Comparación del desempeño entre los algoritmos VMO, DE y ODE

Ernesto Díaz López[1], Amilkar Puris [2], Rafael Bello [3]

Resumen--**La Optimización Basada en Mallas Variables (*Variable Mesh Optimization*; VMO) es una meta-heurística poblacional con características evolutivas donde un conjunto de nodos que representan soluciones potenciales a un problema de optimización, forman una malla (población) que crece dinámicamente y se desplaza por el espacio de búsqueda. En este trabajo se establece una comparación del desempeño entre los algoritmos *Differential Evolution* (DE), *Opposition-based Differential Evolution* (ODE) y VMO para la que se utilizaron siete funciones (*F1 – F7*) propuestas en la sesión especial de CEC-2008. Resulta interesante destacar que VMO resultó superior a ambos algoritmos en las funciones *F1 y F3* para dimensión 500 y en *F1, F3, F5 y F7* para dimensión 1000, por lo que se recomienda probar a VMO en espacios de búsquedas más complejos. Como parte del trabajo se muestran los detalles de la metodología empleada en la comparación, así como los resultados obtenidos.**

Palabras clave--**Optimización Basada en Mallas Variables (VMO), Opposition-based Differential Evolution (ODE), Differential Evolution (DE), Problemas de Optimización de grandes dimensiones.**

I. INTRODUCCIÓN

Los Algoritmos Evolutivos (AE) [1] son métodos de optimización y búsqueda de soluciones inspirados por los procesos de la propia naturaleza, donde la evolución de las especies se produce por tres conceptos: replicación, variación y selección natural. La población en este tipo de algoritmos representa un conjunto de individuos, los cuales se mantienen mediante replicación. La variación es la encargada de introducir diversidad en la población, mediante una serie de diferencias entre los descendientes y sus progenitores. Por esta propia naturaleza, los AE resultan costosos desde el punto de vista computacional, siendo lentos en problemas de optimización de grandes dimensiones. Esto ocurre debido, principalmente, al aumento de las dimensiones del espacio de búsqueda. No obstante a esta dificultad y debido a que múltiples problemas en los campos de las ciencias y la ingeniería contienen una gran cantidad de variables, resulta de vital importancia trabajar con métodos escalables para dar solución a los problemas de optimización.

Uno de los representantes de los algoritmos evolutivos es el *Diferential Evolution* (DE) [2, 3]. Sin embargo, estudios recientes [4] demuestran que el algoritmo *Opposition-based Differential Evolution* (ODE) [5] obtiene un mejor desempeño. Por otra parte, la Optimización basada en Mallas Variables (VMO) [6] ofrece resultados en algunos casos superiores y en múltiples evaluaciones similares que los algoritmos mencionados anteriormente, cuando la dimensión toma valores de 500 y 1000.

VMO emplea una malla (población) que crece dinámicamente y se desplaza por el espacio de búsqueda, es decir, evoluciona. Para ello, se realiza un proceso de expansión en cada ciclo, donde se generan nuevos nodos en dirección a los extremos locales y al extremo global; así como a partir de los nodos fronteras de la malla. Luego se realiza un proceso de contracción de la malla, donde los mejores nodos resultantes en cada iteración son seleccionados como malla inicial para la iteración siguiente.

Los resultados reportados en [6] fueron prometedores para problemas de tamaño pequeño ($D < 50$), concluyendo que el algoritmo fue más competitivo a medida que aumentaba la dimensión de las funciones de prueba. Mediante una verificación experimental el trabajo que se presenta trata de encontrar una respuesta a la pregunta: ¿cuál de los algoritmos (DE, ODE o VMO), presenta mayor calidad en los resultados al resolver problemas de grandes dimensiones?

El trabajo se organiza como sigue: el epígrafe II proporciona una breve descripción del algoritmo VMO. En el epígrafe III se explican los resultados experimentales y se realiza un análisis del desempeño. Finalmente, se establecen las conclusiones en el epígrafe IV.

