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 Particle Swarm Optimization (PSO) es un método de optimización inspirado en el comportamiento colectivo de enjambres naturales, como bandadas de aves o bancos de peces. Cada partícula representa una posible solución y se mueve en el espacio de búsqueda influenciada tanto por su propia experiencia como por la mejor solución encontrada por el grupo. En este reporte se analiza la implementación del PSO en su versión global (GBEST), así como su desempeño en problemas de minimización y maximización.

2. **Descripción del código**

El código implementa un algoritmo PSO global para observar el comportamiento del enjambre a lo largo de las iteraciones.

Variables y parámetros iniciales

Variables y parámetros iniciales

dimension: número de variables del problema.

num_particulas: cuántas partículas forman el enjambre.

minimizar: determina si se busca minimizar o maximizar la función.

c1, c2: coeficientes cognitivo y social.

Tmax: número máximo de iteraciones.

omega: inercia inicial de las partículas.

omega_min: valor mínimo permitido para la inercia cuando es variable.

omega_fijo: si es True, la inercia se mantiene constante; si es False, se actualiza en cada iteración.

3. **Algoritmo Principal**

Inicialización

Las posiciones y velocidades de las partículas se generan aleatoriamente en un rango definido. Se evalúa cada solución y se establece la mejor posición personal y global.

Actualización de velocidades

La velocidad se ajusta mediante:

$vi=\omega vi+c1r1(pbesti-xi)+c2r2(gbest-xi)$

donde:

- ω: inercia
- c1,c2c_1, c_2c1,c2: pesos cognitivo y social

- r1,r2r_1, r_2r1,r2: números aleatorios en [0,1]

Actualización de posiciones

$xi = xi + vi$

Selección del mejor individuo

En cada iteración se actualiza el gbest si se encuentra una mejor solución.

Historial

Se guarda el avance del mejor valor encontrado y se grafica de manera dinámica.

4. Resultados Obtenidos

Ejemplos en funciones de dos dimensiones

1. Esfera

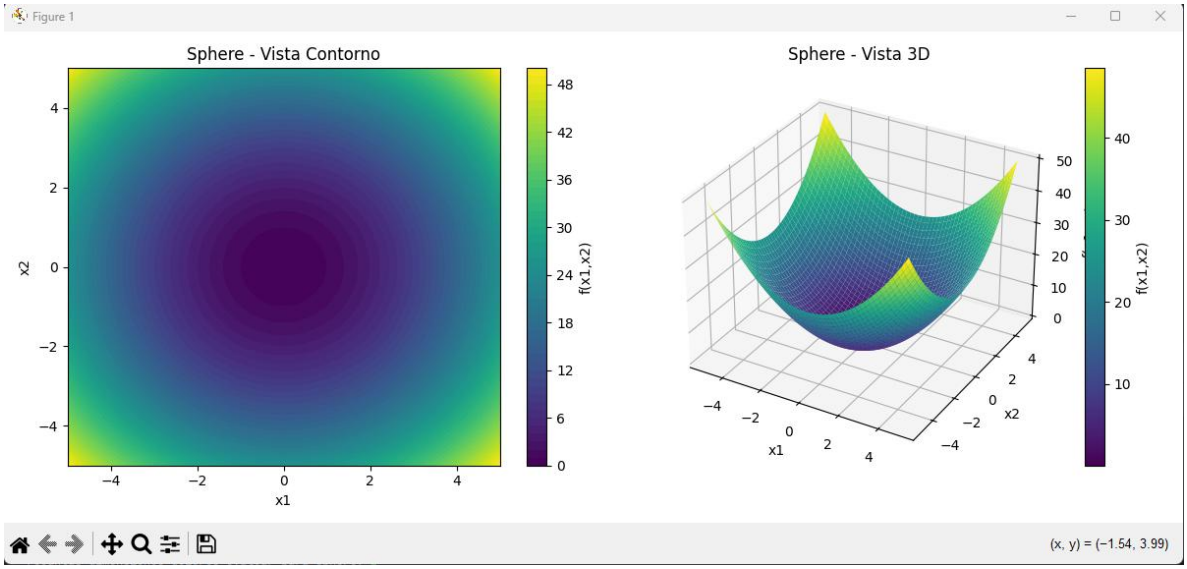

Imagen 1. Grafica de Esfera

Minimización

Tiempo 19/20 - Mejor valor: 0.000296
Tiempo 20/20 - Mejor valor: 0.000296

Mejor solución encontrada: [0.01518203 -0.00808693]
Mejor valor de la función: 0.0002958923556326875

