INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
INTELIGENCIA ARTIFICIAL
ING. EN SISTEMAS COMPUTACIONALES

Laboratorio 6: Búsqueda informada III

Alumno: Hurtado Morales Emiliano - 2021630390

Maestro: Andrés García Floriano

Grupo: 6CV3

Fecha de entrega: 17/10/2024

Ciclo Escolar: 2025 – 1

Introducción

La optimización numérica es un campo fundamental en muchas disciplinas, incluyendo la ingeniería, la ciencia de datos y la investigación operativa. Los problemas de optimización pueden tener formas muy complejas, con múltiples mínimos locales que complican el uso de técnicas tradicionales como el descenso por gradiente, las cuales suelen quedar atrapadas en un mínimo local. Para abordar este tipo de problemas, se han desarrollado algoritmos estocásticos como el **recocido simulado**, un método inspirado en la física.

El recocido simulado toma su nombre del proceso físico de enfriamiento de materiales, conocido como "recocido". Este proceso implica calentar un material hasta un estado líquido y luego enfriarlo lentamente para permitir que alcance una estructura cristalina de baja energía. De manera análoga, el recocido simulado explora el espacio de soluciones de un problema de optimización mediante una "temperatura" que controla la probabilidad de aceptar soluciones peores, favoreciendo la exploración al inicio del proceso y volviéndose más conservador a medida que la temperatura desciende.

En este reporte, se aplica el algoritmo de recocido simulado para minimizar la **función de Himmelblau**, una función no lineal multivariable que tiene múltiples mínimos locales. La función de Himmelblau se define como:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

El dominio de la función está limitado a $[-5,5]$ para ambas variables x e y, y es conocida por tener cuatro mínimos locales dentro de este rango. Dada la complejidad de la función, el objetivo del recocido simulado será aproximar uno de estos mínimos.

Explicación de los Algoritmos

Link: https://github.com/EmilianoHM/Bender-IA/tree/main/Lab_6

El **recocido simulado** es un algoritmo inspirado en el proceso físico de enfriamiento controlado de los metales. En la metalurgia, el objetivo es enfriar lentamente un material para que los átomos encuentren una estructura estable de mínima energía. Este concepto se traslada al campo de la optimización, donde el objetivo es encontrar el mínimo global de una función compleja. A diferencia de los algoritmos deterministas, el recocido simulado permite aceptar soluciones subóptimas (de mayor costo) durante las primeras etapas del proceso, lo que ayuda a escapar de mínimos locales y explorar mejor el espacio de soluciones.

En este contexto, el **recocido simulado** se utiliza para minimizar la **función de Himmelblau**, que presenta múltiples mínimos locales. A continuación, se explican los pasos del algoritmo en detalle.

```python
import numpy as np
import random
import math
```

Primero, se importan las bibliotecas necesarias:

- **numpy**: para manejar operaciones matemáticas si se necesita en otros contextos.
- **random**: para generar números aleatorios, los cuales se utilizarán para explorar nuevas soluciones.
- **math**: para funciones matemáticas como la exponencial, utilizada en la aceptación probabilística de soluciones peores.

```python
# Función de Himmelblau
def funcion_himmelblau(x, y):
    """
    Calcula el valor de la función de Himmelblau dada una pareja de coordenadas
(x, y).

    Parámetros:
    x (float): Coordenada x.
    y (float): Coordenada y.

    Retorna:
    float: El valor de la función de Himmelblau en el punto (x, y).
    """
    return (x**2 + y - 11)**2 + (x + y**2 - 7)**2
```

La **función de Himmelblau** es la función objetivo que queremos minimizar. Dada una pareja de coordenadas *(x, y)*, se calcula su valor usando la fórmula:

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

El valor retornado es el "costo" o "energía" en el contexto del recocido simulado, que intentaremos minimizar.

```
# Algoritmo de Recocido Simulado
def recocido_simulado(limites, max_iteraciones, temp_inicial, alfa):
    """
    Implementa el algoritmo de recocido simulado para minimizar la función de
Himmelblau.

    Parámetros:
    limites (list): Lista con dos valores [min, max] que definen los límites del
rango de búsqueda para x e y.
    max_iteraciones (int): Número máximo de iteraciones que realizará el
algoritmo.
    temp_inicial (float): Temperatura inicial del recocido simulado, que afecta
la probabilidad de aceptar soluciones peores.
    alfa (float): Factor de enfriamiento que determina cómo se reduce la
temperatura en cada iteración.

    Retorna:
    tuple: Una tupla que contiene las mejores coordenadas encontradas (mejor_x,
mejor_y)
           y el costo mínimo asociado (mejor_costo).
    """
```

Esta función implementa el algoritmo de recocido simulado. Sus parámetros son:

- **limites**: define el rango de búsqueda para las coordenadas x e y. En este caso, los límites son [−5, 5].
- **max_iteraciones**: define el número máximo de iteraciones del algoritmo.
- **temp_inicial**: es la temperatura inicial, un valor alto que permite mayor exploración al inicio.
- **alfa**: es el **factor de enfriamiento**, un valor entre 0 y 1 que controla la velocidad de enfriamiento del proceso.

```
    # Generar un punto inicial aleatorio dentro de los límites establecidos
    x = random.uniform(limites[0], limites[1])
    y = random.uniform(limites[0], limites[1])

    # Evaluar la función objetivo en el punto inicial (costo inicial)
    costo_actual = funcion_himmelblau(x, y)

    # Almacenar el mejor punto encontrado hasta ahora
    mejor_x, mejor_y = x, y
    mejor_costo = costo_actual

    # Establecer la temperatura inicial
    temperatura = temp_inicial
```

- Se **genera un punto inicial aleatorio** dentro de los límites dados.

- Se **calcula el valor de la función objetivo** en ese punto usando la función de Himmelblau.
- Se almacenan las coordenadas y el valor como la **mejor solución encontrada** hasta el momento.
- Se **establece la temperatura inicial**.

```
    # Bucle principal del algoritmo
    for i in range(max_iteraciones):
```

El algoritmo realiza tantas iteraciones como se especifique en max_iteraciones. En cada iteración se genera una nueva solución cercana y se decide si se acepta o no.

```
        # Generar un nuevo punto vecino aleatorio agregando un pequeño valor a
las coordenadas actuales
        x_nuevo = x + random.uniform(-0.01, 0.01)
        y_nuevo = y + random.uniform(-0.01, 0.01)

        # Asegurarse de que el nuevo punto esté dentro de los límites definidos
        x_nuevo = max(min(x_nuevo, limites[1]), limites[0])
        y_nuevo = max(min(y_nuevo, limites[1]), limites[0])
```

- Se **genera un nuevo punto vecino** añadiendo un pequeño valor aleatorio entre −0.01 y 0.01 a las coordenadas actuales.
- El nuevo punto se **limita** al rango definido por los límites, para asegurar que se mantenga dentro del dominio permitido.

```
        # Evaluar la función objetivo en el nuevo punto
        nuevo_costo = funcion_himmelblau(x_nuevo, y_nuevo)

        # Calcular el cambio en la función objetivo (diferencia de costos entre
el punto actual y el nuevo)
        delta_costo = nuevo_costo - costo_actual
```

- Se **calcula el costo** de la nueva solución utilizando la función de Himmelblau.
- Se **calcula la diferencia de costo** entre la nueva y la solución actual. Esta diferencia (delta_costo) se utilizará para decidir si se acepta el nuevo punto.

```
        # Condiciones para aceptar el nuevo punto:
        # Si el nuevo costo es menor, se acepta automáticamente.
        # Si es mayor, se acepta con una probabilidad dependiente de la
temperatura.
        if delta_costo < 0 or random.random() < math.exp(-delta_costo /
temperatura):
            # Actualizar las coordenadas actuales y el costo si se acepta el
nuevo punto
            x, y = x_nuevo, y_nuevo
            costo_actual = nuevo_costo
```

