INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Ingeniería en Sistemas Computacionales

Inteligencia Artificial

Profesor:

Andrés García Floriano

Laboratorio 6:

Búsqueda Informada III

Alumnos:

Carmona Santiago Yeimi Guadalupe

Hernández Suárez Diego Armando

Flores Osorio Adolfo Ángel

Grupo:

6CV3

17 de octubre de 2024

Contenido

Introducción 1

Función de Himmelblau 1

Desarrollo 2

Programa para la resolución de la Función de Himmelblau 2

Algoritmo 2

Código 3

Ejecución 4

Reporte de resultados 5

Conclusiones 8

Bibliografía 8

Búsqueda Informada

Introducción

Función de Himmelblau

La función de Himmelblau es una conocida función matemática utilizada principalmente en pruebas de optimización no lineal y en el campo del aprendizaje automático. Es una función de dos variables que tiene múltiples mínimos locales, lo que la convierte en un desafío interesante para los algoritmos de optimización.

La fórmula que define la función es:

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

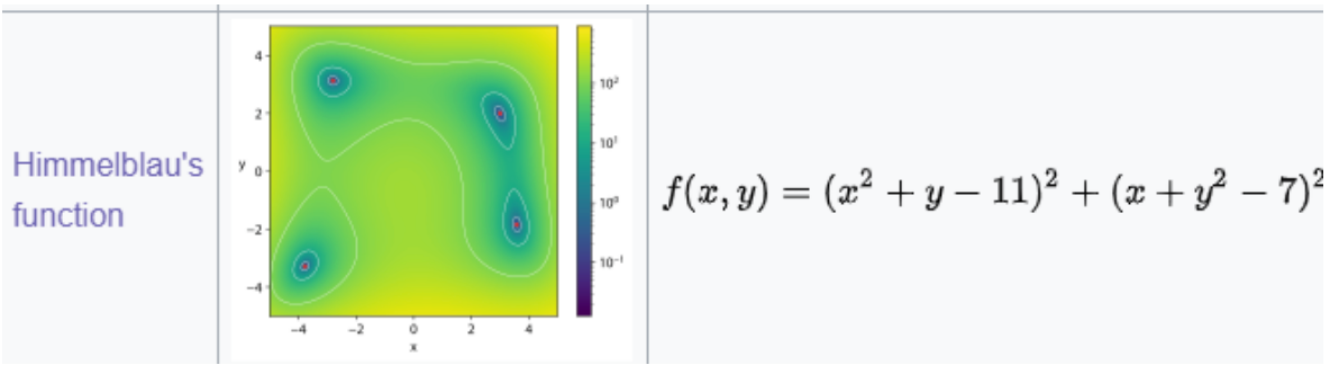
Algunas de las características de esta función son las siguientes:

- **Mínimos locales**: La función tiene cuatro mínimos locales en los siguientes puntos:
  - $(3.0,\ 2.0)$ con $f(3.0,\ 2.0) = 0$
  - $(-2.805118, 3.131312)$ con $f(-2.805118, 3.131312) = 0$
  - $(-3.779310, -3.283186)$ con $f(-3.779310, -3.283186) = 0$
  - $(3.584428, -1.848126)$ con $f(3.584428, -1.848126) = 0$
- **Máximos locales**: También tiene máximos locales y puntos de silla, lo que agrega complejidad al paisaje de la función.
- **Dificultad de optimización**: Debido a los múltiples mínimos locales y la topología de la función, no es fácil encontrar el mínimo global utilizando métodos de optimización simples. Los algoritmos como el **gradiente descendente**, **descenso de Newton** o **algoritmos evolutivos** son probados con esta función para evaluar su capacidad de convergencia.

