# Tarea 5

Ramírez David R.

**Department:** Computación
**Course:** Pensamiento Computacional 108
**Instructor:** Benito Granados-Rojas, PhD.
**Date:** November 10, 2022

**Abstract**

En este trabajo se demuestra la utilidad del algoritmo de Estrategia Evolutiva (EE) $1 + 1$ partiendo del código desarrollado en clase y adaptándolo para encontrar el máximo de una función de $3$ variables. También, se demuestra que los resultados del algoritmo son correctos al solucionar el problema planteado de manera analítica.

# 1 Introduction

Los algoritmos de Estrategia Evolutiva son un tipo de algoritmos evolutivos, donde, a partir de una población inicial se generan nuevos individuos al combinar individuos previos y mutarlos, después de cada generación seleccionar el mejor. En particular EE $1+1$ se refiere a la variación del algoritmo de EE en el que la población inicial es 1 y esta solo genera un descendiente a partir de una mutación del anterior, posteriormente se elige al mejor de los dos y se repite el proceso (Wikipedia 2022).

# 2 Procedures and Results

Primeramente, la ecuación a procesar es descrita en (1). El objetivo consiste en encontrar el punto máximo en la región delimitada por $-2 \leq x_i \leq 2 | i \in [1,2,3]$. Antes de emplear el algoritmo EE $1+1$, se resolverá el problema de manera análitca. Posteriormente se modificara el algoritmo EE $1+1$ desarrollado en clase para aplicarlo en la función y comparar resultados.

$$f(x_1, x_2, x_3) = x_1 + 2x_2 + x_2x_3 - x_1^2 - x_2^2 - x_3^2 \tag{1}$$

## 2.1 Experiment/simulation/measurement 1

Se sabe que un punto crítico de una función de $n$ variables es cuando todas sus derivadas parciales son igual a cero como se muestra en (2).

$$\frac{\partial f}{\partial x_i} = 0 \; \forall \; i \in \mathbb{N} \; : \; i \leq n \tag{2}$$

Así pues, resolviendo el sistema de ecuaciones, se obtendrán todos los puntos críticos de la función. Para el caso de (1) se tiene que las primeras derivadas parciales son:

$$\frac{\partial f}{\partial x_1} = 1 - 2x_1 = 0$$

$$\frac{\partial f}{\partial x_2} = 2 + x_3 - 2x_2 = 0$$
$$\frac{\partial f}{\partial x_3} = x_2 - 2x_3 = 0$$

Resolviendo el sistema de ecuaciones se tiene que:

$$x_1 = \frac{1}{2};\ x_2 = \frac{2}{3};\ x_3 = \frac{4}{3}$$

Solo se ha conseguido un punto crítico y se encuentra en la región delimitada previamente. Por lo que solo queda revisar que sea un punto máximo. Para ello se empleó el criterio de la segunda derivada con la matriz hessiana $H_f$, si todos sus eigenvalores son negativos, el punto crítico es punto máximo. Siendo la matriz hessiana una matriz cuadrada cuyos elementos están definidos por:

$$(H_f)_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

De esta forma, la matriz hessiana de la función (1) es:

$$H_f = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

Posteriormente, los eigenvalores son:

$$\lambda_{H_f} = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}$$

Finalmente, dado que todos los eigenvalores de $H_f$ son negativos, se puede afirmar que el punto máximo de la función (1) está en las coordenadas:

$$x_1 = \frac{1}{2};\ x_2 = \frac{2}{3};\ x_3 = \frac{4}{3}$$

### 2.2 Experiment/simulation/measurement 2

Para emplear el algoritmo desarrollado en clase tuvo que ser adaptado. Primeramente, se establecen los parámetros iniciales: número máximo de generaciones,

límite inferior de las variables (en este caso es el mismo para todos), valores iniciales y aleatorios de las 3 variables. También, con los valores inicales, se establece el primer individuos (padre) evaluando las variables en la función (1). Finalmente, se establecen los vectores $u$, $v$, $w$ que contendrán los diferentes valores por los que pasarán las variables hasta alcanzar el número de generaciones. Esto se muestra en Fig. 1.

```
def main():
    xmin = -2
    gmax = 100000
    m = 0
    x1 = 4 * random() + xmin
    x2 = 4 * random() + xmin
    x3 = 4 * random() + xmin
    xz = [round(x1, 6), round(x2, 6), round(x3, 6)]
    print("x1_0,x2_0,x3_0: ", xz)
    padre = f(x1, x2, x3)
    s = 1
    u = [x1]
    v = [x2]
    w = [x3]
```

Figure 1: Parámetros iniciales

.

Posteriormente, en el segmento de código mostrado en Fig. 2 se ejecutan las mutaciones a cada variable del padre actual y se evaluan en la función (1). Si estas nuevas mutaciones prueban ser mayor que el padre, la mutación pasa a ser el padre y se aumenta el número de mutaciones exitosas. En cada iteración, se actualiza el valor s de $\sigma$ para ser usado en la siguiente iteración, también se guarda el valor actual de las variables en los respectivos vectores.

