Configuration of Harmony Search Parameters based on Fuzzy Logic

Alejandro Centeno[a], Alejandro Bolívar[b], Demetrio Rey[a], Francisco Arteaga[c], César Séijas[d], Ángel D. Almarza M.*[,b]

[a]*Instituto de Matemática y Cálculo Aplicado, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela.*
[b]*Departamento de Computación, Dirección de Estudios Básicos, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela.*
[c]*Centro de Investigación y Tecnología en Automatización, Electrónica y Control. Departamento de Sistemas y Automática, Escuela de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela.*
[d]*Centro de Procesamiento de Imágenes, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela.*

Abstract.- The performance quality of any metaheuristic algorithm exhibits a close relationship with respect to the values assigned to its parameters. This is known as the configuration problem. In this work, a proposal for the configuration of the Harmony Search Algorithm parameters based on Fuzzy Logic is exposed. As a practical application, the proposal is used for the supervised training of a multilayer neural network in order to solve a binary classification problem.

Keywords: harmony search algorithm; fuzzy logic; optimization.

Configuración de los parámetros de la búsqueda armónica mediante lógica difusa

Resumen.- La calidad del desempeño de cualquier algoritmo meta–heurístico exhibe una estrecha relación con respecto a los valores asignados a sus parámetros asociados. Esto se conoce como el problema de la configuración. En el presente artículo, se expone la implementación de una propuesta basada en lógica difusa para la configuración de los parámetros numéricos del algoritmo de búsqueda armónica. Como aplicación práctica, la propuesta es utilizada para el entrenamiento supervisado de una red neuronal multicapa a fin de resolver un problema de clasificación binaria.

Palabras claves: algoritmo de búsqueda armónica; lógica difusa; optimización.

Recibido: 02 noviembre 2018
Aceptado: 08 febrero 2019

1. Introducción

En el campo de la optimización matemática, las meta – heurísticas han demostrado reiteradamente sus bondades y capacidades como métodos numéricos aplicados en la resolución de una amplia colección de modelos [1]. No obstante su éxito, es un hecho bien conocido que la calidad del desempeño de estos algoritmos guarda estrecha relación con respecto a los valores asignados a sus parámetros asociados. Esto se conoce como el problema de la configuración [2], el cual puede ser abordado mediante la implementación de mecanismos fundamentados en un conjunto de reglas que gobiernen tal asignación de forma dinámica, durante la ejecución de una meta – heurística, con el propósito de maximizar el desempeño de la misma. Este esquema de solución se denomina control de parámetros [3]. Un ejemplo de tal esquema consiste en la aplicación de un Sistema de Inferencia Difuso (SID), donde las herramientas brindadas por la lógica difusa son utilizadas para establecer una correspondencia entre el estado del proceso de optimización (variables de entrada) y los valores asignados a los parámetros (variables de salida) de la meta – heurística que lo ejecuta [4]. Este concepto fue implementado recientemente en [5] sobre el Algoritmo de Búsqueda Armónica (ABA) [6] mediante un SID basado en reglas de Mamdani (SIDM) [7]. Trabajos similares han sido reportados

*Autor para correspondencia:
*Correo-e:*adalmarza@gmail.com (Ángel D. Almarza M.)

en [8, 9, 10, 11]. No obstante, sin desconocer sus valiosos aportes, todas estas investigaciones guardan en común el hecho de haber establecido un control sobre un subconjunto de los parámetros del ABA. En el presente artículo se exponen los resultados derivados de la implementación de un SIDM para la configuración dinámica de todo el ensamble de parámetros numéricos del ABA. Se formularon dos (02) versiones, denotadas en lo sucesivo como ABAD01 y ABAD02, diferenciadas solo por la variable de entrada utilizada. En ambas versiones, las variables de interés fueron granuladas mediante funciones de pertenencia de formato trapezoidal. Para propósitos de comparación, ABAD01, ABA y [5], fueron aplicadas sobre una colección de problemas de optimización no lineal. Por su parte, ABAD02 se implementó para el entrenamiento supervisado de una Red Neuronal Multicapa (RNM) a fin de resolver un problema de clasificación binaria. Los resultados obtenidos de este experimento fueron comparados con los derivados a partir de la aplicación del ABA, el Algoritmo de Retro – propagación del Error con Momento (ARPEM) [12] y [5] sobre el mismo problema.

2. Parámetros numéricos del algoritmo de búsqueda armónica (ABA)

La estructura implementada del ABA tiene asociada los siguientes parámetros numéricos [6]:

2.1. Tasa de selección en la memoria de armonías

Denotada HMCR (*Harmony Memory Chose Rate*) por sus siglas en inglés, representa la probabilidad asociada al evento de seleccionar una entrada en la memoria de armonías. Para la configuración dinámica, la estrategia asumida consistió en asignar valores bajos a HMCR durante las primeras etapas del proceso de optimización ejecutado por el ABA, transitando luego hacia valores altos durante las etapas finales [5, 11, 13, 14]. Entre los valores típicamente recomendados para este parámetro, se tiene: HMCR $\in$ [0,95; 0,99] [13] y HMCR $\geq$ 0,90 [15].

