Reporte Evolutivos Tarea 3.
Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE).

Introducción.

De acuerdo al artículo estudiado, este algoritmo de Evolución Diferencial se deriva de dos algoritmos que consisten en modificaciones al algoritmo clásico, a saber, NSDE (Differential Evolution with Neighborhood Search) y SaDE (Self-adapting Differential Evolution); en esta practica, se analizaran e implementaran estos algoritmos utilizando el benchmark que se describe en el artículo "An Adaptive Differential Evolution Algorithm" de Nasimul Noman, Danushka Bollegala y Hitoshi Iba.

I. DE clásico

Todos los algoritmos que analizaremos aquí constan de 3 componentes principales: *Mutación*, *Cruza* y *Selección.*

Mutación. Se genera una población ***v*** de individuos mutados utilizando las siguientes estrategias:

$$\mathbf{v}_i = \mathbf{x}_{i_1} + F \cdot (\mathbf{x}_{i_2} - \mathbf{x}_{i_3}) \quad (1)$$
$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) \quad (2)$$
$$\mathbf{v}_i = \mathbf{x}_i + F \cdot (\mathbf{x}_{best} - \mathbf{x}_i) + F \cdot (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) \quad (3)$$
$$\mathbf{v}_i = \mathbf{x}_{best} + F \cdot (\mathbf{x}_{i_1} - \mathbf{x}_{i_2}) + F \cdot (\mathbf{x}_{i_3} - \mathbf{x}_{i_4}) \quad (4)$$
$$\mathbf{v}_i = \mathbf{x}_{i_1} + F \cdot (\mathbf{x}_{i_2} - \mathbf{x}_{i_3}) + F \cdot (\mathbf{x}_{i_4} - \mathbf{x}_{i_5}) \quad (5)$$

x_{best} denota el mejor individuo de la población ***x***; i_1, i_2, i_3, ... indican posiciones elegidas aleatoriamente (sin repetición) del total del individuos de la población ***x***, y F es una factor de escalamiento.

Cruza. Se genera una población u que consiste en una mezcla de individuos de la población mutada ***v*** y de la población original ***x***, de acuerdo a una cruza binomial que utiliza el siguiente criterio:

$$\mathbf{u}_i(j) = \begin{cases} \mathbf{v}_i(j), & if\ U_j(0,1) \leq CR\ or\ j = j_{rand} \\ \mathbf{x}_i(j), & otherwise. \end{cases}$$

$U_j(0,1)$ es un numero real dentro del intervalo [0,1] que se obtiene de una distribución uniforme, *CR* es una probabilidad de cruza y j_{rand} es un valor entero que nos permite asegurar que al menos un individuo de la población ***u*** sera diferente de la población original ***x***.

Selección. Consiste básicamente en un torneo binario, el individuo con mejor fitness pasa a la siguiente generación:

$$\mathbf{x}'_i = \begin{cases} \mathbf{u}_i, & if\ f(\mathbf{u}_i) \leq f(\mathbf{x}_i) \\ \mathbf{x}_i, & otherwise. \end{cases}$$

II. NSDE

Mutación. Se utilizan las mismas estrategias que el algoritmo clásico, pero el factor de escalamiento se elije de acuerdo al siguiente criterio:

$$F_i = \begin{cases} N_i(0.5, 0.5), & if\ U_i(0,1) < fp \\ \delta_i, & otherwise. \end{cases}$$

N_i(0.5,0.5) es un numero aleatorio que proviene de una distribución normal con media 0.5 y desviación típica de 0.5, δ_i es un numero aleatorio que proviene de una distribución Cauchy con parámetro de escala igual a 1, en esta practica se utilizo un factor de corrimiento igual a 0.5. La probabilidad *fp* de elegir entre una distribución y otra se fija en 0.5.

La Cruza y la Selección se realizan de la misma manera que para el algoritmo clásico.

