ISSN 1870-4069

# Uso de metaheurísticas para entrenamiento de redes neuronales artificiales

Yoqsan Angeles García[1], Hiram Calvo[1],
Álvaro Anzueto Ríos[2]

[1] Instituto Politécnico Nacional,
Centro de Investigación en Computación,
México

[2] Instituto Politécnico Nacional,
Unidad Profesional Interdisciplinaria en
Ingeniería y Tecnologías Avanzadas,
México

{yangelesg2020,hcalvo}@cic.ipn.mx, aanzuetor@ipn.mx

**Resumen.** Los algoritmos basados en métodos metaheurísticos son aplicados a problemas de optimización y han demostrado un gran sentido de convergencia en sus soluciones, es por eso que en este trabajo se ha considerado su aplicación para determinar los valores de los pesos sinápticos en una arquitectura neuronal que ha sido configurada como aproximador de funciones. Se han elegido tres funciones típicas (benchmark) como pruebas de rendimiento, además, de considerar el error cuadrático medio (ECM), como función objetivo o de ajuste para los algoritmos metaheurísticos. El análisis de los resultados indica que los algoritmos de abejas y evolución diferencial funcionan bien para problemas de alta dimensionalidad y recocido simulado presenta la mejor relación entre tiempo y valor óptimo. En general, este estudio demuestra la eficiencia de las metaheurísticas para resolver problemas de entrenamiento de redes neuronales artificiales.

**Palabras clave:** Colonia artificial de abejas, evolución diferencial, recocido simulado, optimización por enjambre de partículas, redes neuraonales artificiales.

# Use of Metaheuristics for Artificial Neural Network Training

**Abstract.** The algorithms based on metaheuristic methods are applied to optimization problems and have demonstrated a high degree of convergence in their solutions. Therefore, this study considers their application to determine the values of synaptic weights in a neural architecture configured as a function approximator. Three typical functions (benchmarks) were chosen as performance tests, in addition to using mean squared error (MSE) as the objective or fitting function for the metaheuristic algorithms. The analysis of the results indicates that artificial bee colony and differential evolution algorithms perform well for

high-dimensional problems, while simulated annealing has the best trade-off between time and optimal value. Overall, our findings demonstrate the efficacy of metaheuristic methods in solving training problems in artificial neural networks.

**Keywords:** Artificial bee colony, differential evolution, simulated annealing, particle swarm optimization, artificial neural networks.

## 1. Introducción

Las redes neuronales tienen diversas aplicaciones en diferentes áreas, tales como clasificación, control, procesamiento de imágenes, procesamiento de lenguaje natural, etc. La literatura clásica sobre redes neuronales muestra el método de descenso de gradiente como la forma de optimización de los pesos de las redes neuronales [1, 2]. El descenso de gradiente es un algoritmo basado en derivadas para encontrar la dirección de máximo decremento de una función [3]. Sin embargo, existen métodos de optimización que involucran procesos estocásticos.

Estos métodos son las metaheurísticas. Estos métodos sirven como una alternativa al método de descenso de gradiente y sus variantes [4]; pues, a pesar de que ha demostrado un desempeño satisfactorio en términos de los resultados obtenidos, existen trabajos en los que se muestra que existen circunstancias en las que puede no converger a un valor óptimo [5, 6]. Aunque existen diferentes algoritmos metaheurísticos [7], para este trabajo se han considerado desarrollar 5 metodologías de tipo metaehurísticas, las cuales son:

- Recocido Simulado.
- Optimización por Enjambre de Partículas.
- Algoritmos Genéticos.
- Colonia Artificial de Abejas.
- Evolución Diferencial.

La elección de metaheurísticas adecuadas para el entrenamiento de redes neuronales es fundamental para lograr una optimización efectiva. En este sentido, los algoritmos genéticos, la colonia artificial de abejas, la optimización por enjambre de partículas, el recocido simulado y la evolución diferencial son cinco metaheurísticas que pueden justificarse por su eficacia en problemas de optimización no lineales y no convexas.

Estas técnicas también tienen la capacidad de manejar múltiples objetivos y soluciones no factibles, lo que las hace ideales para el entrenamiento de redes neuronales complejas.

El objetivo de esta investigación es comparar las características y el desempeño de las metaheurísticas seleccionadas en el entrenamiento de redes neuronales, con el fin de evaluar su eficacia y determinar cuál de ellas es más adecuada para este fin.

