UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS QUIMICAS E INGENIERIA

MAESTRIA Y DOCTORADO EN CIENCIAS QUIMICAS E INGENIERIA

OPTIMIZACION DE REDES NEURONALES MODULARES POR MEDIO DE METODOS DE COMPUTACION EVOLUTIVA

TRABAJO DE TESIS

Presentado por

FEVRIER ADOLFO VALDEZ ACOSTA

Para obtener el Grado de Doctor en Ciencias

DIRECTORA:

DRA. ELBA PATRICIA MELIN OLMEDA

CO-DIRECTOR:

DR. GUILLERMO LICEA SANDOVAL

Tijuana, B. C. Junio de 2009

Universidad Autónoma de Baja California

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

COORDINACIÓN DE POSGRADO E INVESTIGACIÓN

FOLIO No. 025

Tijuana, B. C., a 03 de junio de 2009

C. FEVRIER ADOLFO VALDEZ ACOSTA
Pasante de: Doctor en Ciencias
Presente

El tema de trabajo y/o tesis para su examen profesional, en la Opción TESIS

Es propuesto, por los C. Dres. Elba Patricia Melin Olmeda y Guillermo Licea Sandoval

quienes serán los responsables de la calidad del trabajo que usted presente, referido al tema OPTIMIZACION DE REDES NEURONALES MODULARES POR MEDIO DE METODOS DE COMPUTACIÓN EVOLUTIVA

el cual deberá usted desarrollar, de acuerdo con el siguiente orden:

I.- INTRODUCCIÓN
II.- MARCO TEORICO
III.- MODELO COMPLETO DEL METODO PROPUESTO FPSO+FGA
IV.- RESULTADOS DE SIMULACIÓN DEL METODO PROPUESTO FPSO+FGA
V.- CONCLUSIONES

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA

Dra. Elba Patricia Melin Olmeda

Asesor

MC. Luis Enrique Palafox Maestre

Sub-Director Secretario

Dr. Guillermo Licea Sandoval

Asesor

MC. Rubén Guillermo Sepúlveda Marques

Director

Tabla de contenido

Resumen

En esta tesis un nuevo método híbrido de computación evolutiva, fue desarrollado con el objetivo principal de aplicarlo a la optimización de funciones matemáticas complejas y arquitecturas de redes neuronales modulares. El nuevo método combina las características de dos técnicas, las cuales son, algoritmos inspirados en evolución, en este caso se trabajó, con Algoritmos Genéticos (GAs) y con Algoritmos basados en comportamientos de especies, como es la Optimización por Nube de Partículas (PSO), combinando sus características para así, construir otro nuevo método híbrido a partir de estos dos. También, se incorporó la Teoría de la Lógica Difusa. Para hacer algunas evaluaciones se utilizaron reglas IF-THEN con sistemas difusos para controlar de manera inteligente los resultados obtenidos. Dando así como resultado el nuevo método híbrido al que se le denominó FPSO+FGA. En capítulos posteriores se explica detalladamente acerca de este método.

Abstract

In this thesis a new hybrid approach for optimization combining Particle Swarm Optimization (PSO) and Genetic Algorithms (GAs) using Fuzzy Logic to integrate the results is presented. The new evolutionary method combines the advantages of PSO and GA to give us an improved FPSO+FGA hybrid method. Fuzzy Logic is used to combine the results of the PSO and GA in the best way possible. The new hybrid FPSO+FGA approach is compared with the PSO and GA methods with a set of benchmark mathematical functions. The proposed hybrid method is also tested with the problem of modular neural network optimization. The new hybrid FPSO+FGA method is shown to be superior with respect to both the individual evolutionary methods.

Agradecimientos

En este trabajo de tesis doctoral quiero agradecer, primeramente a Dios, por permitirme llegar a esta etapa de mi vida, darme salud y bienestar para lograr mis metas.

A mis padres, María Elena Acosta Cárdenas y Adolfo Valdés Valdés, por que sin su apoyo no hubiera sido posible cumplir esta meta que desde hace tiempo me había propuesto.

A todos mis hermanos, Juan, Eliezer, Alvin, Tania y Nadia por su apoyo incondicional a lo largo de mis estudios profesionales.

A mi Directora de tesis, la Dra. Elba Patricia Melín Olmeda y al Dr. Oscar Castillo López, por todo el apoyo brindado desde que llegué a la ciudad de Tijuana, porque gracias a su experiencia, consejos y conocimientos he podido llegar hasta este momento tan importante en mi vida.

A todos mis maestros de la Universidad, al Dr. Guillermo Licea Sandoval y Dr. Antonio Rodríguez Díaz, por abrirme las puertas de la Universidad y permitirme cursar los estudios de Doctorado. Al Dr. Oscar Montiel, Dr. Roberto Sepúlveda, Dr. Juan Ramón Castro, por compartir sus conocimientos conmigo y por su apoyo.

A mis compañeros y amigos de generación, con los cuales compartí muchos momentos bonitos y difíciles en este tiempo, Miguel Ángel López, Olivia Mendoza y José Luis.

A mis amigas Diana, Claudia, Liz y Gaby por ser muy buenas amigas y además por apoyarme siempre que las he necesitado y por los momentos bonitos que hemos compartido desde que llegamos a la ciudad de Tijuana. A mi amigos Álvaro Acosta y Alfonso Bravo por la gran amistad que hemos hecho a lo largo de estos años.

Y a todos mis amigos de Maestría del ITT, Fernando y Victor. Y a los de doctorado de la UABC por el apoyo brindado cuando los necesité. En especial Denisse y Ricardo por ser

compañeros de laboratorio y muy buenos amigos. A mi amiga Yared también por su amistad y por sus buenos consejos.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado para cursar los estudios de Maestría y en esta ocasión de Doctorado en Ciencias de la Computación.

A todos, sinceramente, muchas gracias

Capítulo 1

Introducción

1.1. Descripción de la investigación

En esta tesis doctoral, se desarrolló un nuevo método híbrido de computación evolutiva llamado FPSO+FGA, que es una combinación de optimización por nube de partículas (PSO), algoritmos genéticos (GAs) y lógica difusa (F). Tanto el PSO como el GA y la lógica difusa fueron utilizados conjuntamente para lograr un mejor rendimiento y eficiencia en problemas de optimización, tales como, arquitecturas de redes neuronales modulares y minimización de funciones matemáticas complejas con diferentes variables. También estas técnicas fueron utilizadas separadamente para hacer el estudio comparativo con el nuevo método propuesto.

El contenido de esta tesis se divide en varios capítulos; tales como. un marco teórico que contiene información relevante de las disciplinas de computación inteligente que se emplearon en el desarrollo de esta investigación, como son los algoritmos genéticos (GA), optimización por nube de partículas (PSO) y lógica difusa (F); en capítulos posteriores se analiza un modelo completo del método propuesto FPSO+FGA, algunos casos de estudio de los resultados de simulación para la optimización de redes neuronales modulares en el reconocimiento de patrones y funciones matemáticas son presentados, así como algunas conclusiones, bibliografía y anexos de este trabajo de investigación.

Cuando se utiliza un método de computación evolutiva (GA) o un método bioinspirado (PSO) para optimización de problemas, lo que se busca es tener el error más bajo, para obtener la mejor solución del problema, en este caso, se han hecho varias pruebas de simulación para validar el enfoque del método propuesto, se trabajó con una base de datos de rostros llamada Yale para hacer la optimización de la arquitectura de la red neuronal modular, se experimentó con 7 funciones matemáticas complejas con diferentes variables para validar y comprobar el funcionamiento de esta nueva metodología de optimización.

Se utilizó el lenguaje Matlab 7.5 para programar la estructura de un Algoritmo Genético (GA), de una nube de partículas (PSO) y así poder desarrollar el nuevo método híbrido evolutivo FPSO+FGA.

En los resultados de simulación, se comparan gráficamente los resultados con GA, PSO y FPSO+FGA.

En esta tesis se encuentran algunas definiciones y un panorama de implementación de las técnicas que fueron empleadas para lograr el objetivo de la investigación.

1.2. Objetivo General de la investigación

El objetivo general de este proyecto de tesis fue el de investigar y desarrollar un nuevo Método de Computación Evolutiva Híbrido para resolver problemas de Optimización de Redes Neuronales Modulares y funciones matemáticas complejas.

1.3. Objetivos Particulares de la Investigación

- Desarrollar un nuevo Método de Computación Evolutiva combinando varios tipos de algoritmos evolutivos.
- Emplear el método desarrollado para la Optimización de Redes Neuronales Modulares en Aplicaciones de Reconocimiento de Patrones.
- Comparar el método propuesto con los existentes en la literatura empleando métodos estadísticos.
- Evaluar el método propuesto con el fin de ver los resultados de cada uno.

1.4. Aportaciones del Trabajo de Investigación

A continuación se enlistan las aportaciones más importantes que el nuevo método híbrido de computación evolutiva posee.

- Se introduce un nuevo método de computación evolutiva para resolver problemas de optimización (FPSO+FGA)

- Se desarrolló una nueva forma de trabajar con diversas estrategias, unas basadas en evolución y otras bioinspiradas, combinando sus características más importantes para lograr obtener mejores resultados.
- Se propone la adaptación de parámetros importantes en el momento de evolución del método para hacerlo así más inteligente que como se hace tradicionalmente.
- Se introduce la lógica difusa para combinar las características de las dos estrategias utilizadas.
- Se proponen algunos operadores genéticos difusos para controlar de manera óptima la evolución de la población.
- La aplicabilidad del método puede expandirse en diversas áreas de optimización.

Hasta el momento se han hecho varias aportaciones en el área de computación evolutiva y algoritmos bioinspirados aplicada a problemas de optimización, en nuestro caso particular, se han realizado algunas investigaciones aplicadas a optimización de funciones matemáticas y obtención de arquitecturas de redes neuronales. Todos estos estudios han sido publicados y presentados en congresos y revistas internacionales. La tabla 1.1 muestra la lista de trabajos publicados referentes a esta investigación.

Tabla 1.1. Lista de publicaciones referentes a esta investigación.

Nombre del articulo / libro / capítulo / patentes	Título de la revista o editorial	Fecha mes-año
LIBRO: Hybrid Intelligent Systems **CAPITULO:** Evolutionary Computing for Topology Optimization of Type-2 Fuzzy Controllers	Springer-Verlag	2007
LIBRO: Analysis and Design of Intelligent Systems Using Soft Computing Techniques (Advances in Soft Computing). **CAPITULO:** Evolutionary Computing for the Optimization of Mathematical Functions	Springer-Verlag CANCUN	2007
LIBRO: Soft Computing for Hybrid Intelligent Systems. **CAPITULO:** A New Evolutionary Method Combining Particle Swarm Optimization and Genetic Algorithms Using Fuzzy Logic	Springer-Verlag	2008
ARTICULO: "A New Evolutionary Method with a Hybrid Approach Combining Particle Swarm Optimization and Genetic Algorithms Using Fuzzy Logic for Decision Making"	2008 IEEE World Congress on Computational Intelligence (WCCI 2008) HONG KONG, CHINA	2008
ARTICULO: "A New Evolutionary Method with Fuzzy Logic for Combining Particle Swarm Optimization and Genetic Algorithms: The Case of Neural Networks Optimization"	2008 IEEE World Congress on Computational Intelligence (WCCI 2008) HONG KONG, CHINA	2008
ARTICULO: "Neural Network Optimization with a Hybrid Evolutionary Method that combines Particle Swarm and Genetic Algorithms with Fuzzy Rules"	North American Fuzzy Information Processing Society. NAFIPS 2008 Nueva York,. USA	2008
ARTICULO EN JOURNAL: "A New Fuzzy Evolutionary Method Combining Particle Swarm Optimization and Genetic Algorithms using Fuzzy If Then Rules for Dynamic Parameter Adaptation and Aggregation of Results "	JHCR Journal of Hybrid Computing Research, Vol. X, No. X, January-June 200X© Serials Publications	ACEPTADO 2009
ARTICULO: "Fuzzy Logic Adaptation of a Hybrid Evolutionary Method for Pattern Recognition"	IFSA2009/EUSFLAT09 Lisboa, Portugal	ACEPTADO JUL-09
ARTICULO EN JOURNAL: "Comparative Study of Particle Swarm Optimization and Genetic Algoritms for Mathematicals Complex Functions"	Journal of Automation, Mobile Robotics and Intelligent Systems. JAMRIS 2008	ENE-08
ARTICULO EN JOURNAL: "Hierarchical genetic algorithms for topology optimization in fuzzy control systems"	International Journal of General Systems	OCT-07
ARTICULO: "Parallel Evolutionary Computing using a cluster for Mathematical Function Optimization"	North American Fuzzy Information Processing Society. NAFIPS 2007 SAN DIEGO, CA. USA	JUN-07

Capítulo 2

Marco Teórico

En esta sección, se explica una reseña histórica acerca de las técnicas de computación inteligente que fueron empleadas a lo largo de esta investigación.

2.1. Optimización por Nube de Partículas (PSO)

A continuación se describen los conceptos básicos acerca de la optimización por nube de partículas (PSO), así como sus orígenes, estructura, codificación y posibles aplicaciones de esta metodología. Investigaciones con estas técnicas pueden ser vistas en [40,42, 44,45]

Un Algoritmo basado en nube de partículas es una técnica meta heurística [1, 2, 9] basada en poblaciones e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces [43,46,47]. PSO fue originalmente desarrollado por el psicólogo sociólogo James Kennedy y por el ingeniero electrónico Russell Eberhart en 1995, basándose en un enfoque conocido como la metáfora social [20,48,50], que describe a este algoritmo y que se puede resumir de la siguiente forma: los individuos que conviven en una sociedad tienen una opinión que es parte de un conjunto de creencias (el espacio de búsqueda)

compartido por todos los posibles individuos [33,35]. Cada individuo puede modificar su propia opinión basándose en tres factores:

- Su conocimiento sobre el entorno (su valor de fitness).
- Su conocimiento histórico o experiencias anteriores (su memoria).
- El conocimiento histórico o experiencias anteriores de los individuos situados en su vecindario.

Siguiendo ciertas reglas de interacción, los individuos en la población adaptan sus esquemas de creencias al de los individuos con más éxito de su entorno. Con el tiempo, surge una cultura cuyos individuos tienen un conjunto de creencias estrechamente relacionado. El principio natural en el que se basa PSO es el comportamiento de una bandada de aves o de un banco de peces como se muestra en la figura 2.1, supongamos que una de estas bandadas busca comida en un área y que solamente hay una pieza de comida en dicha área [13,52]. Los pájaros no saben dónde está la comida pero sí conocen su distancia a la misma, por lo que la estrategia más eficaz para hallar la comida es seguir al ave que se encuentre más cerca de ella. PSO emula este escenario para resolver problemas de optimización. Cada solución (partícula) es un ave en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere. La nube de partículas (swarm) es un sistema multiagente, es decir, las partículas son agentes simples que se mueven por el espacio de búsqueda y que guardan (y posiblemente comunican) la mejor solución que han encontrado [37,38]. Cada partícula tiene una aptitud, una posición y un vector velocidad que dirige el movimiento. El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual. En la literatura lo podemos encontrar como población, nube, enjambre o colmena (swarm) de partículas. En esta tesis usaremos el término nube.

Figura 2.1 Ejemplo de swarm en la naturaleza

Los algoritmos basados en nube de partículas se han aplicado con éxito en diferentes campos de investigación. Algunos ejemplos son: optimización de funciones matemáticas [58], entrenamiento de redes neuronales [11], aprendizaje de sistemas difusos [36], procesamiento de imágenes [32], problema del agente viajero [49] e ingeniería química [34].

2.1.1. Descripción del Algoritmo PSO

Un algoritmo PSO consiste en un proceso iterativo y estocástico que opera sobre una nube de partículas. La posición de cada partícula representa una solución potencial al problema que se está resolviendo. Generalmente, una partícula p_i está compuesta de tres vectores y dos valores de aptitud:

- El vector $x_i = (x_{i1}, x_{i2}, ..., x_{in})$ almacena la posición actual (localización) de la partícula en el espacio de búsqueda.
- El vector $pBest_i = (p_{i1}, p_{i2}, ..., p_{in})$ almacena la posición de la mejor solución encontrada por la partícula hasta el momento.
- El vector de velocidad $v_i = (v_{i1}, v_{i2}, ..., v_{in})$ almacena el gradiente (dirección) según el cual se moverá la partícula.
- El valor de aptitud $fitness\ x_i$ almacena el valor de adecuación de la solución actual

(vector x_i).

- El valor de aptitud $fitness\ pBest_i$ almacena el valor de adecuación de la mejor solución local encontrada hasta el momento (vector $pBest_i$).

La nube se inicializa generando las posiciones y las velocidades iniciales de las partículas. Las posiciones se pueden generar aleatoriamente en el espacio de búsqueda (quizás con ayuda de un heurístico de construcción), de forma regular o con una combinación de ambas formas. Una vez generadas las posiciones, se calcula la aptitud de cada una y se actualizan los valores de $fitness\ x_i$ y $fitness\ pBest_i$.

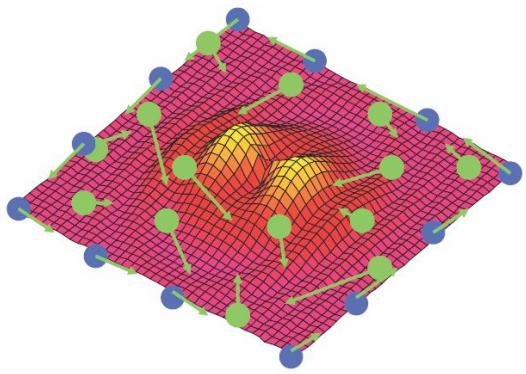


Fig 2.2 Inicialización de la nube en el espacio de búsqueda

Las velocidades se generan aleatoriamente, con cada componente en el intervalo $[-v_{max}, v_{max}]$, donde v_{max} será la velocidad máxima que pueda tomar una partícula en cada movimiento. No es conveniente inicializarlas a cero pues no se obtienen buenos resultados [22].

Inicializado la nube (Figura 2.3), las partículas se deben mover dentro del proceso iterativo. Una partícula se mueve desde una posición del espacio de búsqueda hasta otra,

simplemente, añadiendo al vector de posición x_i el vector velocidad v_i para obtener un nuevo vector posición:

$$x_i \leftarrow x_i + v_i \tag{2.1}$$

Una vez calculada la nueva posición de la partícula, se evalúa actualizando $fitness\ x_i$. Además, si el nuevo valor de aptitud es el mejor encontrado hasta el momento, se actualizan los valores de mejor posición $pBest_i$ y fitness $fitness\_pBest_i$. El vector velocidad de cada partícula es modificado en cada iteración utilizando la velocidad anterior, un componente cognitivo y un componente social. El modelo matemático resultante y que representa el corazón del algoritmo PSO viene representado por las siguientes ecuaciones:

$$v_i^{k+1} = \omega \cdot v_i^k + \varphi_1 \cdot rand_1 \cdot (pBest_i - x_i^k) + \varphi_2 \cdot rand_2 \cdot (g_i - x_i^k) \tag{2.2}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{2.3}$$

La Ecuación 2.3 refleja la actualización del vector velocidad de cada partícula i en cada iteración k. El componente cognitivo está modelado por el factor $\varphi_1 \cdot rand_1 \cdot (pBest_i - x^k$) y representa la distancia entre la posición actual y la mejor conocida por esa partícula, es decir, la decisión que tomará la partícula influenciada por su propia experiencia a lo largo de su vida. El componente social está modelado por $\varphi_2 \cdot rand_2 \cdot (g_i - x^k)$ y representa la distancia entre la posición actual y la mejor posición del vecindario, es decir, la decisión que tomará la partícula según la influencia que el resto de la nube ejerce sobre ella. Una descripción más detallada de cada factor se realiza a continuación:

v^k : velocidad de la partícula i en la iteración k,

w : factor inercia

φ_1, φ_2 : son factores de aprendizaje (pesos) que controlan los componentes de las aceleraciones cognitivo y social,

$rand_1, rand_2$: números aleatorios entre 0 y 1,

x^k : posición actual de la partícula i en la iteración k,

$pBest_i$: mejor posición (solución) encontrada por la partícula i hasta el momento.

g_i : representa la posición de la partícula con el mejor $pBest\ fitness$ del entorno de p_i ($lBest$ o $localbest$) o de toda la nube ($gBest$ o $globalbest$).

La Ecuación 2.3 modela el movimiento de cada partícula i en cada iteración k. En la Figura 1.5 se muestra una representación gráfica del movimiento de una partícula en el espacio de soluciones.

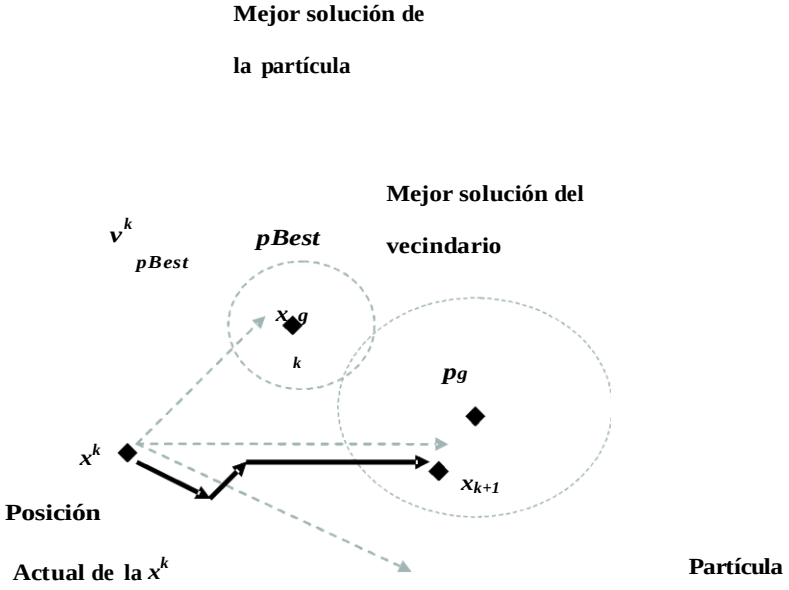


Fig. 2.3 Movimiento de una partícula en el espacio de soluciones

2.2. Algoritmos Genéticos (GA)

En esta sección se describen los conceptos básicos acerca de la computación evolutiva o algoritmos genéticos, así como sus orígenes, estructura, codificación y posibles aplicaciones de los mismos, cabe señalar que este capítulo es muy importante para el lector, ya que en este comprenderá la utilidad de la computación evolutiva en la resolución de problemas de búsqueda y optimización, en capítulos posteriores se explicarán la aplicación de estas técnicas a problemas de optimización de funciones matemáticas y arquitecturas de redes neuronales modulares.

Los algoritmos genéticos fueron introducidos por primera vez por un investigador de la Universidad de Michigan llamado John Holland [16]. Un algoritmo genético, es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual [10,15]. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.

El Algoritmo Genético (GA) [25] supone que la posible solución a un problema puede ser visto como un individuo y representado por un grupo de parámetros. Estos parámetros son los genes de un cromosoma, que representan la estructura del individuo. Este cromosoma es evaluado mediante una función de aptitud, dando un valor de aptitud a cada individuo, referente a la aptitud que tiene para resolver el problema dado. A través de una evolución genética, mediante operaciones de cruce (apareamiento de individuos para

producir hijos) y mutación que simulan el comportamiento biológico, combinado con un proceso de selección, la población de individuos va eliminando a aquellos con menor aptitud, y tiende a mejorar la aptitud global de la población, para producir una solución cercana a lo más óptimo para el problema dado.

John Holland estaba consciente de la importancia de la selección natural [15], y a fines de los 60s desarrolló una técnica que permitió incorporarla en un programa de computadora. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que propuso Holland se le llamó originalmente "planes reproductivos".

2.2.1. Introducción a los Algoritmos Genéticos

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad alrededor del mundo durante los últimos años. Se presentarán en esta tesis los conceptos básicos que se requieren para abordarla, así como un sencillo ejemplo que permita a los lectores comprender cómo aplicarla al problema de su elección. Adicionalmente, se hablará acerca de los diversos ambientes de programación actuales basados en algoritmos genéticos y de las áreas abiertas de investigación. Trabajos realizados con algoritmos genéticos pueden ser vistos en [28, 27,24]

En los últimos años, la comunidad científica internacional ha mostrado un creciente interés en una nueva técnica de búsqueda basada en la teoría de la evolución y que se conoce como el algoritmo genético. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza [4], de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los genes

(unidad básica de codificación de cada uno de los atributos de un ser vivo) de un individuo, y que los atributos más deseables (por ejemplo, los que le permiten a un individuo adaptarse mejor a su entorno) del mismo se transmiten a sus descendientes, cuando éste se reproduce sexualmente [3].

Imitando la mecánica de la evolución biológica en la naturaleza, los algoritmos genéticos operan sobre una población compuesta de posibles soluciones al problema. Cada elemento de la población se denomina "cromosoma". Un cromosoma es el representante, dentro del algoritmo genético, de una posible solución al problema. La forma en que los cromosomas codifican a la solución se denomina "Representación" (ver figura 2.4).

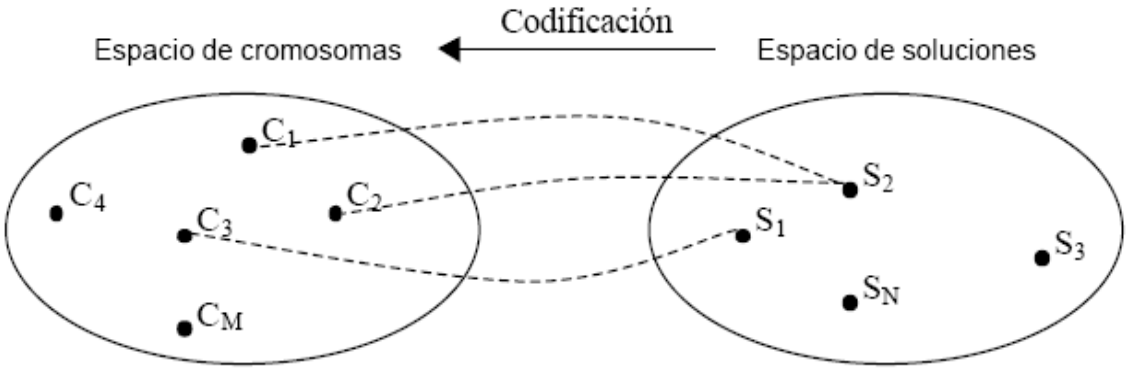


Fig 2.4 Cada cromosoma codifica una posible solución al problema

El algoritmo genético va creando nuevas "generaciones" de esta población, cuyos individuos son cada vez mejores soluciones al problema. La creación de una nueva generación de individuos se produce aplicando a la generación anterior operadores genéticos, adaptados de la genética natural. La figura 2.5 representa el esquema de funcionamiento del algoritmo genético. El proceso comienza seleccionando un número de cromosomas para que conformen la población inicial. A continuación se evalúa la función de adaptación para estos individuos. La función de aptitud da una medida de la aptitud del

cromosoma para sobrevivir en su entorno. Debe estar definida de tal forma que los cromosomas que representen mejores soluciones tengan valores más altos de aptitud. Los individuos más aptos se seleccionan en parejas para reproducirse. La reproducción genera nuevos cromosomas que combinan características de ambos padres. Estos nuevos cromosomas reemplazan a los individuos con menores valores de adaptación. A continuación, algunos cromosomas son seleccionados al azar para ser mutados. La mutación consiste en aplicar un cambio aleatorio en su estructura. Luego, los nuevos cromosomas deben incorporarse a la población; estos cromosomas deben reemplazar a cromosomas ya existentes. Existen diferentes criterios que pueden utilizarse para elegir a los cromosomas que serán reemplazados. El ciclo de selección, reproducción y mutación se repite hasta que se cumple el criterio de terminación del algoritmo, momento en el cual el cromosoma mejor adaptado se devuelve como solución (ver figura 2.5).

Fig. 2.5. Esquema de funcionamiento del algoritmo genético

2.2.2. Algoritmo Genético Simple

En ésta sección se describen los componentes básicos de un Algoritmo Genético Simple, es importante para el lector comprender estos componentes, ya que en capítulos posteriores se hace mención de los mismos.

2.2.3 Cromosoma

Representan individuos que son llevados a lo largo de varias generaciones, en forma similar a las poblaciones naturales, evolucionando en forma similar a la teoría de C. Darwin en su libro " Origen de las especies". Los cromosomas de cualquier especie son pares. Así los cromosomas de la especie humana que son 46, son 23 parejas de cromosomas homólogos. El Algoritmo Genético Simple [17], se representa en la Figura 2.6. Como se verá a continuación, se necesita una codificación o representación del problema, que resulte adecuada al mismo. Además se requiere una función aptitud ó adaptación al problema, la cual asigna un número real a cada posible solución codificada. Durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción, a continuación dichos padres seleccionados se cruzarán generando dos hijos, sobre cada uno de los cuales actuará un operador de mutación. El resultado de la combinación de las anteriores funciones será un conjunto de individuos (posibles soluciones al problema), los cuales en la evolución del Algoritmo Genético formarán parte de la siguiente

población.

```
BEGIN /* Algoritmo genético simple */
        Generar una población inicial.
        Computar la función de evaluación de cada individuo.
        WHILE NOT Terminado DO
        BEGIN /* Producir nueva generación */
                FOR Tamaño población /2 DO
                BEGÍN /* Ciclo Reproductivo */
                        Seleccionar dos individuos de la generación anterior, para el cruce.
                        Cruzar con cierta probabilidad los dos individuos obteniendo dos
                        descendientes.
                        Mutar los dos descendientes con cierta probabilidad.
                        Computar la función de evaluación de los dos descendientes mutados.
                        Insertar los dos descendientes mutados en la nueva generación.
                END
                IF      la población ha convergido THEN
                        Terminado: = TRUE
        END
```

Fig 2.6 Pseudocódigo del Algoritmo Genético Simple

2.2.4 Codificación

Se supone que los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros (que denominaremos genes), los cuales agrupados forman un conjunto de valores (a menudo llamados el cromosoma). Si bien el alfabeto utilizado para representar los individuos no debe necesariamente estar constituido por el {0, l), buena parte de la teoría en la que se fundamentan los Algoritmos Genéticos utiliza dicho alfabeto. En términos biológicos, el conjunto de parámetros representando un cromosoma particular se denomina fenotipo. El fenotipo contiene la información requerida para construir un organismo, el cual se refiere como genotipo. Los mismos términos se utilizan en el campo de los Algoritmos Genéticos. La adaptación al problema de un individuo depende de la evaluación del genotipo. Esta última puede inferirse a partir del fenotipo, es decir puede ser computada a partir del cromosoma, usando la función de evaluación. La función de aptitud debe ser diseñada para cada problema de manera específica. Dado un cromosoma particular, la función aptitud le asigna un número real, que

se supone refleja el nivel de aptitud al problema del individuo representado por el cromosoma.

Durante la fase reproductiva se seleccionan los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos. La selección de padres se efectúa al azar usando un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función aptitud. Este procedimiento se dice que está basado en la ruleta rusa. Según dicho esquema, los individuos que se adapten se escogerán probablemente varias veces por generación, mientras que, los que menos se adapten al problema, no se escogerán más que de vez en cuando.

Una vez seleccionados dos padres, sus cromosomas se combinan, utilizando habitualmente los operadores de cruce y mutación. Las formas básicas de dichos operadores se describen a continuación.

El ***operador de cruce***, toma dos padres seleccionados y combina los cromosomas en una posición escogida al azar, para producir dos cromosomas iniciales y dos cromosomas finales. Después se intercambian los cromosomas finales, produciéndose dos nuevos cromosomas completos (ver figura 2.7). Ambos descendientes heredan genes de cada uno de los padres. Este operador se conoce como operador de cruce basado en un punto. Habitualmente el operador de cruce no se aplica a todos los pares de individuos que han sido seleccionados para emparejarse, sino que se aplica de manera aleatoria, normalmente con una probabilidad comprendida entre 0.5 y 1.0. En el caso en que el

operador de cruce no se aplique, la descendencia se obtiene simplemente duplicando los padres.

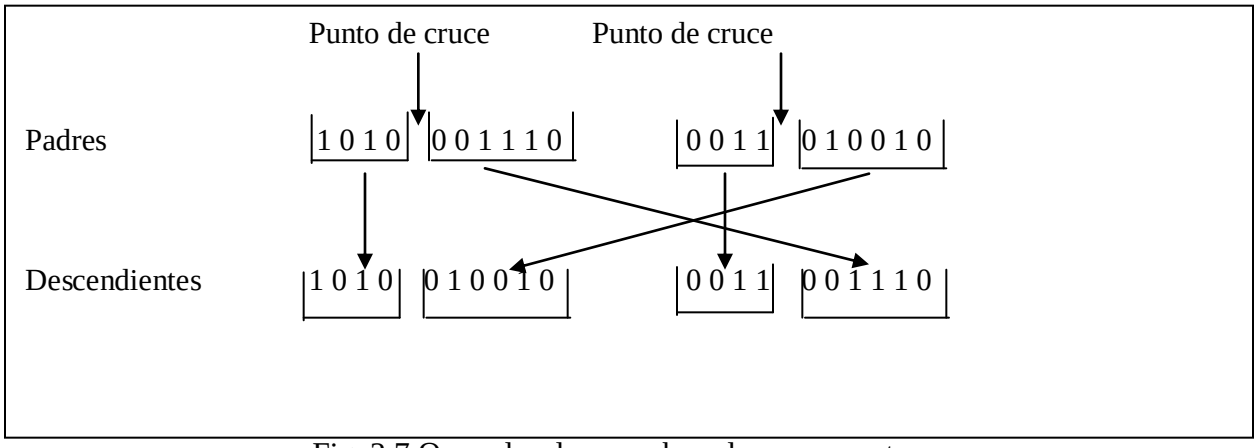


Fig. 2.7 Operador de cruce basado en un punto.

El operador de ***mutación*** se aplica a cada hijo de manera individual, y consiste en la alteración aleatoria (normalmente con probabilidad pequeña, por lo general menor que 0.10) de cada gen componente del cromosoma. La Figura 2.8 muestra la mutación del quinto gen del cromosoma. Si bien puede en principio pensarse que el operador de cruce es más importante que el operador de mutación, ya que proporciona una exploración rápida del espacio de búsqueda, éste último asegura que ningún punto del espacio de búsqueda tenga probabilidad cero de ser examinado, y esto es de gran importancia para asegurar la convergencia de los Algoritmos Genéticos.

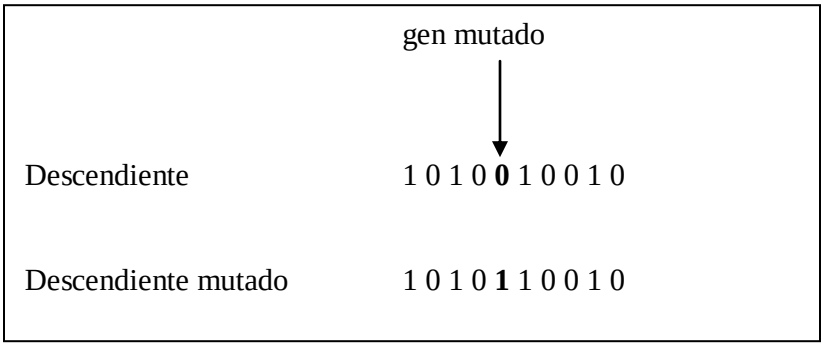


Fig. 2.8 Operador de mutación en el quinto gen.

Para criterios prácticos [8], es muy útil la definición de convergencia introducida en este campo por De Jong [5] en su tesis doctoral. Si el Algoritmo Genético ha sido correctamente implementado, la población evolucionará a lo largo de las generaciones sucesivas de tal manera que la adaptación media extendida a todos los individuos de la población, así como la adaptación del mejor individuo se irán incrementando hacia el óptimo global. El concepto de convergencia está relacionado con la progresión hacia la uniformidad: un gen ha convergido cuando al menos el 95 % de los individuos de la población comparten el mismo valor para dicho gen.

A medida que el número de generaciones aumenta, es más probable que la aptitud media se aproxime a la del mejor individuo.

2.3. Lógica Difusa

En este capítulo se describen algunos conceptos básicos de la teoría de la lógica difusa, orígenes, conjuntos difusos y algunas aplicaciones de ésta áreas necesarias para entender los resultados de esta investigación, para lo cual el lector deberá comprender muy bien el contenido de este capítulo para entender los posteriores. Se aborda también como fue implementada en esta investigación la lógica difusa en conjunto con las PSO y GA y el método propuesto FPSO+FGA, para la resolución de problemas de optimización de redes neuronales modulares y funciones matemáticas.

2.3.1. Orígenes de la Lógica Difusa

La *lógica difusa [31]* ha cobrado una gran fama por la variedad de sus aplicaciones, las cuales van desde el control de complejos procesos industriales, hasta el diseño de

dispositivos artificiales de deducción automática, pasando por la construcción de artefactos electrónicos de uso doméstico y de entretenimiento, así como también de sistemas de diagnóstico [26]. De hecho, desde hace ya, al menos, década y media, la expedición de patentes industriales de mecanismos basados en la lógica difusa tiene un crecimiento sumamente rápido en todas las naciones industrializadas. Se ha considerado de manera general que el concepto de lógica difusa apareció en 1965, en la Universidad de California en Berkeley, introducido por Lotfi A. Zadeh [60]. Las lógicas difusas, pues de hecho hay que hablar de ellas en plural [60,6], son esencialmente lógicas multivaluadas que extienden a las lógicas clásicas [7]. Estas últimas imponen a sus enunciados únicamente valores *falso* o *verdadero*. Por ejemplo, al calificar que ``el cielo es azul" uno está tentado a graduar qué tan ``azul", en efecto, es el cielo, e igualmente, si ``un vehículo se mueve rápido", también se está obligado a considerar qué tan rápido es el vehículo, aunque esto último no implique necesariamente cuantificar la velocidad del vehículo con toda precisión. Las lógicas difusas procuran crear aproximaciones matemáticas en la resolución de ciertos tipos de problemas [6,12]. Pretenden producir resultados exactos a partir de datos imprecisos, por lo cual son particularmente útiles en aplicaciones electrónicas o computacionales. El adjetivo ``difuso" aplicado a ellas se debe a que los valores de verdad no-deterministas utilizados en ellas tienen, por lo general, una connotación de incertidumbre. Un vaso medio lleno, independientemente de que también esté medio vacío, no está lleno completamente ni está vacío completamente. Qué tan lleno puede estar es un elemento de incertidumbre, es decir, de difusidad, entendida esta última como una propiedad del indeterminismo. Ahora bien, los valores de verdad asumidos por enunciados aunque no son deterministas, no necesariamente son desconocidos. Por otra parte, desde un punto de vista optimista, lo difuso puede entenderse como la posibilidad de asignar más valores de verdad a los enunciados que los clásicos ``falso" o ``verdadero". Así pues, reiteramos, las lógicas difusas

son tipos especiales de lógicas multivaluadas [20]. Las lógicas difusas han tenido aplicaciones de suma relevancia en el procesamiento electrónico de datos. En determinadas áreas de conocimiento, a sus enunciados se les asocia valores de verdad que son grados de veracidad o falsedad, mucho más amplios que los meros ``verdadero" y ``falso". En un sistema deductivo se distinguen enunciados ``de entrada" y enunciados ``de salida". El objetivo de todo sistema manejador de una lógica difusa es describir los grados de los enunciados de salida en términos de los de entrada. Más aún, algunos sistemas son capaces de refinar los grados de veracidad de los enunciados de salida conforme se refinan los de entrada. Por estas propiedades es que ciertos sistemas de lógica difusa simulan una labor de aprendizaje, y son excelentes mecanismos de control de procesos. Desde el punto de vista tecnológico, las lógicas difusas se encuadran en el área de la llamada Inteligencia Artificial y han dado origen a sistemas expertos de tipo difuso y a sistemas de control automático. Introduciremos primero la noción de *conjunto difuso*, y las operaciones usuales en ese tipo de conjuntos. Inmediatamente después, presentaremos ciertos tipos de cálculos proposicionales de tipo difuso y de cuantificación difusa.

2.3.2. Conjuntos difusos

Los conjuntos en la teoría clásica son altamente restrictivos en el sentido de que un elemento o pertenece o no pertenece a un conjunto dado. Siendo esta característica deseable en muchos aspectos científicos, pero que en un gran número de ámbitos supone una gran merma de flexibilidad [21,59]. Por ejemplo, supongamos un conjunto universal **U** formado por personas a las cuales se va a medir, asimismo se definen tres subconjuntos **Bajo, Medio** y **Alto** a los que pertenecerán los elementos de **U** según su altura.

Bajo el punto de vista de los conjuntos tradicionales los individuos que midan menos de 165 cm serán bajos, los que midan entre 165 cm y 185 cm tendrán una estatura media y los que superen los 185 cm serán altos. Este sistema, que en principio resuelve la tarea de clasificación, genera un problema añadido, esto es: una persona que mida 184.75 cm ¿es alta, o tiene una estatura media?. El problema en cuestión se forma a partir del hecho de que para el modo de razonamiento humano los conjuntos **Bajo**, **Medio** y **Alto** no tienen unos límites claros, sino que estos son "difusos", por lo que los elementos pierden grado de membresía a medida que se alejan de ellos.

Para abordar estas cuestiones en 1965 Lofti A. Zadeh formalizó matemáticamente el concepto de los conjuntos difusos [59,61], en los cuales los límites están definidos de forma imprecisa y su función característica no afirma ni niega la membresía, sino que la información que aporta es el grado con el que el elemento pertenece al conjunto o el grado de compatibilidad del elemento con la propiedad que caracteriza al conjunto.

La forma analítica de funciones de membresía de los conjuntos tradicionales y los difusos de la siguiente forma:

Conjunto tradicional:

$$\chi_A - = \begin{cases} 1 & \chi \in A \\ 0 & \chi \notin A \end{cases} \tag{2.4}$$

Conjunto difuso:

$$\chi_{A:}\, U \longrightarrow [0,1] \tag{2.5}$$

Como se puede observar, mientras que para los conjuntos tradicionales su función característica sólo tiene como opción los valores 0 y 1, en los conjuntos difusos ésta es

multivaluada, pudiendo tomar cualquier valor en el intervalo real [0, 1] (realmente el rango de valores que ésta puede tomar es cualquiera, aunque se suele utilizar este intervalo). Así, si el resultado de evaluar la función es 0, indicará la negación de membresía o de compatibilidad con la propiedad que caracteriza el conjunto. Siendo 1 si la compatibilidad es total o pertenece indudablemente al conjunto.

Se mostrará de forma gráfica (ver figura 2.9) cómo serían las funciones características para el ejemplo de las alturas, tomando ahora los subconjuntos como difusos.

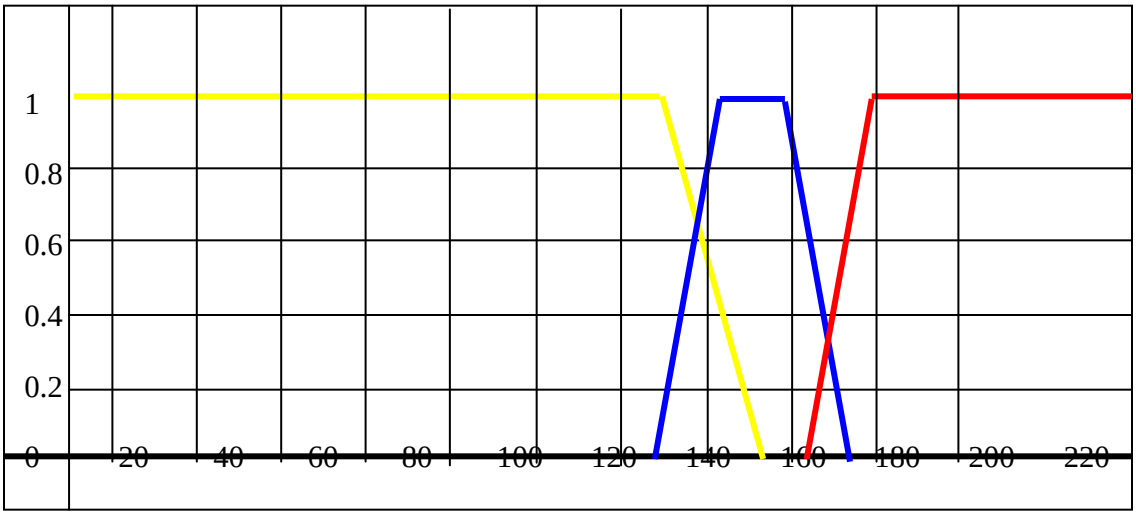


Fig. 2.9 Con color amarillo se representa el conjunto de los individuos bajos, con color azul los medios y con color rojo los altos.

Capítulo 3

Modelo Completo del Método propuesto FPSO+FGA

En este capítulo se describe un panorama completo del nuevo método híbrido de computación evolutiva al cual se le llamó FPSO+FGA que se propuso para esta investigación. En este capítulo, el lector comprenderá el porqué se decidió unir estas dos metodologías para la optimización de funciones matemáticas complejas y redes neuronales modulares. También, se describe el modelo matemático de este método y una herramienta de interfaz gráfica para hacer experimentos con el método.

3.1. Método FPSO+FGA

Este método combina las características de un PSO y un GA utilizando varios sistemas difusos para integrar resultados y colaborar a la adaptación de los parámetros en PSO y GA. En este capítulo se explica a detalle el funcionamiento de este método evolutivo, se decidió llamarle FPSO+FGA porque representamos la unión de un 'PSO' y un 'GA', se antepone una 'F' porque es la encargada de representar la parte difusa que lleva

este método, se decidió poner esta letra porque en inglés difuso se escribe ‘fuzzy’. La idea general de el método propuesto FPSO+FGA se puede observar gráficamente en la figura 3.1. El método puede es descrito a continuación.

1. Un problema de optimización (función matemática, red neuronal) es recibido para intentar optimizarlo.
2. FPSO+FGA evalúa, la función o arquitectura de red intentando minimizar los errores para lograr el mejor desempeño del sistema.
3. Un sistema difuso al cual se le llamó ‘fuzzymain’ es responsable de recibir los valores resultantes del paso 2.
4. El sistema difuso ‘fuzzymain’ decide cual camino escoger dependiendo de los errores resultantes si en un instante de tiempo ‘t’ es viable seguir optimizando con FPSO o un en instante de tiempo ‘t+1’ intentar optimizar con FGA.
5. Existen otros dos sistemas difusos encargados de estar adaptando los parámetros de evolución en GA llamado ‘fuzzyga’ y en PSO llamado ‘fuzzypso’. Esta adaptación se hace cuando el sistema difuso decide en base a los errores obtenidos (error y cambio de error se están considerando para hacer la evaluación), modificar los parámetros; en este caso, se consideraron el cruce, mutación, aceleración social y cognitiva. Se han considerado estos 4 parámetros por ser los más relevantes en la convergencia del método. También, en este método es fácil agregar más parámetros para que sean adaptados dinámicamente.
6. Repetir todos los pasos arriba mencionados hasta que el criterio de parada sea alcanzado, en este caso, puede ser por generaciones o hasta llegar al error meta.

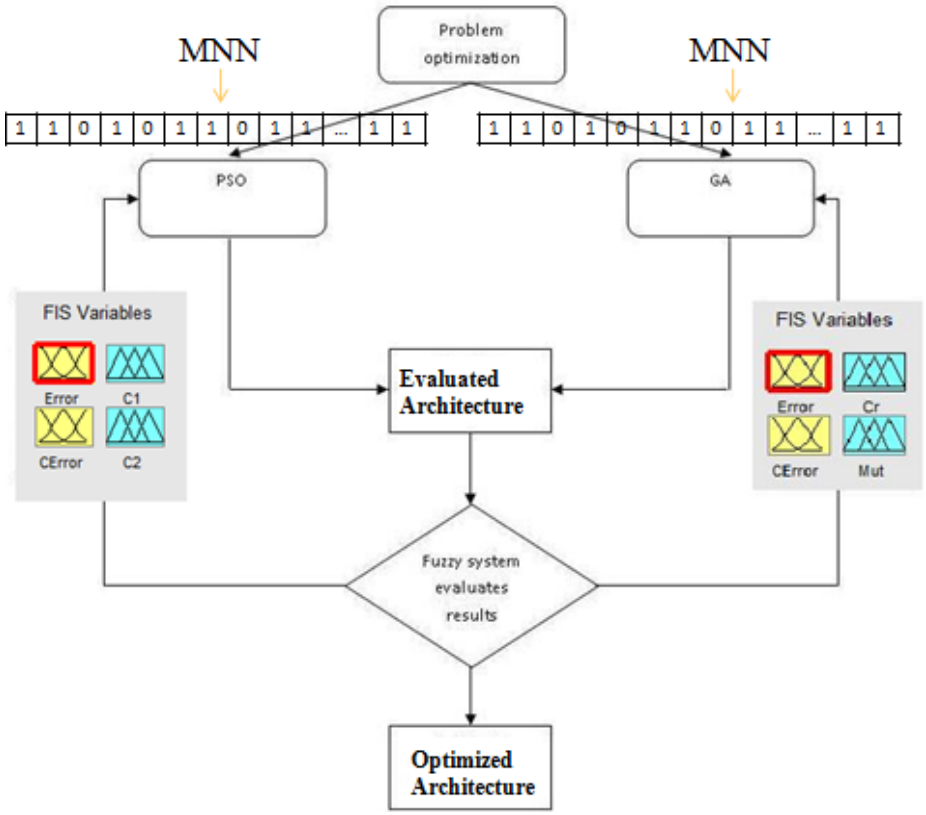


Fig. 3.1 El esquema del método FPSO+FGA

3.2. Descripción del modelo completo del método híbrido FPSO+FGA

La idea básica del método propuesto FPSO+FGA es combinar las ventajas de los dos métodos individuales, utilizando un sistema difuso para tomar decisiones y otros dos sistemas difusos para mejorar y adaptar los parámetros del FGA y el FPSO cuando sea conveniente. Como se puede observar en el método híbrido propuesto FPSO+FGA, este consiste de una estructura interna de un sistema difuso principal, cuyo objetivo es estar recibiendo dos valores como entrada, los cuales son el error y el cambio del error que se están generando en las salidas de FPSO y FGA. El sistema difuso principal llamado 'fuzzymain' es el encargado de integrar y decidir cuáles son los mejores

resultados de optimización que se están obteniendo cuando el método FPSO+FGA esta ejecutándose. De igual forma, también es responsable de estar seleccionando el problema y enviándolo al sistema difuso 'fuzzypso' cuando FGA es activado para intentar resolver el problema, o al sistema difuso 'fuzzyga' cuando FGA es invocado para resolver el problema. También, puede estar activando o desactivando uno o el otro método dependiendo de las condiciones en las que se encuentre el problema. La figura 3.2 muestra las funciones de membresía triangulares del sistema difuso principal 'fuzzymain' que fueron implementadas. En los sistemas difusos 'fuzzypso' y 'fuzzyga' que se encargan de la adaptación de parámetros, las funciones de membresía fueron variables, en algunos casos se hicieron pruebas con gausianas y en otros con triangulares, en el capítulo de resultados de simulación, se muestran los resultados obtenidos al con las diferentes funciones de membresía. Todos los sistemas difusos fueron de tipo mamdani con defusificación tipo centroide, porque en investigaciones anteriores [54, 55, 57], con este tipo de sistemas difusos se obtuvieron buenos resultados, de cualquier forma el método se puede adaptar para tipo sugeno.

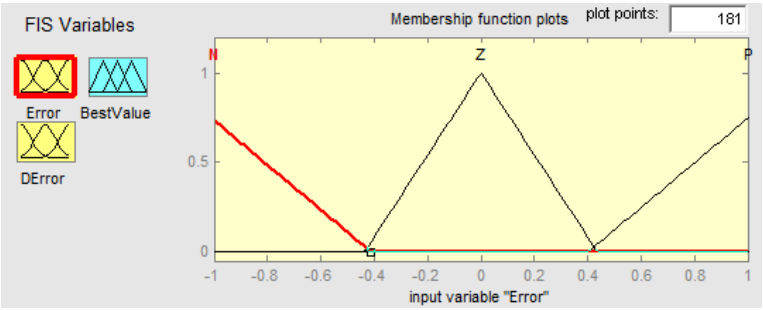


Fig. 3.2 Funciones de membresía del sistema difuso 'fuzzymain'

Todos los sistemas difusos tienen solo 9 reglas, que son las reglas que tiene un sistema difuso genérico y las necesarias para lograr la toma de decisiones y la adaptación de parámetros en este método propuesto. Por ejemplo, una regla es *'si el error es cero y el cambio del error es cero, entonces el mejor valor es cero'* en la figura 3.3 se pueden observar todas las reglas IF-THEN que se utilizaron en este sistema difuso.

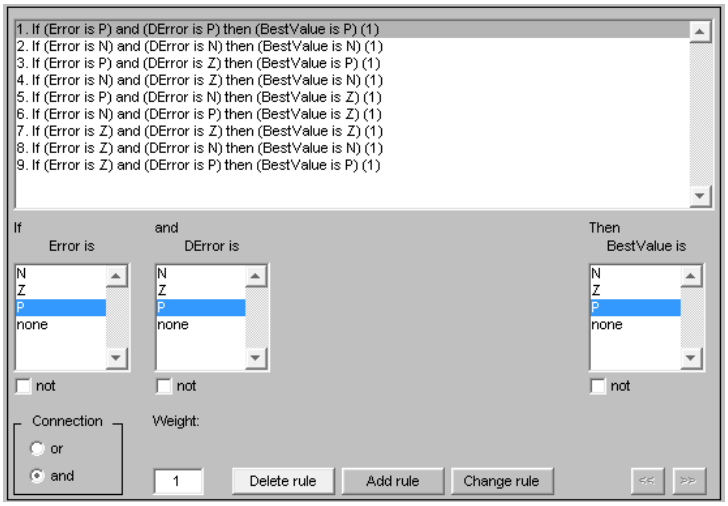


Fig 3.3 Reglas difusas If-Then del sistema difuso

En la figura 3.4 se puede ver el visor de reglas de este sistema difuso y en la figura 3.5 se muestra la superficie de control correspondiente a dicho sistema.

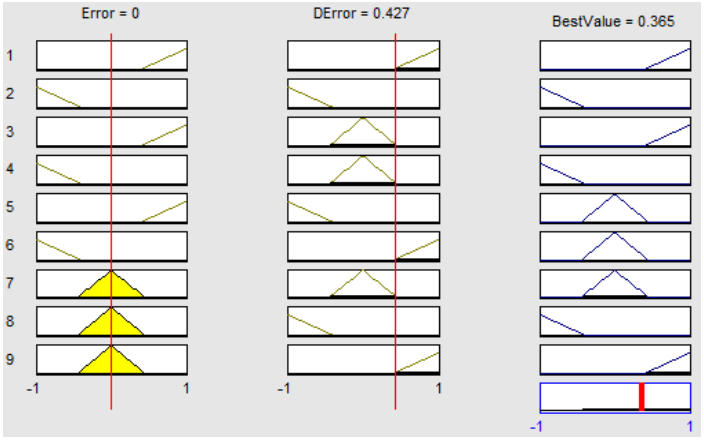


Fig 3.4 Visor de reglas del sistema difuso

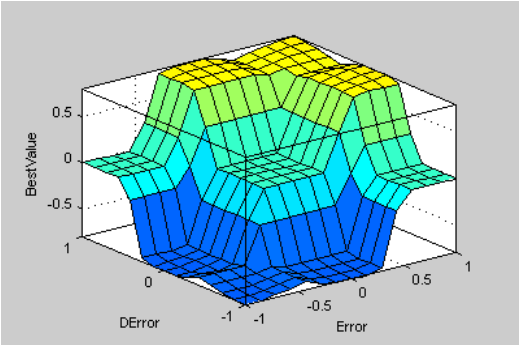


Fig 3.5 Superficie de control del sistema difuso

3.3 FPSO (Optimización por nube de partículas difusa)

Esta sección presenta una descripción detallada del modelo FPSO. La representación clásica del esquema para GAs son vectores binarios. En este caso, de un espacio de de búsqueda dimensional nx, donde cada individuo consiste de nx variables con cada variable codificada como una cadena binaria. La nube es típicamente modelada por partículas en un espacio multidimensional que tiene una posición y velocidad. Estas partículas vuelan a través del hiperespacio (i.e., R^n). Existen características esenciales, las cuales son su memoria, mejor posición local y global respecto a las partículas vecinas con el mejor global. En la figura 3.6 se puede observar una simulación de la función Sphere que fue implementada en esta investigación como problema de minimización de funciones. En capítulos posteriores se muestran los resultados obtenidos de todas las simulaciones hechas para todas las funciones analizadas y la aplicación en redes neuronales modulares. En un problema de minimización, el mejor valor es aquel que está más cercano al valor objetivo, en este caso, puede ser 0 u cualquier otro valor donde se encuentre el mínimo global de la

función analizada. Los individuos (llamados partículas) en una optimización por partículas comunican su mejor posición con cada partícula en cada iteración, y cada una intenta ajustarse a su mejor posición y velocidad basado en la comunicación que hay con sus partículas vecinas. Por lo tanto, una partícula tiene la siguiente información para saber cuándo es conveniente realizar un cambio en posición y velocidad:

- ✓ Cuando el mejor valor global es encontrado, inmediatamente se actualizan las partículas para encontrar la mejor posición en la nube.
- ✓ Los mejores vecinos (los que están más cerca de la solución) comunican su información a sus subconjuntos de partículas en la nube para que ellos converjan a ese punto.
- ✓ El mejor valor local, es aquel valor más cercano a la solución de un conjunto de partículas, el cual se comunica con los mejores locales de toda la nube para encontrar la mejor posición, o en otras palabras, el valor objetivo.

En este caso, la información social es la mejor posición encontrada por la nube y se denota como $\hat{y}(t)$. Para el mejor global de FPSO, la velocidad de la partícula es calculada por la siguiente expresión:

$$v_{ij}(t+1) = vi_j(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \qquad (3.1)$$

Donde $vi_j(t)$ es la velocidad de la partícula i en la dimensión j = 1,…,n_x en el paso del tiempo t, $xi_j(t)$ es la posición de la partícula i en la dimensión j en el paso del tiempo t, C_1 y C_2 representan la aceleración social y cognitiva. En este caso, C_1 y C_2 son

valores difusos que se incorporaron en esta nueva propuesta del método híbrido, ya que en un algoritmo PSO tradicional estos dos valores son constantes y nunca se mueven y $r_{1j}(t)$, $r_{2j} \sim U(0,1)$ son valores aleatorios en el rango [0,1].

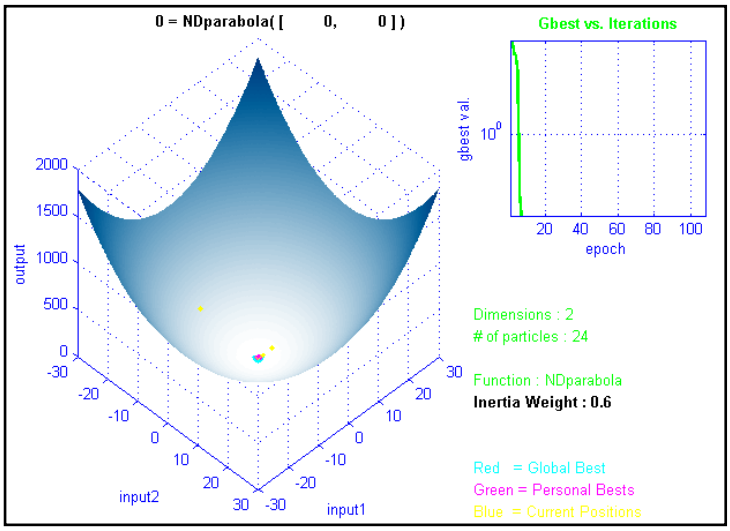


Fig 3.6 Simulación de la función Sphere con FPSO

3.4 FGA (Algoritmo Genético Difuso)

Esta sección presenta una descripción detallada del modelo FGA. Varios operadores de cruce y mutación han sido desarrollados para los algoritmos genéticos, dependiendo en como los individuos pueden ser representados. Para representaciones binarias, cruce uniforme, un solo punto y dos puntos son los más populares. En esta investigación se ha utilizado el cruce en dos puntos. A diferencia de un algoritmo genético tradicional, en esta parte se han incorporado dos operadores difusos, que son el cruce y la mutación, representados por las variables k_1 y k_2. Son difusos porque están cambiando dinámicamente en tiempo de ejecución del método híbrido. Esto es posible, debido a la incorporación de los sistemas difusos encargados de estar monitoreando

cuando es necesario adaptar y con qué porcentaje la tasa de los operadores de cruce y mutación en el GA.

3.5 Definición de los sistemas difusos utilizados en FPSO+FGA

En esta sección, se presentan las estructuras detalladas de los diferentes sistemas difusos que fueron utilizados en esta investigación, a continuación, se muestran los detalles de cada uno de los sistemas:

‘fuzzypso’: Este es un sistema difuso que se diseñó específicamente para manipular los parámetros en ‘fpso’, su función principal es adaptar los parámetros cuando sea conveniente hacerlo, estos parámetros son la aceleración social y cognitiva, mejor conocidas en el sistema como C_1 y C_2 . Su estructura es la siguiente:

Número de entradas: 2

Número de salidas: 2

Número de función es de membresía: 3

Tipo de funciones de membresía: Se hicieron pruebas con triangulares y gaussianas

Defusificación: Centroide.

‘fuzzyga’: Este es un sistema difuso que se diseñó específicamente para manipular los parámetros en ‘fga’, su función principal es adaptar los parámetros cuando sea conveniente hacerlo, estos parámetros son el cruce y la mutación, las cuales conocemos en el sistema como k_1 y k_2. Su estructura es la siguiente:

Número de entradas: 2

Número de salidas: 2

Número de función es de membresía: 3

Tipo de funciones de membresía: Se hicieron pruebas con triangulares y gaussianas

Defusificación: Centroide.

‘**fuzzymain’:** La función principal de este sistema difuso es decidir el mejor camino que se debe tomar en un momento dado para resolver el problema de optimización, en este caso, encontrar el mínimo global de una función matemática, minimizar los errores de la Red Neuronal Modular y encontrar la arquitectura optima de de la red.

3.6 Descripción matemática del modelo propuesto FPSO+FGA

En esta sección, se describe la definición formal matemática del método propuesto FPSO+FGA.

A lo largo de esta investigación se trabajó con algoritmos genéticos y optimización por nube de partículas. Tomando como referencia estas bases, a continuación se explicarán las aportaciones que se hicieron al uso de estas técnicas y que contribuyeron al funcionamiento completo del modelo propuesto. En la ecuación 3.2 se muestra una parte fundamental del algoritmo PSO tradicional, en la ecuación 3.3 se describe la manera en cómo se implemento la ecuación 3.2 para lograr el objetivo y convertir parte de ella en parámetros difusos, se puede observar que las diferencias entre las dos ecuaciones que las c_1 y c_2 en la ecuación 3.3 son las que cambian, debido a que una parte esencial de el método que se propuso radica en esas dos variables. Tradicionalmente estas dos variables son constantes, en este caso, debido a la importancia de esas dos aceleraciones, se decidió obtener estos dos parámetros de manera que un sistema difuso estuviera evaluando los comportamientos de errores y así poder obtener un porcentaje de variación más preciso.

$$v_{ij}(t+1) = vi_j(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{3.2}$$

La ecuación 3.3 muestra estos cambios. Donde v_{ij} es la velocidad de la partícula i en la dimensión j = 1,…, nx en el paso del tiempo t, x_{ij} (t) es la posición de la partícula i en la

dimensión j en el paso del tiempo t, los valores de c_1 y c_2 son difusas y se denotan en la expresión 3.4 y 3.5. r_1 y r_2 representan dos valores aleatorios entre cero y uno.

$$v_{ij}(t+1) = vi_j(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \tag{3.3}$$

$$c_1 = \frac{\sum_{i=1}^{r_{c_1}} \mu_i^{c_1}(c_{1i})}{\sum_{i=1}^{r_{c_1}} \mu_i^{c_1}} \tag{3.4}$$

Donde: c_1 = Porcentaje de aceleración social en determinado momento de la partícula.

r_{c_1} = Numero de reglas del sistema difuso correspondiente a c_1.
c_{1i} = salida de la regla i correspondiente a c_1.
$\mu_i^{c_1}$ = Membresía de la regla i correspondiente a c_1.

$$c_2 = \frac{\sum_{i=1}^{r_{c_2}} \mu_i^{c_2}(c_{2i})}{\sum_{i=1}^{r_{c_2}} \mu_i^{c_2}} \tag{3.5}$$

Donde: c_2 = Porcentaje de aceleración cognitiva en determinado momento de la partícula.

r_{c2} = Numero de reglas del sistema difuso correspondiente a c_2.
c_{2i} = salida de la regla i correspondiente a c_2.
$\mu_i^{c_2}$ = Membresía de la regla i correspondiente a c_2.

Las expresiones mencionadas arriba muestran la parte difusa que se agregó para hacer el ajuste de parámetros dinámico e inteligente de las aceleraciones en lo que respecta a PSO.

En la parte del GA los parámetros que se tomaron en cuenta para ajustarlos dinámicamente y obtener sus valores a partir de reglas difusas fueron los del cruce y la mutación (k_1 y k_2) por ser dos parámetros importantes para la convergencia del método se decidió tomar estos dos. El tamaño de la población también es importante, pero para esta investigación no se ajustó dinámicamente, sino que se propusieron de acuerdo a experiencias pasadas [56], la probabilidad de cruce y mutación se denotan en las expresiones 3.6 y 3.7.

$$k_1 = \frac{\sum_{i=1}^{r_{k1}} \mu_i^{k_1}(k_{1_i})}{\sum_{i=1}^{r_{k_1}} \mu_i^{k_1}} \tag{3.6}$$

Donde: k_1 = Porcentaje de cruce en determinado momento de la población.

r_{k1} = Numero de reglas del sistema difuso correspondiente a k_1.

k_{1i} = salida de la regla i correspondiente a k_1.

$\mu_i^{k_1}$ = Membresía de la regla i correspondiente a k_1.

Donde k_1 denota la probabilidad de cruce que debe ser entre el 0 y 100%.

$$k_2 = \frac{\sum_{i=1}^{r_{k_2}} \mu_i^{k_2}(k_{2i})}{\sum_{i=1}^{r_{k_2}} \mu_i^{k_2}} \qquad (3.7)$$

Donde: k_2 = Porcentaje de mutación en determinado momento la población.

r_{k2} = Numero de reglas del sistema difuso correspondiente a k_2.

k_{2i} = salida de la regla i correspondiente a k_2.

$\mu_i^{k_2}$ = Membresía de la regla i correspondiente a k_2.

Donde k_2 denota la probabilidad de mutación que debe ser entre el 0 y 10%.

3.7 ¿Porque se eligió trabajar con PSO yGA?

La motivación principal de desarrollar un nuevo método de computación evolutiva capaz de resolver problemas de optimización de funciones matemáticas y arquitecturas de redes neuronales modulares se describe a continuación:

- La necesidad de crear una nueva estrategia evolutiva con GA y PSO auxiliados con lógica difusa fue un factor fundamental para desarrollar el método que se propuso.
- Combinar las características fundamentales de GA y PSO de manera diferente a como se han estado trabajando estas dos estrategias.
- Evitar el ajuste de algunos parámetros a prueba y error para llegar al resultado óptimo en menor tiempo y con menor esfuerzo.

- Crear un nuevo método hibrido con ajuste de parámetros que funciona dinámicamente, haciéndolo más inteligente en la toma de decisiones con reglas if-then provenientes de la lógica difusa.

Por los factores que arriba se mencionan, se implemento la nueva estrategia evolutiva denominada FPSO+FGA, que lleva las iniciales de las estrategias utilizadas, la 'F' proviene del término en inglés ' Fuzzy Logic' en honor a la lógica difusa y con eso se indica que el algoritmo genético tiene ajuste de parámetros difusos. El término 'GA' significa proviene de las iniciales en inglés Genetic Algorithm, que significa algoritmo genético. Por otro lado el término 'PSO' proviene de las las iniciales Particle Swarm Optmization, que significa optimización por nube de partículas. Se añadió un símbolo de '+' para indicar la unión de estas dos estrategias evolutivas para la resolución de problemas de optimización.

3.8 Problemas de aplicaciones reales donde puede ser utilizado del método FPSO+FGA

Algunos de los problemas de aplicaciones reales donde puede ser implementada esta propuesta son los siguientes:

- Diseño de alas de aeronaves.
- Diseño de antenas.
- Controladores de redes neuronales Adaptivos.
- Trayectoria de misiles.
- Optimización de funciones matemáticas.

- Optimización de redes neuronales.
- Controladores difusos.
- Método de entrenamiento de redes neuronales.

En particular los resultados de simulación que se desarrollaron en esta investigación para validar el método fueron hechos para la optimización de arquitecturas de redes neuronales modulares y la optimización de un conjunto de funciones matemáticas complejas; sin embargo; queda abierta la posibilidad de hacer experimentos para resolver los problemas arriba mencionado, en el capitulo siguiente se muestran los resultados de simulación hechos con los que se validó el modelo propuesto.

3.9 Herramienta gráfica desarrollada para probar el método

Con el fin de que el nuevo método FPSO+FGA pueda ser utilizado por otros usuarios, se desarrolló una herramienta gráfica en el lenguaje de programación Matlab 7.5. Cabe mencionar que los resultados que se muestran en el siguiente capítulo fueron obtenidos desde la línea de comandos de Matlab.

La herramienta de interfaz grafica todavía se encuentra en fase de desarrollo, sin embargo, ya es posible realizar algunas simulaciones en ella para entender su funcionamiento y obtener algunos resultados en los problemas de optimización de funciones matemáticas complejas. Actualmente se está desarrollando la parte para poder hacer la simulación de la optimización de arquitecturas de redes neuronales modulares de manera grafica. La figura 3.7 muestra la pantalla principal que aparece cuando se invoca esta herramienta en Matlab. Se decidió desarrollar esta sencilla interfaz, con el fin de incluirla en Matlab, ya que actualmente solo existe la posibilidad de trabajar de manera gráfica solo con algoritmos genéticos.

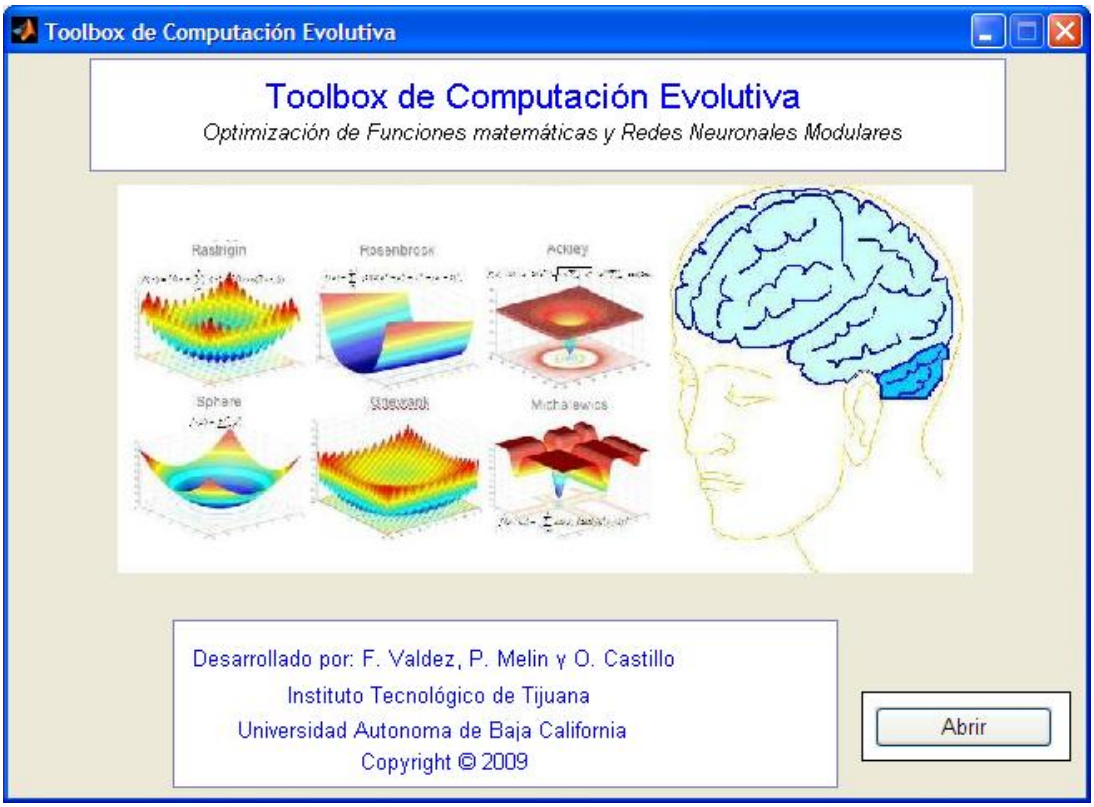


Fig. 3.7 Herramienta gráfica de computación evolutiva

La figura 3.8 muestra un ejemplo de uso de esta interfaz para la optimización de la función rastrigin para dos variables, se puede observar algunos campos los cuales permite inicializar algunos parámetros importantes en el GA y el PSO, pero con el transcurso de evolución del método estos están variando dinámicamente.

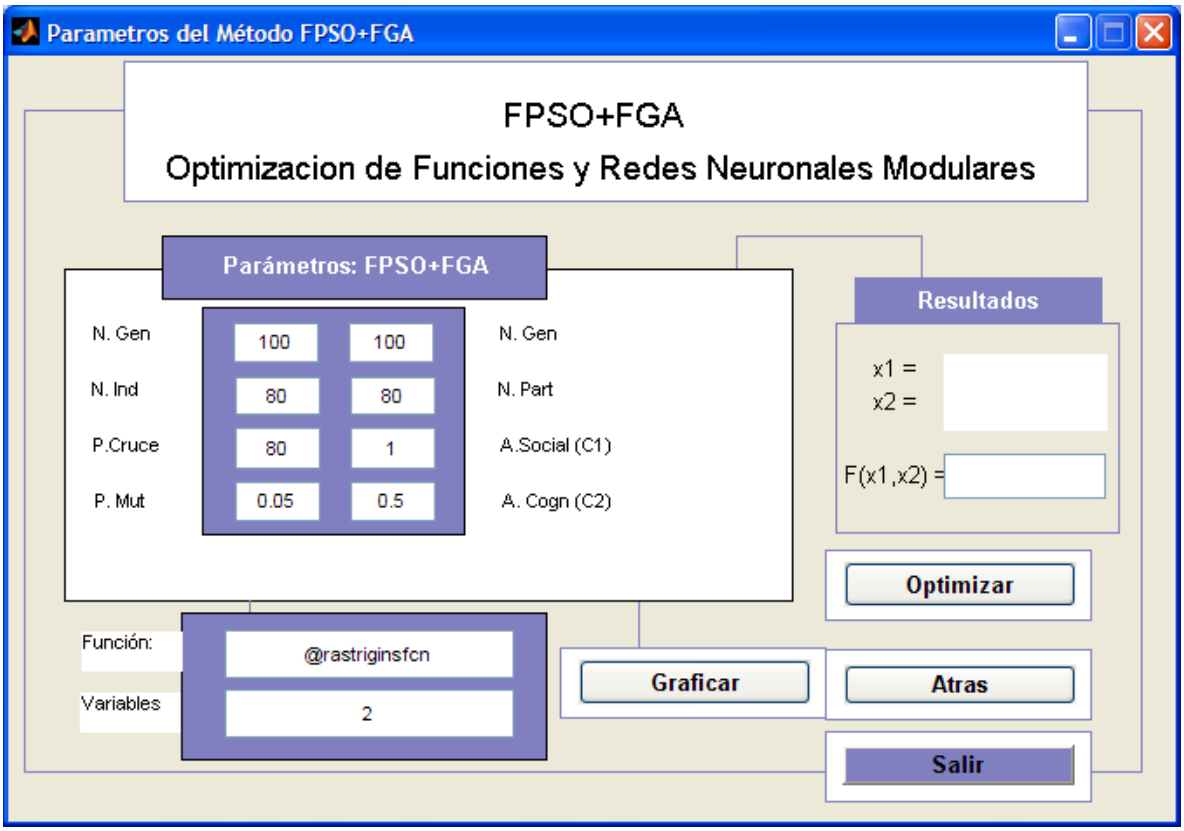


Fig. 3.8 Ejemplo gráfico de optimización.

Para poder hacer uso de esta aplicación, es necesario tener instalado el lenguaje de programación Matlab e invocar desde la línea de comandos la función que llamamos 'psogatool', que significa herramienta de para optimización de funciones y arquitecturas de redes neuronales modulares con PSO (optimización por nube de partículas) y algoritmos genéticos (GA). Una vez que abrimos la herramienta; la pantalla que se ejecutará es la del menú principal, como se muestra arriba en la figura 3.7, posteriormente presionamos abrir para interactuar con la herramienta de optimización de funciones matemáticas, la figura 3.8 muestra esa interfaz. El ejemplo que se muestra es para optimizar la función rastrigin, con dos variables, para poder obtener el resultado, es necesario introducir el nombre de la función precedida por una '@', como se muestra en la figura 3.8, se escribió '@rastrigin' para indicar el nombre de la función a optimizar, y en el cuadro de texto se indica el número de variables con las cuales que se evaluará la función, en este caso, el ejemplo es para 2 variables; sin embargo, esto depende del tipo de función, algunas pueden ser evaluadas para 'n' variables. En el capítulo de resultados de simulación, se describen detalladamente las características de las funciones que se utilizaron en esta investigación y cuantas variables soporta cada una de ellas. En esta interfaz gráfica también se pueden agregar otras funciones diferentes a las que ya trae incluida, solo es necesario crear el archivo de Matlab 'm', grabarlo en la dirección donde se encuentran las demás funciones y mandarlo llamar desde la interfaz o línea de comandos.

También, con el objetivo de visualizar los resultados de las simulaciones, es posible graficar estos valores, los cuales se han generado cuando el número de generaciones o el objetivo es alcanzado. La figura 3.9 muestra la grafica de las dos variables correspondientes al ejemplo que se muestra en la figura 3.8 y la mejor aptitud en todas las generaciones en este caso de un FGA; también es posible graficar otros datos, tales como, las generaciones transcurridas, la gráfica de la función, las mejores partículas o individuos, etc.

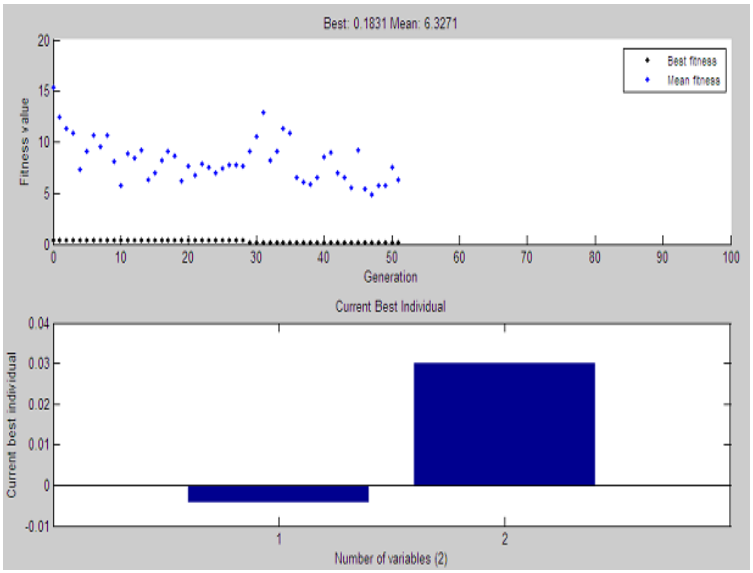


Fig. 3.9 Simulación de un FGA

Capítulo 4

Resultados de Simulación del Método propuesto FPSO+FGA

En este capítulo se describen los resultados de simulación con el nuevo método híbrido de computación evolutiva al cual se le llamó FPSO+FGA que se propuso para esta investigación, después de ser aplicado al reconocimiento de patrones; en este caso, se analizó, la base de datos Yale para validar el método, también fue validado haciendo uso de 7 funciones matemáticas benchmark [56,30] con distintas variables para observar eficiencia y rendimiento del método. En este capítulo, el lector observará las ventajas que tiene implementar este método híbrido, así como un estudio comparativo de cuando se usan GAs y PSOs separadamente. Algunos problemas de optimización pueden ser vistos en [39].

4.1 Descripción de funciones matemáticas benchmark utilizadas.

En esta sección, se describen detalladamente las diferentes funciones matemáticas benchmark que se utilizaron para validar este método.

4.1.1 Función Rastrigin (Ras)

La función rastrigin esta denotada por la siguiente ecuación:

$$f(x) = 10n + \sum_{i=1}^{n} (x^2 - 10\cos(2\pi x_i)) \qquad (4.1)$$

Donde el mínimo global es: $x^* = (0, \ldots, 0), f(x^*) = 0$

La figura 4.1 muestra la gráfica correspondiente para esta función (Ver anexo 1)

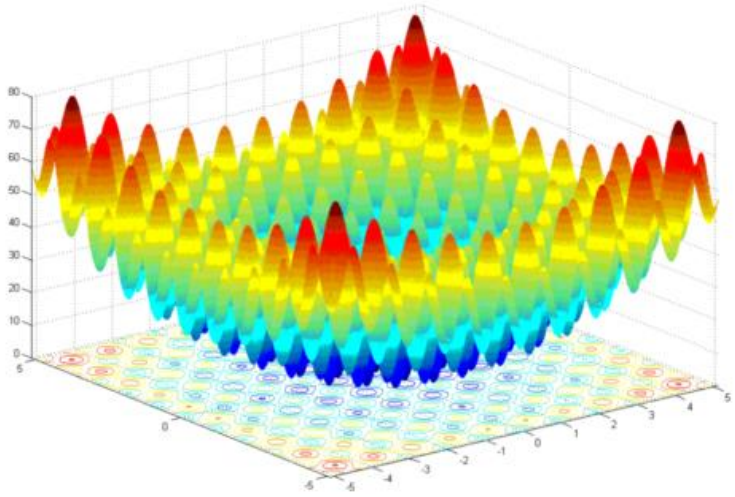


Fig. 4.1 Función rastrigin

4.1.2 Función Rosenbrock (Ros)

La función rosenbrock esta denotada por la siguiente ecuación:

$$f(x) = \sum_{i=1}^{n-1} [100(x_i^2 - x_i^2 + 1)^2 + (x_i - 1)^2] \qquad (4.2)$$

Donde el mínimo global es: $x^* = (1, \ldots, 1), f(x^*) = 0$.

La figura 4.2 muestra la gráfica correspondiente para esta función (Ver anexo 2)

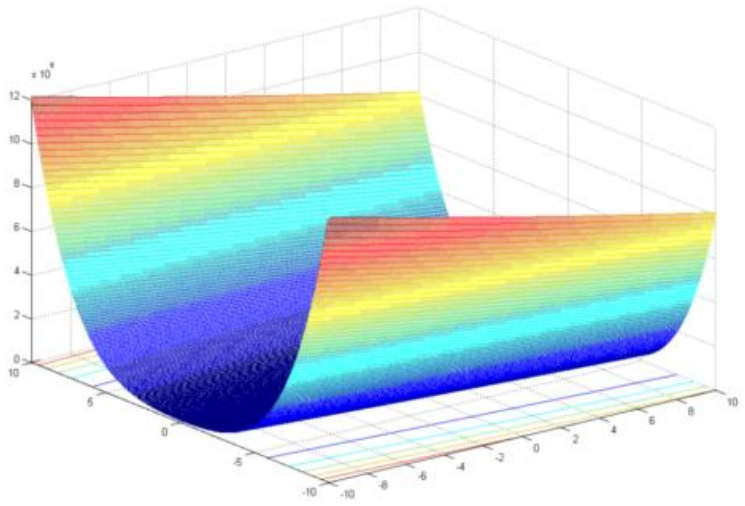


Fig. 4.2 Función rosenbrock

4.1.3 Función Ackley (Ack)

La función ackley esta denotada por la siguiente ecuación:

$$f(x) = 20 + e - 20e^{-1/5}\sqrt{1/n\sum_{i=1}^{n} x_i^2} - e^{1/n}\sum_{i=1}^{n} \cos(2\pi x_1) \qquad (4.3)$$

Donde el mínimo global es: $x^* = (0, \ldots, 0), f(x^*) = 0$.

La figura 4.3 muestra la gráfica correspondiente para esta función (Ver anexo 3)

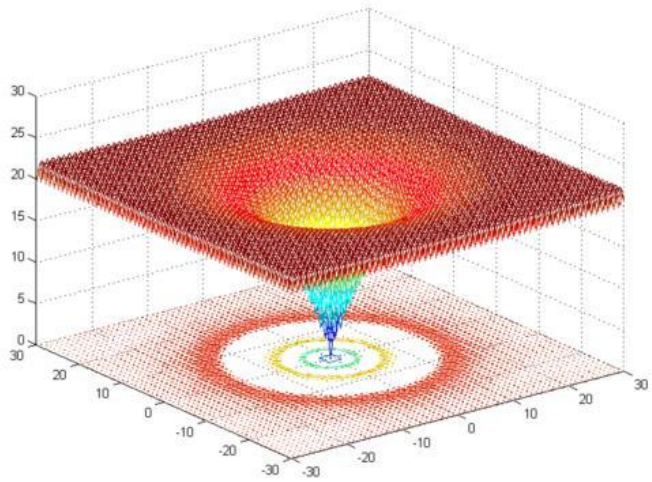


Fig. 4.3 Función ackley

4.1.4 Función Sphere (Sph)

La función sphere esta denotada por la siguiente ecuación:

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{4.4}$$

Donde el mínimo global es: $x^* = (0, \ldots, 0), f(x^*) = 0$.

La figura 4.4 muestra la gráfica correspondiente para esta función (Ver anexo 4)

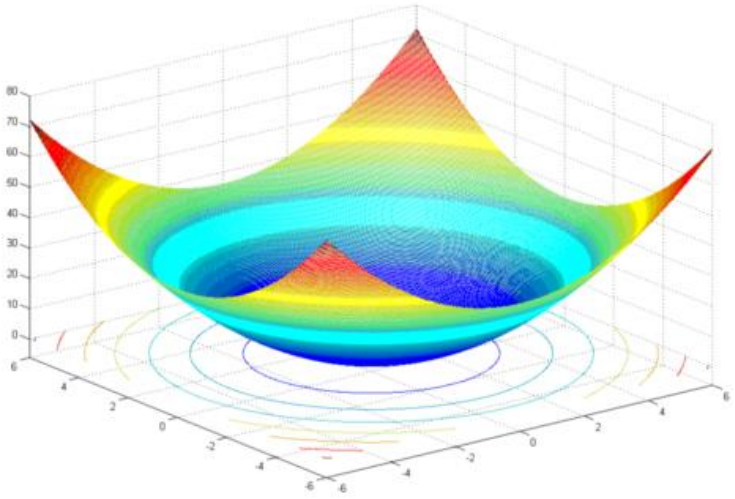


Fig. 4.4 Función Sphere

4.1.5 Función Griewank (Grw)

La función griewank esta denotada por la siguiente ecuación:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(x_i / \sqrt{i}) + 1 \tag{4.5}$$

Donde el mínimo global es: $x^* = (0, \ldots, 0), f(x^*) = 0$.

La figura 4.5 muestra la gráfica correspondiente para esta función (Ver anexo 5)

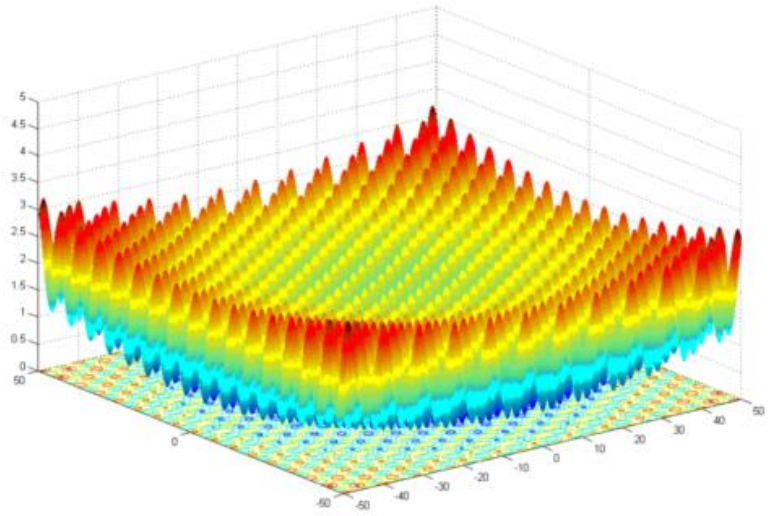


Fig. 4.5 Función griewank

4.1.6 Función Michalewics (Mich)

La función michalewics esta denotada por la siguiente ecuación:

$$f(x_{1\ldots}x_n) = -\sum_{j=1}^{2}\sin(x_j)(\sin(jx_j^2/pi))^{2m} \qquad (4.6)$$

m=10

Número de mínimos locales: varios.

Mínimo global: $f(x^*) = -1.8013$.

La figura 4.6 muestra la gráfica correspondiente para esta función (Ver anexo 6)

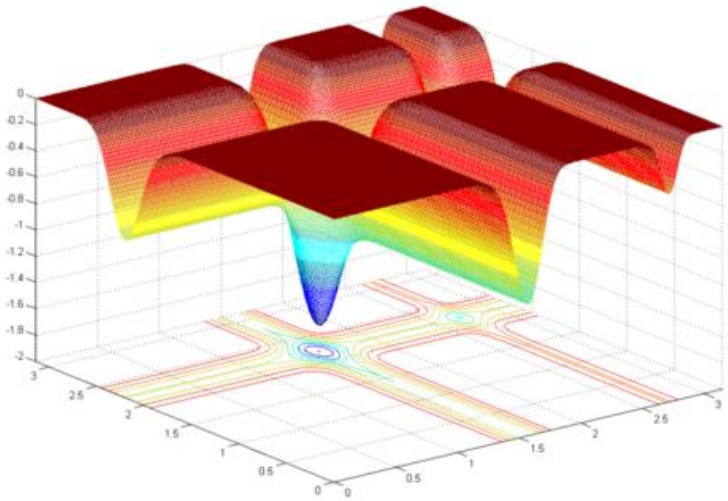


Fig. 4.6 Función michalewics

4.1.7 Función Zakharov (Zak)

La función zakharov esta denotada por la siguiente ecuación:

$$Zn(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4 \qquad (4.7)$$

Mínimo global: $x^* = (0, \ldots, 0)$, $Z_n(x^*) = 0$.

La figura 4.7 muestra la gráfica correspondiente para esta función (Ver anexo 7)

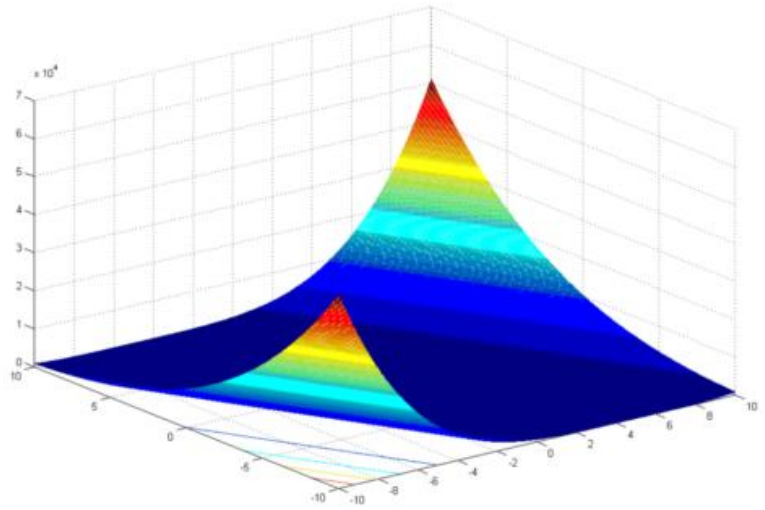


Fig. 4.7 Función zakharov

4.2 Resultados de simulación para la optimización de Redes Neuronales y funciones matemáticas

En esta sección se muestran los resultados de simulación de algunos casos de estudio que sirvieron para validar el enfoque de esta investigación. Diversas pruebas del método FPSO+FGA se desarrollaron para lograr la optimización de Redes Neuronales Monolíticas y Modulares, así como de funciones matemáticas complejas.

Todas las implementaciones fueron desarrolladas utilizando una computadora quadcore de 64 bits que trabaja a una frecuencia de reloj de 2.5 GHz, 4 GB de memoria RAM y Windows Vista como sistema operativo.

4.2.1 Caso de estudio 1: Optimización de arquitectura de red neuronal monolítica y funciones matemáticas.

En estos resultados de simulación se utilizó una red neuronal monolítica con las siguientes características en la topología inicial:

No. De capas: 3

Capa 1: 45 neuronas

Capa 2: 50 neuronas

Capa 3: 50 neuronas

Método de entrenamiento: Gradiente conjugado (gdx)

La idea es encontrar la arquitectura óptima de la red neuronal. Se utilizó una base de datos de rostros llamada Yale [14], que contiene 165 imágenes en escala de grises en formato GIF correspondiente a 15 individuos, para este experimento solo se utilizaron 5 individuos para el entrenamiento de la red neuronal. Se utilizaron 5 imágenes por individuo en diferentes expresiones faciales: centrado iluminado, feliz, luz del lado izquierdo, normal y luz al lado derecho. La figura 4.8 muestra las imágenes que fueron utilizadas en este experimento.

Fig. 4.8 Imágenes de la base de datos Yale

Se utilizaron 3 imágenes por individuo para entrenar a la red neuronal, las otras dos muestras se utilizaron para identificación de los individuos. Se utilizó un algoritmo genético jerárquico para representar la estructura de la red neuronal. Como arriba se mencionan, las capas y nodos por capa, en total fueron 148 bits para poder hacer esta representación, los primeros 3 bits fueron para las capas y el resto fue para representar el número de nodos por capa. En PSO la estructura de la cadena de bits fue similar que en el GA, la diferencia es que el PSO tiene menos parámetros de ajuste. La función aptitud utilizada en este caso para la red neuronal, combina la información del error objetivo y también el número de nodos como segundo objetivo. La siguiente ecuación muestra esta información:

$$f(z) = \left(\frac{1}{\alpha * Ranking(ObjV1) + \beta * ObjV2} \right) * 10 \tag{4.8}$$

El primer objetivo es básicamente el promedio de la suma del cuadrado de los errores calculado por las salidas que arroja la red neuronal comparada con los valores reales de la función, que está dada por la siguiente ecuación:

$$f_1 = \frac{1}{N} \sum_{i=1}^{N} (Y_i - y_i)^2 \tag{4.9}$$

El segundo objetivo es la complejidad de la red neuronal, la cual es medida por el número total de nodos en la arquitectura.

La topología final de la red neuronal se obtuvo con el método propuesto, FPSO+FGA, GA y PSO, pero el error final es diferente. La comparación de los valores objetivos finales es mostrada en la tabla 4.1.

En la arquitectura final, el resultado de la evolución de la red neuronal es una arquitectura con diferente número de capas y neuronas por capas. Se obtuvo la siguiente arquitectura final: solo dos capas con 20 neuronas para la primera capa y 15 neuronas para la segunda capa. El método propuesto optimiza la arquitectura inicial propuesta para el problema de reconocimiento de patrones. Con esta topología final la red fue entrenada y las 5 imágenes fueron identificadas. En este experimento no fue tan complicado para la red neuronal lograr el 100% en identificación de los 5 individuos que se tomaron como prueba, en experimentos posteriores se mostraran resultados con mas imágenes y con redes neuronales modulares, donde la complejidad del problema aumenta pero aun así se obtienen resultados satisfactorios.

En la tabla 4.1, se muestra un estudio comparativo entre GA, PSO y FPSO+FGA. En este caso, se muestran los errores promedios de 50 pruebas, con cada método empleado arriba mencionados. Como se puede observar se incluyen resultados con 7 funciones matemáticas complejas, que también fueron utilizadas para validar este enfoque y la función de la red neuronal. Para este experimento, las funciones solo fueron evaluadas para 2 variables, más adelante se muestran otros resultados con más variables y con otras funciones más, incluyendo una red neuronal modular más compleja. La figura 4.9 muestra la gráfica correspondiente a los resultados de la tabla 4.1.

Tabla 4.1. Comparación de resultados entre los tres métodos analizados

Math Funct	GA	PSO	FPSO+FGA	Objective Value
Ras	2.15E-03	5.47E-05	3.05E-04	0
Ros	1.02E-05	1.97E-03	1.17E-02	0
Ack	2.98	2.98	4.98E-03	0
Sph	1.62E-04	8.26E-11	1.05E-10	0
Grie	2.552E-05	2.56E-02	1.07E-07	0
Mich	-1.7829	-7.44E-01	-1.8201	-1.8013
Zakh	0.00146674	8.10	0.00168	0
NN	3.33E-01	2.3E-01	1.01E-03	0

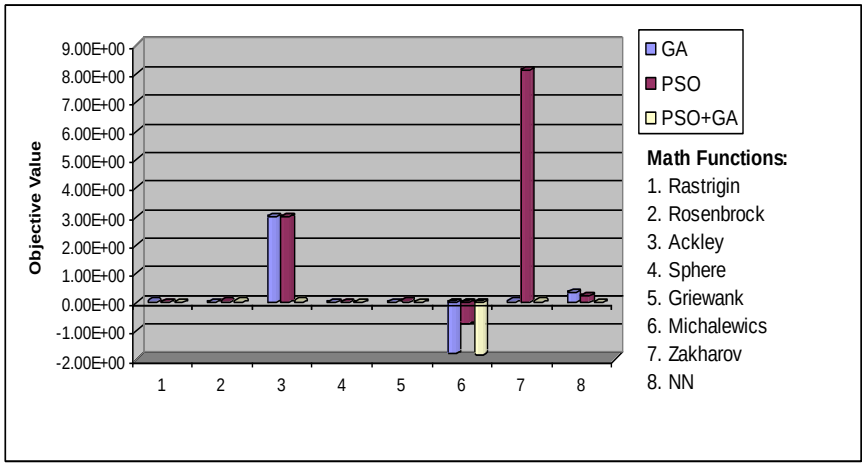


Fig. 4.9 Resultados comparativos entre los 3 métodos.

4.2.2 Caso de estudio 2: Optimización de arquitectura de red neuronal monolítica.

Este caso de estudio, difiere del anterior en el numero de imágenes que se utilizaron, aquí se utilizaron 10 imágenes de la base de datos, en lugar de cinco (ver figura 4.12). El ajuste dinámico de parámetros para este caso solo fue en el cruce y en la aceleración social. La arquitectura inicial propuesta también fue diferente como se menciona a continuación (ver figura 4.10):

No. De capas: 3

Capa 1: 60 neuronas

Capa 2: 50 neuronas

Capa 3: 45 neuronas

Método de entrenamiento: Gradiente conjugado (gdx)

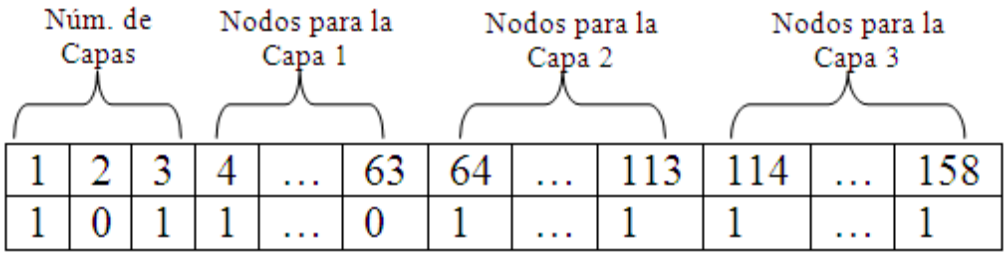


Fig. 4.10 Representación binaria de la arquitectura inicial de la red propuesta

Después de aplicar el método propuesto para este caso de estudio, los resultados que se obtuvieron fueron los siguientes, la red neuronal se simplificó de 158 bits a solo 63 bits, quedando solo con 2 capas la red, 26 nodos para la primera capa y 35 nodos para la segunda. La figura 4.11 muestra la representación binaria de la arquitectura de la red neuronal optimizada.

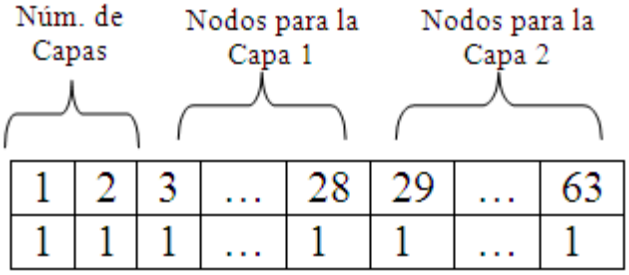


Fig. 4.11 Representación binaria de la arquitectura de la red optimizada.

La tabla 4.2 muestra los resultados correspondientes a esta simulación, como se puede observar, en el experimento número 3 de la tabla 4.2, se obtuvo la identificación de las 10 imágenes, el error meta también fue alcanzado.

Los encabezados de las tabla 4.2 son descritos a continuación:

Ls= Número de capas.

NL= Número de nodos de la capa 1, 2 y 3.

GE= Error meta.

RE = Error alcanzado.

IDENT= Imágenes identificadas.

Los tamaños de las poblaciones en estas pruebas fueron de 100 individuos; el cruce fue difuso ya que el método se encarga de cambiar dinámicamente el porcentaje de cruce, la mutación para este experimento si fue variada a prueba y error en un rango de entre 2 % y 5%. Como se observa en la tabla 4.2, el método propuesto una vez más arroja buenos resultados, en esta ocasión fue posible optimizar las arquitecturas de la red arriba mencionadas, minimizar los errores e identificar al 100% las imágenes que se pusieron de prueba.

Tabla 4.2 Resultados de simulación para 10 imágenes.

Ls	NL 1	NL 2	NL 3	GE	RE	IDENT
3	34	25	33	0.0001	0.0038	8
3	25	29	38	0.0001	0.0075	8
2	26	35	0	0.0001	0.0001	10
2	34	38	0	0.0001	0.0009	9
1	10	0	0	0.0001	0.0086	7

4.2.3 Caso de estudio 3: Optimización de arquitectura de red neuronal modular y funciones matemáticas

Para la optimización de la Red Neuronal Modular, al igual que la Red Neuronal Monolítica, se definió una topología inicial como punto de partida del método. Se

utilizaron tres capas con un método de entrenamiento del gradiente conjugado, el objetivo fue minimizar el error de la red neuronal y optimizar las capas y neuronas por capa de la red. A diferencia de la Red Neuronal Monolítica, aquí se vuelve más compleja la arquitectura de la red, por el motivo de que las imágenes son segmentadas en tres partes y así facilitar la identificación de los individuos. En este experimento se aumentó el número de imágenes a 10 individuos, utilizando la misma base de datos Yale del experimento anterior ahora se añadieron otros 5 individuos más que no se contemplaron con la Red Neuronal Monolítica. En total fueron 50 imágenes, las cuales se observan en la figura 4.12, son las 5 imágenes anteriores más otras 5 más. En experimentos posteriores se muestran resultados de simulación con más imágenes. Las condiciones para el entrenamiento de la RNM son similares a las de la monolítica, es decir, se utilizaron tres imágenes para entrenamiento y dos para probar la identificación de individuos. La figura 4.13 muestra una simulación del entrenamiento de la RNM en Matlab, cuando el método propuesto FPSO+FGA encontrar la arquitectura optima para este problema y minimizar el error del mismo.

Fig. 4.12 Imágenes de la base de datos Yale

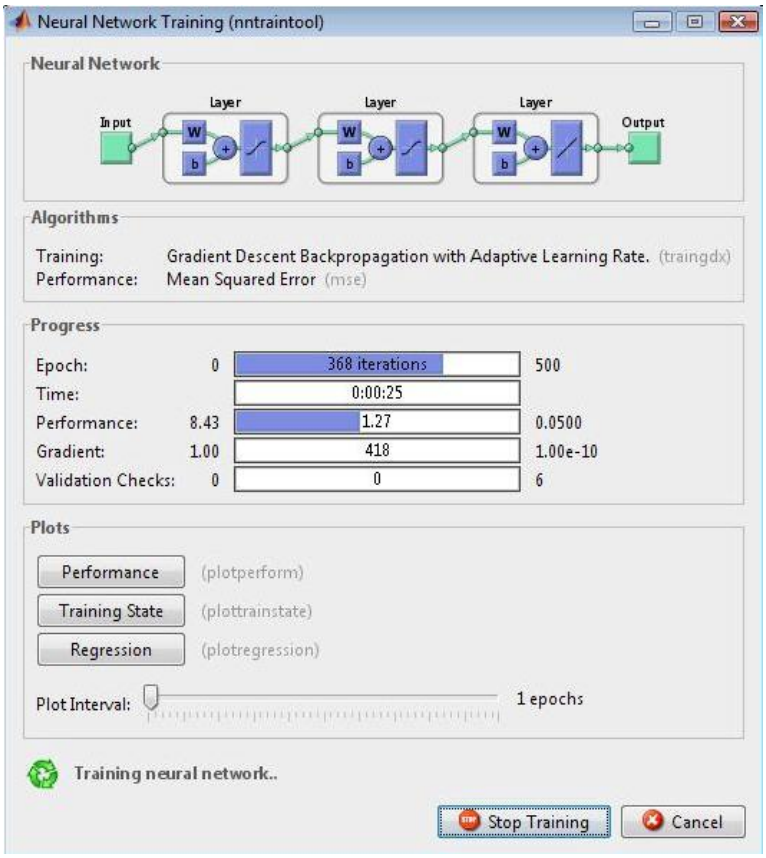


Fig. 4.13 Simulación de entrenamiento de la RNM

En la tabla 4.3, 4.4 y 4.5 se puede observar que este método es una buena alternativa para la optimización de RNM y funciones matemáticas complejas. Los parámetros en FPSO+FGA; como el cruce, mutación, aceleración social y aceleración cognitiva son difusos, el tamaño de la población fue de 100 individuos para todos los experimentos.

En la tabla 4.3 se pueden observar los resultados más relevantes de las simulaciones hechas con el método FPSO+FGA, en esta tabla se observa que el experimento 2 y el 5 lograron la identificación de las 10 imágenes de prueba que se tomaron para validar el

enfoque, alcanzando en los dos casos el error meta propuesto. En este experimento los sistemas difusos empleados fueron diseñados con funciones de membresía triangulares.

Tabla 4.3 Resultados de simulación con funciones de membresía triangulares

LMod	NNL1 M1	NNL2 M1	NNL1 M2	NNL2 M2	NNL1 M3	NNL2 M3	GE	RE	IDENT
2	20	60	80	50	60	120	0.01	0.03	8
2	90	50	100	150	70	90	0.01	0.005	10
2	70	40	80	40	90	30	0.01	0.02	8
2	150	135	200	90	84	40	0.01	0.003	9
2	100	120	100	145	100	70	0.01	0.001	10

La tabla 4.4 muestra los resultados de las simulaciones hechas con el método FPSO+FGA, variando el tipo de función de membresía. En este experimento los sistemas difusos empleados fueron diseñados con funciones de membresía gausianas. Se puede observar que en estos experimentos no se logro identificar al 100% todas las imágenes, en la mayoría de los casos, solo 9 de las 10 imágenes fueron identificadas. Con estos, resultados se puede observar que el cambio en las funciones de membresía afecto el rendimiento de los resultados, ya que todo lo demás permaneció en las mismas condiciones.

Tabla 4.4 Resultados de simulación con funciones de membresía gausianas

LMod	NNL1 M1	NNL2 M1	NNL1 M2	NNL2 M2	NNL 1M3	NNL2 M3	GE	RE	IDENT
2	30	50	90	70	50	115	0.01	0.05	8
2	85	60	103	140	80	130	0.01	0.02	9
2	90	50	95	50	85	40	0.01	0.08	8
2	130	120	180	90	70	50	0.01	0.02	9
2	50	90	75	70	40	40	0.01	0.01	9

La figura 4.14 muestra la representación binaria de la arquitectura final de uno de los experimentos que lograron la identificación de las 10 imágenes.

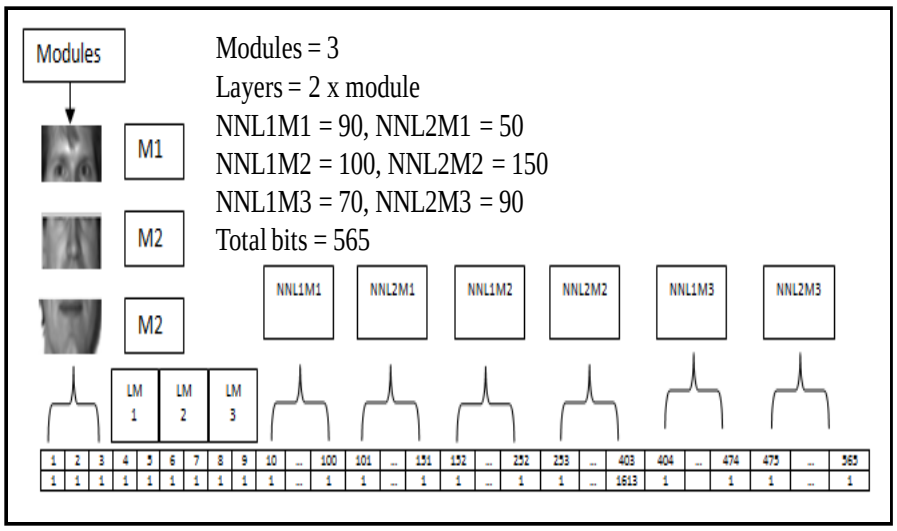


Fig. 4.14 Representación binaria optimizada con FPSO+FGA

La tabla 4.5, muestra los resultados de simulacion con las funciones matemáticas que se propusieron para esta investigacion.

Tabla 4.5. Resultados de simulación para funciones matemáticas.

Math F	**Variables = 2**		**Variables= 4**		**Variables = 8**		**Variables = 16**	
Math	Mejor	Promedio	Mejor	Promedio	Mejor	Promedio	Mejor	Promedio
Ras	1.45E-06	3.05E-04	0.0003	0.0755	0.01318	0.9311	0.5483	5.0946
Ros	1.17E-02	1.17E-02	0.0285	0.5991	0.15800	3.8925	0.2555	4.33334
Ack	8.42E-04	4.98E-03	8.42e-01	4.98E-02	0.7	1.56	2.35	2.63
Sph	5.75E-11	1.05E-10	1.946e-05	4.5109e-004	0.00059	0.0057	0.00248	0.0211
Gri	7.88E-11	1.07E-07	7.18e-06	1.1182e-004	0.00016	9.299e-004	0.00040	0.0043
Mich	-1.8010	-1.8201	-1.80129	-1.8002	-1.80130	-1.8005	-1.801301	-1.8004
Zak	6.00E-07	0.00168	3.237e-07	8.4129e-005	1.3308e-07	6.4901e-005	8.63410e-07	7.0065e-005

4.2.4 Caso de estudio 4: Optimización de funciones matemáticas usando cómputo distribuido.

En este caso de estudio, se hizo uso de lo que es el cómputo distribuido para utilizar las estrategias GA y PSO, con el fin de evaluar los tiempos de ejecución de cada uno de los dos métodos. El método propuesto actualmente se encuentra en desarrollo para hacerlo de manera distribuida, ya que una desventaja que se puede presentar cuando se utiliza localmente en una computadora, es que los tiempos de ejecución son bastante elevados. Con los casos de estudio arriba mencionados no se tuvieron inconvenientes por implementarlo en una sola computadora, pero debido a que se piensa implementar para bases de datos más grandes, se está adaptando para trabajar con el de manera distribuida. A continuación solo se muestran resultados obtenidos a la fecha para la optimización de 2 funciones matemáticas con dos variables. Los experimentos que se han logrado con el método FPSO+FGA, se están adaptando para desarrollarlos de manera distribuida. La tabla 4.6 y 4.7 muestran los resultados de simulación para un GA simple, el objetivo fue encontrar el mínimo global de la función rastrigin y rosenbrock, así como minimizar los tiempos de ejecución del GA en diferentes experimentos, como se observa en la tablas 4.6 y 4.7, el porcentaje de cruce y de mutación fueron escogidos a prueba y error para realizar los experimentos. Las poblaciones del GA fueron diferentes. El experimento 5 y 7 tuvieron los mejores resultados de la tabla 4.6 para la función rastrigin, los promedios fueron obtenidos de 50 pruebas en las mismas condiciones del GA. El experimento 6 tuvo los mejores resultados para la función rosenbrock de la tabla 4.7. En este caso de estudio observamos que los tiempos se reducen cuando hacemos uso de más de un procesador, en este caso se utilizaron 4 procesadores Pentium IV que trabajan a una frecuencia de reloj de

3.2 Ghz. 1 GB de memoria RAM y Windows XP como sistema operativo, se utilizó un modelo de paralelismo llamada maestro-esclavo para hacer estas simulaciones.

Las principales características de este método son las siguientes:

En este esquema, un procesador hace de administrador del sistema, mientras el resto busca la solución en la región que le fuera asignada por el administrador, como se muestra la figura 4.15 [1].

Existe un proceso maestro que es el responsable de repartir el procesamiento entre un número determinado de procesos esclavos o clientes, y de coordinar los cálculos de los mismos [1]. Otros estudios acerca de paralelismo pueden ser vistos en [18,53]

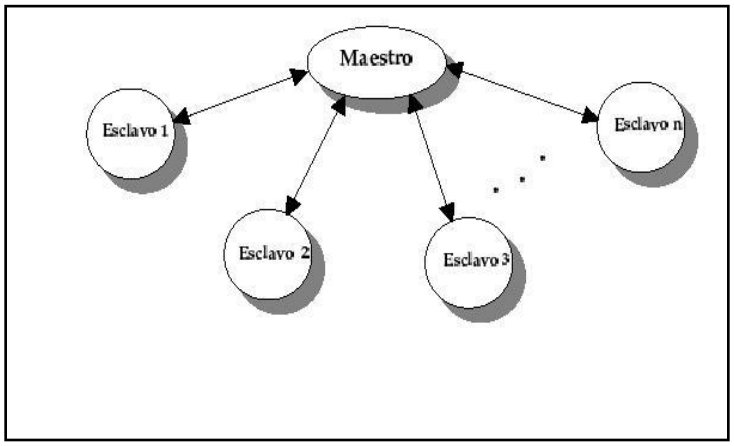


Fig 4.15 Esquema de paralelización Maestro/Esclavo

A continuación se muestran las funciones del Maestro y el Esclavo

Maestro:

- Distribuir y controlar las tareas.
- Integración de las Respuestas.
- Guardar los resultados de los entrenamientos.

Esclavo:

- Realizar cálculos para optimizar la función

- Enviar resultados al maestro.

El método Maestro-Esclavo trabaja de tal forma que se tiene un solo administrador de las tareas de todos los procesadores, los cuales realizan su proceso y regresan resultados a su administrador llamado también Mayordomo o Maestro [19].

Tabla 4.6 Resultados de simulación para la función rastrigin con GA.

Mejor	**Promedio**	**Tiempo Secuencial**	**Tiempo Distribuido**	**%Mut**	**%Cruce**	**Población**
0.06395	13.765	150.2	83.1	9	80	50
0.09526	13.26	75.0	45.1	9	80	50
0.0467	10.6632	150.4	91.7	10	70	80
0.05367	0.05616	200.4	121.4	10	50	100
0.02905	**0.02930**	225.3	137.0	9	50	100
0.000955	1.0073	2528.4	1182.4	9	90	80
7.647e-005	3.5598	4955.7	2413.6	9	90	80

Tabla 4.7 Resultados de simulación para la función rosenbrock con GA

Mejor	**Promedio**	**Tiempo Secuencial**	**Tiempo Distribuido**	**%Mut**	**%Cruce**	**Población**
0.007588	61.55528	3430.5	1381.3509	9	90	80
0.01872	38.372	150.4	71.8	10	80	50
0.01016	35.6924	276.0	132.6	9	90	70
0.00741	120.40786	400.6	194.3	9	60	40
0.006096	**15.64709**	776.1	377.6	9	95	70

La tabla 4.8 y 4.9 muestran los resultados de simulación para un PSO, el objetivo fue encontrar el mínimo global de la función rastrigin y rosenbrock, así como minimizar los tiempos de ejecución del GA en diferentes experimentos, como se observa en las tablas 4.8 y 4.9. Las poblaciones del GA fueron diferentes. El experimento 5 tuvo los mejores resultados de la tabla 4.8 para la función rastrigin, los promedios fueron obtenidos de 50

pruebas en las mismas condiciones del PSO. El experimento 3 tuvo los mejores resultados para la función rosenbrock de la tabla 4.9.

Tabla 4.8 Resultados de simulación para la función rastrigin con PSO

Mejor	Promedio	Tiempo Secuencial	Tiempo Distribuido	Población
0.994959	2.89	205	120	80
0.994992	4.24	140	85	40
0.448E-07	4.24	140	85	40
0.505E-08	2.89	205	120	80
0.909E-10	2.89	205	120	80

Tabla 4.9 Resultados de simulación para la función rosenbrock con PSO

Mejor	Promedio	Tiempo Secuencial	Tiempo Distribuido	Población
0.010087	140	103	65	20
0.162045	63	133	74	40
0.012542	28.6	220	146	80
0.114029	187	110	66	40
0.000935	56.08	103	65	20
0.000262	35.13	220	146	80
0.000020	35.13	220	146	80

Con los resultados obtenidos en estos experimentos, se puede observar que el uso de varios procesadores es de gran utilidad para poder resolver problemas complejos en tiempos muy cortos. El método FPSO+FGA es capaz de resolver problemas de optimización complejos, por ejemplo, los casos arriba mencionados consumen mucho poder de cómputo, simplemente, al introducir la arquitectura de una red neuronal como problema de optimización, lo hace tardado; es por ello, que se está adaptando para que pueda trabajar de manera distribuida, como en este caso, que ya se obtuvieron resultados de manera separada con GA y PSO para estas dos funciones arriba mencionadas.

4.2.5 Caso de estudio 5: Optimización de funciones matemáticas con 32 variables.

En este caso de estudio, se hizo uso de 5 funciones matemáticas que se propusieron para esta investigación. Se aumentó el número de variables hasta 32 para cada una de las funciones para validar con mayor precisión el método y poder ver con claridad las ventajas que se obtienen al utilizar el nuevo método propuesto. La tabla 4.10 muestra los resultados de simulación para las 5 funciones con 32 variables. Los resultados que se muestran son los promedios obtenidos de 50 pruebas después de ejecutar los tres métodos. Los tamaños de las poblaciones fueron de 150 individuos para GA y 150 partículas para PSO. La figura 4.16 muestra la gráfica correspondiente a la tabla 4.10, como se puede observar, a mayor numero de variables FPSO+FGA se mantiene más cercano al objetivo deseado, a diferencia de GA y PSO aisladamente, cuando se incrementan las variables el problema es más complejo y más difícil lograr el objetivo.

Tabla 4.10 Comparación de resultados entre los tres metodos

Funciones	GA promedio 32 Variables	PSO promedio 32 Variables	FPSO+FGA promedio 32 Variables	Valor objetivo
Rastrigin	12.4163	25.11	11.8083	0
Rosenbrock	67.48	125.33	6.7274	0
Ackley	31.03	36.37	3.03	0
Sphere	0.0754	7.14E-06	0.0630	0
Griewank	0.555	0.455	0.0049	0

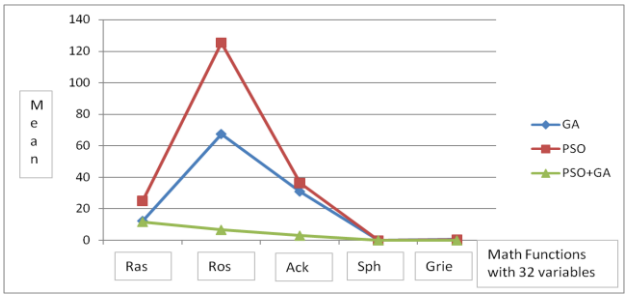


Fig 4.16 Comparación de los tres métodos propuestos

4.3 Comparación de resultados

En esta sección, se presentan similitudes de otro método que han estado realizando otros investigadores en el área de computación evolutiva y que se utilizaron como referencia para realizar este trabajo de investigación. Aunque con enfoques diferentes se hace mención por compartir cosas en común.

En este trabajo de investigación fue necesario estudiar otros métodos similares que se han desarrollado en los últimos años para problemas de optimización, por ejemplo, el método HEM puede ser visto en [30], HEM es un método auto-adaptativo e inteligente, algunas de las funciones que se utilizaron para validar FPSO+FGA también fueron utilizadas en HEM, una de las diferencias principales es que FPSO+FGA también fue utilizado para encontrar arquitecturas óptimas de redes neuronales modulares en el reconocimiento de patrones que se mencionaron en el capítulo 4, y que fueron parte de el objetivo general de esta investigación. Las funciones solo se utilizaron para validar el enfoque, los resultados que se mostraron en el capítulo anterior, son resultados que ya han sido publicados en artículos científicos internacionales, el método propuesto en esta tesis puede ser validado para n variables, tanto FPSO+FGA y HEM son dos buenas alternativas para la solución de problemas de optimización con n variables, las condiciones de la evaluación de los dos métodos fueron distintas, por lo que para compararlos se necesitarían hacer experimentos en condiciones similares, por ejemplo, los promedios de evaluaciones, los tamaños de las poblaciones y el número de variables para cada prueba fueron distintos, como

consecuencia de esto, en algunos casos específicos se lograron mejores resultados con FPSO+FGA y para otros fue mejor HEM , por ejemplo para la función rosenbrock nosotros utilizamos hasta 32 variables y en la tesis doctoral [29]del Dr. Oscar Montiel se hicieron experimentos hasta 16. Cabe mencionar que el enfoque principal de FPSO+FGA es minimizar los errores de redes neuronales modulares para posteriormente encontrar la arquitectura óptima de una aplicación de reconocimiento de patrones. Otros trabajos de investigación desarrollados en el área pueden ser vistos en [23,19].

Capítulo 5

Conclusiones

Después de haber analizado los métodos de computación arriba descritos en esta investigación, se puede observar la importancia que tiene construir sistemas híbridos inteligentes. Los algoritmos genéticos y la optimización por nube de partículas aplicados a problemas de optimización, han sido dos técnicas que se han utilizado para resolver dichos problemas, pero después de realizar esta investigación, se llega a la conclusión que pueden ser mejores al combinar sus características principales y uniéndolas con reglas IF-THEN se pueden obtener resultados más fácilmente y muy significativos. Como se muestra en los resultados de simulación, en problemas con pocas variables no son muy significativas las diferencias con el método propuesto y las otras dos técnicas utilizadas separadamente; pero conforme se aumenta el número de variables, los problemas se vuelven más complejos y difíciles de resolver, y es aquí donde el método que se propuso FPSO+FGA destaca sobre los otros dos. También, el método es utilizado para la optimización de arquitecturas de redes neuronales. Con esta investigación se comprueba la eficiencia de construir sistemas híbridos

inteligentes, en este caso, una de las principales ventajas del método desarrollado FPSO+FGA, fue la adaptación de los operadores difusos en tiempo de ejecución del método, esta fue una aportación que lo hace diferente a los otros dos utilizados aisladamente, ya que con ellos, a veces los resultados no son buenos por cuestiones de que se está trabajando a prueba y error con los parámetros que se tienen que introducir, pero se consideran como buenas técnicas para la resolución de problemas de búsqueda y optimización. En cambio con la propuesta que se desarrolló, es de gran ayuda este ajuste de parámetros para lograr mejores resultados en menor tiempo posible. La pregunta es: ¿Es posible aplicar técnicas siempre?, la respuesta depende del problema hay que considerar el espacio de búsqueda, si es posible hacer una representación de él y detalles que se tienen que tomar en cuenta a la hora de ponerse a hacer el sistema, es muy importante también entender el problema antes que programar, es un consejo que se les da a las personas que trabajen con estos métodos.

Aplicando el método FPSO+FGA se pueden encontrar arquitecturas de redes neuronales y optimizar funciones matemáticas y obtener resultados confiables.

Trabajo Futuro

En esta sección se proponen algunos aspectos importantes que se descubrieron en esta investigación y en donde se pueden abordar muchas cosas derivadas de esta propuesta, a continuación se enlistan las que se consideran más importantes para continuar con esta investigación.

- Validar el método con más aplicaciones del mundo real (Sistemas de control)
- Hacer pruebas con bases de datos de imágenes con más imágenes.
- Hacer comparación equitativa con otros métodos similares, por ejemplo, el HEM
- Terminar de desarrollar la herramienta grafica para hacer simulaciones.
- Hacerlo que funcione en un ambiente paralelo y distribuido para cuando los problemas son demasiado complejos y requieren mucho poder de cómputo el método sea factible.

De lo anterior mencionado ya se está trabajando con algunos aspectos, tales como, la herramienta de interfaz grafica, hacerlo que funcione paralelamente y se están buscando más aplicaciones para poder resolver problemas de optimización. Solo que en

esta investigación todos los resultados que se muestran son resultados que han sido publicados en congresos y revistas internacionales.

Referencias

[1] Alba, E. Parallel Metaheuristics: A New Class of Algorithms. John Wiley & Sons, October 2005.

[2] Blum, C. and Roli, A. "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". ACM Computing Surveys,

[3] Davis, L. “Handbook of Genetic Algorithms”. New York. Van Nostrand Reihold. 1991.

[4] Davis, L. “The GENITOR algorithm and selection pressure – why rank-based allocation of reproduction trials is best”. In: Schaffer, J.D. (ed.): Proceedings of the Third International Conference on Genetic Algorithms, 133-140. Morgan Kaufmann, San Mateo, CA, 1989.

[5] De Jong, K. “The analysis and behaviour of a class of genetic adaptive systems”. PhD thesis, University of Michigan, 1975.

[6] Dubois, D. y Prade, H. “Fuzzy Sets and Systems: Theory and applications”,

[7] Dubois, D. y Prade, H.”Fuzzy Sets in approximate reasoning II (Logical approaches)”, 40, pp. 203-244, 1991.

[8] Estivil, V. Y Duch, A. "Aplicaciones de algoritmos genéticos", www.lania.mx, 2004

[9] Glover, F. and Kochenberger, G. "Handbook of Metaheuristics". Kluwer Academic Publishers, Norwell, MA, 2002.

[10] Golberg, D “Genetic Algorithms in search, optimization and machine learning”. Ed.

Addison Wesley, 1989.

[11] Gudise G. and Venayagamoorthy G. "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks". In Proceedings of the IEEE Swarm Intelligence Symposium 2003, pages 110 117, Indianapolis, Indiana, USA, 2003.

[12] Hájek, P. y Godó L. “Deductive Systems of Fuzzy Logic”, 1997.

[13] Higashi, N. and Iba, H. “Particle swarm optimization with gaussian mutation”, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, Indianapolis (USA), pp. 72-79.,April 2003

[14] Hiremath, P.S. and Prabhakar C.J. ”Extraction and Recognition of Nonlinear Interval Type Features Using Symbolic KDA Algorithm with Application to Face Recognition”. Volume 2008, 5 pags, 2008.

[15] Holland, J. (1981). “Artificial Adaptive Agents in Economic Theory”. American Economic Review, Papers and Proc 81, 365-70

[16] Holland, J., "Adaptation in natural and artificial systems" (University of Michigan Press), 1975.

[17] Holland, John. “Genetic algorithms”. Scientific American, p. 66-72, 1992.

[18] Javid, T. , Albert, Y. and Zomaya, A. "Simulated Annealing Approach for Mobile Location Management. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium - Workshop 6, page 194, Washington, DC, USA, IEEE Computer Society. 2005.

[19] Juang, C-F.“A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design”. IEEE Transactions on Systems, Man, and Cybernetics,

2003.

[20] Kantrowitz, M. “Fuzzy Logic and Fuzzy Expert Systems”, 1995.

[21] Kaufman, A. “Introducción a la teoría de los subconjuntos difusos”, Cía.

[22] Kennedy J., Eberhart R., and Shi Y. "Swarm Intelligence". San Francisco: Morgan Kaufmann Publishers, 2001.

[23] Lu, W.Z. Fan, H-Y. and Lo S.M.. “Application of Evolutionary Neural Network Method in Predicting Pollutant Levels in Downtown Area of Hong Kong”. Neurocomputing, 387-400, 2003.

[24] Mahfoud, S.W. “Niching Methods for Genetic Algorithms” PhD thesis, University of Illionois, Illinois, 1995.

[25] Man K., Tang K., and Kwong, S. “Genetic Algorithms: Concepts And Designs”,

[26] Martin, F. y Mcneill, E. “Fuzzy Logic”, Editorial AP Professional, 1994.

[27] Michalewics, Z. "Genetic Algorithms + Data Structure = Evolution Programs". Springer-Verlag, 1996.

[28] Michalewics, Z. “A survey of Constraint Handling Techniques in Evolutionary Computation Methods”. In proceedings of the Fourth Annual Conference on Evolutionary Programming, pages 135-155, 1955.

[29] Montiel, O. “El Modelo Evolutivo Humano. Un enfoque para optimización global”, Tesis doctoral UABC, 2006.

[30] Montiel, O., Castillo, O., Melin, P. Sepulveda, R. Rodriguez, A. “Human evolutionary model: A new approach to optimization”, Information Sciences: an International Journal Volume 177 , Issue 10, May 2007.

[31] Morales, G. “Introducción a la lógica difusa”, Febrero de 2004.

[32] Omran M., Salman A., and Engelbrecht A. "Image Classication Using Particle Swarm Optimization". In Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002, pages 370 374, Sin- gapore, 2002.

[33] Onwubolu, G. C. and Clerc, M, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization", International Journal of Production Research, vol. 4 pp. 473-491, 2004.

[34] Ourique C., Biscaia E., and Pinto J. "The Use of Particle Swarm Optimization for Dynamical Analysis in Chemical Processes". In Proceedings of the Computers and Chemical Engineering, volume 26, pages 1783 1793, 2002.

[35] Ozcan, E. and Mohan, C.K. "Particle swarm optimization: surfing the waves", Proceedings of the 1999 Congress on Evolutionary Computation-CEC99, Washington (USA), Vol. 3, pp. 1939-1944. July 1999.

[36] Parsopoulos K., Papageorgiou E., Groumpos P., and Vrahatis M. "A First Study of Fuzzy Cognitive Maps Learning Using Particle Swarm Optimization". In Proceedings of the IEEE Congress on Evolutionary Computation 2003, pages 1440 1447, Canbella, Australia, 2003.

[37] Pérez, J.R. and Basterrechea, J. "Particle-swarm optimization and its application to antenna far-field pattern prediction from planar scanning", Microwave and Optical Technology Letters, Vol. 44, No. 5, March 2005, pp. 398-403.2005.

[38] Pérez, J.R. and Basterrechea, "Hibrid particle swarm algorithms and their application to linear array synthesis". *Progress In Electromagnetics Research, PIER 90, 63–74, 2009*

[39] Ray, T.and Liew, K.M. "A Swarm Metaphor for Multiobjective Design

Optimization. Engineering Optimization", 2002.

[40] Ray,T. and Liew, K. M. "A Swarm with an Effective Information Sharing Mechanism for Unconstrained and Constrained Single Objective Optimization Problems, Proc. congress on evolutionary computation 2001, Seoul, Korea, IEEE service center, Piscataway, NJ, IEEE, 75-80, 2001.

[41] Rechenberg, I . "Evolution Strategy", Computational Intelligence: Imitating Life, IEEE Press, Piscataway, NJ, (1994).

[42] Rechenberg, I. "Evolution strategy: Optimization of technical systems by means of biological evolution", Fromman-Holzboog, Stuttgart, 1973.

[43] Reynolds, C. W. "Flocks, herds and schools: a distributed behavioral model", Computer Graphics, 21(4), : 25-34, 1987.

[44] Riget, J. and Vesterstrøm, J.",The Tomato Producing Particle Swarm Optimizing Real-World Dynamic Problems with Particle Swarms", In preparation, 2002.

[45] Riget, J. and Vesterstrøm, J."Controlling Diversity in Particle Swarm Optimization", In preparation, 2002.

[46] Robinson, G. E. " Regulation of Vision of Labor in Social Insect". Societies, Annu. Rev. Entomol. 37, : 637-665. 1992.

[47] Robinson, G. E. "Modulation of Alarm Pheromone Perception in the Honey Bee: Evidence for Division of Labour Based on Hormonally Regulated Response Thresholds", J. Comp. Physiol. A 160:613-619., 1987

[48] Robinson, G. E. , Page, R. E. and Huang, Z.-Y. "Temporal Polyethism in Social Insects is a Developmental Process", Anim. Behav. 48,: 467-469. 1994.

[49] Secrest B, and Lamont G. "Communication in Particle Swarm Optimization

Illustrated by the Traveling Salesman Problem". In Proceedings of the Workshop on Particle Swarm Optimization 2001, Indianapolis, 2001.

[50] Shi , Y. and Eberhart, R.C. "Parameter selection in Particle Swarm Optimization. In Proceedings of the Seventh Annual Conference on Evolutionary Programming", 1998.

[51] Shi, X. Lu, Y. Zhou, C. Lee, H. Lin, W. andLiang, Y ."Hybrid Evolutionary Algorithms Based on PSO and GA. In Proceedings of the IEEE Congress on Evolutionary Computation", vol 4, dec 2003.

[52] Stacey, M. Jancic, I. Grundy, "Particle swarm optimization with mutation", Proceedings of the 2003 Congress on Evolutionary Computation, Canberra (Australia), December 2003, Vol. 2, pp. 1425-1430.

[53] Subatra, R. and Zomaya, A. "A Comparison of Three Artificial Life Techniques for Reporting Cell Planning in Mobile Computing". IEEE Transactions on Parallel and Distributed Systems, 14(2):142 153, Feb 2003.

[54] Valdez, F., and Melin, P. "A New Evolutionary Method with a Hybrid Approach Combining Particle Swarm Optimization and Genetic Algorithms Using Fuzzy Logic for Decision Making" 2008 IEEE World Congress on Computational Intelligence, Hong Kong, China. (WCCI 2008), 2008.

[55] Valdez, F., and Melin, P. "A New Evolutionary Method with Fuzzy Logic for Combining Particle Swarm Optimization and Genetic Algorithms: The Case of Neural Networks Optimization" 2008 IEEE World Congress on Computational Intelligence, Hong Kong, China. (WCCI 2008), 2008.

[56] Valdez, F., and Melin, P. "Comparative Study of Particle Swarm Optimization and

Genetic Algoritms for Mathematicals Complex Functions". Journal of Automation, Mobile Robotics and Intelligent Systems. JAMRIS 2008

[57] Valdez, F., and Melin, P. "Neural Network Optimization with a Hybrid Evolutionary Method that combines Particle Swarm and Genetic Algorithms with Fuzzy Rules".North American Fuzzy Information Processing Society. NAFIPS 2008 Nueva York,. USA, 2008.

[58] Xie X., Zhang W. Z., and Yang. " Solving Numerical Optimization Problems by Simulating Particulates in Potential Field with Cooperative Agents". In International Conference on Articial Intelligence, Las Vegas, NV, USA, 2002.

[59] Zadeh, L. "Fuzzy Logic", IEEE Computer, Vol. 1, pp. 83, 1988.

[60] Zadeh, L. "Fuzzy Sets and Applications". (Selected Papers, edited by R.R. Yager, S. Ovchinnikov, R.M. Tong, H.T. Nguyen), John Wiley, Nueva York, 1987.

[61] Zadeh, L."Fuzzy sets", Journal of Information and Control, vol. 8, pp. 338-353, 1965.

Anexos

En esta sección se describen algunos códigos importantes que se desarrollaron en la investigación en el lenguaje de programación matlab.

Anexo 1.

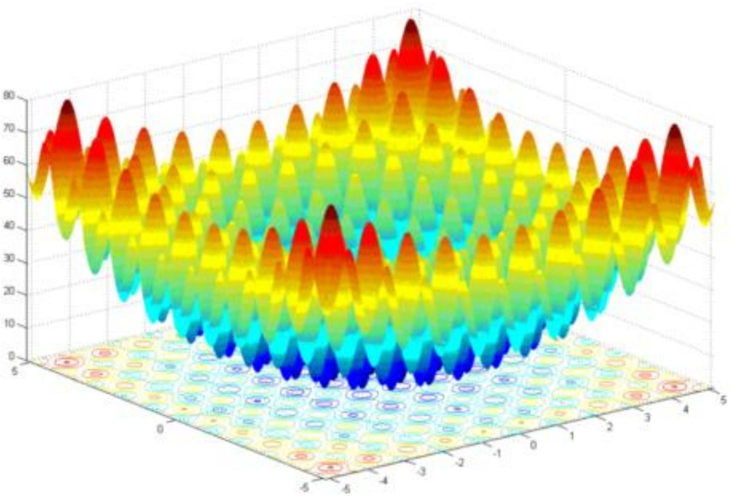
```
% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION RASTRIGIN
% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)
% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO
% EL VALOR POR DEFAULT ES DE N = 2

function y = rast(x)
%
%
n = 2;
s = 0;
for j = 1:n
   s = s+(x(j)^2-10*cos(2*pi*x(j)));
end
y = 10*n+s;
%%%GRAFICAR LA FUNCION RASTRIGIN
```

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

```
function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @rastriginsfcn;
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)
```

%%%%%Grafica de la función



%%

%

Anexo 2.

```
% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION ROSENBROCK
% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)
% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO

function y = rosen(x)
%
n = 2;
sum = 0;
for j = 1:n-1;
   sum = sum+100*(x(j)^2-x(j+1))^2+(x(j)-1)^2;
end
```

```
y = sum;
```

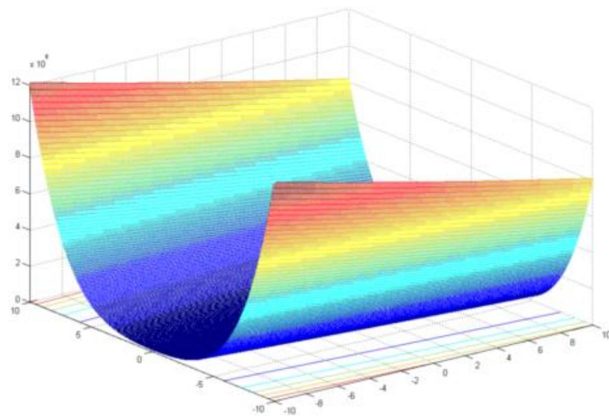
%%%GRAFICAR LA FUNCION ROSENBROCK

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

```
function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @rosen;
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)
```

%%%%%Grafica de la función.



%%

%

%%

%

Anexo 3.

% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION ACKLEY

% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)

% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO

% EL VALOR POR DEFAULT ES DE N = 2

function y = ackley(x)

%

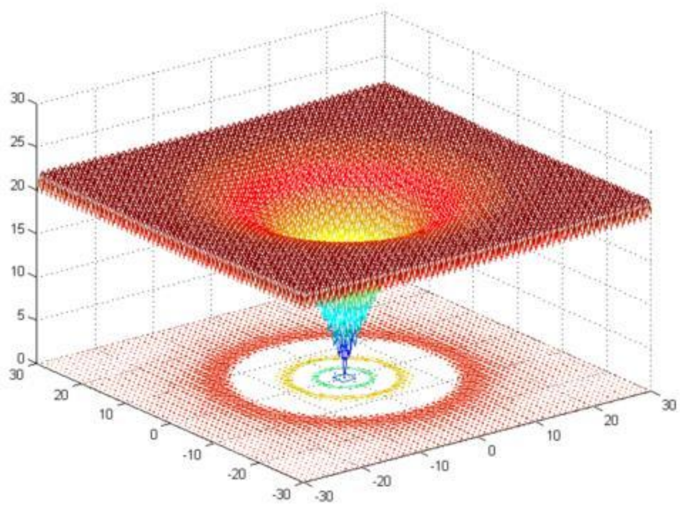
%

n = 2;

```
a = 20; b = 0.2; c = 2*pi;

s1 = 0; s2 = 0;

for i=1:n;

   s1 = s1+x(i)^2;

   s2 = s2+cos(c*x(i));

end

y = -a*exp(-b*sqrt(1/n*s1))-exp(1/n*s2)+a+exp(1);



%%%GRAFICAR LA FUNCION ACKLEY

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @ackley;
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);
```

```
surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)
```

%%%%%Grafica de la función



%%%

%

Anexo 4.

% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION SPHERE

% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)

% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

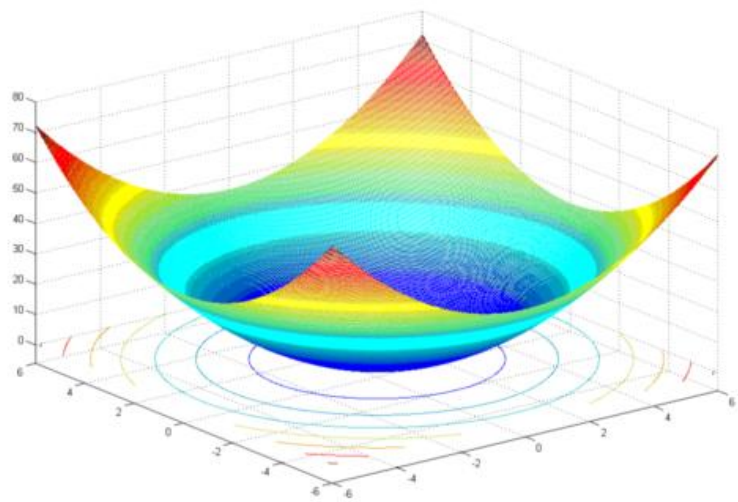
% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO

% EL VALOR POR DEFAULT ES N=30

```
function y = sphere(x)
%
```

```
n = 30;
s = 0;
for j = 1:n
   s = s+x(j)^2;
end
y = s;

%%%GRAFICAR LA FUNCION  SPHERE

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @sphere;
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)
```

%%%%%Grafica de la función



%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%**%**

Anexo 5.

% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION GRIEWANK

% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)

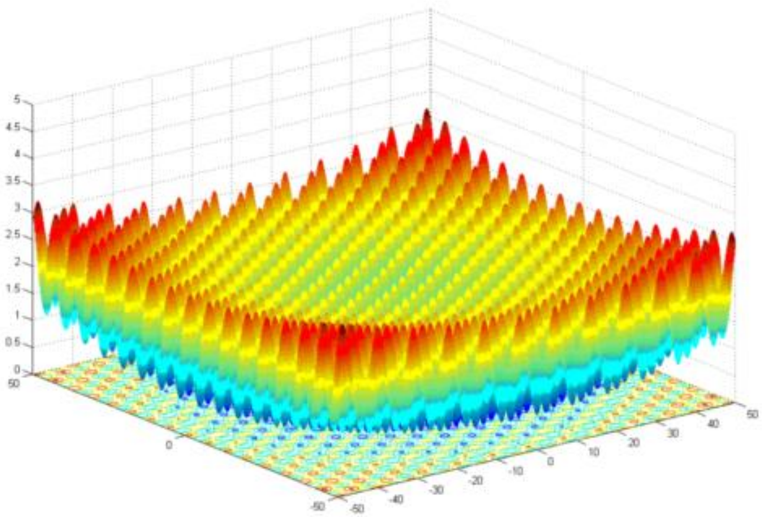
% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO

% EL VALOR POR DEFAULT ES N=2

```
function y = griewank(x)
%
%
n = 2;
fr = 4000;
s = 0;
p = 1;
for j = 1:n; s = s+x(j)^2; end
```

```
for j = 1:n; p = p*cos(x(j)/sqrt(j)); end
y = s/fr-p+1;

%%%GRAFICAR LA FUNCION  GRIEWANK

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @griewank;
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)


%%%%%Grafica de la función
```



%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Anexo 6.

% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION MICHALEWICS

% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)

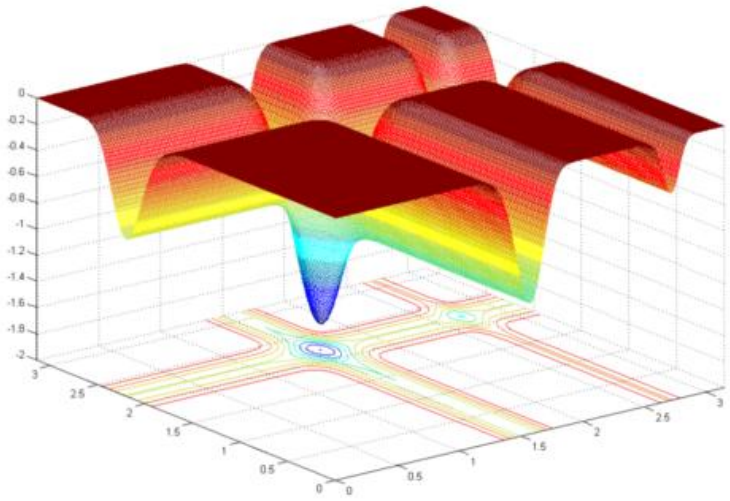
% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO

% EL VALOR POR DEFAULT ES N=2

```
function y = mich(x)
%
%
n = 2;
m = 10;
s = 0;
for i = 1:n;
   s = s+sin(x(i))*(sin(i*x(i)^2/pi))^(2*m);
end
y = -s;
```

```
%%%GRAFICAR LA FUNCION  MICHALEWICS

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @mich;
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)


%%%%%Grafica de la función
```



Anexo 7.

% CODIGO PARA REPRESENTAR EN MATLAB LA FUNCION ZAKHAROV

% ELABORO> MC. FEVRIER ADOLFO VALDEZ ACOSTA. (2009)

% UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA

% EL NUMERO DE VARIABLES PUEDE SER AJUSTADO

% EL VALOR POR DEFAULT ES N=2

```
function y = zakh(x)
%
n = 2;
s1 = 0;
s2 = 0;
for j = 1:n;
    s1 = s1+x(j)^2;
    s2 = s2+0.5*j*x(j);
end
y = s1+s2^2+s2^4;
```

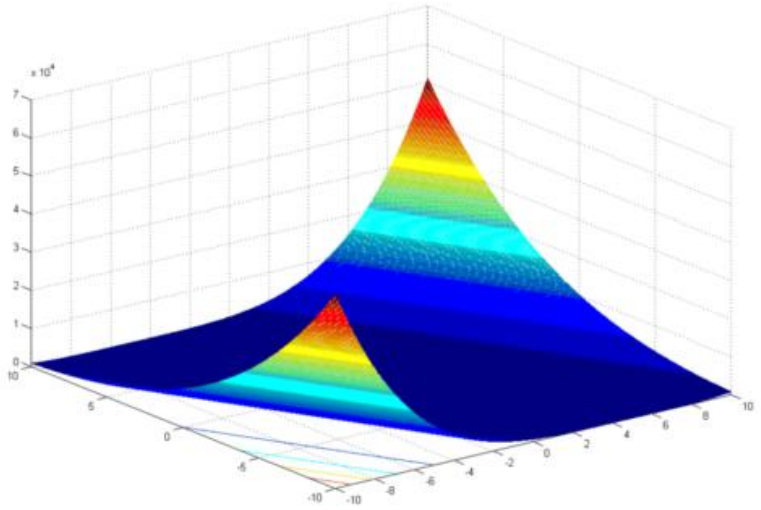
%%%GRAFICAR LA FUNCION MICHALEWICS

%%Se utilize la function plotobjective que viene incluida en matlab %%para hacer la

graficacion de la function.

```
function plotobjective(fcn,range)

if(nargin == 0)
   fcn = @zakh;
```

```
   range = [-15,15;-15,15];
end

pts = 100;
span = diff(range')/(pts - 1);
x = range(1,1): span(1) : range(1,2);
y = range(2,1): span(2) : range(2,2);

pop = zeros(pts * pts,2);
k = 1;
for i = 1:pts
   for j = 1:pts
      pop(k,:) = [x(i),y(j)];
      k = k + 1;
   end
end

values = feval(fcn,pop)
values = reshape(values,pts,pts);

surf(x,y,values)
shading interp
light
lighting phong
hold on
contour(x,y,values)
rotate3d
view(37,60)


%%%%%Grafica de la función
```


x 10^4
7
6
5
4
3
2
1
0
10
5
0
-5
-10
-10
-8
-6
-4
-2
0
2
4
6
8
10