2.2. Tasa de ajuste de tono

Denotada PAR (*Pitch Adjusting Rate*) por sus siglas en inglés, representa la probabilidad asociada al evento de aplicar una variación sobre la entrada previamente seleccionada en la memoria de armonías. En cuanto a la configuración dinámica, se adoptó la estrategia de asignar valores altos a PAR durante las primeras etapas del proceso de optimización, para luego transitar hacia valores bajos durante las etapas finales [16, 17, 18, 19]. Entre los valores típicamente recomendados para este parámetro, se tiene: PAR $\in$ [0,1; 0,9] [16] y PAR $\leq$ 0,5 [15].

2.3. Ancho de banda

Denotado BW (*Band Width*) por sus siglas en inglés, representa la magnitud de la variación aplicada sobre una entrada previamente seleccionada en la memoria de armonías. En cuanto a su configuración dinámica, la estrategia adoptada es idéntica a la descrita para PAR [9, 13, 14, 15, 16, 20, 21]. Los valores típicos recomendados para BW están comprendidos entre el 0,1 y 1,0 por ciento del rango de valores permitidos para las variables asociadas al problema de optimización tratado [16].

2.4. Tamaño de la memoria de armonías

Denotado HMS (*Harmony Memory Size*) por sus siglas en inglés, representa el número de soluciones almacenadas en la memoria de armonías. Para su configuración dinámica se adoptó la misma estrategia de PAR [22, 23]. Entre los valores típicamente recomendados para este parámetro, se tiene: HMS $\in$ [1; 10] [16] y HMS $\in$ [4; 10] [24].

3. Propuesta para la configuración de parámetros

Se implementó un SIDM, cuyos elementos principales se detallan a continuación:

3.1. Variables de entrada

Denotada λ, para ABAD01 viene dada por el cociente entre el número de iteraciones k y el número de iteraciones máximas k^{max}, de acuerdo a la ecuación (1):

$$\lambda^{(k)} = \frac{k}{k^{max}} \quad (1)$$

Para ABAD02, se computa mediante el cociente entre el mejor desempeño $\left(f_{mejor}^{(k)}\right)$ exhibido por las soluciones almacenadas en la memoria de armonías para la k – ésima iteración, y el mejor desempeño inicial $\left(f_{mejor}^{(0)}\right)$, de acuerdo a la ecuación (2):

$$\lambda^{(k)} = 1 - \frac{f_{mejor}^{(k)}}{f_{mejor}^{(0)}} \qquad (2)$$

3.2. *Variables de salida*

Parámetros del ABA, con redondeo de HMS al entero más cercano.

3.3. *Fusificación*

Parámetros del ABA, con redondeo de HMS al entero más cercano. Todas las variables se granularon mediante tres (03) conjuntos difusos, con etiquetas "Bajo", "Medio" y "Alto", caracterizados por las funciones de pertenencia de la Figura 1. Los valores asignados a los puntos de interés esquematizados se listan en la Tabla 1.

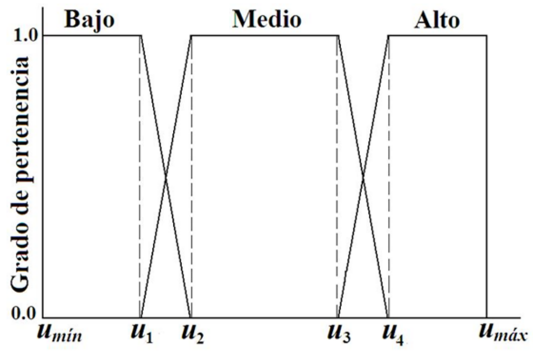

Figura 1: Formato de las funciones de pertenencia implementadas.

3.4. *Base de conocimiento*

Se aplicaron las reglas mostradas en la Tabla 2.

3.5. *Desfusificación*

Idéntico al método utilizado en [5].

4. Implementación de la propuesta

4.1. *Estructura algorítmica*

La integración se realizó de acuerdo al modelo de capas descrito en [3], alojando al SIDM en la capa de diseño.

Tabla 1: Valores asignados a los puntos de interés de acuerdo con el formato de las funciones de pertenencia implementadas

P	u_{min}	u_1	u_2	u_3	u_4	u_{max}
λ	0	0,1	0,15	0,25	0,3	1
HMCR	0,9	0,909	0,9135	0,9225	0,927	0,99
PAR	0,1	0,14	0,16	0,2	0,22	0,5
BW	$1{,}0{\cdot}10^{-3}$	$1{,}9{\cdot}10^{-3}$	$2{,}35{\cdot}10^{-3}$	$3{,}25{\cdot}10^{-3}$	$3{,}7{\cdot}10^{-3}$	$1{,}0{\cdot}10^{-2}$
HMS	4	4,6	4,9	5,5	5,8	10

P:Parámetro

Tabla 2: Reglas Si – Entonces aplicadas a las variables de entrada (λ).

Etiqueta	HMCR	PAR	BW	HMS
Bajo	Bajo	Alto	Alto	Alto
Medio	Medio	Medio	Medio	Medio
Alto	Alto	Bajo	Bajo	Bajo

4.2. *Algoritmos implementados*

Para la resolución de los problemas de optimización no lineal se implementaron los algoritmos ABAD01, ABA [6] y FHS (*Fuzzy Harmony Search* por sus siglas en inglés) [5]. Los valores asignados a los parámetros de estos algoritmos fueron consultados en [5, 17, 18] y se muestran en la Tabla 3.

Tabla 3: Valores asignados a los parámetros de los algoritmos aplicados sobre los problemas de optimización.