Asimismo, la exploración de otras metaheurísticas puede ser de gran interés para futuras investigaciones en el campo de la optimización de redes neuronales. Otras metaheurísticas que también podrían ser consideradas son la búsqueda tabú, la

**Tabla 1.** Algunos trabajos relacionados.

| Título | Parámetros de optimización | Metaheurística | Año |
|---|---|---|---|
| Particle Swarm Optimization of Neural Network Architectures and Weigths | Optimización de pesos y arquitecturas en Perceptrón Multicapa en problemas en el área médica | PSO | 2007 |
| Optimizing connection weights in neural netwokrs using the whale optimization algorithm | Optimización de pesos en Perceptrón Multicapa para 20 problemas de clasificación | Algoritmo de la Ballena (algoritmo poblacional) | 2016 |
| Artificial Neural Network training using metaheuristics for medical data classification: An experimental study | Optimización de pesos en Perceptrón Multicapa para clasificación de imágenes médicas | Comparación de 14 metaheurísticas diferentes | 2022 |

búsqueda armónica, colonia de hormigas, entre otras. A continuación se da un breve resumen de cada una.

### 1.1. Recocido simulado (SA)

El recocido simulado (SA por sus siglas en inglés de Simulated Annealing) es un algoritmo que simula el comportamiento del enfriamiento lento de sistemas físicos simples, por ejemplo en el proceso de producción de acero. La idea de la que parte es que, al calentar los metales, las moléculas que lo componen están en movimiento.

Sin embargo, tras un lento proceso de enfriamiento, tales partículas alcanzan un estado de mínima energía, a pesar del movimiento aleatorio de las moléculas. Este método de recocido simulado permite explotar el conocimiento físico de tales procesos para tratar de encontrar el mínimo absoluto de cualquier función matemática [7].

### 1.2. Optimización por enjambre de partículas (PSO)

El algoritmo de Optimización de Enjambre de Partículas (PSO por sus siglas en inglés de Particle Swarm Optimization), fue desarrollado por Kennedy y Eberhart [8] y se basa en el comportamiento social que se ha observado en distintos grupos de individuos como pueden ser parvadas de aves, enjambres de insectos o bancos de peces.

El comportamiento grupal es el resultado de la interacción de dos factores mutuamente dependientes: las conductas individuales exhibidas por cada miembro y la conducta emergente que se manifiesta en el comportamiento colectivo del grupo. Cada individuo transmite información al resto del grupo, lo cual resulta en un proceso que permite a los miembros encontrar un valor adecuado.

Básicamente, PSO consiste en un algoritmo iterativo basado en una población de individuos, en la que cada partícula, o miembro, explora una parte del espacio de búsqueda en busca de soluciones óptimas y comparte la información con el resto del enjambre.

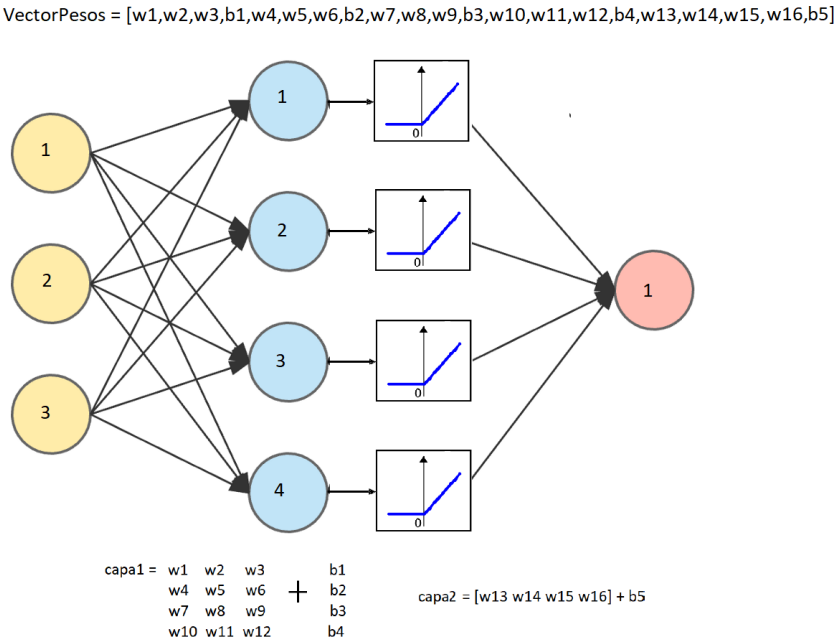


**Fig. 1.** Representación gráfica del acomodo de un vector de pesos en las capas de una red neuronal.

### 1.3. Algoritmos genéticos (GA)

Los Algoritmos Genéticos (GA por sus siglas en inglés de Genetic Agorithm) parten de la idea de la evolución darwiniana. Se utilizan para problemas de optimización con una población de individuos, los cuales se componen de un cromosoma de la forma $x = [x_1, x_2, ..., x_n]$.