- **Si el nuevo costo es menor**, se acepta la nueva solución inmediatamente.
- **Si el nuevo costo es mayor**, se acepta con una probabilidad dada por la fórmula:

$$P = e^{(-\Delta costo/temperatura)}$$

A mayor temperatura, más probable es aceptar una solución peor, lo que permite una mejor exploración del espacio de soluciones.

```
        # Actualizar la mejor solución encontrada
        if costo_actual < mejor_costo:
            mejor_x, mejor_y = x, y
            mejor_costo = costo_actual
```

- Si la nueva solución es mejor que la mejor encontrada hasta el momento, se **actualiza la mejor solución**.

```
        # Enfriar la temperatura multiplicándola por el factor alfa
        temperatura *= alfa
```

- La **temperatura se reduce** multiplicándola por el factor de enfriamiento alfa. Esto hace que el algoritmo se vuelva más conservador a medida que avanza.

```
    # Imprimir los parámetros utilizados
    print(f"Parámetros del recocido simulado:\n - Temperatura inicial:
{temp_inicial}\n - Factor de enfriamiento: {alfa}\n - Iteraciones máximas:
{max_iteraciones}")

    # Devolver las mejores coordenadas encontradas y el costo asociado
    return mejor_x, mejor_y, mejor_costo
```

- Se **imprimen los parámetros** utilizados en la ejecución del algoritmo.
- Se **retornan las mejores coordenadas** encontradas y su costo asociado.

```
# Parámetros ajustados para el recocido simulado
limites = [-5, 5]  # Límites de búsqueda
max_iteraciones = 50000  # Mayor número de iteraciones para exploración
temp_inicial = 10000  # Alta temperatura inicial
alfa = 0.9995  # Enfriamiento lento

# Ejecutar el algoritmo de recocido simulado
mejor_x, mejor_y, mejor_costo = recocido_simulado(limites, max_iteraciones,
temp_inicial, alfa)

print(f"Los valores mínimos encontrados son: x = {mejor_x}, y = {mejor_y}")
print(f"El valor mínimo de la función es: {mejor_costo}")
```

- Se **definen los parámetros** para la ejecución del algoritmo.
- Se **ejecuta el algoritmo** de recocido simulado.
- Se **imprimen los resultados** finales.

Resultados Esperados

A continuación, se presentan algunas combinaciones de parámetros que pueden utilizarse para ajustar la exploración del algoritmo:

1. **Exploración rápida**:
    - max_iteraciones = 10000
    - temp_inicial = 1000
    - alfa = 0.9

    Esta configuración hará que el algoritmo busque soluciones rápidamente, pero con menor precisión.

2. **Exploración exhaustiva**:
    - max_iteraciones = 100000
    - temp_inicial = 50000
    - alfa = 0.9999

    Esta configuración permite una exploración muy lenta, útil para alcanzar una solución más precisa, pero a costa de mayor tiempo de ejecución.

3. **Exploración intermedia**:
    - max_iteraciones = 25000
    - temp_inicial = 5000
    - alfa = 0.95

    Esta configuración ofrece un balance entre velocidad y precisión.

