IMPACTO DE LA AUTO-ADAPTACIÓN EN AMBIENTES DINÁMICOS CON FRECUENCIA DE CAMBIO VARIABLE

Pavel Novoa-Hernández*, Byron Oviedo Bayas*, Jorge Murillo Oviedo*, Amilkar Puris* ,Moisés Menace*,Carlos Cruz Corona**
*Facultad de Ciencias de La Ingeniería, Universidad Técnica Estatal de Quevedo, Ecuador.
[1]**Dept. Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España.

ABSTRACT
Several decision scenarios can be modeled as dynamic optimization problems (DOPs), which have been tackled by meta-heuristics techniques over the last years. However, so far, most related works assume that changes occur every equal time periods, which may be rather idealistic in real-world scenarios. In contrast, DOPs with variable change frequency (DOPVCFs) impose as main challenge to the algorithm: how to adapt to different environments during the run, as fast as possible. In that sense, self-adaptation is parameter control technique with remarkable success in complex scenarios. So, in the present work we aim to analyze the impact of self-adaptation in solving DOPVCFs. To achieve this, we have designed an experimental study by considering a recently proposed self-adaptive technique over several test scenarios. The results confirm that self-adaptation has not only a positive, but also significant impact in the algorithm performance.

KEYWORDS: Self-adaptation, Evolutionary dynamic optimization, Variable change frequency, Differential Evolution.

MSC: 68T20

RESUMEN
Diversos escenarios pueden ser modelados como problemas dinámicos de optimización (PDOs), los cuales han sido abordados por técnicas meta-heurísticas en los últimos años. Sin embargo, hasta el momento, la mayoría de los trabajos existentes asumen que los cambios en el problema ocurren en intervalos iguales de tiempo, lo cual pudiera resultar idealista en escenarios reales. En contraste, los PDOs con frecuencia de cambio variable (PDOFCVs) imponen como principal reto al algoritmo: cómo adaptarse rápidamente a ambientes diferentes a lo largo del tiempo. En este sentido, la auto-adaptación es una técnica de control de parámetros que ha resultado efectiva en contextos similares. El presente trabajo tiene por objetivo por tanto, analizar la influencia de la auto-adaptación en la solución de PDOFCVs. Para lograrlo, hemos diseñado un estudio experimental considerando una técnica de auto-adaptación recientemente propuesta, y en varios escenarios de prueba. Los resultados indican que la auto-adaptación posee un impacto positivo y significativo en el rendimiento de los algoritmos en este contexto.

1. INTRODUCCIÓN

En el mundo moderno, varios escenarios de decisión pueden ser modelados como problemas dinámicos de optimización (PDOS). Ejemplos de este tipo de problema son: la automatización del sistema de semáforos de una ciudad, el control del movimiento de un robot, la distribución óptima de recursos a clientes que aparecen o desaparecen con el tiempo, entre otros. Dado el importante nivel de incertidumbre de los PDOS, la aplicación de técnicas clásicas de optimización resulta en muchos casos imposible. es en este contexto donde las denominadas meta-heurísticas han sido ampliamente estudiadas en la última década, véase por ejemplo [4, 7, 11, 14]. De manera general, se asume que el objetivo para el algoritmo es encontrar la mejor solución a lo largo de la ejecución, esto es, en cada instante de tiempo.

[1] Boviedo@uteq.edu.ec

Un aspecto común en la mayoría de los trabajos existentes sobre optimización en ambientes dinámicos, es que se asume que el problema cambia cada cierto intervalo fijo de tiempo (ej. [3, 5, 12, 16, 17]). Esto provoca que el algoritmo, si logra adaptarse a los cambios, muestre un comportamiento casi oscilatorio en cada ambiente. En otras palabras, es fácil comprobar que si el algoritmo muestra un rendimiento aceptable en las primeras etapas del problema, en las etapas posteriores debe comportarse de manera similar. Sin embargo, en problemas reales, la ocurrencia de los cambios no tiene por qué ser en intervalos de tiempo iguales. Un ejemplo típico de estos escenarios es el mercado bursátil, en el que los cambios difícilmente pueden predecirse y se requiere reaccionar rápidamente. De manera que los PDOS con frecuencia de cambio variable (PDOFCVs) imponen como principal reto al algoritmo: cómo adaptarse rápidamente a ambientes *diferentes* a lo largo del tiempo. Es de notar que si el problema describe patrones regulares en la frecuencia de los cambios (ej. según una función periódica), entonces alguna técnica de predicción puede aplicarse con el objetivo de anticipar los escenarios y reaccionar adecuadamente a estos. en esta dirección en la literatura muestra los trabajos de [18] y [19]. En estos trabajos, los autores aplican técnicas de predicción. la primera técnica emplea regresión lineal [18] o no lineal [19], para estimar la iteración en la que aparecerá el nuevo cambio. La segunda técnica está basada en cadenas de Markov para monitorear las transiciones de ambientes anteriores y estimar qué ambiente aparecerá en el próximo cambio. Como muestran los experimentos en ambos trabajos, un algoritmo evolutivo con estas técnicas alcanza excelentes resultados, particularmente en escenarios con cambios cíclicos. Sin embargo, cuando los cambios no siguen un patrón claro, la aplicación de técnicas de predicción clásicas como regresión lineal o no lineal es poco efectiva. resulta de interés por tanto investigar cómo abordar este tipo de problema y específicamente si los enfoques que han sido efectivos hasta ahora en PDOS con frecuencia de cambios constante, pudieran ser aplicados con éxito en este contexto. precisamente un enfoque que ha mostrado ser efectivo en los últimos años es la aplicación de estrategias de auto-adaptación [10]. La idea detrás de esta técnica de control de parámetros es dotar de un comportamiento inteligente al algoritmo durante la ejecución. Es importante notar no obstante que, hasta donde se conoce, en la literatura no existen trabajos que estudien a los PDOSFCVS desde el punto de vista de la auto-adaptación.

