

## Tarea - 3 Evolución Diferencial

Alumno: Ángel Alonso Galarza Chávez  
 Profesor: Dr. Cuauhtemoc Mancillas López  
 Curso: Programación Avanzada

### Introducción

La evolución diferencial es un método para optimizar de manera iterativa un problema mejorando a los individuos de una población en cada iteración por medio de una función objetivo al que llamaremos costo [1].

El procedimiento de la evolución diferencial es el siguiente:

- **Mutación:** para cada individuo de la población  $P$ , un nuevo individuo es generado por la siguiente ecuación  $u_{i,G+1} = x_{r1,G} + F * (x_{r2,G} - x_{r3,G})$ . Donde  $u_{i,G+1}$  es el mutante,  $x_{r,G}$  son individuos de  $P$  seleccionados aleatoriamente y  $F$  es un escalar de tipo real con rango  $[0, 2]$
- **Cruza:** Para preservar la diversidad en la población de individuos, los parámetros del individuo mutado tienen una probabilidad de conformar al individuo mutante.
- **Selección:** Si el individuo mutante tiene un costo menor que el individuo de  $P$ , sustituye al individuo  $P$ .

A continuación se presentara el código en C donde se realiza el procedimiento antes mencionado.

```
// Mutacion para cada individuo de la poblacion
for (int i = 0; i < n_individuos; i++) {
    int random = 0;
    int CR = 1;

    // Seleccionando aleatoriamente el individuo a
    do {
        random = rand() % 100;
        a_i = random;
    } while (a_i == i);

    // Seleccionando aleatoriamente el individuo b
    do {
        random = rand() % 100;
        b_i = random;
    } while (b_i == i || b_i == a_i);

    // Seleccionando aleatoriamente el individuo c
    do {
        random = rand() % 100;
        c_i = random;
    } while (c_i == i || c_i == a_i || c_i == b_i);

    for (int j = 0; j < parametros; j++) {
        // Selecciona aleatoriamente si mutar el parametro con una probabilidad
        // del 10%
        if (CR == rand() % 10) {
            printf("\nEntro en el crossover en el individuo %d\n", i);
            hijos[i].x[j] = p[a_i].x[j] + F * (p[b_i].x[j] - p[c_i].x[j]);
            while (hijos[i].x[j] < -5.12 || hijos[i].x[j] > 5.12) {
                if (hijos[i].x[j] < -5.12) {
                    hijos[i].x[j] += RAND_REAL(1.0, 0.0);
                }
                if (hijos[i].x[j] > 5.12) {

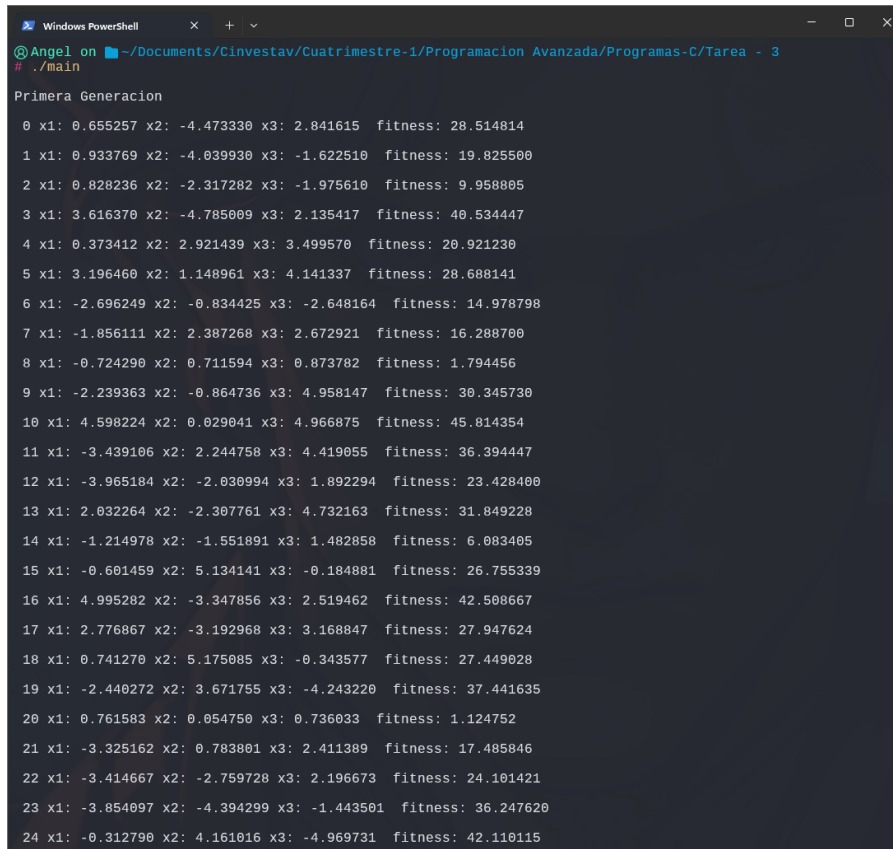
```

```

        hijos[i].x[j] -= RAND_REAL(1.0, 0.0);
    }
}
} else {
    hijos[i].x[j] = p[i].x[j];
}
}
// Seleccion
hijos[i].fitness = f(hijos[i].x);
if (hijos[i].fitness < p[i].fitness) {
    printf("\n-Se intercambio de padre-%d-a hijo-%d\n", i, i);
    p[i] = hijos[i];
}
}
}

