Universidad Autónoma

de Coahuila

**Facultad de sistemas**

ANÁLISIS Y MODELACIÓN DE SISTEMAS

*“Proyecto de Alineamientos múltiples del Algoritmo Genético”*

Doc. Ernesto Ríos Willors.

**Michelle Marilu Guajardo Reyes**

1. Introducción

Durante la década de 1960, John Holland se centró en la investigación de la adaptación y la evolución en sistemas complejos. Fue en esta época que desarrolló los algoritmos genéticos, una técnica inspirada en la teoría de la evolución de Charles Darwin. Estos algoritmos imitan el proceso de selección natural y reproducción que ocurre en la naturaleza, aplicándolos a problemas de optimización y búsqueda en el ámbito de la inteligencia artificial. *(Josep., 2023)*

Los algoritmos genéticos siguen un proceso iterativo que simula la evolución natural.

1. Inicialización: En la primera etapa, se crea una población inicial de soluciones potenciales, a menudo generadas de forma aleatoria. Cada solución se representa como un cromosoma en el algoritmo genético.
2. Evaluación y selección: En esta etapa, se evalúa la aptitud de cada solución en función de una función objetivo definida previamente. Las soluciones más aptas (aquellas que se acercan más a la solución óptima) tienen más probabilidades de sobrevivir y reproducirse.
3. Cruce: Durante esta fase, se seleccionan dos soluciones aptas y se combinan entre sí para crear nuevos individuos. Este proceso recuerda a la reproducción sexual, ya que se mezcla la información genética de ambos padres.
4. Mutación: Para introducir variabilidad en la población, se aplican cambios aleatorios en algunas de las características de los individuos. Estos cambios representan mutaciones genéticas y ayudan a explorar nuevas soluciones potenciales.
5. Reemplazo: En esta etapa, se reemplazan los individuos menos aptos de la población con los nuevos individuos surgidos del cruce y la mutación. Este proceso asegura que las mejores soluciones siempre estén presentes en la población.
6. Convergencia: El algoritmo genético continúa iterando a través de las etapas mencionadas anteriormente hasta alcanzar un criterio de convergencia predefinido. Esto puede ser un número máximo de generaciones o cuando se alcanza la solución óptima.

Este proyecto tiene como finalidad crear un programa en el lenguaje de Python en el cual se ejecute el proceso iterativo de los algoritmos genéticos, sin dejar fuera ninguno de sus pasos claves explicados anteriormente.

1. Codigo

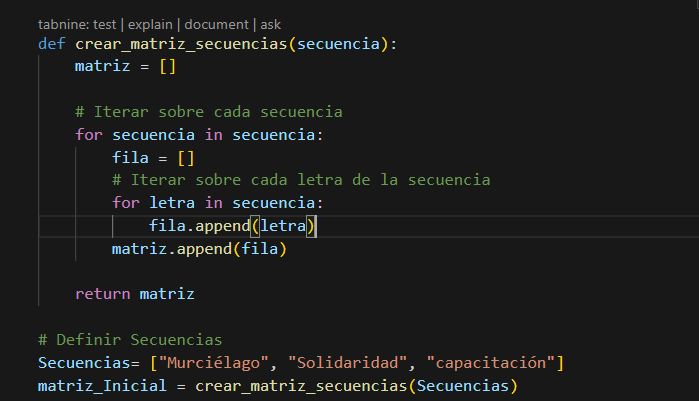
Para iniciar con la creación del programa primero se necesita crear o introducir una poblacion inicial, en este programa se se crea una función llamada crear “crear\_matriz\_secuencias” esta función tomó una lista de secuencias agregada manualmente y crea una matriz donde cada fila representa una secuencia y cada columna representa un carácter de la secuencia.