Algoritmo	HMCR	PAR	BW	HMS
ABAD01	D	D	D	D
ABA	0,90	0,30	0,01	5
FHS	D	0,75	0,01	5

D:Dinámico

Para el problema de clasificación binaria, se implementaron ABAD02, ABA y FHS. Los valores asignados a los parámetros de estos algoritmos fueron consultados en [25] y se muestran en la Tabla 4.

El universo del discurso de HMCR para FHS se estableció en . Adicionalmente, se implementó

Tabla 4: Valores asignados a los parámetros de los algoritmos aplicados sobre el problema de clasificación.

Algoritmo	HMCR	PAR	BW	HMS	Rango para pesos sinápticos
ABAD02	D	D	D	D	[-10,0 10,0]
ABA	0,9	0,3	0,01	10	[-10,0 10,0]
FHS	Dinámico	0,3	0,01	10	[-10,0 10,0]

D:Dinámico

el ARPEM [12] con tasa de aprendizaje igual a 0,008 y momento de 0,7. Estos valores se ajustaron empíricamente tratando de maximizar la velocidad de convergencia y eliminar oscilaciones en la curva de entrenamiento.

4.3. *Problemas tratados*

Se resolvieron problemas de optimización no lineal del tipo $min f(\vec{x})$ sujeto a $\vec{x} \in S \subset R^n$, donde $f(\vec{x})$ es la función objetivo, S representa el espacio de búsqueda y n el número de variables intervinientes. Los problemas tratados fueron consultados en [26], `http://infinity77.net/global_optimization/index.html` y `http://www.sfu.ca/~surjano/optimization.html`, siendo seleccionadas catorce instancias en atención a la modalidad y carácter separable de $f(\vec{x})$[26], tal y como se resume en las Tablas 5, 6, 7 y 8 mostradas a continuación.

Para el problema de clasificación se utilizó la base de datos Pima Diabetes (`http://archive.ics.uci.edu/ml`), eliminándose las entradas con datos faltantes [27] y aplicando normalización estadística [28]. La fase de entrenamiento por lotes se realizó en base al 60 por ciento de los datos con validación sobre el resto [27], cuantificándose el error y la tasa de acierto o clasificaciones correctas obtenidas según las expresiones planteadas en [25]. La RNM implementada se configuró de forma idéntica a [29].

4.4. *Esquema de representación*

Para todos los problemas tratados, los datos se estructuraron mediante arreglos unidimensionales con esquema de codificación uno a uno de tipo punto flotante de doble precisión. Respecto al problema de clasificación, en la Figura 2 se muestra un ejemplo de la correspondencia entre pesos sinápticos y su representación para una red hipotética de dos (02) capas. Esta estructura se utilizó únicamente para los algoritmos meta – heurísticos.

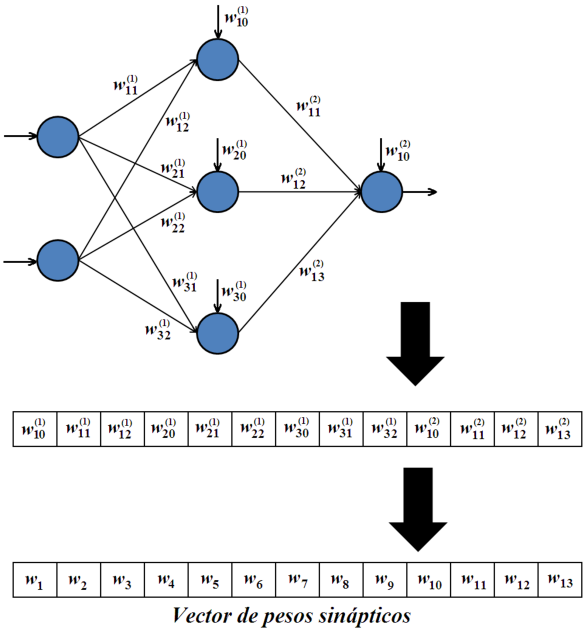

Figura 2: Representación de pesos sinápticos de una red hipotética de dos (02) capas.

4.5. *Condiciones experimentales*

Se efectuaron 50 ensayos independientes para cada par algoritmo/problema, fijándose el número de iteraciones máximas en $5 \cdot 10^4$, para los problemas listados en las Tablas 5, 6, 7 y 8 con $n = 30$, y $1 \cdot 10^4$ épocas para el problema de clasificación. En todos los algoritmos implementados, la estrategia para la asignación de valores al conjunto de soluciones de partida fue aleatoria con distribución uniforme sobre para los problemas de las Tablas 5 a 8, y entre -1 y 1 para el problema de clasificación. Para todos los algoritmos el criterio de parada se estableció en función del cumplimiento del número de iteraciones máximas.

Tabla 5: Instancias de problemas de optimización de tipo multimodal y separable.

$f(\vec{x})$	S
$f_1(\vec{x}) = -20 + e^{0,2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{-\sum_{i=1}^{n}\cos(2\pi x_i)}{n}}$	$[-35,0\ \ 35,0]^n$
$f_5(\vec{x}) = \sin^2\left(\pi\left(1+\frac{x_1-1}{4}\right)\right) + \frac{1}{16}\sum_{i=1}^{n-1}\left[(x_i-1)^2\left(1+10\sin^2\left(\pi\left(1+\frac{x_{i+1}-1}{4}\right)\right)\right)\right] + \frac{(x_n-1)^2}{16}$	$[-10,0\ \ 10,0]^n$
$f_{10}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-100,0\ \ 100,0]^n$

Tabla 6: Instancias de problemas de optimización de tipo multimodal y no separable.