Imagen 2. Descendente Esfera

Maximización

Tiempo 19/20 - Mejor valor: 4856.992177
Tiempo 20/20 - Mejor valor: 5441.546331

Mejor solución encontrada: [-55.75937203 48.29532857]
Mejor valor de la función: 5441.546330772026

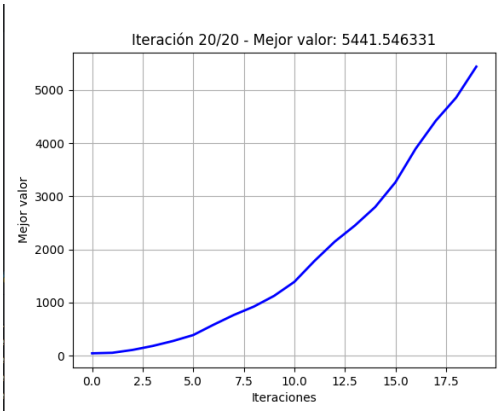

Imagen 3. Ascendente Esfera

2. Quadric

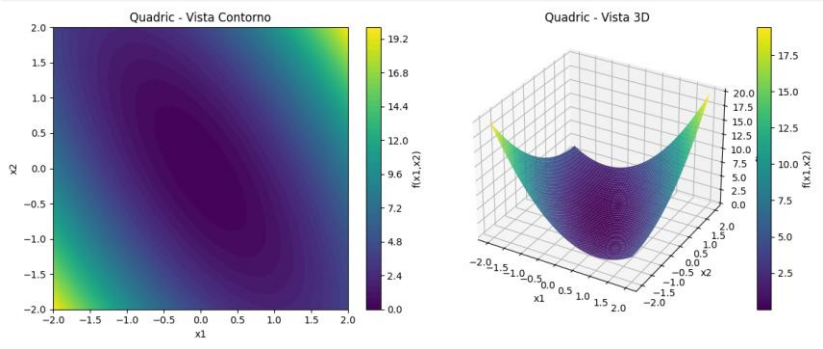

Imagen 4. Grafica de Quadric

Minimización

Tiempo 19/20 - Mejor valor: 0.000083

Tiempo 20/20 - Mejor valor: 0.000083

Mejor solución encontrada: [-0.00870169 0.01137123]
Mejor valor de la función: 8.28458411857738e-05

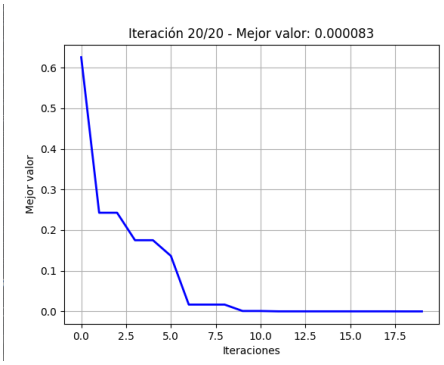

Imagen 5. Descendente Quadric

Maximización

Tiempo 19/20 - Mejor valor: 8967.585716
Tiempo 20/20 - Mejor valor: 10010.560202

Mejor solución encontrada: [-43.30699012 -46.887604]
Mejor valor de la función: 10010.560201950932