```

Para este programa se utilizó una población de 25 individuos y se realizó únicamente una iteración para crear la siguiente generación. A continuación se mostrarán las capturas de la ejecución del código.



```

Windows PowerShell
@Angel on ~\Documents\Cinvestav\Cuatrimestre-1/Programacion Avanzada/Programas-C/Tarea - 3
# ./main

Primera Generacion

0 x1: 0.655257 x2: -4.473330 x3: 2.841615 fitness: 28.514814
1 x1: 0.933769 x2: -4.039930 x3: -1.622510 fitness: 19.825500
2 x1: 0.828236 x2: -2.317282 x3: -1.975610 fitness: 9.958805
3 x1: 3.616370 x2: -4.785009 x3: 2.135417 fitness: 40.534447
4 x1: 0.373412 x2: 2.921439 x3: 3.499570 fitness: 20.921230
5 x1: 3.196460 x2: 1.148961 x3: 4.141337 fitness: 28.688141
6 x1: -2.696249 x2: -0.834425 x3: -2.648164 fitness: 14.978798
7 x1: -1.856111 x2: 2.387268 x3: 2.672921 fitness: 16.288700
8 x1: -0.724290 x2: 0.711594 x3: 0.873782 fitness: 1.794456
9 x1: -2.239363 x2: -0.864736 x3: 4.958147 fitness: 30.345730
10 x1: 4.598224 x2: 0.029041 x3: 4.966875 fitness: 45.814354
11 x1: -3.439106 x2: 2.244758 x3: 4.419055 fitness: 36.394447
12 x1: -3.965184 x2: -2.030994 x3: 1.892294 fitness: 23.428400
13 x1: 2.032264 x2: -2.307761 x3: 4.732163 fitness: 31.849228
14 x1: -1.214978 x2: -1.551891 x3: 1.482858 fitness: 6.083405
15 x1: -0.601459 x2: 5.134141 x3: -0.184881 fitness: 26.755339
16 x1: 4.995282 x2: -3.347856 x3: 2.519462 fitness: 42.508667
17 x1: 2.776867 x2: -3.192968 x3: 3.168847 fitness: 27.947624
18 x1: 0.741270 x2: 5.175085 x3: -0.343577 fitness: 27.449028
19 x1: -2.440272 x2: 3.671755 x3: -4.243220 fitness: 37.441635
20 x1: 0.761583 x2: 0.054750 x3: 0.736033 fitness: 1.124752
21 x1: -3.325162 x2: 0.783801 x3: 2.411389 fitness: 17.485846
22 x1: -3.414667 x2: -2.759728 x3: 2.196673 fitness: 24.101421
23 x1: -3.854097 x2: -4.394299 x3: -1.443501 fitness: 36.247620
24 x1: -0.312790 x2: 4.161016 x3: -4.969731 fitness: 42.110115

