Evolución Diferencial con Factor de Mutación Dinámico

Cecilia Sosa Toranzo y Guillermo Leguizamón

Universidad Nacional de San Luis,
Facultad de Ciencias Físico Matemáticas y Naturales,
Av. Ejército de Los Andes 950, D5700HHW, San Luis, Argentina
ceciliasosatoranzo@gmail.com, legui@unsl.edu.ar

Resumen El algoritmo de Evolución Diferencial (DE) es un método de optimización para problemas complejos. Como todo método de optimización posee parámetros que deben ser debidamente ajustados para proveer soluciones de buena calidad. Entre estos parámetros se encuentra $F \in [0, \infty)$, el factor de escala de la mutación, que afecta la velocidad con la cual evoluciona la población. Dado que dicho factor juega un papel importante en la obtención del óptimo global, en el presente trabajo se realiza un estudio de algoritmos de Evolución Diferencial que implementan factor F constante y otros que lo hacen considerando una variación dinámica del parámetro F en función del tiempo. El estudio se realiza sobre un conjunto de funciones escalables ampliamente difundidas y estudiadas por la comunidad de computación evolutiva.

Keywords: Evolución Diferencial, factor de mutación estático y dinámico, problemas de optimización de alta dimensionalidad, escalabilidad.

1. Introduction

El algoritmo de Evolución Diferencial [9](DE por sus siglas en inglés) es una herramienta simple pero poderosa para resolver problemas de optimización global. Sin embargo, los parámetros de control involucrados en el DE están ampliamente relacionados con el problema bajo consideración y por lo tanto influyen en el rendimiento del mismo. Una de las dificultades que se puede encontrar en el DE es la temprana convergencia a óptimos locales, sobretodo en funciones complejas de alta dimensionalidad. El parámetro relacionado a la velocidad de convergencia es el factor de mutación F. La buena elección del parámetro F incrementa la precisión de la solución y estimula la capacidad de escapar de óptimos locales. En general, la habilidad de realizar búsqueda local (explotación) se logra a través de valores de F pequeños y, por el contrario, realizar búsqueda global (exploración) se consigue con mayores valores de F [12].

Hasta la actualidad se han realizado numerosas aproximaciones para determinar el valor del factor F. Una de ellas es elegir un valor fijo para F (a través de la experiencia) que logre el balance entre explotación y exploración [9]. Otros

estudios han propuesto variar el valor de F de manera aleatoria respondiendo a alguna distribución de probabilidades (generalmente una distribución uniforme) [2,11,1,7]. Por otro lado, también se ha experimentado con un factor de mutación variante en el tiempo [2]. En este caso, cada generación produce un nuevo valor para el factor F siguiendo una función lineal decreciente. Esto significa decrementar el valor de F linealmente en relación al tiempo (generaciones o iteraciones) ya que en la mayoría de los métodos de optimización, es bueno que en las primeras etapas de búsqueda se realice una exploración del espacio de soluciones y durante las últimas una explotación de la zona donde podría encontrarse el óptimo global.

Encontrar el valor adecuado del factor de mutación F, para una tarea específica, puede insumir una gran cantidad de tiempo y de cómputo. Por lo tanto, en este trabajo se propone utilizar una función para generar valores de F dinámicamente. Para lograr un buen desempeño del DE en problemas de dimensionalidad alta ($D \geq 100$), se decidió experimentar con el parámetro F variable en el rango $[0, 1]$ en función del tiempo transcurrido (es decir, el valor del factor de escala en el tiempo t está dado por $F(t)$). Se optaron por cinco funciones $F(t)$ que siguen distintos comportamientos.

El resto del presente trabajo se organiza de la siguiente manera: en la sección 2 se da una breve descripción del algoritmo DE. Posteriormente, en la sección 3 se brinda una descripción detallada de la propuesta realizada sobre el factor de escala F. La sección 4 presenta los estudios experimentales realizados a las diferentes variantes del DE para obtener las conclusiones expuestas en la sección 5.