III. SaDE

Mutación. En este algoritmo, se aplica la estrategia (1) o (3) de acuerdo el siguiente criterio:

$$\mathbf{v}_i = \begin{cases} Eq.\ (1), & if\ U_i(0,1) < p \\ Eq.\ (3), & otherwise. \end{cases}$$

la probabilidad *p* de elegir entre una y otra se inicializa en 0.5 y se actualiza cada 50 generaciones en función del numero de individuos que pasan a la siguiente generación y del numero de individuos que son descartados:

$$p = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)}$$

ns_1, ns_2 = numero de individuos que pasan a la siguiente generación originados de la estrategia (1) y (3) respectivamente.

nf_1, nf_2 = numero de individuos que son descartados originados de la estrategia (1) y (3) respectivamente.

El factor de escalamiento F se elige de una distribución normal con media 0.5 y desviación típica 0.3 para cada individuo mutado.

Cruza. Aquí, el factor de cruza se obtiene de una distribución normal con desviación típica 0.1 y media *CRm*, la cual se inicializa en 0.5 y se actualiza cada 25 generaciones de acuerdo a los valores *CRi* generados que produjeron individuos que pasaron a la siguiente generación, estos valores se almacenan en un vector *CRrec* y la media de la distribución se actualiza de la siguiente manera:

$$CRm = \frac{1}{|CR_{rec}|} \sum_{k=1}^{|CR_{rec}|} CR_{rec}(k)$$

|*CRrec*| denota la magnitud del vector *CRrec*.

Selección. Se realiza de la misma manera que para el algoritmo clásico.

IV. SaNSDE

Mutación. Con relación a las estrategias de mutación, se utiliza el mismo criterio que en SaDE, pero el factor de escalamiento F se elige de la siguiente manera:

$$F_i = \begin{cases} N_i(0.5, 0.3), & if\ U_i(0,1) < fp \\ \delta_i, & otherwise. \end{cases}$$

Luego, la probabilidad fp se actualiza con la misma estrategia que se utilizo en SaDE para actualizar p.

Cruza. Se utiliza una estrategia similar que en SaDE, pero ahora se calcula un promedio ponderado, donde la ponderación depende de que tanta mejora produjo la mutación en el individuo que paso a la siguiente generación:

$$CRm = \sum_{k=1}^{|CR_{rec}|} w_k * CR_{rec}(k)$$

$$w_k = \Delta f_{rec}(k) / \left(\sum_{k=1}^{|\Delta f_{rec}|} \Delta f_{rec}(k) \right)$$

Donde $\Delta f_{rec}(k) = f(\mathbf{x}_k) - f(\mathbf{v}_k)$

Benchmark.

El benchmark utilizado en esta practica consiste en las siguiente funciones:

Sphere	$f_{sph}(\vec{\mathbf{x}}) = \sum_{i=1}^{N} x_i^2$	$[-100, 100]^{30}$
Elliptic	$f_{ell}(\vec{\mathbf{x}}) = \sum_{i=1}^{N} \left(10^6\right)^{\left(\frac{i-1}{N-1}\right)} x_i^2$	$[-100, 100]^{30}$
Schwefel 1.2	$f_{sch}(\vec{\mathbf{x}}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^{30}$
Ackley	$f_{ack}(\vec{\mathbf{x}}) = 20 + e - 20e^{\left(-0.2\sqrt{\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)}\right)} - e^{\left(\frac{1}{N}\sum_{i=1}^{N}\cos(2\pi x_i)\right)}$	$[-32, 32]^{30}$
Rastrigin	$f_{ras}(\vec{\mathbf{x}}) = 10N + \sum_{i=1}^{N}(x_i^2 - 10\cos(2\pi x_i))$	$[-5.12, 5.12]^{30}$
Griewank	$f_{grw}(\vec{\mathbf{x}}) = \sum_{i=1}^{N} x_i^2/4000 - \prod_{i=1}^{N}\cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^{30}$
Rosenbrock	$f_{ros}(\vec{\mathbf{x}}) = \sum_{i=1}^{N-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	$[-100, 100]^{30}$
Weierstrass	$f_{wrs}(\vec{\mathbf{x}}) = \sum_{i=1}^{N}\left(\sum_{k=0}^{k_{max}}(a^k\cos(2\pi b^k(x_i + 0.5)))\right) - N\sum_{k=0}^{k_{max}}(a^k\cos(\pi b^k)$	$[-0.5, 0.5]^{30}$
Schaffer	$f_{scf}(\vec{\mathbf{x}}) = \sum_{i=1}^{N} F(x_i, x_{i+1}); \quad x_{N+1} = x_1$ where, $F(x,y) = 0.5 + \frac{sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.001(x^2+y^2))^2}$	$[-0.5, 0.5]^{30}$
Salomon	$f_{sal}(\vec{\mathbf{x}}) = 1 - \cos(2\pi\|\|\mathbf{x}\|\|) + 0.1\|\|\mathbf{x}\|\|$ where, $\|\|\mathbf{x}\|\| = \sqrt{\left(\sum_{i=1}^{N} x_i^2\right)}$	$[-100, 100]^{30}$

