Alcance de la evolución diferencial en ambientes dinámicos: un análisis empírico

Pavel Novoa-Hernández
David A. Pelta
Carlos Cruz Corona

Resumen— **Un gran número de problemas de optimización reales son dinámicos, lo que significa que algunos del los elementos del modelo varían con el tiempo. Dentro de la Computación Evolutiva, estos problemas han recibido un especial interés en los últimos años con la aplicación de paradigmas computacionales provenientes de la optimización estacionaria. Entre estos paradigmas, la Evolución Diferencial sobresale por su efectividad y fácil implementación. Sin embargo, como la mayoría de los paradigmas aplicados a este contexto, DE ha tenido que ser adaptado con el objetivo resolver principalmente la pérdida de diversidad producto a los cambios del problema. En ese sentido, el presente trabajo investiga el rendimiento de varias extensiones de DE que incluyen algunas de las estrategias existentes más efectivas para tratar esta dificultad. Concretamente, las extensiones son dos algoritmos auto-adaptativos basados en DE, mientras que las estrategias se basan en el conocido algoritmo Multi-enjambre Quantum PSO.**

Palabras clave— **Ambientes dinámicos, Evolución Diferencial, Auto-adaptación**

I. Introducción

Varios son los problemas que, como producto a la complejidad del mundo actual, pueden modelarse como problemas dinámicos de optimización(PDOs). Estos problemas tiene como característica que uno o más elementos del modelo matemático que lo describe varían con el tiempo. Ejemplo de estos elementos dinámicos son la función objetivo, el sistema de restricciones, el número de dimensiones del espacio de búsqueda, entre otros. Sin pérdida de generalidad en el presente trabajo se tratarán PDOs en los que la función objetivo varía en el tiempo. Formalmente, este tipo de problema puede ser definido a través del siguiente modelo:

$$\text{máx } f(\vec{x}, t) \tag{1}$$

donde $f : \Omega \times \mathcal{T} \rightarrow \mathbb{R}$. $\Omega \subseteq \mathbb{R}^D$ es el espacio de búsqueda, mientras que $\mathcal{T} \in \mathbb{Z}^+$ representa el intervalo de tiempo en el que se optimiza la función. De manera que $\vec{x} \in \Omega$, y $t \in \mathcal{T}$.

Debido al importante nivel de incertidumbre de estos problemas, la aplicación de técnicas de optimización provenientes de campos como la Soft Computing parece justificarse. Precisamente, en los últimos años este tema ha recibido un especial interés expresado en el creciente número de trabajos tanto de congresos como de revistas (véase por ejemplo las revisiones [8] y [13]). La tendencia actual en este campo, en relación con la propuesta de nuevos métodos, es el empleo de paradigmas computacionales del campo de la optimización estacionaria que son adaptados producto a los retos que impone la optimización dinámica.

Específicamente, estas adaptaciones tienen como objetivo tratar dos dificultades que pueden aparecer producto a los cambios en el ambiente: la *desactualización de la información* y la *pérdida de diversidad*. La primera ocurre cuando los valores de la función objetivo (fitness) correspondientes a las soluciones del algoritmo quedan obsoletos. Por su parte, la *pérdida de diversidad* aparece cuando el algoritmo queda atrapado en una zona del espacio de búsqueda y es incapaz de localizar al nuevo óptimo del problema. Aunque la primera de estas dificultades puede resolverse relativamente fácil (ej. evaluando nuevamente la población de soluciones), la segunda es más difícil de tratar. En ese sentido, en los últimos años se han propuesto diferentes estrategias que pueden agruparse según [13] en cuatro grupos: 1) diversidad después del cambio, 2) diversidad durante la ejecución, 3) enfoques basados en memorias, y 4) enfoques multipoblacionales.

La Evolución Diferencial (DE) propuesta por Storn y Price [21] es uno de los paradigmas provenientes de la optimización estacionaria que ha sido aplicado con éxito en ambientes dinámicos (véase por ejemplo [6] y [14]). Similarmente a lo que ocurre en otros paradigmas, DE sufre de pérdida de diversidad y en consecuencia tiene que ser adaptado a este contexto. Aunque en la actualidad se han logrado importantes avances en esa dirección, se trata de un tema abierto y de constante investigación. En ese sentido, el presente trabajo analiza de manera empírica el rendimiento de varias extensiones de DE, que incluyen algunas de las estrategias más efectivas para tratar la pérdida de diversidad. Concretamente, el conjunto de extensiones lo integran el esquema original de DE con diferentes estrategias de mutación, y los algoritmos auto-adaptativos jDE [6] y SspDE [17]. Las estrategias para tratar la pérdida de diversidad están basadas en el algoritmo Multienjambre Quantum PSO (mQSO) de Branke y Blackwell [3]. Básicamente nuestra intención es utilizar las extensiones de DE como sub-algoritmos para el en-

Dept. Matemática, Universidad de Holguín, Holguín, Cuba. 80100. E-mail: pnovoa@facinf.uho.edu.cu

Dept. Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, Granada, España. E-mail: {dpelta,carloscruz}@decsai.ugr.es

foque mQSO (ej. como alternativas al Quantum PSO utilizado por este enfoque).

Para una mejor comprensión de los resultados de esta investigación, el trabajo queda organizado como sigue: en la sección 2 explica los fundamentos teóricos de las extensiones basadas en DE. La sección 3 describe los algoritmos propuestos poniendo especial énfasis en su hibridación con el enfoque mQSO. Los resultados de los experimentos realizados se exponen en la sección 4. Finalmente, la sección 4 está dedicada a las conclusiones y trabajos futuros.