Con la intención de responder a las cuestiones anteriormente descritas, el presente trabajo tiene por objetivo analizar el impacto de la auto-adaptación en la solución de PDOSFCVS. Para lograrlo, se ha diseñado un estudio experimental considerando una técnica de auto-adaptación recientemente propuesta en [15] y que ha resultado particularmente efectiva en PDOs con frecuencia de cambios constante. Nuestra hipótesis es que un algoritmo con auto-adaptación orientada a la gestión de diversidad (como es el caso del propuesto en [15]) puede afrontar de manera efectiva la aparición irregular de escenarios de optimización. Con el objetivo de comprobar esta hipótesis hemos diseñado varios problemas de prueba, tomando como base el conocido estándar de comparación *movimiento de picos* [2], en combinación con 5 tipos de cambios para variar las frecuencia de los cambios.

El resto del trabajo queda organizado de la forma siguiente: en la sección "preliminares" se definen formalmente a los PDOs y PDOFCVs, incluyéndose una descripción de los tipos de cambios empleados para variar la frecuencia de cambio. Más adelante, en la sección "algoritmos auto-adaptativos" se explican las principales características de los algoritmos considerados en el estudio, los cuales son analizados en la sección "experimentos computacionales" dedicada a los experimentos computacionales diseñados. Finalmente, en la sección "conclusiones" se brindan las conclusiones y trabajos futuros derivados de esta investigación.

2. PRELIMINARES

En esta sección se definen formalmente a los PDOs y a los PDOFCVs. Adicionalmente se describen los tipos de cambios que se proponen para el estudio experimental.

2.1. Problemas dinámicos de optimización

Formalmente, un problema dinámico de optimización se define de la forma siguiente:

$$PDO := \{ min\ F(x) = f(x, \phi, t) \quad (1)$$

Donde f es una función continua, x es una solución factible en el conjunto solución X, t es el tiempo, y ϕ representa el parámetro de control del sistema, el cual determina la distribución de las soluciones en el espacio de búsqueda.

El dinamismo del problema viene dado precisamente por la presencia de una desviación $\Delta\phi$ que afecta a la distribución de las soluciones en el ambiente t. Formalmente esto sería:

$$\phi(t\ +\ 1)\ =\ \phi(t) \oplus \Delta\phi \tag{2}$$

De manera que cada ambiente $t\ +\ 1$ se obtiene del anterior de la siguiente manera:

$$f(x, \phi, t\ +\ 1)\ =\ f(x, \phi(t)\ +\ \Delta\phi, t) \tag{3}$$

Usualmente ϕ es la posición de la solución óptima del problema, que varía durante el proceso de optimización cada Δe unidades de tiempo. A este intervalo Δe se le conoce como *frecuencia de cambio*. Como se ha advertido anteriormente, en las investigaciones existentes se asume que Δe no varía en el tiempo. En otras palabras, el algoritmo tiene la misma cantidad de tiempo para optimizar cada uno de los ambientes antes de que se produzca un nuevo cambio. En general, los investigadores han estado más enfocados en el *qué* cambia (ej. posición del óptimo, espacio de búsqueda, etc.), y no en el *cuándo*.

2.2. Problemas dinámicos de optimización con frecuencia de cambio variable

Ahora bien, supongamos que la frecuencia de los cambios varía en el tiempo. Esto implica que Δe sería un nuevo parámetro de control del sistema, el cual no necesariamente afectaría la distribución de las soluciones en el espacio de búsqueda, pero sí cuando aparecen los cambios. Denotemos este parámetro como δ, entonces un PDO con frecuencia de cambio variable se definiría como:

$$PDOFCV\ :=\ \{\ min\ F(x)\ =\ f(x, \phi, \delta, t) \tag{4}$$

Aquí, el parámetro δ cambiaría de manera similar a la expresión (2). Nótese que dicha expresión nos permite obtener una secuencia numérica, esto es, de las frecuencias de cambio. En particular, consideraremos los tipos de cambio siguientes, originalmente propuestos en [8]:

Paso corto:

$$\Delta\delta = \alpha \cdot \|\delta\| \cdot r \cdot \delta_{sev} \tag{5}$$

Paso largo:

$$\Delta\delta = \|\delta\| \cdot (\alpha \cdot sign(r) + (\alpha_{max} - \alpha) \cdot r) \cdot \delta_{sev} \tag{6}$$

Caótico:

$$\delta(t+1) = A \cdot (\delta(t) - \delta_{min}) \cdot (1 - \frac{(\delta(t) - \delta_{min})}{\|\delta\|}) \tag{7}$$

Recurrente:

$$\delta(t+1) = \delta_{min} + \|\delta\| \cdot \frac{\text{sen}\left(\frac{2\pi}{P} \cdot t + \varphi\right) + 1}{2} \tag{8}$$

Donde $\|\delta\|$ es el rango de variación de δ, δ_{sev} es una constante que indica la severidad de cambio de δ, δ_{min} es el valor mínimo de δ. Por otro lado, $r \in (-1,1)$ es un número aleatorio, y $\alpha, \alpha_{max} \in (0,1)$ son constantes establecidas en 0.005 y 0.3 respectivamente. En el caso del tipo de cambio caótico, A es una constante igual a 3.67, mientras que en el caso de tipo recurrente, $P = 12$, es el período de la función $\text{sen}\ x$. La expresión $sign$ es la función signo, retornando 1 cuando $r > 0$, -1 si $r < 0$, y 0 por el contrario.

Adicionalmente, proponemos un denominado *ruidoso* que se define de la forma siguiente:

Ruidoso:

$$\delta(t+1) = N(\delta_0, \delta_{desv}) \tag{9}$$

Donde δ_0 y δ_{desv} son la media y desviación estándar de la distribución normal con la que se generan los números en la función N.

