ENTRENAMIENTO DE UNA RED NEURONAL ARTIFICIAL USANDO EL ALGORITMO SIMULATED ANNEALING

RESUMEN
La modificación de los pesos y bias con un algoritmo de entrenamiento de un perceptrón multicapa es un problema clásico de programación no lineal irrestricto. El presente trabajo muestra el desempeño del "Simulated Annealing" como algoritmo de entrenamiento de esta red neuronal resolviendo dos problemas clásicos en redes neuronales, el problema de la "Codificación" y el problema de la "Doble Espiral". Resultados de buena calidad son obtenidos cuando se compara esta propuesta frente a algoritmos clásicos de entrenamiento.

PALABRAS CLAVES: Optimización no lineal irrestricta, Algoritmos Heurísticos, Algoritmos Combinatoriales, Redes Neuronales Artificiales, Simulated Annealing.

ABSTRACT
This paper uses a Simulated Annealing algorithm for training an Artificial Neural Network. The efficiency of this algorithm is testing using two well know benchmark problems. Results are compared with BackPropagation algorithm and they show how this technique can resolve no-linear problem with this technique.

KEYWORDS: Simulated Annealing.

GERMAN ALONSO GÓMEZ ROJAS
Estudiante Maestría Ingeniería Eléctrica
gergo@utp.edu.co

JUAN CARLOS HENAO LÓPEZ
Estudiante Maestría Ingeniería Eléctrica
henaojuancarlos@hotmail.com

HAROLD SALAZAR ISAZA
Profesor Facultad de Ingeniería Eléctrica - UTP
hsi@utp.edu.co

Grupo de Planeamiento en Sistemas Eléctricos
Universidad Tecnológica de Pereira

1. INTRODUCCIÓN

Entrenar una red neuronal artificial (RNA) es un proceso que modifica el valor de los pesos y bias asociados a cada neurona [3, 6] con el fin de que la RNA pueda a partir de unos datos presentados en la entrada, generar una salida; en el caso del aprendizaje supervisado [8], se tiene un conjunto de datos pero no se conoce la función o relación matemática que los representa; al propagar hacia delante cada uno de estos patrones se obtiene una respuesta en la salida de la RNA la cual se compara con la salida deseada, permitiendo obtener el error del desempeño de la red. Existen en la literatura especializada diversos algoritmos con capacidad de entrenar redes neuronales [3, 6], sin embargo, tienen como desventaja requerir grandes tiempos computacionales para redes de mediano y gran tamaño y en muchos casos se obtiene una pobre generalización. Los desarrollos alcanzados por los algoritmos combinatoriales en la solución de problemas de gran complejidad matemática [1, 4] permiten orientar el presente trabajo al estudio de la optimización de las respuestas cuando se intenta entrenar una RNA empleando el algoritmo combinatorial Simulated Annealing.

La búsqueda de la mejor solución al problema del entrenamiento de una RNA es un problema de programación no lineal de variables irrestrictas [9]. Para encontrar la solución a este tipo de problema existen métodos exactos y los algoritmos combinatoriales, los cuales han resuelto con éxito muchos problemas en el campo de optimización matemática. Este trabajo presenta el uso del "Simulated Annealing" (SA), mostrando un muy buen desempeño resolviendo problemas de gran complejidad matemática [5]. Para probar la eficiencia y calidad de las respuestas el método se aplica al entrenamiento de RNA en los problemas de la codificación y el problema de las dos espirales.

El desempeño del SA depende fundamentalmente de mecanismos que le permiten explorar de forma extensiva e intensiva algunas regiones dentro del espacio de soluciones posibles [10]; estos mecanismos son la temperatura del proceso y la cadena de Markov y la exploración de las regiones se hace usando una estructura de vecindad, la cual es un algoritmo que genera un conjunto de soluciones cercanas a la solución actual que evalúa el proceso SA. La estructura de vecindad para la RNA permite obtener otra RNA idéntica a la anterior excepto en uno de los pesos o bias que ha cambiado ligeramente con respecto a su valor anterior. La modificación de estos pesos y bias se plantean en este trabajo de dos formas; la primera forma consiste en modificar de forma heurística los pesos y bias a través de un incremento constante; la segunda forma, más determinística, usa la información del gradiente propagando el error hacia atrás.

2. REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales buscan simular las tres características básicas de una red neuronal biológica [6]; procesamiento paralelo, memoria distribuida y adapatabilidad; lo anterior les permite a las redes neuronales artificiales aprender y generalizar a partir de

Fecha de recepción: 29 Marzo de2004
Fecha de aceptación: 16 Abril de 2004

un conjunto de datos de relación matemática desconocida. Existen muchas definiciones para las RNA, usando la planteada por [6] se establece que una red neuronal artificial es un grafo dirigido que cumple con las siguientes propiedades:

- A cada nodo *i* se le asocia una variable de estado y_i
- A cada conexión *(i,j)* de los nodos *i* y *j* se le asocia un peso sináptico w_{ji} donde w_{ji} . *R*
- A cada nodo *j* se le asocia un bias b_j . *R*
- Para cada nodo *j* se define una función . $_j(y_i, w_{ji}, b_j)$ que depende de los pesos, conexiones, bias y estados de los nodos *i*, conectados a él.

La figura 1, muestra la estructura de una neurona artificial que pertenece a la capa de salida de una RNA multicapa.

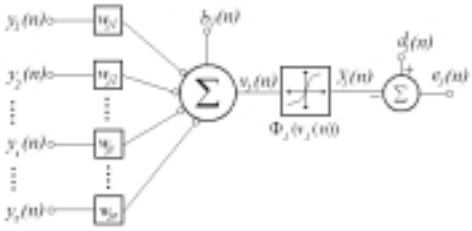

Figura 1. Modelo de una neurona en una RNA

n Denota una iteración cuando se propaga en la red el n-ésimo patrón del conjunto de entrenamiento
$y_j(n)$ Salida de la neurona *j*
$w_{ji}(n)$ Peso de conexión de la entrada *i* en la neurona *j*
$b_j(n)$ Bias de la neurona *j*
$v_j(n)$ Valor de activación de la neurona *j*
. $_j$ Función de transferencia de la neurona *j*
$d_j(n)$ Salida esperada de la neurona *j* al propagar el patrón *n*
$e_j(n)$ Error en la salida de la neurona *j* al propagar el nésimo patrón

Un algoritmo de entrenamiento de una RNA modifica el valor de los pesos de acuerdo con la expresión (1):

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \quad (1)$$

donde . w_{ji} es el valor que se obtiene por medio de una regla que define el algoritmo de entrenamiento. Si *E(n)* es el error de la red cuando se propaga el nésimo patrón de entrenamiento se tienen dos posibilidades de medir el desempeño de la red.

$$E(n) = \frac{1}{2N}\sum_{z}^{N}\sum_{j}(e_j(n)^Z)^2 \quad (2)$$

$$E(n) = \frac{1}{2}\sum_{j} e_j(n)^2 \quad (3)$$

La expresión (2) define el error global en la RNA sobre todo el conjunto de aprendizaje o sobre los *N* patrones, mientras que la expresión (3) define el error medio cuadrático (MSE) instantáneo o de un patrón y es una aproximación a la ecuación (2) con la ventaja de requerir un menor esfuerzo computacional.

El algoritmo de propagación hacia atrás (BP) modifica $w_{ji}(n)$ respecto a *E(n)* usando la información de primer orden (3). Es decir:

$$\Delta w_{ji}(n) = \frac{\partial E(n)}{\partial w_{ji}(n)} \quad (4)$$

Utilizando (3), el algoritmo de propagación hacia atrás establece que [3]:

$$\Delta w_{ji}(n) = -\delta_j(n) y_i(n) \quad (5)$$

Siendo . $_j(n)$ el gradiente local. Si *j* es una neurona de la capa de salida

$$\delta_j(n) = e_j(n)\Phi_j'(v_j(n)) \quad (6)$$

Si la neurona *j* pertenece a una capa oculta

$$\delta_j(n) = \Phi_j'(v_j(n))\sum_{k}\delta_k(n) w_{kj}(n) \quad (7)$$

Por lo tanto algoritmo BP modifica iterativamente los pesos con las ecuaciones (5) y (6) hasta que (2) o (3) alcance un valor de tolerancia lo cual significa que la red ha aprendido el conjunto de entrenamiento.