II. Evolución Diferencial en ambientes dinámicos

La Evolución Diferencial es un paradigma computacional basado en las leyes evolutivas de Darwing propuesto por Storn y Price [21]. De manera formal, cada individuo ι_i de la población puede considerarse por la tupla $\langle \vec{x}_i, f_i \rangle$. Donde $\vec{x}_i$ es el vector solución y $f_i = f(\vec{x}_i)$ el valor correspondiente de la función objetivo. Como todo algoritmo evolutivo consta de dos etapas generales: inicialización y ciclo principal. En el primero, la población es pseudo-aleatoriamente inicializada en el espacio de búsqueda (ej. siguiendo una distribución uniforme), mientras que en el ciclo principal se generan nuevos individuos (soluciones candidatas) mediante cruzamiento y mutación. La aplicación de estos operadores evolutivos es controlada por el parámetro $\mathcal{CR}$. El operador de reemplazamiento es simple y no depende de una generación (iteración) del algoritmo: si el nuevo individuo es mejor (ej. según el fitness) entonces este reemplaza al individuo que le dio origen. En particular, la mutación depende básicamente del parámetro $\mathcal{F}$ conocido como factor de escala. Los principales pasos del esquema original de la Evolución Diferencial se muestran en el Algoritmo 1.

Algoritmo 1 Evolución Diferencial estándar

```
1:  P ← inicializarPoblacion(μ)
2:  while no se cumpla condición de parada do
3:     for cada individuo ι_i ∈ P do
4:        Crear el vector y⃗_i ∈ Ω
5:        Seleccionar ι_r, ι_s, ι_t ∈ P : r ≠ s ≠ t
6:        Seleccionar j ∈ [1, ..., D].
7:        for cada dimension d ∈ [1, ..., D] do
8:           if j == d or u_rand < CR then
9:              y_i^d ← x_r^d + F · (x_s^d − x_t^d)
10:          else
11:             y_i^d ← x_i^d
12:          end if
13:       end for
14:       f_{y_i} ← f(y⃗_i)
15:       if f_{y_i} ≻ f_{x_i} then
16:          x⃗_i ← y⃗_i
17:       end if
18:    end for
19: end while
```

Aquí u_{rand} es una función que devuelve números aleatorios siguiendo una distribución uniforme en el intervalo $[0, 1]$. Asimismo, es fácil observar que el rendimiento de DE depende críticamente de: el tamaño de la población, los parámetros $\mathcal{F}$, $\mathcal{CR}$, y de la estrategia de mutación. En varios trabajos (véase por ejemplo [9] y [15]) los parámetros $\mathcal{F}$, $\mathcal{CR}$ se sugiere que se establezcan en 1.0 y 0.1 respectivamente, el tamaño de la población un valor cercano a 10 veces el número de dimensiones del espacio de búsqueda, mientras que la estrategia de mutación depende del problema en cuestión. Dentro de las estrategias más conocidas se encuentran las siguientes:

DE/rand/1:

$$y_i^d = x_r^d + \mathcal{F}(x_s^d - x_t^d) \tag{2}$$

DE/cur-to-best/1:

$$y_i^d = x_i^d + \mathcal{F}(x_{best}^d - x_i^d) + \mathcal{F}(x_r^d - x_s^d) \tag{3}$$

DE/rand-to-best/2:

$$y_i^d = x_r^d + \mathcal{F}(x_{best}^d - x_i^d) + \mathcal{F}(x_s^d - x_t^d) + \mathcal{F}(x_u^d - x_v^d) \tag{4}$$

DE/current-to-rand/1:

$$y_i^d = x_i^d + +\mathcal{K}(x_r^d - x_i^d) + \mathcal{F}_p(x_s^d - x_t^d) \tag{5}$$

Donde $\mathcal{K}, \mathcal{F}_p \leftarrow u_{rand}$ son factores de escala que toman valores en tiempo de ejecución en el intervalo $[0, 1]$. Además, la componente x_{best}^d pertenece al mejor individuo de la población.

En el contexto de la optimización en ambientes dinámicos, DE ha sido aplicada con éxito en varios contextos. Por ejemplo, la extensión DynDE propuesta por Mendes y Mohai [14] utiliza a DE en varias poblaciones que exploran simultáneamente el espacio de búsqueda, lo que la hace muy adecuada para problemas multimodales. Curiosamente, DynDE no requiere el establecimiento de los parámetros $\mathcal{F}$ y $\mathcal{CR}$ ya que los selecciona en tiempo de ejecución en el intervalo $[0, 1]$. Asimismo, este método incluye técnicas de generación de diversidad mediante la perturbación de uno o más soluciones de la población. DynDE fue aplicada con buenos resultados en el problema test Movimiento de Picos [4].

Más recientemente, en al competición "Evolutionary Computation in Dynamic and Uncertain Environments" (CEC2009), Brest et. al[7] proponen una extensión al algoritmo auto-adaptativo jDE que utiliza varias poblaciones y una estrategia basada en la edad de los individuos. Este algoritmo alcanzó el primer lugar en dicha competición. Por otro lado, en entornos reales se puede apreciar el trabajo de Angira y Santosh [1] que aplican una versión trigonométrica de DE (propuesta inicialmente por [11]) con el objetivo de resolver problemas dinámicos en la ingeniería química.