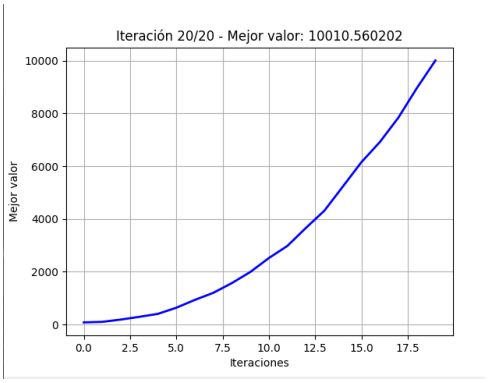

Imagen 6. Ascendente Quadric

3. Ackley

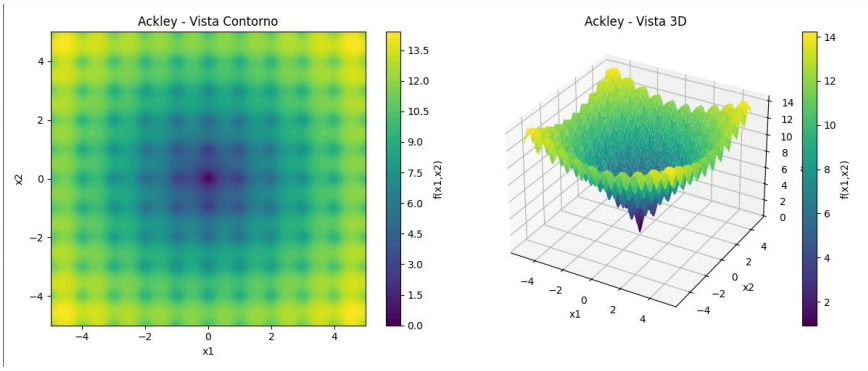

Imagen 7. Grafica de Ackley

Minimización

Tiempo 19/20 - Mejor valor: 0.038589
Tiempo 20/20 - Mejor valor: 0.038589

Mejor solución encontrada: [0.01012861 -0.00686373]
Mejor valor de la función: 0.038588501201701764

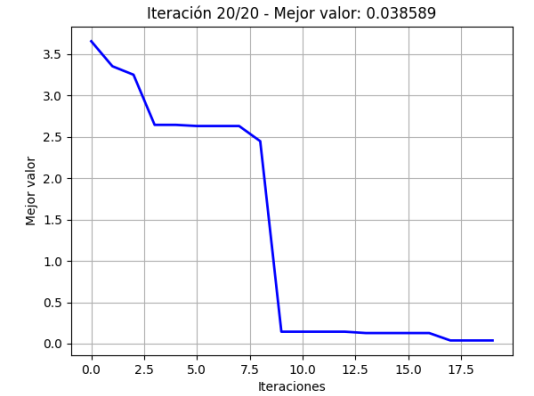

Imagen 8. Descendente Ackley

Maximización

Tiempo 19/20 - Mejor valor: 22.294985
Tiempo 20/20 - Mejor valor: 22.294985

Mejor solución encontrada: [-35.57922532 28.51378758]
Mejor valor de la función: 22.29498520473424

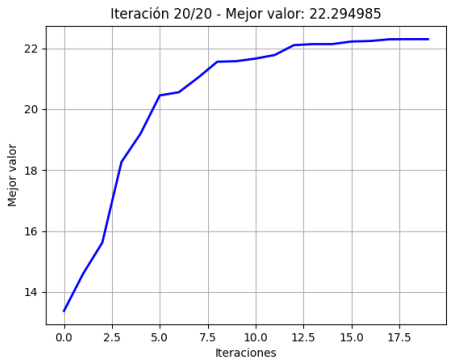

Imagen 9. Ascendente Ackley

Ejemplos en funciones de n dimensiones (usaremos 4 dimensiones para los ejemplos)

1. **Schwefel**

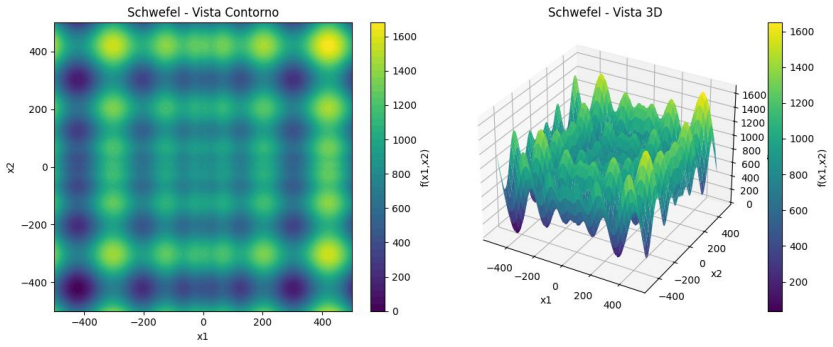

Imagen 10. Grafica de Schwefel

Minimización

Tiempo 19/20 - Mejor valor: 830.075204
Tiempo 20/20 - Mejor valor: 830.075204

Mejor solución encontrada: [-5.23344916 -5.24014083]
Mejor valor de la función: 830.0752036050874