3. SIMULATED ANNEALING

El proceso SA se formula sobre el concepto que cuando una estructura se calienta, sus moléculas empiezan a moverse con altas velocidades debido a la gran cantidad de energía, con el tiempo y por la pérdida de energía, dichas moléculas poserán un movimiento más lento y se acomodaran permitiendo establecer estructuras cristalinas simétricas [5, 10]. El proceso SA se basa en el algoritmo de Metrópolis, el cual genera un conjunto de estructuras vecinas a partir de una estructura actual con sus respectivos niveles de energía; a su vez el algoritmo de metrópolis se basa en el método de Montecarlo que asigna una función de probabilidad de aceptar los cambios hechos en la estructura.

Si se supone una estructura que se encuentra en el estado *i* a un nivel de energía E_i el cual es un indicativo de su comportamiento; obtener otra estructura a partir de este estado energético actual, consiste en efectuar modificaciones que la llevan al estado energético E_j; decidir si la nueva estructura se acepta en dicho estado depende de un criterio de Montecarlo, que asigna la función de probabilidad (8) para aceptar o no este cambio.

$$P(\text{aceptación}) = \begin{cases} 1, & si\ E_j < E_i \\ e^{(E_i - E_j)/T}, & si\ E_j > E_i \end{cases} \quad (8)$$

Siendo *T* la temperatura a la cual se encuentra la estructura, *P* la probabilidad de aceptación. Es importante notar que cuando la temperatura es muy elevada, la probabilidad de aceptar este nuevo estado, así sea de peor calidad que el anterior, es cercana a uno (altamente probable de aceptar), y si la temperatura es

baja, es poco probable que se acepte. Esta propiedad que tiene el algoritmo de aceptar configuraciones cuya función objetivo sea de peor calidad, permite al método explorar en diversas regiones del espacio de soluciones sin quedar atrapado en los óptimos locales.

La figura 2 es una representación aproximada del comportamiento del algoritmo cuando explora diversas regiones en un problema de optimización; la propiedad de aceptar soluciones de peor calidad le permite al método "escalar" en las regiones de óptimos locales para finalmente alcanzar el óptimo global del sistema [10], que por lo general es la configuración que tiene el menor valor en la función objetivo o menor estado energético; la figura 3 muestra la relación entre la probabilidad de aceptar soluciones de peor calidad y la temperatura de la estructura; a altas temperaturas la probabilidad de aceptación es alta, con la disminución de la temperatura se disminuye igualmente la probabilidad de aceptación.

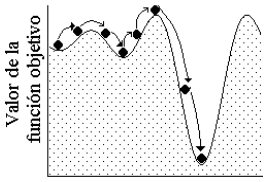

Figura 2. Comportamiento del algoritmo SA

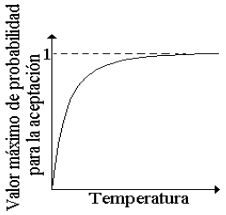

Figura 3. Probabilidad de aceptación de un nuevo estado en el algoritmo de Metrópolis

A medida que la temperatura baja, también lo hace la probabilidad de aceptación y es necesario aumentar el número de posibilidades a evaluar; este número de posibilidades a evaluar en cada paso del proceso iterativo se conoce con el nombre de la cadena de Markov. Cada uno de los elementos dentro de la cadena de Markov es una estructura o configuración que difiere muy poco del la actual, y es evaluado por el algoritmo Simulated Annealing. Existen dos expresiones que permiten calcular la longitud de la cadena de Markov en la próxima iteración

$$N_{k+1} = \eta N_0 \quad (9)$$

donde N_0 es la longitud inicial de la cadena de Markov, la cual para este problema toma valores entre 1 y 4 y . *es* una variable mayor a uno. También es posible calcular la longitud de la cadena con la expresión:

$$N_{k+1} = \rho N_k \quad (10)$$

donde la longitud de la nueva cadena (N_{k+1}) depende del anterior (N_k) y . es un valor constante que hace que la cadena crezca en un porcentaje constante; tanto . como . son valores determinados por la naturaleza y complejidad del problema.