Los tipos de cambio descritos en las expresiones (5)-(9) permiten obtener cinco escenarios distintos, como se verá más adelante en la sección dedicada a los experimentos computacionales. En la Figura 1 se ilustran que aspecto tienen estas funciones, para la configuración de parámetros mencionados anteriormente y $\delta_0 = \delta_{desv} = 5000$, $\delta_{min} = 500$, $\delta_{max} = 10000$ y $\delta_{sev} = 50$. Nótese los distintos patrones que siguen los tipos de cambios definidos. Como elemento adicional de este ejemplo, en la gráfica a) hemos incluido el caso constante, en el que la frecuencia de los cambios no varía.

En la propia Figura 1 es posible confirmar que el mayor reto del algoritmo al enfrentar un PDOFCV es cómo responder (adaptarse) rápidamente a los nuevos escenarios que aparecen en el tiempo. Esto se debe a que los ambientes del problema tienen diferentes ventanas de tiempo, y por tanto, el algoritmo no siempre podrá contar con el tiempo necesario para encontrar una buena solución. Aunque este requerimiento es también válido para los PDOs con frecuencia de cambio constante (dado que en estos problemas también el algoritmo no conoce cuando ocurrirán los cambios) es importante notar que, si el investigador ajusta apropiadamente los parámetros que influyen en la adaptación del algoritmo, entonces es posible obtener un comportamiento casi

oscilatorio del rendimiento del mismo. En otras palabras, es posible que el algoritmo se comporte de manera similar en cada ambiente del problema. En contraste, en el caso de los PDOFCVs, la alternancia entre ambientes con tiempos de vigencia cortos y largos, sugiere que el algoritmo mostrará un comportamiento dependiente del tipo de cambio que muestre el problema. Esto último se podrá observar mejor en la sección dedicada a los experimentos computacionales.

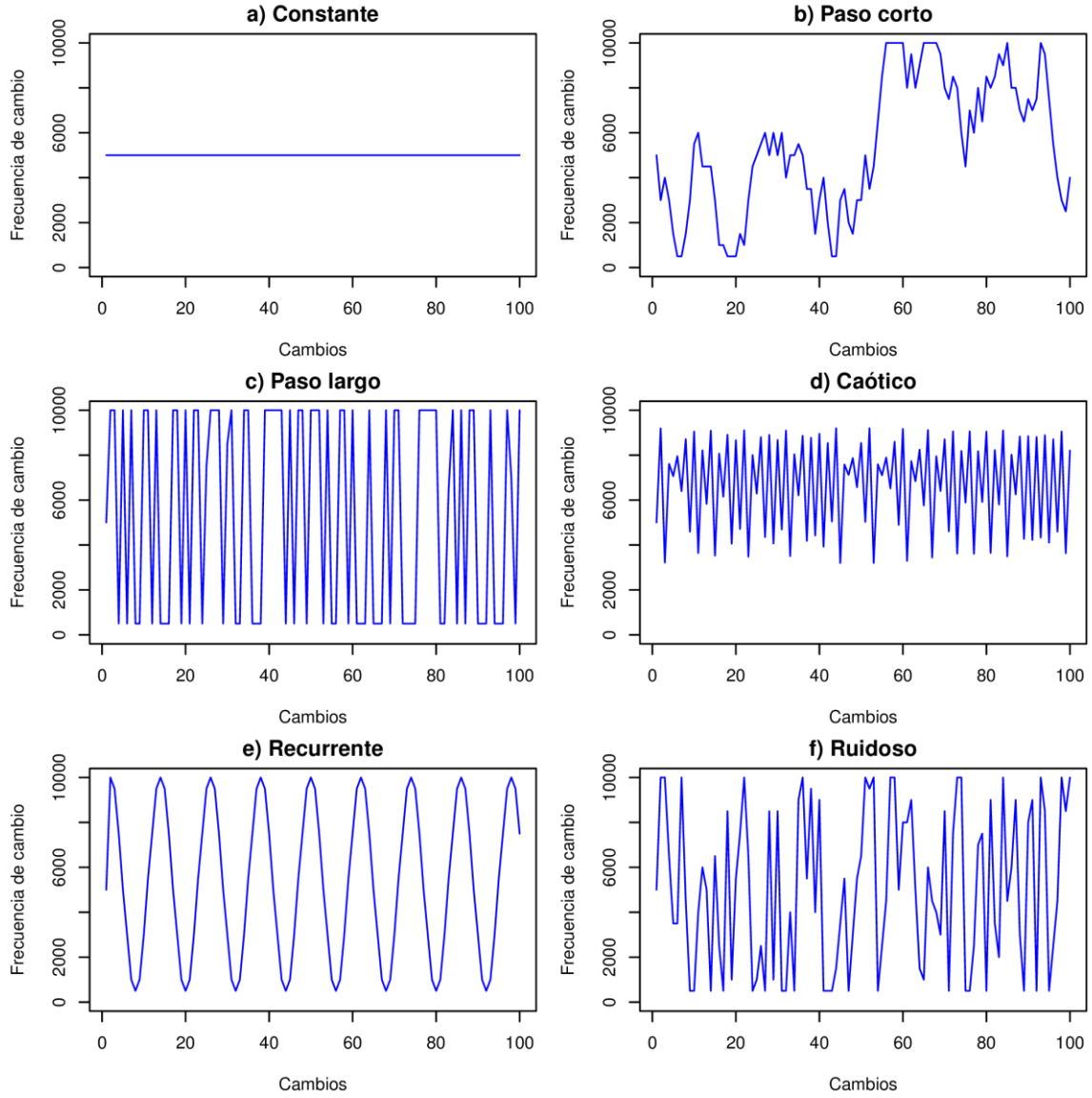

Figura 1. Tipos de cambios propuestos para la frecuencia de cambio (Δe). La gráfica a) corresponde al caso en el que no varía la frecuencia de los cambios, y es el considerado en la mayoría de las investigaciones sobre optimización en ambientes dinámicos.