De forma clásica, el cromosoma es un número binario, donde cada gen o bit $x_i$, únicamente puede tener valores de 1 y 0; sin embargo, también puede adaptarse para números reales en cada gen. El algoritmo genético considera una población inicial de M cromosomas de largo b, generados aleatoriamente.

Para evaluar el desempeño de estos individuos se requiere decodificar cada componente en caso de trabajar con números binarios o evaluar la función en caso de números reales.

Durante el proceso evolutivo, el algoritmo genético selecciona a los miembros más aptos de la población para realizar el proceso de cruza o reproducción, en el cual se tienen nuevos individuos, idealmente mejores que los padres y que formarán la nueva población.

También existe un proceso de mutación, en el cual se muta un gen del cromosoma de acuerdo a una probabilidad dada. La mutación sirve para evitar la pérdida de diversidad, producto de genes que han convergido a un cierto valor para toda la población, y que por tanto, no pueden ser recuperados por el operador de recombinación [9].

**Tabla 2.** Funciones, dimensiones y tamaño de red.

| Función | Dimensión | Dominio | Datos de entrenamiento totales | Parámetros de Red Neuronal |
|---|---|---|---|---|
| Sphere | 2 | -5,5 | 1x250 | 31 |
| Rastrigin | 2 | -5,5 | 1x250 | 31 |
| Griewank | 2 | -50,50 | 1x250 | 31 |
| Sphere | 11 | -5,5 | 10x2500 | 1201 |
| Rastrigin | 11 | -5,5 | 10x2500 | 1201 |
| Griewank | 11 | -50,50 | 10x2500 | 1201 |

**Tabla 3.** Datos estadísticos de ejecución para la función Spehre en 2D.

| Metaheurística | Mejor | Peor | Promedio | Mediana | STD | Promedio de épocas hasta convergencia |
|---|---|---|---|---|---|---|
| SA | 0.2615 | 2.2798 | 1.0512 | 0.9178 | 0.7456 | 1834 |
| GA | 0.3195 | 1.5513 | 0.9079 | 0.9240 | 0.4875 | 950 |
| ABC | 0.1873 | 0.5080 | 0.3271 | 0.33 | 0.1365 | 1092 |
| PSO | 0.5846 | 8.5009 | 3.1326 | 1.2450 | 3.4476 | 685 |
| DE | 0.1351 | 0.5797 | 0.2608 | 0.2065 | 0.1817 | 1133 |

### 1.4. Colonia artificial de abejas (ABC)

Colonia de Abejas Artificiales (ABC por sus siglas en inglés de Artificial Bee Colony) es una técnica propuesta por Karaboga y Basturk [10] que se basa en el comportamiento de las abejas melíferas para encontrar las mejores fuentes de alimento como individuo y también para que el resto de la colmena aproveche ese alimento al comunicar esta información a otras abejas.

ABC es un algoritmo diseñado para resolver problemas de optimización combinatoria, que se basa en poblaciones donde las soluciones, llamadas fuentes de alimento, son exploradas por las abejas. El objetivo de estas abejas es descubrir nuevas fuentes de alimento que tengan cada vez mayores cantidades de néctar.

Para el caso de optimización, entre más cercano al óptimo sea el valor de una función, significa una mayor cantidad de néctar. Sin embargo, si después de explorar una fuente de alimento no encuentran una mejor en la vecindad, otras abejas exploradoras se moverán aleatoriamente por el espacio de búsqueda para encontrar una nueva fuente de alimento. De esta forma se obtienen soluciones óptimas locales e, idealmente, el óptimo global.

En un sistema ABC, las abejas artificiales se mueven en un espacio de búsqueda multidimensional eligiendo fuentes de néctar dependiendo de su experiencia pasada y de sus compañeras de colmena. ABC trata de balancear los métodos de exploración y explotación.

### 1.5. Evolución diferencial (DE)

La evolución diferencial (DE por sus siglas en inglés de Differential Evolution) es una rama de la computación evolutiva desarrollada por Rainer Storn y Kenneth Price [11] para optimización en espacios continuos.