II. DESCRIPCIÓN DEL ALGORITMO

La esencia del método VMO es crear una malla de puntos en el espacio m dimensional, donde se realiza el proceso de optimización de una función FO(x1, x2,…, xm); la cual se mueve mediante un proceso de expansión hacia otras regiones del espacio de búsqueda. Dicha malla se hace más "fina" en aquellas zonas que parecen ser más promisorias. Es dinámica en el sentido que la malla cambia su tamaño y configuración durante el proceso de búsqueda. Los nodos se representan como vectores de la forma n(x1, x2,…, xm). El

[1] e-mail: ediaz@uclv.edu.cu
[2] e-mail: ayudier@uclv.edu.cu
[3] e-mail: rbellop@uclv.edu.cu

proceso de generación de nodos en cada ciclo comprende los pasos siguientes:

1. Generación de la malla inicial.
2. Generación de nodos en dirección a los extremos locales (*nl*).
3. Generación de nodos en dirección al extremo global (*ng*).
4. Generación de nodos a partir de las fronteras de la malla (*nf*).

El método incluye los parámetros:

- Cantidad de nodos de la malla inicial (Ni).
- Cantidad máxima de nodos de la malla en cada ciclo (N), donde 3•Ni ≤ N.
- Tamaño de la vecindad (k).
- Condición de parada (M).

A. Generación de la malla inicial

La malla inicial consta de *Ni* nodos, los que en la primera iteración son generados de forma aleatoria. En las restantes iteraciones se realiza un proceso de contracción de la malla, que se basa en una selección de los nodos con mejor calidad entre los nodos (*N* nodos) existentes al final de cada iteración.

B. Generación de nodos en dirección a los extremos locales

En esta etapa se realiza una exploración en las vecindades de cada uno de los nodos de la malla inicial. Para esto, se buscan los vecinos más cercanos de cada nodo n a través de una función de distancia o semejanza (ej Distancia Euclidiana). Luego, se selecciona cuál de los vecinos tiene mejor calidad (evaluación de la función *FO*) que el nodo actual (denotándose ese mejor nodo por *nl*). Si ninguno de los vecinos es mejor, entonces este se considera un extremo local y no se generan nodos a partir de él en este paso. En otro caso, se genera un nodo (n^*) que estará situado entre el nodo (n) y el extremo local (*nl*). Los valores de las componentes del nuevo nodo se calculan usando la ecuación (1):

$$n^*(i) = f(n(i), nl(i), r) \quad (1)$$

En esta ecuación, i representa el i-ésimo componente de cada nodo. La cercanía del nuevo nodo al actual o al extremo local, depende de un factor (*r)* calculado en base a los valores que alcanza la función *FO* en cada uno de los nodos involucrados. Mientras mayor sea la diferencia entre los valores de la función *FO* en los nodos involucrados, mayor será la cercanía o semejanza de n^* a *nl*, esto lo garantiza el factor *r*.

C. Generación de nodos en dirección hacia el extremo global

En este caso se realiza una exploración global hacia el nodo que mejor calidad ha tenido hasta el momento (extremo global, *ng*). Con este fin se generan nuevos nodos a partir de cada nodo de la malla inicial en dirección a este, utilizando la ecuación (2):

$$n^*(i) = g(n(i), ng(i), r) \quad (2)$$

Este paso es el encargado de acelerar la convergencia del método.

D. Generación de nodos a partir de los nodos más externos de la malla

Este proceso de generación de nuevos nodos tiene lugar con el objetivo de explorar el espacio de búsqueda en dirección a las fronteras de cada dimensión. Para ello, se seleccionan los nodos cuyas posiciones se encuentran en los extremos de la malla inicial (nodos fronteras). El proceso de detección de este tipo de nodo (nf) se realiza siguiendo algún criterio en dependencia del espacio solución. La generación de los nuevos nodos se obtiene a través de la ecuación (3):

$$n^*(i) = h(nf(i), w) \quad (3)$$

El parámetro w tiene como objetivo desplazar los nodos fronteras en dirección a los puntos más y menos externos del espacio de soluciones y se calcula utilizando la expresión (4) tal como aparece en [7, 8].

$$w = (w_0 - 0.4) \times \frac{M-j}{M+0.4} \quad (4)$$

Donde w_0 = 1.4 es el peso inicial, j es el número de la iteración actual y M es el número máximo de evaluaciones de la función objetivo.