Algoritmo 1.
Calcular *(N_o , T_o)*
$N_k=N_o$
Genere una configuración inicial S_o
Haga mientras no se cumpla el criterio de parada
Haga desde *k=1* hasta N_k
Genere una configuración S_j a partir de S_i
Si E_j . E_i
$S_i = S_j$
Sino
Si Aleatorio< $e^{(E_i-E_j)/T}$
$S_i = S_j$
Fin si
Fin Si
$k=k+1$
Fin Haga
Calcular *(N_k,T_k)*
Criterio de Parada
Fin Haga
Fin Programa

El calculo de la temperatura inicial es un factor determinante en el éxito del algoritmo; el valor inicial de la temperatura *(T0)* se obtiene con una heurística constructiva, simulando el número de transiciones hechas para la primera cadena de Markov, lo cual da un estimativo del número y calidad de las degradaciones o mejoramientos en la función objetivo. El algoritmo que se emplea para el cálculo de la temperatura inicial es el siguiente:

Algoritmo 2
Inicio Programa
Inicie $T_o=0$; $m_1= 0$; $m_2 = 0$; . $F=0$
Genere una configuración S_i
Haga desde *k=1* hasta N_o
Genere una configuración S_j a partir de S_i
Si E_i . E_j
$m_1 = m_1 + 1$
Sino
$m_2=m_2+1$
. $E=E_i-E_j$
Fin Si
$S_i=S_j$
Fin Haga

$$\Delta E+ = \sum_{k=1}^{m_2} \frac{\Delta E}{m_2}$$

$$T_0 = \frac{\Delta E+}{ln \dfrac{m_2}{m_2 X - m_1(1-X)}}$$

Fin Programa

T_0 es la salida final de este programa, y es la temperatura con que inicia el proceso SA. En este algoritmo $.E+$ es una medida de la degradación de la función objetivo, m_1 es el número de transiciones de i a j donde mejora la función objetivo, m_2 es el número de transiciones de i a j donde empeora la función objetivo y x es una tasa de aceptación que varía entre 0.8 y 0.9 [5].

4. ALGORITMO SA IMPLEMENTADO EN EL ENTRENAMIENTO DE UNA RNA

En una RNA se desea encontrar una configuración óptima de pesos de tal manera que ella pueda aprender un conjunto de patrones. El entrenamiento se convierte en un problema de programación no lineal, donde las variables de estado, los pesos sinápticos, son variables irrestrictas [9]. La función objetivo es minimizar el error medio cuadrático instantáneo por patrón.

$$\min E(n)$$

$$sa$$

$$E(n) = \frac{1}{2}\sum_j e_j(n)^2 \qquad (11)$$

$$w_{ji} \;\; irrestricto$$

El algoritmo SA para el problema de entrenamiento de una RNA consta de los siguientes pasos.

Paso 1: Definir la arquitectura de la RNA según el problema a resolver. Según la naturaleza del problema, fácilmente se puede determinar la cantidad de neuronas en la capa de entrada y en la capa de salida, sin embargo el número de capas ocultas y la cantidad de neuronas en estas capas no sigue ningún patrón o regla definida y se atiende más a la experiencia en la solución de problemas similares y a la complejidad matemática del problema a resolver; en el caso del problema de la codificación dura y en el problema de las dos espirales se parten de las configuraciones de RNA proporcionadas por [7].

Paso 2: Encontrar el MSE de la RNA. Una vez definida la red neuronal, se inicializan para todas y cada una de las neuronas los pesos y los bias; para esto se puede asignar a los pesos y bias un mismo valor, o también asignar valores diferentes usando alguna heurística constructiva; una vez se tiene inicializada la RNA se pueden propagar los patrones para así encontrar un primer valor para la función objetivo, que en este caso es el MSE del desempeño de la red. Este es el punto de partida para el algoritmo SA.