Se espera que el algoritmo se acerque a uno de los mínimos conocidos de la función de Himmelblau, que están aproximadamente en:

- (3.0, 2.0)
- (−2.805, 3.131)
- (−3.779, −3.283)
- (3.584, −1.848)

Dependiendo de los parámetros seleccionados, el recocido simulado puede acercarse más o menos a estos valores, con la posibilidad de que, en algunas ejecuciones, se quede atrapado en un mínimo local cercano.

```
Parámetros del recocido simulado:
 - Temperatura inicial: 10000
 - Factor de enfriamiento: 0.9995
 - Iteraciones máximas: 50000
Los valores mínimos encontrados son: x = 2.999977368152124, y = 2.0000714077010167
El valor mínimo de la función es: 7.331628754797774e-08
```

```
Parámetros del recocido simulado:
 - Temperatura inicial: 1000
 - Factor de enfriamiento: 0.9
 - Iteraciones máximas: 10000
Los valores mínimos encontrados son: x = -3.779421128530893, y = -3.283170098252405
El valor mínimo de la función es: 7.755881339248699e-07
```

```
Parámetros del recocido simulado:
 - Temperatura inicial: 50000
 - Factor de enfriamiento: 0.9999
 - Iteraciones máximas: 100000
Los valores mínimos encontrados son: x = -2.803287167089878, y = 3.1303823944652325
El valor mínimo de la función es: 0.00014135598144533396
```

```
Parámetros del recocido simulado:
 - Temperatura inicial: 5000
 - Factor de enfriamiento: 0.95
 - Iteraciones máximas: 25000
Los valores mínimos encontrados son: x = -3.7792079170462514, y = -3.283088057353273
El valor mínimo de la función es: 7.487876147261117e-07
```

Conclusión

El recocido simulado es un algoritmo de optimización estocástica robusto que permite abordar problemas complejos con múltiples mínimos locales, como el caso de la función de Himmelblau. A diferencia de los métodos de optimización tradicionales, que suelen quedarse atrapados en un mínimo local, el recocido simulado aprovecha su capacidad de aceptar soluciones peores al principio del proceso para realizar una exploración más amplia del espacio de búsqueda. Esto es especialmente útil en problemas no lineales y multivariables, como el que se ha tratado en este informe.

En este trabajo, se ha demostrado la eficacia del recocido simulado para encontrar aproximaciones cercanas a los mínimos locales de la función de Himmelblau. El algoritmo, al simular el proceso de enfriamiento físico de los materiales, explora y explota de manera progresiva las posibles soluciones. Se ha observado que, a medida que la temperatura disminuye, el algoritmo se vuelve más restrictivo, favoreciendo la explotación de las mejores soluciones encontradas.

Es importante destacar que el rendimiento del algoritmo depende en gran medida de los parámetros seleccionados, como la temperatura inicial, el factor de enfriamiento (α) y el número de iteraciones. Los resultados obtenidos pueden variar dependiendo de la configuración de estos parámetros. Por ejemplo, una temperatura inicial alta y un enfriamiento lento permiten una exploración más exhaustiva, mientras que una menor cantidad de iteraciones y un enfriamiento rápido pueden ofrecer resultados más rápidos a costa de precisión.

En pruebas realizadas con configuraciones que favorecen la exploración exhaustiva, el recocido simulado ha logrado acercarse a los mínimos esperados de la función de Himmelblau, que se encuentran en puntos como $(3.0, 2.0)$ y $(-2.805, 3.131)$. Estos resultados muestran que, con un ajuste adecuado de los parámetros, el recocido simulado es capaz de encontrar soluciones óptimas o cercanas al mínimo global, incluso en escenarios donde hay múltiples mínimos locales.

En resumen, el recocido simulado es una técnica flexible y potente para la optimización de funciones complejas, como la función de Himmelblau. La capacidad de controlar la probabilidad de aceptar soluciones peores mediante la temperatura es lo que le da su fortaleza, permitiéndole superar las limitaciones de otros algoritmos de optimización locales. Sin embargo, la selección de los parámetros debe hacerse cuidadosamente, y en algunos casos puede ser recomendable realizar múltiples ejecuciones del algoritmo para mejorar la probabilidad de alcanzar el mínimo global.

En aplicaciones más amplias, este enfoque puede extenderse a una variedad de problemas de optimización en los que las funciones a minimizar presenten características similares, siendo el recocido simulado una herramienta valiosa dentro del conjunto de técnicas de optimización disponibles.