Imagen 1. Funcion crear\_matriz\_secuencias

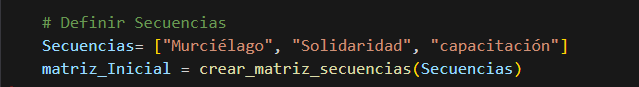
Después de crear la funcion definimos las secuencias y mandamos llamar la funcion “crear\_matriz\_secuencias” sobre las secuencias ingresadas.

Imagen 2. Definición de Secuencia y Función “crear\_matriz\_secuencias”

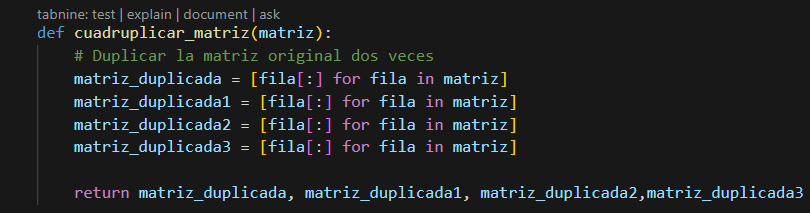
Creamos la función “cuadruplicar\_matriz”, esta función duplica la matriz original cuatro veces creando cuatro copias independientes e idénticas de la matriz de entrada, *matriz\_duplicada = [fila[:] for fila in matriz]* esto crea una copia superficial de la matriz original. Utiliza una comprensión de lista para iterar sobre cada fila de la matriz y crear una nueva lista que contiene los mismos elementos que la fila original. El uso de fila[:] garantiza que cada fila se copie completamente en lugar de solo hacer una referencia a la fila original.

Imagen 3. Duplicar matriz cuatro veces.

Texto

Descripción generada automáticamenteCreamos la población inicial llamando a la función “cuadruplicar\_matriz” para obtener cuatro copias de la matriz inicial y se guardan en variables matriz 1, matriz 2, matriz 3 y matriz 4, continuamos creando una lista llamada población inicial que contiene la matriz inicial y las cuatro matrices duplicadas, al final imprimimos la población inicial numerando a los individuos como matrices.

Imagen 4. Creación de la matriz inicial

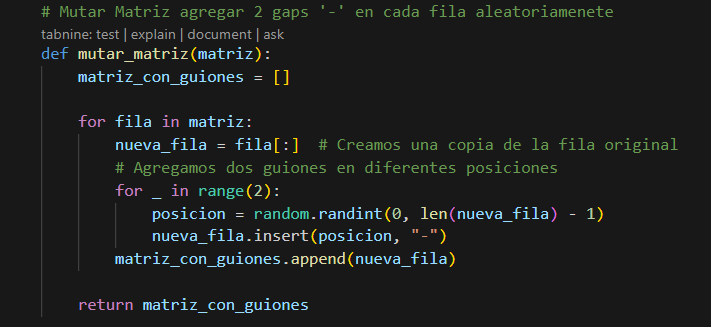
Siguiendo los pasos de los algoritmos genéticos tenemos que llevar a cabo la mutación esta consiste en agregar gaps ”-” en posiciones aleatorias para cada fila del de cada matriz que contiene la población inicial, para esto se crea la función “mutar\_matriz”, que realiza lo que se explicó anteriormente.

Imagen 5. Mutación

Cuando ya tenemos nuestra población mutada, seguiremos con la evaluación de cada matriz, como la población ya ha sido mutada, necesitamos darle un valor a cada individuo de nuestra población, esta evaluación se da si en una columna hay más de un carácter igual sin contar los gaps se sumaran 20 puntos a la puntuación de la matriz, para realizar esto se creó la función “evaluar\_matriz”, esta función evalúa la calidad de la matriz según las reglas predefinidas anteriormente, devolviendo así la puntuación final para cada individuo.