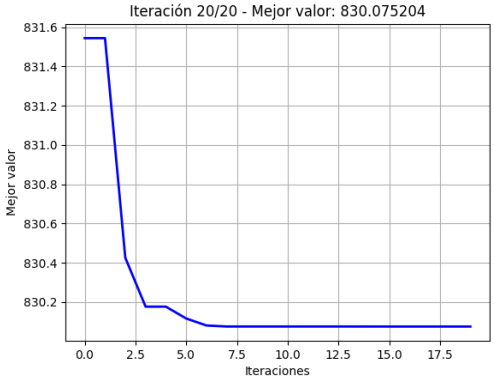

Imagen 11. Descendente de Schwefel

Maximización

Tiempo 19/20 - Mejor valor: 845.856283
Tiempo 20/20 - Mejor valor: 845.856283

Mejor solución encontrada: [5.2498274 5.21723574]
Mejor valor de la función: 845.856283028198

Imagen 12. Ascendente de Schwefel

2. Rosenbrock

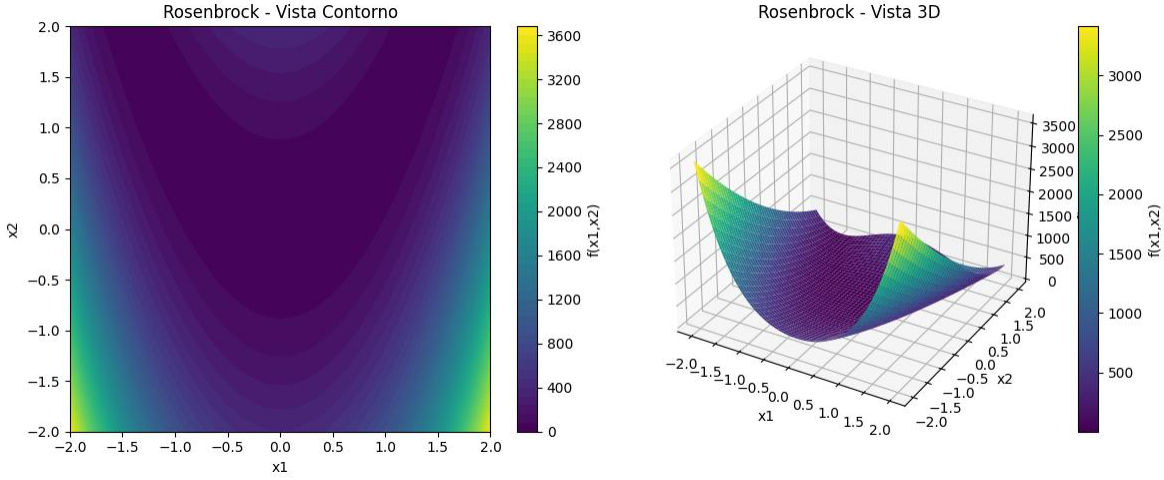

Imagen 13. Grafica de Rosenbrock

Minimización

Tiempo 19/20 - Mejor valor: 0.171753
Tiempo 20/20 - Mejor valor: 0.164906

Mejor solución encontrada: [0.61854297 0.39652245]
Mejor valor de la función: 0.16490572058938835

Imagen 14. Descendente de Rosenbrock

Maximización

Tiempo 19/20 - Mejor valor: 788595875.265427
Tiempo 20/20 - Mejor valor: 941536985.073294

Mejor solución encontrada: [55.09566767 -32.91010099]
Mejor valor de la función: 941536985.0732936