Resultados.

Los algoritmos se ejecutaron un total de 30 repeticiones, para cada una de las funciones del benchmark, en dimensiones d = 30 y d = 50, para cada una de estas repeticiones se guardo el mejor resultado obtenido y se calcularon los valores de fitness mínimo (*Fmin*), máximo (*Fmax*), fitness promedio (*Favg*), mediana (*Fmedian*), y desviación estándar (*FstdDev*), el criterio de paro se fijo en 10^5 x d evaluaciones, o al momento en que el algoritmo alcanza un valor de fitness inferior a 1 x 10^{-8}. Tambien se considero el ratio de éxito (S_ratio) que se obtuvo para cada funcion del bechmark, este porcentaje muestra cuantas veces dentro de las 30 repeticiones se alcanzo el valor de fitness esperado.

Para el algoritmo clásico se eligió un tamaño de población de 200, mientras que para los demás algoritmos se determino por experimentación que debían tener un tamaño mas grande de población, para permitir que las distribuciones de probabilidad utilizadas en cada uno de ellos realizaran una búsqueda eficiente, y no se pudieron obtener buenos resultados con una población de 200, así que el tamaño el población se elevo a 1000, con su respectiva penalización en evaluaciones de función.

Para generar los numero aleatorios utilizados en esta practica, se utiliza la librería <random> de C++11, las funciones que generan estas distribuciones se encuentran codificadas en el fichero “rand.hpp”.

DE									
Function	**Dim**	**Fmin**	**Fmax**	**Favg**	**Fmedian**	**FstdDev**	**CR**	**Factor**	**S_ratio**
Sphere	30	6.36E-009	9.98E-009	8.73E-009	8.92E-009	9.32E-010	0.9	0.6	100
Sphere	50	3.83E+002	1.48E+003	8.33E+002	8.60E+002	2.93E+002	0.8	0.6	0
Elliptic	30	7.26E-009	9.89E-009	9.16E-009	9.31E-009	5.78E-010	1	0.6	100
Elliptic	50	8.75E+003	8.52E+004	3.29E+004	2.68E+004	2.05E+004	1	0.7	0
Schwefel	30	6.28E-009	9.99E-009	8.83E-009	9.02E-009	9.41E-010	1	0.6	100
Schwefel	50	1.64E+002	8.27E+002	3.80E+002	2.82E+002	2.02E+002	0.9	0.7	0
Ackley	30	7.64E-009	1.22E+001	1.12E+000	9.70E-009	3.06E+000	0.9	0.6	83.33
Ackley	50	5.47E-001	3.56E+000	1.48E+000	1.45E+000	6.29E-001	0.9	0.6	0
Rastrigin	30	5.92E+001	1.34E+002	9.25E+001	8.65E+001	1.87E+001	0.9	0.4	0
Rastrigin	50	1.31E+002	2.39E+002	1.98E+002	1.96E+002	2.59E+001	0.4	0.4	0
Griewank	30	6.35E-009	1.23E-002	1.23E-003	9.13E-009	3.23E-003	0.9	0.6	86.66
Griewank	50	3.52E+000	1.36E+001	8.94E+000	8.53E+000	2.56E+000	0.7	0.6	0
Rosenbrock	30	6.32E-009	3.99E+000	3.99E-001	9.32E-009	1.20E+000	0.9	0.6	90
Rosenbrock	50	5.23E+005	1.22E+007	2.86E+006	1.86E+006	2.51E+006	0.8	0.6	0
Weierstrass	30	4.47E+000	9.02E+000	6.32E+000	6.07E+000	1.24E+000	0.9	0.4	0
Weierstrass	50	1.57E+001	2.38E+001	1.99E+001	2.03E+001	2.03E+000	0.9	0.4	0
Schaffer	30	5.98E-009	9.98E-009	8.94E-009	9.21E-009	8.55E-010	0.9	0.6	100
Schaffer	50	1.80E-002	7.74E-002	4.03E-002	3.85E-002	1.36E-002	0.9	0.6	0
Salomon	30	9.99E-002	3.00E-001	1.63E-001	2.00E-001	6.05E-002	0.9	0.6	0
Salomon	50	1.90E+000	4.20E+000	3.00E+000	3.05E+000	5.71E-001	0.9	0.6	0

