Instituto Politécnico Nacional

Centro de Investigación en Computación

Secretaría de Investigación y Posgrado

TÍTULO DE LA TESIS

Diseño, optimización e implementación en FPGA de modelos y métodos de inteligencia computacional

T E S I S

QUE PARA OBTENER EL GRADO DE

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

M. en C. Prometeo Cortés Antonio

DIRECTOR (ES) DE TESIS:

Dr. Herón Molina Lozano
Dr. Ildar Batyrshin

MÉXICO, D.F. **Enero de 2016**

030

SIP-14 BIS

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 10:00 horas del día 13 del mes de enero de 2016 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Diseño, optimización e implementación en FPGA de modelos y métodos de inteligencia computacional"

Presentada por el alumno:

CORTÉS	**ANTONIO**	**PROMETEO**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

A	1	2	0	4	3	3

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Herón Molina Lozano

Dr. Ildar Batyrshin

Dr. Oleksiy Pogrebnyak

Dr. Luis Alfonso Villa Vargas

Dr. Víctor Hugo Ponce Ponce

Dr. Marco Antonio Ramírez Salinas

PRESIDENTE DEL COLEGIO DE PROFESORES

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

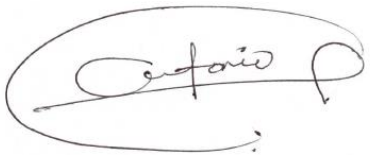
Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESIÓN DE DERECHOS

En la Ciudad de México, D.F., el día 29 de Enero del año 2016, el que suscribe Prometeo Cortés Antonio alumno del Programa de Doctorado en Ciencias de la Computación con número de registro A120433, adscrito a el Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de la tesis bajo la dirección de Dr. Herón Molina Lozano y del Dr. Ildar Batyrshin y cede los derechos del trabajo intitulado "Diseño, optimización e implementación en FPGA de modelos y métodos de inteligencia computacional", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección electrónica prometeo.cortes@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



M. en C. Prometeo Cortés Antonio

Resumen

En este trabajo se desarrollan seis metodologías para la generación de operadores paramétricos de conjunción difusa para su incorporación en un sistema de inferencia difusa con el objetivo de incrementar el grado de libertad del modelo. La característica común a resaltar de estos operadores es que son de simple implementación en hardware debido a que son descritos a partir de funciones aritméticas simples de suma, resta, multiplicación. Se presenta un diseño de generalización de las seis metodologías para crear un operador que pueda generar hasta 135 diferentes operadores.

Se desarrolla una regla de aprendizaje que sea capaz de identificar los valores óptimos de los parámetros de un sistema de inferencia difusa basado en la arquitectura ANFIS-Sugeno con operadores paramétricos de conjunción difusa y la regla de aprendizaje híbrida que combina el método del estimador de mínimos cuadrados y método del gradiente de mayor descenso y que adiciona el algoritmo de evolución diferencial para identificar a los parámetros de los operadores de conjunción difusa propuesto.

Se desarrolla un análisis comparativo en donde se muestra que al utilizar los operadores paramétricos de conjunción difusa, el modelo ANFIS, presentan mejoras relativas en la capacidad de aproximación de hasta el 80%.

Como tercera parte de este trabajo, se desarrollaron diseños lógicos para implementar en FPGAs los modelos de la inteligencia computacional expuestos son: a) un sistema de inferencia difusa Sugeno de primer orden utilizando funciones de membresía triangulares y operadores de conjunción paramétricos y b) el algoritmo de evolución diferencial con representación de punto flotante de doble precisión.

Abstract

In this work, six methodologies for generating parametric conjunction fuzzy operators to incorporate into a fuzzy inference system in order to increase the degree of freedom of a fuzzy model are developed. The common feature of these operators is that they are simple hardware implementation using simple arithmetic functions such as addition, subtraction and multiplication. Design generalization of the six methodologies is presented to create an operator for generating up to 135 different operators.

A learning rule is developed for identifying the optimum values of the parameters of a fuzzy inference system based on the architecture and hybrid learning rule of ANFIS model. Least squares estimator and steepest descent gradient methods and the addition of differential evolution algorithm to identify the parameters of the fuzzy conjunction operators proposed are used.

A comparative analysis of the traditional ANFIS model with parametric operator ANFIS shows improvements of the second up to 80% in the ability to approximate a system.

As the third part of this work, the logical designs was developed for implementation on FPGA of computational intelligence models: a) Sugeno fuzzy inference system first order using triangular membership functions, parametric conjunction operators and b) the differential evolution algorithm using floating point representation with double precision

Agradecimientos

A través de la presente deseo dar mis más sinceros agradecimientos al Concejo Nacional de Ciencia y Tecnología, CONACyT, por el apoyo profesional y económico que me otorgó durante mi estancia doctoral. De la igual manera, expreso mi agradecimiento al Centro de Investigación en Computación y al Instituto Politécnico Nacional por brindarme la oportunidad de realizar mis estudios en tan prestigiado instituto. También agradezco a la Secretaría de Investigación y Posgrado del IPN, por el apoyo otorgado a través de los proyectos SIP-20151625 y SIP-20151589.

Plasmo de manera especial mis agradamientos a los directores de este trabajo el Dr. Ildar Batyrshin y el Dr. Herón Molina que me han compartido su valioso conocimiento para la realización de este proyecto y me bridaron el apoyo tanto técnico, profesional y personal. De igual manera agradezco a los revisores de tesis por sus oportunas y apreciadas recomendaciones para el mejoramiento de este trabajo.

Dedicatoria

Les dedico este trabajo a mis padres por la motivación y formación que me han brindado día a día durante mi permanencia es esta vida. A mis familiares y amistades, y de manera especial a los que he formado en este centro de investigación, tanto a mis estimados amigos del laboratorio de MICROSE, así como a los amigos del equipo de futbol, que me apoyaron y compartieron su valioso tiempo conmigo, haciendo mi estancia doctoral más agradable.

Así mismo dedico este trabajo a mis sobrinos que son parte fundamental en mi motivación e inspiración para ser una mejor persona, esperando que este trabajo les sirva de referencia para que lleven a cabo los objetivos y metas que se tracen en sus vidas.

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1. Contexto de la investigación

La inteligencia computacional (IC) puede considerarse como el sucesor o la evolución de la inteligencia artificial para hacer frente a problemas emergentes que contengan una gran cantidad de volumen de datos o soluciones que conlleven supercómputo entre otros. Desde el punto de vista metodológico la IC utiliza enfoques heurísticos tales como los sistemas difusos, las redes neuronales y la computación evolutiva, diversas técnicas de búsqueda heurística tales como la optimización por enjambre de partículas, colonia de hormigas, fractales y teoría de caos, sistemas inmunes artificiales, ondoletas (*wavelets*), entre otras técnicas.

Las técnicas de la inteligencia computacional tales como aprendizaje, adaptación y evolución son utilizadas en el desarrollo de aplicaciones innovadoras e inteligentes.

Por otro lado, la inteligencia artificial es un concepto muy cercano o relacionado al cómputo suave surgido en la década de los 80's, la cual es una combinación de técnicas tales como las redes neuronales, la lógica difusa y los algoritmos genéticos utilizando sistemas conexionista tales como las redes neuronales, y que son utilizadas también en la inteligencia artificial y la cibernética. Por lo que la IC estudia problemas para los cuales no existen metodologías o algoritmos matemáticos efectivos, ya sea porque es imposible formularlos, o porque la implementación de estos es muy compleja en problemas de la vida real.

En este trabajo de tesis se presenta el modelo neurodifuso ANFIS (*Adaptive Neuro-Fuzzy Inference Systems*-por sus siglas en inglés), mediante el cual los modelos difusos pueden ser ajustados o entonados utilizando una red neuronal funcionalmente equivalente, permitiendo una implementación modular y adaptativa de un sistema difuso tipo Sugeno, y que mediante técnicas matemáticas híbridas de aprendizaje se ajusta el conjunto de parámetros del modelo con el objetivo de obtener una réplica o una aproximación muy cercana del sistema a modelar mediante ANFIS.

Los aportes principales de este trabajo son: a) se proponen nuevos operadores paramétricos de conjunción difusa de fácil implementación en hardware, que puedan ser integrados en un modelo ANFIS para agregar más grados de libertad al modelo; b) se propone una técnica de entrenamiento de los parámetros utilizando una heurística evolutiva; c) se presentan los diseños digitales para la implementación en FPGA de los principales módulos de un sistema de inteligencia computacional.

La inteligencia computacional en este trabajo estará guiada hacia el desarrollo de un sistema difuso específico, ANFIS-Sugeno de primer orden, que integra técnicas de redes neuronales y cómputo evolutivo para el modelado de sistemas. A continuación, se presenta una introducción de estos tres principales enfoques de la inteligencia computacional.

La lógica difusa surge como una herramienta para modelar conceptos o ideas imprecisas comunes utilizadas en la comunicación mediante el lenguaje natural o en el manejo del conocimiento de los seres humanos, que permite a los sistemas computacionales una interpretación

de estos conceptos más adecuada comparada a la forma en que la lógica clásica bi-valuada lo hace. Un ejemplo clásico en el que se observa una desventaja de uso de la lógica clásica en el modelado de sistemas, es el caso donde a una computadora es programada para realizar la clasificación de "personas altas". Utilizando lógica clásica, se puede determinar que una persona mayor o igual a 1.80 m de altura es alta, entonces todas las personas que tengan una estatura mayor o igual a 1.80 m se les clasifica como altas y las personas menores a esta altura son clasificadas como personas de estatura baja. Desde el punto de vista de la "lógica", esta clasificación es correcta, sin embargo, semánticamente tiene un problema o diferencia a como clasifican los seres humanos. Ya que para la computadora una persona de 1.799 m es una persona de estatura baja, mientras que una persona de al menos 1.80 m será clasificada como una persona alta. Algo que para el sentido común de los humanos esta clasificación no sería correcta o lógica. Lofti Zadeh propuso la teoría de conjuntos difusos para manejar de manera más adecuada este tipo de conceptos, en el cual los elementos de un conjunto contienen un valor de membresía, que indica el grado de pertenencia del elemento hacia el conjunto. Así, el conjunto difuso que clasifica a las "personas altas" puede definir personas altas con valor de pertenencia 1 (máximo valor de pertenencia) a las personas mayores e iguales de 1.80 m, y las personas que tengan estaturas más bajas de 1.80 m irán teniendo menor pertenencia al conjunto, una clasificación utilizando la teoría de conjuntos difusos válida para personas menores de 1.50 m tendrían pertenencia cero (mínimo valor de pertenencia) al conjunto de "personas altas", es decir estos elementos no pertenecen al conjunto de personas altas. Las personas que miden de 1.50 m a 1.80 m de estatura, se les puede asignar su pertenencia al conjunto de personas altas, utilizando una función creciente, que les asigne valores de pertenencia desde 0 hasta 1, estas funciones se les conoce en la literatura como funciones de pertenencia.

Los conjuntos difusos se han aplicado en el modelado de sistemas de una gran diversidad de problemas reales, teniendo como estrategia común dividir el espacio de las variables de entrada y de salida, también conocidos como universos de discurso, en un pequeño conjunto de valores llamados conjuntos difusos. Cada conjunto difuso puede representar una idea o un valor dentro de cierto rango, y que pude ser marcado mediante valores lingüísticos, los cuales son etiquetas de expresiones lingüísticas, y que son definidas por conceptos vagos o imprecisos, tales como pequeño, mediano, alto en el caso de la variable altura, o lento, moderado o rápido en el caso de la variable velocidad, si se desean tener más de tres valores lingüísticos por variables. Algunos otros ejemplos serían: muy desenfocado, poco desenfocado, más o menos enfocado, enfocado, en el caso del enfoque de las cámaras, o muy frio, algo frio, tibio, poco caliente, muy caliente para definir los valores lingüísticos de la variable temperatura.

La aplicación más popular de la teoría de los conjuntos difusos es el desarrollo de sistemas basados en reglas, a través de un conjunto reducido de reglas difusas del tipo si-entonces se describe el comportamiento del sistema, donde los valores de las variables son expresados mediante valores lingüísticos, y partir de estas reglas y hechos conocidos que son aproximados a la base de reglas, es posible generar inferencias, generando así el muy conocido razonamiento aproximado, ya que a diferencia de la lógica clásica, donde los hechos que activan a una regla deben cumplir las condiciones de la regla, en la lógica difusa el conjunto de reglas se satisfacen de manera parcial.

Los dos sistemas de inferencia difusa más utilizados son el modelo difuso tipo Mamdani y el modelo difuso tipo Sugeno. El primer modelo tiene la ventaja de ser implementado a través de un conjunto reglas completamente difusas, donde tanto los valores de los conjuntos antecedentes como consecuentes son difusos, permitiendo así describir el comportamiento del sistema totalmente mediante expresiones lingüísticas. Sin lugar a dudas, este modelo es el más adecuado a la teoría de conjuntos difusos y el cómputo con palabras, donde el conocimiento del experto puede ser plasmado en un conjunto de reglas directamente. Sin embargo, uno de los problemas que presenta este modelo es que tanto las entradas como las salidas son conjuntos difusos, lo cual evidentemente no son adecuadas en problemas donde las entradas y salidas son valores numéricos específicos, como en el casos de los sistemas de control, donde los sensores que proporcionan los datos al sistema difuso, que son discretos, y las entradas de los actuadores, es decir la salida del sistema difuso, igualmente opera con valores certeros. Mandani propone para manejar este problema una etapa de *fuzificación* y otra etapa de *desfuzificación* dentro de los sistemas difusos. No obstante, este modelo tiene como talón de Aquiles tanto el ajuste fino de los parámetros de las funciones de membresía que describen a los valores lingüísticos, así como saber que método de desfuzificación para obtener un valor certero utilizar al momento de querer replicar el comportamiento de un sistema, además este modelo carece de algoritmo de aprendizaje autónomo, y no puede considerarse como un sistema adaptativo, a pesar de que este puede modelarse con funciones paramétricas, ya que estas regularmente son ajustadas a prueba y error.

Por otro lado, el sistema de inferencia difusa tipo Sugeno, trata de solucionar el problema de entrenamiento de los modelos difusos. Propone un modelo en donde los antecedentes de las reglas sean descritos mediante valores lingüísticos y utiliza el mecanismo de razonamiento aproximado, pero a diferencia del modelo Mandani, en este modelo la parte consecuente de las reglas son descritas mediante funciones matemáticas, en su publicación, Sugeno et al. [**1**], describen el modelo Sugeno de primer orden, donde las funciones consecuentes son modeladas con funciones de primer orden, es decir, funciones lineales o *hiperplanares*. Sin embargo, deja abierta la posibilidad de que los consecuentes pueden implementarse mediante cualquier tipo de función matemática que mejor se ajuste al problema a solucionar. Pero el modelo de primer orden tiene como ventaja principal, que el ajuste de los parámetros consecuentes pueda realizarse utilizando cualquier método de identificación lineal, tales como el estimador de mínimos cuadrados, que encuentra los valores óptimos de los parámetros lineales en una sola iteración.

Con el objetivo de establecer una metodología de entrenamiento de los sistemas difusos que ajuste los parámetros de los conjuntos difusos antecedentes y consecuentes en función de datos de las entradas y las salidas, Jy Shi Jang propone la arquitectura y entrenamiento del modelo ANFIS, donde básicamente se estable la arquitectura de una red neuronal dividida por capas para implementar a los modelos de inferencia difusa, donde los nodos de algunas de las capas de la red son adaptativos, permitiendo el ajuste de los parámetros mediante algoritmos de entrenamiento. Aunque esta arquitectura permite la implementación de redes neuronales equivalentes de cualquier modelo difuso, la arquitectura ANFIS mostró ser mejor para el modelo Sugeno de primer orden, debido al entrenamiento híbrido de los parámetros de la red.

Jang en [**2**], especifica el método de entrenamiento híbrido para el sistema Sugeno de primer orden que divide el conjunto de parámetros en dos partes, la parte de los parámetros antecedentes que describen a los conjuntos difusos en los antecedentes de las reglas, y la parte de los parámetros consecuentes que especifican a las funciones lineales que describen los consecuentes de las reglas y que son lineales con respecto a los parámetros de identificación. En la regla de aprendizaje híbrida se establece la identificación de los parámetros consecuentes mediante algún método lineal y la identificación de los parámetros antecedentes mediante algún método del descenso del gradiente. En [**2**] se muestra que el entrenamiento híbrido, presenta una mejor aproximación para funciones matemáticas de prueba y diversas aplicaciones comparada con los métodos de entrenamiento clásicos de redes neuronales tales como, la retropropagación (*back-propagation*) y la retropropagación rápida (*fast-back-propagation*). Además, se muestra la potencialidad de aproximación inicial de parámetros consecuentes, es decir, el primer ajuste de los parámetros consecuentes, que permite concluir que uso del estimador de mínimos cuadrados es crucial para la aproximación del sistema.

Sin embargo, en la publicación de ANFIS, Jang recibe la crítica de que los parámetros antecedentes del modelo Sugeno no deberían ajustarse, ya que estos estos especifican el conocimiento del experto. Por lo que el ajuste en de estos parámetros, llevaría a la pérdida del conocimiento del experto acerca del sistema a modelar. Jang discute esta observación, justificando que para sistemas en donde se dispone de un gran número de datos, el entrenamiento fino de los parámetros antecedentes, no sólo es deseable, sino necesario, y acepta que para sistemas con unos pocos datos de entrenamiento, el valor de los parámetros establecidos por los expertos, toman un papel crucial en el modelado. En el capítulo 2, de este trabajo se analizarán los resultados tanto de aproximación del ANFIS, así como de la interpretación semántica del ajuste de ANFIS, en la aproximación de la función *sinc* que servirá como función objetivo.

En su trabajo Jang deja abierto entre otros temas de investigación, el remplazar los nodos fijos que implementan a los operadores de conjunción difusa producto, por operadores de conjunción difusa paramétricos. Además de la posibilidad de establecer una metodología de entrenamiento que ajuste los parámetros de estos operadores. En la literatura existe una gran diversidad de operadores paramétricos de conjunción difusa, definidos por funciones t-norma o *conjunctores*[1], donde la mayoría de estos tiene como característica común, que son definidas por operaciones de multiplicación, exponenciales o incluso operaciones logarítmicas. Sin embargo, actualmente no existe un análisis y/o metodología que establezca la capacidad de aproximación de un sistema de inferencia difusa con operadores conjunción paramétricos. Por lo que es difícil motivar el uso de estos operadores.

Aunque en la literatura de sistemas difusos se han propuesto diversos modelos difusos que contienen operadores paramétricos de conjunción [**3**,**4**,**5**,**6**], se han realizados análisis de la influencia en las propiedades de sensibilidad y estabilidad de sistemas de control difuso al ajustar los parámetros de los operadores de conjunción paramétricos [**7**], e incluso algunas propuestas del

[1] El término *conjunctor* es una función matemática para la implementación de conjunciones que será explicada en el capítulo 2.

entrenamiento de operadores de conjunción paramétrica simples [**8**], hasta el día de hoy no existe una metodología que determine el poder de aproximación de sistemas al realizar el ajuste de los parámetros de los operadores de conjunción de un modelo difuso, que muestre ventajas o por lo menos se acerque al entrenamiento híbrido de ANFIS.

Ildar Batyrshin, *et al.* han trabajado en propuestas e implementación de operadores de conjunción y t-normas paramétricas (ver las referencias [**8**,**9**,**10**,**11**,**12**,**13**,**14**,**15**,**16**]) basadas en t-normas simples para su eficiente implementación en hardware, específicamente para su implementación de diseños digitales en FPGA's. En el capítulo 3, se explican seis metodologías para la generación de t-normas y conjunctores paramétricos propuestas en desarrollo de este trabajo de tesis. Además, en el capítulo 4 se describe una metodología de entrenamiento del modelo ANFIS-Sugeno de primer orden con operadores paramétricos para ajustar los parámetros de los operadores propuestos, utilizando como base el entrenamiento híbrido de ANFIS. Se mostrará, más adelante, la capacidad de aproximación del modelo al ajustar los parámetros de los operadores de conjunción mediante una búsqueda heurística, y el entrenamiento de los parámetros lineales consecuentes utilizando el método de estimador de mínimos cuadrados recursivo. Y que como demostró Jang, es el pilar del poder de aproximación de esta metodología, utilizando como función objetivo la función de dos variables *sinc*, y una función *multisinc* que se genera de la suma de las funciones recorridas de la función *sinc*.

Finalmente, en este trabajo se mostrará: a) el diseño e implementación digital en FPGA de los módulos del operador paramétrico de conjunción difusa que unifica las seis metodologías de generación de t-normas y conjunctores paramétricos; b) el diseño digital de un sistema de inferencia difusa Sugeno de primer orden y; c) La implementación en FPGA del algoritmo de evolución diferencial con representación de punto flotante de doble precisión para su futura implementación en el entrenamiento de un sistema difuso.

1.2. Planteamiento del problema

En el presente trabajo, se desea desarrollar un algoritmo de entrenamiento para la optimización de sistemas neurodifusos. Basado en el modelo conexionista difuso ANFIS se pretende optimizar un sistema Sugeno de primer orden con operadores paramétricos de conjunción, el cual contiene un conjunto de parámetros que pueden clasificarse en tres categorías, es decir, los dos subconjuntos de parámetros que tradicionalmente incluye ANFIS que son: a) los parámetros antecedentes, que establecen el comportamiento de las funciones de membresía; b) los parámetros consecuentes, que definen las funciones lineales que determinan las funciones consecuentes del conjunto de reglas difusas y; c) un conjunto de parámetros que llamaremos, parámetros de operadores (de conjunción) que determinan los parámetros que definen las operaciones de conjunción paramétrica en los antecedentes de las reglas difusas. Con el objetivo de proponer un modelo neurodifuso que tenga un mayor grado de libertad de ajuste y permita: a) una mejor capacidad de aproximación con relación a los modelos ANFIS tradicionales, esto a partir de un modelo ya entrenado de ANFIS, en donde posteriormente sean identificados los parámetros de los operadores, mediante un proceso de dos pasos, en donde los parámetros antecedentes ya fijados por el entrenamiento de ANFIS tradicional, no serán ya ajustados en esta etapa, y los parámetros consecuentes se irán ajustando en función de

los valores determinados por la identificación de parámetros de los operadores utilizando el algoritmo de evolución diferencial; b) con base en la recomendación hecha por el revisor del artículo de ANFIS [**17**], en donde se sugiere que no se realice entrenamiento en los parámetros antecedentes, y con base en la observación de Jang, que dice que si no se entrenan los parámetros antecedentes de un sistema Sugeno de primer orden, ANFIS se convierte en una red lineal, se plantea un algoritmo basado en el supuesto de que los parámetros que definen a las funciones de membresía de los conjuntos antecedentes de las reglas difusas contiene un conocimiento que no puede ser modificado; se desarrolló un algoritmo de entrenamiento donde puedan ser ajustados tanto los parámetros lineales o consecuentes y los parámetros de los operadores de conjunción.

Para realizar esta propuesta de tesis el problema a tratar será dividido en tres partes principales:

1. Se proponen un conjunto de operadores paramétricos de conjunción difusa, generados mediante funciones t-normas o conjunctores básicas, es decir, operadores que puedan ser implementadas con operaciones lógicas simples, tales como sumas, restas, multiplexores y comparadores, con el objetivo de que estos operadores de conjunción (o también llamados operadores de intersección) presenten una complejidad matemática simple que conlleve a un bajo costo computacional, y que presenten considerable ventajas en su implementación y diseño digital en hardware, específicamente en FPGA's.

2. Desarrollar un algoritmo de aprendizaje, para identificar a los parámetros antecedentes, consecuentes y de operadores de conjunción, mediante técnicas de aprendizaje automático tales como el gradiente descendiente (GD), el estimador de mínimos cuadrados (LSE) y el algoritmo de evolución diferencial (DEA), y desarrollar un análisis comparativo de la capacidad de aproximación del modelo propuesto con trabajos anteriormente presentados en la literatura, incluyendo el ANFIS tradicional.

3. Diseñar e implementar en hardware un modelo o método de cada uno de los tres principales enfoques de un sistema de inteligencia computacional, que son: a) un modelo difuso de primer orden ; b) una arquitectura conexionista y; c) un algoritmo de evolución diferencial.

1.3. Justificación

La arquitectura ANFIS, propuesta por Jang en 1997, para implementar un sistema de inferencia difusa utilizando una arquitectura conexionista de redes neuronales, específicamente, redes adaptativas, demostró desde su propuesta ser un modelo con considerables ventajas en el proceso de entrenamiento comparado con los enfoques que existían en ese momento. En su libro Jang, presentó un amplio estudio, no sólo de los fundamentos teóricos del modelo, sino que además, presentó una gran variedad de aplicaciones del modelado de sistemas en áreas tales como, el control, reconocimiento de patrones, procesamiento digital de señales, entre otros.

Actualmente un gran número de aplicaciones en áreas como, control, medicina, agricultura, entre otros, utilizan la arquitectura ANFIS para modelar sus aplicaciones, incluso este modelo es

utilizado en problemas emergentes de grandes datos (Big Data) y soluciones de cómputo en la nube (Cloud Computing). Para mostrar la importancia actual del modelo ANFIS en la inteligencia computacional, podemos encontrar en el motor de búsqueda de la base de datos *scopus*, 442 artículos publicados en el año 2015 que llevan la palabra ANFIS en su título o resumen, y 5289 del año 2000 a la actualidad. Además, de acuerdo al motor de búsqueda *Google Académico* el articulo ANFIS [**17**] y el libro de texto [**2**] tienen 10446 y 7231 citas respectivamente.

1.4. Antecedentes

A continuación se presenta una breve introducción acerca de los fundamentos teóricos del modelo ANFIS, que es parte fundamental de este trabajo. El sistema neuro-difuso, o sistema de inferencia difusa con redes neuronales adaptativas, es un modelo difuso, que utiliza una arquitectura de redes neurales adaptativas o generalizadas. A pesar que este modelo puede tener una gran número de variantes, para fines demostrativos se presenta en [**2**,**17**] una arquitectura de 5 capas para procesar el conjunto de reglas de un modelo de inferencia difuso. Particularmente, se trabajó y se analizó el modelo Sugeno de primer orden, ya que en este modelo los consecuentes de las reglas difusas son caracterizadas mediante funciones paramétricas lineales, siendo esta la parte más fundamental e importante del éxito de modelo ANFIS en el proceso de entrenamiento.

Con relación a la arquitectura de ANFIS; en la capa 1, se tiene que las premisas antecedentes se definieron mediante conjuntos difusos caracterizados con funciones de membresía conocidas como función campana de Cauchy, o función campana generalizada, que contienen tres parámetros para determinar su forma, además de ser funciones continuas diferenciables. Por lo que en el proceso de entrenamiento, estos parámetros son identificados utilizando algún método del descenso del gradiente.

En la capa 2 se definen a los operadores de intersección difusos mediante la t-norma producto algebraico, en la cual cada nodo de esta capa realiza la multiplicación de las variables de entrada. Jang (en [**2**]) plantea el problema de la posibilidad de cambiar la t-norma producto algebraico, ya sea por alguna t-norma básica, o por t-normas paramétricas. Este problema es finalmente el punto central del desarrollo del presenté trabajo de tesis. Sólo para poner en contexto el problema que se está enfrentando, podemos mencionar que al cambiar el operador de intersección del modelo Sugeno de primer orden, el proceso de entrenamiento también cambia, y por lo tanto, el problema de la identificación de los parámetros antecedentes puede llegar a complicarse, de tal manera que el uso del método del descenso del gradiente no sea práctico o funcional.

La capa 3 implementa nodos fijos para normalizar las fuerzas de disparo, resultantes de la capa dos. Y que semánticamente representa el valor de verdad normalizado de las premisas antecedentes de cada una de las reglas difusas.

La capa 4 consta de los nodos que implementan a las premisas consecuentes de las reglas difusas del sistema Sugeno de primer orden, es decir, se implementan funciones lineales y multiplicadores que realizan la multiplicación de las entradas $w_i \times z_i$. Los componentes son identificados utilizando

algún método de estimación de mínimos cuadrados. Donde los coeficientes de la matriz de covarianza son obtenidos evaluando el valor de disparo de cada regla, de acuerdo a las pares de datos del conjunto de entrenamiento.

Finalmente, la capa número 5 contiene únicamente un único nodo que calcula la suma ponderada de cada regla para determinar la salida total del modelo.

Una de las ventajas que tiene el modelo ANFIS, a diferencia de la mayoría de los enfoques de redes neuronales, en donde todos nodos de la red son definidos mediante la misma función, como por ejemplo por funciones lineales en la red neuronal *Adaline*, o por la composición de funciones lineales y funciones *gausianas* en la red neuronal *Perceptron* multicapa, por mencionar los más comunes, es que ANFIS, permite definir a los nodos de diferente manera en cada capa, con el objetivo de mantener el conocimiento adquirido en los modelos difusos.

Una de las características a subrayar de ANFIS, es la capacidad de modelar fácilmente sistemas no lineales reales, que regularmente presentan cambios suaves en su superficie, mediante un pequeño conjunto de reglas que describen su comportamiento. Los sistemas ANFIS han mostrados ser mejores *aproximadores* de funciones que los modelos lineales tales como los modelos estadísticos, de regresión, entre otros. El costo de utilizar ANFIS consiste en que necesita una metodología iterativa para entrenar al modelo, caso contrario de los modelos lineales, los cuales es posible encontrar la solución óptima en un sólo paso.

El sistema ANFIS-Sugeno de primer orden cuando se compara con los sistemas conexionistas, tiene dos ventajas principales, por un lado, como ya se mencionó, es que se adquiere el conocimiento de los expertos de manera heurística, basándose en la teoría de la lógica difusa, a diferencia de los sistemas conexionistas que el conociendo adquirido por la red es interpretado como una caja negra. Por otro lado, en el proceso de entrenamiento, ANFIS clasifica los parámetros de la red en dos partes, y realiza la identificación de los parámetros consecuentes mediante métodos de mínimos cuadrados, es decir, se hace una aproximación lineal del modelo, utilizando valores iniciales preestablecidos en parámetros antecedentes. Una vez que se realiza la aproximación lineal, se entrenan o identifican los parámetros antecedentes fijando los parámetros consecuentes recién ajustados. Este proceso de identificación se repite, con el objetivo de lograr una mejor aproximación. Con esta metodología, el modelo ANFIS reduce el espacio de búsqueda de manera eficiente.