E. Diversidad

VMO presenta un mecanismo para fomentar la diversidad, que se basa en mantener una cierta separabilidad entre cada nodo de la malla inicial. Para ello, se propone un operador de limpieza adaptativo, que funciona de la siguiente manera:

Se ordenan todos los nodos seleccionados como malla inicial en función de su calidad. De forma secuencial, se compara cada nodo de la malla con sus sucesores, eliminando aquellos cuya distancia espacial sea menor que una cota calculada dinámicamente. Este valor de la distancia debe permitir que el proceso sea decreciente; de manera

que se obtenga mayor separabilidad entre los nodos al inicio que al final de la ejecución del método.

El valor adaptativo de la cota de distancia (cd) le permite al método comenzar con exploraciones más generales y luego ir disminuyendo su influencia hasta centrarse en una zona más pequeña del espacio de búsqueda. Procedimiento similar fue propuesto en [9].

A continuación se muestra el esquema general del algoritmo VMO

Algoritmo 1 VMO

Generar la malla inicial (*Ni*) de forma aleatoria.
Evaluar los nodos de la malla inicial, seleccionar el mejor *ng*.
Repetir:
  Para cada nodo en la malla inicial del ciclo hacer
    Encontrar sus k nodos más cercanos
    Determinar el mejor de los vecinos *ne*
    Si *ne* es mejor que el nodo actual, entonces
      Generar nuevo nodo usando ecuación (1)
    Fin Si
  Fin Para
  Para cada nodo en la malla inicial del ciclo hacer
    Generar un nuevo nodo usando la ecuación (2)
  Fin Para
  Seleccionar los nodos más externos de la malla
  Generar nuevo nodo usando la ecuación (3)
  Mientras los *N* nodos de la malla en el ciclo actual no se hayan generado
    Generar nuevo nodo aleatoriamente
  Construir la malla inicial actual de manera elitista
  Ordenar los nodos de la malla inicial según su calidad
Aplicar operador de limpieza adaptativo.
Hasta *M* evaluaciones

F. VMO en detalles

A continuación se describe el algoritmo VMO más detalladamente.

Generación aleatoria de la malla inicial: para cada uno de los nodos de la malla inicial se valoriza aleatoriamente cada dimensión, con un valor real entre el intervalo definido en cada caso.

Generación de nodos en dirección a los extremos locales: para calcular los vecinos más cercanos de cada nodo de la malla se utiliza como función de distancia la euclidiana, definida por:

$$D_{euclidiana}(n1, n2) = \sqrt[2]{\sum_{i=1}^{n}(n_1(i) - n_2(i))^2} \quad (5)$$

El factor (r), que determina la cercanía del nuevo nodo al nodo actual o al extremo local, se calcula usando la ecuación siguiente:

$$r = \frac{1}{1+|FO(n)-FO(ne)|} \quad (6)$$

La función f para la generación de nuevos nodos a partir de cada nodo que no sea extremo local y el mejor vecino, se define por la ecuación (7):

$$n^*(i) = \begin{cases} vm(i), & si\ |vm(i) - nl(i)| > cd\ y\ va[0,1] \le r \\ nl(i) + va[-cd, cd], & si\ [vm(i) - nl(i) \le cd] \\ va[vm(i), nl(i)], & en\ otro\ caso \end{cases} \quad (7)$$

donde *vm*(i) representa el valor medio entre el nodo actual y el extremo local para la i-ésima dimensión y se calcula como:

$$vm(i) = \frac{n(i)+nl(i)}{2} \quad (8)$$

Además, *va*[a, b] denota un valor aleatorio en el intervalo [a, b] y cd es una cota de distancia adaptativa y se calcula según la ecuación (9)