3. ALGORITMOS AUTO-ADAPTATIVOS

Dado que el objetivo de la investigación es analizar la influencia de la auto-adaptación en la solución de PDOFCVs, resultaría apropiado elegir algoritmos que incluyan esta técnica de control de parámetros, y que al mismo tiempo hayan sido aplicados a PDOs. En ese sentido, como se muestra en la revisión hecha en [14], en la literatura existen varias investigaciones que proponen algoritmos auto-adaptativos en ambientes dinámicos. Sin embargo, son muy pocos en relación a los trabajos que no incluyen esta técnica, y solo algunos de estos aplican la auto-adaptación a mecanismos diseñados para ambientes dinámicos. Entre los que sí la aplican se

encuentran los propuestos en [15], los cuales son algoritmos multipoblacionales basados en el paradigma computacional Evolución Diferencial [20] (DE, por sus siglas en inglés). El enfoque multipoblacional empleado por estos algoritmos es básicamente el propuesto en [1], incluyendo también la generación de individuos quantum como estrategia de diversidad durante la ejecución. Como elemento novedoso, en [15] se le incorpora a este enfoque una estrategia de auto-adaptación que tiene por objetivo adaptar inteligentemente un parámetro relacionado con la generación de individuos quantum. Más específicamente, sea $q_i = \langle x_i, f_i, rc_i \rangle$ el individuo (solución) quantum i; donde $x_i \in X$ es el vector solución, $f_i = F(x_i)$ el valor de la función objetivo asociado a x_i, y rc_i el radio de la hiperesfera con centro en el mejor individuo g_{best}, donde será generado el individuo q_i; entonces en cada iteración, el parámetro rc_i es mutado de la forma siguiente:

$$\widetilde{rc_i} = \begin{cases} rand_1 \cdot \lambda \cdot r_{excl}, & si\ rand_2 \leq \tau \\ rc_i, & por\ el\ contrario \end{cases} \quad (9)$$

donde $\lambda \in (0,1)$ es un factor de escala y $\tau \in (0,1)$ es una tasa de mutación. Además, r_{excl} es el radio de exclusión entre subpoblaciones, mientras que $rand_1$ y $rand_2$ son números aleatorios generados de forma uniforme en el intervalo $(0,1)$. Los autores de esta estrategia auto-adaptativa consideraron dos formas de trasmitir los mejores valores de rc_i a las generaciones (iteraciones) futuras del algoritmo. La primera consiste en conservar el $\widetilde{rc_i}$ si la nueva solución generada empleando este radio, es mejor que la mejor solución de la subpoblación. La segunda, ocurre a nivel de individuo convencional. En este caso, se asume que cada subpoblación está compuesta por la misma cantidad de individuos convencionales y de tipo quantum. Entonces, $\widetilde{rc_i}$ es conservado si la nueva solución quantum es mejor que su individuo convencional asociado. En dicho caso, también se copia el vector solución del individuo quantum en su contraparte convencional. Este último esquema de interacción con individuos convencionales permite una mejor explotación en el proceso de búsqueda.

De manera que siguiendo la nomenclatura empleada en [15], los algoritmos que serán objeto de nuestro estudio son los siguientes:

- mQDE: sin estrategia de auto-adaptación, ni interacción con individuos convencionales,
- mSQDE: con estrategia de auto-adaptación pero sin interacción con individuos convencionales, y
- mSQDE-i: tanto con estrategia de auto-adaptación como interacción con individuos convencionales.

El lector puede notar que el algoritmo mQDE, al no poseer la estrategia de auto-adaptación servirá de control para la comparación con los otros dos métodos. En el Algoritmo 1 se ilustran los pasos generales que siguen los métodos considerados en nuestro estudio. El lector interesado podrá encontrar más detalles sobre este esquema en los trabajos [1] y [15].

Algoritmo 1. Pasos generales de los algoritmos considerados en la investigación. Los algoritmos difieren en la forma de generar los individuos quantum.

```
1  Inicializar todas las sub-poblaciones en el espacio de búsqueda.
2  Mientras condición de parada no alcanzada hacer
3  Aplicar principio de exclusión;
4  Detectar cambios;
5  Si no se ha detectado cambio entonces
6  Por cada sub-población hacer
7  Actualizar a los individuos convencionales según los pasos de Evolución Diferencial;
8  Generar individuos quantum de acuerdo al esquema considerado (e.g. original o auto-adaptativo);
9  Actualizar la mejor solución de la sub-población y del algoritmo;
   Fin
10 Si no
11 Por cada sub-población hacer
12 Reevaluar los individuos convencionales;
```

Fin

Fin

Fin

4. EXPERIMENTOS COMPUTACIONALES

En esta sección se describen los experimentos desarrollados en la investigación. Primeramente procederemos a explicar cómo se derivaron los escenarios de prueba empleados en el estudio. Más adelante se discuten los resultados obtenidos y se analizan estadísticamente.

4.1. Escenarios de prueba y configuración de los experimentos

Para obtener escenarios artificiales que representen PDOFCVs, una forma trivial es extender PDOs existentes mediante la inclusión de algún tipo de cambio como los definidos en las expresiones (5)-(9). En este sentido, se seleccionó el problema test *Movimiento de Picos* [2] (MPB, por sus siglas en inglés. Este problema *test* ha sido ampliamente utilizado en la literatura para analizar algoritmos en ambientes dinámicos. Concretamente, el escenario más utilizado del MPB es el denominado *Escenario 2*, el cual posee la configuración de parámetros mostrada en la Tabla 1. El lector interesado puede consultar [2] para más detalles sobre este problema *test*. Además, la Fig. 2 ilustra en dos dimensiones, el aspecto de la función objetivo que provee el MPB. Nótese que se trata de una función multimodal, formada por varios picos de distintas formas.
En particular, se derivaron un total de 18 instancias de este Escenario 2, a partir de la combinación de tres valores de severidad $sev = \{1.0, 5.0, 10.0\}$ y los tipos de cambios definidos en las expresiones (5)-(9), incluyendo el *Constante* (véase Figura 1-a)). El parámetro sev controla la magnitud del desplazamiento que experimentarán las posiciones de los picos (incluyendo el óptimo global del problema), y por tanto, influye de manera significativa en el rendimiento de los algoritmos. Entre mayor sea su valor, más difícil resultará para los algoritmos encontrar el óptimo desplazado.