**Tabla 4.** Datos estadísticos de ejecución para la función Rastrigin en 2D.

| Metaheurística | Mejor | Peor | Promedio | Mediana | STD | Promedio de épocas hasta convergencia |
|---|---|---|---|---|---|---|
| SA | 49.75 | 55.41 | 52.95 | 54.62 | 2.9123 | 2124 |
| GA | 46.02 | 49.51 | 47.52 | 47.46 | 1.3475 | 842 |
| ABC | 44.28 | 46.24 | 45.53 | 45.56 | 0.8037 | 1514 |
| PSO | 45.90 | 75.02 | 54.67 | 51.03 | 11.5921 | 696 |
| DE | 47.44 | 49.32 | 48.38 | 48.62 | 0.7484 | 1221 |

**Tabla 5.** Datos estadísticos de ejecución para la función Griewank en 2D.

| Metaheurística | Mejor | Peor | Promedio | Mediana | STD | Promedio de épocas hasta convergencia |
|---|---|---|---|---|---|---|
| SA | 48.71 | 50.89 | 49.93 | 50.25 | 1.04 | 1343 |
| GA | 45.37 | 47.65 | 46.81 | 47.16 | 0.9372 | 1072 |
| ABC | 47.04 | 47.65 | 47.12 | 47.12 | 0.25 | 1013 |
| PSO | 47.02 | 50.81 | 48.61 | 47.94 | 1.55 | 739 |
| DE | 47.22 | 49.78 | 48.24 | 47.89 | 1.05 | 1970 |

En este método, las variables se representan mediante números reales. Parte de una población inicial generada aleatoriamente y, a diferencia de los algoritmos genéticos, se seleccionan tres individuos como padres. Uno de estos individuos es el padre principal y éste será el que se modifica con la información de los otros dos padres.

Si el valor resultante es mejor que el del padre principal, entonces se reemplaza, de lo contrario, se descarta y se conserva al padre principal. En el algoritmo de ED, para la generación de nuevos vectores, se suma la diferencia de pesos entre dos vectores miembros de la población a un tercer vector miembro o padre principal.

## 2. Estado del arte

El trabajo sobre metaheurísticas para el uso en redes neuronales es amplio, existen trabajos en los que se recopilan diversos trabajos sobre el tema. En el trabajo Advances of metaheuristic algorithms in training neural networks for industrial applications [12] se realiza un resumen los algoritmos metaheurísticos de los últimos 20 años y se indica la eficiencia que han tenido al utilizarse para entrenamiento de redes neuronales artificiales en aplicaciones en la industria.

Se llega a la conclusión de que las metaheurísticas tienen una buena relación entre exploración y explotación, lo que les da cierta ventaja en redes neuronales de propagación hacia adelante. Sin embargo, tal como dice el teorema de No Free Lunch, ningún algoritmo de estos ha servido mejor para resolver todos los problemas, por lo que aún queda la tarea de elegir el mejor método y adaptarlo para la aplicación deseada.

A pesar de la diversidad de trabajos antes mencionada, el enfoque de las metaheuristicas aplicadas en redes neuronales no se utliza para el cálculo de los pesos, se utiliza para optimización de hiperparámetros.

**Tabla 6.** Datos estadísticos de ejecución para la función Sphere en 11D.

| Metaheurística | Mejor | Peor | Promedio | Mediana | STD | Promedio de épocas hasta convergencia |
|---|---|---|---|---|---|---|
| SA | 24.61 | 45.76 | 26.87 | 35.84 | 7.53 | 37020 |
| GA | 144.42 | 202.28 | 164.42 | 171.34 | 30.82 | 10488 |
| ABC | 34.65 | 45.86 | 42.59 | 41.84 | 3.97 | 20124 |
| PSO | 911.38 | 1553 | 951.94 | 1322 | 244 | 2369 |
| DE | 16.38 | 189.2 | 26.45 | 27.47 | 78.14 | 5621 |

**Tabla 7.** Datos estadísticos de ejecución para la función Rastrigin en 11D.

| Metaheurística | Mejor | Peor | Promedio | Mediana | STD | Promedio de épocas hasta convergencia |
|---|---|---|---|---|---|---|
| SA | 2287 | 2416 | 2383.6 | 2337 | 50.36 | 2880 |
| GA | 2359 | 2590 | 2499.0 | 2549 | 38.87 | 9039 |
| ABC | 2458 | 2708 | 2511.8 | 2514 | 95.69 | 9733 |
| PSO | 3773 | 4365 | 3944.4 | 4093 | 222 | 1400 |
| DE | 2199 | 2364 | 2287.1 | 2349 | 31.98 | 11072 |

En la Tabla 1 se presenta una recopilación de trabajos previos que han abordado el problema, tanto, de la obtención de los pesos sinápticos de arquitecturas neuronales con perceptrones multicapa, como los hiperparámetros en redes neuronales convolucionales para la tarea de clasificación de imágenes, demostrando el hecho que este es un tópico de interés actual para los investigadores [13, 14, 15].