$$cd = \begin{cases} \left|\frac{\inf + sup}{4}\right|, & si\ j < 15\%\ M \\ \left|\frac{\inf + sup}{8}\right|, & si\ 15\%\ M \le j < 30\% M \\ \left|\frac{\inf + sup}{16}\right|, & si\ 30\%\ M \le j < 60\% M \\ \left|\frac{\inf + sup}{50}\right|, & si\ 60\%\ M \le j < 80\% M \\ \left|\frac{\inf + sup}{100}\right|, & si\ j \ge 80\%\ M \end{cases} \quad (9)$$

donde M denota el valor máximo de evaluaciones de la función objetivo y j la evaluación actual. En dependencia del por ciento del total que represente la evaluación actual, se definen valores de distancia que representan partes del intervalo permitido.

Generación de nodos en dirección al extremo global: en este paso de la generación se crean nuevos nodos a partir de cada nodo de la malla inicial en dirección al extremo global. Para esto, se utiliza el valor de *r* calculado por la ecuación (6) sustituyendo *nl* por *ng*. Luego se define la función de generación g:

$$n^*(i) = \begin{cases} vm(i), & si\ va[0,1] \le r \\ va[vm(i), ng(i)] & en\ otro\ caso \end{cases} \quad (10)$$

Donde el valor medio (*vm*) entre el nodo actual y el extremo global se calcula utilizando la ecuación 8.
Según la función g, si existe una gran diferencia entre la calidad del extremo global y el nodo actual, hay mayor probabilidad de que la i-ésima componente tome valores más cercanos a ng, en caso contrario se valoriza con el valor medio.

Generación de nodos a partir de los más externos de la malla: en este paso se completa la cantidad total de nodos que debe tener la malla, a partir de los nodos fronteras. Para detectar este tipo de nodos se utiliza, para este caso de estudio, el

valor de la norma de cada uno, definida por la ecuación:

$$\|n\| = \sqrt[2]{\sum_{i=1}^{n}\left(n(i)\right)^2} \quad (11)$$

Los nodos de mayor norma son los que están situados en el contorno (puntos más externos) de la malla inicial y los de menor norma se consideran los nodos que más cerca se encuentran del origen 0^m (puntos más internos). La función h permite generar nuevos nodos en dirección de las fronteras definidas para este caso de estudio, mediante las expresiones:

Para los nodos más externos:

$$n^*(i) = \begin{cases} nl(i) + w, & si\ \ nl(i) > 0 \\ nl(i) - w, & si\ \ nl(i) < 0 \end{cases} \quad (12)$$

Para los nodos más internos:

$$n^*(i) = \begin{cases} |nl(i) - w|, & si\ \ nl(i) > 0 \\ |nl(i) + w|, & si\ \ nl(i) < 0 \end{cases} \quad (13)$$

Donde el desplazamiento w se calcula como:

$$w = \left(w_0 - w_f\right)\left(\frac{M-j}{M} + w_f\right) \quad (14)$$

donde el parámetro M y la variable *j*, están estrechamente relacionados y provocan las variaciones en el valor de *w*; el primero representa el número máximo de evaluaciones de la función objetivo. Por su parte, *j* denota el número actual de evaluaciones de la función objetivo. La variable w_0 representa el desplazamiento inicial y w_f el valor final de este (si $w_0 > w_f$ efecto decreciente). Para obtener desplazamientos decrecientes relacionados con las amplitudes de cada función, se desarrolló una propuesta adaptativa, donde $w_0 = am/10$ y $w_f = am/100$. Por su parte, *am* denota la amplitud media del intervalo permitido para la i-ésima componente y se calcula como:

$$am = \frac{|inf| + |sup|}{2} \quad (15)$$

III. VMO, ODE Y DE COMPARADOS EN PROBLEMAS DE GRANDES DIMENSIONES

Para la comparación de estos algoritmos se utilizan un conjunto de 7 funciones con dimensiones 500 y 1000. En este conjunto existen funciones separables y no separables además de unimodales y multimodales. Todas las funciones han sido tomadas de CEC-2008 (Competencia y Sesión Especial de Optimización en Grandes Dimensiones) [10].