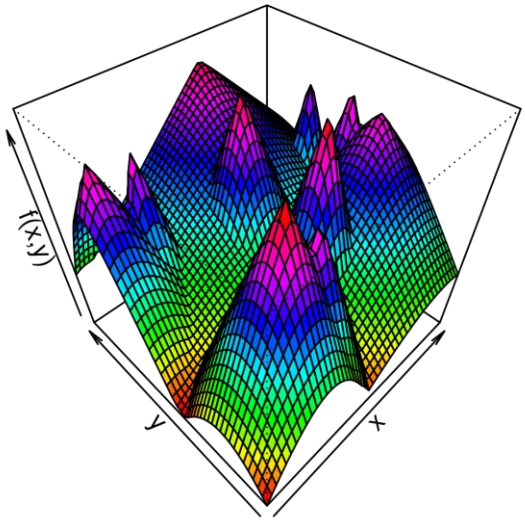

Figura 2. Función objetivo del problema test Movimiento de Picos[13]. La imagen corresponde al Escenario 2 en dos dimensiones.

Aunque en la actualidad existen diversas medidas de rendimiento para analizar los algoritmos en ambientes dinámicos (véase por ejemplo [11, 17, 21]), en la presente investigación emplearemos la media del *error de la mejor solución antes del cambio* [8]. Como su nombre indica, se trata de un error que caracteriza la capacidad del algoritmo en optimizar cada ambiente antes de la ocurrencia de un nuevo cambio. Nuestra elección se

basa principalmente en que esta medida informa de manera efectiva el rendimiento entre etapas del algoritmo, aprovechándose el hecho de que al utilizar un problema test, es posible conocer en todo el momento la solución óptima del problema. Formalmente esta medida se define como:

$$e_{msac} = \frac{1}{C-1}\sum_{c=1}^{C}\left|f_{opt}^{c} - f_{best}^{c}\right| \quad (9a)$$

donde C es el número de cambios que experimenta el problema en cada ejecución, y c es el cambio actual. Por otro lado, f_{opt}^{c} y f_{best}^{c} son los valores de la función objetivo de, la solución óptima del problema y, de la mejor solución del algoritmo en el cambio (ambiente) c. Es importante advertir que, dado que esta medida conceptualmente representa un error, un valor bajo de la misma indica un buen rendimiento del algoritmo en cuestión.

De manera general, se desarrollaron 20 ejecuciones independientes por cada par problema-algoritmo, empleando semillas aleatorias diferentes. La configuración de parámetros de los algoritmos seleccionados, es la misma empleada por los autores de estos [15]: 10 sub-poblaciones, donde cada una posee 5 individuos convencionales y 5 de tipo quantum. Los parámetros relacionados con el paradigma Evolución Diferencial, como son la tasa de cruzamiento (CR) y el factor de mutación (F), fueron establecidos en 0.1 y 1.0, respectivamente. Asimismo, en cuanto a los parámetros de la estrategia de auto-adaptación se emplearon los siguientes: $\tau = 0.5$ y $\lambda = 0.3$. Finalmente, los tipos de cambios utilizados para crear los escenarios artificiales a partir del MPB, son los descritos en la Figura 1. En cada ejecución, el problema experimentará 200 cambios. Tanto los algoritmos como las instancias del MPB, fueron implementados en el software MATLAB 2015[a], empleando como plataforma un ordenador con sistema operativo Debían 8.1, 12GB de RAM y un procesador Intel i7 de 4ta generación.

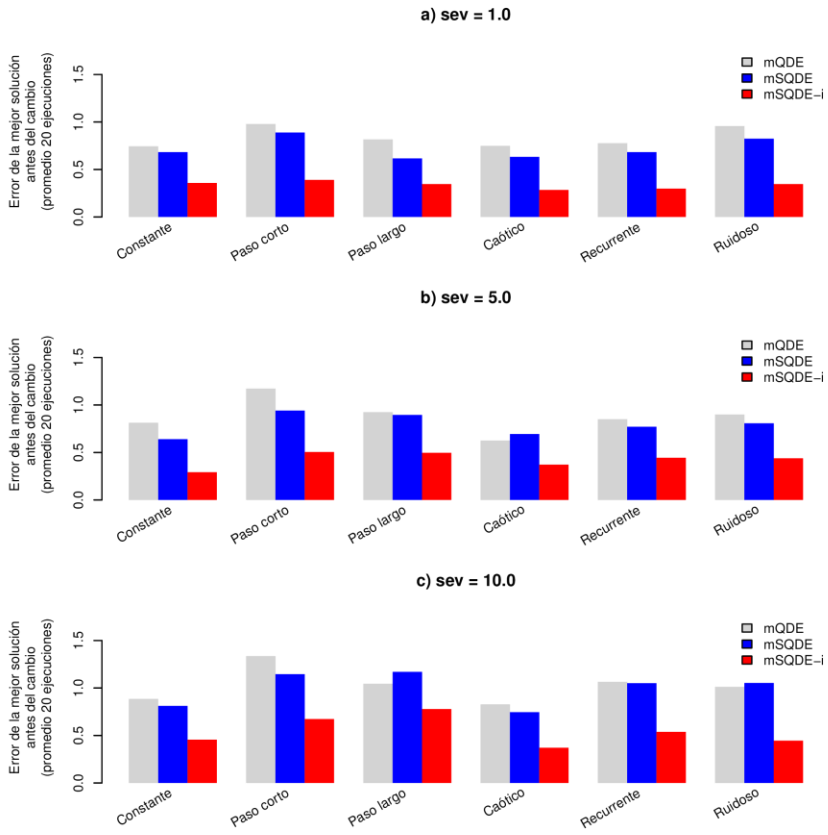

Figura 3. Resultados, en términos de la media del error de la mejor solución antes del cambio, de los algoritmos mQDE, mSQDE, y mSQDE-i en distintas instancias de problema según la severidad de los cambios (sev) y frecuencia de cambio (Δe).