A. Extensiones auto-adaptativas

El esquema original de DE ha sido satisfactoriamente extendido por varios autores con la intención de mejorar su robustez, esto es, su rendimiento en diversos problemas. Dentro de estas extensiones sobresalen las que proponen la auto-adaptación de

los parámetros y estrategias que influyen en el comportamiento del algoritmo. La auto-adaptación, como técnica estado-del-arte en el control de parámetros no es originaria de DE. De hecho su historia se remonta al origen de otros paradigmas evolutivos como las Estrategias Evolutivas[20], la Programación Evolutiva[12] y algunas versiones de Algoritmos Genéticos de codificación real[2]. Básicamente, este técnica consiste en la inclusión en el ciclo evolutivo de los parámetros más importantes del algoritmo, lo que significa que estos *evolucionan* al mismo tiempo que el conjunto de soluciones [10].

En relación a DE, existen en la literatura varias propuestas interesantes que incluyen de una forma u otra auto-adaptación. Por ejemplo, SADE [18] y SspDE [17] aplican auto-adaptación en la selección de la estrategia de mutación, mientras que jDE [5], DESAP [22], y SDE [19] lo hacen en los parámetros $\mathcal{F}$ y $\mathcal{CR}$. Particularmente, DESAP también auto-adapta el tamaño de la población. De manera general, la mayoría de estas extensiones superan en rendimiento a la versión estándar de DE en problemas estacionarios. En el presente estudio, se han seleccionado específicamente los algoritmos jDE y $SspDE$ debido a dos razones: por un lado estos algoritmos son, en relación con las otras variantes, los de mejor rendimiento y por otro, poseen marcadas diferencias entre sí lo que enriquece los resultados del estudio. A continuación se describen las ideas básicas de estos algoritmos.

El algoritmo jDE es muy similar al esquema original de DE, la diferencia radica en que los individuos contienen realizaciones de los parámetros $\mathcal{F}$ y $\mathcal{CR}$ (ej. $\iota_i = \langle \vec{x}_i, f_i, \mathcal{F}_i, \mathcal{CR}_i \rangle$). Entonces, en la generación de $\vec{y}_i$ se utilizan los valores de las expresiones 6 y 7. Si el nuevo individuo es mejor que su padre, en el reemplazamiento se tiene en cuenta también estos valores de $\mathcal{F}_{y_i}$ y $\mathcal{CR}_{y_i}$.

$$\mathcal{F}_{y_i} = \begin{cases} 0.1 + 0.9 \cdot u_{rand} & \text{si } u_{rand} < \tau_1 \\ \mathcal{F}_i & \text{en otro caso.} \end{cases} \quad (6)$$

$$\mathcal{CR}_{y_i} = \begin{cases} u_{rand} & \text{si } u_{rand} < \tau_2 \\ \mathcal{CR}_i & \text{en otro caso.} \end{cases} \quad (7)$$

Los parámetros τ_1 y τ_2 permanecen fijos durante la ejecución y según sus autores deben tomar el mismo valor: igual a 0.1.

Por su parte, el algoritmo SspDE es una extensión más sofisticada de DE. Los individuos vienen dados por la tupla $\iota_i = \langle \vec{x}_i, f_i, \vec{\mathcal{F}}_i, \vec{\mathcal{CR}}_i, \vec{\mathcal{S}}_i, wp_i, \vec{\mathcal{F}}_i^*, \vec{\mathcal{CR}}_i^*, \vec{\mathcal{S}}_i^* \rangle$. Donde $\vec{\mathcal{F}}_i, \vec{\mathcal{CR}}_i$ son listas de posibles valores para los parámetros $\mathcal{F}$ y $\mathcal{CR}$ respectivamente. Asimismo, el vector $\vec{\mathcal{S}}_i$ contiene varias estrategias de mutación, específicamente instancias de las cuatro definidas en la sección anterior (expresiones 2-5). Los vectores $\vec{\mathcal{CR}}_i^*, \vec{\mathcal{S}}_i^*$ son listas de parámetros y estrategias *ganadoras*, esto es, aquellos valores de $\vec{\mathcal{F}}_i, \vec{\mathcal{CR}}_i, \vec{\mathcal{S}}_i$ relacionados con la creación de un hijo de mejor calidad que el padre, siendo estas mejoras contadas por la variable wp_i. De manera general, todos los vectores poseen una longitud predefinida conocida como *período de aprendizaje* denotado por LP. Una vez que este período termina se crean nuevamente las listas de parámetros y estrategias a partir de las listas *ganadoras*. La idea principal de este método es el aprovechamiento de configuraciones de parámetros y estrategias que resultaron útiles en determinados momentos (ej. cada LP generaciones) para favorecer la optimización en generaciones posteriores. El lector puede encontrar más detalles de este algoritmo en el trabajo [17].

III. Algoritmos propuestos

Los algoritmos que serán objeto de análisis en este trabajo se basan en el enfoque Multienjambre Quantum PSO propuesto por Blackwell y Branke en [3]. En esencia, este enfoque emplea varias poblaciones independientes (enjambres de partículas), lo que lo hace muy apropiado para problemas multimodales. Con el propósito de solucionar la pérdida de diversidad (presentada por el paradigma PSO) este enfoque utiliza otro tipo de partículas denominadas Quantum que orbitan alrededor de la mejor solución del enjambre. Estas partículas no tienen un comportamiento convergente y se mantienen explorando según un radio fijo r_{cloud}. Por otra parte, los enjambres son controlados por un operador de *exclusión* que evita que más de un enjambre converja hacia un mismo óptimo, si este fuera el caso, el peor de los enjambres es reiniciado en el espacio de búsqueda. Para determinar si dos enjambres están situados en un mismo óptimo la distancia euclídea entre las mejores soluciones de ambos enjambres tiene que ser menor que cierto radio r_{excl}.