Paso 3: Iniciar el Proceso SA. Es necesario definir la cadena inicial de Markov; como en la determinación de la estructura de la RNA, la longitud de la cadena de Markov atiende a la experiencia en la solución de problemas similares y a la complejidad del problema propio; para los dos problemas de ensayo propuestos en el presente artículo, la cadena inicial de Markov oscila entre tres y cinco configuraciones. La temperatura inicial del proceso igualmente debe ser determinada, para esto se utiliza el algoritmo 2. Para cada cadena de Markov y en la temperatura dada se generan varios estados posibles a través de una estructura de vecindad. Los pesos y los bias de todas y cada una de las neuronas se pueden organizar en un conjunto de matrices; el algoritmo SA genera estos nuevos estados partiendo de una configuración inicial de matrices, escogiendo de forma aleatoria una de ellas e igualmente escogiendo de forma aleatoria cualquiera de las posiciones de memoria de la matriz escogida para realizar un cambio pequeño al valor que posee actualmente. La forma de hacer este cambio se orienta de dos formas. La primera consiste en realizar la actualización del peso escogido sumando un pequeño valor ($. w$) y evaluando el desempeño de la red, luego se resta al peso original el mismo incremento y se opta por aquella configuración que reduzca el MSE; esto es lo que plantea por la ecuación (1). A medida que se reduce el error, es necesario reducir este incremento con el fin de que el algoritmo realice una exploración más intensiva y así pueda converger a buenas soluciones Esta forma de entrenar la RNA es la que se denomina en el presente artículo como *entrenamiento SA*. La segunda forma consiste en escoger de forma aleatoria el peso a actualizar, pero para este peso se calcula su gradiente usando el algoritmo de propagación hacia atrás (BP), según la ecuación (5), y se reemplaza en la ecuación (1); este método se denomina como algoritmo de *entrenamiento SA-BP*.

Si la nueva configuración tiene un MSE menor que la anterior, se acepta esta nueva configuración; de lo contrario se acepta con una probabilidad definida por la expresión (8). Una vez se haya completado toda la cadena de Markov, se actualiza la temperatura que tiene efecto directo sobre la aceptación de nuevas configuraciones y también se calcula la nueva cadena de Markov para realizar una nueva iteración El proceso continúa hasta que se cumple algún criterio de parada; el algoritmo puede emplear uno o varios de los siguientes criterios de parada.

- El algoritmo alcanza un MSE predefinido
- La temperatura llegó a cierto valor mínimo predefinido
- No se cumple con un mínimo de aceptaciones de nuevas configuraciones para alguna cadena de Markov.

5. PRUEBAS Y RESULTADOS

Para medir el desempeño del algoritmo se tienen dos problemas clásicos de RNA tradicionalmente usados para este propósito, el problema de la codificación y el problema de la doble espiral Como algoritmo de entrenamiento se plantea dos alternativas y se comparan con un algoritmo clásico de BP. Estas dos alternativas son el algoritmo de entrenamiento SA y el algoritmo de entrenamiento SA-BP

El problema de la codificación sirve para comparar el entrenamiento SA y el BP, mientras que el problema de la doble espiral compara los algoritmos SA-BP y BP.

5.1. PROBLEMA DE LA CODIFICACIÓN

El problema consiste en medir el desempeño de una RNA con arquitectura N-M-N, donde el número de neuronas en las capas de entrada y salida del sistema sean igual al número de datos a transmitir, y que en la capa oculta se tenga M neuronas, con la condición de que M<N. Ahora, si $M \leq \log_2 N$ el problema recibe el nombre de codificación fuerte, en caso contrario la codificación es de tipo suave.

La figura 4 muestra la arquitectura tipo N-M-N, N neuronas en la capa de entrada, N neuronas en la capa de salida y M neuronas en la capa oculta, usada para transmitir bits de información.

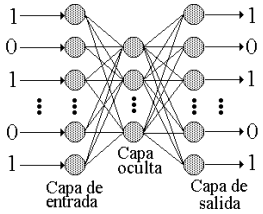

Figura 4. Red Neuronal con arquitectura 5-3-5

La arquitectura definida en la RNA para probar el algoritmo SA, posee 3 capas con 12 neuronas en las primeras capa, 2 neuronas en la capa intermedia y 12 neuronas en la capa de salida. Como conjunto de entrenamiento se escogen las combinaciones binarias que posean un uno lógico en cualquier posición y cero en las demás.

El entrenamiento SA consiste en escoger un $\Delta w o \Delta b$ constante y sumarlo o restarlo mientas la temperatura sea elevada o el error sobre el desempeño de la red sea grande, pero reducirlo paulatinamente mientras se enfría el proceso o se vaya reduciendo el error; para esta aplicación se tienen entonces cuatro pasos, el primero de ±0.1, el segundo de ± 0.05, el tercero de 0.005 y el cuarto de ± 0.0001. Estos valores le dan velocidad y agilidad al método y permite realizar una búsqueda extensiva e intensiva en regiones de interés. La figura 5 muestra el desempeño de la RNA en los distintos entrenamientos usando los algoritmos SA y BP en el problema de la codificación. Se observa que el proceso SA converge rápidamente a un valor adecuado de la función objetivo aunque el entrenamiento BP lo hace más rápido, más no muestra mejorar a medida que avanzan las iteraciones, lo que sí sucede con el entrenamiento SA.