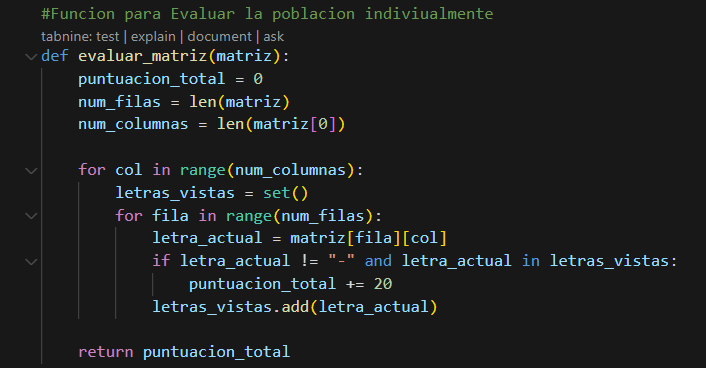


Imagen 6. Evaluación de cada individuo de la población

Captura de pantalla de un celular

Descripción generada automáticamenteObteniendo la puntuación de cada individuo de la población calcularemos un promedio de todas las puntuaciones obtenidas con la función “evaluar\_matriz”, para esto creamos una nueva función llamada “promedio\_puntuaciones”, que calcula no entiendo dicho promedio y además filtra las matrices cuya puntuación sea mayor o igual del promedio, si dicha filtración no es un número par, se selecciona una aleatoria de las que no cumplen la primera condición, pero si no existen que no cumplen la primera condición sólo se mantiene las matrices cuya puntuación sea mayor o igual que el promedio.

Imagen 7. Promedio de la población

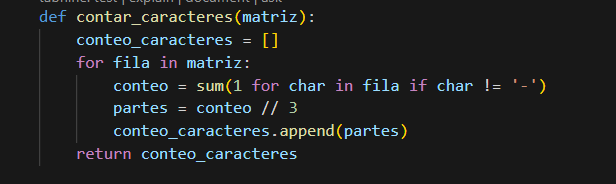
Un paso importante para poder lograr la cruza de 2 individuos es dividir las matrices que tenemos en 3 partes, esto fue un gran desafío, en mi perspectiva lo primero fue crear una función que contabilizará cuántos caracteres distintos a los gaps se encontraban en cada fila de la matriz, obteniendo la cantidad total de caracteres la dividía en 3, consiguiendo así la cantidad de caracteres distintos a los gaps que tendrían que estar en cada partidura de la matriz.

Imagen 7. Contar Caracteres distintos a los gaps por fila.

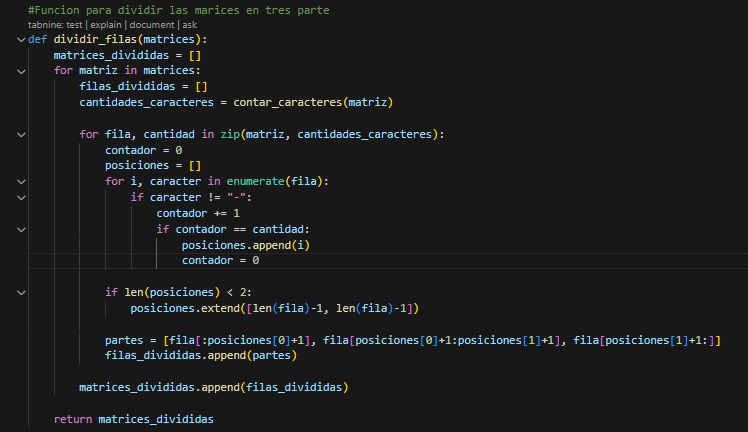
Teniendo en nuestro poder la cantidad de elementos distintos a los gaps que deben contener cada partidura, proseguimos hacer la partición de la matriz creando una función que tomando la matriz y el número de elementos que debe tener cada partición, tomamos la primera fila de la matriz y nuestro posicionamos en el inicio(posición [0] ) de ahí recorremos un puntero hasta la posición en la que se contabilice la cantidad de elementos calculados, tomando así todo lo que se encuentra desde la posición cero a la posición donde se detiene el puntero como la primera partición, prosiguiendo así de la con la posición siguiente recorremos el puntero nuevamente hasta que tengamos la cantidad de elementos calculados, tomando todo lo que se encuentra entre esas 2 posiciones como la partición número 2, y lo que resta de la matriz se toma como la partición 3, este proceso se realiza en cada fila de cada una de las matrices de nuestra población inicial.

