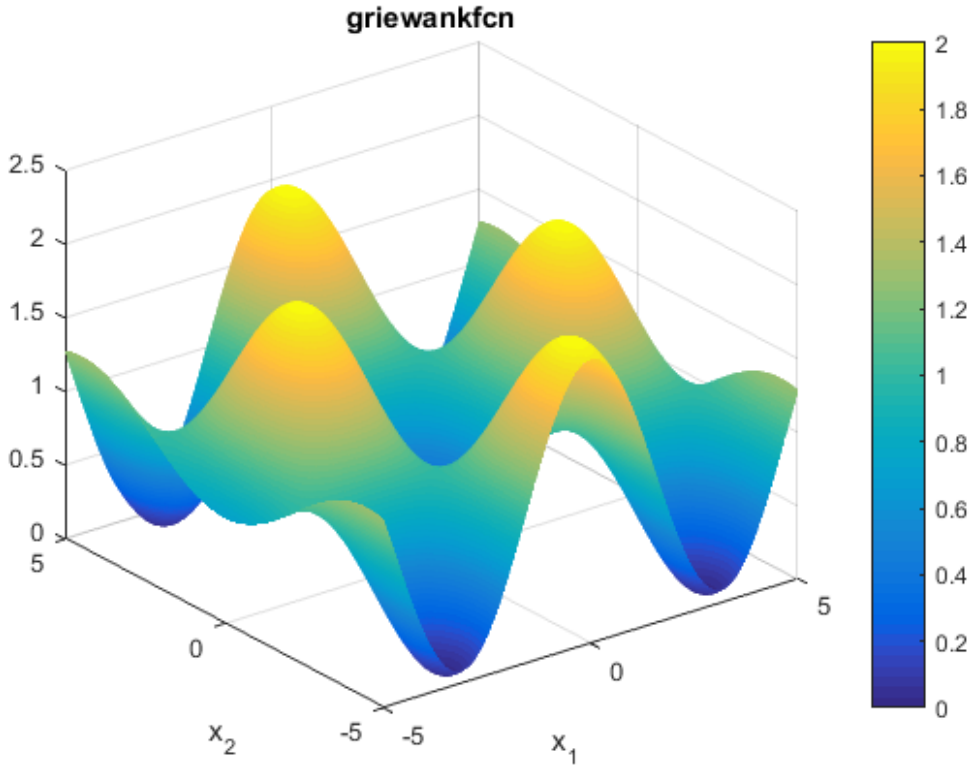
- **Function Rastrigin**
  - $f(x, y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde$\mathbf{x}^* = (0, 0)$



- Bounds: $x_i \in [-5.12, 5.12]$ for $i = 1..n$
- Una posible salida con nuestro programa:

```
 [ 0.01357292  0.00305028  0.00353871 -0.00032503 -0.00053246  0.00213341
-0.00402281  0.00668157  0.0123271   0.00122662]
 Fitness: -0.0843335201155675
```

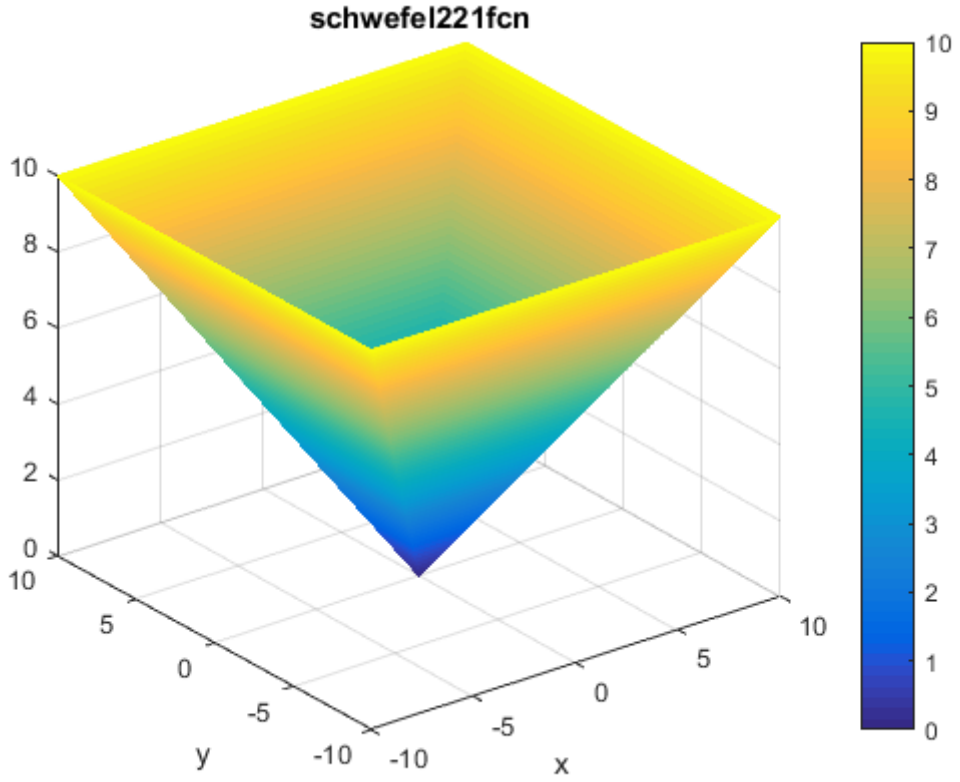
- **Function Griewank**
  - $f(x, y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, 0)$