```

Figure 1: Primera generación

```

Windows PowerShell
24 x1: -0.312790 x2: 4.161016 x3: -4.969731 fitness: 42.110115
Entro en el crossover en el individuo 1
Entro en el crossover en el individuo 4
Entro en el crossover en el individuo 6
Entro en el crossover en el individuo 7
Entro en el crossover en el individuo 11
Entro en el crossover en el individuo 15
Se intercambio de padre 15 a hijo 15
Entro en el crossover en el individuo 17
Entro en el crossover en el individuo 21
Entro en el crossover en el individuo 22
Entro en el crossover en el individuo 24
Se intercambio de padre 24 a hijo 24
Segunda Generacion
0 x1: 0.655257 x2: -4.473330 x3: 2.841615 fitness: 28.514814
1 x1: 0.933769 x2: -4.039930 x3: -1.622510 fitness: 19.825500
2 x1: 0.828236 x2: -2.317282 x3: -1.975610 fitness: 9.958805
3 x1: 3.616370 x2: -4.785009 x3: 2.135417 fitness: 40.534447
4 x1: 0.373412 x2: 2.921439 x3: 3.499570 fitness: 20.921230
5 x1: 3.196460 x2: 1.148961 x3: 4.141337 fitness: 28.688141
6 x1: -2.696249 x2: -0.834425 x3: -2.648164 fitness: 14.978798
7 x1: -1.856111 x2: 2.387268 x3: 2.672921 fitness: 16.288700
8 x1: -0.724290 x2: 0.711594 x3: 0.873782 fitness: 1.794456
9 x1: -2.239363 x2: -0.864736 x3: 4.958147 fitness: 30.345730
10 x1: 4.598224 x2: 0.029041 x3: 4.966875 fitness: 45.814354
11 x1: -3.439106 x2: 2.244758 x3: 4.419055 fitness: 36.394447
12 x1: -3.965184 x2: -2.030994 x3: 1.892294 fitness: 23.428400

```

Figure 2: Procedimiento de mutación

```

Windows PowerShell
Segunda Generacion
0 x1: 0.655257 x2: -4.473330 x3: 2.841615 fitness: 28.514814
1 x1: 0.933769 x2: -4.039930 x3: -1.622510 fitness: 19.825500
2 x1: 0.828236 x2: -2.317282 x3: -1.975610 fitness: 9.958805
3 x1: 3.616370 x2: -4.785009 x3: 2.135417 fitness: 40.534447
4 x1: 0.373412 x2: 2.921439 x3: 3.499570 fitness: 20.921230
5 x1: 3.196460 x2: 1.148961 x3: 4.141337 fitness: 28.688141
6 x1: -2.696249 x2: -0.834425 x3: -2.648164 fitness: 14.978798
7 x1: -1.856111 x2: 2.387268 x3: 2.672921 fitness: 16.288700
8 x1: -0.724290 x2: 0.711594 x3: 0.873782 fitness: 1.794456
9 x1: -2.239363 x2: -0.864736 x3: 4.958147 fitness: 30.345730
10 x1: 4.598224 x2: 0.029041 x3: 4.966875 fitness: 45.814354
11 x1: -3.439106 x2: 2.244758 x3: 4.419055 fitness: 36.394447
12 x1: -3.965184 x2: -2.030994 x3: 1.892294 fitness: 23.428400
13 x1: 2.032264 x2: -2.307761 x3: 4.732163 fitness: 31.849228
14 x1: -1.214978 x2: -1.551891 x3: 1.482858 fitness: 6.083405
15 x1: -0.601459 x2: 5.134141 x3: -2.121610 fitness: 31.222387
16 x1: 4.995282 x2: -3.347856 x3: 2.519462 fitness: 42.508667
17 x1: 2.776867 x2: -3.192968 x3: 3.168847 fitness: 27.947624
18 x1: 0.741270 x2: 5.175085 x3: -0.343577 fitness: 27.449028
19 x1: -2.440272 x2: 3.671755 x3: -4.243220 fitness: 37.441635
20 x1: 0.761583 x2: 0.054750 x3: 0.736033 fitness: 1.124752
21 x1: -3.325162 x2: 0.783801 x3: 2.411389 fitness: 17.485846
22 x1: -3.414667 x2: -2.759728 x3: 2.196673 fitness: 24.101421
23 x1: -3.854097 x2: -4.394299 x3: -1.443501 fitness: 36.247620
24 x1: -0.928056 x2: 4.161016 x3: -4.969731 fitness: 42.873566
@Angel on ~\Documents\Cinvestav\Cuatrimestre-1/Programacion Avanzada/Programas-C/Tarea - 3
#

```

Figure 3: Segunda generación

## **Bibliografía**

[1] R. Storn y K. Price, Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces, Journal of Global Optimization, vol. 11, n.º 4, pp. 341-359, dic. 1997, doi: 10.1023/A:1008202821328.