Desarrollo

Programa para la resolución de la Función de Himmelblau

1. Crea un programa que encuentre los valores (x,y) para los que la función de Himmelblau es mínima.



Con -5<=x,y<=5

Algoritmo

La solución propuesta en esta práctica es un programa que realiza una búsqueda aleatoria para encontrar un mínimo de la función de Himmelblau dentro de un espacio de búsqueda definido, este espacio se define en el rango de [-5,5]. El algoritmo paso a paso es el siguiente:

```
Definir la función Himmelblau(x, y)
Inicializar:
    num_samples ← 1000
    min_value ← +∞
    min_x, min_y ← NULL

Iniciar cronómetro

Repetir para i = 1 hasta num_samples:
    Generar x aleatorio en [-5, 5]
    Generar y aleatorio en [-5, 5]

    value ← Himmelblau(x, y)

    Si value < min_value:
```

min_value ← value

min_x ← x

min_y ← y

Detener cronómetro

Mostrar min_x, min_y

Mostrar min_value

Mostrar tiempo de ejecución

Código

```python
import time
import numpy as np

# Definición de la función de Himmelblau
def himmelblau(x, y):
    return (x**2 + y - 11)**2 + (x + y**2 - 7)**2

# Número de muestras aleatorias que se generarán
num_samples = 1000

# Variables para almacenar el mínimo encontrado
min_x, min_y = None, None
min_value = float('inf')

# Iniciar el cronómetro
start_time = time.time()

# Generar puntos aleatorios dentro del rango [-5, 5] para x e y
for _ in range(num_samples):
    x = np.random.uniform(-5, 5)
    y = np.random.uniform(-5, 5)
    value = himmelblau(x, y)

    # Actualizar el mínimo si se encuentra uno mejor
    if value < min_value:
        min_value = value
        min_x, min_y = x, y
# Detener el cronómetro
end_time = time.time()

# Mostrar los resultados
print(f"Valores mínimos de x e y: ({min_x:.5f}, {min_y:.5f})")
print(f"Valor mínimo de la función de Himmelblau: {min_value:.5f}")
print(f"Tiempo de ejecución: {end_time - start_time:.5f} segundos")

```

Ejecución

A continuación se muestran 5 ejecuciones del programa realizado basado en la búsqueda aleatoria.