Algoritmo 2 Esquema general de los algoritmos

```
1:  for cada población p_i do
2:      Inicializar aleatoriamente a p_i
3:  end for
4:  while no se cumpla condición de parada do
5:      Aplicar operador exclusión
6:      if no ocurre cambio en el ambiente then
7:          for cada población p_i do
8:              Iterar p_i según: DE, jDE, SspDE, o QSO
9:          end for
10:     else
11:         for cada población p_i do
12:             Responder a los cambios
13:         end for
14:     end if
15: end while
```

La manera en que serán incluidos los algoritmos DE, jDE y SspDE en el enfoque mQSO es simple: reemplazando al paradigma PSO como algoritmo de las sub-poblaciones (ver el Algoritmo 2). Nuestra intención es analizar que beneficios traen las posibles hibridaciones con el mencionado enfoque. Adicionalmente, también hemos considerado utilizar otro tipo de estrategia para la generación de diversidad (no incluida en el enfoque mQSO). Esta forma de diversidad está basada en el trabajo [16] y es en principio

TABLA I
ALGORITMOS IMPLEMENTADOS

Algoritmo	**Partículas quantum**	**Diversidad**
mRDE	–	✓
mRJDE	–	✓
mRSspDE	–	✓
mQDE	✓	–
mQJDE	✓	–
mQSspDE	✓	–
mQSO	✓	–

TABLA II
CONFIGURACIÓN PARA EL ESCENARIO 2 DEL MPB

Parámetro	Configuración
Número de picos	10
Dimensiones	5
Altura de los picos	$\in [30, 70]$
Anchura de los picos	$\in [1, 12]$
Frecuencia (Δe)	$\in [1000, 5000, 10000]$
Severidad (sev)	$\in$[1.0,5.0,10.0]
Factor de correlación	0.0

similar a la de las partículas quantum, sin embargo esta estrategia se ejecuta después de cada cambio. Formalmente, después de cada cambio la mitad de la población es reiniciada en una hiper-esfera de radio r_{div} entorno a la mejor solución.

Es importante señalar algunos aspectos que sobresalen en relación con esta hibridación de DE y ambas estrategias de diversidad. Por ejemplo, cuando se utilizan partículas quantum la población de soluciones tendrá individuos de diferentes tipos (quantum y los correspondientes a la extensión de DE). Esto evidentemente afecta la velocidad de convergencia de las sub-poblaciones por las características de las estrategias de mutación (ej. por la elección aleatoria de individuos de la población). Sin embargo, puede ser un comportamiento deseable en situaciones donde se requiera mantener la diversidad (ej. problemas con frecuencia de cambio muy altas). Por tal motivo, los algoritmos propuestos basados en DE que incluyen individuos de tipo quantum tendrán una población mixta, propiciando así la interacción de sus individuos mediante la estrategia de mutación. En relación con la aplicación de la estrategia de diversidad después del cambio no existen dificultades mayores pues se trata de variar el vector solución de individuos existentes.

En la tabla I se listan los algoritmos implementados. Nótese que se han incluido además, para favorecer la comparación, el enfoque mQSO. Las extensiones que emplean partículas quantum utilizan el prefijo mQ, mientras que las que emplean diversidad después del cambio son indentificadas por mR.

IV. EXPERIMENTOS COMPUTACIONALES

En esta sección se exponen los resultados de los experimentos realizados sobre el problema artificial Movimiento de Picos (MPB) [4]. Dada su flexibilidad, este problema permite obtener diferentes instancias a partir de las combinaciones de sus parámetros. En la tabla II se muestran las configuraciones generales utilizadas en este trabajo y que corresponden a lo que se conoce como Escenario 2 de este problema.

Para medir el rendimiento de los algoritmos se empleó el promedio del error del mejor individuo antes de cada cambio (e_{mac}), que se define como:

$$e_{mac} = \frac{1}{T'} \sum_{t=1}^{T'} \Big(f(\vec{x}^*) - f(\vec{x}_{best}) \Big) \qquad (8)$$

Los experimentos fueron divididos en dos grupos con la intención de analizar el rendimiento de los algoritmos en problemas con características diferentes. En ese sentido, el factor que se tomó en cuenta fue el tipo de función que define a los picos: en el primer grupo se utilizarán funciones unimodales, mientras que en el segundo multimodales. La Tabla III muestra las definiciones de las funciones seleccionadas. En ambos grupos de experimentos por cada función pico se consideraron varios valores para la frecuencia de los cambios: $\Delta e = \{1000, 5000, 10000\}$ y la severidad $sev = \{1.0, 5.0, 10.0\}$. Nótese que se tratan de valores que poseen marcadas diferencias entre sí. Todos los algoritmos emplearán 10 sub-poblaciones de 10 individuos cada una, en particular, los algoritmos que utilizan partículas quantum tendrán 5 individuos de este tipo. Además, se empleó la siguiente configuración general: $r_{excl} = r_{div} = 0.3 \cdot \Omega_{ext}$, $r_{cloud} = 0.1 \cdot \Omega_{ext}$. Donde Ω_{ext} es la extensión del espacio de búsqueda. Las extensiones basadas en DE y jDE utilizaron *DE/Rand/1* (expresión 2) como estrategia de mutación. De manera general se consideraron 20 ejecuciones por cada par problema algoritmo con semillas aleatorias diferentes. Para todos los problemas el número de cambios se estableció en 50.