A pesar de que desde su introducción ANFIS mostró ser un modelo con muchas ventajas, Jang evidenció varias tareas a modificar del modelo para que hipotéticamente la capacidad de aproximación de ANFIS pueda mejorar, las cuales se enuncian a continuación:

1. Con ANFIS es posible definir la arquitectura conexionista de otros modelos difusos, tales como sistemas difuso tipo Tsukamoto o Mamdani, o incluso cambiar el orden del modelo Sugeno. Sin embargo, el entrenamiento de estos modelos, indiscutiblemente debe cambiar, siendo muy complicada la identificación de parámetros lineales donde puedan aplicarse métodos lineales, lo que implicaría que ANFIS disminuya la capacidad de aproximación y que el tiempo de

entrenamiento aumente. Por lo tanto, este trabajo, se concentrará el modelo Sugeno de primer orden.

2. Pueden implementarse cualquier tipo de funciones de membresía difusa, es decir, cambiar las funciones de membresía campana generalizada, por alguna otra función paramétrica ampliamente estudiadas en la lógica difusa, tales como funciones gaussianas, triangulares y trapezoidales. Sin embargo, debe considerarse que en el entrenamiento de las funciones no continuas, tales como las funciones triangulares o trapezoidales (también conocidas como funciones a pedazos), se deben considerar las diferentes expresiones que las definen de manera separada, en el caso de que se desee entrenar a sus parámetros con algoritmos del descenso del gradiente.

3. Como en el caso de las funciones de membresía, la operación t-norma producto algebráico que definen al operador de conjunción establecido en los antecedentes de las reglas, pueden cambiarse por otros operadores t-normas, tales como los operadores mínimo, *Lukaseiwicz* y producto drástico, o mejor aún t-normas paramétricas tales como las presentadas en este trabajo. Sin embargo, debemos hacer notar que este cambio no es trivial, principalmente en el proceso de entrenamiento del modelo, ya que las t-normas antes mencionadas no son continuas, por lo que la determinación del gradiente para ajustar los parámetros de funciones de membresía cambia en el proceso de entrenamiento por *retropropagación*. Lo que dificulta a ANFIS tener una propiedad de entrenamiento generalizado.

   Se hace un paréntesis en este apartado por que será el tema principal de nuestro trabajo de investigación. Podemos enumerar dos rutas principales para enfrentar el problema. La primera es utilizar normas paramétricas que sean continuas, para que durante el entrenamiento de ANFIS se pueda entrenar a los conjunto antecedentes utilizando el método de gradiente descendiente, sin embargo, al momento de utilizar las funciones paramétricas continuas se presentan mucho gasto computacional en la implementación en hardware según [**12**]. Otro método propuesto, y ampliamente trabajado por el grupo investigación del laboratorio de MICROSE, es la definición de t-normas o conjunctores paramétricas, basadas en t-normas básicas, que son definidas mediante operaciones simples de la lógica digital, tales como sumadores, restadores, comparadores y multiplexores.

   La definición de estas conjunciones difusas definidas por t-normas o conjunctores, reduce la complejidad matemática, y por ende el costo computacional requeridas en las primeras propuestas de t-normas paramétricas. Sin embargo, la desventaja de utilizar dichas funciones es que éstas no son continuas, por lo que hace prácticamente imposible, el entrenamiento de los parámetros antecedentes de ANFIS, además de los parámetros de los operadores de conjunción definidas con estas t-normas mediante los métodos del gradiente.

   Un método generalizado para la identificación de los parámetros que determinan las t-normas o conjunctores, es utilizando técnicas heurísticas de minimización basadas el comportamiento de la naturaleza como pueden ser los algoritmos evolutivos, agrupación por cúmulos, entre otros.

4. La determinación de otros métodos de agregación en los sistemas difusos también es posible, sin embargo, como este trabajo se limita a modelos de sistemas difusos Sugeno de primer orden, por lo que no se considera este punto en la mejora del modelo propuesto.

1.5. Objetivos

1.5.1. Objetivo general

Diseño, optimización e implementación en FPGA de modelos y métodos de inteligencia computacional basados en sistemas difusos.

1.5.2. Objetivos particulares

1. Proponer y diseñar operadores de intersección difusa mediantes funciones t-normas y conjunctores paramétricas definidas a partir de funciones lógicas simples para su entrenamiento e implementación en hardware.

2. Realizar la metodología de identificación de los parámetros de ANFIS Sugeno de primer orden con operadores de conjunción paramétrica, utilizando los métodos de optimización tradicionales y el algoritmo de evolución diferencial

   a) Se modela una arquitectura ANFIS Sugeno de primer orden tradicional utilizando funciones de membresía tipo campana generalizada, junto con operadores difusos t-norma producto algebraico y funciones consecuentes lineales. Entrenar a ANFIS mediante una regla híbrida utilizando el método de gradiente descendiente y el estimador de mínimos cuadros, para ajustar los parámetros antecedentes y consecuentes respectivamente.

   b) Se sustituye la arquitectura ANFIS tradicional por la arquitectura ANFIS Sugeno de primer orden con operadores de conjunción paramétricos y durante esta etapa de entrenamiento, los nodos que determinan los conjuntos difusos, se consideran fijos, es decir, los valores de los parámetros antecedentes se consideran constantes durante toda esta etapa. El proceso de identificación de los parámetros de los operadores de conjunción y los parámetros consecuentes lineales, se realiza en dos fases: i) considerando a los parámetros consecuentes fijos, se identifican los parámetros de operadores difusos mediante el algoritmo de evolución diferencial; ii) se consideran los parámetros de los operadores fijos, y se identifican los parámetros consecuentes utilizando el método de entrenamiento de mínimos cuadrados recursivos.

   c) Se realizan las variantes siguientes del entrenamiento: i) utilizando el mismo operador, y los mismos parámetros para todas las reglas de ANFIS; ii) se estudiará la posibilidad de utilizar diferentes operadores y diferentes parámetros de ANFIS; iii) sólo hacer el

entrenamiento únicamente como se marca en el inciso b), que incluye las fases i) y ii), de este inciso.

3. Diseñar e implementar en FPGA la familia de operadores de conjunción y t-normas propuestas.

4. Diseñar e implementar en un FPGA del modelo difuso Sugeno de primer orden con operadores de conjunción paramétrica, que se defina con funciones de membresía triangulares, operadores paramétricos y funciones lineales.

5. Diseñar e implementar un algoritmo de evolución diferencial en FPGA.

1.6. Organización del trabajo

En el capítulo dos se presentan las definiciones y fundamentos básicos de las tres principales herramientas del desarrollo de sistemas de cómputo inteligente que son: los sistemas difusos, las redes neuronales y el algoritmo de evolución diferencial. Se realiza un estudio especial al sistema neurodifuso, el cual es un modelo híbrido que utiliza el enfoque de lógica difusa y las redes neuronales. Además, se presenta el algoritmo de evolución diferencial, el cual se utilizará para entrenar los parámetros de los operadores presentados en el capítulo tres.

En el capítulo tres se presentarán las metodologías de generación de familias de conjunciones difusas generadas de funciones conjunctores y t-normas paramétricos, presentando los fundamentos matemáticos de estas metodologías. Se muestra la diferencia entre estas familias. Finalmente, se presentará el diseño en hardware de un operador que puede generar 6 diferentes familias de operadores de conjunción paramétricos.

En el capítulo cuatro se presenta la metodología de entrenamiento de ANFIS Sugeno de primer orden con operadores paramétricos de conjunción. Además, se desarrolla la metodología para el entrenamiento de los operadores paramétricos que mejorara la capacidad de aproximación de la arquitectura ANFIS tradicional. A este capítulo se le anexa una carpeta que contiene las imágenes de los resultados presentados en el ejemplo 1, que se incluye en el disco compacto y que puede conseguirse en la biblioteca del Centro de Investigación en Computación, en la biblioteca central del Instituto Politécnico Nacional o solicitarse al correo electrónico prometeo.cortes@gmail.com.

En el capítulo 5 son presentados los diseños lógicos para la implementación en un FPGA del Sistema difuso Sugeno de primer orden con operadores de conjunción paramétrica, se detalla la implementación del módulo del operador paramétrico de conjunción difusa generalizado que puede configurarse para implementar a las seis metodologías generadoras de las familias de conjunción presentadas en el capítulo 3, y finalmente, se presenta la implementación en FPGA del algoritmo de evolución diferencial utilizando una máquina de estados finitos.

Finalmente, son presentadas las conclusiones, los productos de investigación generados de esta tesis y los trabajos futuros.

1.7. Referencias

1. Sugeno M., K.: Structure identification of fuzzy model 28. Elsevier (1988)
2. Jang J.-S.R., S.: Neuro-fuzzy and soft computing (1997).
3. Jenei, S.: How to construct left-continuous triangular norms - State of the art. Fuzzy Sets and Systems 143(1), 27-45 (2004).
4. Mayor G., M.: Additive generators of discrete conjunctive aggregation operations. IEEE Transactions on Fuzzy Systems 15(6), 1046-1052 (2007).
5. László, L.: A note on Hamacher-operators. Advances in Soft Computing, Intelligent Robotics and Control, 159-163 (2014).
6. Mayor G., T.: Triangular norms on discrete settings. (2005).
7. Koprinkova-Hristova, P. D.: Fuzzy operations' parameters versus membership functions' parameters influence on fuzzy control systems properties., vol. 1, pp.219-224 (2004).
8. Henández-Zavala A., B.: On generation and FPGA implementation of digital fuzzy parametric conjunctions. Applied and Computational Mathematics 11(2), 150-164 (2012).
9. Hernández-Zavala A.H., B.: Conjunction and disjunction operations for digital fuzzy hardware. Applied Soft Computing Journal 13(7), 3248-3258 (2013).
10. Tellez A., M.-L.: Parametric type-2 fuzzy logic systems. E.P. Dadios (Ed.), Fuzzy Logic-Algorithms, Techniques and Implementations, 97-114 (2012).
11. Batyrshin I., R.: Parametric t-norms in reconfigurable digital fuzzy systems., Berkeley, CA, USA (August 2012)
12. Batyrshin I., R.: On the monotone sum of basic t-norms in the construction of parametric families of digital conjunctors for fuzzy systems with reconfigurable logic. Knowledge-Based Systems 38, 27-36 (January 2013)
13. Cortés-Antonio P., B.: FPGA implementation of (p)-monotone sum of basic t-norms., Barcelona, Spain (August 2010)
14. Cortés-Antonio P., B.: FPGA implementation of fuzzy system with parametric membership functions and parametric conjunctions. Lecture Notes in Computer Science (9th Mexican International Conference on Artificial Intelligence) 6438 LNAI(PART 2 Advances in Soft Computing), 487-499 (2010)
15. Cortés-Antonio P., B.: Hardware Design of Digital Parametric Conjunctors and t-Norms. International Journal of Fuzzy Systems 17(4), 559-576 (2015)
16. Cortés-Antonio P., R.-G.: Design and Implementation of Differential Evolution Algorithm on FPGA for Double-Precision Floating-Point Representation. Acta Polytechnica Hungarica 11(4), 139-153 (2014)
17. Jang, J.-S.: ANFIS: adaptive-network-based fuzzy inference system. Systems, Man and Cybernetics, IEEE Transactions on 23(3), 665-685 (1993)
18. Zhang X., H.: A fuzzy logic system based on Schweizer-Sklar t-norm. Science in China, Series F: Information Sciences 49(2), 175-188 (2006).

19. Rudas I.J., B.: Digital fuzzy parametric conjunctions for hardware implementation of fuzzy systems., pp.157-166 (2009) .
20. Król, A.: Dependencies between fuzzy conjunctions and implications., vol. 1, pp.230-237 (2011).
21. Korytkowski, M., Scherer, R.: Modular neuro-fuzzy systems based on generalized parametric triangular norms. Parallel Processing and Applied Mathematics, 332-339 (2010).
22. Cortés-Antonio P., B.: FPGA implementation of (p, I-p) – Monotone Sum of Basic t-norm., san francisco, CA, USA (May 2011)
23. Cortés-Antonio P., B.: FPGA Iimplementation of Parametric Fuzzy Mamdani Systems., Prague, Czech Republic (August 2010)
24. Cortes Antonio P., B.: FPGA implementation of fuzzy Mamdani system with parametric conjunctions generated by monotone sum of basic t-norms. Polibits 44, 53-58 (2011)

Capítulo 2. Fundamentos de la Inteligencia Computacional

2.1. Introducción

A lo largo de este capítulo se presentan los fundamentos de la inteligencia computacional que incluye los conceptos de: a) la teoría de conjuntos difusos y lógica difusa; b) las redes neuronales y; c) las metaheurísticas.

Se presentan los fundamentos teóricos y definiciones que se estarán utilizando a lo largo de la presente tesis. En la primera parte se presentan las definiciones formales de los conjuntos difusos, y de las funciones de membresía tipo triangulo y campana generalizada. También, se presentan las definiciones formales de las funciones t-norma y *conjunctores* difusas notando la principal diferencia que existen entre estas dos funciones. Se presentan los operadores t-normas básicos producto drástico, *Lukasiewicz*, producto algebraico, n*ilpotente* y mínimo que han sido propuestos y utilizados ampliamente en el modelado de sistemas difusos. El sistema difuso tipo Sugeno será presentado, enunciando sus principales características y ventajas al utilizar este modelo comparado con otras propuestas en la literatura, tales como los sistemas difusos tipo Mamdani.

En la segunda parte se presentan las definiciones fundamentales de las redes neuronales para posteriormente describir la arquitectura ANFIS y su método de entrenamiento híbrido, que implementa las metodologías de aprendizaje de gradiente descendiente y el estimador de mínimos cuadrados para la identificación del sistema. Se incluye un análisis de los resultados al entrenar un modelo ANFIS, tanto de la aproximación de funciones, como de la interpretación de los valores finales de los parámetros que describen a los conjuntos difusos, cuando ANFIS replica dos funciones objetivos no lineales de dos y tres variables de entrada, para un diferente número de conjuntos difusos por variable de entrada y los dos tipos de funciones de membresía de estos, con el objetivos de ver el resultado del entrenamiento de ANFIS al utilizar diferentes variantes durante dicho entrenamiento.

Finalmente, se presentará una breve revisión acerca de los algoritmos de optimización utilizando metaheurísticas haciendo énfasis en el algoritmo de evolución diferencial

2.2. Teoría de conjuntos difusos y lógica difusa

Antes de dar las definiciones formales de la teoría de conjuntos y la lógica difusa se presentan los conceptos de variable lingüística y valores lingüísticos, los cuales formalizan la teoría de la lógica difusa.

2.2.1. Variable lingüística y valores lingüísticos

Una variable lingüística, es una variable que al igual que en todos los enfoques de modelado y análisis de sistemas es una entidad que se utiliza para representar una cantidad, un aspecto, una

dimensión, la propiedad de un objeto, o un fenómeno, que presenta variaciones en mediciones temporales sucesivas. A diferencia de otros enfoques, las variables lingüísticas reciben valores lingüísticos, en vez de valores numéricos, que no son más que expresiones lingüísticas cotidianas que se utilizan en el acervo diario de los humanos. Por ejemplo, la variable temperatura, en lugar de tener un valor de *temperatura* = 10ºC, se le asigna el valor lingüístico *temperatura = fría*.

Este enfoque hace que, por un lado, las variables lingüísticas sean más adecuadas en el modelado de sistemas de la vida real, donde su comportamiento está fuertemente influenciado por juicios, percepciones, emociones y la subjetividad del ser humano (ver el principio de incompatibilidad en [**1**]), permitiendo describir el comportamiento de un sistema a modelar, utilizando sentencias simples mediante reglas de inferencia lógica del tipo si-entonces.

Por otro lado, la asignación de valores lingüísticos a las variables, permite que éstas tengan un conjunto pequeños de valores como dominio, regularmente, y sin que esto sea una condición, en la práctica a una variable lingüística se asignan entre 2 a 7 valores lingüísticos.

Por ejemplo, el dominio de la variable de temperatura del agua y el nivel del agua para tres, cuatro y cinco valores lingüísticos puede ser el siguiente:

$$Temperatura_{mf_3} = \{fría, templada, caliente\}$$
$$Temperatura_{mf4} = \{fría, tibia, caliente, irviendo\ \}$$
$$Temperatura_{mf5} = \{muy\ negativa, negativa\ , cero, positiva, muy\ positiva\}$$
$$NivelAgua_{mf3} = \{bajo, medio, alto\}$$
$$NivelAgua_{mf4} = \{vacio, bajo, alto, lleno\}$$
$$NivelAgua_{mf5} = \{vacío, bajo, medio, alto, lleno\}$$

El número de valores lingüísticos depende del sistema específico y del nivel de abstracción con que se quiera describir a este. Otras variables lingüísticas comunes utilizadas en aplicaciones industriales, de medicina y biología pueden ser: la presión de gas; electricidad; frecuencia; consumo de diésel; frecuencia cardiaca; presión sanguínea; nivel de nutriente de un suelo; entre otras.

La forma, ubicación, distribución, descripción de los valores lingüísticos de una variable lingüística se realiza mediante conjuntos difusos y funciones de membresía, las cuales se definirán a continuación.

2.2.2. Conjunto difuso

Un conjunto difuso extiende o generaliza el concepto de conjunto de la teoría de conjuntos clásicos. Permitiendo que cada elemento del universo, tenga un valor de pertenencia al conjunto, el cual está normalizado al intervalo unitario [0, 1], siendo 0 una pertenencia nula al conjunto o no

perFigura 2. 1tenencia, y 1 representa la máxima pertenencia que un elemento puede tener a un conjunto. Por lo tanto, un conjunto difuso se define mediante un par ordenado, donde uno de los elementos define al elemento de discurso y el otro elemento define el valor de pertenencia al conjunto. La definición formal de un conjunto difuso es como sigue.

Definición

Sea *X* una colección de objetos denotada generalmente por *x*, entonces un conjunto difuso *A* en *X* está definido como el conjunto de pares ordenados siguiente:

$$A = \{(x, \mu_A(x)) \mid x \in X \text{ y } \mu_A(x) : U \to [0,1]\},$$

donde $\mu_A(x)$ indica el valor de pertenencia del elemento *x* al conjunto difuso *A*, el cual puede ser determinado por una función de membresía.

Esta definición generaliza el concepto de conjunto clásicos cuando el intervalo unitario que asigna las pertenencias es remplazado por el conjunto $\{0,1\}$. Los conjuntos difusos, pueden ser definidos para espacios continuos o discretos, las notaciones comunes más utilizadas en la literatura para definir los conjuntos difusos es la siguiente:

Espacio discreto

$$A = \sum_{x_i \in X} \frac{\mu_A(x_i)}{x_i}.$$

Espacio continuo

$$A = \int_X \frac{\mu_A(x)}{x}$$

Donde el signo de suma y el signo de integral establecen la unión de los grados de membresía; el símbolo "/" establece un marcador y no implica la división.

2.2.3. Función de membresía

Una función de membresía se utiliza para describir a los conjuntos difusos asignando a cada elemento de *X* un grado de membresía o pertenecía que se encuentra entre 0 y 1, es decir, dentro del espacio difuso definido en el intervalo [0, 1] mediante una función matemática. Estas funciones matemáticas regularmente satisfacen ciertas características comunes que son: a) funciones parametrizables; b) están normalizadas, es decir, al menos uno de los elementos tiene un valor de membresía igual a 1; c) son convexas y; d) tiene una forma abierta a la izquierda, abierta a la derecha o son cerradas, es decir, son funciones crecientes, decrecientes o crecientes-decrecientes. Para ver a más detalle estas propiedades revisar [**2**,**3**,**4**].

Existen diversas funciones matemáticas comunes que se utilizan como funciones de membresía para caracterizar a los conjuntos difusos, entre ellas se pueden mencionar las siguientes: a) triángulo; b) trapezoidal; c) gaussiana; d) campana generalizada; e) *singletone*. Sin embargo, en este trabajo sólo utilizarán las funciones triángulo y campana generalizada.

Función de membresía triangular

La función triángulo está definida por tres parámetros {a, b, c}, que dividen el universo de discurso en cuatro partes, y se define a continuación:

$$\mu_{Triangular}(x; a, b, c) = \begin{cases} 0 & x \leq a \\ \dfrac{x-a}{b-a} & a \leq x \leq b \\ \dfrac{c-x}{c-b} & b \leq x \leq c \\ 0 & c \leq x \end{cases}$$

Una manera alternativa de definir la función triangular utilizando operadores de conjuntos difusos máximo y mínimo es:

$$\mu_{Triangular}(x; a, b, c) = \max(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0)$$

La figura 2.1 muestra un ejemplo de la función triangular con parámetros (x; a = 40, b = 80, c = 90), donde los parámetros a = 40 y b = 80 se utilizan para establecer una función lineal creciente, que asigna valores de membresía desde 0 hasta 1 a los elementos que pertenecen al intervalo [40, 80]. De manera similar, los valores de membresía para los elementos que están en el intervalo [80 a 90] son asignados por una función lineal decreciente. Los elementos que no están dentro del intervalo [a, c] se les asigna un valor de pertenencia cero, semánticamente, estos elementos no pertenecen al conjunto.

En el ejemplo de la figura 2.1 se muestra un conjunto difuso asimétrico sesgado hacia la derecha. Si se desea definir un conjunto difuso simétrico, la diferencia absoluta entre a y b debe ser igual a la diferencia absoluta entre b y c.

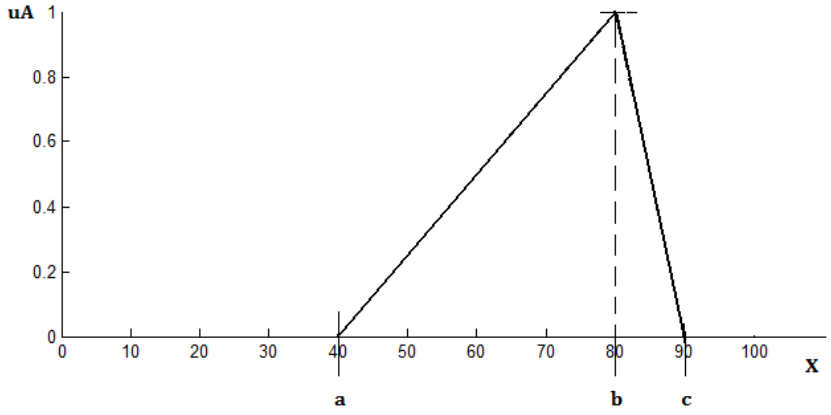


Figura 2.1 Función de membresía triangular paramétrica con valores *a* = 40, *b* = 80 y *c* = 90.

Como se ve en la figura 2.1 la definición de la función de membresía triangular es muy intuitiva, y permite al usuario utilizarla de manera fácil y directa. Esta función se caracteriza por ser

la función de membresía con menor complejidad matemática propuesta en la literatura, y es ampliamente utilizada en la implementación en hardware de sistemas difusos. Sin embargo, una desventaja es que la función tiene discontinuidades que hacen que el entrenamiento de los parámetros utilizando métodos del gradiente sea llevado acabo por partes.

Función campana generalizada

Esta función es definida mediante tres parámetros como sigue:

$$\mu_{Bell}(x; a, b, c) = \frac{1}{1+|\frac{x-c}{a}|^{2b}}.$$

Al igual que en la función triangular los parámetros de la función campana generalizada, tiene un significado semántico, en el cual el parámetro *c* determina el centro de la función, el parámetro *a* determina el ancho de la función establecido por el intervalo [*c* - *a*, *c* + *a*], donde los extremos del intervalo, tienen un valor de pertenencia igual a 0.5, el valor del parámetro *b*, usualmente positivo, conjuntamente con el valor del parámetro a, definen la forma de la función. La función campana es una generalización de la distribución probabilística de *Cauchi*. En la figura 2.2 se muestra la forma de una función de membresía campana generalizada con los parámetros $a = 10$, $b = 5$, y $c = 70$, donde puede observarse que varios valores de *x*, pueden tener un valor de pertenencia igual a 1.

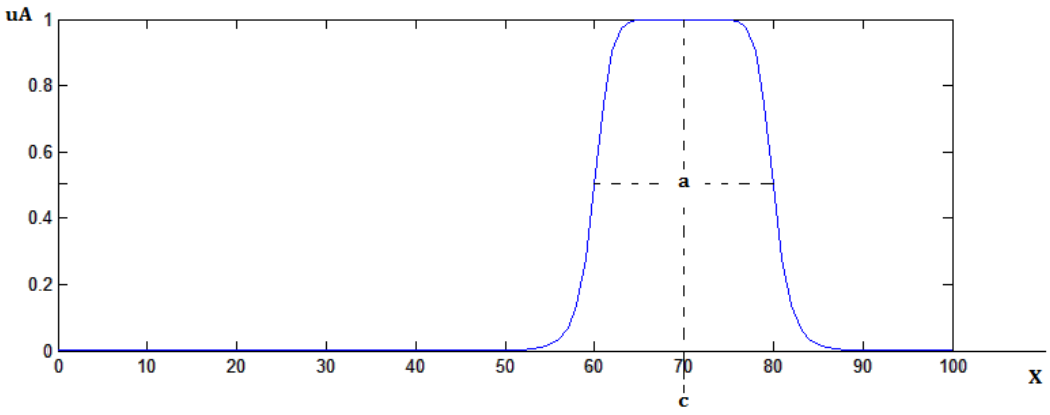


Figura 2.2 Función de membresía campana generalizada de bell con valores a = 10, b = 5, c = 70.

Como se ve en la figura 2.1 la definición de la función de membresía triangular es muy intuitiva, y permite al usuario utilizarla de manera fácil y directa. Esta función se caracteriza por ser la función de membresía con menor complejidad matemática propuesta en la literatura, y es ampliamente utilizada en la implementación en hardware de sistemas difusos. Sin embargo, una desventaja es que la función tiene discontinuidades que hacen que el entrenamiento de los parámetros utilizando métodos del gradiente sea llevado acabo por partes.

La figura 2.3 muestra ejemplos de la asignación de valores lingüísticos que cubren el dominio de la variable lingüística estatura. La primera fila muestra la asignación de valores lingüísticos distribuidos de forma uniforme de acuerdo al número de conjuntos, es decir para 3, 3 y 5 valores

lingüísticos descritos con funciones de membresía triangular, y campana generalizada respectivamente. La segunda fila muestra posibles asignaciones que un experto configuraría para una hipotética aplicación.

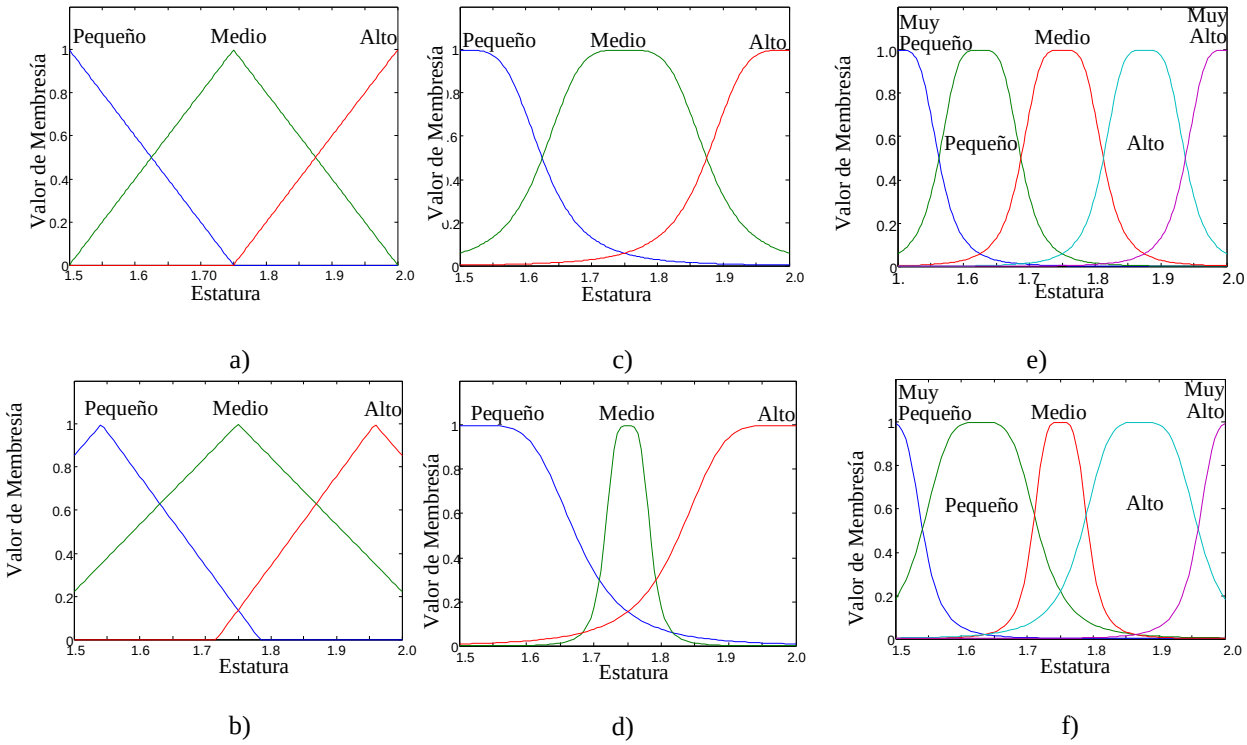


Figura 2.3. Diferentes configuraciones de los valores lingüísticos en la variable estatura.

2.2.4. Operadores difusos

Al igual que en la teoría conjuntos clásicos, las operaciones de unión, intersección y complemento son utilizadas en la teoría de conjuntos difusos, adecuando estas operaciones al dominio difuso [0,1].

En este trabajo únicamente se va a presentar de manera detallada la operación de intersección de conjuntos difusos. Las operaciones de unión y negación difusas pueden encontrarse en cualquier libro de texto de lógica difusa o cómputo suave [**2**,**3**,**4**].

La operación de intersección difusa puede definirse mediante diferentes funciones que cumplan algunas propiedades que generalicen la operación intersección en la teoría de conjuntos clásicos. Existen dos funciones principales para extender el concepto de intersección al dominio difuso, que son: a) los *conjunctores* que cumplen con las propiedades frontera $T(a, 1) = a$, $T(1, a) = a$ y de monotonía y; b) las t-normas que cumples las propiedades de los conjunctores además de las propiedades de conmutatividad y asociatividad.

Definiciones para operadores de intersección difusa

Sean las siguientes condiciones definidas en el espacio difuso [0,1]

A_1. Propiedad de Frontera: $T(0,0) = 0,\ T(a,1) = a \quad T(1,b) = b$
A_2. Propiedad de Monotonicidad: $T(a,b) \leq T(c,d) \quad si\ a \leq c\ y\ b < d$
A_3. Propiedad de Conmutatividad: $T(a,b) = T(b,a)$
A_4. Propiedad de Asociatividad: $T(a,T(b,c)) = T(T(a,b),c)$

Conjunctor

Es una función definida en el dominio difuso $T: [0,1] \times [0,1] \rightarrow [0,1]$ que satisface las propiedades A_1 y A_2.