2. Evolución Diferencial

El algoritmo DE es un algoritmo de optimización estocástico introducido por Storn y Price en 1996 [9]. Sea $S \subseteq \mathbb{R}^D$ el espacio de búsqueda del problema en consideración, el DE involucra una población de NP vectores (soluciones candidatas) $\mathbf{x}_{i,g} = \{x_{1i,g}, x_{2i,g}, ..., x_{Di,g}\} \in S$, $i = 1, 2, ..., NP$. Cada $x_{ji,g}$ se corresponde con una variable de decisión del problema y g indica la generación a la cual pertenece el vector. Dichos vectores, luego de ser inicializados, son sometidos a operaciones de mutación, recombinación y selección en cada generación g.

Inicialización Se definen previamente los límites inferiores y superiores para cada variable de decisión: $x_j^i \leq x_{ji,1} \leq x_j^s$.

Posteriormente se seleccionan, aleatoria y uniformemente, los valores iniciales de las variables de decisión sobre los intervalos $[x_j^i, x_j^s]$.

Mutación Por cada vector $\mathbf{x}_{i,g}$ (vector objetivo) en la generación g, se crea un vector mutado $\mathbf{v}_{i,g} = \{v_{1i,g}, v_{2i,g}, ..., v_{Di,g}\}$ utilizando, algunas de varias estrategias. Para clasificar dichas variantes se utiliza la notación DE/x/y/z, donde x indica el vector a ser mutado ("rand" o "best"), y la cantidad de restas de

vectores realizadas (1 o 2), y z denota el esquema de recombinación utilizado ("bin": binomial o "exp":exponencial).

Las estrategias mas comúnmente utilizadas para generar $\mathbf{v}_{i,g}$ son:

1. DE/rand/1/bin: $\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F(\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g})$
2. DE/best/1/bin: $\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$

donde los índices $r1, r2, r3$ son enteros aleatorios y mutuamente diferentes generados en el rango $[1, NP]$. F es un factor entre $[0, \infty)$ para escalar la diferencia de vectores (mutación) y $\mathbf{x}_{best,g}$ es el vector con mejor valor de fitness en la población en la generación g.

Recombinación La operación de recombinación (crossover, en inglés) es aplicada a cada parte del vector mutado generado $\mathbf{v}_{i,g}$ y su correspondiente vector objetivo $\mathbf{x}_{i,g}$ para generar un vector de prueba $\mathbf{u}_{i,g} = \{u_{1i,g}, u_{2i,g}, ..., u_{Di,g}\}$.

$$u_{ji,g} = \begin{cases} v_{ji,g} & \text{si } (rand_j[0,1] \leq CR) \text{ ó } (j = j_{rand}), \\ x_{ji,g} & \text{en otro caso;} \end{cases} \tag{1}$$

donde $j = 1, 2, ..., D$, CR es una constante que indica la probabilidad de recombinación en el rango $[0, 1)$ y j_{rand} es un entero aleatorio elegido en el rango $[1, NP]$ para asegurar que el vector de prueba sea diferente del vector objetivo correspondiente. El operador dado en la ecuación 1 corresponde al crossover binomial.

Selección Se compara el valor de fitness de cada vector de prueba $f(\mathbf{u}_{i,g})$ con su correspondiente vector objetivo $f(\mathbf{x}_{i,g})$ en la población actual. El vector con mejor valor de fitness es el que entrará en la población de la siguiente generación.

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{si } f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{en otro caso.} \end{cases} \tag{2}$$

Las últimas tres operaciones son repetidas de generación en generación hasta que sea satisfecho un criterio de detención específico.

3. Propuesta para el Factor de Escala de Mutación *F*

Teniendo en cuenta la idea de variar el factor de escala en el tiempo, explorar el espacio de soluciones en las primeras iteraciones y explotarlo en las últimas; el presente trabajo introduce otras funciones (además de la lineal) para el factor de mutación F en el rango $[0, 1]$. Las funciones consideradas se presentan en la figura 1.

Las funciones $F(t)$ pueden ser agrupadas en tres clases. Las dos primeras funciones ($F_1(t)$ y $F_2(t)$) descienden rápidamente y toman valores cercanos a 0 en las últimas iteraciones. $F_3(t)$ obtiene valores proporcionales al paso del tiempo. Las últimas dos funciones se corresponden a funciones Gaussianas con $altura = 1$, $media = 0$ y una alta varianza (2000000 y 5000000 respectivamente), y se mantienen valores de F altos durante una mayor cantidad de tiempo.