Tabla 1. Configuración de parámetros correspondiente al Escenario 2 del problema test Movimiento de Picos.

Parámetro	Configuración
Dimensión	$D = 5$

Espacio de búsqueda	$\mathrm{X} = [0,100]^5$
Números de picos	$n = 10$
Función de los picos	$f_{cone}(x) = \|x\|_2$
Severidad de cambio de las posiciones de los picos	$sev = 1.0$
Altura de los picos	$H_i \in [30.0,70.0]$
Anchura de los picos	$W_i \in [1.0,12.0]$
Severidad de cambio de la altura de los picos	$H_{sev} = 7.0$
Severidad de cambio de la anchura de los picos	$W_{sev} = 1.0$
Factor de correlación de los cambios	$\gamma = 0.0$

4.2. Resultados y análisis estadístico

Los resultados de los algoritmos en las instancias de problema del MPB, son mostrados en la Figura 3, en términos de la media de e_{msac} a partir de 20 ejecuciones. Nótese que cada gráfica agrupa los resultados según el valor de la severidad (sev) empleado por las instancias de problema. Por ejemplo, en la gráfica a) se muestran los resultados de los algoritmos en instancias de problema con severidad igual a 1.0, y con tipos de cambio *Constante, Paso corto, Paso largo, Caótico, Recurrente y Ruidoso.*

Como se aprecia en la Fig. 3, los algoritmos que incluyen auto-adaptación (mSQDE y mSQDE-i) son en general, mejores que el algoritmo mQDE. Esta superioridad se evidencia incluso para el escenario *Constante*, lo cual está en concordancia con los resultados obtenidos en [15]. En el caso del algoritmo más sofisticado, esta superioridad es mucho más significativa, dado que se logran mejoras por encima del 50% con respecto a mQDE (ej. instancias con tipo de cambio *Paso Corto*). En cuanto al algoritmo mSQDE, en la mayoría de las instancias se mantiene en la segunda posición, excepto en dos casos: instancia con $sev = 5.0$ y tipo de cambio *Caótico* (Figura 3-b), e instancia con $sev = 10.0$ y tipo de cambio *Paso largo* (Fig. 3-c). Lo anterior indica que la estrategia de interacción entre individuos, incluida en mSQDE-i, es realmente efectiva en todas las instancias consideradas. En sentido general, nótese que las instancias que más afectan el rendimiento de los algoritmos son las que incluyen tipos de cambio *Paso corto*, y *Ruidoso*. Una posible razón de este comportamiento es que ambos tipos de cambios provocan que el algoritmo se enfrente a ambientes con frecuencia de cambios bajas de manera seguida. Es importante notar que una frecuencia de cambio baja significa que el algoritmo cuenta con poco tiempo para optimizar el ambiente actual, por lo que su rendimiento tiende a deteriorarse.

Para complementar este análisis, en la Fig. 4, se ilustra la evolución en el tiempo, de la media del error de la mejor solución del algoritmo. Las instancias de problema utilizadas para estas gráficas son las que poseen $sev = 5.0$. Como se advertía al inicio, en escenarios con frecuencia de cambio constante, los algoritmos tienden a comportarse de manera oscilatoria, y por tanto, es de esperar un rendimiento similar entre ambientes. Véase en este sentido la Fig. 4-a), en la que aparecen picos de manera regular, indicando la presencia de cambios en el problema. En contraste con este caso, el resto exhiben patrones menos regulares.

Finalmente, con el objetivo de confirmar estadísticamente la superioridad de los algoritmos con auto-adaptación frente al que no la posee, se procedió con la metodología sugerida en [6], tomando como conjunto de datos los resultados mostrados en la Figura 2. En este sentido, primeramente se aplicó la prueba de Friedman para detectar diferencias a nivel de grupo, la cual arrojó un p-valor igual a 1.86×10^{-7}, esto es, inferior a 0.05 (α, nivel de significación). Esto indica que efectivamente, sí existen diferencias significativas entre los algoritmos. Como consecuencia, lo apropiado es proseguir con pruebas post-hoc (ej. prueba de Holm), con el objetivo de detectar entre que pares de algoritmos existen tales diferencias. La Figura 5 muestra los rangos promedio obtenidos por los algoritmos según Friedman, y los p-valores ajustados según de la prueba de Holm, correspondientes a las comparaciones múltiples entre algoritmos. Téngase en cuenta que un p-valor inferior a 0.05, indica que los algoritmos son diferentes entre sí.

Como se aprecia en estos resultados, se puede concluir que los tres algoritmos son significativamente diferentes entre sí. Es importante notar, no obstante, que la diferencia entre el algoritmo mQDE y mSQDE es menor que la de estos con respecto a mSQDE-i. El significado de estos resultados indica que la auto-adaptación tiene una influencia muy positiva en la solución de problemas dinámicos con frecuencia de cambio variable. Su capacidad de dotar al algoritmo de un comportamiento inteligente durante la ejecución, sin dudas influye de manera muy significativa en su rendimiento.