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab6_BusquedaInformadaII
I.py"
Valores mínimos de (x, y): (-2.83633, 3.06945)
Valor mínimo de la función de Himmelblau: 0.18510
Tiempo de ejecución: 0.04144 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab6_BusquedaInformadaII
I.py"
Valores mínimos de (x, y): (3.61119, -1.77236)
Valor mínimo de la función de Himmelblau: 0.13328
Tiempo de ejecución: 0.07409 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab6_BusquedaInformadaII
I.py"
Valores mínimos de (x, y): (-3.74766, -3.26127)
Valor mínimo de la función de Himmelblau: 0.05927
Tiempo de ejecución: 0.06853 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab6_BusquedaInformadaII
I.py"
Valores mínimos de (x, y): (3.57717, -1.70275)
Valor mínimo de la función de Himmelblau: 0.28274
Tiempo de ejecución: 0.02997 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab6_BusquedaInformadaII
I.py"
Valores mínimos de (x, y): (3.54439, -1.80792)
Valor mínimo de la función de Himmelblau: 0.09512
Tiempo de ejecución: 0.02883 segundos
```

Reporte de resultados

- Ejecuciones del programa con algoritmo de Recocido Simulado

A continuación se muestran 5 ejecuciones del programa de la práctica anterior que se basó en el algoritmo de Recocido Simulado.

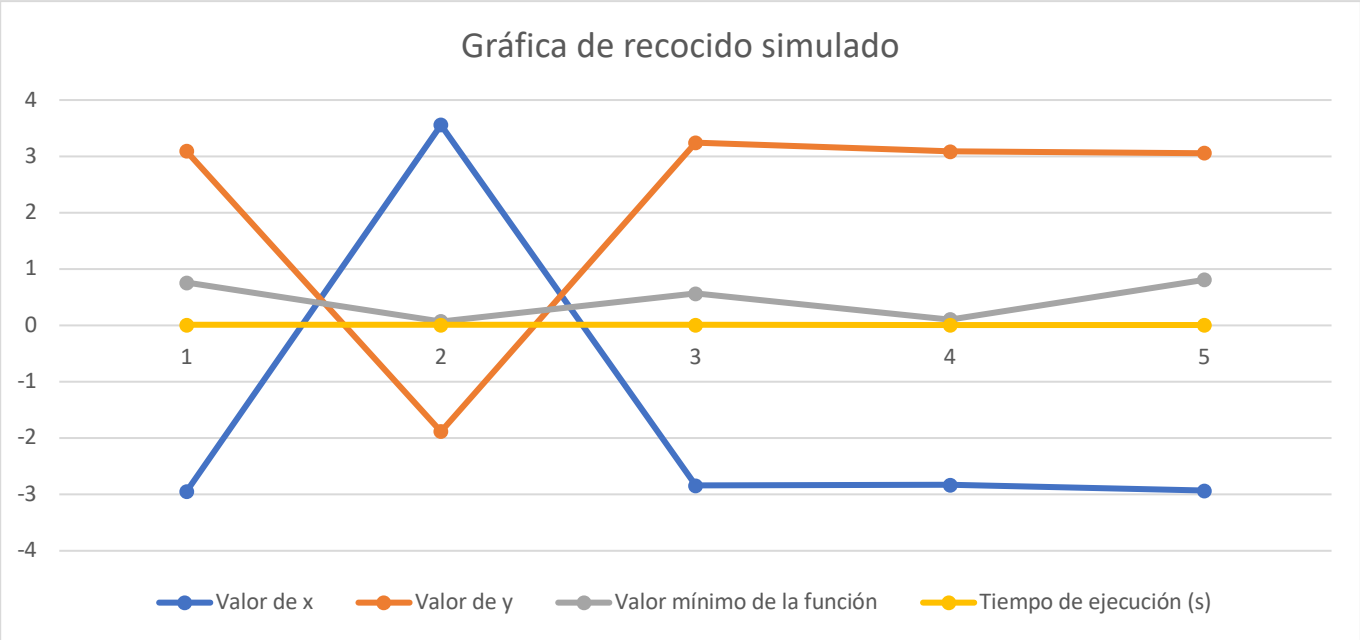
```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab5_BusquedaInformada2/
lab5.py"
Mejor solución encontrada: x = -2.947321, y = 3.090548
Valor mínimo de la función: f(x, y) = 0.760800
Tiempo de ejecución: 0.006396 segundos
```

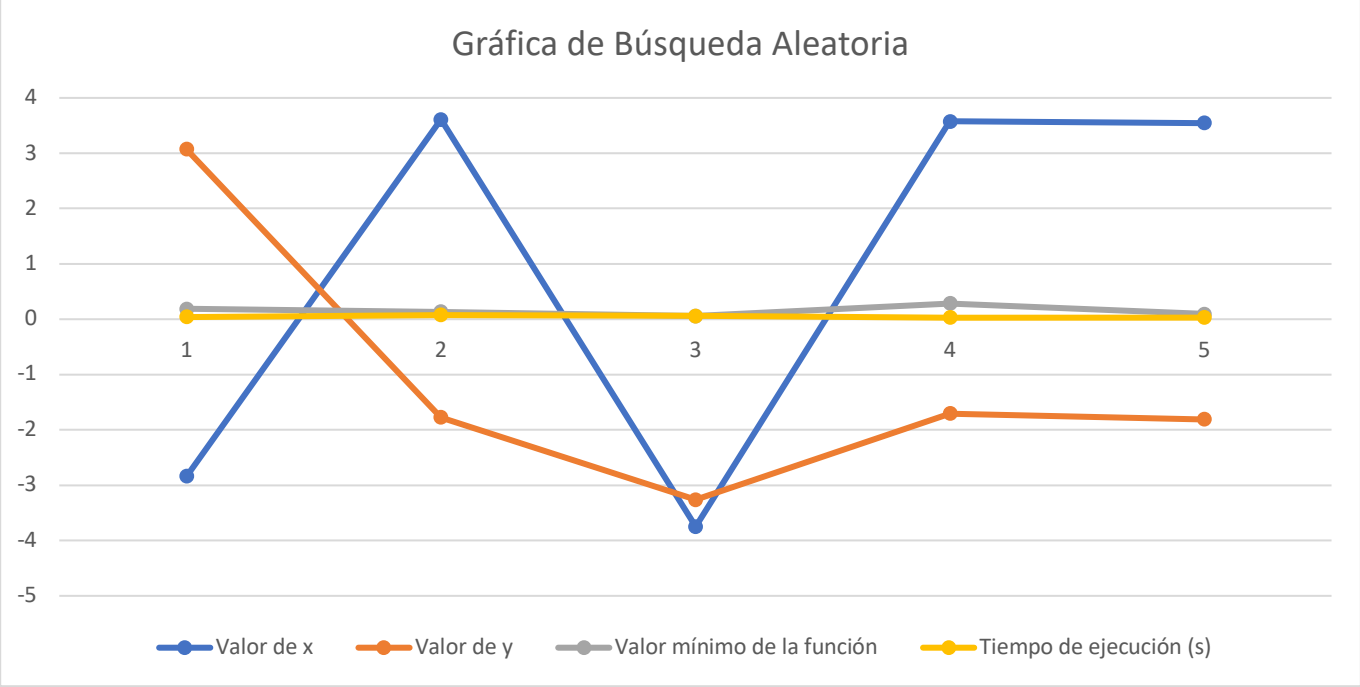
```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab5_BusquedaInformada2/
lab5.py"
Mejor solución encontrada: x = 3.555877, y = -1.884162
Valor mínimo de la función: f(x, y) = 0.068778
Tiempo de ejecución: 0.006396 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab5_BusquedaInformada2/
lab5.py"
Mejor solución encontrada: x = -2.843145, y = 3.243573
Valor mínimo de la función: f(x, y) = 0.566126
Tiempo de ejecución: 0.006374 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab5_BusquedaInformada2/
lab5.py"
Mejor solución encontrada: x = -2.828868, y = 3.085102
Valor mínimo de la función: f(x, y) = 0.104402
Tiempo de ejecución: 0.006369 segundos
```