G. Breve descripción de las funciones

Las dos primeras funciones (F_1 y F_2) son unimodales y las restantes (F_3-F_7) multimodales[4]. Según la separabilidad las funciones (F_1, F_4 y F_6) son separables mientras que (F_2, F_3, F_5 y F_7) no lo son. Una función de *n* variables es separable si puede ser escrita como la suma de *n* funciones de sólo una variable [11]. La descripción matemática de las funciones de prueba aparece en el Anexo 1.

H. Parámetros de VMO para las corridas

Para cada una de las 25 corridas se emplearon los siguientes parámetros en el algoritmo VMO:

- Dimensión del problema, *D=500* y *D=1000[10]*
- Total de evaluaciones de la función objetivo *Max_FO= 5000 x D[10]*

Los próximos parámetros se seleccionaron por ser los que mejores resultados produjeron:

- Tamaño Malla Inicial, Tam_*Malla_Inicial=100*
- Tamaño Malla Generada, *Tam_Malla_Generada=* 3*Tam_Malla_Inicial+Tam_Malla_Inicial/2
- Vecindad, *k=3*

I. Análisis de los Resultados

En las Tablas I y II se presentan los resultados comparativos de los algoritmos DE, ODE y VMO. Se realizaron 25 corridas para cada una de las funciones una única vez sin modificar ninguno de los parámetros. Los valores de DE y ODE se tomaron de [4].

Para cada una de las 7 funciones se brinda el mejor, la mediana, el peor, el promedio, la desviación estándar y el 95% del intervalo de confianza (95% IC) del valor del error calculado como $F(x) - F(x^*)$.

De la Tabla I se observa que VMO con Dimensión = 500, es superior a ambos algoritmos (DE y ODE) en las Funciones *F1 y F3*. En *F4 y F5* supera sólo a DE y es inferior a ambos en el resto (*F2, F6 y F7)*.

[4] Una función es unimodal si sólo tiene un óptimo (relativo o absoluto). En caso de que tenga varios óptimos se dice multimodal.

TABLA I

RESULTADOS DE VMO PARA DIMENSIÓN = 500 Y 25 CORRIDAS PARA CADA FUNCIÓN. LOS VALORES DE DE Y ODE FUERON TOMADOS DE [4]. LOS MEJORES RESULTADOS SE RESALTAN EN NEGRITAS Y LOS PEORES SUBRAYADOS

F	Error	DE	ODE	VMO
F1	Mejor	2, 636.54	15.66	**1.47**
	Mediana	3, 181.45	36.61	**4.70**
	Peor	4, 328.80	292.65	**7.04**
	Promedio	3, 266.24	80.17	**4.49**
	Dev. Std	409.68	79.24	**1.56**
F2	Mejor	79.74	**3.60**	94.62
	Mediana	82.39	**4.86**	95.51
	Peor	85.92	**11.91**	118.01
	Promedio	82.93	**5.78**	96.39
	Dev. Std	2.09	**2.37**	4.52
F3	Mejor	76, 615, 772.08	39, 718.90	**838.01**
	Mediana	119, 733, 049.20	137, 279.03	**2352.48**
	Peor	169, 316, 779.50	407, 661.64	**3453.42**
	Promedio	123, 184, 755.70	154, 306.34	**2342.88**
	Dev. Std	29, 956, 737.58	114, 000.53	**609.73**
F4	Mejor	5, 209.99	**2, 543.51**	4,686.92
	Mediana	5, 324.57	**4, 279.56**	5,048.45
	Peor	5, 388.24	**6, 003.94**	5,215.51
	Promedio	5, 332.59	**4, 216.34**	5,153.49
	Dev. Std	**43.82**	1, 017.94	312.82
F5	Mejor	24.29	**1.25**	15.90
	Mediana	24.71	**1.55**	19.93
	Peor	27.59	**2.13**	27.46
	Promedio	25.16	**1.75**	20.77
	Dev. Std	1.10	**0.37**	2.85
F6	Mejor	*4.66*	**2.49**	4.58
	Mediana	4.97	**4.12**	14.65
	Peor	**5.15**	6.73	*20.80*
	Promedio	4.94	**4.51**	15.91
	Dev. Std	**0.17**	1.44	4.10
F7	Mejor	−6, 764.16	**−7, 326.71**	-4,528.27
	Mediana	−6, 705.17	**−7, 290.84**	-4,059.40
	Peor	−6, 692.63	**−7, 103.89**	-3,671.00
	Promedio	−6, 711.71	**−7, 256.45**	-4,048.49
	Dev. Std	21.08	**−87.08**	245.01