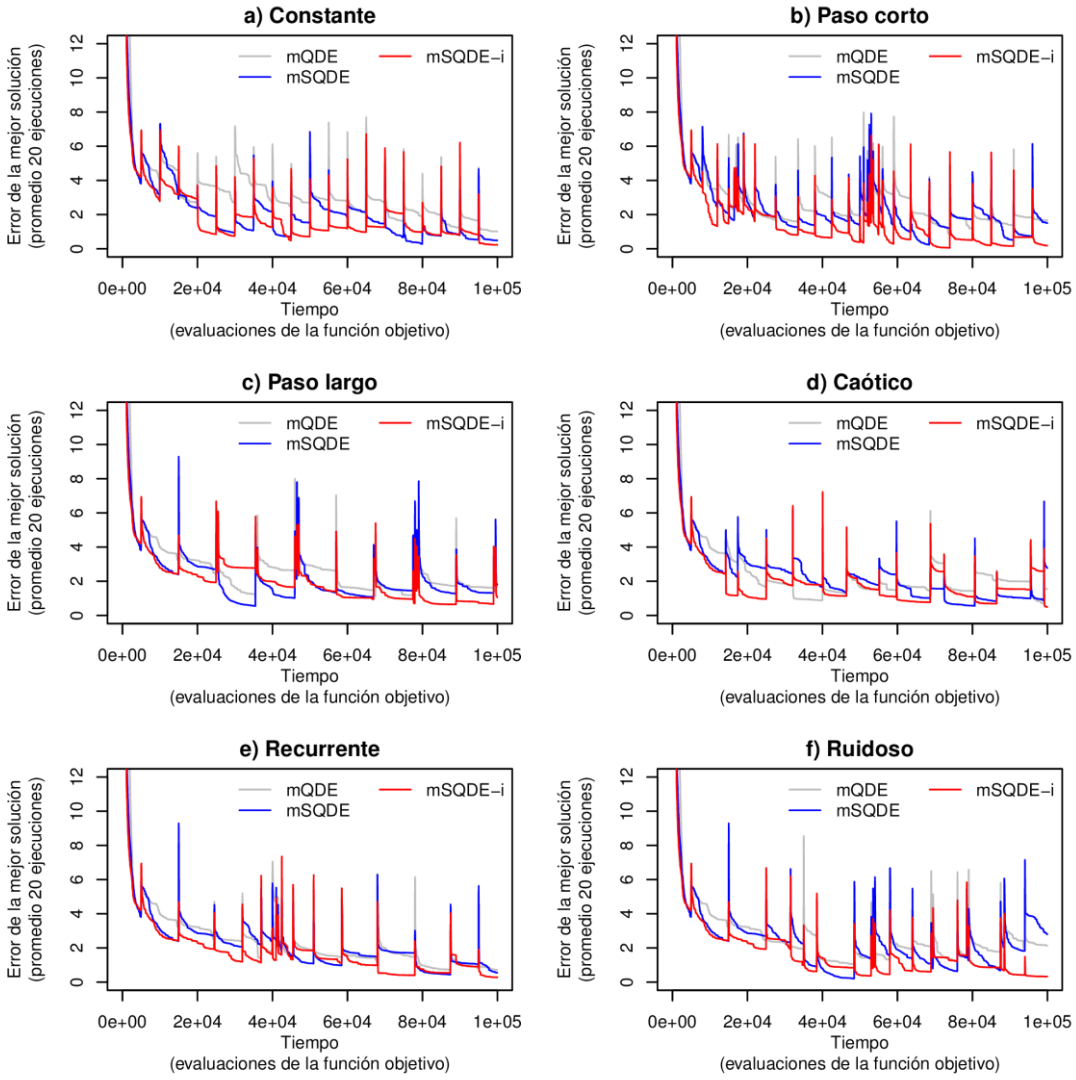

Figura 4. Evolución en el tiempo del error de la mejor solución para instancias de problema con $\boldsymbol{sev = 5.0}$

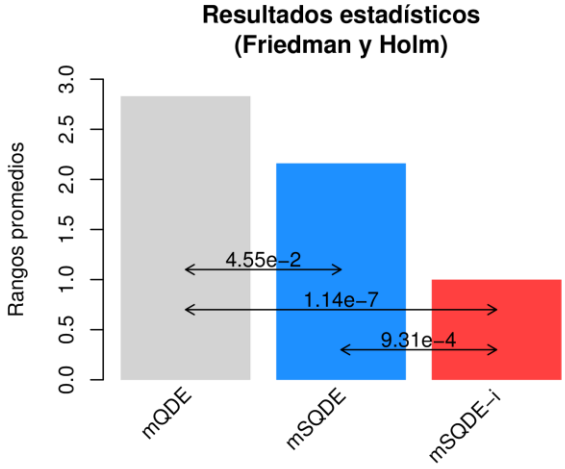

Figura 5. Resultados estadísticos obtenidos a partir de la prueba de Friedman (rangos promedios) y la prueba de Holm (p-valores entre algoritmos).

5. CONCLUSIONES

En este trabajo se presentó un estudio experimental con el objetivo de analizar el impacto de la auto-adaptación, como técnica de control de parámetros, en la solución de problemas de optimización con frecuencia de cambio variable. Los resultados de los experimentos en problemas con frecuencia de cambio que varían de diferentes formas, muestran que los algoritmos que incluyen esta técnica de control de parámetro superan en rendimiento al algoritmo tomado como base, no auto-adaptativo.
No obstante los resultados obtenidos, en nuestra opinión, se debe profundizar en el estudio de este tipo de problema. En especial, abordando tanto nuevos escenarios como otros algoritmos que han sido aplicados en ambientes dinámicos. Por ejemplo, algoritmos que se basan en esquemas de memoria [9, 22], para *aprender* elementos del ambiente, pudieran ser aplicados con éxito aquí. Además, recientemente se ha propuesto en [13] una aplicación que facilita la experimentación en ambientes dinámicos. De manera que para motivar el estudio en este tipo de problemas, sería de gran utilidad incluir en esta herramienta los problemas test diseñados en este trabajo. Nuestros trabajos futuros estarán orientados al desarrollo de estos y otros temas similares.