A. Resultados en problemas con picos unimodales

En este apartado se analizan los resultados obtenidos en problemas que emplean picos definidos por funciones unimodales. En la tabla IV se muestra la media del error antes del cambio y el correspondiente error estándar de dicha media. De manera general se observa una tendencia similar en las tres funciones, por ejemplo, el algoritmo mQSO, utilizado como control en los experimentos, es el que mejor rendimiento exhibe en este tipo de problemas. En relación con los algoritmos basados en DE, las variantes que utilizan partículas quantum como estrategia de diversidad poseen una marcada superioridad con respecto al resto. Particularmente, dentro de este grupo las que presentan auto-adaptación

TABLA III

FUNCIONES UTILIZADAS PARA LOS PICOS DEL ESCENARIO 2 DEL MPB.

	Función	Fórmula	Espacio de búsqueda
Unimodal	Cone	$f(\mathbf{x}) = \sqrt{\sum_{i=1}^{n} x_i^2}$	$[0, 100]^5$
	Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$[0, 100]^5$
	Schwefel	$f(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[0, 100]^5$
Multimodal	Rastrigin	$f(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10\cos 2\pi x_i + 10)$	$[-5.12, 5.12]^5$
	Ackley	$f(\mathbf{x}) = -20\exp(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + \exp$	$[-32, 32]^5$
	Griewank	$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-32, 32]^n$

TABLA IV

RESULTADOS EN PROBLEMAS CON PICOS UNIMODALES. ERROR ANTES DEL CAMBIO ± ERROR ESTÁNDAR (AVG. 20 EJECUCIONES). LOS VALORES EN NEGRITA CORRESPONDEN AL MEJOR ALGORITMO.

Función	Δe	Sev.	mRDE	mRJDE	mRSspDE	mQDE	mQJDE	mQSspDE	mQSO
Cone	1000	1.0	17.68±0.40	10.07±0.30	12.64±0.33	5.08±0.22	4.58±0.17	4.84±0.16	**3.68±0.07**
		5.0	22.94±0.36	15.19±0.21	17.85±0.17	6.90±0.26	5.76±0.10	6.69±0.28	**4.63±0.14**
		10.0	26.25±0.31	19.37±0.22	22.21±0.30	13.24±0.31	11.65±0.24	13.16±0.31	**5.89±0.11**
	5000	1	6.94±0.29	3.57±0.10	4.81±0.25	1.79±0.08	1.51±0.09	1.37±0.07	**0.97±0.09**
		5	8.68±0.30	4.98±0.21	6.21±0.27	2.38±0.09	1.68±0.11	1.82±0.08	**1.36±0.12**
		10	9.87±0.21	5.62±0.21	8.04±0.18	3.04±0.14	1.90±0.09	2.42±0.14	**1.54±0.09**
	10000	1	6.37±0.27	3.22±0.07	3.24±0.12	1.26±0.05	0.80±0.07	0.87±0.11	**0.43±0.06**
		5	7.12±0.26	4.09±0.09	3.70±0.16	1.56±0.06	0.90±0.09	1.13±0.12	**0.66±0.08**
		10	7.87±0.30	4.06±0.18	5.59±0.22	2.20±0.09	1.04±0.07	1.40±0.11	**0.88±0.08**
Sphere	1000	1	40.92±1.46	16.88±0.79	20.54±1.03	7.26±0.84	5.21±0.68	5.04±0.63	**2.32±0.33**
		5	72.94±2.14	32.37±1.30	37.47±1.33	9.87±0.74	6.71±0.64	7.78±0.61	**3.20±0.27**
		10	105.89±2.33	51.33±1.28	61.73±1.23	23.35±1.00	19.34±0.84	22.38±0.73	**5.46±0.30**
	5000	1	5.93±0.40	2.34±0.17	3.10±0.25	0.44±0.04	0.18±0.04	0.19±0.03	**0.06±0.02**
		5	7.20±0.47	3.75±0.18	5.42±0.23	0.93±0.06	0.22±0.03	0.33±0.05	**0.08±0.02**
		10	10.48±0.64	3.94±0.20	7.83±0.30	1.04±0.05	0.39±0.04	0.69±0.09	**0.18±0.04**
	10000	1	4.74±0.28	1.77±0.11	2.29±0.22	0.27±0.04	0.06±0.03	0.07±0.02	**0.00±0.00**
		5	5.58±0.30	2.52±0.22	3.79±0.27	0.54±0.05	0.06±0.03	0.15±0.04	**0.01±0.01**
		10	8.04±0.35	2.99±0.18	4.97±0.28	0.59±0.05	0.05±0.02	0.20±0.04	**0.02±0.01**
Schwefel P2.22	1000	1	24.60±0.46	12.90±0.23	16.61±0.28	7.99±0.32	6.67±0.24	7.22±0.29	**5.21±0.10**
		5	33.73±0.48	21.40±0.24	24.91±0.24	12.33±0.26	10.83±0.23	11.78±0.34	**7.46±0.13**
		10	39.83±0.50	28.84±0.35	31.73±0.30	23.32±0.28	23.12±0.38	23.66±0.34	**12.20±0.23**
	5000	1	8.33±0.30	4.34±0.18	6.17±0.21	2.90±0.11	2.00±0.13	1.95±0.13	**1.55±0.11**
		5	10.49±0.32	5.75±0.24	7.44±0.25	3.81±0.11	2.56±0.12	2.56±0.07	**1.76±0.11**
		10	12.33±0.29	6.18±0.17	9.23±0.26	4.50±0.18	2.89±0.10	4.00±0.15	**2.04±0.12**
	10000	1	7.85±0.24	4.06±0.08	4.78±0.21	1.98±0.10	1.38±0.13	1.33±0.12	**0.66±0.06**
		5	8.85±0.14	4.55±0.15	5.32±0.25	2.63±0.11	1.40±0.11	1.49±0.12	**0.98±0.11**
		10	10.18±0.26	4.32±0.17	6.75±0.21	3.02±0.11	1.57±0.10	1.73±0.11	**0.81±0.07**