$f(\vec{x})$	S
$f_2(\vec{x}) = \sum_{i=1}^{n-1}\left[x_i^2 + 2x_{i+1}^2 - 0,3\cos(3\pi x_i) - 0,4\cos(4\pi x_{i+1}) + 0,7\right]$	$[-15,0\ \ 15,0]^n$
$f_4(\vec{x}) = \sum_{i=1}^{n}\left[x_i^6\left(2+\sin\left(\frac{1}{x_i}\right)\right)\right]$	$[-1,0\ \ 1,0]^n$
$f_9(\vec{x}) = 418,982887272434\cdot n - \sum_{i=1}^{n} x_i\sin\left(\sqrt{\lvert x\rvert}\right)$	$[-500,0\ \ 500,0]^n$
$f_{11}(\vec{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$[-5,12\ \ 5,12]^n$

Tabla 7: Instancias de problemas de optimización de tipo unimodal y no separable.

$f(\vec{x})$	S
$f_3(\vec{x}) = \sum_{i=1}^{n-1}\left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)} + \left(x_{i+1}^2\right)^{\left(x_i^2+1\right)}\right]$	$[-1,0\ \ 4,0]^n$
$f_8(\vec{x}) = \sum_{i=1}^{n} x_i^2 + \left[\frac{1}{2}\sum_{i=1}^{n} i x_i\right]^2 + \left[\frac{1}{2}\sum_{i=1}^{n} i x_i\right]^4$	$[-5,0\ \ 10,0]^n$
$f_{12}(\vec{x}) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2 + (x_i-1)^2\right]$	$[-30,0\ \ 30,0]^n$
$f_{14}(\vec{x}) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	$[-100,0\ \ 100,0]^n$

Tabla 8: Instancias de problemas de optimización de tipo unimodal y separable.

$f(\vec{x})$	S
$f_6(\vec{x}) = \sum_{i=1}^{n}\left[\sum_{j=1}^{i} x_j^2\right]$	$[-65,536\ \ 65,536]^n$
$f_7(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$[-100,0\ \ 100,0]^n$
$f_{13}(\vec{x}) = \sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	$[-1,0\ \ 1,0]^n$

5. Resultados y análisis

En las Figuras 3, 4, 5 y 6, se presentan diagramas de caja y bigotes elaborados a partir del valor de $f(\vec{x})$ proporcionado por cada algoritmo al término de su ejecución.

A partir de los datos mostrados en las Figuras 3, 4, 5 y 6, se computaron los siguientes índices [30]: mejor valor, peor valor, mediana, promedio y desviación típica. Los resultados obtenidos se listan en las Tablas 9, 10, 11 y 12 mostradas a continuación.

Se aplicó una prueba de Wilcoxon en base al índice promedio de las Tablas 9 a 12, tomando

Tabla 9: Índices estadísticos computados para problemas de optimización de tipo multimodal y no separable.

Problema	Algoritmo	Mejor Valor	Mediana	Peor Valor	Promedio	Desviación Típica
f_1	ABA	8,7073E-1	1,3344E+0	1,6069E+0	1,2921E+0	1,6168E-1
	FHS	2,4988E+0	2,8520E+0	3,2407E+0	2,8726E+0	1,6872E-1
	ABAD01	1,7479E-3	3,9784E-3	9,3139E-1	2,4891E-2	1,3240E-1
f_5	ABA	2,4755E-3	4,1418E-3	6,0103E-3	4,0886E-3	7,2810E-4
	FHS	1,4469E-2	2,0834E-2	2,8424E-2	2,1358E-2	3,4596E-3
	ABAD01	1,2951E-8	7,5596E-8	3,1799E-7	8,2626E-8	4,7443E-8
f_{10}	ABA	1,8009E-1	2,4002E-1	3,6387E-1	2,4230E-1	3,7717E-2
	FHS	6,0196E-1	7,9784E-1	8,8585E-1	7,8831E-1	6,5563E-2
	ABAD01	1,8788E-6	7,8338E-6	9,5406E-2	4,9683E-3	1,4149E-2

Tabla 10: Índices estadísticos computados para problemas de optimización de tipo multimodal y separable.

Problema	Algoritmo	Mejor Valor	Mediana	Peor Valor	Promedio	Desviación Típica
f_2	ABA	3,9782E+0	6,4177E+0	1,0774E+1	6,6464E+0	1,4293E+0
	FHS	1,3436E+1	1,7496E+1	2,0948E+1	1,7205E+1	1,7767E+0
	ABAD01	6,9604E-5	2,5696E-4	1,0534E+0	1,3675E-1	3,4778E-1
f_4	ABA	2,1924E-12	9,0805E-12	2,7046E-11	1,0170E-11	5,1955E-12
	FHS	2,6255E-10	8,2081E-10	3,0297E-9	1,0604E-9	6,5462E-10
	ABAD01	2,7527E-27	7,1250E-26	1,9730E-24	1,8374E-25	3,3586E-25
f_9	ABA	1,1934E+1	1,7576E+1	2,6589E+1	1,8706E+1	3,8236E+0
	FHS	5,8138E+1	9,0470E+1	2,2797E+2	9,6150E+1	3,5619E+1
	ABAD01	1,0555E-4	2,9611E-4	6,7224E-4	3,1060E-4	1,0801E-4
f_{11}	ABA	1,1107E+1	1,6921E+1	2,5440E+1	1,7060E+1	2,7588E+0
	FHS	2,3827E+1	3,3888E+1	4,5108E+1	3,4477E+1	4,9195E+0
	ABAD01	5,5664E-5	1,9992E-4	9,9607E-1	1,4006E-1	3,4861E-1

Tabla 11: Índices estadísticos computados para problemas de optimización de tipo unimodal y no separable.

