*CARLOS ISLAS ESCAMILLA*

*21081125 | PROYECTO FINAL*

*Algoritmos geneticos*

ANALISIS Y MODELACION DE SISTEMAS

INTRODUCCION:

¿Te has preguntado alguna vez cómo la naturaleza encuentra soluciones óptimas para problemas complejos como la supervivencia, la adaptación y la reproducción? Los algoritmos genéticos imitan este proceso de evolución natural para resolver problemas en el ámbito de la computación.

Para entender mejor cómo funcionan los algoritmos genéticos, primero debemos comprender algunos conceptos básicos de la biología evolutiva. En la naturaleza, los organismos pasan por un proceso de reproducción, selección natural y herencia genética que conduce a cambios y adaptaciones en las poblaciones a lo largo del tiempo. Este proceso se basa en la idea de que los individuos con características más favorables tienen más probabilidades de sobrevivir y reproducirse, transmitiendo sus genes a la siguiente generación.

Ahora, traslademos este concepto al mundo de la informática. En lugar de organismos vivos, tenemos una población de posibles soluciones para un problema dado, representadas como "individuos" o "cromosomas". Cada individuo tiene un conjunto de características o "genes" que representan una posible solución al problema.

El proceso comienza generando una población inicial de manera aleatoria o utilizando alguna heurística específica. Luego, se evalúa cada individuo en términos de su "aptitud" o "fitness", que indica qué tan buena es su solución para el problema en cuestión. Los individuos con una mejor aptitud tienen más probabilidades de ser seleccionados para reproducirse y producir descendencia.

La reproducción en los algoritmos genéticos se realiza mediante operadores genéticos como la cruza y la mutación. Durante la cruza, se combinan características de dos individuos para producir descendencia. La mutación introduce variaciones aleatorias en los genes de los individuos para explorar nuevas posibles soluciones.

Este proceso de selección, cruza y mutación se repite a lo largo de varias generaciones, con la esperanza de que la población evolucione hacia soluciones cada vez mejores. Finalmente, el algoritmo genético converge hacia una solución óptima o aproximadamente óptima para el problema dado.

Los algoritmos genéticos se utilizan en una amplia variedad de campos, desde la optimización y la planificación hasta el diseño de sistemas complejos y la inteligencia artificial. Su capacidad para encontrar soluciones eficientes y adaptativas los convierte en una herramienta poderosa en el arsenal de cualquier estudiante o profesional de la informática.

DESARROLLO:

Una vez entendido el proceso teórico o sus conceptos básicos de un algoritmo genético vamos a realizar el proceso a través de código; como primera etapa de este informe vamos a detallar el funcionamiento de nuestro programa (lógica del programa) y posteriormente especificaremos a fondo línea por línea o sus respectivas funciones.

Lógica del programa:

1. El programa necesita saber ciertas variables que el usuario necesita especificar manualmente como, por ejemplo: Numero de generaciones, Numero de secuencias
2. Posteriormente de acuerdo con el Numero de secuencias que el usuario establecido, el programa tiene que pedir los valores de cada secuencia de acuerdo con el número de secuencias proporcionado anteriormente
3. El siguiente paso es realizar una matriz anexando cada una de las secuencias separando carácter por carácter en cada secuencia; a esta matriz le llamaremos “Población”
4. A continuación, haremos una copia de nuestra matriz población de modo que queden idénticamente iguales; vamos a repetir este proceso hasta lograr tener 4 matrices las cuales todas son iguales
5. El siguiente paso vamos a utilizar nuestra primera función llamada “Mutación” en la cual va agregar aleatoriamente caracteres llamados “Gaps” (“-“), esta toma como parámetro una matriz; La va agregar de manera que ya que es aleatorio puede agregar en medio de cualquier carácter de las secuencias. En este proceso no se remplazan gaps por caracteres si no solo recorrer caracteres a la derecha en caso de anexar gaps. Esto nos puede dar como resultado que en cada fila de la matriz pueda cambiar su longitud, Esta función la vamos a utilizar con nuestras cuatro matrices de modo que van a modificar las matrices por lo que en este paso ninguna matriz va quedar igual o la probabilidad es muy baja
6. Posteriormente vamos a utilizar nuestra segunda función llamada “Filtración” en la cual se divide en dos pasos, esta función toma como parámetros cuatros matrices;