Algoritmo	MSE objetivo	MSE alcanzado	Iteraciones	t(s)
BP	0.001	0.001	14000	100
SA	0.001	0.001	13200	920
SA-BP	0.001	0.001	24000	560

Tabla 1. Entrenamiento de una red neuronal artificial para el problema de la codificación 12-2-12

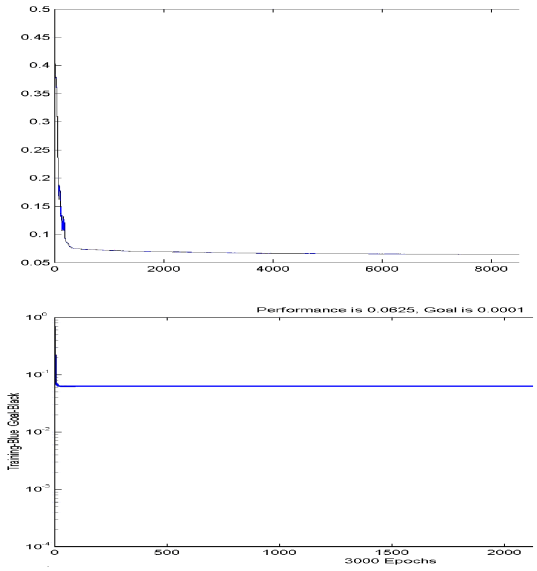

Figura 5. Comportamiento de los algoritmos SA y BP

La tabla 1 muestra los resultados en cómputo de los algoritmos de entrenamiento.

Se puede concluir que los tres algoritmos convergen al mismo valor en MSE para este problema particular y que BP lo hace significativamente más rápido que los demás, sin embargo, los otros dos algoritmos que usan SA aunque que tardan más, pueden seguir explorando en el espacio de soluciones pues ellos no quedan atrapados en óptimos locales y así mejorar la respuesta.

5.2. PROBLEMA DE LAS DOS ESPIRALES

Este problema consiste en dos espirales que giran en sentidos contrarios; las espirales alcanzan a dar tres vueltas y de cada una se tienen 194 puntos; la idea consiste en que una vez entrenada la RNA, esta reconozca a cual de las dos espirales pertenece un punto dado; este es un problema de mayor complejidad. La figura 6 muestra las dos espirales de Arquímedes.

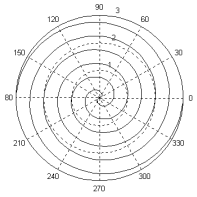

Figura 6. Espirales de Arquímedes

La arquitectura definida para esta RNA posee 4 capas con 5 neuronas en las tres primeras capas y 1 neurona en la capa de salida. La figura 7 muestra el desempeño de los entrenamientos usando los algoritmos SA y BP en este problema. La tabla 2 muestra los resultados en cómputo de los algoritmos de entrenamiento.

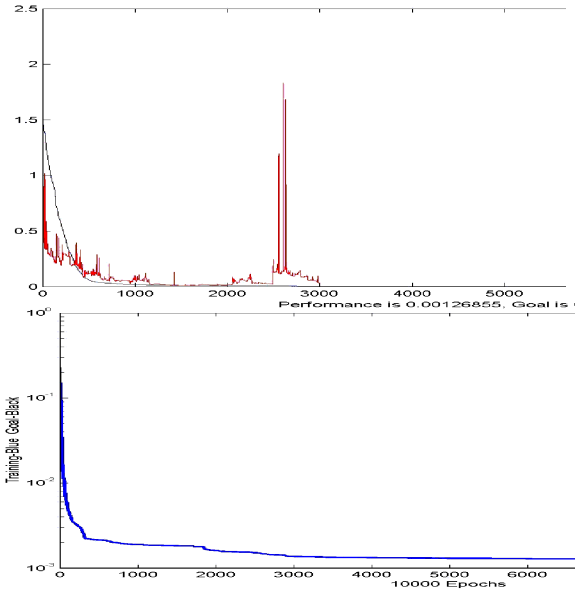

Figura 7. Comportamiento de los algoritmos SA-BP y BP

Algoritmo	MSE objetivo	MSE alcanzado	Iteraciones	t(s)
BP	0.001	0.001	10000	100
SA	0.001	0.001	31000	610
SA-BP	0.001	0.001	18000	240