En este caso, no es posible graficar la función, pero si es posible graficar las variables. En Fig. 3 se muestra las funciones para visualizar el cambio en las variables. La función $evol$, gráfica cada variable de manera independiente una de otra siendo el número de iteración la variable independiente global. Por su parte, la función $path3d$, gráfica una curva paramétrica en 3 dimensiones, las dimensiones de las variables.

```python
for g in range(1, gmax):
    x1_n = mutation(x1, s)
    x2_n = mutation(x2, s)
    x3_n = mutation(x3, s)
    hijo = f(x1_n, x2_n, x3_n)
    if hijo > padre:
        x1 = x1_n
        x2 = x2_n
        x3 = x3_n
        m += 1  # mutación exitosa
        padre = f(x1, x2, x3)
    else:
        x1 = x1
        x2 = x2
        x3 = x3
        m = m
    s = sigma(s, g, m)
    u.append(x1)
    v.append(x2)
    w.append(x3)
```

Figure 2: Ciclo de mutaciones

.

```python
def evol(u, v, w):
    plt.figure(3)
    plt.plot(u)
    plt.plot(v)
    plt.plot(w)
    plt.legend(('x1', 'x2', 'x3'))
    plt.ylim([-2, 2])
    plt.show()

def path3d(u, v, w):
    plt.figure(1)
    ax = plt.axes(projection='3d')  # ax definition
    ax.plot3D(u, v, w, 'red')
    # labeling
    ax.set_xlabel("x1")
    ax.set_ylabel("x2")
    ax.set_zlabel("x3")
    plt.title('Función Objetivo')
```

Figure 3: Graficación de cambio en variables

.

Al ejecutar el algoritmo, se generaron las variables inciales:

$$x_{1_0} = -1.867;\ x_{2_0} = -0.1471;\ x_{3_0} = -0.4927$$

Los resultados obtenidos después de $100,000$ iteraciones fue:

$$x_{1_0} = 0.5;\ x_{2_0} = 1.333333;\ x_{3_0} = 0.666666$$

Como se puede apreciar, se los resultados son exactamente igual al resultado analítico. Finalmente, se en Fig. 4 se muestran los cambios en las variables con cada iteración, mientras que en Fig. 5 se muestran las variables como coordenadas de una curva paramétrica.

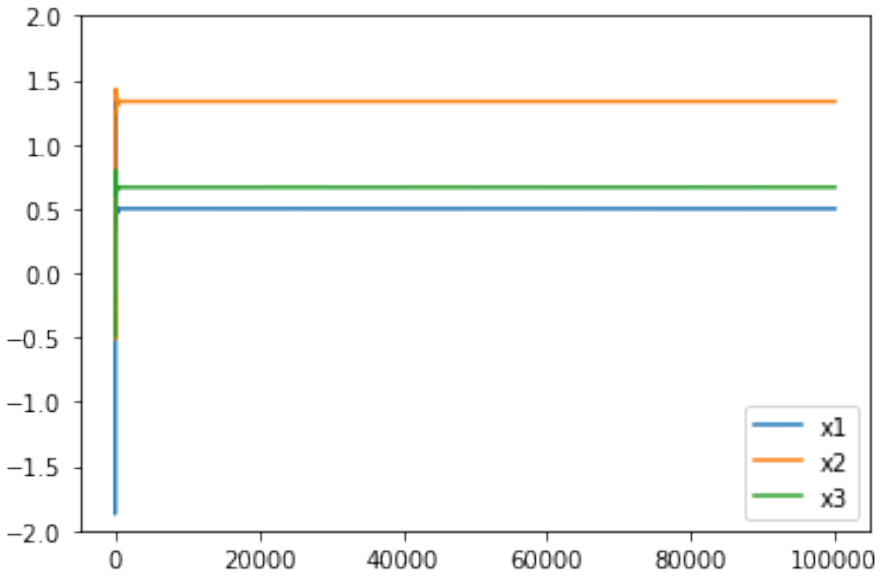


Figure 4: Grafica de cambio en variables con respecto de cada iteración

.

## 3 Conclusions

A través de este trabajo se puede apreciar la utilidad de los algoritmos de evolutivos, pues es relativamente sencillo aplicar el algoritmo para una sola función sin importar el número de variables obteniendo resultados muy cercanos al resultado analítico. Por otro lado, aunque se utilizaron $100,000$ iteraciones, a partir de Fig. 4 es evidente que no es necesario una cantidad tan grande iteraciones.

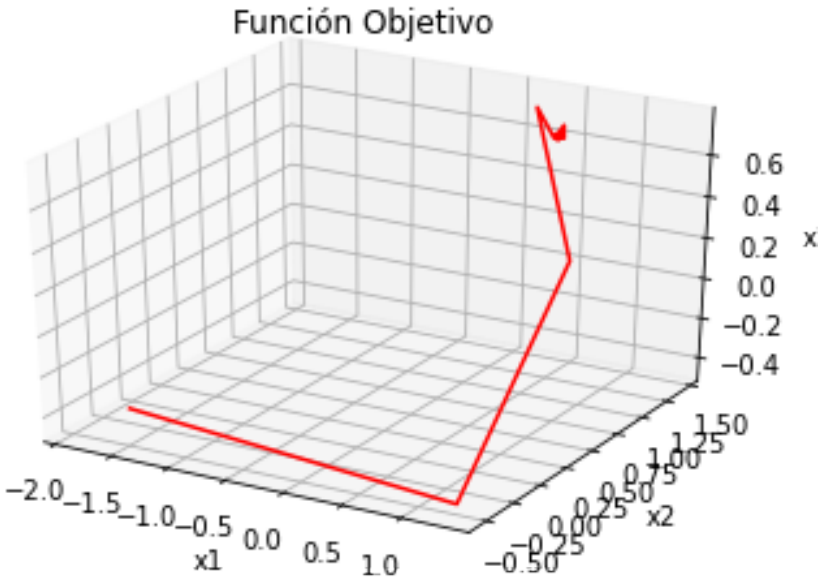


Figure 5: Variables como coordenadas de una curva paramétrica

.

## 4 Repository

`https://github.com/A01338802/ComputacionAplicada_T5.git`

## References

[1] Wikipedia. *Estrategia Evolutiva*. 2022. URL: `https://es.wikipedia.org/wiki/Estrategia_evolutiva`.