- Bounds: $x_i \in [-600, 600]$ for $i = 1..n$
- Una posible salida con nuestro programa:

```
[-1.63665742 -0.63387212  1.06524781 -2.93834095 -0.81735668 -0.84701633
-0.60488459 -3.34330734 -2.21048473  0.91448804]
 Fitness: -1.007875167722615
```

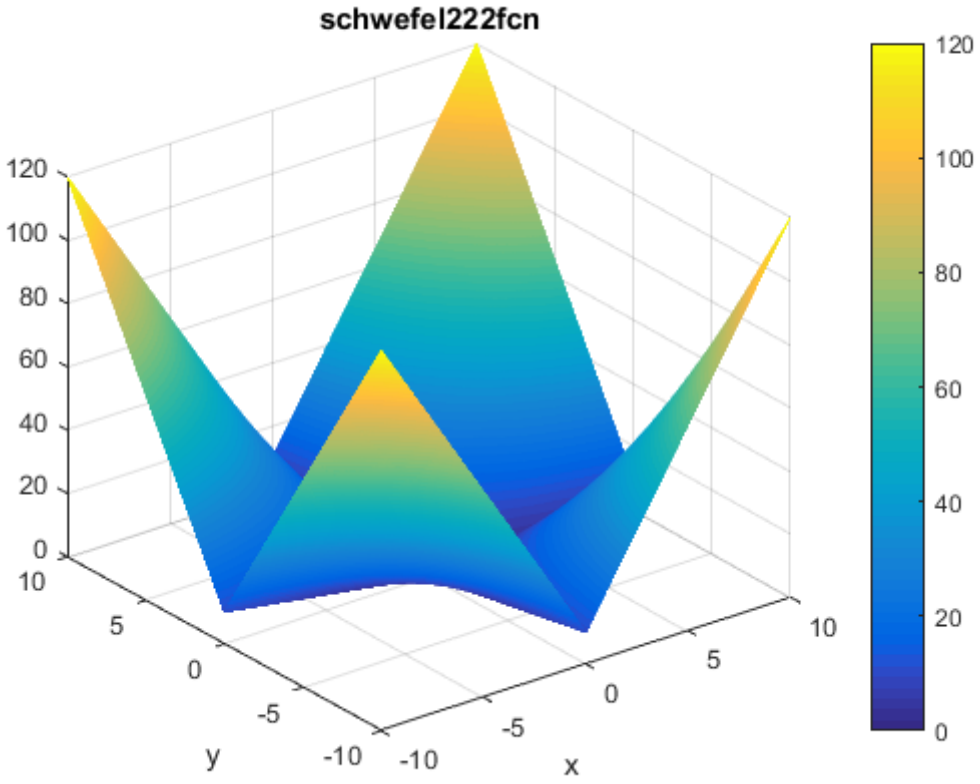
- **Function Schwefel 2.21**
  - $f(\mathbf{x}) = f(x_1, \ldots, x_n) = \max_{i=1,\ldots,n} |x_i|$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, 0)$