- $F_1(t) = 0,9987^t$
- $F_2(t) = \frac{3}{0,01t+3}$
- $F_3(t) = -\frac{1}{5000}t + 1$
- $F_4(t) = e^{\frac{t^2}{4000000}}$
- $F_5(t) = e^{\frac{t^2}{10000000}}$

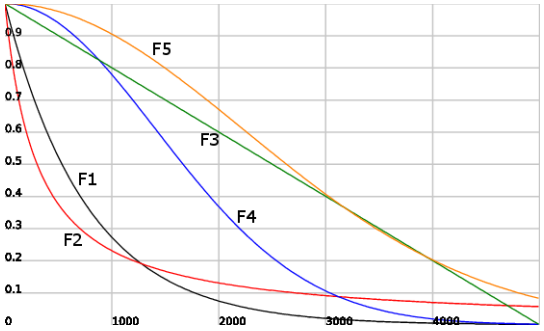

Figura 1. Funciones alternativas para F variables

4. Estudio Experimental

Para determinar las diferentes variantes del factor F propuestos en la sección anterior, se realizaron estudios experimentales para compararlos, incluyendo un algoritmo estándar DE con factor fijo $F = 0,75$. Se utilizó un máximo de iteraciones $MaxIt = 5000$, parámetro de recombinación $CR = 0,7$ y tamaño de población $NP = 200$. Se probaron dos de las estrategias de mutación: DE/best/1/bin (estrategia **B** por **B**est) y DE/rand/1/bin (estrategia **R** por **R**andom) debido a que son las más comúnmente utilizadas. Para cada combinación de parámetros se realizaron 30 ejecuciones con distintas semillas. Los algoritmos bajo observación tienen la siguiente nomenclatura: $E/F(t)$, donde E es la estrategia del algoritmo (B o R), y $F(t)$ es la estrategia de F elegida (1,2,3,4,5, para $F_i(t)$ o Cte para $F = 0,75$).

Las variantes de DE bajo estudio fueron aplicadas a seis funciones, un subconjunto de las funciones *desplazadas y escalables* descritas en el benchmark del CEC 2008 [10]: f_1 (Sphere), f_2 (Problema 2.21 de Schwefel), f_3 (Rosenbrock), f_4 (Rastrigin), f_5 (Griewank) y f_6 (Ackley).

La finalidad del presente trabajo es obtener conclusiones relevantes en cuanto al desempeño del DE con las variantes antes mencionadas. Para ello, se realizaron estudios iniciales con dimensión $D = 100$ y luego con $D = 200$ y $D = 500$ para analizar sus respectivas capacidades de escalabilidad.

4.1. Estudios preliminares

Inicialmente se estudió el comportamiento del DE con todas las variantes del factor de mutación sobre el benchmark antes mencionado para $D = 100$ (ver Cuadro 1). Para poder establecer las diferencias entre dichas variantes, se realizaron diversos tests estadísticos según se explica a continuación.

Dado que las muestras obtenidas no corresponden a una distribución normal, se aplicó para la comparación de los diferentes algoritmos, el test no paramétrico *Kruskal-Wallis*. Dado que esta prueba no identifica en dónde se producen las diferencias, es necesario utilizar algún test post-hoc para ello. En este trabajo se

utilizó el test *Tukey-Kramer* para encontrar cuáles medianas son significativamente diferentes entre sí y a partir de allí concluir si hay algún algoritmo mejor que otro.

