Instituto Politécnico Nacional

Escuela Superior de Computo

Ingeniería en Sistemas Computacionales

Sexto semestre

Grupo: 6CV3

Inteligencia Artificial

Practica 6: Función Himmelblau-Recocido Simulado

Presentado por:

Flores Vicencio Marco Isaí

Pérez González Jesús

Reyes Núñez Sebastián

Fecha: 10/10/2024

Indice

Introducción

La función de Himmelblau es una función multimodal que se usa en la optimización matemática para comprobar el rendimiento de los algoritmos de optimización.

Fue propuesta por el estadounidense David Mautner Himmelblau. Es una función definida en un espacio $\mathbb{R}^2$ que permite obtener parámetros de algoritmos de optimización como ratio de convergencia, precisión y robustez.

Su representación matemática es mostrada como una función de dos variables tal que:

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

Manteniendo su dominio en todos los reales dentro del intervalo [-5,5]

La función Himmelblau es conocida por tener valores de x,y para hallar cuatro mínimos locales y un máximo local, representados de la siguiente forma:

$$f(x) = 0 \; con \; (x,y) = 3{,}2$$

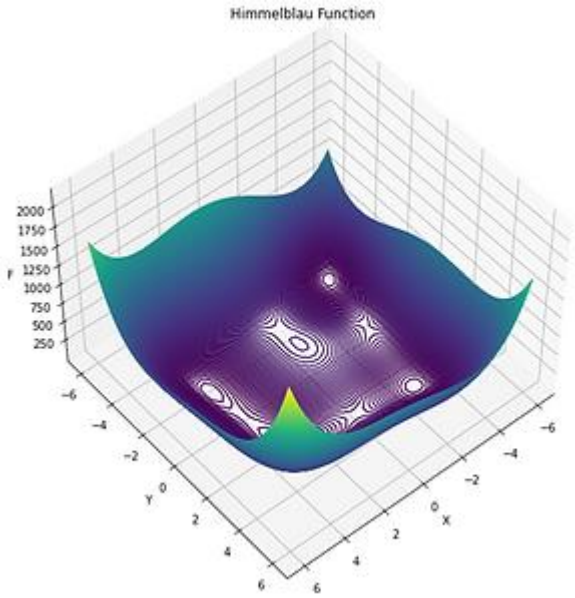
$$f(x,y) = 0 \; con \; (x,y) = -2.805118, 3.283186$$

$$f(x,y) = 0 \; con \; (x,y) = -3.779310, -3.283186$$

$$f(x,y) = 0 \; con \; (x,y) = 3.584458, -1.848126$$

$$M\acute{a}x \; f(x,y) = 181.617 \; con \; (x,y) = -0.270845\,, -0.923039$$

Finalmente, la función Himmelblau puede ser graficada de la siguiente forma:



Recocido Simulado

El recocido simulado es un algoritmo de optimización que está inspirado en el proceso de recocido en la metalurgia, donde un material (como el metal o el vidrio) se calienta a una temperatura elevada y luego se enfría lentamente para minimizar su energía interna y alcanzar un estado estable y óptimo.

En el contexto de la optimización, el recocido simulado se utiliza para encontrar una solución aproximada a problemas complejos de optimización global. Es útil cuando el espacio de búsqueda es grande, tiene muchas soluciones posibles y existen múltiples óptimos locales, lo que podría hacer que otros métodos de optimización se queden atascados en soluciones subóptimas.

¿Para qué se utiliza?

El recocido simulado es ideal para resolver problemas de optimización complejos, como aquellos que tienen múltiples óptimos locales, grandes espacios de búsqueda, o no son fácilmente resolubles con métodos deterministas. Algunos ejemplos incluyen:

- Optimización de rutas (como el problema del viajante).
- Optimización de diseño de circuitos.
- Ajuste de parámetros en modelos de machine learning.
- Problemas de asignación de tareas o distribución de recursos.

Desarrollo

Para el desarrollo de esta práctica, se utilizó un algoritmo para encontrar los valores donde la función de Himmelblau es mínima. Para hallar estos valores, se toma un dominio de [-5,5] para los valores de x,y. Se utiliza el algoritmo de recocido simulado en n iteraciones para encontrar el mejor valor.

```
import numpy as np #Importar biblioteca para manejar expresiones matematicas complejas
```

Importamos la biblioteca *numpy*, ya que se encarga de manejar expresiones matemáticas complejas y realizar operaciones sobre esta con mayor facilidad.

```
def himmelblau(x, y): #Definir metodo para la función Himmelblau
    return (x**2 + y - 11)**2 + (x + y**2 - 7)**2
```

Definimos a la función Himmelblau como fue mencionada en la introducción del reporte.