Tabla 1. Desempeño del algoritmo DE clásico.

La Tabla 1 muestra los resultados que se obtuvieron para el algoritmo clásico, en general los 4 algoritmo resultaron bastante malos en dimensión 50, en este caso especialmente evidente en la función elíptica y la función de Rosenbrock, aunque para esta función, pudo alcanzar el valor esperado en el 90% de los casos. Los parámetros de cruza y factor de escalamiento se eligieron experimentalmente, buscando los parámetros que produjeran el mejor comportamiento.

NSDE								
Function	Dim	Fmin	Fmax	Favg	Fmedian	FstdDev	Factor	S_ratio
Sphere	30	8.41E-009	9.98E-009	9.61E-009	9.72E-009	3.57E-010	0.8	100
Sphere	50	2.14E+002	8.76E+002	4.57E+002	4.27E+002	1.62E+002	1	0
Elliptic	30	8.83E-009	9.99E-009	9.64E-009	9.68E-009	2.96E-010	0.9	100
Elliptic	50	4.79E+004	7.68E+005	1.75E+005	1.38E+005	1.42E+005	0.4	0
Schwefel	30	8.09E-009	1.00E-008	9.53E-009	9.66E-009	4.23E-010	1	100
Schwefel	50	1.29E+002	1.19E+003	4.61E+002	4.59E+002	2.09E+002	0.7	0
Ackley	30	2.74E-008	2.74E+000	1.51E+000	1.78E+000	8.37E-001	0.8	0
Ackley	50	4.69E+000	1.18E+001	6.80E+000	6.39E+000	1.60E+000	0.8	0
Rastrigin	30	1.39E+001	4.38E+001	2.45E+001	2.29E+001	7.48E+000	0.7	0
Rastrigin	50	5.55E+001	1.22E+002	8.54E+001	8.36E+001	1.48E+001	0.7	0
Griewank	30	9.85E-009	6.15E-002	2.12E-002	1.85E-002	1.58E-002	1	13.33
Griewank	50	2.62E+000	1.27E+001	5.67E+000	5.30E+000	2.21E+000	0.9	0
Rosenbrock	30	4.01E-001	1.91E+001	9.84E+000	1.16E+001	4.65E+000	0.9	0
Rosenbrock	50	3.70E+004	3.24E+006	7.75E+005	4.94E+005	8.29E+005	1	0
Weierstrass	30	1.36E+000	5.04E+000	3.27E+000	3.18E+000	1.03E+000	0.6	0
Weierstrass	50	1.23E+001	1.85E+001	1.49E+001	1.52E+001	1.73E+000	0.3	0
Schaffer	30	6.93E-009	1.00E-008	9.55E-009	9.79E-009	6.26E-010	0.6	100
Schaffer	50	9.45E-003	7.86E-002	3.01E-002	2.60E-002	1.53E-002	0.9	0
Salomon	30	4.00E-001	1.40E+000	8.23E-001	9.00E-001	2.03E-001	0.9	0
Salomon	50	2.20E+000	4.10E+000	3.22E+000	3.25E+000	4.37E-001	0.8	0

Tabla 2. Desempeño del algoritmo NSDE.