son mejores que la basada en el esquema original de DE (mQDE), siendo la variante mQJDE la de mejor rendimiento. Esto también es válido para los algoritmos que utilizan diversidad después de los cambios, lo que significa que, al igual que lo que ocurre en la optimización de problemas estacionarios, las extensiones jDE y SspDE son mejores optimizadores que el esquema original DE (al menos para las funciones seleccionadas).

Con el objetivo de analizar estadísticamente las diferencias entre los algoritmos se aplicó la prueba no paramétrica de Friedman, la cual arrojó la existencia de diferencias significativas en sentido general ($\rho < 0.05$). En consecuencia, se aplicó la prueba post-hoc de Bonferroni para detectar diferencias por cada par de algoritmos. Los resultados de ambas pruebas aparecen en la Figura 1. Nótese que por el eje de las abscisas se indica la posición (rango) según Friedman, de manera que un valor cercano a 1 corresponde a un buen rendimiento (ej. mQSO) y viceversa. Asimismo, el solapamiento de las diferencias críticas, representadas por las líneas horizontales de las marcas expresan que no existen diferencias significativas entre los algoritmos involucrados. Por ejemplo, los algoritmos mQJDE y mQSspDE no son significativamente diferentes pues sus marcas se solapan. De igual forma las variantes mRJDE y mRSspDE no son diferentes entre sí. Es impor-

tante aclarar además que las marcas en azul y gris corresponden a las mejores variantes basadas en DE. Como puede ser observado, no existe en sentido general una variabilidad importante entre las relaciones de los algoritmos para las tres funciones.

En resumen, para problemas con funciones unimodales en los picos, el empleo de partículas quantum favorece el rendimiento de los algoritmos, a diferencia de la diversidad después de los cambios, la cual parece no ser suficiente.

B. Resultados en problemas con picos multimodales

El objetivo de los experimentos en esta sección es analizar el rendimiento de los algoritmos en problemas con funciones más complejas (ej. multimodales). Anteriormente, se pudo concluir que para funciones simples (unimodales) las variantes que usan partículas quantum son mejores. Sin embargo, en los problemas de esta sección la diversidad juega un papel importante si se tiene en cuenta la estructura de las funciones, las cuales aumentan la probabilidad de que el algoritmo quede atrapado en óptimos locales. En la tabla V se muestran los resultados para este tipo de funciones.

Nótese que esta vez existe un nuevo orden en relación al rendimiento de los algoritmos (Figura 2). Por ejemplo, el algoritmo usado como control (mQSO) es uno de los que peor rendimiento tiene de manera general: es el último en la función Griewank, y antepenúltimo en Ackley y Rastrigin. Resulta interesante la superioridad de la variante mQSspDE en las tres funciones, la cual se hace común para otras variantes en los casos Ackley y Rastrigin. Sorprendentemente, dentro de estas variantes exitosas aparecen algunas que emplean diversidad después del cambio (mRJDE y mRSspDE). Aunque similar a los experimentos anteriores se hace patente el pobre rendimiento de la variante mRDE. Curiosamente, en esta ocasión se une a este grupo de bajo rendimiento la variante mQJDE.

V. Conclusiones y trabajos futuros

En este trabajo se estudiaron de manera empírica algunas de las potencialidades del paradigma computacional Evolución Diferencial en ambientes dinámicos. Específicamente, la hibridación de extensiones del mencionado paradigma con algunas de las estrategias más eficaces para tratar la pérdida de diversidad. Dos de las extensiones consideradas presentan tipos diferentes de auto-adaptación en sus parámetros: jDE y SspDE. Por su parte, las estrategias de diversidad estuvieron basadas en el conocido enfoque Multienjambre Quantum PSO (mQSO) y en la generación de diversidad después del cambio. Un total de seis algoritmos fueron propuestos, los cuales, en conjunto con el propio mQSO, fueron probados en un importante número de instancias basadas en el escenario 2 del problema artificial Movimiento de Picos.

Los resultados indican que en problemas con funciones unimodales la aplicación de estrategias de diversidad durante la ejecución (partículas quantum) garantiza un rendimiento aceptable en comparación a la diversidad después del cambio. Sin embargo los resultado obtenidos por los algoritmos propuestos no superan al enfoque mQSO. En problemas multimodales, la utilidad de la diversidad durante la ejecución es apreciable para el esquema original DE y la extensión SspDE. En este último caso se reportan resultados significativamente mejores que el enfoque mQSO. De manera general los algoritmos que emplean auto-adaptación son mejores que el esquema original DE.

