Facultad de Matemática y Computación. Universidad de La Habana.
Modelos de Optimización.
Función Langerman-5

Repositorio del Proyecto en GitHub

Osvaldo R. Moreno Prieto - osvaldo0202013@gmail.com
September 5, 2024

Introducción

La función de Langerman es una función de prueba comúnmente utilizada en problemas de optimización para evaluar el rendimiento de algoritmos de optimización. Su expresión es la siguiente:

$$f(\mathbf{x}) = -\sum_{i=1}^{m} c_i \exp\left(-\frac{1}{\pi}\sum_{j=1}^{D}(x_j - a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{D}(x_j - a_{ij})^2\right)$$

donde:

- **x** es un vector de D dimensiones.
- **m** es el número de términos en la sumatoria (Langerman-5, m = 5).
- **A** es una matriz de dimensión m x D (generalmente 10)
- **C** es un vector columna de dimensión m.

La función de Langerman es especialmente útil para evaluar el rendimiento de algoritmos de optimización global debido a las siguientes características:

1. **Multimodal:** Tiene múltiples mínimos locales y globales, lo que desafía a los algoritmos a encontrar el mínimo global evitando quedarse atrapados en mínimos locales.
2. **No Separable:** La función objetivo o las restricciones no pueden descomponerse en sumas de funciones más simples que dependen de subconjuntos disjuntos de variables. Esto significa que las variables están interrelacionadas de manera compleja, y no se pueden optimizar de forma independiente.
3. **Escalabilidad:** Es escalable en términos de dimensión, permitiendo probar algoritmos en espacios de búsqueda de diferentes tamaños.
4. **Combinación de Componentes Exponenciales y Trigonométricos:** Incluye términos que crean un paisaje de optimización complejo con múltiples picos y valles.
5. **Dificultad Controlada por Parámetros:** Los parámetros pueden ajustarse para modificar la dificultad del problema, adaptándose a diferentes escenarios de prueba.

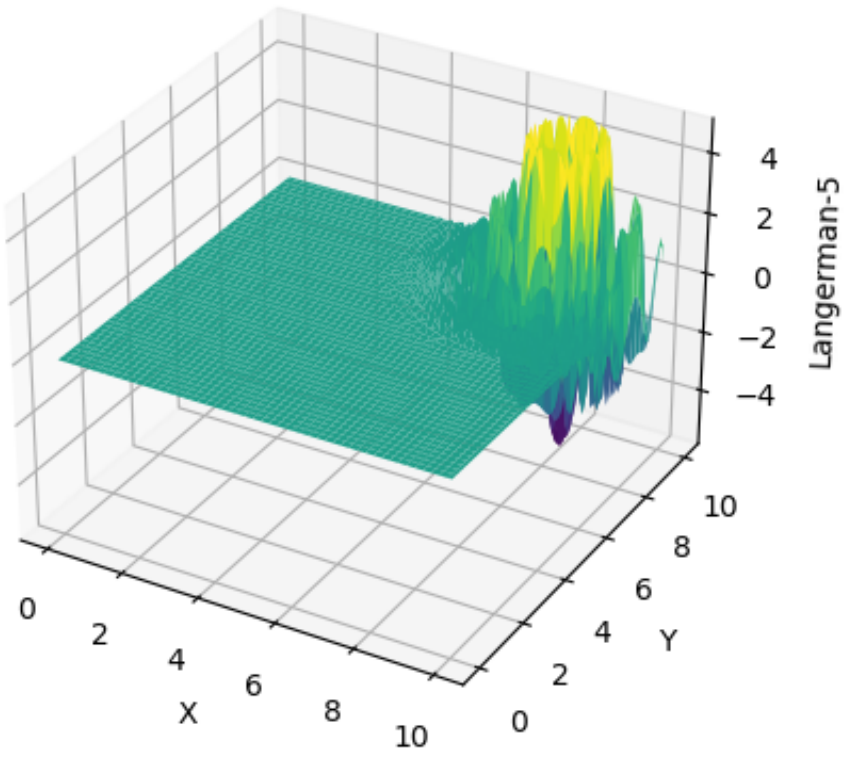

Figure 1: Ejemplo de los picos de la función de Langerman, donde $xi = 0, i \geq 3$