  - Bounds: $x_i \in [-100, 100]$ for $i = 1..n$
  - Una posible salida con nuestro programa:

```
[-0.04560432  0.24343054 -0.33134579 -0.20688494  0.1855236  -0.24691806
-0.30072995  0.0131173  -0.47835121  0.26264957]
 Fitness: -0.4783512141489581
```

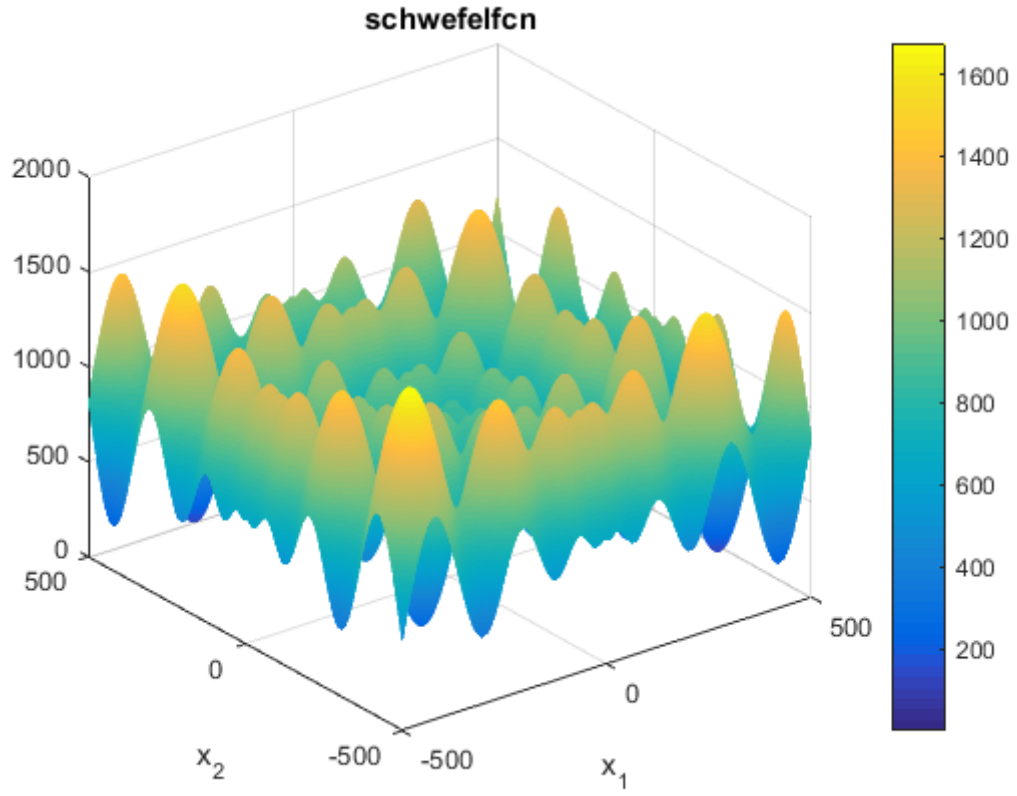
- **Function Schwefel 2.22**
  - $f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, 0)$



- Bounds: $x_i \in [-100, 100]$ for $i = 1..n$
- Una posible salida con nuestro programa:

```
[ 0.5630586  -0.8778864  -2.36791013 -1.91646273  4.32366397  1.243555
-0.13521534  0.73974573 -0.36022494 -0.28107385]
 Fitness: -1.007845579473045
```

