Josue Santana Robledo Corona 325073061

Maestría en Ciencias de la Robótica e Inteligencia Artificial

Algoritmos Bio-Inspirados

Centro Universitario de Ciencias Exactas e Ingeniería.

Mtro. Carlos Alberto López Franco

1. **Introducción**

El presente estudio aplica el algoritmo Simúlate Annealing (Recocido Simulado) a una función objetivo cuadrática. Este método metaheurístico, inspirado en el proceso de recocido en metalurgia, permite escapar de óptimos locales mediante la aceptación controlada de soluciones peores, convergiendo gradualmente al óptimo global.

2. **Descripción del código**

El código implementa el algoritmo de Simulated Annealing para encontrar el mínimo de una función unidimensional dentro de un intervalo dado. El funcionamiento consiste en:

Movimientos amplios con alta temperatura

Movimientos más finos a medida que disminuye la temperatura

Acepta soluciones peores con probabilidad exp(-Δf/T)

3. **Componentes del Código**

Variables y parámetros iniciales

```
rango = [-10, 10]      # Intervalo de búsqueda
T0 = 100               # Temperatura inicial
alpha = 0.95           # Tasa de enfriamiento (95%)
max_iter = 1000        # Máximo de iteraciones
paso = 0.5             # Tamaño de perturbación
```

Función Objetivo

```
def funcion(x):
  return 50 * x * np.exp(-0.01 * x)
```

4. **Algoritmo Principal**

Consiste en varias partes

Generación de vecino

```
x_vecino = x_actual + np.random.normal(0, paso)
```

Criterio de aceptación

```
if f_vecino < f_actual:
  aceptar = True
else:
  probabilidad = exp(-(f_vecino - f_actual) / T)
```

aceptar = random() < probabilidad

Enfriamiento gradual

T = T0 * (alpha ** iteracion)

Criterio de parada

Despues de max_iter iteraciones

5. Resultados Obtenidos

El código genera una tabla detallada que muestra en cada iteración

Primero vamos con el descendente

997 | -10.0000 | -552.5855 | 0.00 | 1.0000 | 1 | -552.5855
998 | -10.0000 | -552.5855 | 0.00 | 0.0000 | 0 | -552.5855
999 | -10.0000 | -552.5855 | 0.00 | 0.0000 | 0 | -552.5855
Minimo en -10 con valor de -552.5854590378239

En esta tabla podemos ver el número de ciclos y las variables que se fueron utilizando durante el código, lo que nos ayuda a ver la evolución del código hacia encontrar el mínimo. Y finalmente vemos el valor final al que llega y su evaluación en la función.

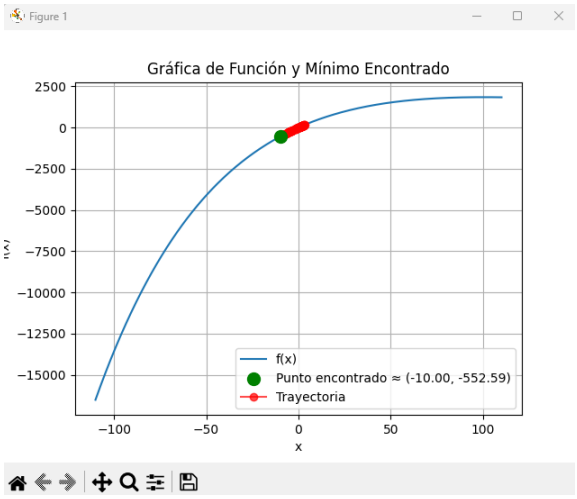

Imagen 1. Descendente

Ahora vamos con el ascendente

Para lograr este cometido, solo debemos cambiar un signo
if f_vecino>f_actual:

math.exp((f_vecino - f_actual) / temperatura)
y quitar el negativo aquí

997 | 10.0000 | 452.4187 | 0.00 | 1.0000 | 1 | 369.2609
998 | 10.0000 | 452.4187 | 0.00 | 0.0000 | 0 | 369.2609
999 | 10.0000 | 452.4187 | 0.00 | 0.0000 | 0 | 369.2609
Minimo en 10 con valor de 452.4187090179798

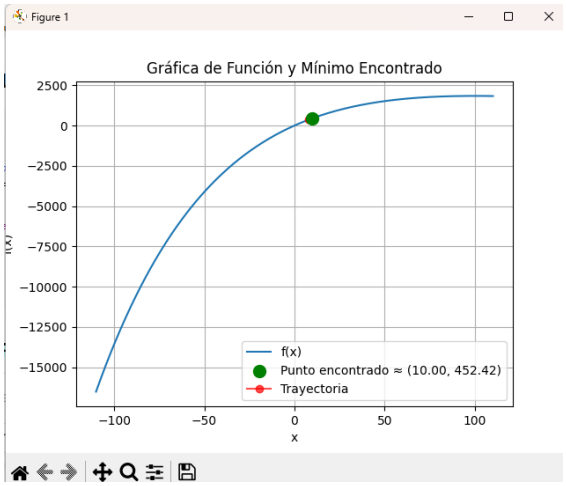

Imagen 2. Ascendente

6. Ventajas del Método

Acepta soluciones peores temporalmente
Aplicable a diversos tipos de funciones
No requiere información derivadas

7. Limitaciones

T0, alpha y paso requieren ajuste
No hay garantía del óptimo global

8. Conclusiones

El código implementa correctamente el algoritmo de Simulated Annealing y demuestra su efectividad para encontrar el mínimo de la función cuadrática especificada. Implementarlo no fue ya tan difícil, a estas alturas esta bien comprendido el concepto de generar vecinos, eso vino directamente de anteriores códigos, lo nuevo fue la función de aceptación, que, si requirió investigación para comprenderla, pero posteriormente basta con un criterio de evaluación y listo.

9. Código

https://github.com/santana-robledo/Algorithm-Lab/blob/main/15.%20Simulated%20Annealing.py

Aquí se encuentra el repositorio