	f_1	f_2	f_3	f_4	f_5	f_6
B1	$4{,}75e^{+01}$	$7{,}09e^{+01}$	$2{,}71e^{+06}$	$3{,}43e^{+02}$	1,48	$2{,}05e^{+01}$
B2	$5{,}75e^{+02}$	$7{,}32e^{+01}$	$4{,}95e^{+07}$	$3{,}66e^{+02}$	5,95	$1{,}71e^{+01}$
B3	$3{,}18e^{+04}$	$1{,}12e^{+02}$	$1,40e^{+10}$	$4{,}85e^{+02}$	$2{,}78e^{+02}$	$2{,}01e^{+01}$
B4	$6e^{-02}$	$5{,}96e^{+01}$	$3{,}31e^{+01}$	$2{,}65e^{+02}$	$4e^{-02}$	$2{,}05e^{+01}$
B5	$4,05e^{-05}$	$5{,}35e^{+01}$	$5{,}25e^{+02}$	$2{,}31e^{+02}$	$5,40e^{-05}$	$2{,}11e^{+01}$
BCte	0	$4{,}79e^{+02}$	$1{,}47e^{+02}$	$7{,}26e^{+02}$	0	$3,20e^{-05}$
R1	0	$2{,}34e^{+01}$	$2,10e^{+02}$	7,24	0	$2{,}06e^{+02}$
R2	0	$2{,}17e^{+01}$	$1{,}49e^{+02}$	6,02	0	$2{,}11e^{+01}$
R3	$4{,}93e^{+05}$	$1{,}54e^{+02}$	$4,05e^{+11}$	$2{,}25e^{+03}$	$4{,}18e^{+03}$	$2{,}11e^{+01}$
R4	0	$2{,}06e^{+01}$	$1,69e^{+02}$	9,64	0	$2{,}11e^{+01}$
R5	0	$1{,}55e^{+01}$	$9{,}68e^{+01}$	$1{,}49e^{+02}$	0	$2{,}12e^{+01}$
RCte	$3{,}30e^{+05}$	$1{,}50e^{+02}$	$2,82e^{+11}$	$1{,}84e^{+03}$	$2{,}97e^{+03}$	$2{,}12e^{+02}$

Cuadro 1. Error medio de todos los algoritmos para $D = 100$.

Las muestras comparadas son los resultados obtenidos al aplicar cada variante del DE a cada función f_i. Primero se compararon sólo aquellas variantes con estrategia B, luego todas aquellas con estrategia R, y finalmente todas las variantes (tanto con estrategia B como R). Para realizar un estudio sobre muestras `no apareadas` se aplicaron los tests a cada función f_i por separado [1]. Las conclusiones obtenidas a partir de los tests antes mencionados se muestran en el cuadro 2. Para todos los test, de aquí en adelante, se ha considerado tener un 95 % de confianza en las conclusiones obtenidas.

En base al cuadro 2 se puede decir que, teniendo en cuenta los algoritmos con estrategia B, el algoritmo BCte es mejor para todas las funciones, excepto para f_4 donde es el peor. Pero sólo se puede decir esto sin considerar a B5, con el cual no se encontraron diferencias significativas en la mayoría de las funciones.

Considerando los algoritmos con estrategia R, los algoritmos R1, R2, R4 y R5 muestran muy buen desempeño y el o los mejores de cada función se encuentra/n entre ellos. R3 y/o RCte demuestran ser los de peor desempeño. Sólo se pudo demostrar diferencias significativas entre el primer grupo (el de los mejores) y el segundo (el de los peores).

Comparando todas las variantes del DE podemos decir que BCte, R1, R2, R4 y R5 tienen un mejor desempeño que B1, B2, B3, B4, R3 y RCte. Por otro lado R3 y/o RCte son los peores para casi todas las funciones.

Dado que es necesario comparar las variantes del algoritmo DE sobre el conjunto total de las 6 funciones f_i, se realizaron los tests de *Friedman* [3,4], *Friedman Alineado* [6] (F.Alineado) y *Quade* [8] para comparaciones múltiples `apareadas`, no paramétricas.

Estos tests detectan diferencias significativas entre el conjunto total de algoritmos pero sin detectar entre cuáles de ellos se producen. Para ello se aplicaron algunos de los tests post-hoc sugeridos en [5] los cuales toman como algoritmo

[1] Para los tests Kruskal-Wallis y Tukey-Kramer se utilizó la herramienta MATLAB

de *control* al primero en el ordenamiento de los tests antes mencionados para poder concluir que efectivamente es mejor que el resto si se hubieran detectado diferencias significativas.