Tabla 2. Entrenamiento de una red neuronal artificial para el problema de las dos espirales

Las conclusiones que se obtienen a partir del problema de la doble espiral son semejantes a las que se obtienen del problema de la codificación; BP es un algoritmo más rápido, pero SA y SA-BP pueden explorar de forma más intensiva; adicionalmente, cuando se usan la información del gradiente, el algoritmo SA se vuelve más rápido acercándolo en velocidad al resultado obtenido por el algoritmo BP.

T_0	N_0	.	Tasa de enfriamiento	t(s)
0.25	2	5%	10%	190
0.25	10	1%	5%	260
100	2	5%	10%	1100
100	50	1%	5%	4600
0.01	2	5%	10%	Inf
0.01	50	1%	5%	Inf

Tabla 3. Efectos de la temperatura, tasa de enfriamiento y incremento de la cadena de Markov sobre la respuesta.

Valores de temperatura entre 0,5 y 1.0 con tasas de enfriamiento mayores a 0.9 y cadenas iniciales de Markov de menos de 4 configuraciones y tasas de incremento cercanas al 5% hacen que el algoritmo converja de forma más eficiente.

6. CONCLUSIONES

El entrenamiento de una red neuronal artificial usando un algoritmo BP o un algoritmo combinatorial en problemas con arquitectura sencilla converge a los mismos resultados en el valor de MSE.

Aunque el tiempo de cómputo es mas elevado en el proceso SA, se llegan a soluciones de mejor calidad para la función objetivo con este entrenamiento, mientras que algunos algoritmos BP quedan atrapados en determinadas regiones del espacio de soluciones posible. Se observa claramente en las gráficas 5 y 6 que SA camina en regiones de función objetivo de peor calidad y sin embargo, esto le da la posibilidad de construir estructuras de muy buena calidad. Además se observa en las pruebas realizadas, que la determinación de la temperatura inicial del proceso, el programa de enfriamiento, la cadena inicial de Markov y el incremento de la misma tienen un efecto directo en la velocidad de convergencia pero eventualmente se pueden llegar a ellas.

7. AGRADECIMIENTOS

Los autores expresan su agradecimiento a la Universidad Tecnológica de Pereira, Colombia, por su apoyo al grupo de Planeamiento de Sistemas Eléctricos de la Maestría en Ingeniería Eléctrica.

8. BIBLIOGRAFÍA

[1] Gallego R.A., Romero R.A., Escobar A. (2003). "Algoritmos Genéticos", Primera edición, Maestría en Ingeniería Eléctrica, Universidad Tecnológica de Pereira.

[2] Sanz A., Martín del Ríos B. (2002). "Redes Neuronales y Sistemas Difusos", Alfa omega, 2ª edición.

[3] Haykin Symon. (1999). "Neural Network. A Comprehensive Foundation". 2nd edition. Prentice-Hall.

[4] Gallego R. A. (1997). "Planejamento a Longo Prazo de Sistemas de Transmissao Usando Técnicas de Otimizaçao Combinatorial", Tese de Doutorado, UNICAMP.

[5] R Romero, R A Gallego, and A Monticelli. (1996). "Transmission System Expanssion Planning by Simulated Annealing", IEEE Trans on Power System, Vol 11, No 1, pp 364-369.

[6] Hilera J. R., Martínez V. J.(1995) "Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones". Ra-ma Editorial. Madrid.

[7] Riedmiller M. (1994). "Advanced Supervised Learning in Multi-Layer Perceptrons From Backpropagation to Adaptive learning Algorithms". Computer Standards and Interfaces, 5. Special Issue on Neural Networks .

[8] Mollen M.F. (1993). "A scaled conjugate gradient algorithm for fast supervised learning", Neural Networks, 6(3), pag 525-533.

[9] Bazaraa M. S, Sherali H. D and Shetty M. C. (1993). "Nolinear Programming Theory and Alorithms". 2nd edition. Wiley.

[10] Aarts E., Korst J. (1989). "Simulated Annealing and Boltzmann Machines", John Wiley & Son.