```
yeimicarmona@syn-172-100-093-080 ~ % python -u "/Users/yeimicar
mona/Library/CloudStorage/OneDrive-InstitutoPolitecnicoNacional
/Documents/ipn/SextoSem/IA/laboratorio/Lab5_BusquedaInformada2/
lab5.py"
Mejor solución encontrada: x = -2.935494, y = 3.056064
Valor mínimo de la función: f(x, y) = 0.808362
Tiempo de ejecución: 0.007262 segundos
```





Con estos resultados obtenidos podemos observar cómo la eficiencia en la búsqueda del mínimo la logra en su mayoría el programa basado en el algoritmo de Recocido Simulado ya que los valores mínimos de la función encontrados son más cercanos a cero. En contraste, la Búsqueda Aleatoria no parece ser tan eficiente para encontrar valores mínimos cercanos al óptimo. Aunque en algunas iteraciones los resultados son aceptables, no logran alcazar valores mínimos tan bajos como el recocido simulado.

Referente al tiempo de ejecución, ambos algoritmos son rápidos (del orden de milisegundos), sin embargo, el algoritmo de Recocido Simulado tiende a tener tiempos de ejecución más constantes y ligeramente más rápidos, mientras que la Búsqueda Aleatoria tiene tiempos más variables y generalmente más lentos.

En cuanto al comportamiento de las variables x y y, en el algoritmo de Recocido Simulado, los valores de x y y muestran una variabilidad controlada a lo largo de las iteraciones, pero tienden a estaiblizarse en torno a ciertos valores óptimos. Ahora, en la Búsqueda Aleatoria, las variaciones de estas variables parecen ser más erráticas, lo que indica que este enfoque es menos eficiente y más dependiente del azar para encontrar el mínimo.

Conclusiones

La comparación entre el algoritmo de Recocido Simulado y la Búsqueda Aleatoria para encontrar el mínimo de la función de Himmelblau revela una clara superioridad del Recocido Simulado. Este algoritmo no solo encuentra soluciones más cercanas al valor óptimo, sino que también muestra una mayor consistencia en sus resultados. Los valores de las variables y de la función mínima se estabilizan más rápidamente, lo que refleja un enfoque más dirigido y eficiente.

En contraste, la Búsqueda Aleatoria presenta una mayor variabilidad y depende del azar, lo que reduce su efectividad y aumenta los tiempos de ejecución. A pesar de que ambos algoritmos son rápidos, el Recocido Simulado es más eficiente y fiable en términos de rendimiento y precisión.

Bibliografía

Bertsekas, D. P., & Tsitsiklis, J. N. (2000). *Introducción a la optimización*. Ediciones McGraw-Hill.