f_i	Est	p-value	Resultados
f_1	B	$6,75620e^{-36}$	**BCte** es mejor que B1, B2, B3, B4./ B3 es peor que B1, B4, B5, B6
	R	$6,66797e^{-36}$	**R1**, **R2**, **R4** y **R5** son los mejores./ R3 y RCte son los peores.
	BR	$9,88082e^{-70}$	**BCte**, **R1**, **R2**, **R4** y **R5** son los mejores.
f_2	B	$1,45500e^{-31}$	**BCte** es el mejor.
	R	$8,29583e^{-26}$	**R5** es el mejor.
	BR	$4,13644e^{-66}$	**R1**, **R2**, **R4** y **R5** son los mejores.
f_3	B	$4,25961e^{-35}$	**BCte** es mejor que B1, B2, B3 y B4.
	R	$7,21483e^{-29}$	**R5** es mejor que R1, R3 y RCte./ R3 y RCte son los peores.
	BR	$1,03206e^{-64}$	**BCte**, **R2** y **R5** son mejores que B1, B2, B3, B4, R3 y RCte.
f_4	B	$2,03994e^{-28}$	**B5** es mejor que B1, B2, B3 y BCte./ BCte es peor que B1, B2, B4 y B5
	R	$3,92147e^{-33}$	**R1** y **R2** son mejores que R3, R5 y RCte./ R3 es peor que R1, R2, R4 y R5.
	BR	$3,75781e^{-67}$	**R1**, **R2** y **R4** son los mejores./ R3 es el peor.
f_5	B	$1,47982e^{-35}$	**BCte** es mejor que B1, B2, B3 y B4./ B3 es peor que B1, B4, B5 y BCte
	R	$2,78199e^{-36}$	R3 y RCte son peores que R1, R2, R4 y R5.
	BR	$2,28013e^{-69}$	**R1**, **R2**, **R4** y **R5** son los mejores./ R3 es el peor.
f_6	B	$1,46614e^{-24}$	**BCte** es mejor que B1, B3, B4 y B5./ B5 es el peor.
	R	$3,98513e^{-21}$	**R1** es el mejor.
	BR	$1,53811e^{-52}$	**BCte** es mejor que B1, B4, B5, R1, R2, R3, R4, R5 y RCte.

Cuadro 2. Resultados de los tests Kruskal-Wallis y Tukey-Kramer

Al igual que el estudio previo realizado a los algoritmos, se consideraron primeros los algoritmos con estrategia B, luego aquellos con estrategia R, y finalmente se consideraron todas las variantes, de ambas estrategias.

Test	p-value
Friedman	$0,013892$
F. Alineado	$0,474194$
Quade	$0,0002845262$

Cuadro 3. Estrategia B

Test	p-value
Friedman	$0,005608$
F. Alineado	$0,427357$
Quade	$0,0001152417$

Cuadro 4. Estrategia R

Test	p-value
Friedman	$0,0000380084$
F. Alineado	$0,936611$
Quade	$0,0000000015243$

Cuadro 5. Estrat. B y R

Algoritmos con estrategia DE/best/1/bin

Considerando los algoritmos y todas las funciones, los test Friedman y Quade encontraron diferencias significativas. El cuadro 3 muestra los p-values obtenidos con los tres tests.

En base a los ranking obtenidos con los tests (Cuadro 6) se encontró que BCte es el mejor algoritmo según los tests de Friedman y Quade. Se encontraron además diferencias significativas de éste con B3, y con menos fuerza con B2 (según test post-hoc de Holm).

Algoritmos	Friedman	Friedman Alineado	Quade
B1	4.0	16.166	3.857
B2	4.333	17.166	4.666
B3	5.333	32.166	5.666
B4	2.999	15.0	2.857
B5	2.5	14.5	2.0
BCte	1.833	16.0	1.952

Cuadro 6. Ranking de los algoritmos con estrategia B.

Algoritmos con estrategia DE/rand/1/bin

Considerando los algoritmos con estrategia R, los test Friedman y Quade encontraron diferencias significativas entre los algoritmos (Cuadro 4).

Algoritmos	Friedman	Friedman Alineado	Quade
R1	2.666	12.666	2.928
R2	2.166	12.166	2.166
R3	5.5	30.5	5.857
R4	2.833	12.833	2.738
R5	2.833	12.833	2.309
RCte	5.0	30.0	5.0

Cuadro 7. Ranking de los algoritmos con estrategia R.

Se observó que R2 es el mejor algoritmo (Cuadro 7) y presenta diferencias significativas con R3 para varios tests post-hoc (incluyendo el de Holm). Otros tests post-hoc mostraron diferencias también con respecto a RCte y R4.