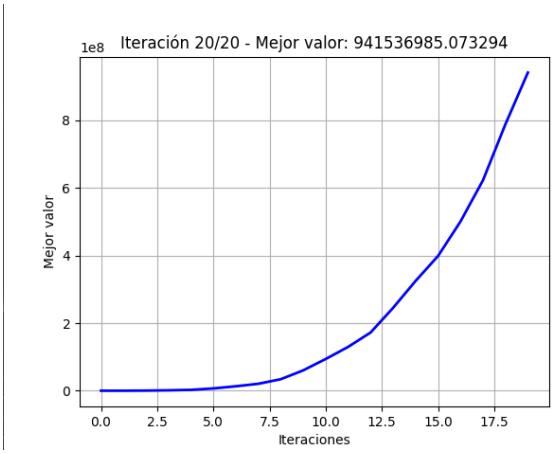

Imagen 15. Ascendente de Rosenbrock

3. Rosenbrock

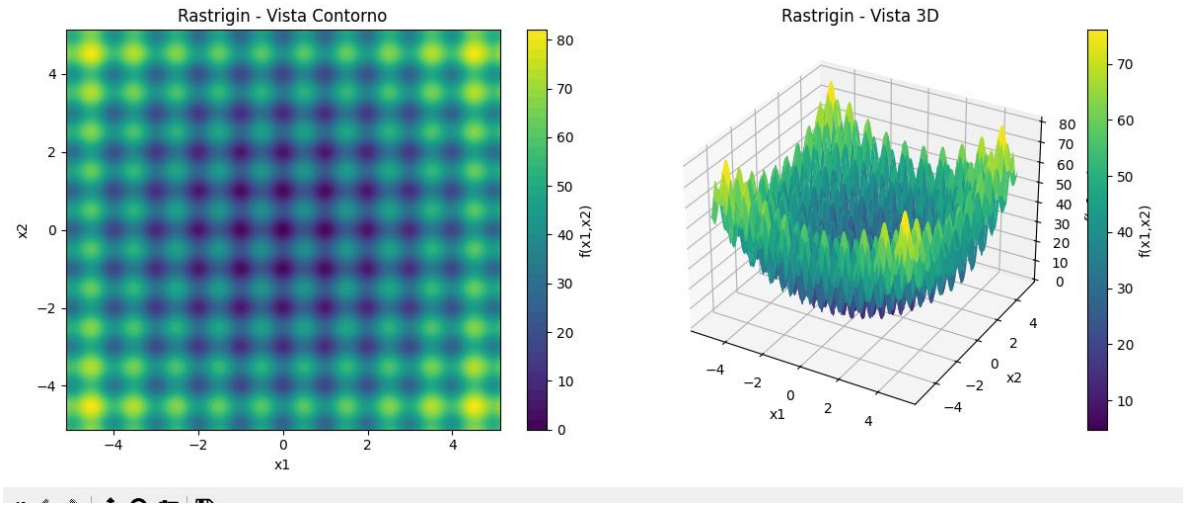

Imagen 13. Grafica de Rastringin

Minimización

Tiempo 19/20 - Mejor valor: 9.211611
Tiempo 20/20 - Mejor valor: 9.211611

Mejor solución encontrada: [-1.99596688 -0.92733259 -0.9472218 0.92679834]
Mejor valor de la función: 9.211611230064547

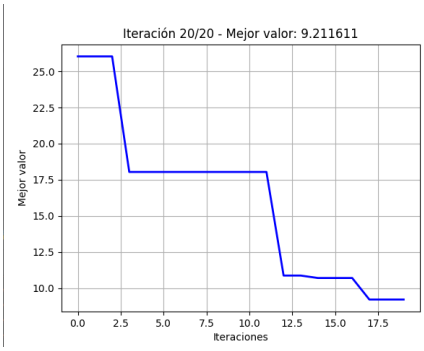

Imagen 14. Descendente de Rastringin

Maximización
Tiempo 19/20 - Mejor valor: 6110.326011
Tiempo 20/20 - Mejor valor: 7068.993068

Mejor solución encontrada: [-56.41907762 -17.68130746 -56.54875714 -17.96744082]
Mejor valor de la función: 7068.9930677441525

Imagen 15. Ascendente de Rastringin

5. Ventajas del Método

Muy fácil de implementar y adaptar.

No requiere derivadas, ideal para funciones no diferenciables o ruidosas.

Permite ajustar su comportamiento mediante pocos parámetros.

6. Limitaciones

Depende fuertemente de la inicialización y los parámetros.

Puede perder diversidad rápidamente.

En problemas de alta dimensión requiere muchas partículas para buen desempeño.

7. Conclusiones

El PSO en versión GBEST es un algoritmo muy intuitivo y eficiente para la optimización continua. Me parece que la implementación es muy intuitiva, pude rehusar la estructura de códigos anteriores, creo que con este se podría implementar animaciones muy interesantes con las partículas desplegándolas a lo largo de la función, y siento que este es uno de los algoritmos que mas terreno cubre con las partículas, creo que en pruebas de escritorio puede llegar a ser un poco confuso, pero me parece un algoritmo que cumple con el propósito de la optimización. Además, es de esos algoritmos que una vez que lo entiendes te dan ganas de seguirlo afinando: probar lbest, agregar topologías diferentes, ajustar el omega dinámicamente, etc. En general, me dejó la impresión de que es un método muy flexible y con mucho potencial para seguir explorándolo.