SaDE							
Function	imensio	Fmin	Fmax	Favg	Fmedian	FstdDev	S_ratio
Sphere	30	6.37E-009	9.84E-009	8.73E-009	8.83E-009	7.70E-010	100
Sphere	50	2.09E-001	1.16E+001	3.23E+000	1.99E+000	3.30E+000	0
Elliptic	30	2.59E-001	1.05E+004	1.42E+003	6.18E+002	2.14E+003	0
Elliptic	50	3.86E+004	3.74E+005	1.19E+005	8.94E+004	7.98E+004	0
Schwefel	30	7.15E-009	9.99E-009	8.78E-009	8.77E-009	7.20E-010	100
Schwefel	50	1.54E+000	3.54E+001	1.23E+001	1.02E+001	9.01E+000	0
Ackley	30	7.90E-009	1.16E+000	3.85E-002	9.36E-009	2.07E-001	96.66
Ackley	50	1.59E+000	3.48E+000	2.44E+000	2.49E+000	4.51E-001	0
Rastrigin	30	1.19E+002	1.58E+002	1.37E+002	1.39E+002	9.05E+000	0
Rastrigin	50	1.88E+001	3.08E+002	2.39E+002	2.76E+002	8.75E+001	0
Griewank	30	7.06E-009	3.69E-002	7.14E-003	9.81E-009	1.02E-002	50
Griewank	50	1.26E-001	1.17E+000	8.09E-001	9.62E-001	3.11E-001	0
Rosenbrock	30	2.06E+001	2.49E+002	4.68E+001	2.64E+001	4.63E+001	0
Rosenbrock	50	2.11E+002	6.08E+003	1.64E+003	8.90E+002	1.68E+003	0
Weierstrass	30	6.10E-002	1.75E+000	4.77E-001	2.95E-001	4.70E-001	0
Weierstrass	50	4.96E+000	1.10E+001	8.13E+000	8.00E+000	1.62E+000	0
Schaffer	30	6.67E-009	9.97E-009	8.89E-009	9.08E-009	7.71E-010	100
Schaffer	50	1.64E-005	4.54E-004	1.40E-004	1.25E-004	1.13E-004	0
Salomon	30	9.99E-002	2.00E-001	1.53E-001	2.00E-001	4.99E-002	0
Salomon	50	4.00E-001	1.60E+000	7.60E-001	7.00E-001	2.70E-001	0

Tabla 3. Desempeño del algoritmo SaDE.

Las tablas 2 y 3, muestran los resultados obtenidos para el NSDE y el SaDE respectivamente, en general, no resultaron ser mejores que el algoritmo clásico con una buena elección de parámetros, incluso el SaDE resulto ser sorpresivamente malo con la función elíptica en dimensión 30.

SaNSDE							
Function	imensio	Fmin	Fmax	Favg	Fmedian	FstdDev	S_ratio
Sphere	30	7.19E-009	9.97E-009	9.21E-009	9.24E-009	6.13E-010	100
Sphere	50	3.12E-002	4.71E+000	1.21E+000	8.53E-001	1.16E+000	0
Elliptic	30	4.61E-008	1.57E+003	1.57E+002	1.36E+001	3.56E+002	0
Elliptic	50	3.28E+004	2.10E+005	9.77E+004	9.04E+004	5.48E+004	0
Schwefel	30	6.92E-009	9.95E-009	8.85E-009	8.96E-009	8.38E-010	100
Schwefel	50	4.73E-001	1.75E+001	5.19E+000	2.93E+000	5.12E+000	0
Ackley	30	8.66E-009	9.31E-001	3.10E-002	9.56E-009	1.67E-001	96.66
Ackley	50	1.56E+000	1.96E+001	5.24E+000	2.69E+000	5.94E+000	0
Rastrigin	30	8.95E+000	4.28E+001	2.28E+001	2.14E+001	7.85E+000	0
Rastrigin	50	3.06E+001	8.48E+001	5.56E+001	5.74E+001	1.30E+001	0
Griewank	30	6.76E-009	2.22E-002	5.25E-003	9.73E-009	7.10E-003	60
Griewank	50	5.50E-002	1.04E+000	5.38E-001	5.87E-001	2.93E-001	0
Rosenbrock	30	2.02E+001	1.00E+002	2.74E+001	2.38E+001	1.47E+001	0
Rosenbrock	50	2.25E+002	2.73E+003	8.80E+002	6.46E+002	6.08E+002	0
Weierstrass	30	1.23E-002	1.69E+000	5.60E-001	4.58E-001	4.86E-001	0
Weierstrass	50	4.59E+000	1.04E+001	7.46E+000	7.31E+000	1.48E+000	0
Schaffer	30	7.02E-009	9.97E-009	9.00E-009	9.24E-009	7.43E-010	100
Schaffer	50	2.60E-006	2.41E-004	6.28E-005	4.70E-005	5.07E-005	0
Salomon	30	9.99E-002	2.00E-001	1.57E-001	2.00E-001	4.96E-002	0
Salomon	50	5.00E-001	1.00E+000	7.03E-001	7.00E-001	1.47E-001	0