1 Estudio

1.1 Descripción del problema

El propósito del estudio es evaluar la eficacia de varios algoritmos en la búsqueda de soluciones, comparando su proximidad con la solución conocida, así como analizar su rendimiento en términos de cantidad de iteraciones y tiempo requerido para converger. Se conoce que la función Langerman-5 (Continua, diferenciable, no separable, escalable y multimodal) con la siguiente restricción:

$$0 \leq x_j \leq 10, \quad \text{donde } j \in [0, D-1], \quad \text{y } m = 5.$$

La función tiene un mínimo global con un valor de:

$$f(x^*) = -1.4.$$

Usando la matriz A y el vector columna c siguientes:

$$A = \begin{bmatrix} 9.681 & 0.667 & 4.783 & 9.095 & 3.517 & 9.325 & 6.544 & 0.211 & 5.122 & 2.020 \\ 9.400 & 2.041 & 3.788 & 7.931 & 2.882 & 2.672 & 3.568 & 1.284 & 7.033 & 7.374 \\ 8.025 & 9.152 & 5.114 & 7.621 & 4.564 & 4.711 & 2.996 & 6.126 & 0.734 & 4.982 \\ 2.196 & 0.415 & 5.649 & 6.979 & 9.510 & 9.166 & 6.304 & 6.054 & 9.377 & 1.426 \\ 8.074 & 8.777 & 3.467 & 1.863 & 6.708 & 6.349 & 4.534 & 0.276 & 7.633 & 1.567 \end{bmatrix}$$

$$c = \begin{bmatrix} 0.806 \\ 0.517 \\ 1.5 \\ 0.908 \\ 0.965 \end{bmatrix}$$

Para el estudio usamos los siguientes algoritmos (implementados en la biblioteca scipy):

- **Método BFGS:** El método BFGS (Broyden-Fletcher-Goldfarb-Shanno) es un algoritmo de optimización que pertenece a la clase de los métodos de Quasi-Newton. Estos métodos buscan aproximar la matriz hessiana sin necesidad de calcularla explícitamente. En lugar de ello, construyen una aproximación mediante actualizaciones basadas en las evaluaciones de la función y su gradiente. El BFGS es uno de los métodos más eficientes en esta categoría debido a su convergencia superlineal, Este método es especialmente útil en problemas sin restricciones o con restricciones simples.
- **Algoritmo de Enjambre de Partículas (PSO):** El algoritmo de enjambre de partículas es un método de optimización basado en poblaciones que se inspira en el comportamiento colectivo de los enjambres naturales, como las bandadas de aves o los bancos de peces. Cada "partícula" en el enjambre representa una posible solución al problema de optimización, y las partículas se mueven por el espacio de búsqueda influenciadas tanto por su propia mejor posición como por las mejores posiciones encontradas por sus compañeras de enjambre. Este enfoque es particularmente efectivo para problemas multimodal debido a su capacidad para explorar globalmente el espacio de búsqueda mientras explota las mejores soluciones locales. Además, su simplicidad y flexibilidad lo hacen ideal para problemas complejos y de alta dimensionalidad.
- **Evolución Diferencial con L-BFGS-B:** La Evolución Diferencial es un algoritmo de optimización basado en poblaciones que utiliza operadores de mutación, cruce y selección para explorar el espacio de soluciones de manera estocástica. Este método es particularmente efectivo para problemas con múltiples óptimos locales y funciones objetivo complejas. Al combinar la Evolución Diferencial con el algoritmo L-BFGS-B (Limited-memory BFGS with Bound constraints), se logra una optimización más eficiente en presencia de restricciones de caja (bound constraints). El L-BFGS-B es una variante del método BFGS que maneja de manera efectiva las restricciones de límite, utilizando una representación de memoria limitada para aproximar la matriz hessiana, lo que lo hace adecuado para problemas de gran escala.