Todos los algoritmos

Comparando todas las variantes del DE propuestas, los tests Friedman y Quade encontraron diferencias significativas entre los algoritmos (Cuadro 5). R2 demostró ser el de mejor rendimiento para los tests de Friedman y Friedman alineado (Cuadro 8). Este algoritmo presenta fuertes diferencias con relación a R3, RCte, y con menos fuerza con B3. Considerando la dificultad de los problemas, R5 resultó ser el mejor algoritmo según el test Quade, presentando diferencias con respecto a R3.

De todos los estudios previos, se puede concluir que un F variable es mejor cuando la estrategia adoptada es R. Sin embargo, para la estrategia B lo mejor es optar por un F constante, en este caso $F = 0{,}75$ (buen desempeño para todo el benchmark excepto para f_4).

Considerando el conjunto completo de algoritmos, R2 demostró ser mejor que los algoritmos R3, RCte y B3, por lo que se la considera una buena opción para tratar diversos problemas en general. En el algoritmo R2 los valores de F descienden muy rápido y se mantienen en valores cercanos a 0 a partir de la mitad de la ejecución. Otro algoritmo con buen desempeño es el R5 que usa la variación $F_5(t)$ que se comporta como una función gaussiana manteniendo

valores altos de F por una buena cantidad de tiempo, y decreciendo de manera progresiva después de la mitad de la ejecución sin llegar a $F = 0$.

Tanto para R y B, $F_3(t)$ mostró el peor de los desempeños, descartando la idea de una variación de F proporcional al tiempo.

Algoritmos	Friedman	Friedman Alineado	Quade
B1	7.333	33.5	7.714
B2	7.666	33.833	8.523
B3	8.666	35.5	9.523
B4	6.333	30.666	6.714
B5	6	30.333	5.904
BCte	4.0	31.833	3.809
R1	3.833	27.833	3.666
R2	3.5	27.5	2.952
R3	11.5	65.666	11.857
R4	4.166	28.166	3.524
R5	4.0	28.0	2.809
RCte	11.0	65.166	11.0

Cuadro 8. Ranking de todos los algoritmos.

4.2. Escalabilidad

Como se ha mencionado, las dificultades para converger al óptimo global se dan, por lo general, en problemas de alta dimensionalidad. Por este motivo se realizó un estudio adicional para observar la escalabilidad de los algoritmos. Se ejecutaron todas las variantes sobre el mismo benchmark pero para las dimensiones $D = 200$ y $D = 500$. Las comparaciones realizadas entre los mejores algoritmos se encuentran sintetizados en el Cuadro 10.

Se puede observar que para $D = 200$, en f_1, f_5 y f_6 los errores se mantienen bajos, lo que muestra una buena escalabilidad. Sin embargo, para $D = 500$ las soluciones para f_i con $1 \leq i \leq 4$, el error crece significativamente. Aunque sucede algo completamente diferente con f_6 donde los errores disminuyen o se mantienen cercanos a medida que se aumenta la dimensionalidad.

Cuando se comparan los mejores algoritmos entre sí, vemos que BCte alcanza errores muy por encima de los otros algoritmos, a medida que se aumenta la dimensionalidad. Otra observación respecto de BCte, considerando solo los algoritmos con estrategia B, es que conforme aumenta la dimensión en f_4 (función para la cual BCte era el peor algoritmo) comienza a comportarse mejor que otros algoritmos dejando de ser el peor y acercándose a los valores de B5 (el mejor algoritmo para f_4 en todas las dimensiones probadas).

En síntesis, se puede decir que R5 logra el mejor comportamiento en casi todas las funciones, logrando una buena escalabilidad con $D = 200$ en f_1, f_5 y f_6. Además escala bien para $D = 500$ en las funciones f_5 y f_6. En el resto de las funciones, excepto para f_4, R5 es el que obtiene el mínimo error en relación a los otros algoritmos.