Con la Dimensión = 1000 mostrada en la Tabla II los resultados son más satisfactorios. Para esta dimensión VMO resulta el mejor en las funciones *F1, F3, F5 y F7*. Supera a DE en *F2* y *F4* y sólo cede ante ambos en *F6,* aunque para esta función VMO reporta la menor desviación estándar.

TABLA II

RESULTADOS DE VMO PARA DIMENSIÓN = 1000 Y 25 CORRIDAS PARA CADA FUNCIÓN. LOS VALORES DE DE Y ODE FUERON TOMADOS DE [4]. LOS MEJORES RESULTADOS SE RESALTAN EN NEGRITAS Y LOS PEORES SUBRAYADOS

F	Error	DE	ODE	VMO
F1	Mejor	223, 944.73	19, 548.90	**61.99**
	Mediana	236, 805.13	21, 104.67	**70.81**
	Peor	258, 806.47	43, 417.84	**99.22**
	Promedio	238, 923.73	23, 903.98	**73.50**
	Dev. Std	12, 141.16	8, 009.82	**8.85**
F2	Mejor	119.57	**0.44**	97.36
	Mediana	121.68	**0.77**	97.99
	Peor	123.11	**2.88**	98.39
	Promedio	121.33	**1.31**	97.98
	Dev. Std	1.05	0.94	**0.21**
F3	Mejor	35, 743, 891, 601	213, 105, 668	**29,220.80**
	Mediana	40, 519, 312, 742	572, 841, 466	**265,630.76**
	Peor	44, 559, 917, 677	1, 069, 602, 053	**1,192,040.61**
	Promedio	40, 215, 461, 419	516, 296, 792	**392,925.34**
	Dev. Std	2, 838, 193, 442	326, 088, 526	**355,771.56**
F4	Mejor	11, 589.29	**5, 704.55**	9,962.21
	Mediana	11, 791.33	**5, 904.49**	10,815.28
	Peor	11, 898.50	**7, 309.99**	11,962.96
	Promedio	11, 782.84	**6, 168.10**	10,886.24
	Dev. Std	105.06	*620.58*	**493.78**
F5	Mejor	1, 845.36	138.99	**71.45**
	Mediana	2, 040.99	182.19	**85.94**
	Peor	2, 101.89	185.05	**102.56**
	Promedio	2, 016.90	179.01	**86.26**
	Dev. Std	79.15	14.13	**7.46**
F6	Mejor	14.80	**10.07**	20.15
	Mediana	15.14	**12.64**	20.97
	Peor	15.51	**13.40**	21.00
	Promedio	15.13	**12.14**	20.93
	Dev. Std	0.23	1.14	**0.16**
F7	Mejor	−6, 764.16	−7, 326.71	**-10,377.62**
	Mediana	−6, 705.17	-7, 290.84	**-7,773.72**
	Peor	−6, 692.63	−7, 103.89	**-7,284.87**
	Promedio	−6, 711.71	−7, 256.45	**-8,285.34**
	Dev. Std	**21.08**	87.08	1,068.28

J. Análisis Estadístico

Con el objetivo de determinar diferencias significativas entre los resultados del algoritmo VMO propuesto, para una dimensión de 1000 y los algoritmos DE y ODE se aplicó un test de Iman-Davenport. El test encuentra diferencias significativas en el grupo de algoritmos por lo que la hipótesis de igualdad es rechazada tal como se observa en la Tabla III.