1.2 Análisis de BFGS

Al utilizar el método BFGS estándar implementado en SciPy, el algoritmo no realizó ninguna iteración obteniendo como mínimo valor 0, realizó 11 evaluaciones en la función y una sola evaluación del gradiente. Este comportamiento puede deberse a que si el punto inicial está cerca de un mínimo local o si la función objetivo es plana en esa región, el gradiente calculado en este punto puede ser muy pequeño. Como BFGS depende del gradiente para actualizar la solución, si el gradiente es prácticamente cero en el primer paso, el algoritmo concluirá que ya ha encontrado una solución (dentro de la tolerancia predeterminada) y no realizará más iteraciones. Para conseguir un efecto distinto fuimos bajando el error desde la tolerancia por defecto de 1e-7 hasta 1e-12.
El resultado de este cambio, realizando 6 iteraciones y 64 evaluaciones en el gradiente de la función, reporto dos advertencias: Method BFGS cannot handle bounds, Desired error not necessarily achieved due to precision loss. Lo cual es esperado ya que el algoritmo BFGS estándar está diseñado para problemas de optimización sin restricciones o con restricciones muy simples. La función Langerman-5 tiene un dominio restringido (generalmente $0 \leq xj \leq 10, 0 \leq j \leq 10$), y como el BFGS no incorpora mecanismos para respetar estos límites, la optimización podría realizarse fuera de los rangos permitidos, y se genera el aviso de que BFGS no puede manejar restricciones de límites. Y el segundo aviso, relacionado con la pérdida de precisión, sugiere que el algoritmo no pudo alcanzar la precisión deseada durante la búsqueda del mínimo.

2 Particle Swarm Optimization (PSO)

Al ser un algoritmo basado en poblaciones, es capaz de explorar el espacio de búsqueda de manera global, lo que le permite evitar quedarse atrapado en un óptimo local. Las partículas del enjambre exploran múltiples áreas del espacio simultáneamente, y su interacción permite un equilibrio entre la exploración global y la explotación local, mejorando la probabilidad de encontrar el óptimo global. Requiere pocos parámetros para ajustar (principalmente el tamaño del enjambre y los factores de inercia y aceleración), lo que facilita su uso en una variedad de problemas sin necesidad de un ajuste fino exhaustivo. Además PSO tiende a converger rápidamente en las primeras etapas de la optimización,

lo que significa que es capaz de proporcionar soluciones de buena calidad en un tiempo relativamente corto, aunque puede requerir más iteraciones para refinar la solución final, es probable que encuentre una solución cercana al óptimo global más rápido que otros métodos en la fase inicial de la búsqueda.
La siguientes tablas muestran los resultados obtenidos al correr PSO con 1e-12 de tolerancia, y con un enjambre de tamaño 100. Los datos han sido redondeados 4 cifras significativas por temas de visualidad en la tabla.

Solución (x)	**Valor obtenido** (y)	**Diferencia con -1.4**
$[9.5856, 1.9113, 3.8136, 7.9867, 3.7313, 2.3887, 3.7246, 1.6332, 6.0614, 7.5179]$	-0.2749	1.1251
$[8.8641, 9.5517, 5.5733, 7.6481, 3.9883, 4.5225, 2.9350, 5.7984, 0.6758, 5.6231]$	-0.7977	0.6023
$[7.8651, 8.2509, 3.9229, 1.3407, 6.8043, 6.4320, 5.5810, 0.1968, 7.7290, 1.3695]$	-0.5132	0.8868
Valor esperado	-1.4	0

Table 1: Comparación de los resultados obtenidos mediante PSO con el valor esperado -1.4

Valor obtenido (y)	**Iteraciones**	**Tiempo en segundos**
-0.2749	101	1.10
-0.7977	74	0.82
-0.5132	80	0.85

Table 2: Comparación de los resultados obtenidos, según la cantidad de iteraciones y el tiempo

3 Análisis de Diferenciación Evolutiva con L-BFGS-B (DE L-BFGS-B)

La Evolución Diferencial es un algoritmo de optimización basado en poblaciones, lo que significa que explora el espacio de búsqueda de manera global. Al igual que PSO, este algoritmo es sensibles a los parámetros de entrada.
En un primer intento usamos la estrategia *rand1bin* que elige individuos al azar para la mutación, proporcionando más exploración. Usamos de tolerancia 1e-5, y tamaño de población 30.