RECEIVED DECEMBER 2015
REVISED MARCH 2016

REFERENCIAS

[1] BLACKWELL, T. M. AND BRANKE, J. (2006): Multiswarms, exclusion, and anti-convergence in dynamic environments. **IEEE Transactions on Evolutionary Computation**, 10,459–472.

[2] BRANKE, J. (1999): Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems In Angeline, P. J., Michalewicz, Z., Schoenauer, M., Yao, X., and Zalzala, A., editors, **Proceedings of the Congress on Evolutionary Computation**, volume 3, 1875–1882, Mayflower Hotel, Washington D.C., USA. IEEE Press.

[3] BREST, J., ZAMUDA, A., BOSKOVIC, B., MAUCEC, M. S., AND ZUMER, V. (2009): Dynamic optimization using selfadaptive differential evolution In **CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation**, 415–422, Piscataway, NJ, USA. IEEE Press.

[4] CRUZ, C., GONZÁLEZ, J. R., AND PELTA, D. (2011): Optimization in dynamic environments: a survey on problems, methods and measures. **Soft Computing**, 15,1427–1448.

[5] DU PLESSIS, M. C. AND ENGELBRECHT, A. P. (2013): Self-Adaptive Differential Evolution for Dynamic Environments with Fluctuating Numbers of Optima In Alba, E., Nakib, A., and Siarry, P., editors, **Metaheuristics for Dynamic Optimization** , volume 433 of Studies in Computational Intelligence , 117–145. Springer Berlin Heidelberg.

[6] GARCÍA, S., MOLINA, D., LOZANO, M., AND HERRERA, F. (2009): A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. **J Heuristics**, 15,617–644.

[7] JIN, Y. AND BRANKE, J. (2005): Evolutionary optimization in uncertain environments-a survey. **IEEE Transactions on Evolutionary Computation**, 9, 303–317.

[8] LI, C., YANG, S., NGUYEN, T. T., YU, E. L., YAO, X., JIN, Y., BEYER, H.-G., AND SUGANTHAN, P. N. (2008): Benchmark Generator for CEC'2009 Competition on Dynamic Optimization **Technical report, Department of Computer Science**, University of Leicester, U.K.

[9] MAVROVOUNIOTIS, M. AND YANG, S. (2012): Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem In Li, X., editor, **Proceedings of the 2012 IEEE Congress on Evolutionary Computation**, 2766–2773, Brisbane, Australia.

[10] MEYER-NIEBERG, S. AND BEYER, H.-G. (2007): Self-Adaptation in Evolutionary Algorithms In Lobo, F., Lima, C., and Michalewicz, Z., editors, **Parameter Setting in Evolutionary Algorithms** , volume 54 of **Studies in Computational Intelligence** , 19–46. Springer Berlin / Heidelberg.

[11] NGUYEN, T. T., YANG, S., AND BRANKE, J. (2012): Evolutionary dynamic optimization: A survey of the state of the art. **Swarm and Evolutionary Computation** , 6, 1–24.

[12] NOVOA-HERNÁNDEZ, P., CORONA, C., AND PELTA, D. (2011): Efficient multi-swarm PSO algorithms for dynamic environments. **Memetic Computing** , 3, 163–174.

[13] NOVOA-HERNÁNDEZ, P., CRUZ CORONA, C., AND PELTA, D. (2015a): A Software Tool for Assisting Experimentation in Dynamic Environments. **Applied Computational Intelligence and Soft Computing** , 2015 (Article ID302172):12p.

[14] NOVOA-HERNÁNDEZ, P., CRUZ CORONA, C., AND PELTA, D. (2016): Self-adaptation in dynamic

environments – a survey and open issues. **International Journal of Bio-inspired Computation**, 8,1-13.
[15] NOVOA-HERNÁNDEZ, P., CRUZ CORONA, C., AND PELTA, D. A. (2013): Self-adaptive, multipopulation differential evolution in dynamic environments. **Soft Computing** , 17, 1861–1881.
[16] NOVOA-HERNÁNDEZ, P., PELTA, D., AND CORONA, C. (2010): Improvement Strategies for Multi-swarm PSO in Dynamic Environments In Gonz´alez, J., Pelta, D., Cruz, C., Terrazas, G., and Krasnogor, N., editors, **Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)** , volume 284 of **Studies in Computational Intelligence** , 371–383. Springer Berlin / Heidelberg.
[17] PELTA, D. A., CRUZ, C., AND VERDEGAY, J. L. (2009): Simple control rules in a cooperative system for dynamic optimization problems. **International Journal of General Systems** , 38, 701–717.
[18] SIMÕES, A. AND COSTA, E. (2008): Evolutionary algorithms for dynamic environments: Prediction using linear regression and markov chains In Rudolph, G., Jansen, T., Beume, N., Lucas, S., and Poloni, C., editors, **Parallel Problem Solving from Nature – PPSN X ,** volume 5199 of **Lecture Notes in Computer Science** , 306–315. Springer Berlin Heidelberg.
[19] SIMÕES, A. AND COSTA, E. (2009): Improving prediction in evolutionary algorithms for dynamic environments. In **Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation , GECCO '09**, 875–882, New York, NY, USA. ACM.
[20] STORN, R. AND PRICE, K. (1997): Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. **Journal of Global Optimization**, 11, 341-359.
[21] WEICKER, K. (2002): Performance Measures for Dynamic Environments In Guervós, J. J. M., Adamidis, P., Beyer, H.-G., Schwefel, H.-P., and Fernández-Villacañas, J.-L., editors, **Parallel Problem Solving from Nature PPSN VII** , volume 2439 of **Lecture Notes in Computer Science** , 64–73. Springer Berlin Heidelberg.
[22] YANG, S., CHENG, H., AND WANG, F. (2010): Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks**. IEEE Trans Syst Man Cybernet C: Appl Rev.**, 40, 52-63.