TABLA III

RESULTADOS DEL TEST DE IMAN-DAVENPORT

VALOR DEL TEST	VALOR P	HIPÓTESIS
5.588235	0.023481	R

Posteriormente, para detectar específicamente entre qué algoritmos existen diferencias

significativas se aplica un test de comparaciones múltiples de Holm, donde se utiliza como muestra de control el algoritmo VMO.

TABLA IV
RESULTADOS DEL TEST DE HOLM, VMO ALGORITMO DE CONTROL

ALGORITMO	Z	*p*	α/i	HIP.
DE	2.020725	0.043308	0.05	R
ODE	0.288675	0.772829	0.1	A

Como se aprecia en la Tabla IV, la hipótesis de igualdad es rechazada para el algoritmo DE, no así para el algoritmo ODE donde es aceptada. Esto se debe a que en el primer caso el valor de *p* es inferior al valor de α/*i* y superior en el segundo.

De esta manera se puede concluir que el algoritmo VMO tiene un comportamiento similar al algoritmo ODE y que ambos superan al algoritmo DE.

Para dimensión 500 se realizó un análisis estadístico análogo al anterior y los resultados alcanzados fueron similares.

IV. CONCLUSIONES

El algoritmo VMO presentó resultados alentadores en dimensiones pequeñas (D=10, 30 y 50) pero no había sido probado en dimensiones muy superiores. Los resultados de este estudio evidencian cómo la precisión del algoritmo mejora con el aumento de la dimensión del problema. Con dimensión 500, VMO fue superior en *F1* y *F3* y para dimensión 1000 resultó el mejor en *F1, F3, F5 y F7*. Las pruebas estadísticas evidencian un comportamiento similar entre los algoritmos ODE y VMO, presentando ambos un mejor desempeño que el algoritmo DE.

Resulta de interés seguir trabajando en variantes de este algoritmo así como el estudio de versiones paralelas del mismo, lo que será objetivo de próximos trabajos.

REFERENCIAS

[1] Bäck, T., U. Hammel, and H. Schwefel, *Evolutionary Computation: Comments on the History and Current State.* 1997: Estados Unidos. p. 3-17.

[2] Storn, R. and K. Price, *Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces*, in *Tech. Rep. TR-95-012*. 1995, Berkeley,CA.

[3] Rainer, S. and P. Kenneth, *Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*, in *Journal of Global Optimization*. 1997, Kluwer Academic Publishers: Netherlands. p. 341-359.

[4] Rahnamayan, S. and G.G. Wang, *Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE).* WSEAS TRANSACTIONS on COMPUTERS, 2008.

[5] Rahnamayan, S., H.R. Tizhoosh, and M.M.A. Salama, *Opposition-Based Differential Evolution.* IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 2008. **12**(1).

[6] Puris, A., *Desarrollo de meta-heurísticas poblacionales para la solución de problemas complejos* 2009, Universidad Central "Marta Abreu" de Las Villas: Santa Clara.

[7] Bello, R. and Y. Gómez. *Two-step particle swarm optimization to solve the feature selection problem*. in *Proceedings of the International Symposium on Data Analysis (ISDA)*. 2007. Rio de Janeiro, Brasil.

[8] Wang, X., *Feature selection based on rough sets and particle swarm optimization.* Pattern Recognition Letters **28**, 2007: p. 459-471.

[9] Eshelman, L. *The CHC Adaptive Search Algorithm. How to Have Safe Search When Engaging in Nontraditional Genetic Recombination.* in *Foundations of Genetic Algorithms*. 1991.

[10] Tang, K., et al., *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*, in *Technical Report*. 2007, Nature Inspired Computation and Applications Laboratory, USTC: China.

[11] Hadley, G., *Nonlinear and Dynamics Programming*. World Student. Dec, 1964, Reading, MA, USA: Addison-Wesley Professional.