Finalmente, es apropiado decir que debido a la variabilidad de los resultados obtenidos, la aplicación de técnicas de control de parámetros como la adaptación o auto-adaptación parece justificarse en este contexto. De hecho en este trabajo pudo comprobarse que el éxito de las extensiones más sofisticadas de DE en ambientes dinámicos se debe a la inclusión de alguna de estas técnicas. Sin embargo, la auto-adaptación utilizada por estos algoritmos tiene por objetivo adaptar el proceso de optimización en los períodos de tiempo en que el problema no varía. En ese sentido resulta interesante, estudiar si la aplicación de algunas de estas técnicas puede ser aplicada en el control de la diversidad durante la ejecución o después del cambio. Nuestros trabajos futuros estarán orientados a estudiar estas cuestiones.

Agradecimientos

Referencias

[1] R. Angira and A. Santosh. Optimization of dynamic systems – a trigonometric differential evolution approach. *Comput. Chem. Eng.*, 31:1055–1063, Sep. 2007. 2

[2] H.-G. Beyer and K. Deb. On self-adaptive features in real-parameter evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 5(3):250 –270, jun. 2001. 3

[3] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10(4):459–472, 2006. 1, 3

[4] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1875–1882, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press. 2, 4

[5] J. Brest, S. Greiner, M. Boskovic, B. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *Transactions on Evolutionary Computation*, 10(6):646–657, 2006. 3

[6] J. Brest, A. Zamuda, B. Boskovic, S. Greiner, and V. Zumer. An analysis of the control parameters adaptation in de. In U. Chakraborty, editor, *Advances in Differential Evolution*, number 143, pages 89–110. Springer Berlin / Heidelberg, 2008. 1

[7] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer. Dynamic optimization using self-adaptive differential evolution. In *CEC09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, pages 415–422, Piscataway, NJ, USA, 2009. IEEE Press. 2

[8] C. Cruz, J. González, and D. Pelta. Optimization in dynamic environments: a survey on problems, methods and

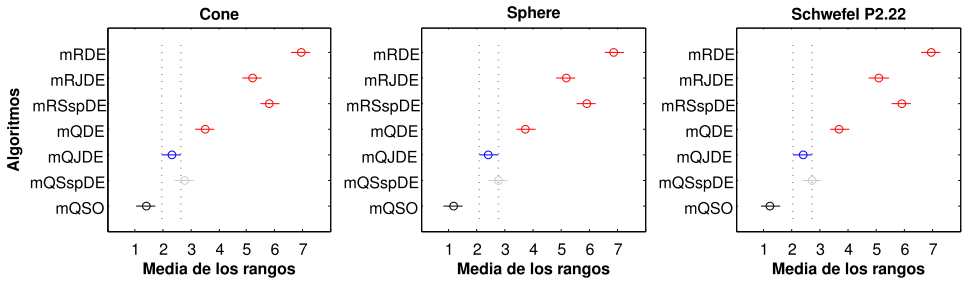

Fig. 1

Relación entre los algoritmos según el test post-hoc de Bonferroni (nivel de significación 5.0 %).

TABLA V

Resultados en problemas con picos multimodales. Error antes del cambio ± error estándar (Avg. 20 ejecuciones). Los valores en negrita corresponden al mejor algoritmo.