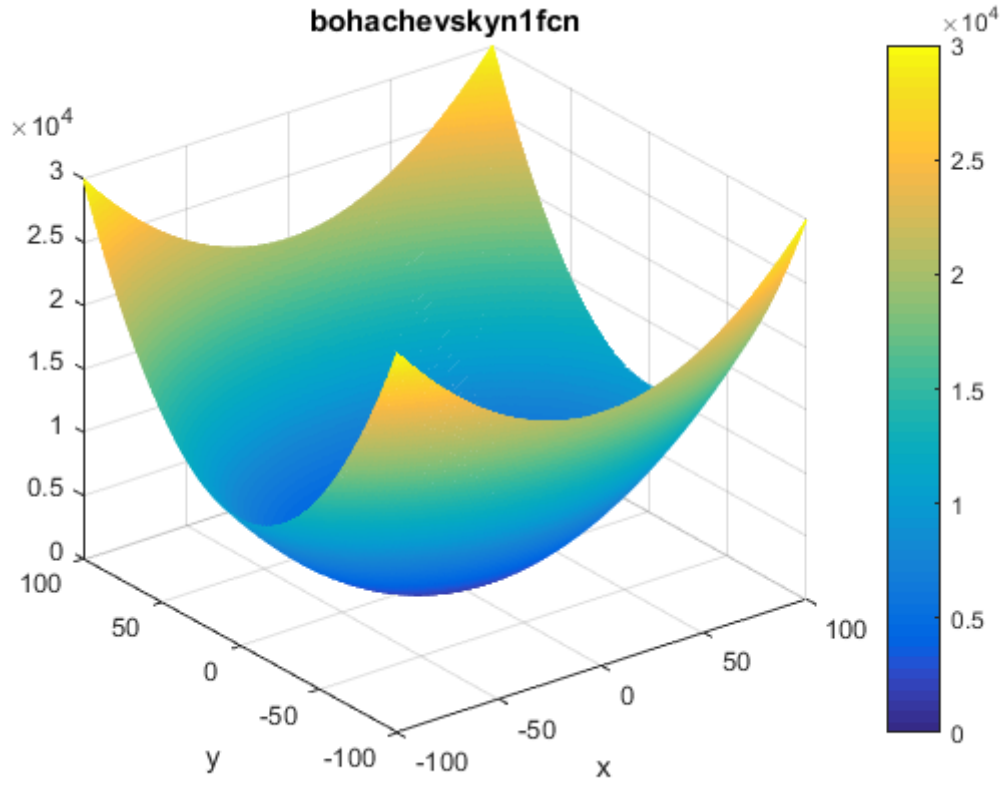
- **Function Schwefel 1.2**
  - $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n) = 418.9829d - \sum_{i=1}^{n} x_i sin(\sqrt{|x_i|})$.
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (420.9687, \ldots, 420.9687)$



- Bounds: $x_i \in [-500, 500]$ for $i = 1..n$
- Una posible salida con nuestro programa:

```
[ 0.62319837 -0.73560384  0.28831212 -0.41259132 -0.00641023  0.70275989
-0.75575393  0.3279226  -0.19957433  0.198604  ]
 Fitness: -0.8761325390587813
```

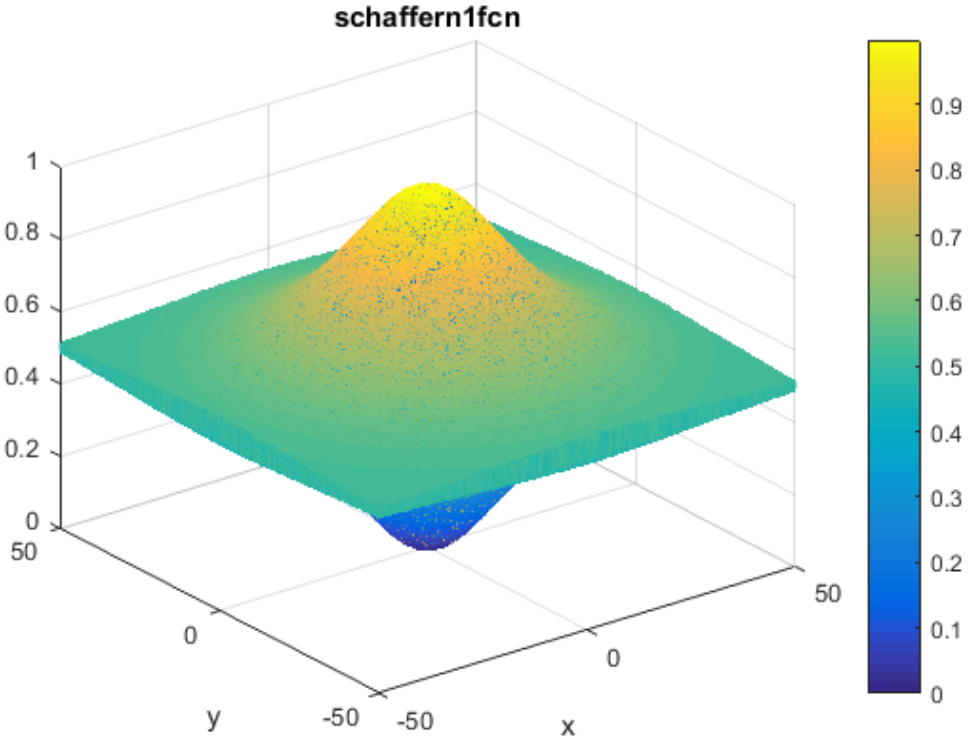
- **Function bohachevsky 1.2**
  - $f(x, y) = x^2 + 2y^2 - 0.3cos(3\pi x) - 0.4cos(4\pi y) + 0.7$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, 0)$