Tabla 4. Desempeño del algoritmo SaNSDE.

La Tabla 4 muestra el desempeño del algoritmo propuesto en el articulo, funciona bastante bien para varias funciones en dimensión 30, pero no es significativamente mejor que el algoritmo clásico, resulto especialmente malo para la función elíptica en dimensión 50 y en general deficiente para la función Rosenbrock, función en la cual el algoritmo clásico fue bastante bueno, en general supera a las variantes NSDE y SaDE, y es mucho mas robusto que el algoritmo clásico, porque no se requiere ajustar ningún parámetro, con excepción del tamaño de la población en el caso de funciones con dimensiones muy grandes, mientras que se sabe que el algoritmo clásico es bastante sensible a los parámetros de cruza y mutación.

La Tabla 5 resume los mejores resultados obtenidos en cada algoritmo, se resalta en negro el mejor de todos los resultados, en el caso de que mas de un algoritmo haya alcanzado el valor de fitness esperado, se resaltan todos los valores. Al final se indica el “Score” que obtuvo cada uno.

Como ya se menciono anteriormente, las variantes NSDE y SaDE no resultaron ser mejores al DE clásico bien ajustado, aunque el SaDE es mas robusto ya que no se le tiene que ajustar ningún parámetro, el SaNSDE es solo un poco mejor que en clásico en sus resultados, mientras que también es robusto; se observa también que en general, ningún algoritmo pudo minimizar eficientemente la función elíptica en dimensión 50, y que el algoritmo clásico resulto ser mucho mejor que los demás con la función de Rosenbrock.

Function	DE	NSDE	SaDE	SaNSDE
Sphere	**6.36E-009**	**8.41E-009**	**6.37E-009**	**7.19E-009**
Sphere	3.83E+002	2.14E+002	2.09E-001	**3.12E-002**
Elliptic	**7.26E-009**	**8.83E-009**	2.59E-001	4.61E-008
Elliptic	**8.75E+003**	4.79E+004	3.86E+004	3.28E+004
Schwefel	**6.28E-009**	**8.09E-009**	**7.15E-009**	**6.92E-009**
Schwefel	1.64E+002	1.29E+002	1.54E+000	**4.73E-001**
Ackley	**7.64E-009**	2.74E-008	**7.90E-009**	**8.66E-009**
Ackley	**5.47E-001**	4.69E+000	1.59E+000	1.56E+000
Rastrigin	5.92E+001	1.39E+001	1.19E+002	**8.95E+000**
Rastrigin	1.31E+002	5.55E+001	**1.88E+001**	3.06E+001
Griewank	**6.35E-009**	**9.85E-009**	**7.06E-009**	**6.76E-009**
Griewank	3.52E+000	2.62E+000	1.26E-001	**5.50E-002**
Rosenbrock	**6.32E-009**	4.01E-001	2.06E+001	2.02E+001
Rosenbrock	5.23E+005	3.70E+004	2.11E+002	**2.25E+002**
Weierstrass	4.47E+000	1.36E+000	6.10E-002	**1.23E-002**
Weierstrass	1.57E+001	1.23E+001	4.96E+000	**4.59E+000**
Schaffer	**5.98E-009**	**6.93E-009**	**6.67E-009**	**7.02E-009**
Schaffer	1.80E-002	9.45E-003	1.64E-005	**2.60E-006**
Salomon	**9.99E-002**	4.00E-001	**9.99E-002**	**9.99E-002**
Salomon	1.90E+000	2.20E+000	**4.00E-001**	5.00E-001
Score	**10**	**5**	**8**	**14**

Tabla 5. Resumen de los mejores resultados para cada algoritmo.