## 3. Desarrollo de la solución

En esta sección se describen los pasos que se llevaron a cabo para dar solución al problema planteado.

### 3.1. Descripción de la solución al problema planteado

Este trabajo se realizó en MATLABr2022B bajo la licencia de estudiante. Sin embargo, no se hizo uso de la DeepLearningToolbox incluida. La forma en la que se minimiza el error en todos los algoritmos aquí presentados es la modificación de un número n de pesos de la red neuronal por cada época, cambiando entre cada algoritmo la forma en la que se modifican estos pesos.

La forma en la que se codifican las entradas para realizar el proceso de optimización es el siguiente: Se inicializa un vector con n datos, siendo n el número de variables que puede modificarse en la red neuronal, para este caso, es el conjunto de pesos W y bias b de la red neuronal.

Teniendo el vector de pesos, es posible realizar modificaciones dentro del vector para posteriormente ejecutar la red y calcular el error cuadrático medio entre la salida deseada y la salida obtenida de la red neuronal. La Figura 1 muestra un ejemplo de red neuronal de tres capas, en la parte superior, el vector que se utiliza para ejecutar la red neuronal.

**Tabla 8.** Datos estadísticos de ejecución para la función Griewank en 11D.

| Metaheurística | Mejor | Peor | Promedio | Mediana | STD | Promedio de épocas hasta convergencia |
|---|---|---|---|---|---|---|
| SA | 130.19 | 174.19 | 170.08 | 150.26 | 19.08 | 6440 |
| GA | 51.02 | 113.02 | 66.99 | 97.86 | 36.18 | 6446 |
| ABC | 116.96 | 230.75 | 121.05 | 127.25 | 6.69 | 1226 |
| PSO | 45.95 | 170.1 | 49.27 | 67.69 | 50.73 | 4071 |
| DE | 206.24 | 230.75 | 216.96 | 227.25 | 6.69 | 1226 |

**Tabla 9.** Datos estadísticos de ejecución para la función Sphere, Rastrigin y Griewank en 11D con datos desconocidos.

| Función | Metaheurística | Error de entrenamiento | Error de prueba | Tiempo por 100 épocas en segundos |
|---|---|---|---|---|
| Sphere | SA | 26.87 | 15.82 | 0.1297 |
| Sphere | GA | 162.42 | 107.86 | 5.3941 |
| Sphere | ABC | 42.59 | 23.33 | 12.0833 |
| Sphere | PSO | 951.94 | 848.56 | 5.42 |
| Sphere | DE | 26.45 | 13.80 | 8.6071 |
| Rastrigin | SA | 2383.6 | 3008 | 0.1297 |
| Rastrigin | GA | 2499.0 | 2846 | 5.3941 |
| Rastrigin | ABC | 2511.8 | 2851 | 12.0833 |
| Rastrigin | PSO | 3944.4 | 4087 | 5.42 |
| Rastrigin | DE | 2287.1 | 2721 | 8.6071 |
| Griewank | SA | 170.08 | 94.06 | 0.1297 |
| Griewank | GA | 66.99 | 41.97 | 5.3941 |
| Griewank | ABC | 121.05 | 90.32 | 12.0833 |
| Griewank | PSO | 49.27 | 32.27 | 5.42 |
| Griewank | DE | 216.96 | 162.39 | 8.6071 |

En la parte inferior se muestra cómo se acomodan los elementos del vector en forma de matrices para realizar las operaciones necesarias en la ejecución de la red. A continuación se describen los parámetros para cada uno de los algoritmos metaheurísticos:

– **ABC**: Este algoritmo requiere conocer el tamaño de la población inicial, llamado Np con valor de 50, también requiere el límite N para mejorar una solución.

– **GA**: Este algoritmo requiere conocer el tamaño de la población inicial, llamado Np con valor de 50, la probabilidad de cruza, llamada Mc con valor de 1, el tipo de selección de padres que es por torneo con una población de 2 miembros por torneo, $M = 0.01$, el tipo de cruza y el tipo de mutación.

Para este trabajo el tipo de cruza es el método conocido como cruza de dos puntos y la mutación se realiza cambiando el gen x por un número aleatorio dentro del espacio de búsqueda. La selección de la nueva población se hace de forma elitista eligiendo los Np miembros con mejor aptitud, incluyendo padres e hijos.