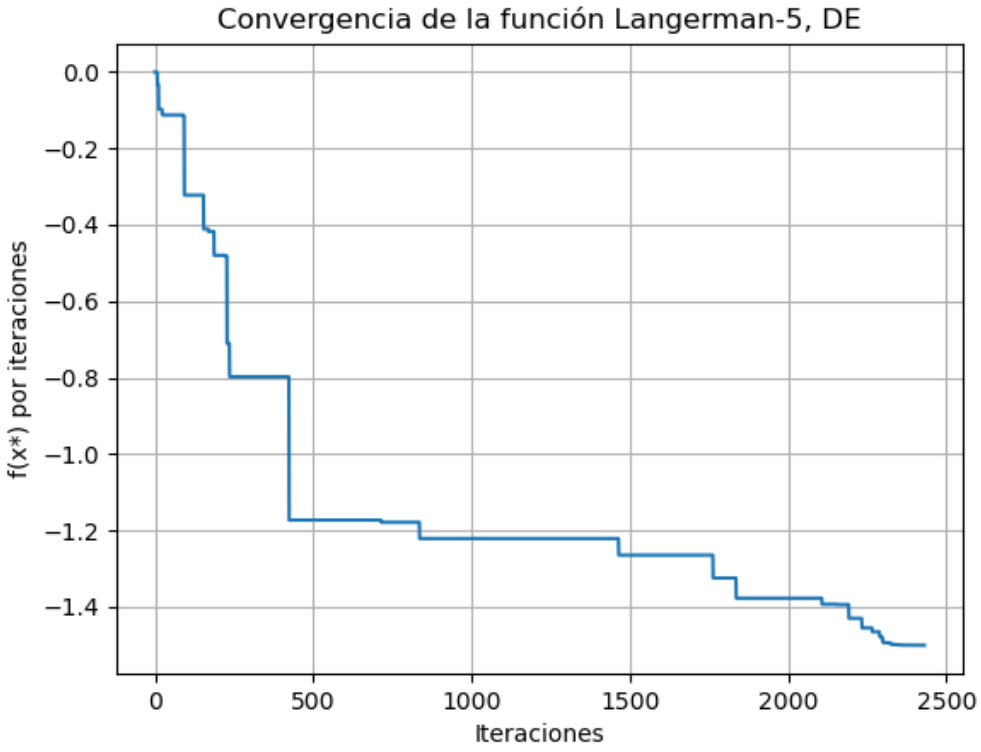

Figure 2: Soluciones encontradas por iteraciones

Con esta configuración obtuvimos, tras 2431 iteraciones y 80.57s :
$x* = [8.02500001, 9.15199997, 5.113999987.62100002, 4.56399994, 4.71099993, 2.99599997, 6.12599992, 0.73400003, 4.98200003]$
$f(x*) = -1.49999$
Como se puede observar en la figura 2, el algoritmo pasa por varios óptimos locales en los que se detiene varias iteraciones, incluyendo los anteriormente encontrados con PSO.

En un el segundo intento usamos la estrategia *best*1*bin* selecciona el mejor individuo de la población y aplica mutación a él. Usamos de tolerancia 1e-5, y tamaño de población 30, igual al anterior. Con esta configuración obtuvimos, tras 313 iteraciones y 13.12s :
$x* = [8.025, 9.15199995, 5.11399999, 7.62099994, 4.56400003, 4.71099997, 2.996, 6.12600006, 0.73400002, 4.98199999]$
$f(x*) = -1.49999$

