ENTRENAMIENTO DE RED NEURONAL ARTIFICIAL MORFOLÓGICA DE DENDRITAS CON ALGORITMO DE OPTIMIZACIÓN POR INTELIGENCIA DE ENJAMBRES

OMAR JORDÁN JORDÁN

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA
CALI – COLOMBIA
2018

ENTRENAMIENTO DE RED NEURONAL ARTIFICIAL MORFOLÓGICA DE DENDRITAS CON ALGORITMO DE OPTIMIZACIÓN POR INTELIGENCIA DE ENJAMBRES

OMAR JORDÁN JORDÁN
Código 1038239

Anteproyecto de Trabajo de Grado

Directores:
EDUARDO FRANCISCO CAICEDO BRAVO, Ph.D.
WILFREDO ALFONSO MORALES, Ph.D.

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
PROGRAMA ACADÉMICO DE INGENIERÍA ELECTRÓNICA
CALI – COLOMBIA
2018

FICHA RESUMEN DEL TRABAJO

Título:	ENTRENAMIENTO DE RED NEURONAL ARTIFICIAL MORFOLÓGICA DE DENDRITAS CON ALGORITMO DE OPTIMIZACIÓN POR INTELIGENCIA DE ENJAMBRES
Área temática o línea de investigación del Trabajo de grado:	REDES NEURONALES, INTELIGENCIA ARTIFICIAL
Modalidad de trabajo de grado:	**Investigación e Innovación**
Duración del trabajo de grado:	10 MESES
Entidades participantes:	…
Áreas Académicas EIEE:	□ Arquitecturas Digitales □ Bionanoelectrónica □ Conversión de Energía □ **INFORMÁTICA INDUSTRIAL** □ Sistemas de control y accionamientos □ Sistema de Potencia □ Telecomunicaciones
Grupos de investigación:	□ Alta Tensión (GRALTA) □ Arquitecturas Digitales y Microelectrónica (GADYM) □ Bionanoelectrónica □ control industrial (GICI) □ Conversión de Energía (CONVERGIA) □ **PERCEPCIÓN Y SISTEMAS INTELIGENTES (PSI)** □ Sistemas de Telecomunicaciones (SISTEL-UV) □ Sistemas Hidroeléctricos de Generación (SHG) □ Otro:

Estudiante	
Nombre: Omar Jordán Jordán	Código: 1038239
Correo electrónico: omar.jordan@correounivalle.edu.co	Tel.: 3173499624
Director	
Nombre: EDUARDO FRANCISCO CAICEDO BRAVO, Ph.D.	
Institución: Universidad del Valle	
Correo electrónico: eduardo.caicedo@correounivalle.edu.co	Tel:
Director	
Nombre: WILFREDO ALFONSO MORALES, Ph.D.	
Institución: Universidad del Valle	
Correo electrónico: wilfredo.alfonso@correounivalle.edu.co ó wilfredo.morales@correounivalle.edu.co	Tel.:

TABLA DE CONTENIDO

RESUMEN

En este trabajo se propone implementar el algoritmo de optimización por enjambre de partículas (PSO), para la sintonización de pesos sinápticos que mejoren las capacidades de clasificación de una red neuronal morfológica de dendritas (DMNN), con el propósito de identificar la respuesta y/o mejora en las opciones de entrenamiento y por lo tanto ampliar la aplicabilidad de la misma. Adicionalmente, se pondrá en contraste con los algoritmos de gradiente descendente estocástico (SGD) y evolución diferencial (DE), dado que son los trabajos más recientes y significativos en el estado del arte. La comparación se medirá a través de problemas de clasificación usando métricas de desempeño obtenidas de las matrices de confusión.

1 INTRODUCCIÓN

Los avances en neurobiología proponen modelos más complejos de la neurona orgánica, planteando que en las dendritas se lleva a cabo parte importante del procesamiento de la información, incluso se dice que quizá la parte más importante. Luego de varios estudios en dicho campo y de primeras propuestas matemáticas, en 2003 se desarrolló un perceptrón con dendritas [22], donde se mostró claras ventajas sobre un perceptrón clásico de cero o más capas ocultas:

- Puede resolverse la XOR sin necesidad de capas ocultas.
- Encierra las regiones de decisión alrededor de los patrones (clústers), mientras una red MLP con una capa oculta requiere más neuronas en dicha capa que neuronas de entrada tenga la red para que las superficies de decisión no queden abiertas.
- El procesamiento tanto software como hardware es más veloz y fácil de implementar por las operaciones morfológicas [21].
- Una de las mayores ventajas es que el número de dendritas crece con el entrenamiento, logrando clasificar todos los patrones, alcanzando un desempeño donde el error de entrenamiento es cero, es decir, no hay problema de convergencia y no necesariamente cae en sobre-entrenamiento.

Este sistema basado en dendritas, que se clasifica dentro de redes morfológicas, tienen ventajas que se podrían implementar para resolver problemas de clasificación sin tener que hacer una alta demanda en los recursos computacionales; sin embargo, la limitación de este tipo de arquitecturas se debe a sus algoritmos de entrenamiento. Este trabajo de grado reconoce las ventajas de los algoritmos de inteligencia computacional como mecanismos para la sintonización de parámetros en las arquitecturas de redes neuronales, por lo que se verificará la capacidad de sintonización de un algoritmo basado en enjambre de partículas respecto a otros métodos de entrenamiento basados en meta-heurísticas presentados recientemente: El método de gradiente descendente estocástico [8] y el algoritmo de evolución diferencial [9].

2 PLANTEAMIENTO DEL PROBLEMA

A la hora de buscar los parámetros que mejoren las características de clasificación de una red neuronal morfológica tipo DMNN se han utilizado algoritmos de optimización como SDG o algoritmos evolutivos como el DE, los cuales han presentado inconvenientes en las búsquedas estocásticas debido al fenómeno denominado estancamiento prematuro; es decir, solo consiguen llegar a soluciones óptimas locales.

El grupo de investigación en percepción y sistemas inteligentes (PSI) ha trabajado desde sus inicios en el campo de la inteligencia computacional, adquiriendo conocimiento de vanguardia en distintas áreas como lo son las Deep Networks y la Inteligencia de Enjambres, se busca en dicho campo llegar cada vez a soluciones óptimas, que requieran menos requisitos de hardware y por consiguiente menores costos; para esto las redes neuronales morfológicas ofrecen operaciones más simples de computar, pero los algoritmos de aprendizaje están aún en una importante etapa de investigación, ya que los valores de los parámetros de sintonización del modelo resultan de operaciones matemáticas, donde la sumatoria en una MLP es reemplazada por operadores máximo o mínimo, siendo evidente que la ecuación de esta neurona artificial no es derivable ni lineal, incluso desde antes de la función de activación [12]. A partir de esta necesidad se plantea entonces la siguiente pregunta:

¿Qué tan eficiente es la técnica inteligencia de enjambres PSO en la sintonización de parámetros de una red neuronal DMNN en comparación con los otros algoritmos recientemente utilizados (SGD, DE)?

3 JUSTIFICACIÓN

Se ha motivado a través de este trabajo, el conocer las características y las capacidades de las redes neuronales morfológicas de tipo DMNN en la solución de problemas de clasificación, las cuales están en la capacidad de crear límites de decisión complejas, permitiendo obtener una clasificación no-lineal con solo una neurona, lo cual no es posible con una red neuronal tipo MLP, dando a entender que las DMNN superan ampliamente las capacidades de las redes neuronales clásicas.

El uso de inteligencia de enjambres ofrece ventajas al enfocarse en el trabajo de búsqueda colectiva realizado por unidades simples, esto puede suponer una solución significativa al problema de estancamiento prematuro en mínimos locales, dado que es capaz de encontrar soluciones que un solo individuo podría no resolver.

Ahora que están en pleno desarrollo estas redes morfológicas, es necesario para grupos investigativos como PSI, hacer una apropiación del conocimiento para su reproducción y experimentación a nivel local, así como llevarlo a los estudiantes y profesionales interesados en esta área teniendo en cuenta que la institución tiene entre sus fines el desarrollo tecnológico e intelectual de la comunidad a nivel de ciudad y país.

4 OBJETIVOS

4.1 Objetivo General

Desarrollar un algoritmo de sintonización de los pesos sinápticos de una DMNN mediante un mecanismo de optimización por enjambre de partículas (PSO), el cual debe ser comparado con los algoritmos: Gradiente Descendente Estocástico (SGD) y Algoritmo de Evolución Diferencial (DE).

4.2 Objetivos Específicos

1. Realizar una búsqueda bibliográfica de los algoritmos de aprendizaje utilizados para la sintonización de una DMNN.

2. Desarrollar una herramienta software para poner a prueba la sintonización de parámetros de la red DMNN mediante los algoritmos SGD, DE y PSO.

3. Evaluar los algoritmos de aprendizaje para la red neuronal morfológica a través de la comparación de métricas de desempeño estandarizadas.

5 MARCO TEÓRICO

Inspirándose en los sistemas nerviosos pluricelulares biológicos que poseen múltiples unidades básicas de procesamiento interconectadas, surgen las redes neuronales artificiales, la cuales iniciaron en 1943 con Warren McCulloch y Walter Pitts. Análogo al comportamiento de una neurona biológica, la cual transmite la información a través de las dendritas y el axón para entregar una respuesta a diversos estímulos, la neurona artificial básica es modelada como una ecuación matemática que combina varias entradas ponderadas para generar una respuesta a los estímulos [13].

Esta propuesta simple posteriormente fue transformada en un sistema con múltiples capas (Multi-Layer Perceptrón), que logró resolver problemas de mayor complejidad, permitiendo obtener sistemas de clasificación no-lineal a partir de unidades simples de procesamiento. Aunque dicha arquitectura MLP ha tenido una aceptación enorme en la comunidad científica, otra propuesta como lo son las redes neuronales artificiales morfológicas (MNN) presentada por Davidson y Ritter, resultó mostrar un mejor comportamiento en los procesos de clasificación no-lineal con un modelo matemático más simple, al punto de resolver problemas de clasificación de patrones sin necesidad de usar capas ocultas como sí lo hace una red MLP.

Sin embargo, los esquemas de aprendizaje para este tipo de redes morfológicas aún siguen siendo tema de investigación con el ánimo de mejorar el proceso de sintonización de los parámetros, donde han prevalecido algoritmos como: gradiente descendente estocástico (SGD) y el algoritmo de evolución diferencial. Dada la gran aceptación de algoritmos basados en meta-heurísticas para la sintonización de estos parámetros, se podría intentar resolver este problema a través de algoritmos basados en inteligencia de enjambres, ya que evitan la convergencia prematura y la diversidad de soluciones, donde los métodos anteriormente mencionados podrían estancarse debido a las condiciones iniciales como única semilla en SGD o la incapacidad de afinación de la solución en algoritmos genéticos o evolutivos.

5.1 Redes Neuronales Artificiales (ANN)

En la estructura básica clásica de la neurona artificial, sean “n” entradas, la salida “y” es el resultado de una función de activación “f(x)” que puede ser o no derivable, lineal, continua, etc; entre las más comunes se encuentran: identidad (lineal), lineal a tramos, escalón, escalón binario, sigmoidal, sigmoidal binaria, gaussiana y rampa. Esta función recibe la sumatoria de el valor bias “b” (que evita matemáticamente que la solución pase necesariamente por el origen) con las entradas “x” multiplicadas por un peso sináptico “w” asociado a cada una de ellas. La solución es un hiperplano en el hiperespacio de entradas (combinación lineal de parámetros).

$$y = f\left(b + \sum_{i=1}^{n} x_i . w_i\right)$$

Sea una red neuronal artificial un conjunto de neuronas interconectadas, son los pesos sinápticos y el bias los parámetros que deben ser sintonizados correctamente para dar solución al problema presentado a la red, pues aquí se halla el “conocimiento / memoria” de la misma. Esto se puede llevar a cabo mediante aprendizaje supervisado o no supervisado, donde el primero presenta una serie de patrones a la entrada cuya salida deseada es conocida (conjunto de entrenamiento) mientras el segundo organiza los datos de entrada según reglas pre-impuestas.

En 1957 Frank Rosenblatt inventa el Perceptrón, cuya función de activación es el escalón binario y utiliza un algoritmo de aprendizaje basado en el error de salida; el peso sináptico “w” es modificado sumándole la diferencia entre el valor de salida deseada “d” y calculada “y”, multiplicada por la entrada asociada “x” y por un factor de aprendizaje “α”.

$$w_i(t+1) = w_i(t) + \alpha . \left(d_p - y_p\right) . x_{pi}$$

Esta funcionaba con una sola neurona o con una capa de estas (arreglo de neuronas) y presentó serios inconvenientes al resolver problemas de clasificación pues solo resolvía problemas de clasificación lineal e históricamente tuvo fuertes críticas por Minsky y Papert en 1969 por no poder resolver la compuerta básica XOR.

5.1.1 Perceptrón Multicapa (MLP)

Rosenblatt intuía que introduciendo capas de neuronas entre los nodos de entrada y las neuronas de salida podría resolverse la XOR, a estas capas se les llamó “capas ocultas” y en efecto más de éstas pueden clasificar patrones con formas geométricas más complejas. En 1975 Paul Werbos propone el algoritmo BackPropagation el cual puede ajustar sus pesos a partir del error de la salida usando derivación propagada hacia las neuronas ocultas (gradiente descendente) [13], se utiliza la función de error cuadrático medio para cada patrón “p” suponiendo “m” neuronas en la capa de salida.

$$\varepsilon_p = \frac{1}{2}.\sum_{k=1}^{m}\left(d_{pk} - y_{pk}\right)^2$$

El nuevo valor del peso “w” asociado a la “j-ésima” neurona de la capa anterior (entrada) está en función de la derivada parcial del error total de salida respecto del mismo peso “w” a ser modificado.

$$w_{kj}(t+1) = w_{kj}(t) - \alpha.\frac{\partial \varepsilon_p}{\partial w_{kj}(t)}$$

Aquí cabe resaltar que las funciones de activación deben ser derivables, suelen usarse para la red MLP las funciones sigmoidal e identidad (lineal); dado que usando regla de la cadena puede derivarse el error total de la capa de salida en función de pesos sinápticos de las capas ocultas.

Esta solución es muy efectiva y la más ampliamente utilizada hoy en día para resolver la mayoría de problemas básicos de clasificación con redes neuronales, siendo antecesora de otras técnicas como las Deep Networks que la superan ampliamente en contextos de mayor abstracción. Así mismo, hay otras estructuras de neurona artificial, como algoritmos utilizados para entrenamiento y topologías de conectividad.

5.1.2 Redes Neuronales Morfológicas (MNN)

Una concepción diferente de neurona artificial nace de los avances en la teoría del álgebra de imagen [14], la cual usa a su vez operaciones matemáticas morfológicas [15] de donde se resaltan las operaciones de Erosión y Dilatación pues son usadas por la neurona artificial morfológica. En ésta cambia las operaciones de multiplicación de los pesos sinápticos con las entradas en la neurona clásica por operaciones de suma y la sumatoria de dichas multiplicaciones es cambiada por operadores máximo o mínimo, con lo cual queda claro que la ecuación de esta neurona artificial no es derivable ni lineal incluso desde antes de la función de activación [12].

$$y = f\left(b + a_0 . \bigwedge_{i=1}^{n} a_i . (x_i + w_i)\right)$$

Luego parámetros extra que valen ±1 son añadidos, el “ai” multiplica cada sumatoria dentro de la función mínimo o máximo, habilitando o deshabilitando dicha entrada; el “a0” multiplica a toda la función mínimo o máximo con lo cual deshabilita o habilita la salida (ya que suele usarse la función de activación hard-limiter o escalón con este tipo de neuronas).

Las propiedades son diferentes a la clásica ANN, por ejemplo, un perceptrón monocapa basado en MNN puede ser entrenado con un número finito de iteraciones, evitando el problema de convergencia; además separa las clases a clasificar en forma de superficie de decisión correspondiente a un impulso infinito (como una curva de potencia en 2D) en lugar de una separación lineal como lo haría un clásico perceptrón.

Las MNN se han utilizado en el área de reconocimiento de patrones y como memorias asociativas; en el primer caso tienen una gran afinidad por el procesamiento de imágenes [16], [17], [18].

5.2 Redes Neuronales Morfológicas de Dendritas (DMNN)

Para explicar el funcionamiento de una neurona morfológica con dendritas se requiere primero de “n” neuronas en la capa inmediatamente anterior que proporcionen las entradas a la red “x”, a diferencia del perceptrón clásico donde todas las entradas llegan directamente al cuerpo de la neurona siguiente. Aquí dicho cuerpo posee una cantidad dada de dendritas o terminales de entrada “K”, estos enviarán su información al cuerpo de la neurona para ser operada con la función máximo o mínimo y luego pasar por la función de activación “f(x)” para generar la salida [22].

$$y = f\left(\bigwedge_{k=1}^{K}\left(p_k.\bigwedge_{i=1}^{n} l_{ik}.(x_i + w_{ik})\right)\right)$$

Cada neurona “n” de la capa anterior divide su salida “x” (axón) en varias ramas, donde cada una puede adherirse a la dendrita “k” de la neurona siguiente; a su vez cada rama del axón tiene su peso sináptico “w” con la dendrita y un parámetro “l” ±1 que denota excitación o inhibición. La salida “x” que viaja por el axón es sumada a dicho peso “w” y esta suma se multiplica por -1 si el parámetro “l” es inhibitorio. Todos estos axones que llegan a la dendrita son operados con la función mínimo o máximo y finalmente multiplicado por el parámetro “p” ±1 de excitación o inhibición de la dendrita. Este es el procesamiento de la dendrita, Luego ¿cuantas dendritas debe tener una neurona? ahí es donde radica el reto de entrenar la red generando la cantidad adecuada de dendritas para solucionar el problema, además de estimar todos los parámetros.

La interpretación geométrica es que cada dendrita crea un hipercubo en el hiperespacio [21], de aquí que un algoritmo de aprendizaje se basa en crear un hipercubo que englobe todos los patrones e irlo dividiendo iterativamente confirmando que cada hipercubo encierre solo una clase de los patrones, haciendo además combinación de hipercubos cercanos para optimizar. Finalmente, una vez todos los patrones están encapsulados, se genera la red neuronal asociando a cada dendrita uno de estos hipercubos [8]; en otros algoritmos este método se usa exclusivamente como un inicializador de la DMNN a causa de su menor capacidad de generalización y optimización en cuanto al número de dendritas [6].

5.3 Algoritmos para la sintonización de parámetros en una red DMNN

5.3.1 Gradiente Descendente Estocástico (SGD)

Desde la aparición del algoritmo BackPropagation, el cálculo a partir de métodos de gradiente para entrenar las redes neuronales ha sido la herramienta más usada para proponer soluciones eficazmente (porque opera sobre una sola red), además que genera mayor capacidad de generalización que otros algoritmos de naturaleza heurística [7]. Sin embargo, este método tiende a atascarse rápidamente en mínimos locales [19] y depende de la localización inicial de sus pesos sinápticos.

Los pesos sinápticos “w” de las capas son sintonizados mediante la sustracción iterativa de un parámetro de aprendizaje “α” multiplicado por el gradiente del error que se calcula a partir de las derivadas parciales del error cuadrático medio de la capa de salida “ε” respecto del peso sináptico “w” que está siendo modificado, mediante la regla de la cadena; la complejidad aumenta en la medida que la cantidad de capas ocultas y neuronas dentro de la red también aumenta [13].

$$-\alpha.\frac{\partial \varepsilon_p}{\partial w(t)}$$

El gradiente descendente estocástico no calcula en cada iteración el nuevo delta de cambio del peso sináptico para todos y cada uno de los patrones “p”; en su lugar acude a la aleatoriedad para seleccionar uno o varios patrones en cada iteración, haciendo que el algoritmo corra más rápido y es útil para aprender nuevos patrones sobre la marcha “online” [20]; esto agrega una fluctuación (ruido) en el descenso del error y tiene la ventaja de poder salir de mínimos locales cercanos.

```
repetir k hasta h:
        w(k) = rand
repetir:
        p = seleccionar patrón aleatorio
        yp = evaluar_red(xp)
        ep = 0.5 . sumatoria( ( dp - yp ) ^ 2 )
        repetir k hasta h:
                w(k) = w(k) - a . ( ∂.ep / ∂.w(k) )
        condición parada
```

En el algoritmo “xp, dp” son los datos de entrada “x” y salida deseada “d” del p-ésimo patrón, “ep” es el error cuadrático medio y la función “derivada” es simbólica puesto que su cálculo se hace previamente a la realización del algoritmo; también es de aclarar que este es el algoritmo básico puesto que parámetros como el alfa “a” se autoajustan en la marcha, (“h” representa la cantidad de parámetros en “w”).

Para utilizar el SGD con la red DMNN es necesario aplicar Softmax (función exponencial normalizada) en una capa, a causa de que las funciones máximo y mínimo no son derivables [7], esta función entrega valores en el rango 0–1 correspondientes a la distribución de probabilidad sobre K diferentes salidas; por esto es muy usada en clasificadores con ANN, su forma matemática es:

$$S(xi) = \frac{e^{xi}}{\sum_{j=1}^{K} e^{xj}}$$

5.3.2 Evolución Diferencial (DE)

El principio de funcionamiento de los algoritmos evolutivos, consiste en empaquetar los parámetros del problema a resolver como una cadena, que representa las características de una posible solución. Dicha cadena corresponde a un individuo y emula la idea del genoma biológico [1]. Por lo tanto, ante un número de individuos, seleccionados a través de diversos métodos, se produce una nueva generación de potenciales soluciones; entre más iteraciones (generaciones) transcurran, se iran alcanzando soluciones más óptimas. Las funciones aleatorias de selección, cruce y mutación juegan un rol importante y hacen que este tipo de algoritmos estén en el grupo de los llamados estocásticos.

La evolución diferencial usa la idea de perturbar con diferencias vectoriales la creación de una nueva generación [9], estos vectores diferenciales de perturbación “v” son creados a partir de (en la variante más básica) la selección de 3 individuos diferentes al azar “r1, r2 y r3” y su combinación matemática vectorial controlada por un parámetro “h” (tasa de mutación). La nueva generación se crea entonces en dos pasos; primero se compara componente a componente un valor aleatorio con un parámetro “c” (tasa de cruce) para escoger componentes de “x” o “v” y obtener los llamados vectores de prueba “u” (cruzados de los anteriores); luego éstos son comparados en desempeño (mediante la función de costo) con los individuos de la población actual “x” (de tamaño “m”) y son ordenados de tal manera que representan la nueva generación de individuos.

```
Repetir i hasta m:
        xi = rand
repetir:
        repetir i hasta m:
                r1 = rand % m
                r2 = rand % m
                r3 = rand % m
                vi = x(r1) + h . ( x(r2) – x(r3) )
                repetir k hasta tamaño(vi):
                        si rand < c
                                ui(k) = vi(k)
                        sino
                                ui(k) = xi(k)
        repetir i hasta m:
                si error(ui) < error(xi)
                        xi = ui
        condición parada
```

Este algoritmo ha dado resultados superiores a los clásicos algoritmos genéticos, y es ampliamente usado en el campo de la inteligencia computacional; por ejemplo, este algoritmo ha sido implementado para la sintonización de parámetros de una DMNN [6].

5.3.3 Optimización por Enjambre de Partículas (PSO)

Es parte de la familia de algoritmos meta-heurísticos. Dentro de este tipo de algoritmos se encuentran aquellos inspirados en el comportamiento de animales simples, que se mueven en colonias o enjambres, reconocido como Inteligencia de Enjambres (SI, por sus siglas en inglés). Dentro de los algoritmos SI, se destacan: Optimización por Enjambre de Partículas [10], Optimización por Colonias de Hormigas [4] y la Optimización por Forrajeo de Bacterias [3].

El enjambre de partículas [5], [11] funciona asociando a cada “k” individuo una posición “p” en el hiperespacio de soluciones y agregando una velocidad “v” que es afectada por su propio momento de inercia “m”, por su mejor posición alcanzada o cognitiva “b” (memoria local) y por la mejor posición social (solución) encontrada por todo el conjunto de partículas “g” (memoria global); los últimos dos elementos de la velocidad de los individuos están acompañados de parámetros ajustables “c1

y c2” así como aleatoriedad que agrega el componente estocástico. Se evalúan funciones de costo “error” para seleccionar las mejores posiciones alcanzadas.

```
g = rand
repetir i hasta k:
        vi = rand
        pi = rand
        bi = pi
repetir:
        repetir i hasta k:
                vi = vi . m + c1 . rand . (bi - pi) + c2 . rand . (g - pi)
                pi = pi + vi
                si error(pi) < error(bi)
                        bi = pi
                        si error(bi) < error(g)
                                g = bi
        condición parada
```

A diferencia de los algoritmos evolutivos que no tienen uso de memoria más allá de la propia generación actual, el PSO cuenta con la experiencia pasada de los individuos en cualquier momento de la ejecución [1] para mejorar su desempeño basado en la experiencia adquirida durante el proceso de búsqueda.

En general los algoritmos evolutivos, así como la inteligencia de enjambres y otros métodos de soluciones emergentes tienen su punto fuerte en problemas con funciones matemáticas de costo no diferenciables, no continuas, no lineales, multidimensionales (alta complejidad), con muchos mínimos locales o funciones no conocidas, ya que no se requieren el cálculo de expresiones analíticas como lo son los métodos de gradiente. Su debilidad radica en el costo computacional que representa la evaluación de las funciones de costo cuando el problema aumenta su dimensionalidad, así como el costo computacional relativo a la cantidad de individuos [2].

6 METODOLOGÍA

1. Realizar una búsqueda bibliográfica de los algoritmos de aprendizaje utilizados para la sintonización de una DMNN.
   - 1.1. Identificar la arquitectura y modelamiento de la red DMNN.
   - 1.2. Comprender el algoritmo SGD en asociación a la red morfológica.
   - 1.3. Comprender el algoritmo DE en asociación a la red morfológica.
   - 1.4. Comprender y diferenciar los algoritmos basados en Optimización de Enjambre de Partículas PSO.
   - 1.5. Presentar un informe de avance interno a los tutores del trabajo.
2. Desarrollar una herramienta software para poner a prueba la sintonización de parámetros de la red DMNN mediante los algoritmos SGD, DE y PSO.
   - 2.1. Definir el software para realizar el proceso de aprendizaje de las redes DMNN a través de las prestaciones computacionales y la accesibilidad de los paquetes de programación.
   - 2.2. Construir el algoritmo SGD para la sintonización de los parámetros de una red tipo DMNN sobre el software seleccionado.
   - 2.3. Construir el algoritmo DE para la sintonización de los parámetros de una red tipo DMNN sobre el software seleccionado.
   - 2.4. Construir el algoritmo PSO para la sintonización de los parámetros de una red tipo DMNN sobre el software seleccionado.
   - 2.5. Crear una GUI para manejar intuitivamente el software.
   - 2.6. Proporcionar un mecanismo de prueba de la GUI para verificar su funcionamiento.
   - 2.7. Presentar un informe de avance interno a los tutores del trabajo.
3. Evaluar los algoritmos de aprendizaje para la red neuronal morfológica a través de la comparación de métricas de desempeño estandarizadas.
   - 3.1. Identificar qué clase de problemas son adecuados para poner a prueba la red DMNN.
   - 3.2. Buscar bases de datos que cumplan los requisitos.
   - 3.3. Seleccionar al menos tres problemas para verificar el funcionamiento de los algoritmos de aprendizaje implementados en la GUI.
   - 3.4. Elegir qué criterio estandarizado se utilizará para comparar.
   - 3.5. Evaluar las distintas bases de datos con los algoritmos.
   - 3.6. Verificar los resultados y análisis a partir de los criterios de comparación seleccionados.
   - 3.7. Presentar un informe de avance interno a los tutores del trabajo.
4. Presentar trabajo de grado escrito.

7 CRONOGRAMA

Tarea	Mes1		Mes2		Mes3		Mes4		Mes5		Mes6		Mes7		Mes8	
1.1																
1.2																
1.3																
1.4																
2.1																
2.2																
2.3																
2.4																
2.5																
2.6																
3.1																
3.2																
3.3																
3.4																
3.5																
3.6																

8 PRESUPUESTO

Descripción R.	**R. Univalle**	**R. Propios**
Director	3’400.000	
Co-Director	3’400.000	
Estudiante		2’480.000
Computador	1’000.000	2’000.000
Matlab	8’510.000	
Papelería		30.000
Internet	40.000	576.000
Subtotal	**16’350.000**	**5’086.000**
Total		**21’436.000**

9 REFERENCIAS

[1] Carlos A. Coello Coello, David A. Van Veldhuizen, Gary B. Lamont, “Evolutionary Algorithms for Solving Multi-Objective Problems”, 2002 ISBN: 978-0-306-46762-2.

[2] Abraham A, “Hybrid Artificial Intelligence Systems”, 2007, In: Corchado E., Corchado J.M., Abraham A. (eds) Innovations in Hybrid Intelligent Systems. Advances in Soft Computing, vol 44. Springer, Berlin, Heidelberg.

[3] Kevin M. Passino, “Biomimicry of Bacterial Foraging”, 2002 IEEE Control Systems Magazine.

[4] Eric Bonabeau and Guy Théraulaz, “Swarm Smarts”, 2000 Scientific American.

[5] Eric Bonabeau, Marco Dorigo and Guy Théraulaz, “Swarm Intelligence from Natural to Artificial Systems”, 1999 Oxford University Press, A Volume In The Santa Fe Institute Studies In The Sciences Of Complexity.

[6] Fernando Arce, Erick Zamora, Humberto Sossa and Ricardo Barron, “Dendrite Morphological Neural Networks Trained by Differential Evolution”, 2017, from 2016 IEEE Symposium Series on Computational Intelligence.

[7] Erik Zamora and Humberto Sossa, “Dendrite Morphological Neurons Trained by Stochastic Gradient Descent”, 2017 Neurocomputing 260, pp. 420-431.

[8] Humberto Sossa and Elizabeth Guevara, “Efficient Training for Dendrite Morphological Neural Networks”, 2014 Neurocomputing 131, pp. 132-142.

[9] Kenneth V. Price, Rainer M. Storn and Jouni A. Lampinen, “Differential Evolution a Practical Approach to Global Optimization”, 2005 Springer, Natural Computing Series, ISBN-10 3-540-20950-6.

[10] James Kennedy and Russell Eberhart, “Particle Swarm Optimization”, 1995 Purdue School of Engineering and Technology, Indianapolis, IN 46202-5160.

[11] Ioan Cristian Trelea, “The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection”, 2003 NH Elsevier, ComputerScienceWeb.

[12] Gerhard X. Ritter and Peter Sussner, “An Introduction to Morphological Neural Networks”, 1996 Proceedings – International Conference on Pattern Recognition 4,547657, pp. 709-717.

[13] Eduardo F. Caicedo Bravo y Jesus A. Lopez Sotelo, “Una Aproximación Práctica a las Redes Neuronales Artificiales”, 2013 Universidad del Valle, ISBN: 978-958-670-767-1.

[14] Joseph N. Wilson and Gerhard X. Ritter, “Handbook of Computer Vision Algorithms in Image Algebra”, 2000 by CRC Press, ISBN: 9780849300752.

[15] Vincent Morard, Etienne Decenciere and Peter Dokladal, “Mathematical Morphology and Its Applications to Image and Signal Processing”, 2011 CMM-Centre de Morphologie Mathematique, ISBN: 978-3-642-21569-8.

[16] B. Raducanu and M. Grana, “Morphological Neural Networks for Robust Visual Processing in Mobile Robotics”, 2002 Proceeding of the 2000 IEEE-INNS-ENNS International Joint Conference on.

[17] Yonggwan Won, Paul D. Gader, "Comparison of linear and morphological shared-weight neural networks", Proc. SPIE 2662, Nonlinear Image Processing VII, (25 March 1996); doi: 10.1117/12.235821.

[18] Jennifer L. Davidson and Frank Hummer, “Morphology neural networks: An introduction with applications”, 1993 Volume 12, pp 177-210.

[19] Thomas Weise, “Global Optimization Algorithms Theory and Application”, 2009, 2nd ed.

[20] Herbert Robbins and Sutton Monro, “A Stochastic Approximation Method”, 1951 volume 22 no. 3, pp 400-407.

[21] Gerhard X. Ritter and Gonzalo Urcid, “Lattice Algebra Approach to Single Neuron Computation”, 2003 IEEE Transactions On Neural Networks, vol 14 no. 2.

[22] Gerhard X. Ritter, Laurentiu Iancu and Gonzalo Urcid, “Morphological Perceptrons With Dendritic Structure”, 2003 Fuzzy Systems The 12th IEEE International Conference On.