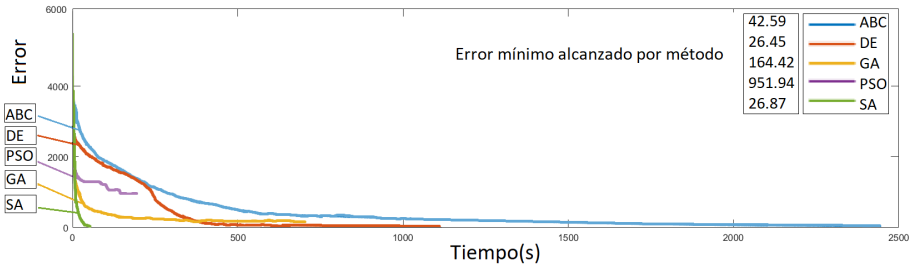


**Fig. 2.** Comparación de error en el entrenamiento para la función Sphere en 11 dimensiones.

– **SA**: Este algoritmo requiere conocer el tipo de enfriamiento, el tamaño de vecindad para elección de nuevas soluciones. El tipo de enfriamiento usado en este trabajo es un enfriamiento lineal, el cual requiere conocer la temperatura inicial, de 10 y DeltaT de 0.5.

– **DE**: Este algoritmo requiere conocer el tamaño de la población, llamado Np con valor de 50, un factor de escala, llamado alpha con un valor de 0.3 y la probabilidad de cruza, llamada Pcr con un valor de 0.4.

– **PSO**: Este algoritmo requiere conocer el tamaño de la población, llamado Np con valor de 50, un coeficiente de aceleración personal con un valor de 1, un coeficiente de aceleración social con un valor de 2 y un factor de escala de desaceleración, llamado w con un valor de 0.75.

En cada algoritmo usado en este trabajo, el número de pesos a modificar por cada época varía de acuerdo a la diferencia del error nuevo y el anterior. En un principio se modifica 2 % de los pesos totales. Al alcanzar un número de épocas con una diferencia de error menor a un $\delta$ dado, el número de pesos a modificar se reduce a 1 %, posteriormente a 0.1 % y por último a un peso por época. Para los algoritmos se tienen dos criterios de paro: uno es el número máximo de épocas, que es de 45,000 y un máximo de iteraciones en las que la diferencia entre el error actual y el error previo es menor a un $\epsilon$ dado, en este caso $\epsilon = 0.1$.

### 3.2. Evaluación

Para la evaluación de las redes neuronales, se eligió que éstas resuelvan el problema de aproximación de funciones. Se toman tres funciones matemáticas a aproximar: Sphere [1], Rastrigin [2] y Griewank [3].

Estas funciones se toman de [16], donde se hace la comparación similar a la propuesta aquí planteada; pero comparando tres tipos de redes neuronales en lugar de métodos de entrenamiento. Se prueban las funciones en dos y once dimensiones:

$$f(x) = \sum_{i=1}^{D} x_i^2, \tag{1}$$

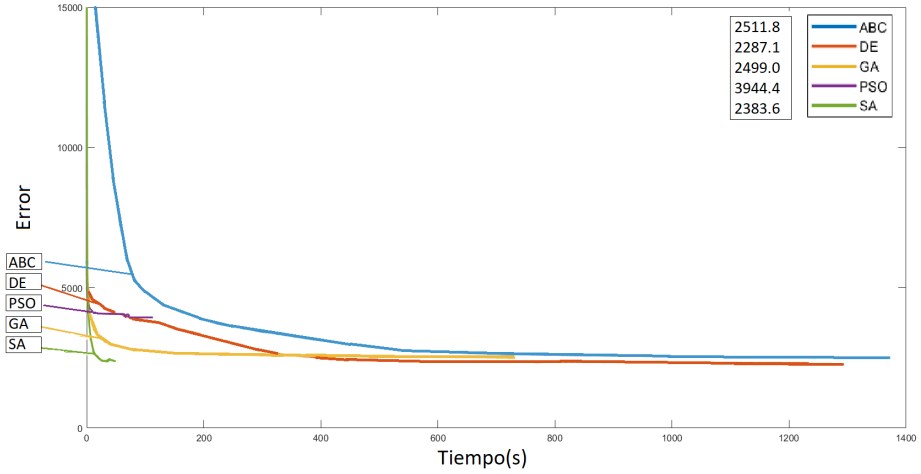


**Fig. 3.** Comparación de error en el entrenamiento para la función Rastrigin en 11 dimensiones.

$$f(x) = \sum_{i=1}^{D} x_i^2 - 10cos(2\pi x_i) + 10, \tag{2}$$

$$f(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1. \tag{3}$$

La Tabla 2 muestra características de las funciones. El número de datos en la capa de entrada es el número de dimensión menos uno. Por ejemplo, para la función de dos dimensiones, sólo hay una neurona en la capa de entrada. Se utilizan dos arquitecturas neuronales dependiendo del número de dimensión de la función.