Norma triangular (t-norma)

Es una función definida en el dominio difuso $T: [0,1] \times [0,1] \rightarrow [0,1]$ que satisface las propiedades A_1, A_2, A_3 y A_4.

Como se estable en las definiciones anteriores, es visible que la función *conjunctor* es un concepto más general, mientras que la definición de una t-norma es más específica, además es claro que una t-norma es un *conjunctor*.

Batyrshin, et al. [**5**,**6**], plantean el desarrollo, las ventajas, y usabilidad de los conjunctores en aplicaciones industriales de áreas como el control, ya que en la mayoría de estas aplicaciones no es necesario que las entradas al sistema sean conmutativas ya que éstas tienen significados físicos diferentes. Aunado a esto, en problemas de dos entradas la propiedad distributiva evidentemente no es necesaria. Sin embargo, para problemas en aplicaciones de minería de datos, sistemas expertos, la implementación de conjunctores como operador de intersección probablemente no sea muy adecuada, por lo que en este tipo de aplicaciones, podría ser mejor utilizar t-normas como operador de intersección o conjunción en el contexto de la lógica.

Notemos que la operación de intersección en la teoría de conjuntos, tiene sus operaciones análogas en la lógica proposicional y el álgebra de Boole, referenciadas como operaciones de *conjunción* o intersección u operación *and* respectivamente, por lo que en este trabajo se estarán utilizando estos tres términos de manera indistinta. Sin embargo, se debe tener claro, que una conjunción no es lo mismo que un *conjunctor*. Ya que como se ha mencionado un *conjunctor* es una de varias funciones que se utilizan para extender la operación de *conjución* (*and* o intersección) al dominio difuso.

Zadeh en su artículo de origen, *Fuzzy Sets* [**7**] propone como operador fundamental de intersección difusa a la t-norma tipo mínimo, estableciendo que es la función natural para generalizar la intersección de conjuntos al dominio difuso. Sin embargo, deja abierta la posibilidad

del uso de diferentes t-normas para definir este operador. Cabe mencionar, que las funciones t-normas inicialmente son funciones desarrolladas en el contexto de funciones de probabilidad. Sin embargo, esto no significa que la extensión de los operadores de intersección sean funciones probabilísticas. Por lo tanto, estas funciones tienen diferentes interpretaciones de acuerdo al contexto o marco de trabajo.

T-normas básicas

Las funciones t-normas más frecuentes que han sido utilizadas en la literatura de lógica difusa son: a) mínimo; b) producto algebraico; c) producto acotado y; d) producto drástico, y que han sido clasificadas como t-normas básicas por diferentes autores (ver [**2**,**8**,**9**]). En este trabajo, la t-norma *nilpotente* también será considerada como una t-norma básica, ya que ésta comparte la característica de tener una definición matemática sencilla.

A continuación, se muestra la definición matemática de los operadores t-norma básica. En la figura 2.4., se muestra la gráfica de superficie de las t-nomas básicas, donde obviamente el valor de los ejes ordenadas están definidos en el intervalo cerrado [0, 1] ya que estos corresponden a valores de membresía de conjuntos difusos.

Mínimo:	$T_M(a,b) = \min(a,b) = a \wedge b$
Producto algebraico:	$T_P(a,b) = a.b$
Nilpotente:	$T_N(x,y) = \begin{cases} \min(x,y), & if\ x+y>1 \\ 0 & otherwise \end{cases}$
Producto acotado o Lukasiewicz:	$T_L(a,b) = \max(0,(a+b-1)$
Producto drástico:	$T_D(a,b) = \begin{cases} a & if\ \ b=1 \\ b & if\ \ a=1 \\ 0 & en\ otro\ caso \end{cases}$

De manera general, se puede observar que:

$$T_D(a,b) \le T_L(a,b) \le T_P(a,b) \le T_M(a,b).\ \text{ó}$$
$$T_D(a,b) \le T_L(a,b) \le T_N(a,b) \le T_M(a,b).$$

Para todo $a, b \in [0,1]$, es decir, cumplen el principio de monotonicidad en este orden lo cual puede ser verificado matemáticamente, ver [**2**,**8**] .

También se ha verificado que la relación de monotónica para las t-normas nilpotente y producto no se cumple, es decir $T_N(a,b) \le T_P(a,b)$ o $T_P(a,b) \le T_N(a,b)$ no se cumplen.

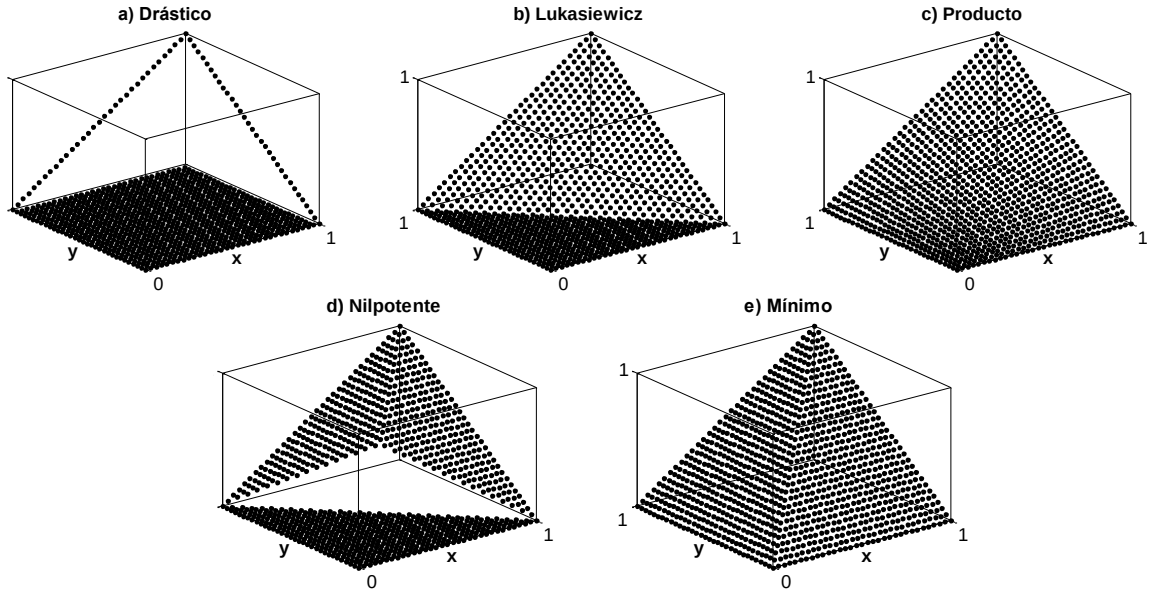


Figura 2.4. t-normas básicas. a) Drástico, b) Lukasiewicz, c) Producto, d) Nilpotente y e) Mínimo.

T-normas paramétricas

Las funciones t-normas [**9**] y s-normas (operador de unión) paramétricas han sido propuestas por diferentes investigadores tales como Yager, Schweizer y Sklar, Dubois y Prade, Hamacher, Frank, Sugeno, Dombi como una motivación de generar operadores que puedan ser ajustados en función de una aplicación específica. Otras investigaciones se han enfocado al entrenamiento de estos operadores implementados en modelos difusos [**10**,**11**].

A continuación, se muestran las ecuaciones que definen algunas funciones t-normas paramétricas.

$$T_{SS}(a,b;p) = [\max(0,(a^{-p}+b^{-p}-1))]^{-\frac{1}{p}}$$

$$T_{Y}(a,b;p) = 1 - \min\{\,1,\,[(1-a)^{p}+(1-b)^{p}]^{-\frac{1}{p}}\,\}$$

$$T_{DP}(a,b;p) = \frac{ab}{\max(ab,p)}$$

$$T_{H}(a,b;p) = \frac{ab}{\mathrm{p}+(1-\mathrm{p})(a+b-ab)}$$

$$T_{F}(a,b;p) = log_{p}[\frac{1+(p^{a}-1)(p^{b}-1)}{p-1}]$$

$$T_{S}(a,b;p) = \max[0,(p+1)(a+b-1)-pab]$$

$$T_{D}(a,b;p) = \frac{1}{1+[(a^{-1}-1)^{p}+(b^{-1}-1)^{p}]^{\frac{1}{p}}}$$

Uno de los ejes centrales de esta investigación es la propuesta de operadores de conjunción paramétricas. Sin embargo, en esta tesis no se trabajó con los operadores paramétricos que se acaban de presentar debido principalmente a que estos involucran una complejidad matemática que requiere de muchos recursos en su implementación y diseño en dispositivos FPGA's. Otras propuestas más recientes sobre operadores de conjunción paramétricos a través de conjunctores han sido presentadas en [**5**,**6**], que se caracterizan por su implementación simple en sistemas digitales. Sin embargo, estos operadores se apoyan de funciones generadoras para *parametrizar* los operadores de conjunción y disyunción básicos.

En este trabajo se presenta la metodología de suma *monotónica* de t-normas básicas para la generación de operadores conjunctores, llamados operadores paramétricos de conjunción de suma monotónica-*p* introducidos en [**12**,**13**]. A partir de esta metodología, durante este trabajo de tesis, se ha extendido la metodología construcción de operadores de conjunción paramétrica, basándose en diferentes ideas de la suma ordinal, suma *monotónica* de t-normas básicas y la norma drástica. Dichas metodologías y la propuesta para la generación de t-normas unificada será presentada en el siguiente capítulo.

2.2.5. Lógica difusa

Al igual que los conjuntos difusos, la lógica difusa es una extensión de lógica proposicional. En la cual el valor de verdad de las proposiciones se extiende al dominio difuso. Por lo que prácticamente todos los conceptos de la lógica proposicional son aplicables a lógica difusa. En este trabajo únicamente se hablará de la máquina de inferencia tipo Sugeno de primer orden, la cual está basada en la regla inferencia *modus ponens* de la lógica clásica. Para los lectores que requieran un análisis más profundo de este tema referirse a ([**2**,**3**,**4**,**14**,**15**]).

Antes de describir la regla de inferencia difusa en el cual está basado razonamiento aproximado o difuso, se presenta un ejercicio de un sistema basado en reglas.

2.2.6. Sistema basado en reglas

Los sistemas basados en reglas tienen la característica de modelar a un sistema, describiendo el comportamiento del sistema mediante reglas del tipo si-entonces, adquiriéndose el conocimiento de un *experto* como fuente principal de conocimiento.

Ejemplo1: Suponga que se desea implementar un sistema que calcule las calorías quemadas de un corredor basándose en la complexión del corredor y la velocidad promedio del corredor en un tiempo de 50 min. Se desea realizar una base de reglas que describa el comportamiento del sistema, utilizando dos valores lingüísticos por cada variable de entrada y una base de reglas que consideras todas las combinaciones al utilizar la operación de conjunción.

Solución.

1. Se asignan valores lingüísticos a cada variable.

Complexión = {*Delgado*, *Gordo*}
Velocidad = {*Lento* y *Rápido*}
Calorias Quemadas = {*Muy Pocas*, *Pocas*, *Muchas*, *Demasiadas*}

2. Describir la base de reglas utilizando reglas de inferencia difusa si-entonces:

R1: Si *Complexión* es *Delgado* y *Velocidad* es *lenta* entonces *CalQuemadas* es *Pocas*
R2: Si *Complexión* es *Delgado* y *Velocidad* es *Rápido* entonces *CalQuemadas* es *Muchas*
R3: Si *Complexión* es *Gordo* y *Velocidad* es *lenta* entonces *CalQuemadas* es *Muy Pocas*
R4: Si *Complexión* es *Gordo* y *Velocidad* es *Rápido* entonces *CalQuemadas* es *Demasiadas*

Hay que notar que un sistema basado en reglas, no es un modelo específico para sistema difusos, más bien puede considerarse como un modelo general que utiliza un enfoque granular, debido a que los valores lingüísticos pueden ser definidos mediante conjuntos clásicos *per se* o con alguna técnica de computo granular [**16**,**17**]. Por lo tanto, un sistema difuso, puede catalogarse como una técnica de cómputo granular, donde los valores lingüísticos se definen mediante funciones de membresía, es decir, por conjuntos difusos.

Un sistema difuso utiliza una máquina de inferencia para poder procesar las reglas si-entonces y generar conclusiones, basada en el razonamiento aproximado, que extiende o generaliza la regla de *inferencia modus ponens*, derivando la inferencia de conclusiones del conjunto reglas difusas si-entonces y/o hechos conocidos o aproximados.

2.2.7. Regla composicional de inferencia

Matemáticamente se puede expresar la regla composicional de inferencia, la cual desempeña un papel clave en el razonamiento difuso, como sigue. Sea F es una relación difusa sobre $X \times Y$ y A un conjunto difuso en X, para encontrar el conjunto difuso B generado al evaluar la relación difusa, es decir $B = F(A)$, y se expresa de manera general como:

$$\mu_B(y) = \vee_x[\, \mu_A(x) \wedge \mu_F(x, y)]$$

De manera específica, si las operaciones de conjunción y disyunción son implementadas por la t-norma, y la s-norma mínimo y máximo respectivamente, entonces la regla composicional queda expresada como.

$$\mu_B(y) = \max_x \min[\mu_A(x), \mu_F(x, y)]$$

Por lo que el conjunto B inferido puede ser representado como la composición máx-mín de A con F, es decir: $B = A \circ F$. Usando la regla composicional de inferencia, se puede formalizar un procedimiento de inferencia a llamado razonamiento aproximado o difuso.

2.2.8. Modus ponens generalizado

La regla de inferencia básica *modus ponens*, se utiliza para inferir el valor de verdad de la proposición *B*, a partir del valor de verdad de *A* y la proposición $A \rightarrow B$. Debido a que la mayor parte del razonamiento humano es aproximado, la regla de inferencia *modus ponens* puede ser empleada de manera aproximada. Por ejemplo, si se tiene la regla de implicación, "*si la ropa está sucia, entonces agregar jabón*", y si tenemos el hecho, "*la ropa esta poco sucia*", entonces se puede inferir la proposición "*agregar poco jabón*", esto se escribe como:

Premisa 1 (hecho):	*x* es *A*',
Premisa 2 (regla):	si *x* es *A* entonces *y* es *B*,
Consecuencia (conclusión):	*y* es *B*',

Al procedimiento de inferencia anterior se le llama razonamiento aproximado o razonamiento difuso, y también puede llamársele *modus ponens* generalizado (GMP por sus siglas en inglés).

Definición

Si *A*, *A*' y *B* son conjuntos difusos en *X*, *X* y *Y* respetivamente y la implicación $A \rightarrow B$ es expresada como una relación difusa *R* sobre *X* x *Y*. Entonces el conjunto difuso *B*' inducido por "*x* es *A*'" y la regla difusa "si *x* es *A* *entonces y es B*" basado en la composicional es definido por:

$$B' = A' \circ R = A' \circ (A \rightarrow B)$$

Por lo que a la función de membresía de B' se expresa

$$\mu_{B'}(y) = \vee_x [\, \mu_{A'}(x) \wedge \mu_R(x, y)]$$

Específicamente, si la regla de implicación $A \rightarrow B$ es interpretada como A acoplada con B, entonces $A \rightarrow B$ es equivalente a $A \wedge B$, entonces el conjunto difuso inferido se puede expresar como

$$\mu_{B'}(y) = \vee_x [\, \mu_{A'}(x) \wedge (\mu_A(x) \wedge \mu_B(y))]$$
$$\mu_{B'}(y) = \vee_x [\, \mu_{A'}(x) \wedge \mu_A(x)] \wedge \mu_B(y)$$

Hay que notar que $\vee_x [\, \mu_{A'}(x) \wedge \mu_A(x)]$ es un escalar, referenciado en la literatura de la lógica difusa como fuerza o valor de disparo del antecedente, simbólicamente w, entonces, $\mu_{B'}(y)$ queda expresada como:

$$\mu_{B'}(y) = w \wedge \mu_B(y)$$

2.2.9. Inferencia difusa con múltiples reglas y antecedente múltiple

La interpretación de múltiples reglas dentro de una base de reglas o de conocimiento, es usualmente tomada como la unión de las relaciones difusas correspondientes a las reglas difusas. De acuerdo al método *modus ponen* generalizado tenemos:

Premisa 1 (hecho): x es A' y y es B',
Premisa 2 (regla 1): si x es A_1 y y es B_1 entonces z es C_1
Premisa 3 (regla 2): si x es A_2 y y es B_2 entonces z es C_2
Consecuencia (conclusión): z es C'

Si decimos que las reglas de inferencia difusa, son expresadas como las siguientes relaciones, $R_1 = (A_1B_1 \rightarrow C_1)$ y $R_2 = (A_2B_2 \rightarrow C_2)$. Considerando que el operador de composición max-min es distributivo sobre el operador de unión ∪, tenemos:

$$C' = (A' \wedge B') \circ (R_1 \cup R_2),$$
$$C' = [(A' \wedge B') \circ R_1] \cup [(A' \wedge B') \circ R_2],$$
$$C' = C_1' \cup C_2'.$$

De igual manera, si la regla de implicación es interpretada como A ∧ B acoplado con C, se tiene la equivalencia A ∧ B ∧ C, entonces $R_1 = A_1 \wedge B_1 \wedge C_1$ y $R_2 = A_2 \wedge B_2 \wedge C_2$. Por lo tanto C' puede expresarse como:

$$C' = [(A' \wedge B') \circ (A_1 \wedge B_1 \wedge C_1)] \cup [(A' \wedge B') \circ (A_2 \wedge B_2 \wedge C_2)]$$
$$C' = \vee_{x,y} [(A' \wedge B' \wedge A_1 \wedge B_1 \wedge C_1)] \cup \vee_{x,y} [(A' \wedge B' \wedge A_2 \wedge B_2 \wedge C_2)]$$
$$C' = [\vee_x (A' \wedge A_1) \wedge \vee_y (B' \wedge B_1) \wedge C_1)] \cup [\vee_x (A' \wedge A_2) \wedge \vee_y (B' \wedge B_2) \wedge C_2)]$$
$$C' = [w_{11} \wedge w_{12} \wedge C_1)] \cup [w_{21} \wedge w_{22} \wedge C_2)]$$

Por lo que la función de pertenencia del conjunto difuso C' puede calcularse de la siguiente manera:

$$\mu_{C'}(z) = \vee_{x,y}\big[\ [\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_{A_1}(x) \wedge \mu_{B_1}(y) \wedge \mu_{C_1}(z)]\ \big] \cup \vee_{x,y}\big[\ [\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_{A_2}(x) \wedge \mu_2(y) \wedge \mu_{C_2}(z)]\ \big]$$

$$\mu_{C'}(z) = [\vee_x\big(\mu_{A'}(x) \wedge \mu_{A_1}(x)\big) \wedge \vee_y\big(\mu_{B'}(x) \wedge \mu_{B_1}(x)\big) \wedge \mu_{C_1}(z)] \cup [\vee_x\big(\mu_{A'}(x) \wedge \mu_{A_2}(x)\big) \wedge \vee_y\big(\mu_{B'}(x) \wedge \mu_{B_2}(x)\big) \wedge \mu_{C_2}(z)]$$

$$\mu_{C'}(z) = [w_{11} \wedge w_{12} \wedge \mu_{C_1}(z)] \cup [w_{21} \wedge w_{22} \wedge \mu_{C_2}(z)]$$

Si $w_1 = w_{11} \wedge w_{12}$ y $w_2 = w_{21} \wedge w_{22}$, entonces $\mu_{C'}(z) = [w_1 \wedge \mu_{C_1}(z)] \cup [w_2 \wedge \mu_{C_2}(z)]$

Ya que C_1' y C_2' son conjuntos difusos inferidos por las reglas difusas 1 y 2 respectivamente y el operador ∪ puede ser calculado con cualquier s-norma. Las t-normas y s-normas utilizadas por Mamdani fueron la función Mínimo y Máximo respectivamente. En la figura 2.5., se muestra la gráfica de la implicación difusa con múltiples reglas de antecedentes múltiples.

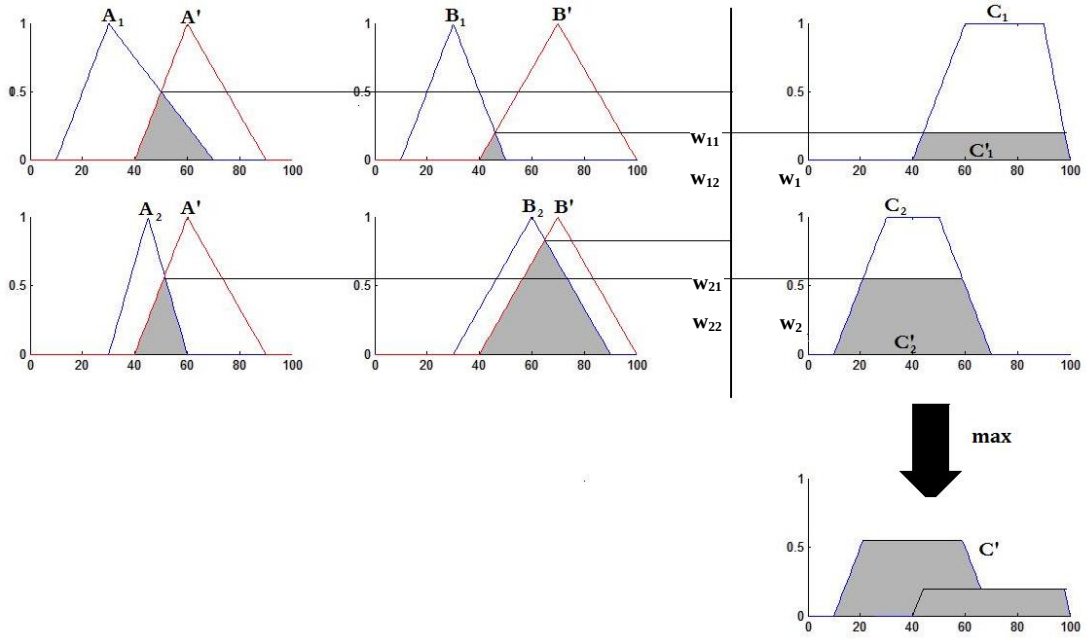


Figura 2.5 Razonamiento difuso con múltiple reglas de antecedentes múltiple.

2.2.10. Sistemas de inferencia difusa

La estructura básica de los sistemas de inferencia difusa consiste en tres componentes: a) una base de reglas en donde se definen el conjunto de reglas difusas; b) base de datos o diccionario, donde se guardan los parámetros de las funciones de membresía, los tipos, y los parámetros de las funciones lineales de salida (en el caso de los sistemas Sugeno) y; c) un mecanismo de razonamiento, en el cual se establece el procesamiento del razonamiento.

Los modelos difusos más populares son los sistemas difusos Mamdani y Sugeno. El primer sistema tiene como ventaja principal la capacidad de plasmar el conocimiento del experto de manera directa, pero como desventajas principales, es que este sistema proporciona una salida difusa, que en muchas aplicaciones no es deseable, como en el caso de sistemas de control, por lo que el modelo debe de agregar una etapa de *defuzificación*, que normalmente produce un cuello de botella en la generación de las inferencias. Por otro lado, el modelo del sistema tipo Mamdani, no proporciona mucha flexibilidad para el entrenamiento del modelo, ya que el ajuste de las funciones de los parámetros de las funciones de membresía que caracterizan a los conjuntos difusos puede llevar a la pérdida del conocimiento adquirida a través del experto. Por lo tanto, es visible que la fuente de información principal es la experiencia del humano.

A diferencia del modelo Mamdani, el modelo Sugeno fue diseñado para agregar la propiedad de *adaptatividad*, y por ende la libertad de ajuste de los parámetros de acuerdo a un conjunto de entrenamiento. Con ello se pierde, en cierto grado, la propiedad de adquisición del conocimiento del experto a través de un conjunto de reglas que describen el comportamiento del sistema, ya que los consecuentes de las reglas son caracterizadas por funciones matemáticas en lugar de valores lingüísticos. Por lo que este modelo puede ser catalogado como un sistema *semidifuso*.

El costo de perder la característica de tener un modelo completamente adecuado para procesar la información a partir de un conjunto de reglas enunciadas mediante lenguaje natural, y establecido por un experto que conoce el comportamiento del sistema a modelar. Se ve altamente compensado, principalmente en el modelo Sugeno de primer orden, ya que el ajuste de los parámetros de las funciones lineales consecuentes en función de un conjunto de entrenamiento pueden ser realizada mediante técnicas de identificación lineal, tales como el método del estimador de mínimos cuadrados, el cual se encarga de encontrar los valores óptimos que minimizan el error entre el valor deseado o *target* y el valor de salida del modelo, aunado a lo mencionado, los modelos Sugenos no necesitan una etapa de *defuzificación*, ya que la salida siempre genera valores certeros, por lo que el costo de procesamiento es reducido.

Los diferentes tipos de modelos difusos, difieren uno del otro. Por un lado, en la manera en cómo se describen las funciones consecuentes del conjunto de reglas que describe al sistema destino y por otro lado, en la manera de se agregan las fuente de conocimiento al modelo. Se puede concluir que el modelo Mamdani tiene como fuente principal de conocimiento la experiencia humana, mientras que el modelo Sugeno muestra un mayor equilibro de la importancia de las fuentes de información adquiridas del conocimiento la experiencia humana y de los datos obtenidos del sistema.

En la siguiente sección se explica la estructura básica del modelo Sugeno de primer orden, si se desea mayor información de los otros modelos referirse a [**2**,**3**,**4**].

2.2.11. Modelo de inferencia difusa Sugeno

El modelo difuso Sugeno (también conocido como modelo difuso Takagi-Sugeno-Kang o TSK) fue propuesto por Takagi, Sugeno y Kang, en un esfuerzo para desarrollar un modelo difuso sistemático para la generación y ajuste de reglas difusas a partir de un conjunto de datos de entrada y salida. Una regla difusa típica en un modelo difuso Sugeno tiene la forma:

$$\text{Si } x \text{ es } A \text{ y } y \text{ es } B \text{ entonces } z = f(x, y)$$

Donde A y B son conjuntos difusos antecedentes, mientras que el consecuente $z = f(x, y)$ es una función en los reales. De manera general, $f(x, y)$ es un polinomio en función de las variables de entrada x e y. Si $f(x, y)$ es igual a una constante el sistema difuso se conoce como modelo difuso Sugeno de orden cero.

Cuando el polinomio $f(x,y)$ es descrito por una ecuación de primer orden al sistema se le conoce como sistema difuso Sugeno de primer orden. El valor de salida *z* se calcula con la suma de productos de las fuerzas de disparo de cada regla w_i por el valor de la función consecuente z_i, entre la suma de las fuerzas de disparos de la base de reglas.

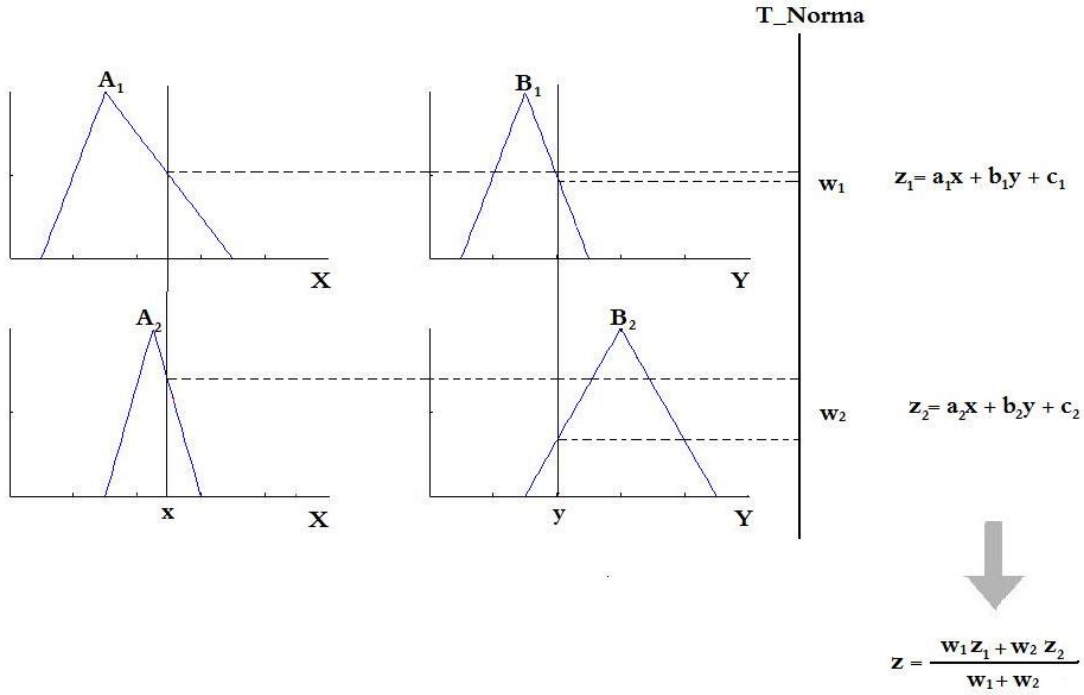


Figura 2.6 Inferencia difusa de modelo difuso Sugeno de primer orden.

En la siguiente ecuación se muestra el cálculo de *z*, mientras que la figura 2.6 muestra gráficamente el proceso de inferencia del modelo Sugeno.

$$z_C = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

2.3. Redes neuronales artificiales

Las redes de neuronas artificiales son un paradigma utilizado en la inteligencia computacional para el aprendizaje y procesamiento automático, que se inspira en la forma en que funciona el sistema neuronal y nervioso de los seres vivos, replicando el sistema de interconexión de neuronas que colaboran entre sí y exhiben un comportamiento inteligente. Existe un gran diversidad de referencias que presenta la historia, introducción, diferentes arquitecturas y reglas de aprendizaje de las redes neuronales artificiales, ver [**18**,**19**,**2**]. En este trabajo se presentan únicamente los conceptos fundamentales de este enfoque utilizados en la red ANFIS.

Las redes neuronales artificiales, también conocidas como sistemas o modelos conexionista son muy utilizados en la solución de cómputo inteligente en la identificación de sistemas, mediante modelos modulares y adaptativos, los van aprendiendo o ajustándose siguiendo una regla de aprendizaje que utiliza métodos de optimización, basando en datos entrada-salida de un sistema a replicar o modelar.

2.3.1. Redes adaptativas

Una red adaptativa o de nodos adaptativos, como su nombre indica, es una red neuronal artificial que consiste de un número de nodos conectados a través de enlaces direccionales. Cada nodo representa una unidad de procesamiento y los enlaces entre nodos especifican la relación causal entre los nodos conectados. Todos los nodos o parte de ellos son adaptativos, lo que significa que las salidas de estos nodos dependen tanto de las entradas como de parámetros modificables pertenecientes a estos nodos.