Función	Δe	Sev.	mRDE	mRJDE	mRSspDE	mQDE	mQJDE	mQSspDE	mQSO
Griewank	1000	1.0	0.40±0.00	0.36±0.00	0.36±0.00	0.18±0.01	0.27±0.01	**0.14±0.00**	0.24±0.01
		5.0	0.45±0.00	0.40±0.00	0.39±0.00	0.17±0.00	0.24±0.01	**0.15±0.00**	0.19±0.01
		10.0	0.46±0.00	0.40±0.00	0.40±0.00	0.18±0.00	0.26±0.01	**0.15±0.00**	0.21±0.01
	5000	1	0.13±0.00	0.12±0.00	0.14±0.00	0.13±0.01	0.19±0.01	**0.05±0.00**	0.20±0.01
		5.0	0.16±0.00	0.15±0.00	0.16±0.00	0.09±0.00	0.19±0.01	**0.06±0.00**	0.18±0.01
		10.0	0.16±0.00	0.14±0.00	0.16±0.00	0.09±0.00	0.21±0.01	**0.06±0.00**	0.19±0.01
	10000	1.0	0.06±0.00	0.06±0.00	0.07±0.00	0.12±0.01	0.19±0.01	**0.04±0.00**	0.19±0.01
		5.0	0.07±0.00	0.07±0.00	0.09±0.00	0.08±0.00	0.19±0.01	**0.04±0.00**	0.18±0.01
		10.0	0.07±0.00	0.07±0.00	0.09±0.00	0.09±0.00	0.21±0.01	**0.04±0.00**	0.19±0.01
Ackley	1000	1.0	7.78±0.10	4.68±0.11	5.52±0.10	4.71±0.26	5.56±0.32	**3.62±0.13**	3.85±0.22
		5.0	8.52±0.08	**6.05±0.07**	6.56±0.07	6.85±0.20	10.77±0.35	6.14±0.08	7.28±0.43
		10.0	9.22±0.07	**7.25±0.07**	7.63±0.06	9.87±0.15	15.41±0.18	7.93±0.08	10.74±0.49
	5000	1.0	1.95±0.06	0.79±0.03	0.69±0.02	0.66±0.06	1.27±0.17	**0.40±0.03**	1.01±0.22
		5.0	2.60±0.05	1.48±0.05	1.24±0.05	**1.13±0.04**	5.46±0.38	1.96±0.07	4.99±0.30
		10.0	2.99±0.05	1.59±0.06	**1.42±0.06**	2.17±0.10	13.11±0.30	2.70±0.08	12.01±0.41
	10000	1.0	1.52±0.05	0.86±0.04	0.43±0.03	0.24±0.04	0.63±0.13	**0.11±0.01**	0.49±0.13
		5.0	2.11±0.06	1.40±0.05	0.72±0.05	**0.45±0.05**	3.49±0.28	1.58±0.05	4.97±0.26
		10.0	1.94±0.06	1.39±0.06	0.90±0.03	**0.62±0.05**	12.34±0.39	1.95±0.06	11.49±0.30
Rastrigin	1000	1.0	10.53±0.20	9.27±0.20	10.90±0.16	5.87±0.31	8.74±0.47	**4.72±0.20**	6.78±0.37
		5.0	16.78±0.15	14.73±0.12	15.54±0.14	7.12±0.24	8.94±0.54	**5.98±0.12**	7.54±0.28
		10.0	17.45±0.13	15.70±0.15	16.66±0.14	9.52±0.18	14.96±0.42	**7.65±0.07**	10.39±0.25
	5000	1.0	2.37±0.12	**1.29±0.05**	1.92±0.08	4.24±0.25	8.63±0.47	2.37±0.11	6.35±0.50
		5.0	5.06±0.12	3.35±0.08	4.77±0.09	4.13±0.26	7.01±0.38	**2.54±0.07**	5.43±0.26
		10.0	5.70±0.08	3.88±0.08	5.49±0.08	6.36±0.20	12.67±0.47	**3.36±0.07**	10.74±0.39
	10000	1.0	1.60±0.06	**0.93±0.04**	0.78±0.04	4.25±0.26	8.17±0.54	1.69±0.08	5.87±0.40
		5.0	2.71±0.07	**1.17±0.05**	1.79±0.08	3.53±0.19	7.40±0.37	1.92±0.07	5.00±0.33
		10.0	2.88±0.04	**1.42±0.06**	2.07±0.06	6.05±0.26	12.88±0.53	2.40±0.04	10.15±0.34

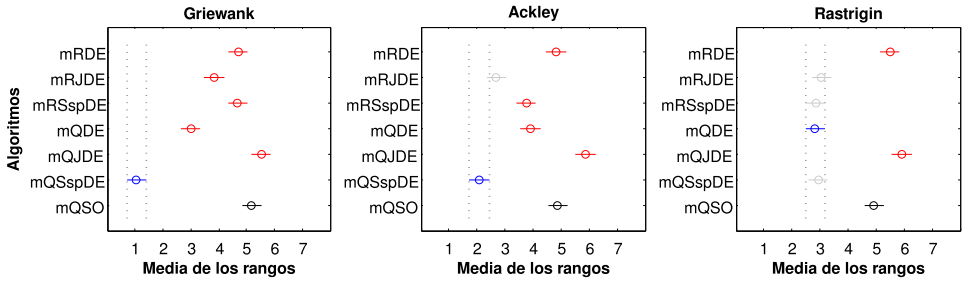

Fig. 2

Relación entre los algoritmos según el test post-hoc de Bonferroni (nivel de significación 5.0 %).

measures. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, pages 1–22, 2010. 1

[9] S. Das and P. N. Suganthan. Differential evolution – a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, February 2011. 2

[10] A. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith. Parameter control in evolutionary algorithms. In F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 19–46. Springer Berlin / Heidelberg, 2007. 3

[11] H.-Y. Fan and J. Lampinen. A trigonometric mutation operation to differential evolution. *J. Global Optimization*, 27(1):105–129, 2003. 2

[12] D. B. Fogel. An analysisof evolutionary programming. In D. B. Fogel and J. W. Atmar, editors, *First Annual Conference on Evolutionary Programming*, pages 43–51, 1992. 3

[13] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, 9(3):303–317, 2005. 1

[14] R. Mendes and A. S. Mohais. Dynde: A differential evolution for dynamic optimization problems. In *Proc. IEEE Congr. Evol. Comput.*, volume 2, pages 2808–2815, 2005. 1, 2

[15] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artif Intell Rev*, 33:61–106, 2010. 2

[16] P. Novoa-Hernández, D. Pelta, and C. Corona. Improvement strategies for multi-swarm pso in dynamic environments. In J. González, D. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, volume 284 of *Studies in Computational Intelligence*, pages 371–383. Springer Berlin / Heidelberg, 2010. 3

[17] Q.-K. Pan, P. Suganthan, L. Wang, L. Gao, and R. Mallipeddi. A differential evolution algorithm with self-adapting strategy and control parameters. *Computers & Operations Research*, 38:394–408, 2011. 1, 3

[18] A. Qin and P. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *The 2005 IEEE Congress on evolutionary computation CEC2005*, pages 1785–1791, 2005. 3

[19] E. A. O. M. Salman, A. Empirical analysis of self-adaptive differential evolution. *European Journal of Operational Research*, 183(2):785–804, 2007. 3

[20] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley, Chichester, UK, 1981. 3

[21] R. Storn and K. Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997. 1, 2

[22] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, 10(8):673–686, 2006. 3