El proceso imita el enfriamiento físico de un material, y el algoritmo se basa en los siguientes pasos:

1. **Inicialización**:
    - Comienza con una **solución aleatoria** y una temperatura inicial elevada.

```
# Inicializar la solución con un punto aleatorio dentro de los límites
x = random.uniform(limites[0], limites[1])
y = random.uniform(limites[0], limites[1])
```

2. **Generación de vecinos**:

    Para la primera iteración tomamos a los primeros valores generados para evaluar la función, pero con cada iteración se toman vecinos de los puntos ya tomados, siempre y cuando no se salgan los límites establecidos ([-5,5] y no difieran en mas de una unidad de los puntos anteriores.

```
# Evaluar la función en la solución inicial
current_energy = himmelblau(x, y)

# Guardar la mejor solución encontrada
best_x, best_y = x, y
best_energy = current_energy
```

- En cada paso, se modifica ligeramente la solución actual para generar una nueva solución "vecina" en el espacio de búsqueda.

```
# Generar un nuevo punto vecino aleatorio
new_x = x + random.uniform(-1, 1)
new_y = y + random.uniform(-1, 1)

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

3. **Evaluación de soluciones**:
    - Se calcula la **energía** (o función objetivo) tanto de la solución actual como de la nueva. En el contexto de optimización, la "energía" corresponde al valor de la función que se desea minimizar.

```
# Evaluar la función en el nuevo punto
new_energy = himmelblau(new_x, new_y)

# Calcular la diferencia de energía
delta_energy = new_energy - current_energy
```

4. **Decisión de aceptación**:
    - Si la nueva solución es mejor (tiene menor energía), se acepta directamente.
    - Si la nueva solución es peor (mayor energía), se puede aceptar con cierta probabilidad, que depende de la diferencia de energías y de la temperatura actual. Esta probabilidad disminuye conforme baja la temperatura.

```
# Decidir si aceptar la nueva solución
if delta_energy < 0 or math.exp(-delta_energy / temp) > random.random():
    x, y = new_x, new_y
    current_energy = new_energy
```

5. **Enfriamiento**:
    - A lo largo del tiempo, la temperatura disminuye (controlado por una tasa de enfriamiento). Al principio, es más probable aceptar soluciones peores para explorar el espacio de búsqueda de manera más amplia, pero a medida que el proceso avanza y la temperatura baja, se hace menos probable aceptar soluciones peores, enfocándose más en mejorar la solución actual.

```
# Reducir la temperatura de acuerdo con la tasa de enfriamiento
temp *= enfriamiento
```

6. **Criterio de parada**:

   - El algoritmo termina cuando la temperatura llega a un valor muy bajo o se alcanza un número máximo de iteraciones. Al final, se espera haber encontrado una solución cerca del óptimo global.

```
# Actualizar la mejor solución encontrada
if current_energy < best_energy:
    best_x, best_y = x, y
    best_energy = current_energy
```

Resultado

```
# Parámetros iniciales
limites = (-5, 5)
temp_inicial = 10000
enfriamiento = 0.99
iteraciones = 10000


best_x, best_y, best_energy = recocido(himmelblau, limites, temp_inicial, enfriamiento, iteraciones)

print(f"Los valores (x, y) que minimizan la función son: x = {best_x}, y = {best_y}")
print(f"Valor mínimo de la función de Himmelblau: {best_energy}")
```

```
Los valores (x, y) que minimizan la función son: x = -2.8067722903423977, y = 3.13908994180426
Valor mínimo de la función de Himmelblau: 0.0025107350102548964
```

Conclusión

En esta práctica se implementó y evaluó el algoritmo de recocido simulado para la optimización de la función de Himmelblau. Este enfoque permitió identificar múltiples mínimos locales dentro del espacio de búsqueda definido, destacando la capacidad del algoritmo para explorar soluciones subóptimas inicialmente y refinar la búsqueda con el enfriamiento progresivo. A lo largo de las iteraciones, se observó cómo el recocido simulado es eficaz para evitar quedar atrapado en mínimos locales, gracias a su probabilidad controlada de aceptar soluciones peores en las primeras fases.

Los resultados obtenidos demuestran que el recocido simulado es un método robusto para optimización en funciones multimodales como la de Himmelblau, confirmando su utilidad en problemas donde otros métodos deterministas podrían fallar. Además, el comportamiento del algoritmo se alinea con las expectativas teóricas de encontrar soluciones cercanas al óptimo global a medida que la temperatura disminuye.

Referencias

- Limited, I. P. (2021, July 4). HIMMELBLAU FUNCTION. *Indusmic*. https://www.indusmic.com/post/himmelblau-function

- *Minimization of the Himmelblau Function — algopy documentation*. (n.d.). https://pythonhosted.org/algopy/examples/minimization/himmelblau_minimization.html

- Kleimann, S. G. A. (2020, April 27). *Algoritmo de recocido simulado o Simulated Annealing*. Data Science Street. https://medium.com/estudio-de-datos/algoritmo-de-recocido-simulado-o-simulated-annealing-234f567677d9

- Flores, M. Á. (2021, December 10). Graficando la función de Himmelblau - Miguel Ángel Flores - Medium. *Medium*. https://medium.com/@hdezfloresmiguelangel/graficando-la-funci%C3%B3n-de-himmelblau-6045c4b865f