- Bounds: $x_i \in [-100, 100]$ for $i = 1, 2$
- Una posible salida con nuestro programa:

```
 [-0.02571173 -0.00842974 -0.01202293  0.02523098  0.04017672 -0.06709364
0.00889741 -0.02672166 -0.08121992 -0.03592722]
 Fitness: -0.70623177179338
```

- **Function Schaffer**
  - $f(x, y) = 0.5 + \frac{sin^2(x^2+y^2)^2 - 0.5}{(1+0.001(x^2+y^2))^2}$
  - El mínimo global se encuentra en: $f(\mathbf{x}^*) = 0$ donde $\mathbf{x}^* = (0, 0)$



- Bounds: $x_i \in [-100, 100]$ for $i = 1, 2$
- Una posible salida con nuestro programa:

```
 [-0.00071607 -0.00710123 -0.00116371 -0.01384613 -0.00973399  0.01484788
0.00604145 -0.00321972  0.00067127 -0.00734846]
 Fitness: -0.9990715308450793
```

Uso de la librería

Posicione en la carpeta donde se encuentra el paquete y ejecute la siguiente línea para instalar:

```
pip install group10-1.0.0-py3-none-any.whl
```

El siguiente programa es un ejemplo para usar la librería

```
from group10.EA import EA

# función que se desea sacar el mínimo
def f(sol):
    return sum(sol)

# máximos y mínimos de las soluciones
mybounds=[(0,1)]*10

# 50 es número de iteraciones
myEA=EA(f, mybounds, 50)
# itera 1000 generaciones
myEA.run(1000)
# obtener el mejor genoma de la población actual
bestGenome=myEA.best()
print(bestGenome.solution, bestGenome.fitness)
```

Conclusiones

Este trabajo se ha enfocado en la solución de funciones de minimización, con unos métodos específicos y algunos óptimos que otros. Según las soluciones de las pruebas que hemos conseguido durante la realización del trabajo, podemos concluir que cuando más muestras o ejecuciones realicemos, más nos acercamos a la solución óptima mínima. Pero lo que si ha quedado claro es su funcionamiento y su gran utilidad en la optimización de funciones.

Pasando a un enfoque más personal sobre la práctica, consideramos que ha sido interesante y beneficiosa para nosotros, pues no conocíamos este tipo de algoritmo y mucho menos las aplicaciones y las ventajas que ofrece.

Esto ocurre también con la herramienta Anaconda, pues desconocíamos su gran cantidad de funcionalidades y las facilidades que entrega al programador en Python. Y siguiendo por este camino, además hemos aprendido un nuevo lenguaje de programación, Python. Otra ventaja para nosotros, pues hemos salido de nuestra zona de confort con Java y hemos visto las diferencias y ventajas que ofrece Python ante otros lenguajes. Lo que nos ayudará y seguramente nos resultará útil en nuestro futuro.

Bibliografía

Holland, John H. "Adaptation in Natural and Artificial Systems".

https://es.wikipedia.org/wiki/Selecci%C3%B3n_natural#:~:text=La%20selecci%C3%B3n%20natural%20fue%20propuesta,los%20individuos%20de%20una%20poblaci%C3%B3n.

https://en.wikipedia.org/wiki/John_Henry_Holland

https://www.tecnologias-informacion.com/algoritmosgeneticos.html#:~:text=En%20la%20actualidad%2C%20el%20algoritmo,humanas%20y%20no%20tan%20automatizables.

https://www.cienciadedatos.net/documentos/py01_optimizacion_ga