En el caso más general, una red adaptativa es una red heterogénea donde cada nodo puede tener una función de nodo específica diferente a las demás, por otro lado una red adaptativa puede generalizar a cualquier red neuronal, por ejemplo si en los nodos de la red se implementan funciones lineales, la red neuronal es una *Adaline*, o si en los nodos se implementa la composición de funciones lineales y funciones *gausianas*, la red neuronal *perceptron* multicapa, por citar los ejemplos más comunes.

Las redes adaptativas pueden clasificarse de acuerdo a la forma de conectar los nodos que la integran: a) la red de alimentación directa (o *feedforward*), donde la salida de cada nodo se propaga desde el lado de entrada (izquierda) al lado de salida (a la derecha) en todos los casos y b) la red recurrente (recurrent) que permite conexiones cíclicas o circulares dentro en la red.

La estructura o arquitectura de una red neuronal puede tener una representación por capas, agrupando un número específico de nodos en cada capa, permitiendo una estructura modular. Un ejemplo de una red neuronal de alimentación directa de dos entradas y dos salidas, con representación por capas, y un total de 9 nodos se muestra en la figura 2.7. Este esquema diferencia a los nodos adaptativos y a los nodos fijos utilizando cuadrados y círculos respectivamente.

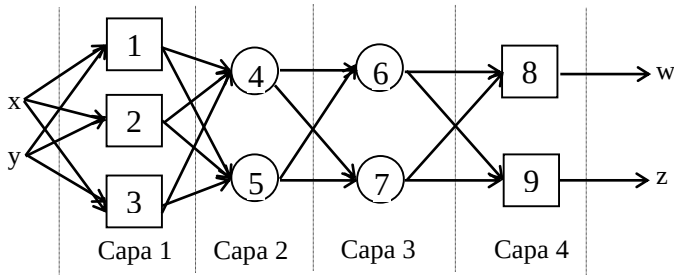


Figura 2.7 Red neuronal de cuatro capas.

Una red adaptativa se utiliza para la identificación del sistema, por lo que la tarea del diseñador es encontrar una arquitectura de red adecuada y un conjunto de parámetros que puedan modelar de la mejor manera un sistema destino mediante un conjunto de pares de datos de entrada-salida.

2.3.2. Regla de aprendizaje

En la regla o algoritmo de aprendizaje se especifica la manera de actualizar a los parámetros de los nodos para minimizar una medida de error prescrita, que es una expresión matemática y mide la discrepancia entre la salida real de un sistema objetivo y la salida de la red.

La regla de aprendizaje básica de la red adaptativa que contiene nodos con funciones continuas y diferenciables, está basada en el método del gradiente de mayor descenso, en la que el vector gradiente se obtiene mediante invocaciones sucesivas de la regla de la cadena. Algunos métodos sistemáticos para calcular el vector gradiente en redes neuronales fueron propuestos en la literatura independientemente (ver [**20**,**21**]). Sin embargo, debido a que la investigación de las redes neuronales artificiales se encontraba todavía en sus inicios, los primeros trabajos de estos investigadores no recibieron la atención merecida, y fue hasta en 1986 que resurge esta línea de investigación, cuando Rumelhart et al. [**22**] utilizan el mismo procedimiento para encontrar los gradientes en una red neural multicapa. Su procedimiento se conoce como regla de aprendizaje de propagación hacia atrás o retropropagación (*backpropagation*).

2.3.3. Aprendizaje en-línea y fuera-de-línea

En el aprendizaje fuera de línea, también conocido como aprendizaje por lotes, el ajuste de los parámetros de la red considera todo el conjunto de datos de entrenamiento a la vez, por lo que se debe disponer de un conjunto de datos de entrenamiento preestablecido. Este tipo de aprendizaje es utilizado principales en sistemas invariantes en el tiempo.

En contraste el aprendizaje automático en línea es un método de aprendizaje en donde los datos de entrenamiento del modelo están disponibles en un orden o tiempo secuencial. Por lo que en cada momento que un dato está disponible es utilizado para entrenar al modelo. El aprendizaje en línea es una técnica común utilizada en áreas de aprendizaje automático en los que es computacionalmente difícil o imposible entrenar en todo el conjunto de datos. También, se utiliza en situaciones en las que queremos que el algoritmo de aprendizaje permita el ajuste dinámico de los parámetros de la red en función de los nuevos patrones de datos o cuando los datos en si se generan en función del tiempo.

Una vez establecidos los fundamentos de los sistemas difusos y redes neuronales, es posible presentar a la red ANFIS.

2.4. Sistema neuro-difuso ANFIS

La arquitectura del Sistema de Inferencia Difusa basados en Redes Neuronales Adaptativas-(ANFIS por sus siglas en inglés), es un modelo neuronal de redes adaptativas que son

funcionalmente equivalentes a los sistemas de inferencia difusa, permitiendo la modulación por capas con el objetivo de facilitar el entrenamiento del modelo. A pesar de que la arquitectura ANFIS puede implementar cualquier tipo de modelo difuso, el modelo Sugeno de primer orden, ha sido el más popular, debido a que la identificación de los parámetros consecuentes puede realizarle con algún método de mínimos cuadrados que lleva a una pronta convergencia del modelo. A continuación, se describe la arquitectura del modelo ANFIS para un sistema Sugeno de primer orden.

2.4.1. Arquitectura ANFIS

Por simplicidad, se considera un sistema de inferencia difuso Sugeno de dos entradas X e Y y una salida Z, con un conjunto de dos reglas difusas si-entonces como sigue:

Regla 1: Si x es $A1$ y y es $B1$, entonces $f1 = a_1 \cdot x + b_1 \cdot y + c_1$
Regla 2: Si x es $A2$ y y es $B2$, entonces $f2 = a_2 \cdot x + b_2 \cdot y + c_2$

La arquitectura ANFIS equivalente correspondiente al sistema de reglas anterior, es decir, al modelo Sugeno de Primer orden mostrado en la figura 2.5., se muestra en la figura 2.8., donde los nodos con forma cuadrada, son implementados con funciones paramétricas, mientras que los nodos con formas circular se implementan con funciones fijas. Es importante mencionar que correspondiente a una misma capa tiene la misma funcionalidad. A continuación, describen las funcionalidades de cada capa de modelo ANFIS correspondiente al sistema de reglas definido anteriormente.

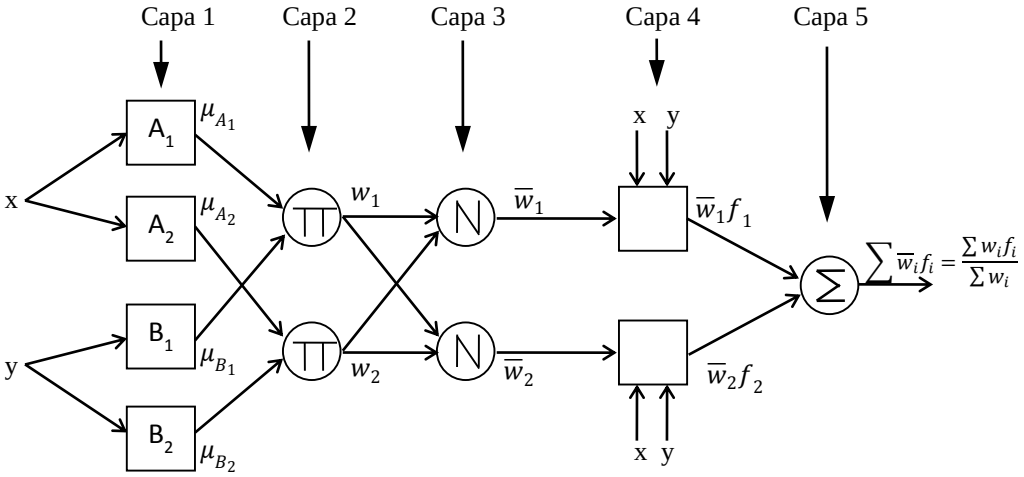


Figura 2.8 Arquitectura ANFIS.

Capa 1. Nodos adaptativos conjuntos difusos

La capa 1 se utiliza para implementar las funciones de membresía paramétricas de cada valor lingüístico de las variables de entrada, es decir se implementan los conjuntos difusos A_1, A_2,

$B_1, y\ B_2$. Los parámetros en esta capa son referidos como parámetros antecedentes del modelo. Simbólicamente las salidas de estos nodos pueden expresarse como sigue

$O_{1,i} = \mu_{A_i}(x)$, donde *i* = 1, 2,

$O_{1,2+j} = \mu_{B_j}(y)$, donde *j* = 1, 2.

Obviamente la funciones de membresía $\mu_{A_i}(x)$ y $\mu_{B_j}(y)$ pueden ser definidas mediante diferentes tipos de funciones de membresía.

Capa 2. Nodos fijos de conjunción

En esta capa se implementan nodos fijos para definir las operaciones de conjunción difusa, utilizando el operador t-norma producto algebraico. Por lo que los nodos de esta capa sólo contienen un multiplicador. Los nodos de esta capa son representados con el símbolo Π. La salida de esta capa representa la fuerza de cada disparo de una regla w_i. De manera general, cualquier otro operador t-norma puede ser utilizada en esta capa. Simbólicamente las salidas de los nodos de esta capa pueden expresarse como:

$O_{2,i} = w_i(x) = \mu_{A_i}(x)\mu_{B_j}(y)$ donde *i* = 1, 2.

Capa 3. Nodos fijos de normalización

Esta capa contiene únicamente nodos fijos representados con la letra *N*. Cada nodo *N* normaliza la fuerza de disparo w_i entre la suma de todas las fuerzas de disparo del modelo. Por qué las salidas de esto son referenciadas como las fuerzas de disparo normalizadas. Simbólicamente las salidas de los nodos de esta capa pueden expresarse:

$$O_{3,i} = \overline{w}_i(x) = \frac{w_i}{w_1+w_2}$$

Capa 4. Nodos adaptativos de funciones lineales

Esta capa contiene nodos adaptativos para implementar la multiplicaciones funciones lineales paramétricas por la fuerza de disparo de cada regla. La salida de estos nodos puede expresarse como sigue:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i(a_i x + b_i y + c_i).$$

Donde $\{a_i,\ b_i,\ c_i\}$ conforma el conjunto de parámetros cada nodo. Los parámetros de esta capa son referidos como parámetros consecuentes.

Capa 5. Nodo fijo de sumatoria

Esta capa contiene un único nodo fijo, que es etiquetado con el símbolo Σ, y calcula la salida, realizando la suma de todas las señales de entrada al nodo. La salida de este nodo puede expresarse como:

$$O_{5,i} = \sum \overline{w}_i f_i = \frac{\sum w_i f_i}{\sum w_i}$$

Hay que notar que aunque en esta arquitectura se realizó la normalización de cada regla disparo, matemáticamente las salidas de ANFIS y del sistema difuso tipo Sugeno son las mismas. Esta adecuación se hace para tener más claro el proceso de entrenamiento de esta red neuronal. A continuación, se describe el algoritmo de aprendizaje de ANFIS.

2.4.2. Algoritmo de aprendizaje híbrido de ANFIS

En la arquitectura ANFIS mostrada en la figura 2.8 se observa que si se establecen valores predeterminados a los parámetros antecedentes, es decir, si los nodos adaptativos de la capa 1 se sustituyen por nodos fijos, la salida total del sistema difuso, puede expresarse como una combinación lineal de los parámetros consecuentes. Matemáticamente, la salida *f* de la figura 2.7 se expresa como:

$$f = (\overline{w}_1 x)a_1 + (\overline{w}_1 y)b_1 + (\overline{w}_1)c_1 + (\overline{w}_2 x)a_2 + (\overline{w}_2 y)b_2 + (\overline{w}_2)c_2$$

Que es lineal con respecto a los parámetros consecuentes a_1, b_1, c_1, a_2, b_2 y c_2. A partir de esta observación se establece la siguiente clasificación de los parámetros:

S = Es el conjunto total de parámetros del sistema difuso
S_1= Representa al conjunto de parámetros antecedentes (no lineales, capa 1)
S_2= Define al conjunto de parámetros consecuentes (lineales, capa 4)

Con esta clasificación de los parámetros, es posible establecer el algoritmo de aprendizaje híbrido basados en un conjunto de entrenamiento en dos pasos principales:

1) Realizando un propagación hacia adelante (o directa), la salida de los nodos de las primeras tres capas se calculan, identificando los parámetros consecuentes utilizando algún método de mínimos cuadrados.

2) Se realiza un propagación hacía atrás, propagando las señales de error, para actualizar los valores de los parámetros antecedentes (no lineales), mediante métodos del descenso del gradiente.

Como se mencionó, los parámetros consecuentes identificados por LSE, son óptimos bajo la condición de que los parámetros antecedentes tienen un valor fijo preestablecido. Como consecuencia de la regla de aprendizaje híbrida el algoritmo converge mucho más rápido

comparado con las otras reglas de aprendizaje de redes neuronales, ya que en ésta se reduce la dimensión del espacio de entrada que si se utiliza el método de *retropropagación* original.

Hay varias formas de combinar los métodos de gradiente descendiente con el método del estimador de mínimos cuadrados. La elección de estos métodos se realiza en función de los recursos computacionales que se dispongan y del nivel de rendimiento que se requiera. En este trabajo se utilizará el método de mínimos cuadrados recursivos, por tres razones fundamentales: a) por un lado este método no requiere el cálculo de la inversa de una matriz, lo que permite establecer un algoritmo más rápido y de menor complejidad; b) durante las pruebas realizadas en este trabajo, se observó que el algoritmo de mínimos cuadrados que utiliza la *matriz pseudo inversa*, en algunas ocasiones genera números muy pequeños que puede llevar a la generación de errores debido a que la matriz se puede indeterminar; c) de acuerdo a las expectativas de trabajos futuros, donde se trabará en la implementación en un FPGA de la arquitectura ANFIS, tanto la metodología de entrenamiento de ANFIS tradicional, así como de la metodología de entrenamiento propuesta en este trabajo, la cual será descrita en los siguientes capítulos, el estimador de mínimos cuadrados (LSE por sus siglas en inglés) recursivos presenta la mejor opción en la implementación que otras técnicas de LSE.

Como fue señalado por el revisor del documento introductorio de ANFIS [**23**], el mecanismo de aprendizaje no debe ser aplicado para ajustar los valores de los parámetros de las funciones de membresía en el modelo de inferencia difusa, ya que estos parámetros conllevan descripciones lingüísticas y subjetivas de conceptos ambiguos definidos por un experto.

Sin embargo, los autores de ANFIS justificaron válidamente la defensa de su artículo evidenciando que este es un problema que debe ser considerado caso por caso, y la decisión se debe dejar al usuario, en donde: a) si el tamaño del conjunto de entrenamiento es muy grande, un ajuste fino de las funciones de membresía es recomendable e incluso necesario, ya que las funciones de membresía determinadas por el humano son raramente óptimas en función de replicar salidas deseadas y; b) si el conjunto de datos de entrenamiento es muy pequeño, entonces no se tendrá suficientes datos de información acerca del sistema a modelar, por lo que la información del experto humano mediante funciones de membresía representan la principal fuente de información del sistema, por lo que en este caso, los parámetros de las funciones de membresía no deben de modificarse en el proceso de aprendizaje.

Otro caso que ha reflejado la no modificación de los parámetros antecedentes del ANFIS-Sugeno, es cuando el modelo se integra a un sistema donde los conjuntos, gránulos, grupos, clases, conceptos, entre otros, han sido previamente establecidos mediante otra técnica del cómputo inteligente.

En los casos donde únicamente el conjunto de parámetros consecuentes son ajustados, convierte a ANFIS-Sugeno de primer orden en una red de enlace funcional lineal, cuyos parámetros pueden ser ajustados mediante técnicas LSE.

El modelo ANFIS-Sugeno ha probado su eficacia al contar actualmente con gran popularidad en aplicaciones de modelado de sistemas. A continuación, se realiza el análisis de los resultados y comportamiento de la arquitectura ANFIS-Sugeno de primer orden, en la aproximación de dos funciones no lineales que Jang uso para mostrar la capacidad de ANFIS en la aproximación en funciones no lineales.

El análisis que va a ser presentado a continuación se modifican el número de conjuntos difusos por variable de entrada y el tipo de funciones de membresía: a) triangular y; b) campana generalizada. Revisando y comparando los siguientes aspectos: a) las gráficas de superficie de la salida; b) las formas de los conjuntos difusos y los valores de los parámetros antecedentes y; c) los errores finales, después del entrenamiento de ANFIS.

2.4.3. Análisis de ANFIS en aproximación de funciones

Como se ha mencionado los modelos de inferencia difusa se utilizan para modelar o replicar un sistema de la vida real, sin embargo, la aproximación de funciones es una tarea muy utilizada en el análisis de sistemas difusos y los sistemas inteligentes. Desde la perspectiva del modelado estas sirven como pruebas para medir la capacidad de aproximación de un modelo, y desde la perspectiva de la aplicación se han utilizado para implementar operadores u operaciones complejas.

A continuación, se presenta el análisis de ANFIS en la aproximación de dos funciones no lineales que servirán como sistema target. La tarea es obtener datos de las funciones $z = sinc(x, y)$ y $T = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2$ para generar un conjunto de entrenamiento. La otra fuente de conocimiento será obtenida mediante un conjunto de reglas difusas si-entonces cuya cardinalidad se establece con el número de total de combinaciones de valores lingüísticos definidos en cada variable de entrada, es decir la partición.

Como en estos ejercicios no se cuenta con un experto que determine el número ideal de conjuntos difusos necesarios por variable de entrada. La tarea de este análisis es variar el número de conjuntos difusos desde *mf_n* = 2 hasta 5 utilizando funciones de membresía triangular y campana generalizada. El criterio para la ubicación inicial de las funciones de membresía, es la distribución uniforme a lo largo del universo de discurso de las variables de entrada, la t-norma utilizada es el producto algebraico y la optimización de los parámetros del modelo ANFIS se realizó utilizando la función de MATLAB *anfis*, del *toolbox fuzzy* la cual utiliza las técnicas del gradiente descendiente abrupto y el método LSE recursivo en la metodología de aprendizaje híbrida.

Ejemplo 1: Modelo de la función *sinc* de dos entradas

En este ejemplo, se utiliza ANFIS para modelar o aproximar la función $sinc(x, y)$ de dos dimensiones definida por:

$$z = sinc(x, y) = \frac{\sin(x)\sin(y)}{xy}$$

El conjunto de entrenamiento se formó por 121 pares de datos de entrenamiento de entrada-salida, igualmente distribuidos en el rango de entrada $[-10, 10]\times[-10,10]$, es decir, se tomaron los valores de *x*, *y* = {-10, -8 -6 -4 -2 0 2 4 6 8 10}. Se utilizaron un conjunto de reglas difusas que realizan una partición completa del universo de discurso, es decir, se utilizaron 9, 16 y 25 reglas difusas del tipo si-entonces para los casos de 3, 4 y 5 conjuntos difusos por variable respectivamente.

El formato utilizado en las gráficas del análisis de ANFIS mostrados en las figuras 2.9 es el siguiente, en la primera columna se muestra la gráfica de superficie del conjunto de entrenamiento, es decir la función objetivo. En segunda columna se muestran la superficie de salida de ANFIS, y la forma de los conjuntos difuso iniciales, es decir, esta gráfica corresponde a la identificación de los parámetros consecuentes por primera vez, definiendo una asignación igualmente distribuida de los conjuntos difusos. La tercera columna muestra los resultados del entrenamiento de ANFIS después de 100 épocas de entrenamiento (100 iteraciones de entrenamiento). Finalmente, se muestran los resultados finales del entrenamiento de ANFIS después de 1000 épocas en la cuarta columna. Para todas las gráficas el eje de las abscisas representa el universo de discurso de las variables de entrada, que en este caso no tienen dimensiones. Y el eje de las ordenadas representa los valores de pertenencia con un rango de [0, 1].

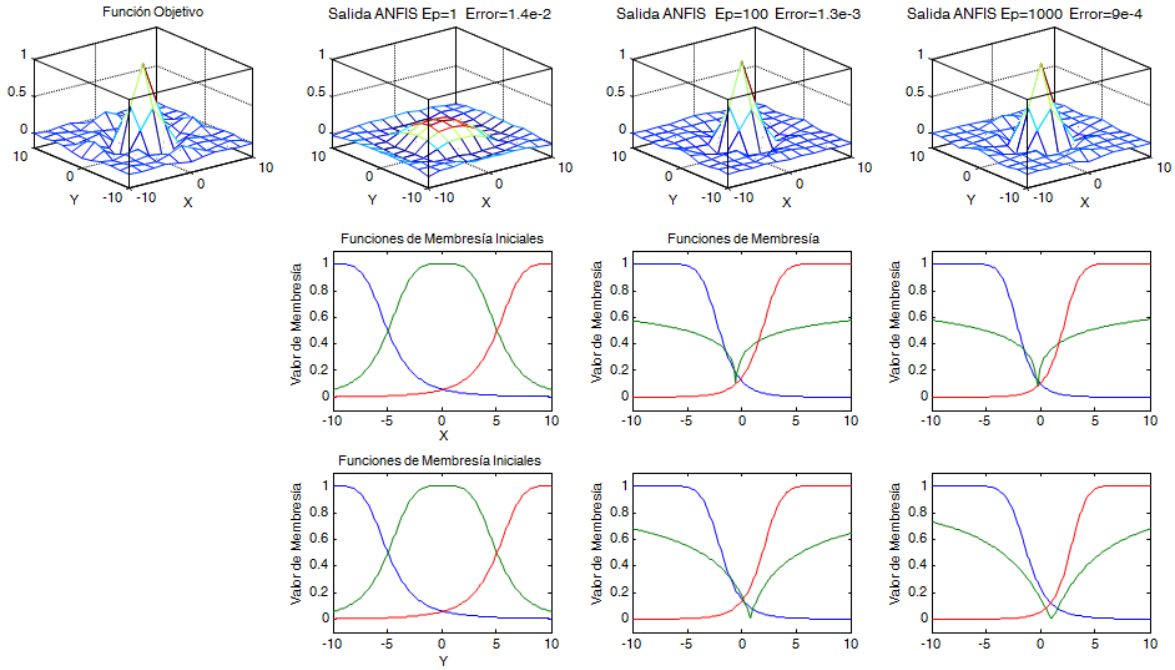


Figura 2.9.a Modelo ANFIS utilizando tres conjuntos difusos del tipo gbell.

Como se muestra en la figuras 2.9 las primeras columnas muestran la aproximación inicial de ANFIS, que equivale al Sugeno tradicional modelado sin ningún tipo de conocimiento, siendo esta respuesta muy buena y sus superficies son muy parecidas a las de la función deseada. Se muestra que entre más conjuntos difusos, se tiene una mejor aproximación, y que la función de membresía del tipo triangular parece ser más adecuada a este sistema.

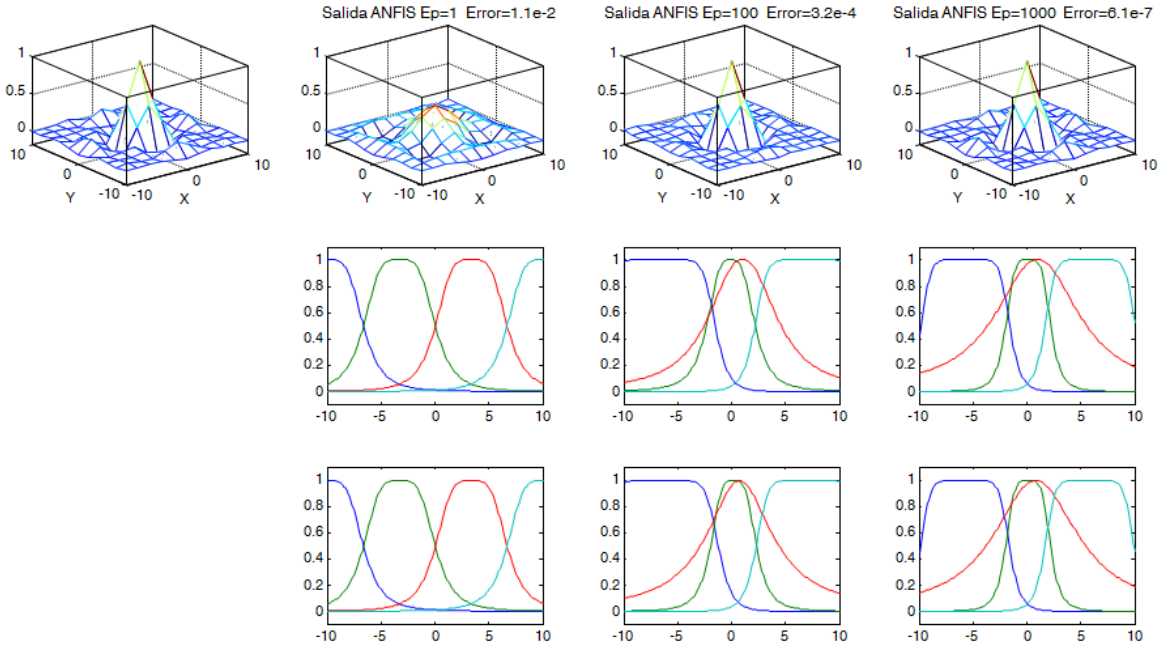


Figura 2.9.b Modelo ANFIS utilizando cuatro conjuntos difusos del tipo gbell

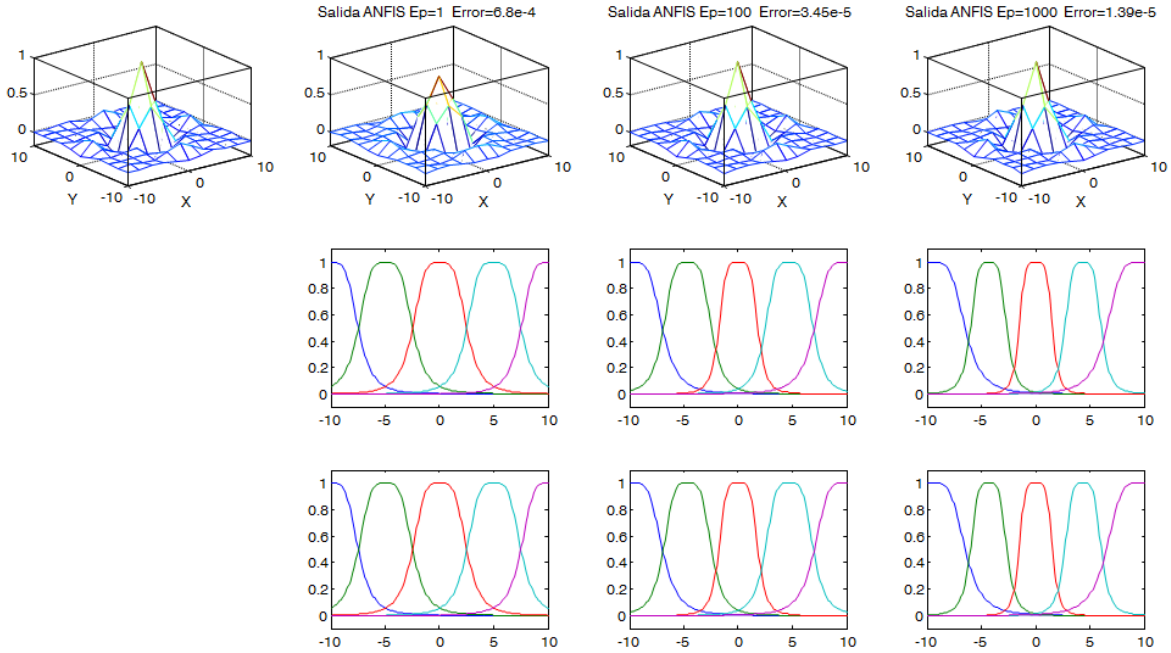


Figura 2.9.c Modelo ANFIS utilizando cinco conjuntos difusos del tipo gbell.

Se muestra también que después de 100 épocas de entrenamiento, prácticamente en todos los casos, la función se aproxima considerablemente a la función objetivo. Sin embargo, se puede ver la perdida de información que muestran los conjuntos difusos o simplemente se ve que el ajuste de los conjuntos es insignificante.

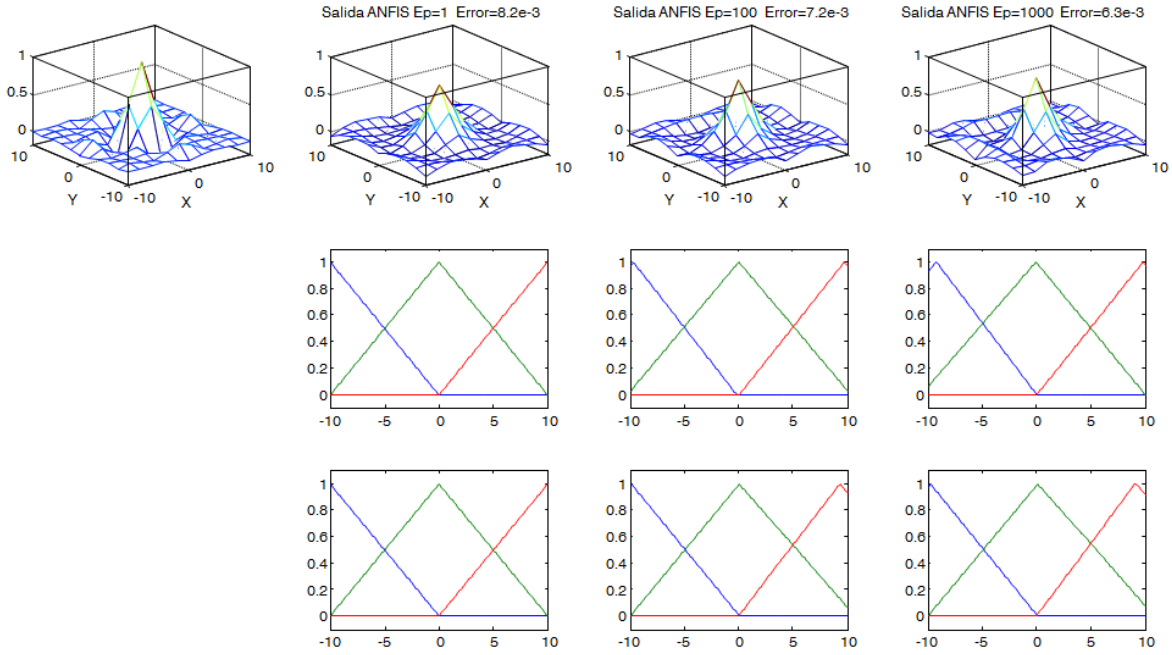


Figura 2.9.d Modelo ANFIS utilizando tres conjuntos difusos del tipo triángulo.