Conclusiones.

Se concluye en primer lugar, que no se pudo obtener los resultados presentados en el articulo, al menos con el tamaño de población que ahí se sugería (N = 100), esto puede deberse a que los generadores que se utilizaron en esta practica pudieran no ser tan buenos como los utilizados en el articulo, o que haya faltado implementar algún detalle que no es mencionado en el articulo.

Por otra parte, se verifico la tesis propuesta en el articulo, es decir, el SaNSDE resulto ser en general mejor que las otras dos variantes de algoritmo de Evolución Diferencial clásico, esto se puede apreciar por la tabla 5.

También se recalca que hubo ciertos detalles que no quedaron claros del articulo, por ejemplo, no se especifica el factor de corrimiento de la distribución de Cauchy para el NSDE, se utilizo el mismo valor que la media de la distribución normal que se utiliza en ese mismo criterio, por razón de que las dos distribuciones estuvieran centradas en el mismo punto, y que la distribución de Cauchy, por ser de colas pesadas, permitiera mucha mas variación entre los coeficientes.

Por ultimo, se concluye que por un margen muy angosto, el algoritmo propuesto resulta ser el mejor, ya que pudo muchas veces igualar o superar a un DE clásico con parámetros ajustados, con la ventaja de que este no requiere de ningún ajuste de parámetros y puede ser fácilmente adaptado a cualquier problema.

Compilación/Ejecución.

Los programas implementados incluyen un makefile para compilarse, que soporta los comandos *make*, *make run* y *make clean*, así como un script de ejecución en el cluster para correr todas las instancias, los resultados se generan en un archivo para cada ejecución.

Los programas reciben tres parámetros:

1. Tamaño de la población.
2. Dimensión de las variables (se utilizó 30 y 50).
3. Referencia a la función objetivo a evaluar:
   [0] = Sphere;
   [1] = Elliptic;
   [2] = Schwefel;
   [3] = Ackley;
   [4] = Rastrigin;
   [5] = Griewank;
   [6] = Rosenbrock;
   [7] = Weierstrass;
   [8] = Schaffer;
   [9] = Salomon;

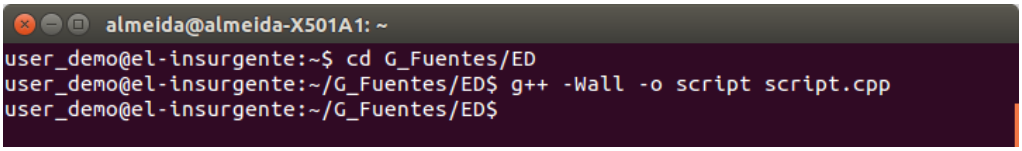
El NSDE recibe además:

4. Probabilidad de cruza CR

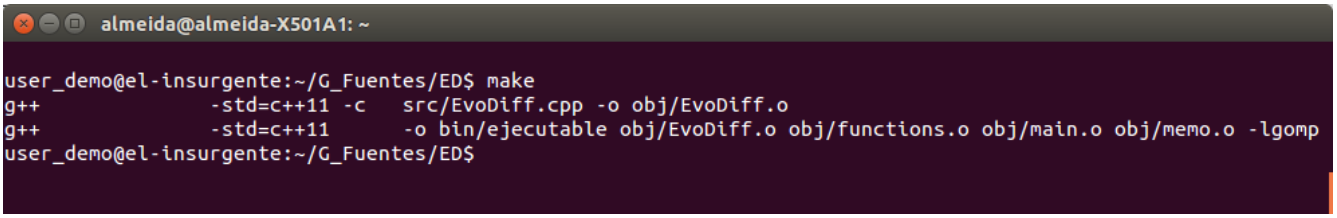
El algoritmo clásico recibe además:

4. Probabilidad de cruza CR
5. Valor del factor de escalamiento

Compilación del script:



Compilación del programa:



Ejecución del script: