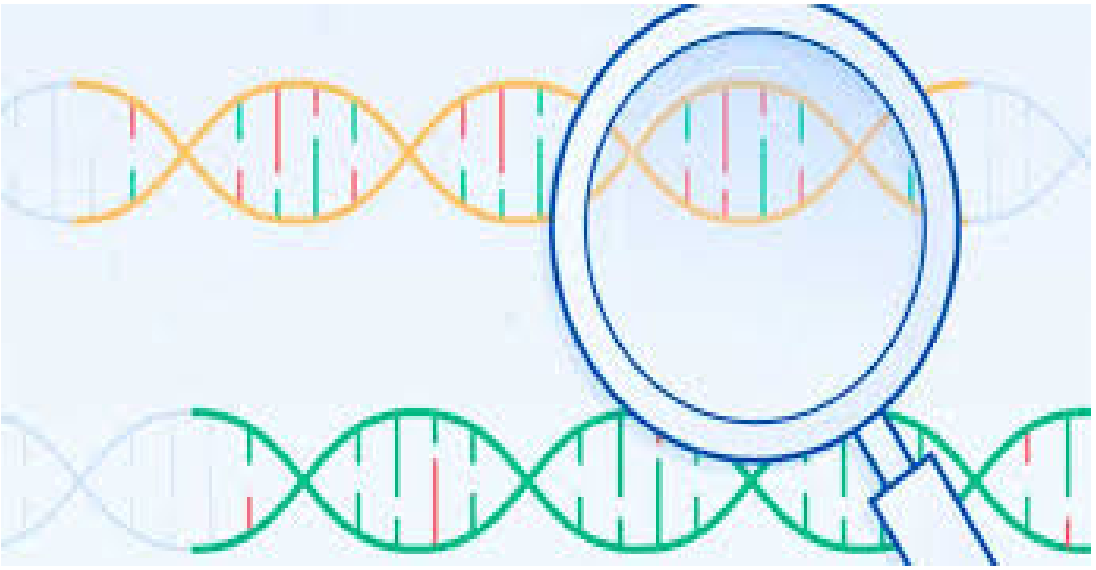
ALGORITMO GENÉTICO SIMPLE



Inteligencia Artificial

20 de mayo de 2021.

Tabla Informativa.

Periodo	Mayo - Agosto 2021
Cuatrimestre	14° “A”
Asignatura	Inteligencia Artificial
Corte	1
Actividad	171048 ROQUE JIMENO_C3.A3 PROYECTO
Fecha de asignación	2021.07.07
Fecha de entrega	2021.07.23

Matricula	Nombre
171048	Alexis Martin Roque Jimeno

Contenido

Listado de figuras.

Listado de tablas.

Introducción.

Un algoritmo genético es un método de búsqueda que imita la teoría de la evolución biológica de Darwin para la resolución de problemas. Se parte de una población inicial de la cual se seleccionan los individuos más capacitados para luego reproducirlos y mutarlos, para finalmente obtener la siguiente generación de individuos que estarán más adaptados que la anterior generación [1].

Los pasos básicos de un algoritmo genético son:

- Evaluar la puntuación de cada uno de los cromosomas generados.
- Permitir la reproducción de los cromosomas siendo los más aptos los que tengan más probabilidad de reproducirse.
- Con cierta probabilidad de mutación, mutar un gen del nuevo individuo generado.
- Organizar la nueva población.

Estos pasos se repetirán hasta que se de una condición de terminación.

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

Parámetros de diseño.

Parámetro	Valor
Error permisible (ε)	0.06
Población inicial	50
Población Máxima	70
Generaciones	100
Rango en abscisa y ordenada en X	[10 - 80]
Rango en abscisa y ordenada en Y	[25 - 100]
Probabilidad de mutación	0.4
Individuos seleccionados	5

Tabla 1. Parámetros de diseño

Tabla de fórmulas.

Nombre	Fórmula	Referencia	Página.
Error permisible	$\Delta x \leq 2\,\varepsilon$	*Fórmula 1. Error permisible*	6
Coordenadas.	$x(i) = min + i * \Delta x$	*Fórmula 2. coordenadas.*	8
Función Fitness	$z(x, y) = x\cos(y) + y\sin$	*Fórmula 3. Fórmula de fitness.*	8

Tabla 2. Fórmulas.

Implementación.

1. **Cálculo del error permisible.**

El cálculo del error permisible servirá para determinar en cuántos bits se representan los individuos de la población. Para poder calcular dicho error, se utiliza la fórmula:

$$\Delta x \leq 2\,\varepsilon$$

Fórmula 1. Error permisible.

Donde ε ha sido previamente asignado como un parámetro de diseño. Lo siguiente es calcular el rango que existe entre abscisa y la ordenada. Una vez teniendo el valor del rango y de Δx , se debe conocer cuantas Δx se necesitan para poder alcanzar el rango. Este total, será la representación de los bits para cada individuo.

Este mismo cálculo se realizará para obtener el valor de la Δy .

2. **Generación de la población.**

La población se creará a partir del tamaño de la población inicial y podrá llegar a crecer de un tamaño dado por la población máxima, ambos definidos como parámetros de diseño.

La población consta de individuos generados aleatoriamente.

3. **Selección de los individuos.**

Los individuos serán evaluados para determinar quienes son los más óptimos para pasar a la siguiente etapa. La forma de evaluación es por medio del método de competición. Se seleccionan únicamente los 5 individuos con mejores resultados. Dado a que el problema consiste en la minimización, se tomarán los resultados que hayan dado menores.

4. **Conversión de los individuos a binario.**

Para poder pasar a la siguiente fase, se necesita conocer el valor de los individuos en formato binario. Para ello, ya se conoce el valor de los bits en que serán representados los individuos.
Para poder convertirse de un decimal a binario, se utilizó el método de python:

format (parametro1 , parametro2)

Dónde el primer parámetro es el individuo que se convertirá, y el segundo parámetro es al total de bits en que va a representarse.

5. **Reproducción o cruza de los individuos.**

Una vez teniendo a todos los individuos de la población representados de manera binaria, se lleva a cabo la cruza, donde se utiliza el método de un punto.

Se crea un ciclo del tamaño de los individuos seleccionados para la reproducción, seguido se toma aleatoriamente quién de estos individuos se reproducirá. Después se crea una posición aleatoria la cual servirá para obtener un individuo de la población. Una vez teniendo a los dos individuos que van a reproducirse, se crea el punto aleatorio de cruce en un rango de 1 hasta el tamaño total de los bits. De esta manera se crea el punto de cruce entre los individuos.

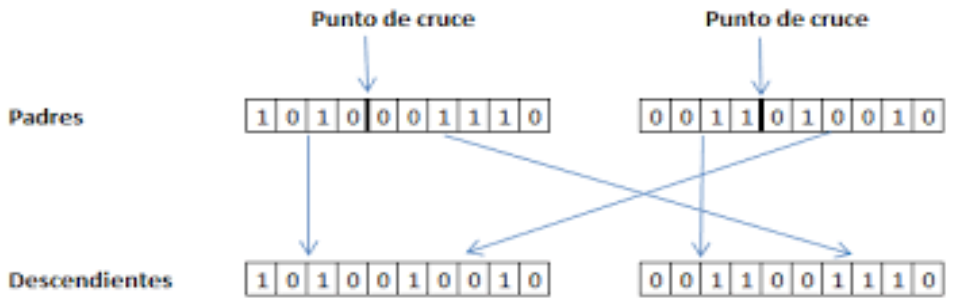


Imagen 1. Punto de cruce

De esta manera estaríamos obteniendo dos nuevos descendientes, los cuales son agregados a la población actual.

6. **Mutación.**

Tras el cruce o reproducción, toma lugar la mutación, para esto se crea aleatoriamente un número entre 1 y el tamaño total de la población actual, este número representa el total de individuos que mutaron.

Una vez teniendo este total, se vuelve a generar un número aleatorio entre 0 y 1, y si el resultado de este número es menor a la probabilidad de mutación, entonces el individuo tendrá lugar a su mutación. Se vuelve a generar una posición aleatoria para tomar un individuo de la población al azar, el cual será sometido a la mutación

Ya que se tiene al individuo seleccionado, se evalúa cada uno de sus bits, donde todos aquellos que tengan un valor de ‘1’ , serán cambiados a un valor de ‘0’. Y aquellos que tienen un valor de ‘0’, serán cambiados por un valor de ‘1’.

1	**0**	**0**	**1**

0	**1**	**1**	**0**

Imagen 2. Mutación

7. Conversión de binario a decimal.

Con la población actual, ya reproducida y mutada. El siguiente paso es evaluar cada uno de sus fitness, pero antes de eso, hay que convertir los números a decimal ya que se encuentran en binarios. Para la conversión únicamente se realiza una conversión de la siguiente manera: int (str(parametro1) , base) .

Donde el primer parámetro es el individuo a convertir, y el segundo es la base en que se convertirá, en este caso es la base 2.

8. Evaluación del fitness

El fitness es calculado independientemente por cada individuo en cada generación. Para ello, hay que representar a los individuos en coordenada (x , y). Para esto, se utiliza la siguiente fórmula:

$$x\,(i) \;=\; min \;+\; i \;*\; \Delta x$$

Fórmula 2. coordenadas.

Donde el ‘min’ corresponde al primer rango , la ‘i’ corresponde al individuo actual , y Δx al resultado obtenido del cálculo del error permisible. De esta manera convertimos los individuos en coordenadas para (x). Es necesario hacer este mismo cálculo para obtener la coordenada en (y).

Una vez obtenidas las coordenadas (x , y) , estas son sustituidas en nuestra fórmula para calcular el valor del fitness.

$$z(x\,,\,y) \;=\; x\cos(y) \;+\; y\sin(x)$$

Fórmula 3. Fórmula de fitness.

Cada uno de los resultados son guardados en una lista, después se obtiene el mejor fitness en cada una de las generaciones.

9. **Poda.**

Una vez que se obtuvo el valor de los fitness de cada individuo, la población actual se divide en dos, únicamente conservando la mitad que tuvieron mejores resultados y eliminando a la otra mitad. Esto únicamente se lleva a cabo cuando el tamaño de la población ha superado el tamaño máximo de la población establecida.

Esta estrategia de poda consiste en eliminar a la mitad de la población que no está apta para continuar, procurando así mantener durante las generaciones únicamente aquellos individuos que son óptimos.

10. **Graficación de resultados.**

Para la graficación, se muestran tanto los mejores resultados obtenidos durante cada generación , así como los menos óptimos. Mostrando de esta manera como fue el comportamiento de los individuos para encontrar la solución más óptima, y los menos óptimos, para mostrar la veracidad del AG.

Resultados.

1. Interfaz gráfica.

El programa está realizado en Python, con una interfaz gráfica realizada con la librería de PyQT5 y la interfaz de QtDesigner. En esta interfaz, se solicitan los datos para los parámetros de diseño (Véase la tabla 1. parámetros de diseño).

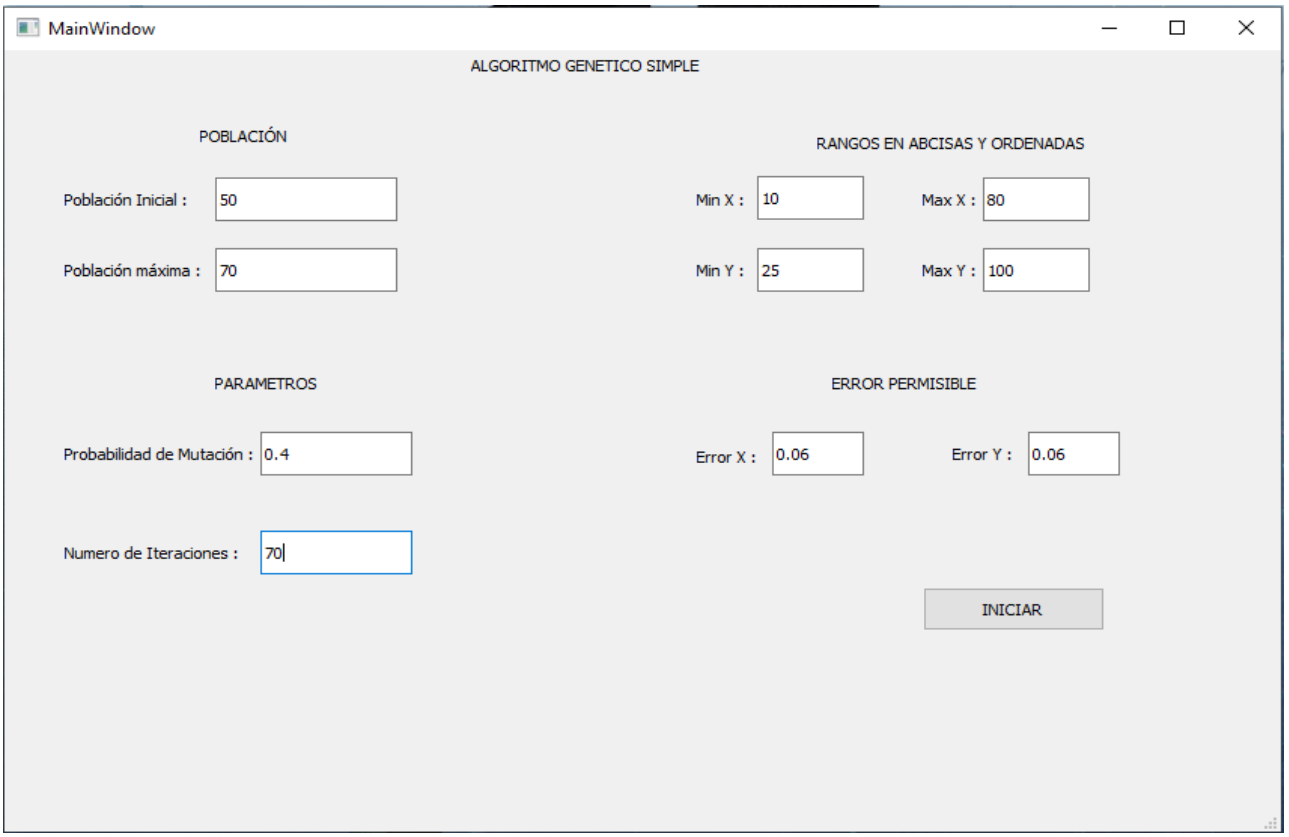


Imagen 3. Interfaz gráfica

2. Minimización de resultados

Los siguientes datos son la representación de cómo se va generando la minimización por cada una de las generaciones (iteraciones) del algoritmo genético.

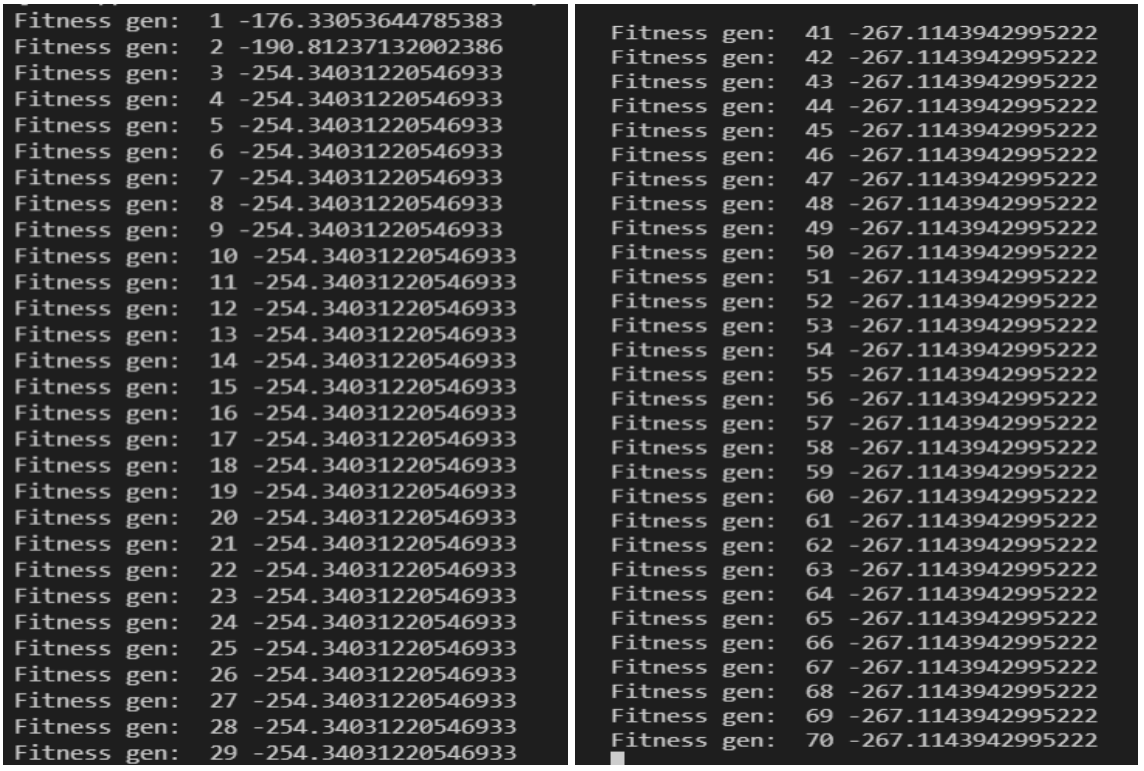


Imagen 4. Minimización fitness.

3. Gráfica de fitness

Todos los valores obtenidos por generación, tanto los óptimos como los no tan óptimos, van siendo guardados para que al final de todas las generaciones estos datos sean representados en una gráfica, en la cual se puede observar como por cada generación va realizando el proceso de minimización.

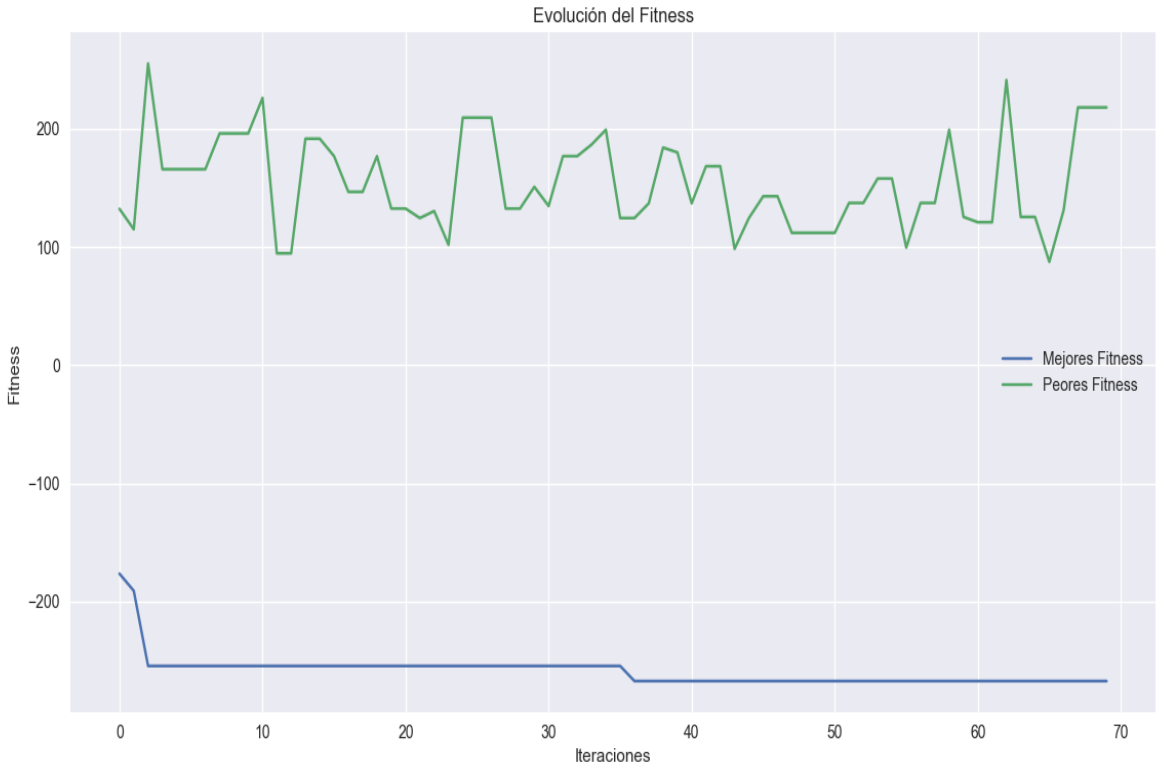


Imagen 5. Gráfica resultados.

Discusiones.

1. **14 de mayo del 2021.**

Se analizó el programa planteado, y se crearon diferentes clases en las cuales se resolvería un problema en específico. Se crearon 2 clases:

- Genetico.py: En esta clase se llevaría a cabo todo el proceso del algoritmo genético, dividido en diferentes funciones, cada una con su proceso correspondiente.
- Calculos.py: En esta clase se llevaría a cabo todo el proceso de operaciones matemáticas, entre otras.

2. **17 de mayo del 2021.**

Se asignó la fórmula correspondiente al problema, así como el objetivo que era la minimización.

3. **19 de mayo del 2021.**

Se implementó la librería de PyQT5 [3] para el desarrollo de la interfaz visual, y matplotlib [2] para la realización de gráficas.

4. **20 de mayo del 2021.**

Se creó una clase más llamada Principal.py, en la cual se ejecuta todo el entorno visual y se muestran las gráficas.

Conclusiones.

Los algoritmos genéticos pueden ayudarnos a resolver diversos problemas, en este trabajo se utilizaron para minimizar una función 3D.

Todos los individuos que representaron a la población fueron generados aleatoriamente para así obtener más diversidad en cuanto a los datos. De igual manera para los métodos de cruza y mutación, se obtuvieron números aleatorios para realizar las validaciones de las probabilidades de que cada una sucedieran.

Se realizaron dos validaciones del fitness, la primera se hacía para obtener a los mejores individuos, los cuales serían los elegidos para realizar la cruza con un individuo de la población generado aleatoriamente y así obtener aún más diversidad. Y la segunda fue realizada tanto con la población actual como su descendencia, y de esta manera obtener a los más óptimos.

Por último, se debe tener cuidado en cuanto a mantener la diversidad de la población, así como conservar aquellos que son más óptimos para poder continuar, ya que el hecho de que el valor de un individuo sea mayor, no significa que siempre será el que tenga un mejor resultado.

Referencias bibliográficas.

[1] Arranz de la peña, J. A. P. (s. f.). *Algoritmos Genéticos*. Algoritmos Genéticos. http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf

[2] *Matplotlib: Python plotting — Matplotlib 3.4.2 documentation*. (s. f.). Matplotlib.org. https://matplotlib.org/

[3] *PyQt5*. (2021, 10 marzo). PyPI. https://pypi.org/project/PyQt5/