Problema	Algoritmo	Mejor Valor	Mediana	Peor Valor	Promedio	Desviación Típica
f_3	ABA	4,2114E-3	6,9762E-3	1,0508E-2	7,0557E-3	1,4153E-3
	FHS	2,1995E-2	3,1086E-2	4,4470E-2	3,1897E-2	4,9929E-3
	ABAD01	4,9485E-8	1,2692E-7	2,3263E-7	1,3093E-7	4,7940E-8
f_8	ABA	3,5123E+0	9,4746E+0	1,9491E+1	9,6415E+0	4,1468E+0
	FHS	6,4078E+0	1,4293E+1	2,6747E+1	1,5183E+1	4,8575E+0
	ABAD01	3,4896E-4	7,5197E-3	3,6766E-1	4,0018E-2	7,9672E-2
f_{12}	ABA	9,0057E+1	2,5142E+2	2,8115E+3	3,6762E+2	4,1917E+2
	FHS	2,6666E+2	6,9611E+2	2,9185E+3	8,5798E+2	5,7204E+2
	ABAD01	1,6956E+1	7,8971E+1	3,7359E+2	8,4416E+1	7,3163E+1
f_{14}	ABA	6,8161E+0	9,0334E+0	1,0728E+1	9,0139E+0	1,0097E+0
	FHS	2,0372E+1	2,5865E+1	2,6880E+2	4,3481E+1	4,8490E+1
	ABAD01	2,9347E-2	3,7114E-2	5,2874E-2	3,7961E-2	6,3080E-3

al ABAD01 como algoritmo de control [31]. Los valores se normalizaron entre 0 y 1 según el método Mín – Máx [28]. El nivel de significancia se fijó en 0,05. Los resultados se resumen en la Tabla 13.

Los resultados de la prueba permiten afirmar, con un nivel de significancia del 0,05, que

Tabla 12: Índices estadísticos computados para problemas de optimización de tipo unimodal y separable.

Problema	Algoritmo	Mejor Valor	Mediana	Peor Valor	Promedio	Desviación Típica
f_6	ABA	2,2564E+1	3,5806E+1	5,1548E+1	3,6464E+1	6,9574E+0
	FHS	1,1387E+2	1,8001E+2	3,1319E+2	1,8232E+2	3,3425E+1
	ABAD01	2,4023E-4	5,9908E-4	1,3328E-3	6,5428E-4	2,6917E-4
f_7	ABA	3,6620E+0	6,0205E+0	7,5671E+0	5,8166E+0	1,0648E+0
	FHS	2,1041E+1	2,6705E+1	3,6977E+1	2,7109E+1	3,3808E+0
	ABAD01	4,5126E-5	9,6170E-5	2,1709E-4	9,8965E-5	3,3735E-5
f_{13}	ABA	2,6967E-13	2,0366E-10	3,0820E-9	5,1686E-10	7,3611E-10
	FHS	3,6598E-10	3,2386E-9	2,3060E-8	4,9935E-9	4,9324E-9
	ABAD01	7,1520E-15	3,6387E-13	7,0567E-12	1,2018E-12	1,7537E-12

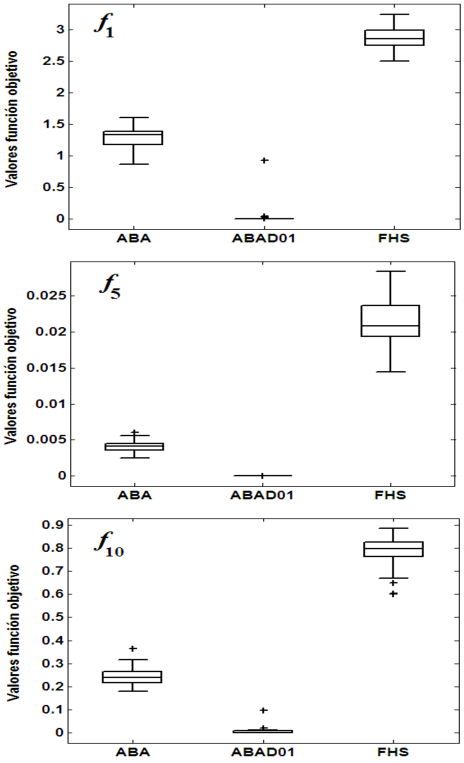

Figura 3: Diagramas de caja y bigotes para los resultados obtenidos sobre las instancias de problemas de optimización multimodal y separable

Tabla 13: Resultados de prueba de Wilcoxon sobre los promedios de los índices estadísticos computados para problemas de optimización.

Algoritmos	W	$p-value$
ABAD01 contra ABA	0	1,2210E-4
ABAD01 contra FHS	0	2,1050E-4

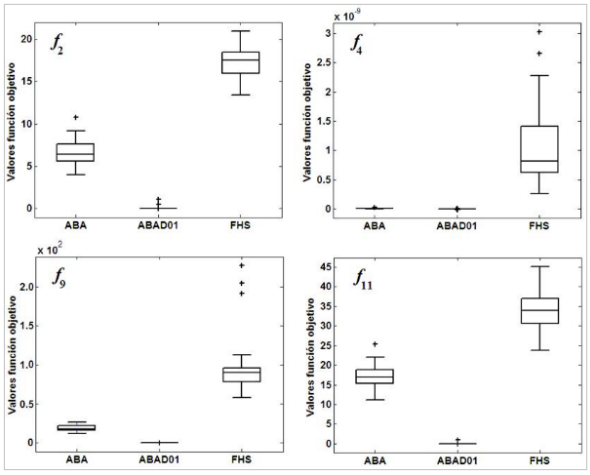

Figura 4: Diagramas de caja y bigotes para los resultados obtenidos sobre las instancias de problemas de optimización multimodal y no separable.

el ABAD01 representa una mejora significativa, respecto a ABA y FHS, como método de solución para los problemas de optimización tratados y bajo las condiciones experimentales antes referidas.

En la Figura 7, se presentan diagramas de caja y bigotes elaborados a partir del valor de la tasa de acierto, proporcionado por ABA, FHS y ABAD02 al término de sus ejecuciones, sobre los datos para entrenamiento (a) y validación (b) asociados al problema de clasificación tratado.

A partir de los datos mostrados en la Figura 7, se computaron los índices estadísticos: mejor valor, peor valor, mediana, promedio y desviación típica. Los resultados obtenidos se presentan en la Tabla 14.

Tabla 14: Índices computados para la tasa de acierto de los algoritmos ABA, FHS y ABAD02 tanto en entrenamiento como en validación.

	Fase					
	Entrenamiento			Validación		
Índice	ABA	FHS	ABAD02	ABA	FHS	ABAD02
Mejor valor	0,9064	0,9447	0,9660	0,8662	0,8662	0,8790
Mediana	0,8255	0,8702	0,9362	0,7643	0,7803	0,8089
Peor valor	0,7447	0,8000	0,8723	0,6561	0,6752	0,7070
Promedio	0,8249	0,8720	0,9345	0,7665	0,7792	0,8079
Desviación típica	0,0330	0,0319	0,0185	0,0401	0,0419	0,0370

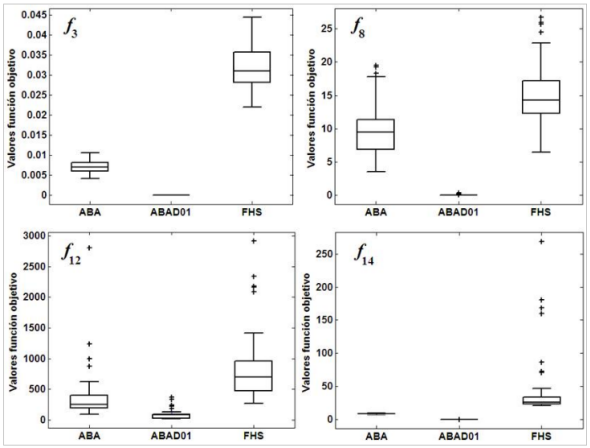

Figura 5: Diagramas de caja y bigotes para los resultados obtenidos sobre las instancias de problemas de optimización unimodal y no separable

Se aplicó una prueba de Wilcoxon en base a los datos asociados a la tasa de acierto en fase de validación (Figura 7b), tomando al ABAD02 como algoritmo de control. El nivel de significancia de la prueba se fijó en 0,05. Los resultados se resumen en la Tabla 15.

Tabla 15: Resultados de prueba de Wilcoxon sobre la tasa de acierto en fase de validación.

Algoritmos	W	$p-value$
ABAD02 contra ABA	0	1,2300E-7
ABAD02 contra FHS	0	1,235E-5

Los resultados de la prueba permiten establecer, con un nivel de significancia del 0,05, que el ABAD02 representa una mejora significativa, en relación a ABA y FHS, como método de solución para el problema de clasificación tratado y bajo las condiciones experimentales planteadas.

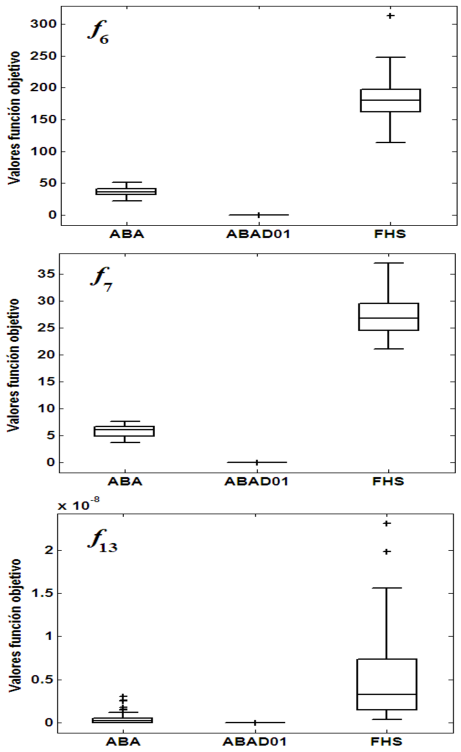

Figura 6: Diagramas de caja y bigotes para los resultados obtenidos sobre las instancias de problemas de optimización unimodal y separable.

Respecto a la comparación de la propuesta y el ARPEM, en la Figura 8 se muestra diagramas de caja y bigotes elaborados a partir del valor de la tasa de acierto, proporcionado por estos algoritmos al término de sus ejecuciones, sobre los datos de entrenamiento (a) y validación (b).

A partir de los datos esquematizados en la

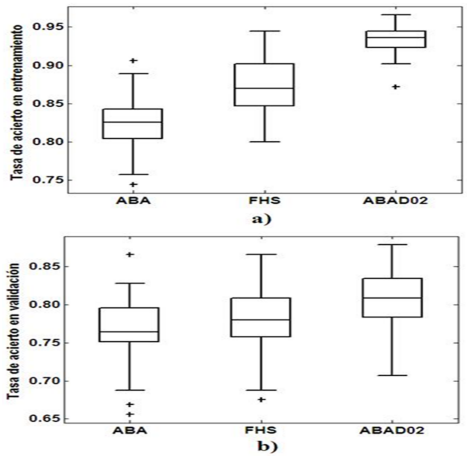

Figura 7: Tasa de acierto para ABA, FHS y ABAD02 en: a) Entrenamiento. b) Validación.

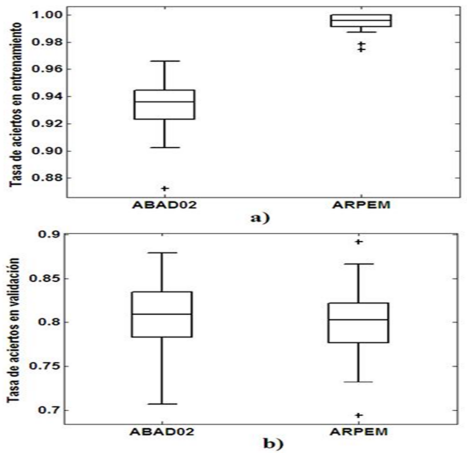

Figura 8: Tasa de acierto para ABAD02 y ARPEM en: a) Entrenamiento. b) Validación.

Figura 8, se computaron los índices: mejor valor, peor valor, mediana, promedio y desviación típica. Los resultados obtenidos se presentan en la Tabla 16 mostrada a continuación.

Se aplicó una prueba de Wilcoxon sobre los datos asociados a la tasa de acierto en fase de validación. El nivel de significancia se fijó en 0,05. Los resultados se resumen en la Tabla 17.

Luego, aunque durante la fase de entrenamiento el ARPEM es claramente superior al ABAD02, los resultados de la prueba señalan, con una significancia de 0,05, que desde un punto de vista estadístico no existe diferencia significativa en cuanto a la capacidad de generalización exhibida por la RNM tras ser entrenada mediante ambos algoritmos. Por supuesto, estas afirmaciones

Tabla 16: Índices computados para la tasa de acierto de los algoritmos ABAD02 y ARPEM en fase de entrenamiento y validación.

	Fase			
	Entrenamiento		Validación	
Índice	ABAD02	ARPEM	ABAD02	ARPEM
Mejor valor	0,9660	1,0000	0,8790	0,8917
Mediana	0,9362	0,9957	0,8089	0,8025
Peor valor	0,8723	0,9745	0,7070	0,6943
Promedio	0,9345	0,9948	0,8079	0,8023
Desviación Típica	0,0185	0,0056	0,0370	0,0391

Tabla 17: Resultados de la prueba de Wilcoxon sobre la tasa de acierto en fase de validación para ABAD02 y ARPEM.

Algoritmos	W	$p - value$
ABAD02 contra ARPEM	597,5	0,2336

son válidas únicamente en el contexto de las condiciones experimentales establecidas.

6. Conclusiones

A partir de los resultados obtenidos en el presente estudio, se evidenció que la implementación de un SIDM para la configuración dinámica de todo el ensamble de parámetros numéricos asociados al ABA se tradujo en una significativa mejora de este algoritmo, en relación a su versión de parámetros fijos, como método numérico aplicado en la resolución de problemas de optimización. Más aún, al ser comparada con FHS, la cual está basada en principios idénticos, la propuesta aquí planteada también representó una mejora sustancial. En lo que respecta a la experiencia sobre el entrenamiento supervisado de una RNM, no se halló diferencia estadísticamente significativa en cuanto a la capacidad de generalización obtenida mediante la aplicación del ARPEM y el esquema propuesto. Finalmente, se insiste en señalar que las conclusiones derivadas son válidas únicamente

en el contexto experimental establecido, siendo necesarias ulteriores investigaciones en el futuro.

7. Referencias

[1] A. Gogna and A. Tayal. Metaheuristics: review and application. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4):503–526, 2013.

[2] T. Stützle, M. López-Ibáñez. Automated design of metaheuristic algorithms (technical report institut de recherches interdisciplinaires et de developpements en intelligence artificielle (iridia) number tr/iridia/2018-008). Technical report, Université Libre de Bruxelles, Bruxelles, Belgium: Technical Report Series, 2018.

[3] G. Karafotias, M. Hoogendoorn, A. Eiben. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, 19(2):167–187, 2015.

[4] O. Castillo, F. Valdez, and P. Melin. A survey on nature – inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation. *Expert Systems with Applications*, 41(14):6459 – 6466, 2014.

[5] O. Castillo C. Peraza, and F. Valdez. An Improved Harmony Search Algorithm Using Fuzzy Logic for the Optimization of Mathematical Functions. In J. Kacprzyk P. Melin, O. Castillo, editor, *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature – Inspired Optimization*, volume 601, page 605 – 615. Springer – Verlag, Berlín, Germany, 2015.

[6] K. Lee and Z. Geem. A new meta – heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36):3902 – 3933, 2005.

[7] E. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man – Machine Studies*, 7(1):1 – 13, 1975.

[8] K. Ameli, A. Alfi, and M. Aghaebrahimi. A fuzzy discrete harmony search algorithm applied to annual cost reduction in radial distribution systems. *Journal of Engineering Optimization*, 48(9):1529 – 1549, 2015.

[9] K. Pandiarajan and C. Babulal. Fuzzy harmony search algorithm based optimal power flow for power system security enhancement. *International Journal of Electrical Power & Energy Systems*, 78(1):72 – 79, 2016.

[10] C. Peraza, F. Valdez, and O. Castillo. An Adaptive Fuzzy Control Based on Harmony Search and Its Application to Optimization. In P. Melin, O. Castillo, J. Kacprzyk, editor, *Nature – Inspired Design of Hybrid Intelligent Systems*, volume 667, page 269 – 283. Springer – Verlag, Berlín, Germany, 2017.

[11] C. Peraza, F. Valdez, J. Castro, and O. Castillo. Fuzzy Dynamic Parameter Adaptation in the Harmony Search Algorithm for the Optimization of the Ball and Beam Controller. *Advances in Operations Research*, 2018(7):1–16, 2018.

[12] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back – propagating errors. *Nature*, 323(9):533 – 536, 1986.

[13] M. Ahangaran and P. Ramezani. Harmony Search Algorithm: Strengths and Weaknesses. *Journal of Computer Engineering & Information Technology*, 2(1):1–7, 2013.

[14] R. Diao and Q. Shen. Deterministic Parameter Control in Harmony Search. In S. Lucas, editor, *2010 UK Workshop on Computational Intelligence (UKCI)*, pages 1–7. University of Essex, Colchester, United Kingdom, 2010.

[15] M. Omran and M. Mahdavi. Global – best harmony search. *Applied mathematics and computation*, 198(2):643 – 656, 2008.

[16] H. Adeli and N. Siddique. Harmony Search Algorithm and its Variants. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(8):1 – 22, 2015.

[17] C. Wang and Y. Huang. Self – adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37(4):2826 – 2837, 2010.

[18] P. Yadav, R. Kumar, S. Panda, and S. Chang. An intelligent tuned harmony search algorithm for optimisation. *Information Sciences*, 196:47 – 72, 2012.

[19] J. Chen, Q. Pan, and J. Li. Harmony search algorithm with dynamic control parameters. *Applied Mathematics and Computation*, 219(2):592 – 604, 2012.

[20] J. Kalivarapu, S. Jain, and S. Bag. An improved harmony search algorithm with dynamically varying bandwidth. *Engineering Optimization*, 48(7):1091 – 1108, 2016.

[21] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567 – 1579, 2007.

[22] F. Fernandez, M. Tomassini, and L. Vanneschi. Saving computational effort in genetic programming by means of plagues. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'03)*, page 2042 – 2049, Canberra, Australia, 2003.

[23] F. de Vega, E. Cantú–Paz, J. López, and T. Manzano. Saving resources with plagues in genetic algorithms. In X. Yao et al., editor, *Parallel Problem Solving from Nature — PPSN VIII, Vol. 3242: Lecture Notes in Computer Science*, page 272 – 281. Springer – Verlag, Berlín, Germany, 2004.

[24] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh. Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer methods in applied mechanics and engineering*, 197(40):3080 – 3091, 2008.

[25] S. Kulluk, L. Ozbakir, and A. Baykasoglu Training neural networks with harmony search algorithms for classification problems. *Engineering Applications of Artificial Intelligence*, 25(1):11–19, 2012.

[26] M. Jamil and X. Yang A literature survey of benchmark functions for global optimization problems. *Int. Journal of Mathematical Modelling and Numerical Optimization*, 4(2):150 – 194, 2013.

[27] A. Karegowda, A. Manjunath, and M. Jayaram. Application of Genetic Algorithm Optimized Neural Network Connection Weights for Medical Diagnosis of Pima Indians Diabetes. *International Journal on Soft Computing (IJSC)*, 2(2):15–23, 2011.

[28] T. Jayalakshmi and A. Santhakumaran. Statistical Normalization and Back Propagation for Classification. *International Journal of Computer Theory and Engineering*, 3(1):1793 – 8201, 2011.

[29] H. Faris, I. Aljarah, and S. Mirjalili. Improved monarch butterfly optimization for unconstrained global search and neural network training. *Applied Intelligence*, 48(2):445 – 464, 2018.

[30] X. Li, K. Tang, M. Omidvar, Z. Yang, and K. Qin. Benchmark Functions for the CEC'2013 Special Session and Competition on Large – Scale Global Optimization. *Gene*, 7(33):8–30, 2013.

[31] J. Derrac, S. García, D. Molina, and F. Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18, 2011.