* El primero paso vamos a agregar una función dentro de la función “Filtración” la cual es llamada “Evaluación”; esta toma como parámetro una matriz. El procedimiento de esta función es calcular el pesaje de dicha matriz, te preguntaras ¿cómo determina el pesaje de una matriz? Vamos a obtener el peso de una función de una matriz recorriendo columna por columna de la matriz y va determinar un carácter el cual se repita más número de veces de cierta columna, a cada carácter establecimos que tiene un valor de 20 puntos, por ejemplo si se tiene que en una columna repiten tres veces el carácter ‘a’ se obtendría un valor de 60 puntos en esa columna; posteriormente se va sumar los valores de todas las columnas en caso de que se repitan caracteres y este sería el pesaje de la matriz. NOTA: esta función específica en que columna se repiten los caracteres y cuál es el carácter que se repite más.
* Por último vamos a determinar el pesaje de las cuatro matrices y las vamos a guardar en una lista con el objetivo de posteriormente eliminar dos con los pesajes más bajos y guardar las dos matrices con mayores pesajes

1. Posteriormente vamos a utilizar la función llamada “Cruzar” en la cual va tomar como parámetros dos matrices y tiene como objetivo crear dos matrices nuevas, este proceso funciona de la siguiente manera: tomemos como ejemplo esta secuencia:

**[['a','-','b','f','+','+','+','c','+','f']]** En esta función busca partir la matriz en tres partes de modo que la primera partición va ir recorriendo elemento por elemento que va ir comparando el elemento con las letras del abecedario y en caso de que la comparación sea exitosa va subir un contador a 1 y va agregar el elemento a la primera partición, en caso de que la comparación no sea exitosa aun así va agregar el elemento a la primera partición; esta primera partición va parar hasta que contenga dos caracteres los cuales sean letras, la segunda partición va ser exactamente igual el proceso y la tercera partición va ser el resto de los caracteres.

Primera partición de acuerdo con el ejemplo: **['a','-','b']**

Segunda partición de acuerdo con el ejemplo: **['f','+','+','+','c']**

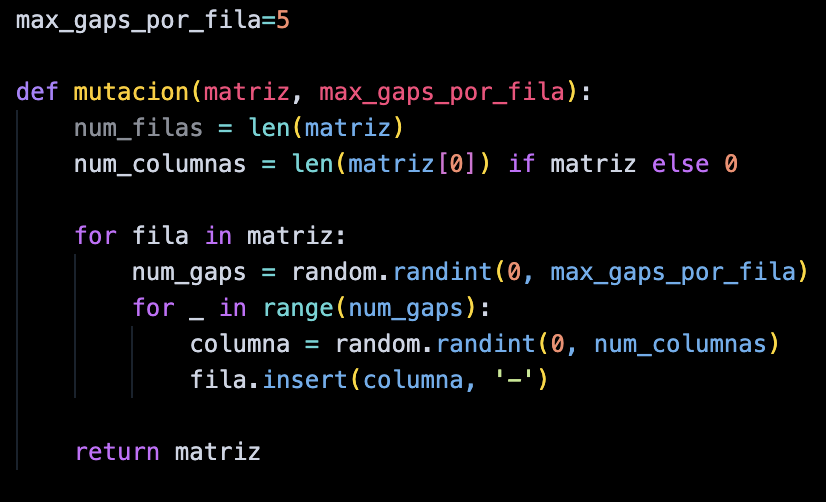
Tercera partición de acuerdo con el ejemplo: **['+','f']**

Este proceso se va repetir con dos matrices y el proceso de la creación de dos matrices es el siguiente: Para la primera matriz vamos a tomar la dos primeras particiones de la primera matriz y la tercera partición de la segunda matriz, para la creación de la segunda matriz el proceso es similar pero al reverso de modo que tomamos las dos primeras particiones de la segunda matriz y le tercera partición de la primera matriz con el objetivo que las secuencias se conserven y no se eliminen caracteres de algunas las posibles secuencias que tenemos

1. Como ultimo parte de este proceso es que el programa se va reiniciar de acuerdo al número de generaciones establecidos por el usuario pero tenemos que contemplar que la en la primera generación las cuatro matrices son las que cuatriplicamos pero para la segunda generación las cuatro matrices que toma como parámetros ahora va tomar las matrices creadas en la primera generación en la función “filtración” y en la función “Cruzar” y así consecutivamente en las siguientes generaciones

Código del programa:

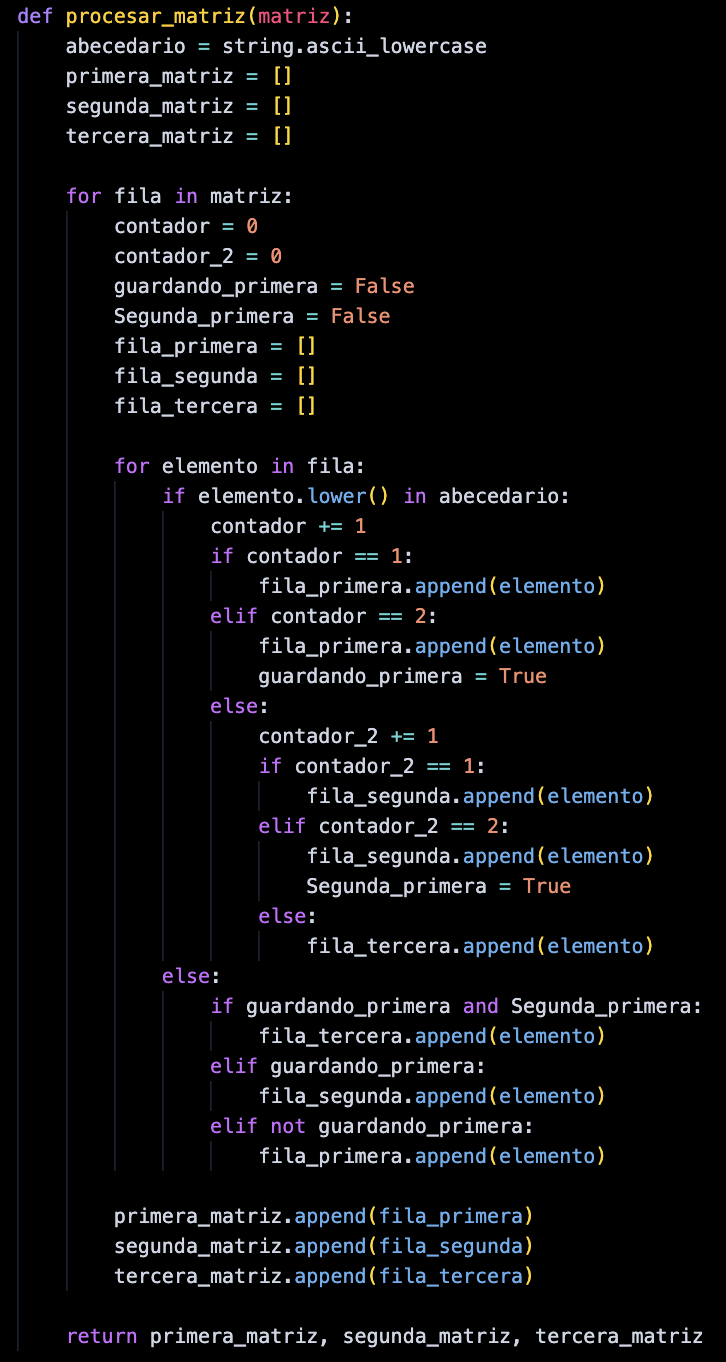
Funciones



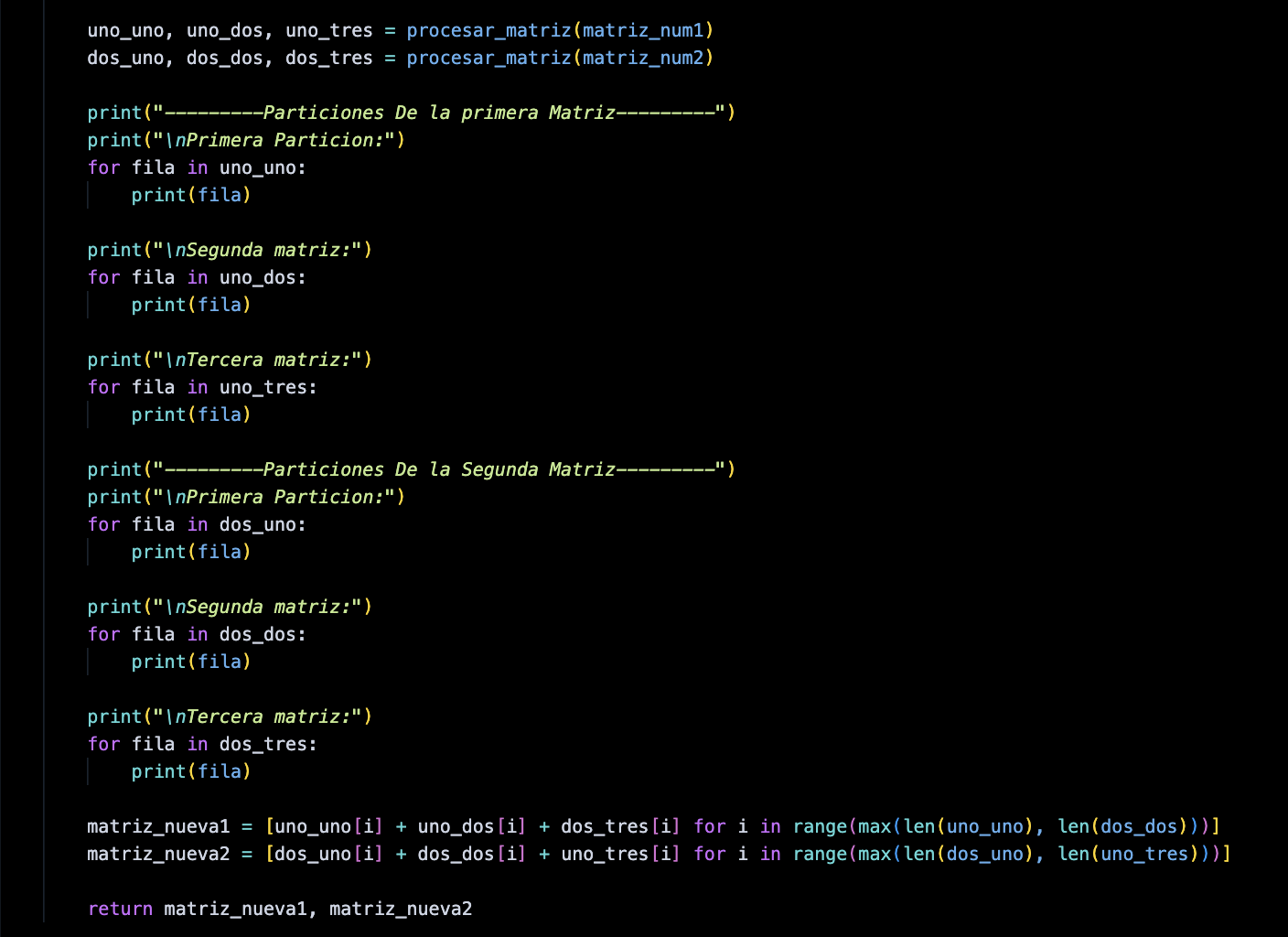
En esta función tiene como objetivo agregar gaps aleatoriamente



En esta función tiene dentro otra función la cual es la que mencionamos para obtener el pesaje de cada matriz de acuerdo y posteriormente guardar solo las dos con mayor pesaje



En función hace las particiones fila por fila hasta partir en tres partes una matriz cumpliendo las normas que comentamos anteriormente



Esta seria la siguiente parte en la función “cruza” en la cual solo une las particiones para crear las nuevas matrices



Conclusión:

En resumen, este proyecto ha demostrado la aplicación efectiva de algoritmos genéticos en la resolución de problemas complejos, como la optimización de secuencias genéticas. Utilizando una combinación de técnicas de mutación, filtración y cruzamiento, pudimos generar y mejorar sucesivas generaciones de secuencias, buscando aquellas con mayores pesos y, por ende, mayor idoneidad para los objetivos planteados.

A lo largo del experimento, se observó cómo las poblaciones de secuencias evolucionaban con el paso de las generaciones, adaptándose gradualmente para cumplir con los criterios de selección establecidos. Además, la introducción de operadores genéticos como la mutación y el cruzamiento permitió una exploración más amplia del espacio de soluciones, incrementando así la diversidad y la calidad de las secuencias obtenidas.

Si bien este proyecto se centró en la optimización de secuencias genéticas, las técnicas de algoritmos genéticos presentadas aquí tienen aplicaciones más amplias en campos como la inteligencia artificial, la optimización combinatoria y la ingeniería de sistemas, donde la búsqueda de soluciones óptimas es un desafío constante.