Para el caso de dos dimensiones, la red neuronal consta de tres capas con las siguientes características: la capa uno consta de una sola neurona y una función de activación sigmoide, la capa dos consta de 10 neuronas y una función de activación sigmoide, la capa tres consta de una sola neurona y una función de activación lineal.

Para el caso de once dimensiones: la capa uno consta de 10 neuronas y una función de activación sigmoide, la capa dos consta de 100 neuronas y una función de activación sigmoide, la capa tres consta de una sola neurona y una función de activación lineal.

Para evaluar el desempeño de las redes neuronales, se calcula el error cuadrático medio (MSE por sus siglas en inglés), el cual compara la diferencia entre la salida deseada o target y la salida obtenida por la red. La fórmula para el cálculo del error cuadrático medio se muestra en la ecuación 4:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2. \tag{4}$$

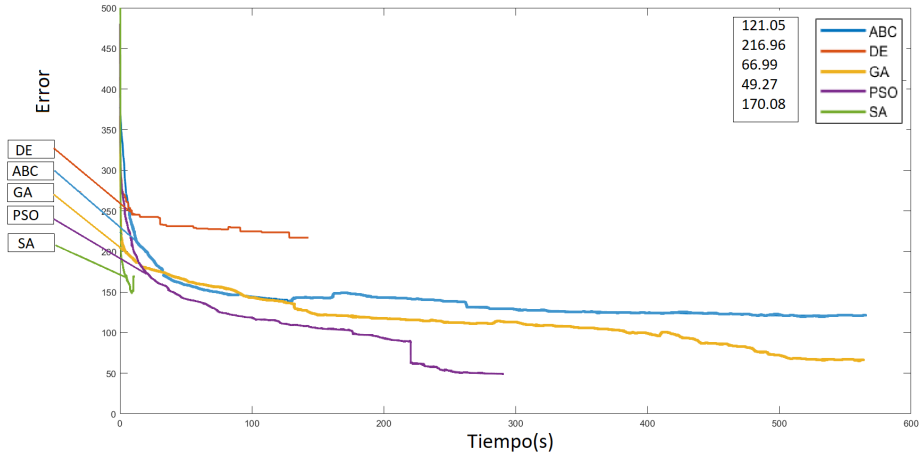


**Fig. 4.** Comparación de error en el entrenamiento para la función Griewank en 11 dimensiones.

## 4. Experimentos y resultados

En esta sección se presenta los resultados obtenidos al evaluar cada uno de los algoritmos metaheurísticos con las tres funciones de prueba. Cada función fue ejecutada en 20 ocasiones y se registraron de ellas los valores de error cuadrático medio, considerando el valor mínimo, máximo y promedio de las 20 ejecuciones. Se han recopilado 60 registros por cada sistema metaheurístico al evaluar las tres funciones.

Los resultados para las funciones Sphere, Rastrigin y Griewank en dos dimensiones se pueden ver en las Tablas 2, 3 y 4 respectivamente. Para el caso de once dimensiones se pueden ver en las Tablas 5, 6 y 7. Es importante mencionar que los resultados mostrados son únicamente del entrenamiento, los resultados con datos de prueba se muestran más adelante en las Tablas 8, 9 y 10.

Como se mencionó anteriormente, las tablas [3, 8] muestran los datos tomados del error cuadrático medio para el entrenamiento. Para la fase de prueba, se generaron nuevos datos aleatoriamente. Para este caso, se prueba únicamente con las funciones en 11 dimensiones pues fueron las más complejas para aproximar.

El vector de entrada de prueba es de 10x2500 datos; y se generaron aleatoriamente en el mismo rango del de los datos de entrenamiento. Para esta fase de prueba se tomaron las mejores configuraciones de pesos de las pruebas de entrenamiento con cada algoritmo.

