**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 presente estudio aplica el algoritmo Iterated Local Search (ILS) a una función objetivo cuadrática. Este método metaheurístico combina búsquedas locales intensivas con perturbaciones estratégicas para escapar de óptimos locales, manteniendo un balance entre exploración y explotación del espacio de búsqueda.

## 2. Descripción del código

El código implementa el algoritmo de Iterated Local Search para encontrar el óptimo de una función unidimensional dentro de un intervalo dado. El funcionamiento consiste en:

- Búsquedas locales intensivas usando Hill Climbing
- Perturbaciones fuertes para escapar de óptimos locales
- Criterio de aceptación para decidir entre soluciones
- Ciclos iterativos que combinan exploración y explotación

## 3. Componentes del Código

**Variables y parámetros iniciales**

```
rango = [-10, 10]      # Intervalo de búsqueda
max_iter_ILS = 30      # Iteraciones globales ILS
max_iter_local = 100   # Iteraciones por búsqueda local
fuerza_perturbacion = 2.0 # Tamaño de perturbación
umbral_aceptacion = 0.1  # Umbral para aceptar soluciones peores
```

**Función Objetivo**

```
def funcion(x):
  return x**2
```

## 4. Algoritmo Principal

Consiste en varias partes

**Búsqueda Local(Hill Climbing)**

```
x_actual, f_actual = busqueda_local(x_inicial, funcion, rango)
```

**Genereración de Perturbación**

```
x_perturbado = perturbacion(x_actual, rango, fuerza=2.0)
```

**Criterio de Aceptación**

```
if f_nuevo < f_actual:
   aceptar = True
elif f_nuevo < f_actual + umbral:
   aceptar = random() < 0.3
```

**Ciclo principal ILS**

```
for iter_ILS in range(max_iter_ILS):
   # 1. Perturbación
   # 2. Búsqueda local desde punto perturbado
   # 3. Criterio de aceptación
   # 4. Actualización del mejor global
```

## 5. Resultados Obtenidos

El código genera una tabla detallada que muestra en cada iteración

**Primero vamos con el descendente**

```
27 |   -10.0000 | -159.0000 | -159.0000 |      1
28 |   -10.0000 | -159.0000 | -159.0000 |      1
29 |   -10.0000 | -159.0000 | -159.0000 |      0
```

Mínimo encontrado: x = -10.000000, f(x) = -159.000000

En esta tabla podemos ver el número de ciclos y las variables que se fueron utilizando durante el código, lo que nos ayuda a ver la evolución del código hacia encontrar el mínimo. Y finalmente vemos el valor final al que llega y su evaluación en la función.

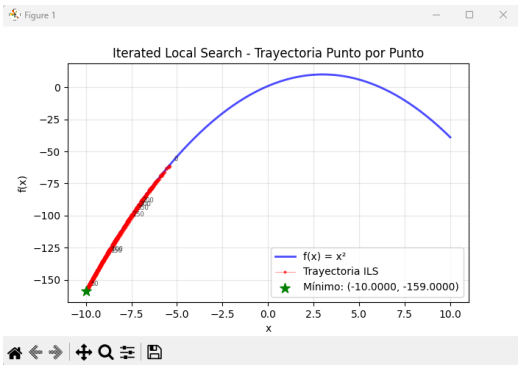


**Imagen 1. Descendente**

**Ahora vamos con el ascendente**
Para lograr este cometido, solo debemos cambiar un signo

if f_vecino > f_actual:
if f_nuevo > f_actual:
elif f_nuevo > f_actual + umbral:
if f_actual > mejor_f:

| | | | | |
|---|---|---|---|---|
| 27 | 3.0000 | 10.0000 | 10.0000 | 0 |
| 28 | 3.0000 | 10.0000 | 10.0000 | 0 |
| 29 | 3.0000 | 10.0000 | 10.0000 | 0 |

Mínimo encontrado: x = 2.999995, f(x) = 10.000000

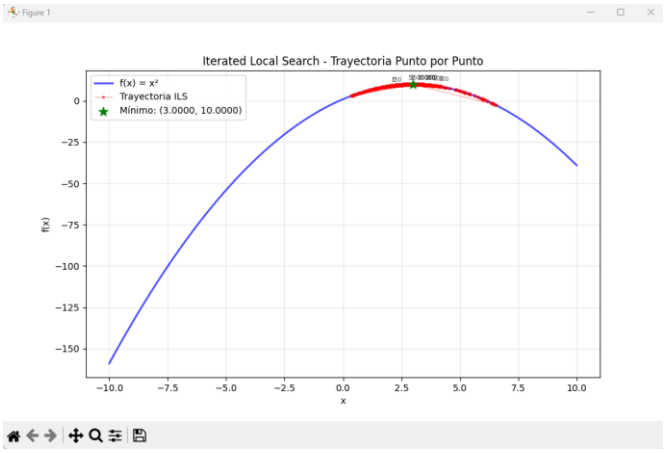


**Imagen 2. Ascendente**

## 6. Ventajas del Método

Escape efectivo de óptimos locales mediante perturbaciones

Búsquedas locales intensivas en regiones prometedoras

Balance automático entre exploración y explotación

Aplicable a diversos tipos de funciones

## 7. Limitaciones

Parámetros sensibles (fuerza de perturbación, umbral)

Convergencia no garantizada al óptimo global

## 8. Conclusiones

El código implementa correctamente el algoritmo de Iterated Local Search y demuestra su efectividad para optimización tanto de minimización como maximización. Me recuerda un poco al algoritmo anterior, en el sentido de que tiene una perturbación, tiene un Hill Climbing incluido, se generan vecinos y hay criterio de aceptación, hay diferencias,  pero pude reutilizar mucho del código anterior, lo que ayudó bastante, este algoritmo me parece mejor ya que puede realizar búsquedas intensivas en regiones prometedoras, escapa de óptimos locales y se siente como una evolución natural de los conceptos del algoritmo anterior.

## 9. Código

https://github.com/santana-robledo/Algorithm-Lab/blob/main/16.%20Iterated%20Local%20Search.py

Aquí se encuentra el repositorio