		D=100	D=200	D=500
f_1	Bcte	0	$7{,}98e^{-02}$	$8{,}43e^{+03}$
	R2	0	$2e^{-06}$	$5{,}44e^{+02}$
	R5	0	0	$3{,}88e^{+02}$
f_2	Bcte	$4{,}79e^{+01}$	$8{,}77e^{+01}$	$1{,}09e^{+02}$
	R2	$2{,}17e^{+01}$	$7{,}06e^{+01}$	$1{,}07e^{+02}$
	R5	$1{,}56e^{+01}$	$6{,}41e^{+01}$	$1{,}06e^{+02}$
f_3	Bcte	$1{,}47e^{+02}$	$7{,}78e^{+03}$	$1{,}63e^{+09}$
	R2	$1{,}49e^{+02}$	$1{,}84e^{+03}$	$3{,}36e^{+08}$
	R5	$9{,}68e^{+02}$	$8{,}51e^{+02}$	$2{,}53e^{+08}$
f_4	Bcte	$7{,}26e^{+02}$	$1{,}55e^{+03}$	$4e^{+03}$
	R2	6,02	$3{,}94e^{+01}$	$3{,}35e^{+02}$
	R5	$1{,}49e^{+02}$	$5{,}67e^{+02}$	$1{,}76e^{+03}$
f_5	Bcte	0	$2{,}19e^{-02}$	$6{,}98e^{+01}$
	R2	0	0	6,99
	R5	0	0	3,25
f_6	Bcte	$3,20e^{-05}$	1,04	9,61
	R2	$2{,}11e^{+01}$	$1{,}36e^{-03}$	2,19
	R5	$2{,}12e^{+01}$	$2{,}14e^{+01}$	1,94

Cuadro 9. Escalabilidad de BCte, R2 Y R5. Se muestra el error promedio.

5. Conclusiones

Del estudio realizado, se concluye que DE con F constante es mejor que uno variable bajo la estrategia B. Por el contrario, usando la estrategia R, un F variable obtiene mejores resultados. Más precisamente, esto ocurre cuando $F(t)$ corresponde a una función que promueve la exploración durante bastante tiempo para luego disminuir progresivamente (promoviendo explotación) en etapas avanzadas de la búsqueda.

Examinando los algoritmos con estrategia B, se concluye que BCte logró un mejor comportamiento. Para el caso de la estrategia R, los mejores algoritmos fueron R2 y R5. Comparando entre sí a los mejores algoritmos de cada estrategia del DE, se observó que BCte es el algoritmo que peor escala. Por otra parte, R5 mostró mejores capacidades en cuanto a obtener soluciones cercanas al óptimo global. Por lo tanto, se puede decir, que un algoritmo de Evolución Diferencial con estrategia R (DE/rand/1/bin) y factor de mutación variable $F_5(t) = e^{\frac{t^2}{10000000}}$ mejora notablemente su desempeño global. Así, la elección de un factor F variable cuyo comportamiento está dado por una función de Gauss con una desviación estándar en relación al máximo de iteraciones ($\sigma = MaxIte * 1000$), mejora la convergencia para dimensiones altas, y provee un mejor rendimiento que un F constante y uno con decremento lineal.

Futuros trabajos, involucran el diseño de DE con incorporación de buscadores locales y el estudio de benchmarks extendidos junto con la comparación con algoritmos del estado del arte.

Referencias

1. J. Brest, V. Zumer, and M. S. Maucec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 215–222, 2006.
2. S. Das, A. Konar, and U. K. Chakraborty. Two improved differential evolution schemes for faster global search. In *ACM-SIGEVO GECCO*, pages 991–998, 2005.
3. M. Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.
4. M. Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.
5. S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.*, 180(10):2044–2064, May 2010.
6. J. Hodges and E. Lehmann. Rank methods for combination of independent experiments in analysis of variance. *Ann. Math. Statist*, 33:482–497, 1962.
7. P. Kaelo and M. Ali. A numerical study of some modified differential evolution algorithms. *European Journal of Operational Research*, 169(3):1176–1184, 2006.
8. D. Quade. Using weighted rankings in the analysis of complete blocks with additive block effects. *Journal of the American Statistical Association*, 74(367):pp. 680–683, 1979.
9. R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, Dec. 1997.
10. K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark functions for the CEC 2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.
11. R. Thangaraj, M. Pant, and A. Abraham. New mutation schemes for differential evolution algorithm and their application to the optimization of directional overcurrent relay settings. *Applied Mathematics and Computation*, 216(2):532–544, 2010.
12. J. Zhang and A. Sanderson. *Adaptive Differential Evolution: A Robust Approach to Multimodal Problem Optimization*, chapter Theoretical Analysis of Differential Evolution.