Con el propósito de facilitar una comparación visual, las Figuras 2, 3 y 4 muestran una representación del comportamiento del error promedio de entrenamiento para cada método. Es preciso destacar que la variación en el tamaño de las gráficas mostradas es consecuencia de la divergencia en el número de épocas necesarias para alcanzar la convergencia en cada método. Sin embargo, el tiempo de ejecución de los métodos no es dependiente de este factor, pues cada método tarda una cantidad diferente de tiempo por época.

## 5. Conclusiones y trabajo futuro

Este estudio demostró que las metaheurísticas pueden ser una alternativa al descenso de gradiente para entrenar redes neuronales, aunque cada metaheurística tiene ventajas y desventajas. Se recopilaron y almacenaron los resultados para crear una base de conocimiento que ayude a seleccionar el tipo de red neuronal y el algoritmo metaheurístico más apropiado para una tarea específica.

En la Tabla 9 se muestra que en varios casos el error en los datos de prueba fue menor que en los de entrenamiento, lo que indica una buena generalización de las redes neuronales. Además, la gran diferencia en el error cuadrático medio en la función Rastrigin en comparación con las funciones Sphere y Griewank se debe a las diferencias en el dominio de las funciones, lo que afecta el desempeño de los algoritmos de optimización y, a su vez, el error de la red neuronal. Es crucial considerar este factor al evaluar el desempeño de las metaheurísticas en diferentes problemas de optimización.

## Referencias


1. Hagan, M. T., Demuth, H. B., Beale, M.: Neural network design. PWS Publishing Co (1997)
2. Rafiq, M. Y., Bugmann, G., Easterbrook, D. J.: Neural network design for engineering applications. Structures, vol. 79, no. 17, pp. 1541–1552 (2001) doi: 10.1016/s0045-7949 (01)00039-6
3. Ruder, S.: An overview of gradient descent optimization algorithms (2016) doi: 10.48550/arXiv.1609.04747
4. Zhang, Z.: Improved adam optimizer for deep neural networks. In: IEEE/ACM 26th International Symposium on Quality of Service, pp. 1–2 (2018) doi: 10.1109/iwqos. 2018.8624183
5. Sankararaman, K. A., De, S., Xu, Z., Huang, W. R., Goldstein, T.: The impact of neural network overparameterization on gradient confusion and stochastic gradient descent. In: International conference on machine learning, pp. 8469–8479 (2020) doi: 10.48550/ARXIV.1904.06963
6. Dogo, E. M., Afolabi, O. J., Nwulu, N. I., Twala, B., Aigbavboa, C. O.: A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In: International Conference on Computational Techniques, Electronics and Mechanical Systems, pp. 92–99 (2018) doi: 10.1109/CTEMS.2018.8769211
7. Talbi, G.: Metaheuristics: From design to implementation. John Wiley and Sons (2009)
8. Kennedy, J., Eberhart, R. C.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, vol.5, pp. 4104–4108 (1997) doi: 10.1109/ICSMC.1997.637339
9. Mitchell, M.: Genetic algorithms: An overview. Complex, vol. 1, no. 1, pp. 31–39 (1995)
10. Karaboga, D., Basturk, B.: Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. Lecture Notes in Computer Science, pp. 789–798 (2007) doi: 10.1007/978-3-540-72950-177
11. Storn, R., Kenneth, P.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, vol. 11, no. 4, pp. 341–359 (1997) doi: 10.1023/a:1008202821328
12. Chong, H. Y., Yap, H. J., Tan, S. C., Yap, K. S., Wong, S. Y.: Advances of metaheuristic algorithms in training neural networks for industrial applications. Soft Computing, vol. 25, no. 16, pp. 11209–11233 (2021 )doi: 10.1007/s00500-021-05886-z

13. Carvalho, M., Ludermir, T. B.: Particle swarm optimization of neural network architectures and weights. In: 7th International Conference on Hybrid Intelligent Systems, pp. 336–339 (2007) doi: 10.1109/his.2007.45
14. Aljarah, I., Faris, H., Mirjalili, S.: Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing, vol. 22, no. 1, pp. 1–15 (2016) doi: 10.1007/s00500-016-2442-1
15. Si, T., Bagchi, J., Miranda, P. B. C.: Artificial neural network training using metaheuristics for medical data classification: An experimental study. Expert Systems with Applications, vol. 193, pp. 116423 (2022) doi: 10.1016/j.eswa.2021.116423
16. Yang, S., Ting, T. O., Man, K. L., Guan, S. U.: Investigation of neural networks for function approximation. Procedia Computer Science, vol. 17, pp. 586–594 (2013) doi: 10.1016/j.procs.2013.05.076