ANEXO 1 DEFINICIÓN MATEMÁTICA DE LAS FUNCIONES

F_1: Shifted Sphere Function

$$F_1(x) = \sum_{i=1}^{D} z_i^2 + f_{bias_1},$$
$$x \in [-100,100]^D, z = x - o, x = [x_1, x_2, \ldots, x_D]$$
$$Optimo\ Global: x^* = o, F_1(x^*) = f\_bias_1 = -450$$
$$o = [o_1, o_2, \ldots, o_D]: Optimo\ Global\ Desplazado$$

Características: Unimodal, Separable, Escalable, Desplazada

F_2: Schwefel's Problem 2.21

$$F_2(x) = max_i\{|z_i|, 1 \le i \le D\} + f\_bias_2,$$
$$x \in [-100,100]^D, z = x - o, x = [x_1, x_2, \ldots, x_D]$$
$$Optimo\ Global: x^* = o, F_2(x^*) = f\_bias_2 = -450$$
$$o = [o_1, o_2, \ldots, o_D]: Optimo\ Global\ Desplazado$$

Características: Unimodal, No Separable, Escalable, Desplazada

F_3: Shifted Rosenbrock's Function

$$F_3(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f\_bias_3,$$
$$x \in [-100,100]^D, z = x - o + 1, x = [x_1, x_2, \ldots, x_D]$$
$$Optimo\ Global: x^* = o, F_3(x^*) = f\_bias_3 = 390$$
$$o = [o_1, o_2, \ldots, o_D]: Optimo\ Global\ Desplazado$$

Características: Multimodal, No Separable, Escalable, Desplazada

F_4: Shifted Rastrigin's Function

$$F_4(x) = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_4,$$
$$x \in [-5,5]^D, , z = x - o, x = [x_1, x_2, \ldots, x_D]$$
$$Optimo\ Global: x^* = o, F_4(x^*) = f_{bias_4} = -330$$
$$o = [o_1, o_2, \ldots, o_D]: Optimo\ Global\ Desplazado$$

Características: Multimodal, No Separable, Escalable, Desplazada

F_5: Shifted Griewank's Function

$$F_5(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f\_bias_5,$$
$$x \in [-600,600]^D, z = x - o, x = [x_1, x_2, \ldots, x_D]$$
$$Optimo\ Global: x^* = o, F_5(x^*) = f_{\_bias_5} = -180$$
$$o = [o_1, o_2, \ldots, o_D]: Optimo\ Global\ Desplazado$$

Características: Multimodal, No Separable, Escalable, Desplazada

F_6: Shifted Ackley's Function

$$F_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f_{bias_6},$$
$$x \in [-32,32]^D, z = x - o, x = [x_1, x_2, \ldots, x_D]$$
$$Optimo\ Global: x^* = o, F_6(x^*) = f_{\_bias_6} = -140$$
$$o = [o_1, o_2, \ldots, o_D]: Optimo\ Global\ Desplazado$$

Características: Multimodal, Separable, Escalable, Desplazada

F_7: FastFractal “DoubleDip” Function

$$F_7(x) = \sum_{i=1}^{D} fractal1D(x_i + twist(x_{(imodD)+1}))$$

$$twist(y) = 4(y^4 - 2y^3 + y^2)$$

$$fractal1D(x) \approx \sum_{k=1}^{3} \sum_{1}^{2^{k-1}} \sum_{1}^{ran2(o)} doubledip\left(x, ran1(o), \frac{1}{2^{k-1}(2 - ran1(o))}\right)$$

$$doubledip(x, c, s) = \begin{cases} (-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1)s, -0.5 < x < 0.5 \\ 0, \quad en\ otro\ caso \end{cases}$$

$$x = [x_1, x_2, \ldots, x_D]$$

ran1(o): doble, elegido pseudoaleatoriamente con semilla o, con igual probabilidad en el intervalo {0,1}
ran2(o): entero, elegido pseudoaleatoriamente con semilla o, con igual probabilidad en el conjunto {0,1,2}
fractal1D(x) es una aproximación al algoritmo recursivo
x^*=desconocido, $F_7(x^*)$=desconocido
Características: Multimodal, No Separable, Escalable