UNIVERSIDAD METROPOLITANA
FACULTAD DE INGENIERÍA
COMPUTACIÓN EMERGENTE

TAREA 4 - GENETIC ALGORITHM

Pointud, Victor. C.I: 30124022

Caracas, 1 de marzo de 2025.

Los algoritmos genéticos (AG) son una de las técnicas más representativas dentro del campo de la inteligencia computacional y la optimización evolutiva. Inspirados en los principios de la evolución natural y la selección darwiniana, estos algoritmos buscan encontrar soluciones óptimas o cercanas al óptimo en espacios de búsqueda complejos, donde los métodos deterministas o de gradiente no resultan adecuados.

El objetivo principal de esta tarea fue implementar y visualizar el funcionamiento de un algoritmo genético simple, aplicándolo a dos funciones matemáticas distintas:

1. $f_1(x) = \frac{sen^2(x)}{x}$, una función unidimensional no lineal.
2. $f_2(x, y) = 20 + x - 10\,cos(2x) + y - 10\,cos(2y)$, una función bidimensional tipo Rastrigin modificada, con múltiples óptimos locales.

1. Estructura del sistema

El desarrollo se dividió en tres módulos principales:

- **Frontend:** Implementado en HTML, CSS y JavaScript, alojado en la carpeta template y static. Proporciona una interfaz gráfica para ingresar los parámetros y visualizar los gráficos resultantes.
- **Backend:** Implementado en Python dentro de src/ga_server.py, encargado de procesar las peticiones, ejecutar el algoritmo genético, generar los gráficos y devolver los resultados al navegador.
- **Funciones objetivo:** Definidas en src/ga_functions.py, donde se programaron las funciones matemáticas $f_1(x)$ y $f_2(x, y)$ utilizadas para las pruebas.

2. Representación y operadores genéticos

Cada individuo del algoritmo se representa como un vector real de una o dos variables (dependiendo de la función a optimizar).

El proceso evolutivo incluye los siguientes pasos:

- **Inicialización:** Se genera una población aleatoria uniforme dentro de los límites

definidos (por defecto, [-10, 10]).

- **Evaluación:** Se calcula el valor de la función objetivo para cada individuo. En este caso, el algoritmo busca **maximizar** el valor de la función.
- **Selección:** Se aplica una **selección por ruleta**, donde la probabilidad de elegir a un individuo es proporcional a su fitness (valor de la función).
- **Cruzamiento:** Se emplea un **crossover tipo blend (BLX-α simplificado)**, que combina los genes de los padres en proporciones aleatorias, manteniendo la diversidad genética.
- **Mutación:** Se modifica aleatoriamente un subconjunto de los genes con una probabilidad determinada (tasa de mutación), aplicando ruido uniforme dentro del rango definido (mutvar).
- **Elitismo:** Se conservan los mejores individuos (top 2) en cada generación para evitar pérdida de soluciones de alta calidad.
- **Criterio de parada:** El algoritmo detiene la evolución cuando la diferencia entre los mejores valores consecutivos es menor al umbral definido (threshold) o cuando se alcanza el número máximo de iteraciones.

Resultados

Función 1: $f_1(x) = \frac{sen^2(x)}{x}$

El algoritmo mostró una rápida convergencia hacia los picos principales de la función, particularmente alrededor de x \approx 1.4, donde se alcanza un valor máximo local. Durante las primeras generaciones, se observó un incremento sostenido del valor medio y máximo del fitness, estabilizándose a partir de la iteración 40–60. El gráfico de historial mostró la evolución de los valores máximo, mediano y mínimo, confirmando una convergencia estable y sin oscilaciones.

Función 2: $f_2(x, y) = 20 + x - 10\,cos(2x) + y - 10\,cos(2y)$

En esta función bidimensional, el algoritmo logró encontrar un máximo en torno a x, y ≈ 0, donde se concentra el valor más alto del terreno ondulado. La visualización del

mapa de contornos (contour plot) permitió observar cómo el algoritmo se desplaza desde regiones aleatorias hacia la zona óptima. La presencia de múltiples óptimos locales evidenció el papel clave de la mutación para mantener la diversidad y evitar convergencia prematura. Al ajustar la tasa de mutación (mr) y el rango de variabilidad (mutvar), se comprobó que un valor moderado (entre 0.1 y 0.3) produce resultados más robustos.

Conclusiones

- El algoritmo genético implementado demostró ser una herramienta eficaz para la optimización de funciones no derivables o multimodales, donde los métodos clásicos de gradiente no son aplicables.
- El enfoque visual permitió comprender de forma intuitiva cómo la población evoluciona generación tras generación, acercándose progresivamente al máximo de la función objetivo.
- La combinación de selección proporcional, crossover tipo blend, mutación controlada y elitismo aseguró un equilibrio entre exploración y explotación del espacio de búsqueda.
- En términos computacionales, la convergencia se logró en pocas iteraciones, y los resultados fueron consistentes con las expectativas teóricas de ambos problemas.
- Se destaca la importancia de ajustar los parámetros del algoritmo (población, tasas de cruce y mutación) según la complejidad del problema.
  Una mutación excesiva introduce ruido, mientras que una baja mutación puede provocar estancamiento.
- La implementación en entorno web demostró la posibilidad de combinar interfaces interactivas con cómputo evolutivo, ofreciendo una herramienta didáctica y visualmente atractiva para el aprendizaje de algoritmos bioinspirados.