Imagen 8. Dividir fila en tres particiones

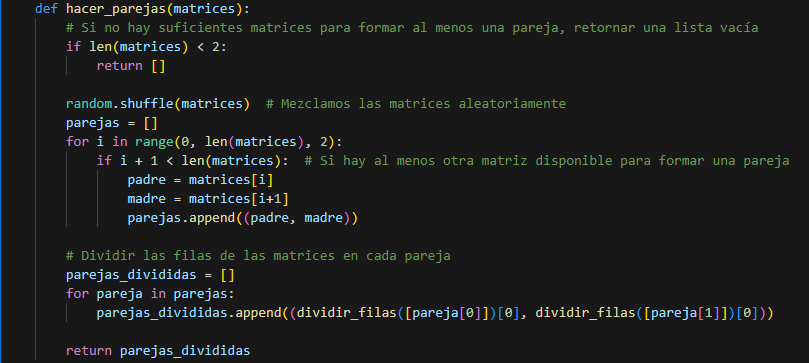
Después de evaluar tenemos qué realizar la cruz, pero antes tenemos que formar parejas entre los individuos de la población, esto se realiza de manera aleatoria en la función “hacer\_pareja”, donde se mezclan las matrices de la población conservada, tomando 2 de ellas definiendo una como padre y otra como madre, al tener las parejas formadas mandamos llamar la función “dividir\_filas”, para que nuestras parejas ya estén divididas para realizar la cruza más fácil.

Imagen 9. Crear parejas y dividier en tres particiones a los individuoas de las parejas

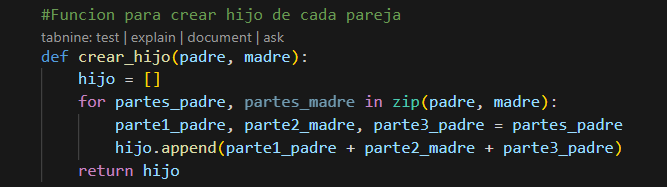
Cuando tenemos nuestras parejas ya definidas y divididas en 3 particiones, es momento de realizar la cruza, teniendo como apoyo la función “crear\_hijo”, que se crea una nueva matriz con la primera partición del padre, seguida de la segunda partición de la madre, finalizando con la tercera partición del padre, así obteniendo nuevos individuos para la población.

Imagen 10. Cruza

Por último, realizamos la iteración del algoritmo genético donde el usuario ingresa el número de generaciones, en cada iteración, la población se muta y se evalúa, se calcula el promedio de las puntuaciones y se conservan las matrices con puntuaciones mayores o iguales al promedio, se forman parejas de padres y madres, se crean hijos a partir de ellas y se muestra la nueva generación, asi hasta terminar las iteraciones.

Texto

Descripción generada automáticamente

Imagen 11. Iteración del Algoritmo Genético.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Conclusiones

En este proyecto, hemos explorado los fundamentos y la implementación de los Algoritmos Genéticos, dado a esto he comprobamos que no existe una solo una forma de desarrollar loa algoritmos genéticos, tanto que no existe una solución única o exacta, nunca se sabe cual es el resultado siempre es cambiante, cómo para mejorar o empeorar la poblacion, este proyecto me ha dejado desarrollar mi pensamiento crítico y abstracto para poder idear múltiples formas de realizarlo, tanto como para elegir la mejor opcion según mi criterio. En resumen, los Algoritmos Genéticos son una herramienta poderosa con los cuales podemos experimentar infinidad de veces y rara ves o nunca obtener una solución igual.