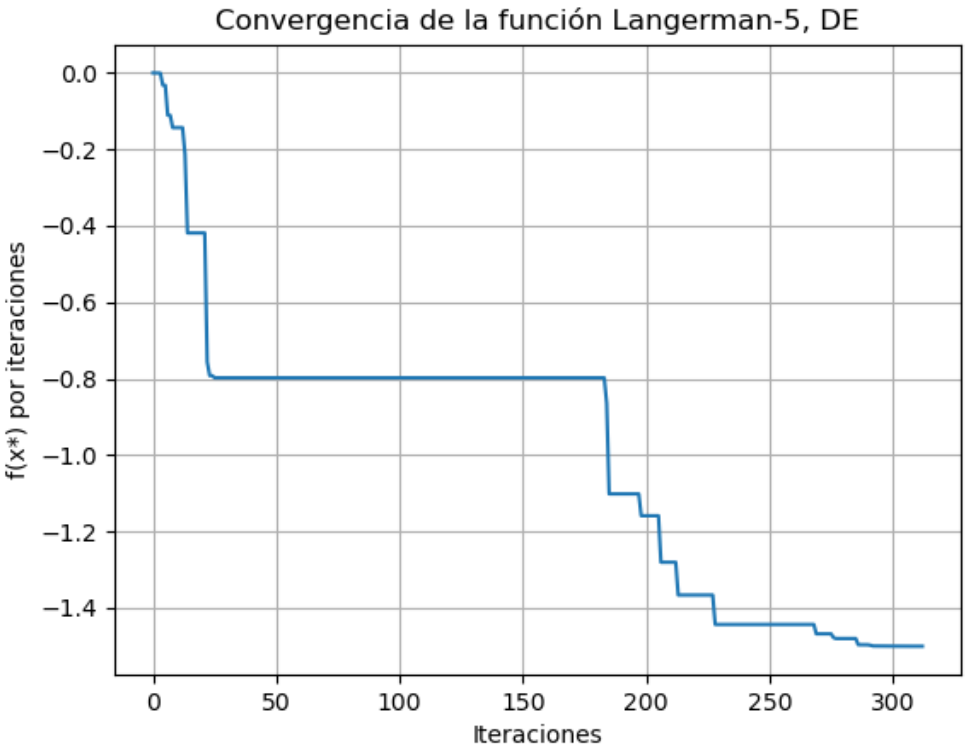

Figure 3: Soluciones encontradas por iteraciones

Comparando ambos resultados, con los parámetros escogidos *best*1*bin* obtuvo un mejor rendimiento que *rand*1*bin* por cuestiones de tiempo. Y ambos con una evaluación de $f(x*) = -1.4999$ que es la mas cercana a la conocida en el problema(-1.4)

4 Resumen

A pesar de que los métodos de Máximo Descenso, Newton y Quasi Newton, son muy útiles y son buenos encontrando las soluciones, no son los adecuados para resolverlos problemas de optimización si la función es multimodal debido a que pueden quedarse atrapados en óptimos locales debido a la cantidad de picos y puntos de ensilladura que pueden existir. Por eso es mejor usar enfoques híbridos para su resolución.
El método analizado para hallar la solución de Langerman-5 fue DE-L-BFGS-B, a continuación comparamos los mejores resultados de PSO y DE-L-BFGS-B

Solución (x)	**Valor obtenido** (y)	**Diferencia con -1.4**
$[8.8641, 9.5517, 5.5733, 7.6481, 3.9883, 4.5225, 2.9350, 5.7984, 0.6758, 5.6231]$	-0.7977	0.6023
$[8.025, 9.1520, 5.1140, 7.6210, 4.5640, 4.7110, 2.9960, 6.1260, 0.73400, 4.9819]$	-1.4999	-0.0999
Valor esperado	-1.4	0

Table 3: Comparación de los resultados obtenidos mediante PSO y DE-L-BFGS-B

Valor obtenido (y)	**Iteraciones**	**Tiempo en segundos**
-0.7977	74	0.82
-1.4999	313	13.12

Table 4: Comparación de los resultados obtenidos, según la cantidad de iteraciones y el tiempo