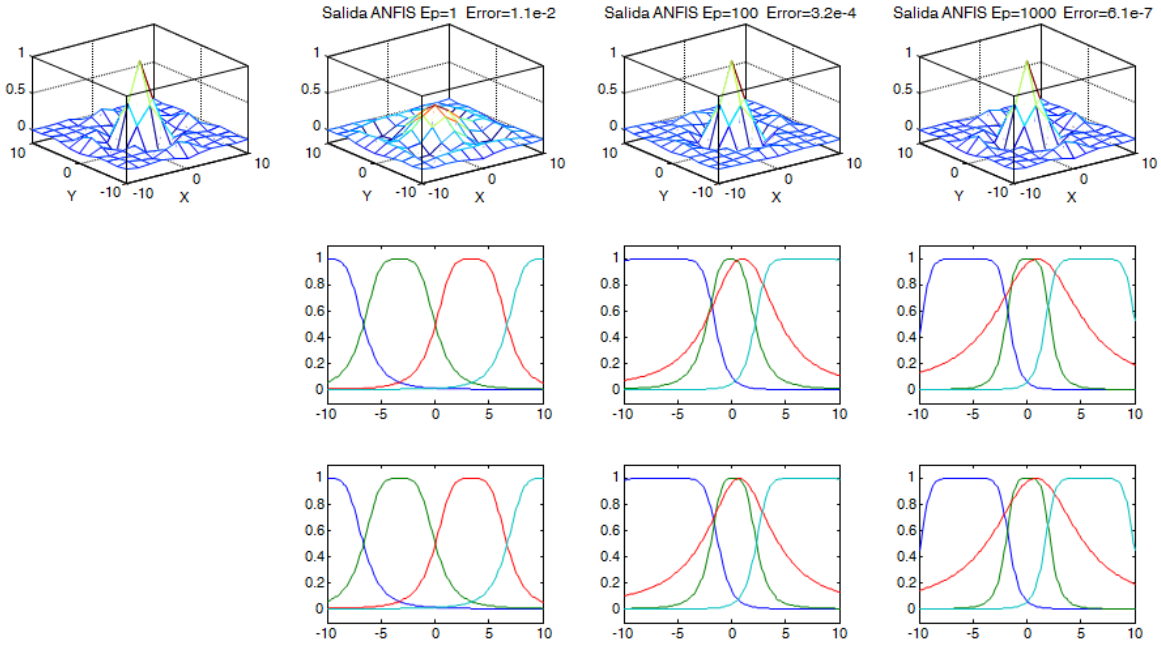


Figura 2.9.e Modelo ANFIS utilizando tres conjuntos difusos del tipo triángulo.

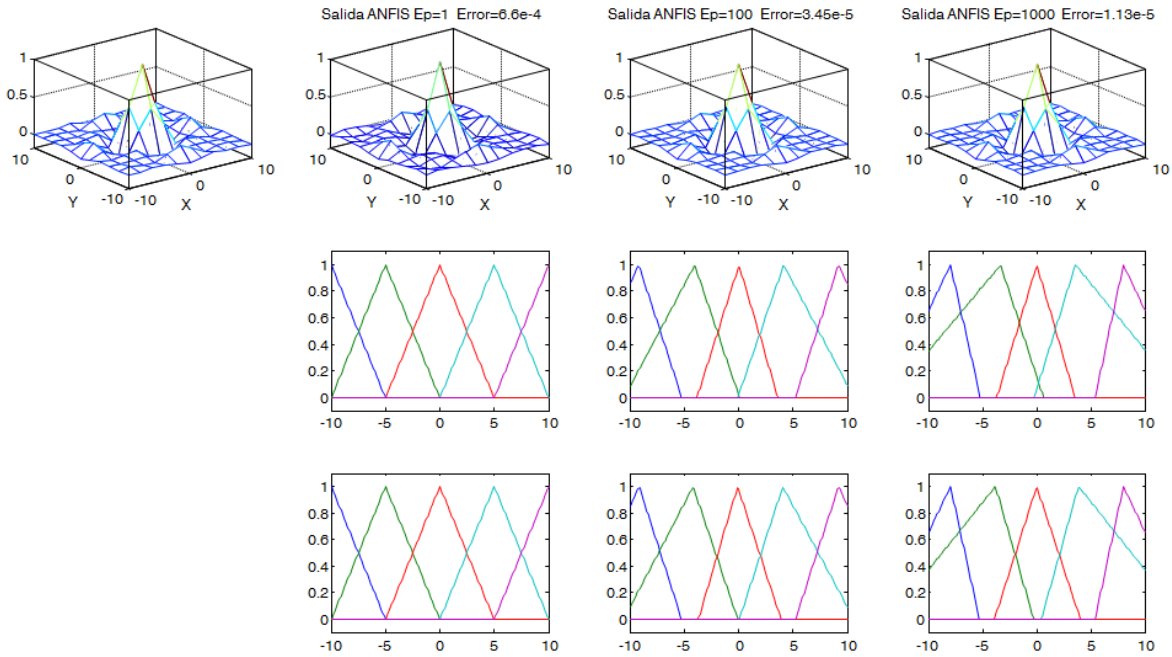


Figura 2.9.f Modelo ANFIS utilizando cinco conjuntos difusos del tipo triángulo.

La tabla 2.1, muestra el resumen del valor de errores obtenidos en de las diferentes pruebas que se realizaron en este análisis. Donde podemos concluir que al usar la función campana generalizada se puede llegar a una mejor aproximación. También, es notorio que cuando se usan cuatro funciones de membresía tipo campana generalizada el poder de aproximación de ANFIS es superior a los demás casos. Por lo que se puede concluir que no necesariamente el aumentar el número de conjuntos difusos, nos lleva a una mejor aproximación, lo que indica que el mejor procedimiento para entrenar un sistema ANFIS es el que hemos desarrollado, es generar varias posibilidades cambiando el número de conjuntos difusos hasta obtener el mejor resultado.

Tabla 2.1 Errores de la aproximación de ANFIS a la función sinc(x,t)

Figura	Número de MFS	Tipo de MF	Error			
			Ep=1	Ep=100	Ep=1000	
2.9 a	3	Gbell	1.44E-02	1.34E-03	9.02E-04	1-11-16
2.9 b	4	Gbell	1.10E-02	3.20E-04	6.10E-07	1-34-18068
2.9 c	5	Gbell	6.79E-04	3.45E-05	1.39E-05	1-20-49
2.9 d	3	Triangular	8.16E-03	7.15E-03	6.32E-03	1-1-1
2.9 e	4	Triangular	9.03E-03	1.40E-03	9.84E-05	1-6-92
2.9 f	5	Triangular	6.67E-04	3.46E-05	1.14E-05	1-19-59

La figura 2.10 muestra las curvas del error de ANFIS para diferentes épocas de entrenamiento. Como se observa, de manera general después de 100 épocas de entrenamiento, el modelo converge al mejor resultado. Sólo en el caso donde se obtuvo el mejor resultado, cuatro funciones de membresía tipo campana generalizado, el modelo convergía al mejor resultado aproximadamente en la época 500.

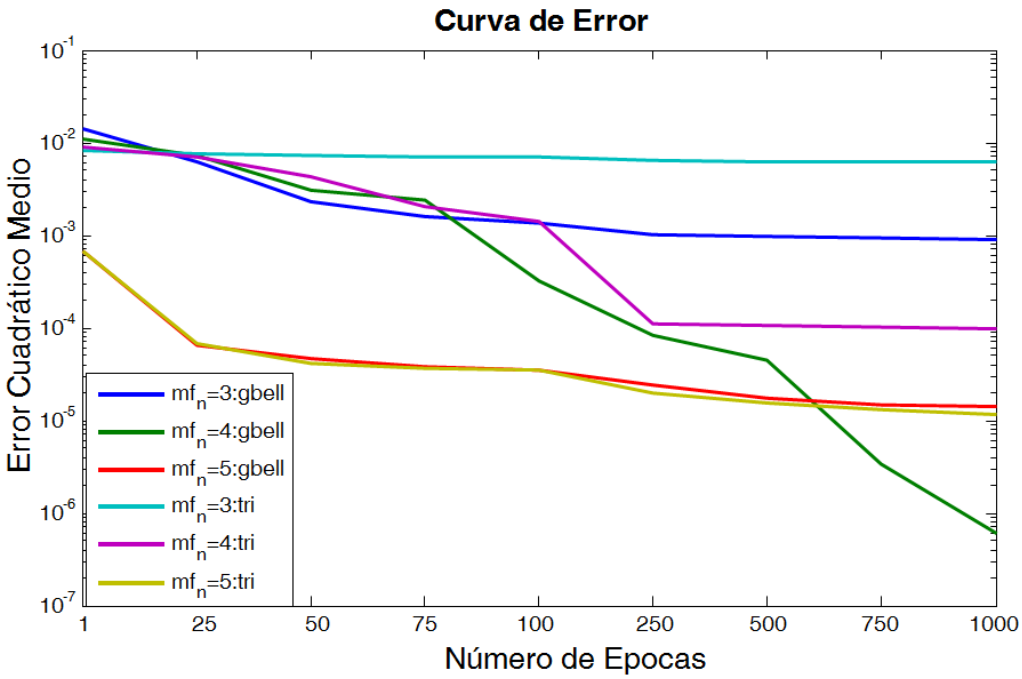


Figura 2.10 Gráfica de comparación del error de ANFIS para diferentes funciones de membresía.

Ejemplo 2: Modelado de una función no lineal de tres entradas

El conjunto de entrenamiento utilizado en este ejemplo fue obtenido de la ecuación no lineal deseada de tres entradas definida como:

$$output = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2$$

De acuerdo a Jang, esta ecuación fue también utilizada por Takaji an Hayashi, Sugeno and Kang y Komdo para probar sus modelos. Aquí la arquitectura ANFIS (ver la figura. 2.11), utiliza 8, 24, 64 y 125 reglas correspondiente a la combinación de 2, 3, 4 y 5 conjuntos difusos asignadas a cada variable de entrada. Un total de 256 datos de entrenamiento y 125 datos para el conjunto de prueba fueron utilizados y repartidos uniformemente, con los rangos de entrada $[1,6]^3$ y $[1.5,5.5]^3$ respectivamente. El conjunto de datos de entrenamiento fue utilizado para entrenar a los parámetros de ANFIS, es decir el proceso de identificación de ANFIS, mientras que el conjunto prueba se utiliza para verificar el ANFIS ya identificado. Para fines de comparativos, se utilizó el mismo índice de rendimiento adoptado diversas referencias [**2**].

$$APE = Porcentaje\ de\ Error\ Promedio = \frac{1}{p}\sum_{i=1}^{P}\frac{|T(i) - O(i)|}{|T(i)|}.100\%$$

Donde APE se define como la razón entre el valor absoluto de la diferencia de la función target menos la salida del sistema difuso ANFIS, entre el valor absoluto del sistema objetivo.

Las figuras 2.11 y 2.12 ilustran las funciones de membresía antes (columna 1) y después del proceso de entrenamiento (columnas 2-4) de cada variable de entrada. Las curvas de error en función del conjunto de entrenamiento y el conjunto de prueba en el proceso de entrenamiento para 200 épocas de entrenamiento se muestran en la columna 4.

Las gráficas de error muestran que se obtuvo una mejor aproximación cuando el modelo utiliza dos funciones de membresía, además que utilizando tres funciones de membresía el proceso ANFIS realiza un cambio evidente de los conjuntos difusos. Sin embargo, para más de 3 funciones de membresía el modelo prácticamente no cambien la forma y distribución de los conjuntos difusos, incluso en estos casos el error de testeo es inestable o crece exponencialmente.

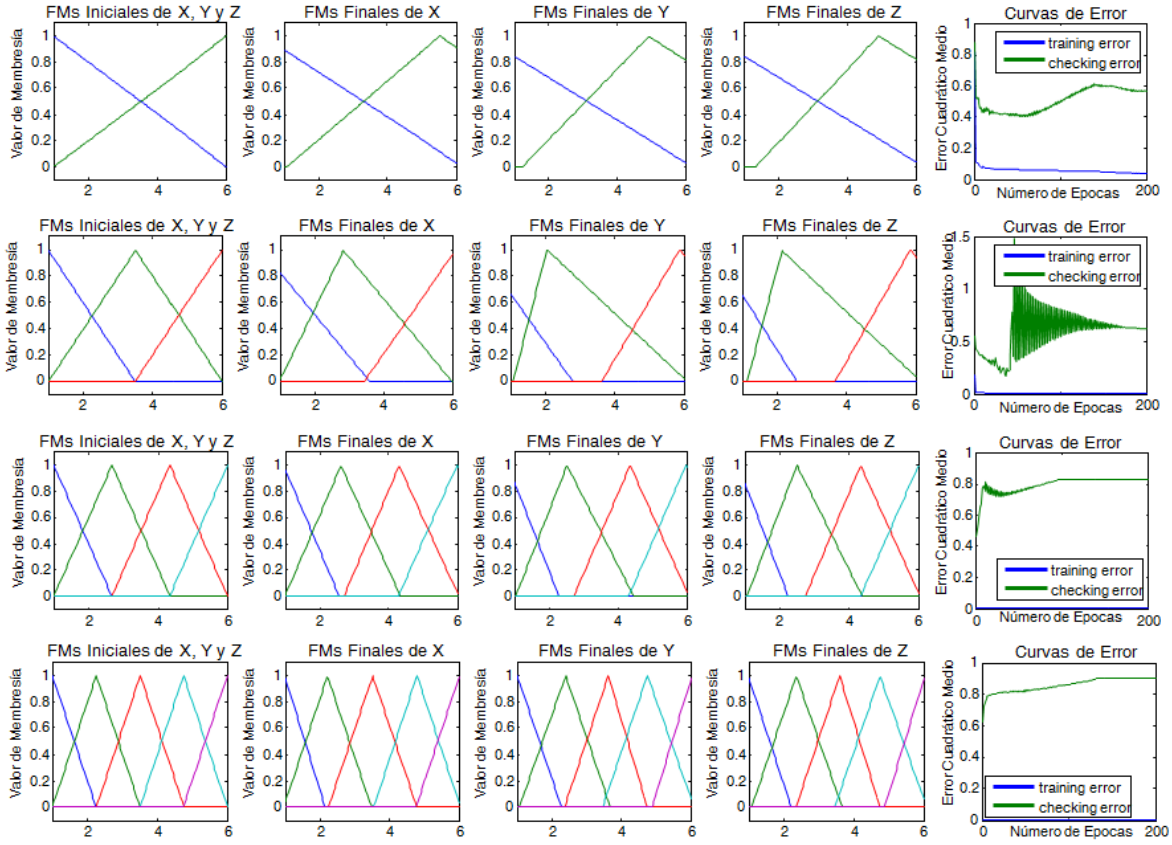


Figura 2.11 Funciones de membresía triangulares en el proceso de entrenamiento de $(1 + x^{0.5} + y^{-1} + z^{-1.5})^2$.

Para obtener una mejor capacidad de aproximación de ANFIS, es posible remplazar los nodos fijos que implementan al operador t-normas producto algebraico, por nodos adaptativos que implementen intersecciones paramétricas. Sin embargo, debe considerarse que este cambio afecta el proceso de entrenamiento, principalmente en el ajuste de los parámetros de los nodos antecedentes.

En el siguiente capítulo, se presentan las metodologías de generación de la operación de intersección definidas como funciones paramétricas, que tienen como característica principal, su implementación con poca complejidad matemáticas, y por ende una implementación digital en

hardware que consume pocos recursos. En los siguientes capítulos se presenta la metodología de entrenamiento del ANFIS-Sugeno con operadores de intersección paramétrica.

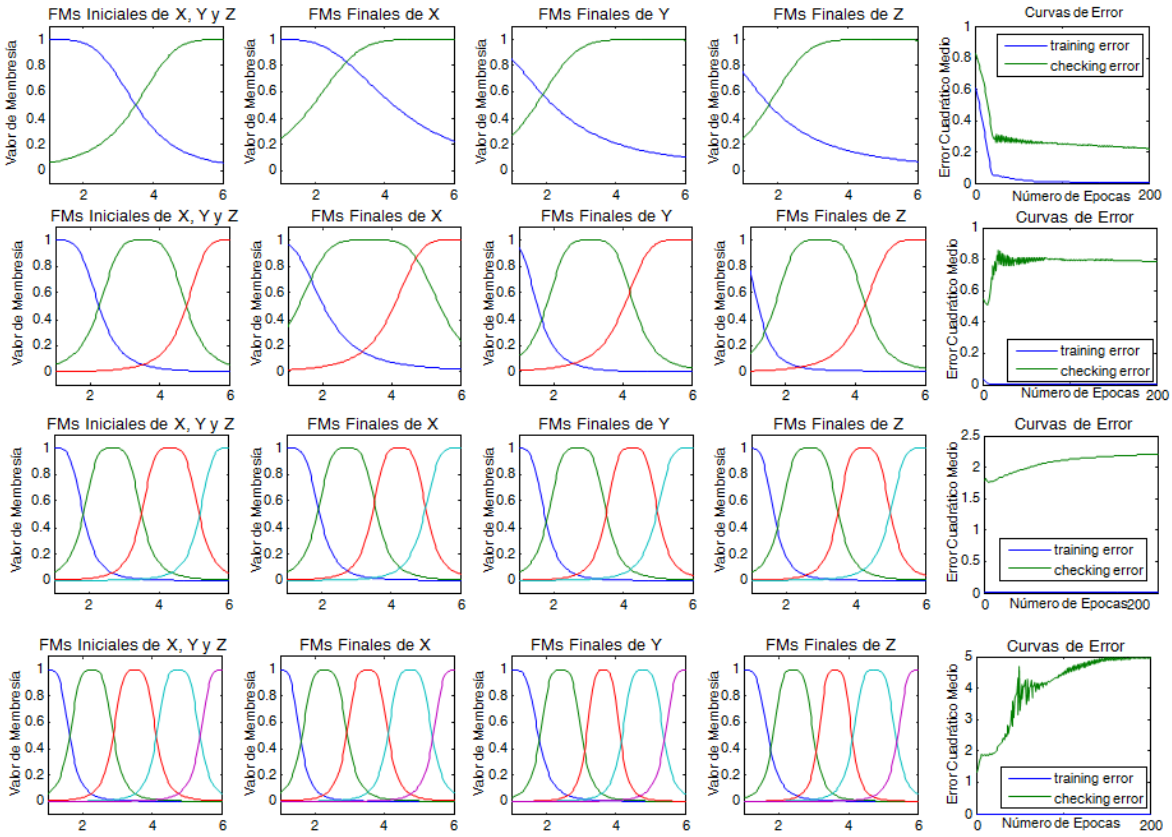


Figura 2.12 Funciones de membresía campana en el proceso de entrenamiento de $(1 + x^{0.5} + y^{-1} + z^{-1.5})^2$.

Otros temas importantes en el proceso de entrenamiento de ANFIS que no se tratarán en este trabajo son: a) requerir la característica de *e-completitud*; b) La característica de *fuzificación* moderada cuando se requiere que dentro de la mayoría de las regiones de entrada, exista una regla de inferencia difusa dominante; c) la identificación de la estructura de ANFIS; d) una efectiva selección del método de partición del espacio de entrada, con el objetivo de disminuir el número de reglas; e) aumentar la velocidad de las fases de entrenamiento y aplicación de ANFIS.

2.5. Metaheurísticas: algoritmos de búsqueda

En los problemas matemáticos y de ingeniería aplicados a problemas reales, donde la búsqueda de una solución óptima a través de un método de optimización utilizando el gradiente (para funciones derivables), o la búsqueda exhaustiva no son posible o imprácticos. Los algoritmos de optimización utilizando metaheurísticas tales como algoritmos genéticos, algoritmos de distribución de estimación, algoritmos de evolución diferencial, algoritmos de optimización de enjambre de partículas, algoritmos de optimización de colonia de hormigas, entre otros, han sido ampliamente

aceptados y son utilizados en problemas de optimización en áreas de ingeniería, economía biotecnología, por mencionar algunos [**24**,**25**]

El artículo del algoritmo de evolución diferencial [**26**] presenta un diseño poblacional muy simple que utiliza únicamente tres parámetros de configuración con operaciones básicas de suma, resta, multiplicación y comparadores, con un rendimiento de optimización comparable o incluso superior a otros algoritmos evolutivos o metaheurísticas de optimización en problemas con variables de representación real.

2.5.1. Algoritmo de evolución diferencial

El algoritmo de evolución diferencial, también referenciado como DEA por siglas en inglés *differential evolution algorithm*, pertenece a la familia de algoritmos evolutivos, el cual tiene como objetivo encontrar una solución óptima global de una función en un espacio continuo. En particular, y sin pérdida de generalidad, este problema puede reducirse al problema de encontrar el mínimo global de una función, que puede expresarse como:

$$mínimo f(x) = f(x_1, x_1, \dots, x_n)$$

Donde *x* es in vector *n*-dimensional y *f* es una función escalar con representación real. El DEA propuesto en [**26**], es un algoritmo que utiliza únicamente tres parámetros de configuración, que son: 1.) el parámetro *CR*, para realizar la cruza de diferentes soluciones y la mutación de la solución de prueba, 2.) el parámetro *F*, utilizado con factor de escalamiento de la diferencia entre dos individuos y 3. El parámetro *NP* para determinar el tamaño de la población.

En el DEA se permiten las mismas las condiciones de paro que otros algoritmos metaheurísticos, que son, número máximo de generaciones o iteraciones, el tiempo algorítmico, diferencia entre el error secuencial, alcance de un error mínimo.

En el algoritmo de evolución diferencial, se genera una solución de prueba de la información de tres soluciones de población, *v*, se compara la aptitud de la prueba *f(v)*, y si la aptitud de *f(v)* es mejor que de la solución analizada $f(x_i)$, entonces la solución prueba *v* sustituye a la solución x_i. Este proceso se repite para toda la población mientras la condición de paro no haya sido satisfecha.

La figura 2.13 muestra el algoritmo de evolución diferencial y la figura 2.14 muestra el proceso de generación de un vector de prueba, donde se tienen una población de 10 individuos, y seleccionados 4 soluciones aleatoriamente generándose la solución *v*, según la ecuación presentada en la línea 12 del algoritmo de la figura 2.13.

```
1  Definir los parámetros del AED; tamaño de la población NP, el número de generaciones
   g, la probabilidad de Cruza-Mutación Cr y el valor del factor de escalamiento: F
2  Crear aleatoriamente una población inicial P de soluciones
3  Evaluar la función de aptitud (Error Cuadrático medio ) de cada individuo de la
   población y determinar el MINIMO de aptitud
4  para G = 1 hasta MaxGeneración hacer
5      para i = 1 hasta NP hacer
6          ## Seleccionar aleatoriamente tres índices de la población actual
7          r1, r2, r3 ∈ NP  r1 ≠ r2 ≠ r3 ≠ i
8          jrand=randInt[1:D]
9          ##Generar solución de prueba v
10         para j = 1 hasta D do
11             si (rand[0,1] < CRorj == jrand) entonces
12                 v_{i,j} = x_{i,r1} + F * (x_{i,r2} − x_{i,r3})
13             en otro caso
14                 v_{i,j} = x_{i,j}
15             fin
16         fin
17         si (f(v) < f(x_i)) entonces
18             x_i = v
19             f(x_i) = f(v)
20         end
21         si (f(v) < MINIMO) entonces
22             MINIMO = f(v)
23         fin
24     fin
25 fin
```

Figura 2.13 Pseudocódigo del algoritmo de evolución diferencial

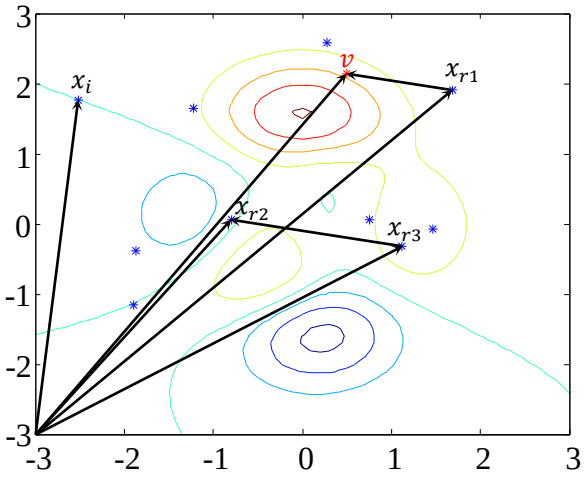


Figura 2. 14 Obtención del vector de prueba del DEA.

2.7. Referencias

1. Zadeh, L.: Outline of a new approach to the analysis of complex systems and decision processes. Systems, Man and Cybernetics, IEEE Transactions on(1), 28-44 (1973)
2. Jang, J.-S., Sun, C.-T., Mizutani, E.: Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence. (1997)
3. Ross, T.: Fuzzy logic with engineering applications. John Wiley \& Sons (2009)
4. Yen, J., Langari, R.: Fuzzy logic: intelligence, control, and information. Prentice-Hall, Inc. (1998)
5. Batyrshin I., K.: Parametric classes of generalized conjunction and disjunction operations for fuzzy modeling. IEEE Transactions on Fuzzy Systems 7(5), 586-596 (1999).
6. Batyrshin I., K.: Fuzzy modeling based on generalized conjunction operations. IEEE Transactions on Fuzzy Systems 10(5), 678-683 (2002).
7. Zadeh, L.: Fuzzy sets. Information and control 8(3), 338-353 (1965)
8. Klement, E., Mesiar, R., Pap, E.: Triangular norms 8. Springer Science \& Business Media (2013)
9. Butnariu, D., Klement, E.: Triangular norm-based measures and games with fuzzy coalitions 10. Springer Science \& Business Media (2013)
10. Bikbulatov A., B.: Tuning of operations in fuzzy models by neural nets. 7th Zittau Fuzzy Colloquium, 142-147 (1999).
11. P.D., K.-H.: Fuzzy operations' parameters versus membership functions' parameters influence on fuzzy control systems properties., vol. 1, pp.219-224 (2004).
12. Batyrshin I. Z., R.: On generation of digital fuzzy parametric conjunctions. Studies in Computational Intelligence 243, 79-89 (2009).
13. Batyrshin I. Z., R.: On the monotone sum of basic t-norms in the construction of parametric families of digital conjunctors for fuzzy systems with reconfigurable logic. Knowledge-Based Systems 38, 27-36 (2013).
14. Zilouchian, A., Jamshidi, M.: Intelligent control systems using soft computing methodologies. CRC Press, Inc. (2000)
15. Maimon, O., Rokach, L.: Soft computing for knowledge discovery and data mining. Springer Science \& Business Media (2007)
16. Apolloni, B., Pedrycz, W., Bassis, S., Malchiodi, D.: The puzzle of granular computing 138. Springer (2008)
17. Pedrycz, W., Skowron, A., Kreinovich, V.: Handbook of granular computing. John Wiley \& Sons (2008)
18. Fausett, L.: Fundamentals of neural networks: architectures, algorithms, and applications. Prentice-Hall, Inc. (1994)
19. Hagan, M., Demuth, H., Beale, M., others: Neural network design. Pws Pub. Boston (1996)
20. Werbos, P.: Beyond regression: New tools for prediction and analysis in the behavioral sciences. (1974)

21. Parker, D.: Learning logic. (1985)
22. Rumelhart, D., Hinton, G., Williams, R.: Learning internal representation by back propagation. Parallel distributed processing: exploration in the microstructure of cognition 1 (1986)
23. Jang, J.-S.: ANFIS: adaptive-network-based fuzzy inference system. Systems, Man and Cybernetics, IEEE Transactions on 23(3), 665-685 (1993)
24. Glover, F., Kochenberger, G.: Handbook of metaheuristics. Springer Science \& Business Media (2003)
25. Osman, I., Kelly, J.: Meta-heuristics: an overview. In : Meta-Heuristics. Springer (1996) 1-21
26. Storn R., P.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341-359 (1997)
27. Omondi, A., Rajapakse, J.: FPGA implementations of neural networks 365. Springer (2006)
28. Bryson, A.: Applied optimal control: optimization, estimation and control. CRC Press (1975)

Capítulo 3. Desarrollo de metodologías generadoras de operadores paramétricos de conjunción difusa

3.1. Introducción

En este capítulo se presentan las bases teóricas para la generación de conjunciones difusas paramétricas propuestas en este trabajo de tesis, que pueden ser implementadas por funciones matemáticas tales como las normas triangulares (t-normas) o los conjunctores, y que han sido utilizadas en la lógica difusa, como una extensión de la operación de *intersección*, la cual considera valores de verdad continuos. De manera particular se presentan las bases teóricas de las metodologías: a) suma monotónica de t-normas básicas para generar conjunctores paramétricos y; b) suma ordinal de t-normas básicas para generar t-normas paramétricas. Se presenta también una novedosa propuesta de generación de t-normas paramétricas basadas en la suma ordinal de t-normas básicas y la t-norma drástica en dominios que pueden ser recorridos.

Retomando los conceptos de conjunción, t-normas y conjunctores presentados en el capítulo anterior, podemos enunciar los siguientes conceptos fundamentales que se trabajan en el presente capítulo.

Conjunción difusa es una operación que extiende la operación de conjunción en la lógica clásica al dominio difuso, donde son considerados, valores de verdad continuos que pertenecen al intervalo unitario [0,1]. Estas pueden ser generadas mediantes funciones matemáticas tales como normas triangulares, conjunctores y pueden ser paramétricas y digitales.

Conjunción difusa paramétrica es una conjunción difusa que está en función de sus entradas y de uno o más parámetros configurables.

Conjunción difusa paramétrica digital. Es una conjunción difusa paramétrica definida mediante un intervalo de números enteros que mapean al espacio difuso [0,1] a un intervalo discreto definido de [0, n] donde $\boldsymbol{I}=2^m-1$ y m es el número de bits para representar un número dentro de este intervalo.

La presentación de las metodologías de generación de operadores paramétricos de conjunción difusa (o intersección difusa) será realizada en el dominio digital, debido a que una de las motivaciones principales de este trabajo es desarrollar operadores paramétricos de conjunción simples, que sean descritos con operaciones matemática que no involucren funciones exponenciales, logaritmos, divisiones o funciones de similar o mayor complejidad, con el objetivo de lograr una implementación eficiente en el diseño digital en FPGAs. Sin embargo, debe tenerse claro que estas metodologías y operadores presentados son igualmente validos en el domino difuso [0,1].

3.1.1. Conjunctores y t-normas digitales

Considere el conjunto de números enteros $L = \{0, 1, \ldots, n\}$, con $n \geq 1$, que define la representación digital del conjunto de valores de verdad (grados, membresía, etc.). Por lo que para una representación digital de enteros de *m*-bits se tiene que $n \leq 2^m - 1$. Se considerarán las funciones conjunctores y t-normas definidas en el conjunto L con un valor máximo del conjunto $n = \mathbf{I}$, que corresponde al valor 1 del conjunto tradicional de valores de verdad [0, 1]. Considere la función $T: L \times L \to L$ con las siguientes posibles propiedades en L:

A1. $T(x, \mathbf{I}) = x, \quad T(\mathbf{I}, y) = y,$ (condiciones de frontera)

A2. $T(x, y) \leq T(u, v), \quad if \ x \leq u, y \leq v.$ (monotonicidad)

A3. $T(x, y) = T(y, x),$ (conmutatividad)

A4. $T(x, T(y, z)) = T(T(x, y), z).$ (asociatividad)

A5. $T(x, y) \leq min(x, y).$ (condición de rango)

Esta función será referenciada como una **conjunción**, una **conjunción conmutativa**, una **t-norma** o una **t-subnorma** digital si las propiedades: {A1, A2}, {A1–A3}, {A1–A4}, {A2–A5} correspondientemente son satisfechas para todo $x, y, z \in L$. Ver las propiedades de estas operaciones en [**1**,**2**,**3**,**4**,**5**] tanto para el caso del intervalo continuo [0,1] como en el conjunto discreto $L = \{0,1,\ldots, n\}$.

Es evidente que de A1, A2, cualquier conjunción cumple para $x, y \in L$:

$T(x, 0) = 0, \quad T(0, y) = 0.$

Desde el punto de vista formal, es posible considerar conjunctores y t-normas definidas en un conjunto L de un sólo elemento $L = \{0\}$. En este caso tenemos $\mathbf{I} = 0$, sin embargo estos no tienen un sentido semántico. A continuación se listan t-normas simples que serán consideradas como t-normas básicas para la generación de conjunctores y t-normas digitales paramétricas difusas.

$T_M(x,y) = min\{x,y\},$ (mínimo)

$$T_N(x,y) = \begin{cases} \min(x,y), & if \ x + y > \mathbf{I} \\ 0, & otherwise \end{cases}$$ (nilpotente)

$T_L(x,y) = max\{x+y-\mathbf{I}, 0\},$ (Lukasiewicz)

$$T_D(x,y) = \begin{cases} x, & if \ y = \mathbf{I} \\ y, & if \ x = \mathbf{I} \\ 0, & if \ x, y < \mathbf{I} \end{cases}$$ (Producto drástico)

$T_P(x,y) = x * y / \mathbf{I}$ (Producto)

Estas t-normas presentan una implementación eficiente en FPGA debido a que en su definición utilizan operaciones matemáticas y lógicas simples. La figura 3.1 muestra las superficies de estas t-normas definidas en $L = \{0,1,\ldots,31\}$.

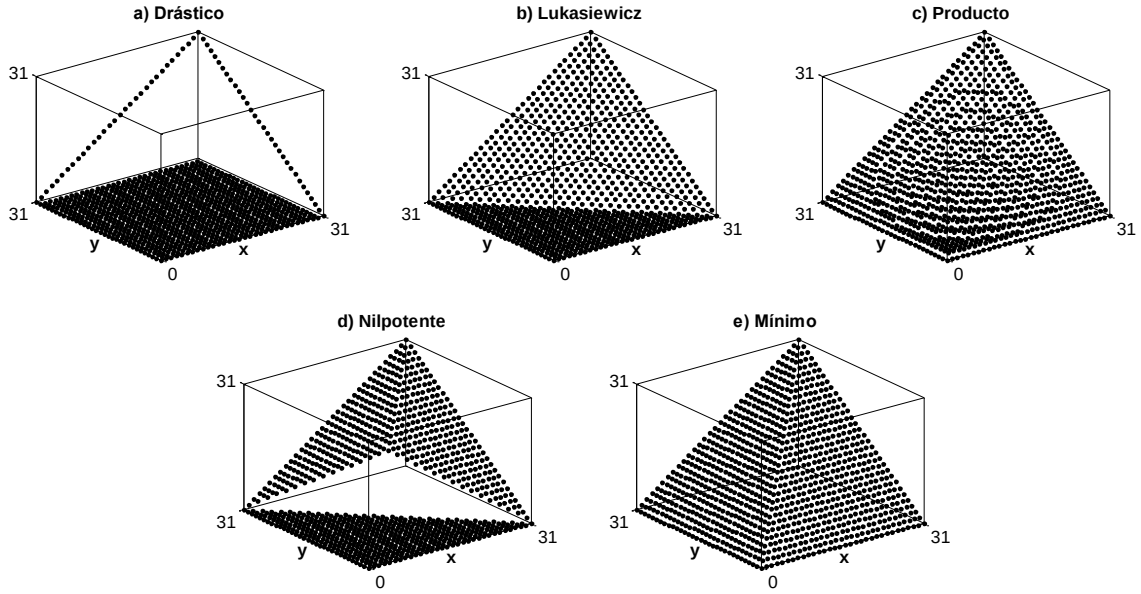


Figura 3.1 T-normas básicas digitales: a) Drástico, b) Lukasiewicz, c) Producto, d) Nilpotente y e) Mínimo.

La función producto tiene una implementación en hardware más costosa tanto matemáticamente, como en recursos utilizados de hardware comparada con las otras t-normas básicas, además la versión digital de la t-norma producto no es realmente asociativa y es únicamente considerada en la generación de conjunctores, ya que esta puede definirse únicamente como una aproximación de la t-norma producto.

Se define la siguiente relación. Sea la t-norma T_1 menor o igual a la t-norma T_2, $T_1 \leq T_2$, cuando $T_1(x,y) \leq T_2(x,y)$ para todo x,y que pertenece a L. A partir de la relación anterior y de las definiciones de conjunctor, t-normas se puede observar (o demostrar) que:

$$T_D \leq T \leq T_M, \qquad T_D \leq T_L \leq T_N \leq T_M, \qquad T_D \leq T_L \leq T_P \leq T_M.$$

De la condición $T \leq T_M$ se observa que toda t-norma es una t-subnorma. Nota que las t-normas T_N y T_P no satisfacen las relación de menor o igual para todo todo x, y que pertenece a L.

3.1.2. Construción de conjunctores and t-normas paramétricas digitales

Sea $L = \{0, 1, \ldots, n\}$ un conjunto de valores enteros y $a, b \in L$, tal que $a \leq b$, la secuencia de números consecutivos cambiando de a hasta b es definida como un intervalo cerrado, y se denotada como $[a, b]$ y $p \in \{0, 1, \ldots, n\}$ un parámetro entero. Cuando $p \geq 1$ el intervalo L se divide en dos 2 intervalos: $X_1 = [0, p\text{-}1]$, $X_2 = [p, n]$, por lo que el espacio $L \times L$ se divide en 4 sectores: $D_1 = X_1 \times X_1$, $D_2 = X_2 \times X_1$, $D_3 = X_1 \times X_2$, $D_4 = X_2 \times X_2$, como ilustra en la figura 3.2.a, donde $n = 15$. Note que cuando

$p = 0$ se tiene únicamente el intervalo $X_2 = [0,n]$ y un único sector $D_4 = X_2 \times X_2 = L \times L$ que coincide con el dominio completo $L \times L$. En este caso los otros sectores D_1, D_2, D_3 están vacíos.

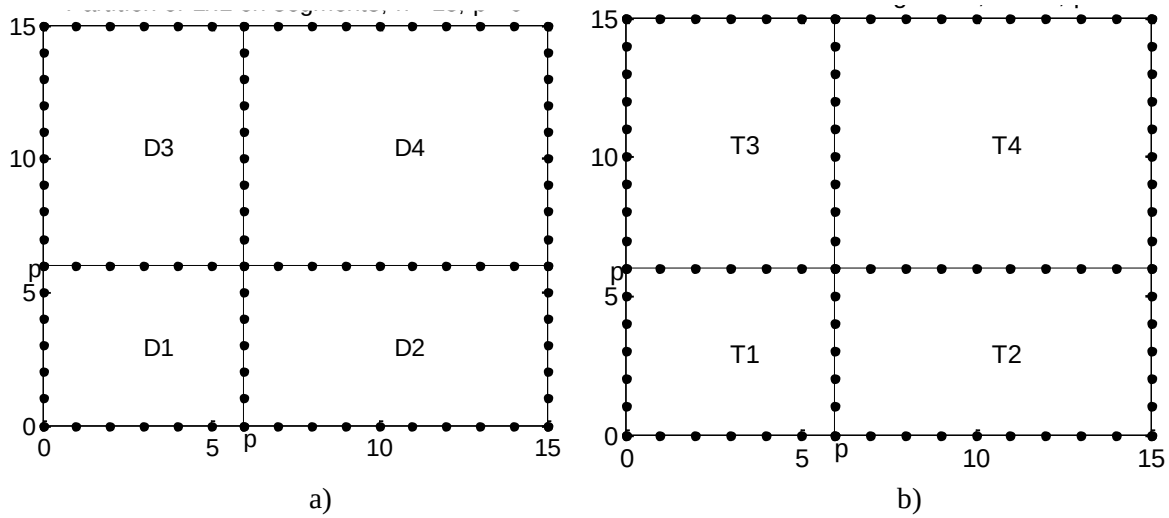


Figura 3.2 a) Partición del dominio *LxL* en segmentos definidos por el parámetro p. b) Método 4T para la construcción de conjunctores.

Basados en la división del espacio $L \times L$ mediante el parámetro p y de las t-normas básicas T_i asociadas con los sectores D_i, con $i \in \{1,2,3,4\}$ son presentados los métodos de construcción de conjunctores y t-normas en el dominio $L \times L$. Los métodos presentados en este documento son casos particulares de los métodos generales discutidos en [**1**,**2**,**3**,**4**,**5**]. Para simplificar las referencias a estos métodos se definen nombres cortos para referirnos a ellos [**6**].

Método 4T. Este método puede utilizar cuatro diferentes t-normas básicas. Sea la t-norma T_i, donde $i \in \{1, 2, 3, 4\}$, definida en $L = \{0,1, \ldots, n\}$, tal que

$T_1 \leq T_2 \leq T_4$, $T_1 \leq T_3 \leq T_4$.

Entonces la función $T{:}L \times L \to L$ definida por la ecuación

$$T(x,y) = \begin{cases} T_1(x,y), & if\ x,y < p \\ T_2(x,y), & if\ y < p, p \leq x, \\ T_3(x,y), & if\ x < p, p \leq y \\ T_4(x,y), & if\ p \leq x,y \end{cases},$$

Es un conjunctor, donde $p \in \{0, 1, \ldots, n\}$ es un parámetro. La figura 3.2.b muestra la asignación de las t-normas a los segmentos D_1-D_4 en la región $L \times L$ para la construcción del conjunctor $T(x,y)$ mediante el método 4T. **El conjunctor *T(x, y)* es conmutativo** si todos los T_i son conmutativos y $T_2 = T_3$. Hay que notar que cuando p=0 el conjunctor T es igual a T_4 definido en todo el dominio $L \times L$. Por otro lado, cuando p=n, el conjunctor resultante T es igual a T_1 ya que T utiliza únicamente los valores frontera de las t-normas T_2, T_3 and T_4 según la condición **A1.**

El método de generación de conjunctores 4T está basado en la suma monotónica de t-normas básicas presentado en [**7**,**5**]. En este trabajo se ha referido a este método como 4T en lugar de (T_1, T_2, T_3, T_4) utilizado en [**5**] y se utiliza la notación **4Tc** para definir **conjunctores conmutativos**. En

[**5**] también se observó que los conjunctores conmutativos (*T, M, M, M*), (*D, D, D, T*), (*T, M, M,T*) cumplen la propiedad asociativa, por lo que estos conjunctores también son t-normas. La notación corta de estas t-normas es T3M, 3DT, TMMT respectivamente.

Método 3DT. Suponga que la t-norma T_D está definida en el L= {0,1,…, n}, $p \in \{0,1,\ldots, n\}$ y la t-norma T_4 está definida en L_4= {0,1,…, *n-p*}, entonces para todo $x, y \in L = \{0,1,\ldots,n\}$ la siguiente función define una t-norma en L:

$$T(x,y) = \begin{cases} p + T_4(x-p, y-p), & if\ \ p \le x, y \\ T_D(x,y), & otherwise \end{cases}.$$

El método 3DT está basado en la proposición 3 de [**5**]. La figura 3.3a muestra la asignación de las t-normas básicas en los segmentos D_i de $L \times L$ de éste método. Note que la t-norma T_4 ha mapeado su dominio al dominio de la sección D_4 más una compensación con valor *p*.

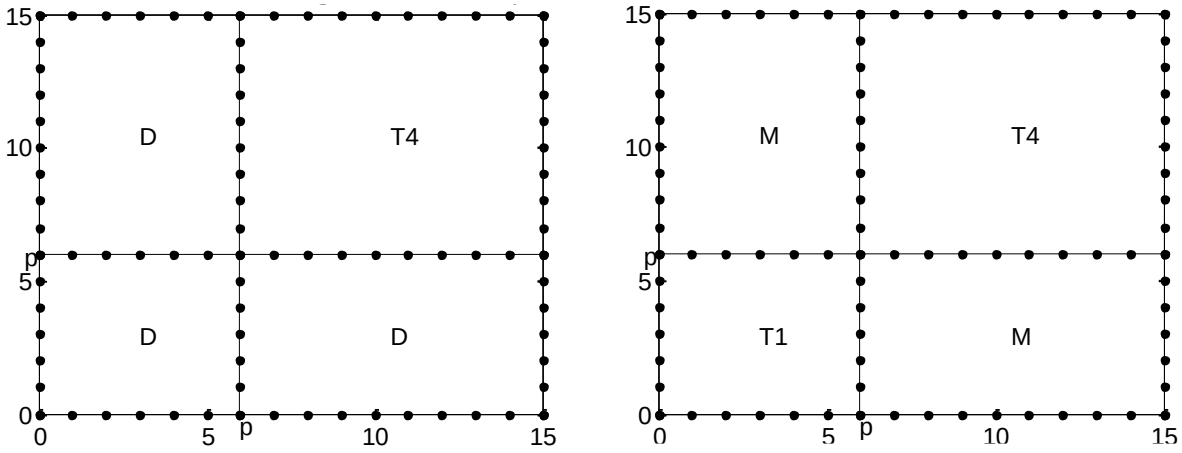


Figura 3.3 Métodos de construcción de t-normas a) 3DT and b) TMMT (suma ordinal).

Método TMMT. Sean $p \in \{0,1,\ldots,n\}$, T_1 es una t-norma definida en L_1= {0,1,…, p} y T_4 es una t-norma definida en L_4= {0,1,…, *n-p*}, entonces la siguiente función es una t-norma definida en L= {0,1,…,*n*}:

$$T(x,y) = \begin{cases} T_1(x,y) & if\ x, y < p \\ p + T_4(x-p, y-p) & if\ \ p \le x, y. \\ \min(x,y) & otherwise \end{cases}$$

Este método está basado en la suma ordinal de t-normas [**1**,**3**,**4**,**5**]. La figura. 3.3b muestra la ubicación de las t-normas utilizadas en este método en los segmentos $L \times L$. De la misma manera, los dominios de las t-normas T_1 y T_4 han sido mapeados de manera similar que en caso del método 3DT

Los siguientes métodos utilizan dos y tres parámetros de configuración de t-normas paramétricas y son una extensión de los métodos anteriormente presentados.

Método T3Ms. Suponer $p \in \{0,1,\ldots,n\}$ y $i_1 \in \{0,1,\ldots,2n\}$ son parámetros y T_1 es una a t-norma definida en $[0,i_1]$. La siguiente función define a la t-norma en $L = \{0,1,\ldots,n\}$:

$$T(x,y) = \begin{cases} T_1(x,y) & if\ x,y < \min(i_1,p) \\ \min(x,y) & otherwise \end{cases}.$$

El método T3Ms está basado en la suma ordinal de t-subnormas [**2**] y de la proposición 6 de [**5**] es referido como un método de construcción de t-normas a partir de t-normas básicas con dominios de corrimiento, la letra 's' incluida en el nombre del método T3Ms denota la palabra "*shifted*" (corrimiento). La figura 3.4 muestra la asignación de las t-normas básicas utilizadas en esté método en los segmentos de $L \times L$, para los casos cuando $i_1 > p$ y cuando $i_1 < p$. En el primer caso $i_1 > p$, aunque la t-norma T_1 esté definida en una región mayor al espacio D_1, los únicos valores que son integrados en la definición de la $T(x,y)$ son los que están dentro de la región D_1, por lo que las regiones D_2, D_3, D_4 se les asigna la t-norma T_M. En el segundo caso, $i_1 < p$, la t-norma T_1 se define en una región menor a D_1, por lo que los espacios D_2, D_3, D_4 son extendidos a $[i_1, n]$x$[0, i_1\text{-}1]$, $[0, i_1\text{-}1]$x$[i_1, n]$, $[i_1, n]$x$[i_1, n]$ respectivamente en lugar de [p, n]x[0, p-1], [0, p-1]x[p, n], [p, n]x[p, n]. Este mismo principio de extensión de dominios se utiliza en las siguientes metodologías que serán presentadas a continuación.

Notar que los dominios de las t-normas T_1 definidos en la suma monotónica de t-normas (método 4T), en la suma ordinal de t-normas (método TMMT) y en el método T3Ms generalmente son diferentes. En la suma monotónica de t-normas T_1 está definida en $L = \{0,1,\ldots,n\}$, mientras que en la suma ordinal de t-normas T_1 está definida sobre la "ventana" $L_1 = [0, p] \times [0, p]$, donde $p \in \{0,1,\ldots,n\}$, finalmente en el método T3Ms el dominio de la T_1 generalmente difiere de ambos dominios ya que está definida en el intervalo $[0,i_1]$. Desde este punto de vista, el dominio de T_1 en el método T3Ms es de "corrimiento" con respecto al dominio T_1 de los dos primeros métodos.

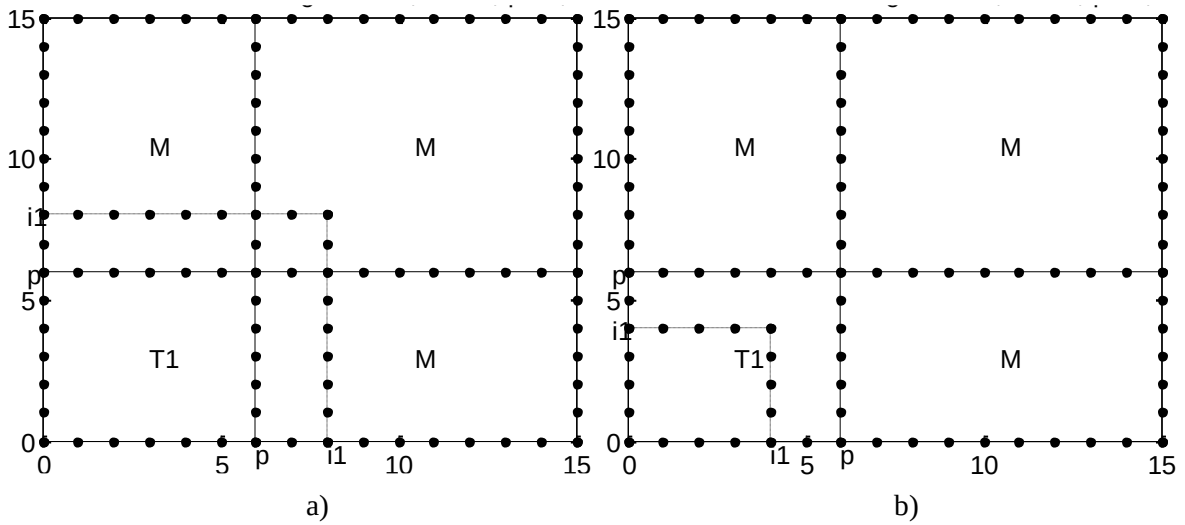


Figura 3.4 Método T3Ms, a) La t-norma T_1 está definida en $[0,i_1]$ y está sólo es visible dentro de la "ventana" $[0,p\text{-}1]$x$[0,p\text{-}1]$, b) La t-norma T_1 está definida en $[0,i_1]$ y los dominios D_2, D_3, D_4 son extendidos a $[i_1, n]$x$[0, i_1\text{-}1]$, $[0, i_1\text{-}1]$x$[i_1, n]$, $[i_1, n]$x$[i_1, n]$

Método 3DTs. Suponga que se definen la t-norma T_D en L= {0,1,…, n}, $p\in${0,1,…, n}, $i_4\in${0,1,…,2n} y la t-norma T_4 en L_4= {0,1,…, i_4}. Entonces para todo x, $y\in L$ = {0,…,n} la siguiente función es una t-norma definida en L:

$$T(x,y)=\begin{cases} p+T_4(x-p,y-p) & if\ \ p\ \le x,y<\min(n,p+i_4) \\ min(x,y) & if\ p+i_4\le x,y\le n \\ T_D(x,y) & otherwise \end{cases}.$$

La figura 3.5 muestra la asignación de las t-normas básicas utilizadas en método 3DTs en la región *LxL* cuando i_4 > a *I-p*

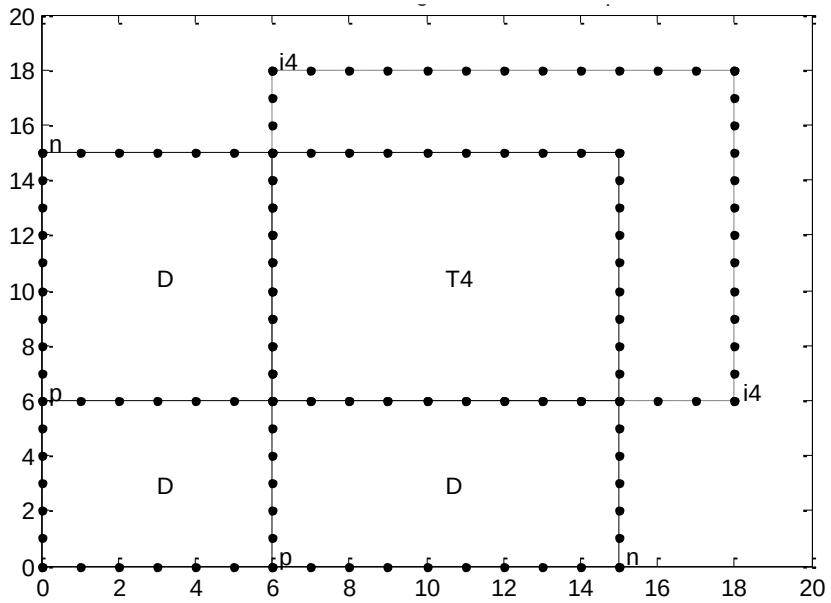


Figura 3.5 Método 3DTs. Sea la t-norma T_4 definida en [0,i_4], donde i_4=12. T_4 únicamente está "visible" en la ventana [p,n-1] x [p,n-1] = [6,14] x [6,14].

Método TMMTs. Sean $p\in${0,1,…,n}, $i_1\in${0,1,…,2n}, $i_4\in${0,1,…,2n}, la t-norma T_1 en L_1= {0,1,…, i_1} y T_4 una t-norma en L_4= {0,1,…,i_4}. Entonces la siguiente función es una t-norma en L= {0,1,2,…,n}:

$$T(x,y)=\begin{cases} T_1(x,y), & if\ x,y<\min(i_1,p) \\ p+T_4(x-p,y-p), & if\ \ p\le x,y<\min(p+i_4,n). \\ \min(x,y), & otherwise \end{cases}$$

Notar que este método TMMTs además de ser una extensión del método TMMT, también es una extensión de los métodos T3Ms.

En la tabla 3.1 se lista la configuración de las gráficas de superficie de diferentes operadores de intersección paramétrica digital con funcionalidad de conjunctores o t-normas generadas de los métodos descritos en este capítulo, con una representación de enteros de 5 bits, es decir para valores de verdad de [0,31]. La primera columna hace referencia a las gráficas de superficie mostradas en la

figura 3.6. Los parámetros que no son utilizados en alguna metodología se les asigna el símbolo X, que es interpretado como la condición de no importa, utilizado en la literatura de lógica digital [**8**].

Notar que ningún operador generado de la metodología T3Ms es incluido en esta tabla, ya que está metodología puede verse como un método particular de metodología TMMTs, cuando a la t-norma T_4 se le asigna la t-norma básica T_M.

Tabla 3.1 Configuración de familias de operadores de intersección

Figura	Metodología	t-normas básicas	P	i_1	i_4
a)	4T	$T_D, T_L T_N, T_M$	11	X	X
b)	4Tc	T_D, T_L, T_L, T_M	10	X	X
c)	3DT	T_P	11	X	X
d)	3DT	T_D	11	X	X
e)	3DTs	T_L	11	X	15
f)	3DTs	T_L	11	X	25
g)	TMMT	T_L, T_D	11	X	X
h)	TMMT	T_L, T_L	11	X	X
i)	TMMTs	T_1, T_4	31	22	X
j)	TMMTs	T_L, T_M	25	35	X
k)	TMMTs	T_L, T_N	11	15	25
l)	TMMTs	T_D, T_L	11	18	15

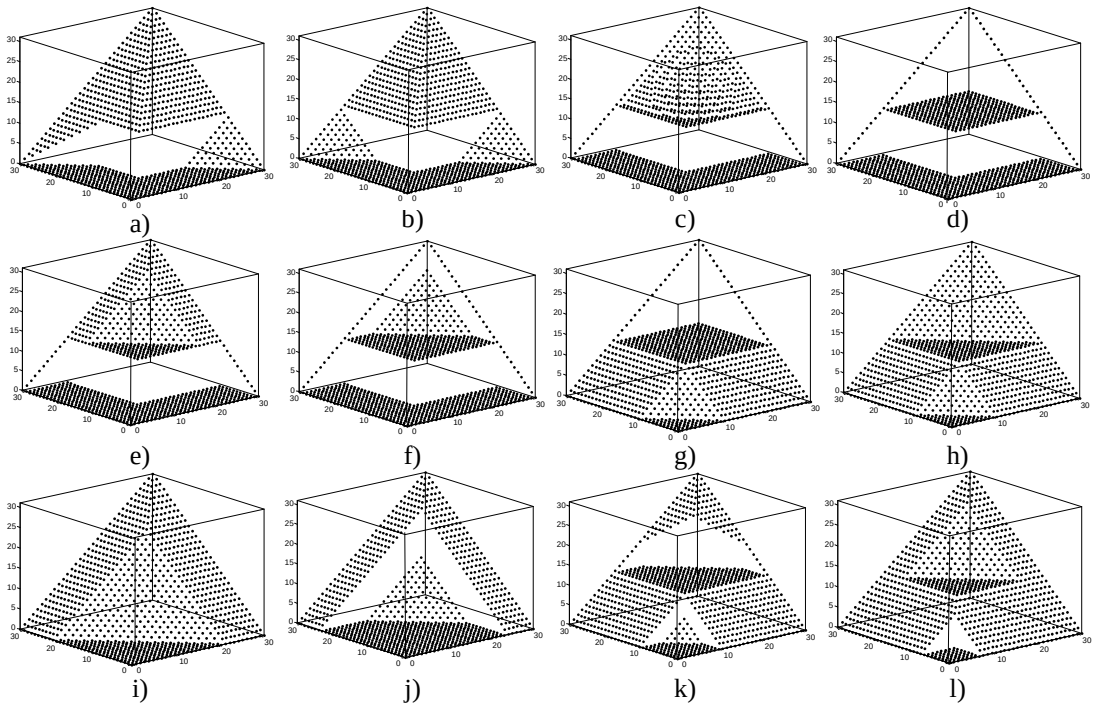


Figura 3.6 Superficies de las t-normas y conjunctores correspondientes a la tabla 3.1

Finalmente, la tabla 3.2 muestra el número de familias de conjunctores y t-normas generadas por cada metodología en función del número de t-normas básicas utilizadas, es decir, para 3 (discutidas en [**9**]), 4 y 5 t-normas básicas. Para los casos de las metodología generadoras de conjunctores, fue tomado en cuenta que las t-normas básicas T_P y T_N no pueden ser comparadas, por lo que ambas no pueden ser incluidas en la generación de un conjunctor. En el último renglón de esta tabla se indica el total de familias de conjunctores y t-normas paramétricas generadas por las metodologías presentadas en este capítulo, que es 41, 95 o 135, cuando 3,4,5 diferentes t-normas básicas son utilizadas.

Tabla 3.2. Número de familias de conjunctores y t-normas paramétricas por metodología

Metodología	Número de operadores cuando se usan *K* t-normas		
	K=3	K=4	K=5
4T (Conjunctor no commutativo)	10	30	50
4Tc (Conjunctor commutativo)	7	16	25
TMMT	9	16	25
TMMTs	9	25	25
3DT	3	4	5
3DTs	3	4	5
Total Operatores	41	95	135

En el siguiente capítulo se presentan la metodología de entrenamiento de estos operadores de intersección, cuando son integrados al modelo neurodifuso ANFIS, y finalmente en el capítulo 5 se presentan la implementación en hardware de las seis metodologías, además de que se introduce un diseño que integra las seis metodologías dentro de un operador generalizado que presenta tiempo de respuesta y consumo de hardware eficiente.

3.2. Referencias

1. Klement, E. P., Mesiar, R., Pap, E.: Triangular Norms (2000).
2. Jenei, S.: How to construct left-continuous triangular norms—state of the art. Fuzzy Sets and Systems 143(1), 27-45 (2004)
3. Mayor, G., Torrens, J.: 7 Triangular norms on discrete settings. Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms, 189 (2005)
4. Mayor, G., Monreal, J.: Additive generators of discrete conjunctive aggregation operations. Fuzzy Systems, IEEE Transactions on 15(6), 1046-1052 (2007)
5. Batyrshin, I., Rudas, I., Villa, L., Antonio, P.: On the monotone sum of basic t-norms in the construction of parametric families of digital conjunctors for fuzzy systems with reconfigurable logic. Knowledge-Based Systems 38, 27-36 (2013)
6. Cort, Batyrshin, I., Villa-Vargas, L., Rudas, I., Molina-Lozano, H., Ram: Hardware Design of Digital Parametric Conjunctors and t-Norms. International Journal of Fuzzy Systems 17(4), 559-576 (2015)
7. Batyrshin, I., Rudas, I., Panova, A.: On generation of digital fuzzy parametric conjunctions. In :

Towards Intelligent Engineering and Information Technology. Springer (2009) 79-89

8. Nelson, V., Nagle, H., Carroll, B., Irwin, J.: Digital logic circuit analysis and design. Prentice-Hall, Inc. (1995)
9. Cortés-Antonio P., B.: FPGA implementation of (p)-monotone sum of basic t-norms., Barcelona, Spain (August 2010)

Capítulo 4. Regla de aprendizaje del modelo ANFIS-Sugeno con operadores paramétricos de conjunción difusa

4.1. Introducción

En este capítulo se expone el algoritmo de optimización para la red ANFIS-Sugeno de primer orden, que contiene operadores de conjunción paramétricos, además de los parámetros antecedentes y consecuentes de la arquitectura tradicional de ANFIS.

En la primera parte de este capítulo se estudia la regla de aprendizaje de la red ANFIS tradicional, que utiliza los métodos de aprendizaje del gradiente de mayor descenso y el estimador de mínimos cuadrados. Posteriormente, se presenta la regla de aprendizaje para entrenar el sistema ANFIS-Sugeno de primer orden con operadores paramétricos de intersección, el cual puede utilizarse de dos maneras:

a) Fase 1. Se aplica al sistema ANFIS, la regla de aprendizaje tradicional, teniendo t-normas producto no paramétricas, para ajustar los parámetros antecedentes y los parámetros consecuentes. Fase 2. Una vez que se ha realizado esta fase de entrenamiento, se reemplazan los operadores t-normas producto por operadores de intersección paramétricos (los presentados en el capítulo anterior), y son entrenados iterativamente, utilizando el algoritmo de evolución diferencial y el estimador de mínimos cuadrados. Como se estará explicando detalladamente en este capítulo.
b) La otra manera de utilizar la regla de aprendizaje propuesta, es suponiendo que los parámetros que caracterizan a los conjuntos difusos no pueden ser modificados, y sólo aplicar la segunda fase del aprendizaje al sistema.

Finalmente, en el capítulo se presenta un análisis comparativo del poder de aproximación del sistema y regla de aprendizaje propuesto con los resultados de ANFIS tradicional.

4.2. Regla de aprendizaje híbrida para entrena la red ANFIS: combinación del algoritmo del gradiente de mayor descenso con el estimador de mínimos cuadrados

Aunque es posible aplicar el método del gradiente de mayor descenso, para identificar a todos los parámetros de una red adaptativa (algoritmo de retropropagación en redes neuronales), este método de optimización sencillo suele demorar mucho tiempo antes de converger a una solución satisfactoria para el usuario. La regla de aprendizaje híbrida para entrenar a un modelo ANFIS, surge de observar que la salida de la red adaptativa (asumiendo que hay sólo una sola salida) es lineal en función de algunos de los parámetros; siendo posible la identificación de estos parámetros lineales utilizando algún método de mínimos cuadrados lineales. Este enfoque conduce a una regla de aprendizaje híbrida presentada en [**1**,**2**], que combina el método del gradiente de mayor descenso (GD) y el estimador de mínimos cuadrados (LSE) con el objetivo de acelerar la identificación de los parámetros.

Por simplicidad, suponer que la red adaptativa bajo consideración tiene una única salida que puede ser representada por

$$y = F(x, S) \qquad \textbf{(4.1)}$$

donde *x* es el vector de las variables de entrada, *S* es el conjunto de parámetros a identificar y *F* es la función general implementada por la red adaptativa. Si existe una función *H* tal que la función compuesta por $H \circ F$ es lineal para algunos de los elementos de *S*, entonces estos elementos lineales pueden ser identificados por el método de mínimos cuadrados. Formalmente, si el conjunto de parámetros *S* se puede dividir en dos conjuntos

$$S = S_1 + S_2 \qquad \textbf{(4.2)}$$

donde +, representa la suma aritmética de tal manera que $H \circ F$ es lineal en los elementos de S_2, entonces al aplicar la ecuación (4.1) a H se tiene

$$H(\circ) = \mathrm{H} \circ \mathrm{F}(x, \mathrm{S}) = \mathrm{H}(\mathrm{F}(x, S), S) = \mathrm{H}(\mathrm{F}(x, S_1), S_2) \qquad \textbf{(4.3)}$$

que es lineal para los elementos de S_2. Por lo que el aprendizaje de los parámetros S_2 se realiza mediante algún método de mínimos cuadrados, como puede ser el que está en función de la matriz pseudoinverza o el método LSE recursivo.

Estimador de mínimos cuadrados

Si se tienen valores preestablecidos en los elementos de S_1, y es presentando el conjunto de datos de entrenamiento *P* con *p* pares de datos en la ecuación (4.3), se puede obtener una ecuación matricial como la siguiente:

$$\boldsymbol{y = A\theta} \qquad \textbf{(4.4)}$$

donde $\boldsymbol{\theta}$ es un vector desconocido cuyos elementos son los parámetros pertenecientes al conjunto S_2. Por lo que el problema se reduce al problema típico de mínimos cuadrados, donde **la mejor solución** para $\boldsymbol{\theta}$**,** es el estimador de mínimos cuadrados $\boldsymbol{\theta^*}$**,** que minimiza el error $\|\boldsymbol{y - A\theta}\|^2$ mediante la siguiente ecuación

$$\boldsymbol{\theta^* = (A^T A)^{-1} A^T y} \quad \textbf{(4.5)}$$

donde $\boldsymbol{A^T}$ es la transpuesta de $\boldsymbol{A}$ y $\boldsymbol{(A^T A)^{-1} A^T}$ es conocida como la matriz pseudoinversa de $\boldsymbol{A}$ si $\boldsymbol{A^T A}$ es no singular.

Es posible también emplear la fórmula recursiva del estimador de mínimos cuadrados para calcular $\boldsymbol{\theta}^*$. Especificamente, si el renglón *i*-ésimo de la matriz $\boldsymbol{A}$ definida en la ecuación (4.4) es determinado como $\boldsymbol{a_i^T}$**,** y el elemento *i*-esimo del vector de salida $\boldsymbol{y}$ como $y_i{}^T$, entonces $\boldsymbol{\theta}$ puede ser calculada iterativamente mediante la siguiente formula:

$$\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i} + \boldsymbol{P_{i+1}}\boldsymbol{a_{i+1}}(y_{i+1} - \boldsymbol{a_{i+1}^T}\boldsymbol{\theta_i})$$

$$\boldsymbol{P_{k+1}} = \boldsymbol{P_k} - \frac{\boldsymbol{P_i}\boldsymbol{a_{i+1}}\boldsymbol{a_{i+1}^T}\boldsymbol{P_i}}{\mathbf{1} + \boldsymbol{a_{i+1}^T}\boldsymbol{P_i}\boldsymbol{a_{i+1}}} \quad \textbf{(4.6)}$$

donde el estimador de mínimos cuadrados $\boldsymbol{\theta}^*$ se alcanza en la iteración *p*, es decir $\boldsymbol{\theta}^* = \boldsymbol{\theta_P}$. Las condiciones iniciales necesarias para arrancar la ecuación (4.6) son $\boldsymbol{\theta_o} = \mathbf{0}$ y $\boldsymbol{P_0} = \gamma \boldsymbol{I}$, estableciendo a γ con un número positivo grande e $\boldsymbol{I}$ es la matriz identidad de dimensión *MxM*. Los efectos de estas condiciones iniciales sobre la identificación del $\boldsymbol{\theta}^*$ pueden estudiarse en [**3**,**4**,**5**,**6**]. El estimador de mínimos cuadrados recursivo, tiene la ventaja de no requerir el cálculo de la matriz inversa, además de ser una metodología muy utilizada en problemas en línea reales, ya que se puede realizar el ajuste de los parámetros, conforme los datos son adquiridos.

Método del gradiente de mayor descenso

De manera general, la regla de actualización de parámetros de una red neuronal mediante un método del gradiente está dada por:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha \nabla G(\boldsymbol{x}^k)$$

donde $\boldsymbol{x}^{k+1}$ indica el vector de parámetros nuevo, $\boldsymbol{x}^k$ indica el vector de parámetros anterior, α determina el tamaño de cambio o peso de actualización, y $\nabla G(.)$, indica el vector gradiente de la función G en función de los parámetros a actualizar. La función G, normalmente se define mediante una medida del error, que puede ser el error cuadrático medio, entonces se tiene que

$$G = \frac{1}{2}e^2$$

dado el error

$$e = T - F$$

La regla de entrenamiento de un sistema Sugeno de primer orden utilizando la regla de aprendizaje de retropropagación para actualizar todos sus parámetros será presentada mediante un ejemplo de un sistema de 4 reglas.

Ejemplo de obtención de la regla de aprendizaje de retropropagación para un sistema difuso Sugeno de primer orden con 4 reglas

Sea *T* es el valor deseado para un par de datos entrada-salida y *F* es la salida del sistema difuso Sugeno de primer orden bajo las mismas condiciones de entrada. Entonces *F* puede expresarse como:

$$F = \frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i}$$

Siendo el sistema difuso Sugeno de primer orden caracterizado por las siguientes condiciones; todos los conjuntos difusos están definidos por funciones de membresía gaussiana, entonces los conjuntos difusos se expresan como sigue

$$A_1 = e^{-\frac{1}{2}(\frac{x_1 - c_{A_1}}{\sigma_{A_1}})}$$
$$A_2 = e^{-\frac{1}{2}(\frac{x_1 - c_{A_2}}{\sigma_{A_2}})}$$
$$B_1 = e^{-\frac{1}{2}(\frac{x_2 - c_{B_1}}{\sigma_{B_1}})}$$
$$B_2 = e^{-\frac{1}{2}(\frac{x_2 - c_{B_2}}{\sigma_{B_2}})}$$

Las t-normas que definen las operaciones de conjunción en las reglas difusas son definidas por el producto algebraico, por lo tanto las t-normas pueden expresarse como sigue

$$w_i = A_j B_k$$
$$w_1 = A_1 B_1$$
$$w_2 = A_1 B_2$$
$$w_3 = A_2 B_1$$
$$w_4 = A_2 B_2$$

Y sean las funciones de salida f_i estan determinadas por funciones lineales parametrizadas

$$f_i = p_i x_1 + q_i x_2 + r_i$$
$$f_1 = p_1 x_1 + q_1 x_2 + r_1$$
$$f_2 = p_2 x_1 + q_2 x_2 + r_2$$
$$f_3 = p_3 x_1 + q_3 x_2 + r_3$$
$$f_4 = p_4 x_1 + q_4 x_2 + r_4$$

Entonces el conjunto de parámetros ajustables para este sistema contiene los siguientes 20 parámetros:

$$c_{A_1}, c_{A_2}, \sigma_{A_1}, \sigma_{A_2}, c_{B_1}, c_{B_2}, \sigma_{B_1}, \sigma_{B_2}, p_1, q_1, r_1, p_2, q_2, r_2, p_3, q_3, r_3, p_4, q_4, r_4$$

Por lo que el vector gradiente, es un vector de 20x1 elementos determinado por la derivada de cada parámetro. A continuación, se describe la obtención de cada componente del gradiente. El componente del gradiente, correspondiente al parámetro que define al centro del conjunto difuso *A*, está dado por:

$$\frac{\nabla E}{\partial c_{A_1}} = \frac{\partial}{\partial c_{A_1}}\left(\frac{1}{2}e^2\right) = e\frac{\partial}{\partial c_{A_1}}(T - F) = -e\frac{\partial}{\partial c_{A_1}}(F)$$

$$= -e\frac{\partial}{\partial c_{A_1}}\left(\frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i}\right) = -e\left(\frac{\sum_{i=1}^{n} w_i \frac{\partial}{\partial c_{A_1}}(\sum_{i=1}^{n} w_i f_i) - \sum_{i=1}^{n} w_i f_i \frac{\partial}{\partial c_{A_1}}(\sum_{i=1}^{n} w_i)}{(\sum_{i=1}^{n} w_i)^2}\right)$$

Se puede observar que las funciones f_i y w_3, w_4 no dependen del parámetro c_{A_1} y especificando las siguientes constantes:

$$\mathbb{C}_1 = \sum_{i=1}^{n} w_i \qquad \mathbb{C}_2 = \sum_{i=1}^{n} w_i f_i$$

Se tiene

$$= \frac{\partial}{\partial c_{A_1}}\left(\frac{1}{2}e^2\right) = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1 \frac{\partial}{\partial c_{A_1}}\left(w_1 f_1 + w_2 f_2\right) - \mathbb{C}_2 \frac{\partial}{\partial c_{A_1}}(w_1 + w_2)\right]$$

$$= \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1 B_1 \frac{\partial}{\partial c_{A_1}} A_1 + f_2 B_2 \frac{\partial}{\partial c_{A_1}} A_1\right) - \mathbb{C}_2\left(B_1 \frac{\partial}{\partial c_{A_1}} A_1 + B_2 \frac{\partial}{\partial c_{A_1}} A_1\right)\right]$$

$$= \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1 B_1 + f_2 B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]\frac{\partial A_1}{\partial c_{A_1}}$$

$$= \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1 B_1 + f_2 B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]\frac{\partial A_1}{\partial c_{A_1}}$$

Como

$$\frac{\partial A_1}{\partial c_{A_1}} = A_1\left(\frac{x_1 - c_{A_1}}{\sigma_{A_1}{}^2}\right)$$

Entonces

$$\frac{\nabla E}{\partial c_{A_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1 B_1 + f_2 B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_1\left(\frac{x_1 - c_{A_1}}{\sigma_{A_1}{}^2}\right)$$

Para el parámetro σ_{A_1}, se repite el proceso y únicamente se cambia la derivada;

$$\frac{\partial A_1}{\partial \sigma_{A_1}} = A_1\frac{\left(x_1 - c_{A_1}\right)^2}{\sigma_{A_1}{}^3}$$

$$\frac{\nabla E}{\sigma_{A_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1 B_1 + f_2 B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_1\frac{\left(x_1 - c_{A_1}\right)^2}{\sigma_{A_1}{}^3}$$

Para el parámetro que define el centro del conjunto c_{A_2}

$$\frac{\partial}{\partial c_{A_2}}\left(\frac{1}{2}e^2\right) = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1 \frac{\partial}{\partial c_{A_2}}\left(w_3 f_3 + w_4 f_4\right) - \mathbb{C}_2 \frac{\partial}{\partial c_{A_2}}(w_3 + w_4)\right]$$

Siguiendo los procedimientos anteriores, la obtención de los componentes del vector gradientes correspondiente a los parámetros antecedentes son las siguientes

$$\frac{\nabla E}{\partial c_{A_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3B_1 + f_4B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_2\left(\frac{x_1 - c_{A_2}}{\sigma_{A_2}{}^2}\right)$$

$$\frac{\nabla E}{\sigma_{A_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3B_1 + f_4B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_2\frac{\left(x_1 - c_{A_2}\right)^2}{\sigma_{A_2}{}^3}$$

Para los parámetros de los conjuntos difusos de la variable x_2 los vectores gradientes quedan determinados de la siguiente manera

$$\frac{\nabla E}{\partial c_{B_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1A_1 + f_2A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_1\left(\frac{x_2 - c_{B_1}}{\sigma_{B_1}{}^2}\right)$$

$$\frac{\nabla E}{\sigma_{B_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1A_1 + f_2A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_1\frac{\left(x_2 - c_{B_2}\right)^2}{\sigma_{B_1}{}^3}$$

$$\frac{\nabla E}{\partial c_{B_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3A_1 + f_4A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_2\left(\frac{x_2 - c_{B_2}}{\sigma_{B_2}{}^2}\right)$$

$$\frac{\nabla E}{\sigma_{B_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3A_1 + f_4A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_2\frac{\left(x_1 - c_{A_2}\right)^2}{\sigma_{B_2}{}^3}$$

En resumen se tienen las siguientes 8 ecuaciones mostradas en la tabla 4.1 para actualizar los parámetros antecedentes, ubicados en la capa 1 de la arquitectura ANFIS que implementa al sistema de inferencia difusa Sugeno.

Tabla 4.1 Componente del gradiente para una red ANFIS-Sugeno de primer orden, de la capa 1

c_{A_1}	$\frac{\nabla E}{\partial c_{A_1}} = \frac{-e}{\mathbb{C}_1{}^2}[\mathbb{C}_1(f_1B_1 + f_2B_2) - \mathbb{C}_2(B_1 + B_2)]A_1\left(\frac{x_1 - c_{A_1}}{\sigma_{A_1}{}^2}\right)$
σ_{A_1}	$\frac{\nabla E}{\sigma_{A_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1B_1 + f_2B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_1\frac{\left(x_1 - c_{A_1}\right)^2}{\sigma_{A_1}{}^3}$
c_{A_2}	$\frac{\nabla E}{\partial c_{A_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3B_1 + f_4B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_2\left(\frac{x_1 - c_{A_2}}{\sigma_{A_2}{}^2}\right)$
σ_{A_2}	$\frac{\nabla E}{\sigma_{A_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3B_1 + f_4B_2\right) - \mathbb{C}_2(B_1 + B_2)\right]A_2\frac{\left(x_1 - c_{A_2}\right)^2}{\sigma_{A_2}{}^3}$
c_{B_1}	$\frac{\nabla E}{\partial c_{B_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1A_1 + f_2A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_1\left(\frac{x_2 - c_{B_1}}{\sigma_{B_1}{}^2}\right)$
σ_{B_1}	$\frac{\nabla E}{\sigma_{B_1}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_1A_1 + f_2A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_1\frac{\left(x_2 - c_{B_2}\right)^2}{\sigma_{B_1}{}^3}$
c_{B_2}	$\frac{\nabla E}{\partial c_{B_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3A_1 + f_4A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_2\left(\frac{x_2 - c_{B_2}}{\sigma_{B_2}{}^2}\right)$
σ_{B_2}	$\frac{\nabla E}{\sigma_{B_2}} = \frac{-e}{\mathbb{C}_1{}^2}\left[\mathbb{C}_1\left(f_3A_1 + f_4A_2\right) - \mathbb{C}_2(A_1 + A_2)\right]B_2\frac{\left(x_1 - c_{A_2}\right)^2}{\sigma_{B_2}{}^3}$

Los componentes del gradiente correspondiente a los parámetros consecuentes se encuentran como sigue:

Sea

$$\frac{\nabla E}{\partial p_1} = \frac{\partial}{\partial p_1}\left(\frac{1}{2}e^2\right) = -e\frac{\partial}{\partial p_1}(F)$$

y

$$\frac{\nabla E}{\partial p_1} = -e\frac{\partial}{\partial p_1}\left(\frac{\sum_{i=1}^{n} w_i f_i}{\sum_{i=1}^{n} w_i}\right)$$

Como sólo la función f_1 depende del parámetro p_1 entonces la ecuación anterior utilizando la constante $\mathbb{C}_1$ se reduce a:

$$\frac{\nabla E}{\partial p_1} = -e\left(\frac{\frac{\partial f_1}{\partial p_1}}{\mathbb{C}_1}\right) = -\frac{e}{\mathbb{C}_1}(x_1)$$

Los parámetros q_1 y r_1 quedan dados por:

$$\frac{\nabla E}{\partial q_1} = -e\left(\frac{\frac{\partial f_1}{\partial q_1}}{\mathbb{C}_1}\right) = -\frac{e}{\mathbb{C}_1}(x_2)$$

$$\frac{\nabla E}{\partial r_1} = -e\left(\frac{\frac{\partial f_1}{\partial r_1}}{\mathbb{C}_1}\right) = -\frac{e}{\mathbb{C}_1}$$

Los otros parámetros consecuentes se obtienen utilizando el mismo razonamiento, por lo que en resumen la tabla 4.2 presenta las ecuaciones de optimización de los parámetros consecuentes del sistema Sugeno de primer orden.

Tabla 4.2 Componentes del gradiente para una red ANFIS-Sugeno de primer orden, de la capa 4.

$\frac{\nabla E}{\partial p_1}, \frac{\nabla E}{\partial p_2}, \frac{\nabla E}{\partial p_3}, \frac{\nabla E}{\partial p_4}$	$\frac{\nabla E}{\partial q_1}, \frac{\nabla E}{\partial q_2}, \frac{\nabla E}{\partial q_3}, \frac{\nabla E}{\partial q_4}$	$\frac{\nabla E}{\partial r_1}, \frac{\nabla E}{\partial r_2}, \frac{\nabla E}{\partial r_3}, \frac{\nabla E}{\partial r_4}$
$-\frac{e}{\mathbb{C}_1}(x_1)$	$-\frac{e}{\mathbb{C}_1}(x_2)$	$-\frac{e}{\mathbb{C}_1}$

Regla híbrida

La regla híbrida combina el método del gradiente de mayor descenso y el estimador de mínimos cuadrados para actualizar los parámetros en una red adaptativa. Para que el algoritmo de aprendizaje híbrido pueda ser aplicado en modo de procesamiento por lotes, cada época de

entrenamiento se compone de dos pasos, en el primer paso se realiza el cálculo de los nodos de salida mediante una propagación hacia adelante y se identifican los parámetros consecuentes y posteriormente se realiza una propagación de las señales de error hacia atrás para la identificación de los parámetros antecedentes.

En el paso hacia adelante, se presenta cada componente de datos del vector de entrada, se calculan las salidas de cada nodo de la red capa por capa hasta obtener el reglón correspondiente de las matrices $\boldsymbol{A}$ e $\boldsymbol{y}$ de la ecuación (4.4), repitiendo este proceso para todos los datos del vector de entrada del conjunto de entrenamiento hasta formar por completo la matriz $\boldsymbol{A}$ y con el correspondiente vector $\boldsymbol{y}$; se identifican el conjunto de parámetros lineales S_2 ya sea mediante la ecuación de la matriz pseudoinversa (4.5) o las ecuaciones recursivas del estimador de mínimos cuadrados (4.6). Después de identificar los parámetros de S_2, se inicia el paso dos de la época de entrenamiento, considerando a los parámetros S_2 fijos y calculando la señal de error (medida de error) para cada par de datos de entrenamiento, que se utilizará para la actualización del conjunto S_1.

En el paso hacia atrás, las señales de error (señales obtenidas de la derivada de la medida de error con respecto a cada nodo de salida, propagándose desde el extremo de la salida hacia el extremo de entrada de la red; actualizando los valores del vector gradiente en cada entrada de datos de entrenamiento. Finalmente, el último paso hacia atrás incluye la presentación de todos los datos de entrenamiento, y los parámetros del conjunto del conjunto S_1 que han sido actualizados mediante el método del gradiente de mayor descenso.

Cuando están fijados los valores de los parámetros de S_1, los valores de los parámetros en S_2 encontrados mediante el algoritmo de estimador de mínimos cuadrados garantizan los valores óptimos (global) en el espacio de parámetros de S_2. La regla de aprendizaje híbrida no sólo disminuye la dimensión del espacio de búsqueda original (que incluye a los parámetros de S_1 y S_2) a S_1 cuando se aplica el método de retropopagación, sino que en general, también reduce sustancialmente el tiempo necesario para alcanzar una convergencia deseada.

Regla híbrida utilizando aprendizaje en línea (aprendizaje patrón-por-patrón)

Si los parámetros de la red adaptativa se actualizan en la presentación de cada par de datos del conjunto de entrenamiento, tenemos el esquema en-línea, también conocido como aprendizaje patrón por patrón. Esta estrategia de aprendizaje es necesaria en la identificación de parámetros en línea para sistemas con características cambiantes. Para modificar la regla de aprendizaje por lotes y obtener una versión en línea, el método del gradiente de mayor descenso debe actualizarse cada que se presenta un par de entrada-salida. En el sentido estricto, esto no es un procedimiento de búsqueda basado en el gradiente para minimizar el error *E*, sin embargo, este procedimiento se aproxima al método del gradiente, si la tasa de aprendizaje es pequeña según [**7**,**8**,**9**].

Para que ecuación de mínimos cuadrados recurrente tome en cuenta las características cambiantes en el tiempo que tienen los datos de entrada-salida, los efectos de los pares de datos antiguos deben decaer conforme los nuevos datos pares estén disponibles.

Un método simple para solucionar este problema es formulando la medida de error cuadrática como una versión ponderada que da factores de ponderación más altos a los pares de datos más recientes. Esto equivale a la agregación de un factor de olvido a la fórmula de mínimos cuadrados recursiva original

$$\boldsymbol{\theta_{i+1}} = \boldsymbol{\theta_i} + \boldsymbol{P_{i+1}a_{i+1}}\left(y_{i+1} - \boldsymbol{a_{i+1}^T\theta_i}\right)$$

$$\boldsymbol{P_{i+1}} = \frac{\mathbf{1}}{\boldsymbol{\lambda}}\left(\boldsymbol{P_i} - \frac{\boldsymbol{P_i a_{i+1} a_{i+1}^T P_i}}{\boldsymbol{\lambda} + \boldsymbol{a_{i+1}^T P_i a_{i+1}}}\right)$$

Donde en la práctica el valor típico de $\boldsymbol{\lambda}$ toma valores entre 0.9 y 1. Cuanto más pequeño sea $\boldsymbol{\lambda}$, más rápido decae los efectos de los datos viejos, sin embargo, debe tomarse en cuenta que un valor muy pequeño de $\boldsymbol{\lambda}$ a veces causa inestabilidad numérica, por lo que debe evitarse.

Diferentes formas de combinar la regla de aprendizaje híbrida

Generalmente, la complejidad computacional del estimador de mínimos cuadrados (LSE) es más alta que la del método del gradiente de mayor descenso (GD) en una época de entrenamiento. Sin embargo, el LSE generalmente converge a la solución deseada mucho más rápido que el método del SD. En consecuencia, dependiendo de la disposición recursos informáticos y el nivel requerido de rendimiento (aproximación), se puede elegir de entre al menos cuatro tipos de reglas de aprendizaje híbridas que combinan los método GD y LSE a diferentes grados, como sigue:

1. Aplicar únicamente el método del LSE: parámetros no lineales se fijan con un valor preestablecido, mientras que los parámetros lineales se identifican mediante la aplicación LSE una única vez.
2. Aplicar únicamente el método de mayor descenso GD: Todos los parámetros se tratan como no lineales y son actualizados mediante un proceso iterativo utilizando el método de SD.
3. Una sola iteración del LSE seguido del método de GD: Se emplea LSE una sola vez al inicio para obtener los valores iniciales de los parámetros lineales, y a continuación se realiza el proceso iterativo utilizando el método de GD para identificar todos los parámetros.
4. Aplicar los métodos GD y LSE iterativamente: Como paso preliminar se deben identificar los parámetros lineales y los no lineales. En cada iteración (o época) se utiliza el método de GD para ajustar los parámetros no lineales seguidos del método del LSE para identificar los parámetros lineales.

La elección de uno de los métodos antes mencionados se basa en un equilibrio entre la complejidad computacional y la capacidad de aproximación que se desea. En el capítulo 2 se mostraron dos ejemplos de aproximación de funciones utilizando ANFIS.

4.3. Regla de aprendizaje para el entrenamiento de la red ANFIS-Sugeno de primer orden con operadores de intersección paramétricos

En este trabajo se plantea el algoritmo de evolución diferencial para entrenar a los parámetros de los operadores de intersección de la red ANFIS-Sugeno de primer, ya que estos operadores son caracterizados mediante funciones discontinuas, por lo que evidentemente un algoritmo no derivativo es más adecuado en este tipo de problemas. Es posible emplear cualquier tipo de algoritmos no determinísticos que existen en la literatura, tales como, los algoritmos evolutivos, los algoritmos bioinspirados, entre otros [**10**,**11**]. Sin embargo, se ha optado por el algoritmo de evolución diferencial, ya que este algoritmo ha tenido mucha popularidad, por su simpleza matemática, transparencia en el proceso iterativo y mayor velocidad de convergencia a la solución óptima como ha sido demostrado en diversos trabajos [**12**,**13**,**14**,**15**,**16**].

La regla de aprendizaje de la red ANFIS-Sugeno de primer orden con operadores paramétricos de conjunción utilizando el algoritmo de evolución diferencial puede implementarse mediante cuatro pasos elementales.

1. Determinar todos los parámetros de la red ANFIS-Sugeno de primer orden; número, tipo de conjuntos, el valor de los parámetros antecedentes, la familia y el valor de p de los operadores de intersección, el número de reglas,
2. Encontrar los valores óptimos de los parámetros lineales consecuentes, utilizando un método del estimador de mínimos cuadrados
3. Aplicar el algoritmo de evolución diferencial para entrenar los parámetros de los operadores de intersección, utilizando como función objetivo el estimador de mínimos cuadrados
4. Ir al paso 2, hasta que la condición de paro sea satisfecha

4.4. Algoritmo de evolución diferencial para el entrenamiento de ANFIS

Finalmente, debe ser implementada la estrategia del algoritmo de evolución diferencial simple que conlleve a una solución satisfactoria. La implementación del AED difiere de una aplicación específica de otra en la manera de definir la representación de la población y la función objetivo. A continuación, se presenta el algoritmo de evolución diferencial, para una red ANFIS-Sugeno de primer orden que utiliza operadores paramétricos de la metodología 4T con los mismos valores de parámetros para todas las reglas.

4.5. Algoritmo de evolución diferencial para el entrenamiento de la red ANFIS con mismo operador y mismo valor de *p* para todas las reglas

El algoritmo de evolución diferencial implementado en un modelo ANFIS-Sugeno de primer orden que contienen operadores de intersección paramétricos iguales en todas las reglas, es decir, todos los operadores paramétricos pertenecen a la misma familia de la metodología 4T y tienen el mismo valor de *p* se muestra en el algoritmo de la figura 4.1.

```
1  Definir los parámetros del AED; tamaño de la población NP, el número de Generaciones g,
   la probabilidad de Cruza-Mutación Cr y el valor del factor de escalamiento: F
2  Crear una Población inicial P de de parámetros p
3  Evaluar la función de aptitud (Error Cuadrático medio ) de cada individuo de la población
   y determinar el MINIMO de aptitud
4  para G = 1 hasta MaxGeneración hacer
5     para i = 1 hasta NP hacer
6           ## Seleccionar aleatoriamente tres índices de la población actual
7           r1, r2, r3 ∈ NP  r1 ≠ r2 ≠ r3 ≠ i
8           Generar solución de prueba v
9           si (rand[0,1] < CR) entonces
10             v = x(r1) + F * (x(r2) − x(r3))
11          fin
12          si (V > 1 o V < 0) entonces
13             v = x(r1) − F * (x(r2) − x(r3))
14          fin
15          si (V > 1 o V < 0) entonces
16             v = rand()
17          fin
18          ##Aplicar el estimador de mínimos cuadrados para el parámetro p
19          f = LSE(v)
20          aux=Evaluar la función de aptitud def (v)
21          si (aux < Aptitud P(i)) entonces
22             P(i) = v
23             Aptitud(i) = aux
24          fin
25          si (aux < MINIMO) entonces
26             MINIMO = aux
27             actualizar el parametro p
28          fin
29    fin
30 fin
```

Figura 4.1 Algoritmo de evolución diferencial en ANFIS

Para probar el poder de aproximación de la arquitectura ANFIS-Sugeno de primer orden que incluye operadores de intersección paramétricos, se utiliza: a) la función *sinc* de dos entradas y; b) una función llamada *multisinc*, que es la suma ponderada de funciones *sinc* desfasadas. Dos pruebas principales diferentes se realizan: a) la aproximación de la función, utilizando conjuntos difusos igualmente espaciados y; b) la aproximación después que la arquitectura ANFIS-Sugeno de primer orden con t-norma producto ha sido entrenada con la regla híbrida LSE-GD, con el objetivo de mejorar la aproximación de la red ANFIS.

4.5.1. Mejoramiento Relativo.

El mejoramiento relativo es una medida para determinar el porcentaje de mejora en la aproximación de una función, dada una aproximación inicial. El mejoramiento relativo puede expresarse como sigue:

$$MejoramientoRelativo = \frac{Error\ Inicial - Error\ Final}{Error\ Inical} * 100\%$$

Note que si el error final es mayor al error inicial, entonces se generará un resultado negativo que indica que no existe mejoramiento. En los siguientes ejemplos la medida del mejoramiento relativo estará determinada por la siguiente expresión.

$$\underset{Relativo}{Mejoramiento} = \frac{Error\ ANFIS\ tradicional - Error\ ANFIS\ con\ operaores\ parametricos}{Error\ ANFIS\ tradicional} * 100\%$$

4.5.2. Ejemplo 1: Arquitectura ANFIS-Sugeno de primer orden que incluye operadores de intersección paramétricos en la aproximación de la función *sinc* de dos entradas

La función a aproximar $sinc(x, y)$ de dos dimensiones se define como:

$$z = sinc(x, y) = \frac{\sin(x)\sin(y)}{xy}$$

El conjunto de entrenamiento se formó utilizando 121 pares de datos de entrenamiento de entrada-salida, igualmente distribuidos en el rango de entrada $[-10, 10]\times[-10,10]$, es decir, se tomaron los valores de *x*, *y* = {-10, -8 -6 -4 -2 0 2 4 6 8 10} y se realiza una combinación completa de cada entrada. La gráfica de superficie del conjunto de entrenamiento de la función *sinc* se muestra en la figura 4.2.

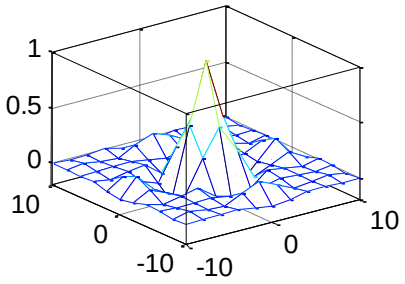


Figura 4.2 Función objetivo *sinc(x,y)*

Para la aproximación del modelo ANFIS-Sugeno se utilizan un conjunto de reglas difusas que realizan una partición completa del universo de discurso, es decir, se utilizaron 9, 16 y 25 reglas difusas del tipo si-entonces para cuando se tienen 3, 4 y 5 conjuntos difusos por variable respectivamente. Los operadores paramétricos se dividen en 6 familias, que se presentaron en el capítulo anterior, sin embargo, en este trabajo sólo se presentan los resultados de la arquitectura ANFIS-Sugeno de primer orden con operadores paramétricos utilizando la metodología 4T. Todas las pruebas que se presentan, el valor de los parámetros *CR* y *F* del algoritmo de evolución diferencial fueron configurados en 0.7, considerando las sugerencias del análisis realizado en [**14**].

La tabla 4.3 y 4.4 muestran las configuraciones del modelo ANFIS especificando el número y tipo de conjuntos difusos, suponiendo que estos son igualmente espaciados sobre las variables de entrada, el error inicial (identificando los parámetros utilizando LSE) y el error final después de 50 generaciones de entrenamiento para una población *NP*=10. Por otro lado, también es considerado el caso donde se desea mejorar el entrenamiento de ANFIS tradicional (parte derecha de la tabla),

después de que la red ha sido entrenada con 100 épocas de entrenamiento, se entrenan los operadores de intersección utilizando los mismo parámetros del algoritmo de evolución diferencial.

La tabla 4.3 considera un operador paramétrico de la familia DLLM, que no tiene implementado el operador producto, y se observa que sólo en un caso, tres conjuntos difusos triangulares por variable, el entrenamiento de la red ANFIS es mejorado. Mientras que en la tabla 4.4 es considerado el operador de la famila PPPM, y es visible que el entrenamiento de ANFIS es mejorado en tres de 6 casos (obviamente el peor caso es que no mejore la aproximación). Sin embargo, el mejor mejoramiento, casos 4 conjuntos difuso tipo bell, sólo es del 2.5 por ciento.

Tabla 4.3 Resultados de optimización de regla de un sólo operador DLLM con mismo valor *p*

Número y tipo de conjuntos	Conjuntos Igualmente espaciados				Conjuntos preentrenados				
	Error Inicial	Error Final	P inicial	P final	Error ANFIS	Error Inicial	Error Final	P inicial	P final
Bell(3)	1.69e-2	1.49e-2	0.5	0.99	1.34e-3	1.79e-2	5.94e-3	0.5	0.17
Bell(4)	1.08e-2	1.05e-2	0.5	0.05	3.20e-4	3.16e-3	2.64e-3	0.5	0.13
Bell(5)	7.13e-3	7.59e-4	0.5	0.01	3.45e-5	1.53e-3	9.76e-5	0.5	0.02
Triangular(3)	1.63e-2	1.28e-2	0.5	0.85	7.15e-3	1.59e-2	**6.23e-3**	0.5	0.85
Triangular(4)	1.16e-2	1.08e-2	0.5	0.09	1.40e-3	5.37e-3	2.27e-3	0.5	0.90
Triangular(5)	8.05e-3	2.18e-3	0.5	0.01	3.46e-5	1.68e-3	2.04e-4	0.5	0.04

Tabla 4.4 Resultados de optimización de regla de un sólo operador PPPM con mismo valor *p*

Número y tipo de conjuntos	Conjuntos Igualmente espaciados					Conjuntos preentrenados				
	Error Inicial	Error Final	P inicial	P final	Error ANFIS	Error Inicial	Error Final	Mejoramiento relativo(%)	P inicial	P final
Bell(3)	1.69e-2	1.49e-2	0.5	0.99	1.34e-3	1.24e-2	**1.34e-3**	**0.00**	0	0.95
Bell(4)	1.08e-2	1.05e-2	0.5	0.05	3.20e-4	3.29e-3	**3.12e-4**	**2.50**	0	0.96
Bell(5)	7.13e-3	7.59e-4	0.5	0.01	3.45e-5	1.39e-4	**3.45e-5**	**0.00**	0	0.86
Triangular(3)	1.63e-2	1.28e-2	0.5	0.85	7.15e-3	9.58e-3	**7.14e-3**	**0.14**	0	0.97
Triangular(4)	1.16e-2	1.08e-2	0.5	0.09	1.40e-3	3.64e-3	**1.39e-3**	**0.71**	0	0.94
Triangular(5)	8.05e-3	2.18e-3	0.5	0.01	3.46e-5	2.07e-4	**3.46e-5**	**0.00**	0	0.71

Para mejorar los resultados de la aproximación de la arquitectura ANFIS-Sugeno con operadores de conjunción paramétrica, se realizó una mejora que permita que cada operador de la red ANFIS pueda tener un valor de *p* diferente, teniendo así un mayor grado de libertad, encontrando el valor de *p* de cada regla *i*, al fijar los valores de *p* de las demás reglas ejecutando el algoritmo de la figura 4.1. El proceso se repite 6 veces, que es el valor encontrado a prueba y error.

La tabla 4.5 muestra los resultados del entrenamiento del operador PPPM, que pertenece a la metodología 4T, para una red ANFIS que puede tener diferentes valores de *p*, examinado para un número de población *NP*=16 y 40 generaciones. Como puede observarse de la tabla, el entrenamiento de la red ANFIS, mejora considerablemente, teniendo un mejoramiento en 5 de 6 casos, presentado mejoras en la aproximación del 14 a 37%.

Finalmente, se hace un entrenamiento de la red ANFIS con operadores de intersección, en la cual se permita que los operadores de intersección pertenecientes a la base de reglas, puedan

pertenecer a diferentes familias de operadores de la metodología 4T, con el valor de *p* arbitrario, con el objetivo de tener una mejor optimización.

Tabla 4.5 Resultados de optimización de regla con misma familia PPPM, diferentes valores de *p*

Número y tipo de conjuntos	Conjuntos Igualmente espaciados		Conjuntos preentrenados		
	Error Inicial	Error Final	Error ANFIS	Error Final	Mejoramiento relativo(%)
Bell(3)	1.44e-002	1.41e-002	1.34e-3	1.34e-3	0
Bell(4)	1.10e-002	9.66e-003	3.20e-4	2.70E-04	15.6
Bell(5)	6.79e-004	3.78e-004	3.45e-5	2.93E-05	15.1
Triangular(3)	8.16e-003	6.74e-003	7.15e-3	6.14e-003	14.1
Triangular(4)	9.03e-003	4.17e-003	1.40e-3	8.79E-04	37.2
Triangular(5)	6.67e-004	3.77e-004	3.46e-5	2.82e-005	18.5

La tabla 4.6, muestra los resultados de entrenar la red ANFIS, utilizando los mismos parámetros de entrenamiento configurados en la Tabla 4.5. De estos resultados se puede observar que la capacidad de aproximación de la red ANFIS puede mejorar drásticamente, teniendo mejoras de hasta el 80 % y un error de orden mili. Las gráficas que muestran los resultados de las tablas 4.3 a la 4.6 se anexan en la carpeta /FIGURAS /4T.

Tabla 4.6 Diferente familia, diferentes valores de *p* por regla

Número y tipo de conjuntos	Conjuntos Igualmente espaciados		Conjuntos preentrenados		
	Error Inicial	Error Final	Error ANFIS	Error Final	Mejoramiento relativo(%)
Bell(3)	1.44e-002	1.16e-002	1.34e-3	1.31e-3	**2.2**
Bell(4)	1.10e-002	3.02e-003	3.20e-4	1.92E-04	**40.1**
Bell(5)	6.79e-004	164 e-004	3.45e-5	1.60e-005	**53.6**
Triangular(3)	8.16e-003	6.16e-003	7.15e-3	3.00E-03	**58.0**
Triangular(4)	9.03e-003	1.80e-003	1.40e-3	6.60E-04	**52.9**
Triangular(5)	6.67e-004	3.05e-004	3.46e-5	6.88e-006	**80.1**

Con el objetivo de mostrar la capacidad de aproximación de la red ANFIS utilizando operadores de intersección en otras funciones de prueba, se repetirá el ejercicio anterior para cuatro funciones de mayor complejidad matemática basada en funciones *sinc*.

4.5.3. Ejemplo 2. Aproximación de funciones *multisinc*, utilizando ANFIS con operadores de intersección paramétricos.

Las funciones de prueba *multisinc* utilizadas para comparar la capacidad de aproximación de la red ANFIS-Sugeno con operadores de intersección parameticos, son generadas de la suma de funciones recorridas de la función *sinc* del ejemplo 1. Las funciones matemáticas que describen las funciones a analizar son:

$$sinc_2(x,y) = sinc(x,y) + sinc(x+4, y+2)$$
$$sinc_3(x,y) = sinc_2(x,y) + sinc(x-5, y-4)$$

$$sinc_4(x,y) = sinc_3(x,y) + sinc(x+2, y-6)$$
$$sinc_5(x,y) = sinc_4(x,y) + sinc(x, y+8)$$

Desarrollando explícitamente la función $sinc_2(x,y)$ se tiene que

$$sinc_2(x,y) = \frac{\sin(x)\sin(y)}{xy} + \frac{\sin(x+4)}{x+4}\frac{\sin(y+2)}{y+2}$$

El desarrollo explícito de las demás funciones es similar. Un algoritmo que implementa a las funciones de prueba puede definirse utilizando un arreglo de parámetros de corrimiento.

```
xys=[ 0 0 ;  -4 -2;  5 4; -2 6;  0 -8 ];
```

Para generar la suma de funciones *sinc* se debe ponderar la siguiente iteración

```
ZS=0
para  k = 1 hasta nsinc;
   Z=sinc(X-xys(k,1), Y-xys(k,2))
   ZS=ZS+Z;
fin
```

La figura 4.3, muestra las gráficas de superficie de la suma ponderada de funciones *multisinc*, para *n* igual a 2, 3,4, 5, observe que cuando *n* =1, entonces la función objetivo es la presentada en el ejemplo 1.

La tabla 4.7, muestra los resultados de aproximación de las funciones objetivos de múltiples *sinc*, presentado un mejoramiento relativo de la aproximación al utilizar 5 funciones triangulares y operadores de intersección de la familia PPPM. Como puede observarse de los errores presentados en la tabla, la aproximación de ANFIS tradicional a las funciones *multisinc* es buena en todos los casos y ANFIS con operadores de conjunción paramétrica se mejoró la aproximación prácticamente en todos los casos.

Tabla 4.7 Error de aproximación de las funciones *multisinc* y mejoramiento de aproximación de ANFIS

Número de *sincs*	Misma operador, mismo p; PPPM			Misma operador, diferente p; PPPM		Diferente familia, diferente p	
	Error ANFIS	Error Final	Mejora relativa(%)	Error Final	Mejora relativa(%)	Error Final	Mejora relativa(%)
1	3.46e-5	3.46e-5	0.00	2.82e-5	**18.5**	6.88e-6	**80.1**
2	9.46e-5	9.46e-5	0.00	7.14e-5	**24.5**	3.31e-5	**65.0**
3	9.60e-5	**9.49e-5**	**1.15**	7.48e-5	**22.1**	3.33e-5	**65.3**
4	2.00e-4	**1.99e-4**	**0.50**	8.67e-5	**56.7**	4.59e-5	**77.1**
5	8.42e-4	**6.74e-4**	**19.95**	3.78e-4	**55.1**	2.17e-4	**74.2**

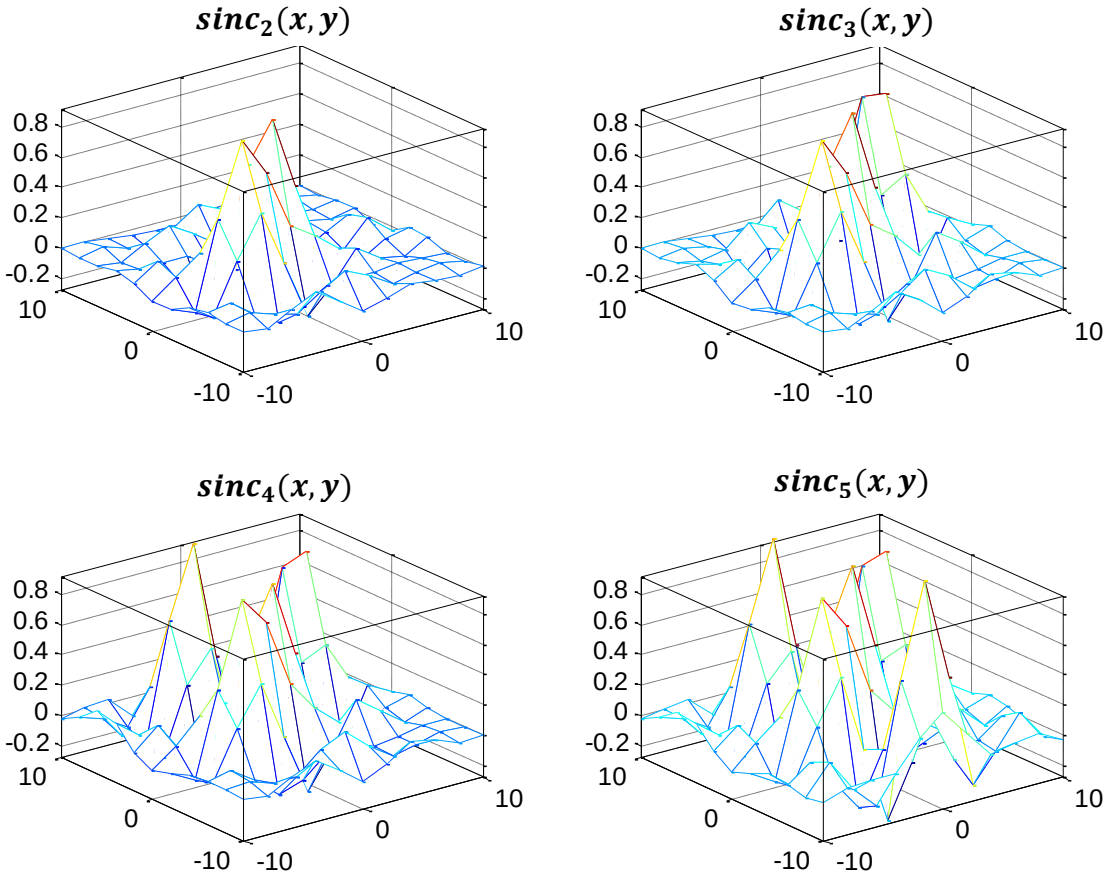


Figura 4.3 Graficas de superficie de las funciones *multisinc*

4.6. Observaciones

En este capítulo se realizó una revisión de la regla de aprendizaje híbrida que propone Jang para un sistema difuso tipo Sugeno de primer orden. Además se presentó la regla de aprendizaje de la red ANFIS-Sugeno de primer orden con operadores de intersección de la familia 4T presentada en el capítulo anterior, que utiliza una combinación del algoritmo del estimador de mínimos cuadrados y el algoritmo de evolución diferencial.

Se hace un análisis de la capacidad de la red ANFIS bajos dos supuestos: a) considerando que los conjuntos antecedentes no pueden ser ajustados, bajo el supuesto que estos ya han sido optimizados utilizando el conocimiento de un experto humano o que los conjuntos difusos fueron optimizados bajos un algoritmo inteligente, y como se mostró, la regla de aprendizaje siempre logró llegar a una mejor aproximación comparada a la inicial, b) se considera a la regla de aprendizaje propuesta como una mejora a la aproximación de la red ANFIS, y como se mostró en los ejemplos, la regla de aprendizaje cuando se consideran diferentes familias de operadores de la metodología 4T, el mejoramiento en la aproximación de la red ANFIS logra mejoras de hasta el 80%.

Con la regla de aprendizaje presentada, es posible extender a las otras familias de operadores de intersección paramétricas presentadas en el capítulo anterior. Dicha tarea, se presenta como un trabajo futuro de esta tesis.

Se presentaron los resultados de la aproximación de ANFIS-Sugeno para aproximar funciones multimodales complejas y se mostraron la mejora al tener operadores de intersección paramétricos.

El algoritmo de evolución diferencial utilizado en el entrenamiento de la red ANFIS fue propuesto por ser de simple implementación, transparencia en el proceso evolutivo y debido a que este algoritmo ha sido reportado en la literatura como unos de los mejores algoritmos de búsqueda en problemas de minimización de funciones, sin embargo, debe tomarse considerarse como trabajo futuro, un análisis comparativo de la regla de aprendizaje propuesta utilizando otras estrategias de búsqueda heurística.

Debe notarse que el costo al mejorar la aproximación es que el tiempo de entrenamiento de la regla de aprendizaje propuesta es mayor que el de la regla híbrida de ANFIS tradicional. Este costo es obvio y esperado al utilizar un método de optimización no diferenciable.

4.7 Referencias

1. Jang, J.-S.: ANFIS: adaptive-network-based fuzzy inference system. Systems, Man and Cybernetics, IEEE Transactions on 23(3), 665-685 (1993)

2. Jang, J.-S., others: Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm. In : AAAI, vol. 91, pp.762-767 (1991)

3. Bates, D., Watts, D.: Nonlinear regression: iterative estimation and linear approximations. Wiley Online Library (1988)

4. Jang, J.-S., Sun, C.-T., Mizutani, E.: Neuro-fuzzy and soft computing; a computational approach to learning and machine intelligence. (1997)

5. Montgomery, D., Montgomery, D., Montgomery, D.: Design and analysis of experiments 7. Wiley New York (1984)

6. Vahidi, A., Stefanopoulou, A., Peng, H.: Recursive least squares with forgetting for online estimation of vehicle mass and road grade: theory and experiments. Vehicle System Dynamics 43(1), 31-55 (2005)

7. Hagan, M., Demuth, H., Beale, M., others: Neural network design. Pws Pub. Boston (1996)

8. Pandya, A., Macy, R.: Pattern recognition with neural networks in C++. CRC press (1995)

9. Omondi, A., Rajapakse, J.: FPGA implementations of neural networks 365. Springer (2006)

10. Glover, F., Kochenberger, G.: Handbook of metaheuristics. Springer Science \& Business Media (2003)

11. Osman, I., Kelly, J.: Meta-heuristics: an overview. In : Meta-Heuristics. Springer (1996) 1-21

12. Storn R., P.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341-359 (1997)

13. Qin A.K., H.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13(2), 398-417 (2009)

14. Mezura-Montes E., V.-R.: A comparative study of differential evolution variants for global optimization., vol. 1, pp.485-492 (2006)

15. Chakraborty, U.: Advances in Differential Evolution, Mathematics \& Computer Science Department, University of Missouri, St. Louis. USA, Springer-Verlag Berlin Heidelberg (2008)

16. Price, K., Storn, R., Lampinen, J.: Differential evolution: a practical approach to global optimization. Springer Science \& Business Media (2006)

17. Hiendro, A.: Multiple switching patterns for SHEPWM inverters using differential evolution algorithms. International Journal of Power Electronics and Drive Systems 1(2), 94-103 (2011)

Capítulo 5 Diseño e implementación en FPGA de métodos de la Inteligencia Computacional

5.1. Introducción

En este capítulo se presentan los diseños de tres módulos correspondientes a los componentes más importantes de un modelo de inteligencia computacional. a) el modelo de lógica difusa Sugeno de primer orden con operadores paramétricos utilizando la representación modular de las redes neuronales, b) el algoritmo de evolución diferencial, como caso particular de una heurística de algoritmos evolutivos.

5.2. Implementación del modelo neuro-difuso Sugeno paramétrico

A continuación, se presentan los esquemas lógicos de la implementación en FPGA y los resultados de los módulos de un sistema difuso Sugeno de primer orden. Se presenta el módulo para implementar funciones de membresía triangulares, los operadores paramétricos de intersección difusa, así como la implementación del mecanismo de inferencia del modelo Sugeno.

5.2.1. Implementación de las funciones

En este trabajo se presenta una forma simple de implementación de funciones de membresía para n=2, 3, 4, 5 conjuntos difusos caracterizados con funciones de membresía triangulares, cuyo diseño puede ser extendido a un número n de conjuntos difusos deseado. Las características y restricciones del módulo se listan a continuación.

1. El universo de discurso de las variables de entrada está normalizado al dominio discreto de m bits. Siendo el intervalo de enteros [0,I] el dominio discreto, con valor máximo $I=2^m-1$.

2. Los valores de membresía de los conjuntos difusos, es decir el dominio difuso continuo [0,1] es mapeado igualmente al dominio discreto [0,I].

3. Los conjuntos difusos cumplen con la propiedad de *complitud-e*, con e=0.5, y solamente dos conjunto difuso pueden tener un valor de membresía diferente de cero, para cada valor del universo de discurso, cumpliéndose la siguiente condición:

$$\mu_{\overline{A_i}} = I - \mu_{A_i+1}$$

4. Para n conjuntos difusos, son necesarios n-2 parámetros de configuración. La figura 5.1 muestra un ejemplo para una variable de entrada con 5 conjuntos difusos y tres parámetros de configuración.

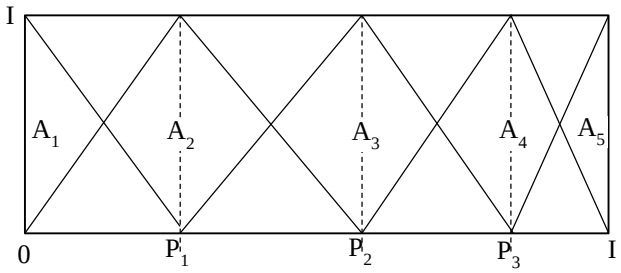


Figura 5.1 Funciones de membresía caracterizadas por *n*-2 parámetros

5. Dado que sólo dos conjuntos difusos pueden tener un valor diferente a cero, y el valor de membresía de conjuntos difusos contiguos son complementarios. Entonces es posible calcular el valor de membresía de todos los conjuntos difusos únicamente mediante la ecuación general de la recta como sigue.

$$y_+(x) = \frac{I(X - X_{min})}{X_{max} - X_{min}}$$

Donde los parámetros X_{min} y X_{max} son configurados de acuerdo al valor de entrada *x* y el número de conjuntos difusos que tienen la variable. En la tabla 5.1 se muestra las configuraciones de estos parámetros de acuerdo al número de conjunto difusos *n*.

Tabla 5.1 Configuración de los parámetros de la pendiente positiva

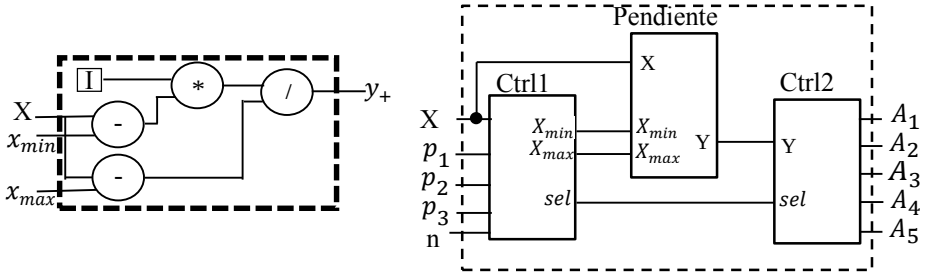
n=2	X_{min}	X_{max}	*sel*
Sin P	0	*I*	0

n=3	X_{min}	X_{max}	*sel*
$x \le P_1$	0	P_1	*0*
$x > P_1$	P_1	*I*	*1*

n=4	X_{min}	X_{max}	*sel*
$x \le P_1$	0	P_1	0
$x \le P_2$	P_1	P_2	1
$x > P_3$	P_2	*I*	2

n=5	X_{min}	X_{max}	sel
$x \le P_1$	0	P_1	0
$x \le P_2$	P_1	P_2	1
$x \le P_3$	P_2	P_3	2
$x > P_3$	P_3	*I*	3

La figura 5.2.a muestra el esquema lógico para implementar la ecuación de la recta, llamado *Pendiente*, mientras que en la figura 5.2.b se muestra el módulo de implementación del cálculo de las funciones de membresía para una variable con *n*=2, 3, 4 o 5 conjuntos difusos.



a) Bloque de Pendiente b) modulos de la función de membresía

Figura 5.2 Diagrama a bloques de la función de membresía.

El módulo Ctrl1, corresponde a la lógica mostrada en la tabla 5.1 mientras que el módulo Ctrl2 es implementado de acuerdo a la lógica que se presenta en la tabla 5.2

Tabla 5.2 Decodificación de la pendiente de salida

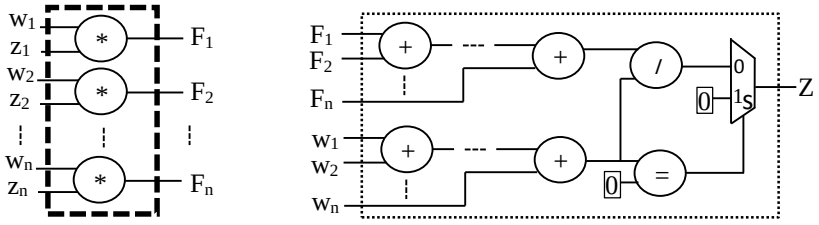
sel	A_1	A_2	A_3	A_4	A_5
0	$\overline{y_+}$	y_+	0	0	0
1	0	$\overline{y_+}$	y_+	0	0
2	0	0	$\overline{y_+}$	y_+	0
3	0	0	0	$\overline{y_+}$	y_+

5.2.2. Cálculo del valor certero de salida de los sistemas difusos

El valor de salida del sistema difuso Sugeno de primer orden, se calcula mediante el factor de la suma de productos $w_i * z_i$ entre la suma de los pesos calculados en el módulo *regla-antecedente*, como lo indica la siguiente ecuación:

$$z_C = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

En la implementación del numerador se realizan *n* multiplicaciones de las entradas w_i por las correspondientes *n* entradas que implementan a las funciones lineales de salida z_i. De manera general, se deben desarrollar *n* módulos que multipliquen las entradas w_i y z_i como puede observase de la figura 5.3.a. Posteriormente, se debe implementar un módulo donde se sumen las multiplicaciones, y otro módulo donde se suman los pesos w_i, y finalmente, se realiza la división algebraica correspondiente. Ver la figura 5.3 b.



a) Multiplicación de W_i*Z_i b) Obtención de valor certero.

Figura 5.3 Modulo de inferencia difusa del modelo Sugeno

5.2.3. Función de salida del modelo Sugeno de primer orden

La función de salida de salida para el cálculo de los consecuentes, corresponde a un polinomio de la forma siguiente:

$$z_i = a_i * x + b_i * y + c_i,$$

donde i = 0, 1, …, 8, y corresponde a la regla difusa *i*.

Podemos observar que si los coeficientes a_i y b_i son igual a cero, entonces la función de salida se modela por medio de una función *singletone*, que es la forma más simple en que pueden ser modelados los sistemas difusos tipo Mamdani. Si todos los coeficientes de la ecuación a_i, b_i y c_i son igual a cero, entonces se puede interpretar que la regla difusa R_i, no se agregó en la base de reglas del modelo a diseñar y, por lo tanto, con este modelo también es posible identificar la estructura del sistema. El diagrama lógico para implementar el módulo de una ecuación lineal se muestra en la figura 5.4., el cual está compuesto por dos multiplicadores y dos sumadores.

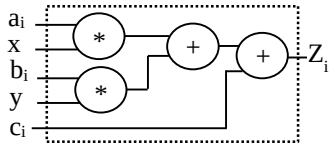


Figura 5.4. Diagrama lógico del módulo polinomio de conjuntos consecuentes del modelo Sugeno.

5.3. Implementación de la operación de conjunción difusa paramétrica

El diseño en FPGA de la representación unificada de las metodologías de suma monotónica, la versión simplificada de la suma ordinal de t-normas y t-subnormas, y el método de extensión de t-normas a partir de la t-norma drástica generadoras de familias t-normas y conjunctores paramétricos se presenta en esta sección, que está basada en los resultados de [**1**]. Además, se realiza un análisis comparativo de tiempo de latencia y los recursos utilizados en la implementación el diseño de implementación de cada una de estas metodologías por separado.

La tabla 5.3 lista la clasificación de las metodologías generadoras de operadores de intersección, paramétricos mediante un código de selección que es implementado como entrada de configuración del operador. La figura 5.5 muestra el diagrama a flujo de la configuración del operador de intersección.

Tabla 5.3 Clasificación de los conjunctores y t-normas de acuerdo al tipo de configuración

Código	Tipo de método
0,1	No-conmutativa 4T
2,3	Conmutativa 4Tc
4	3DT
5	3DTs
6	TMMT
7	TMMTs

5.3.1. Diseño en FPGA de las operaciones paramétricas

En esta sección se presenta el diseño y la implementación en un FPGA de los conjunctores y t-normas digitales paramétricas. En primer lugar se muestran los esquemas lógicos de la implementación de las t-normas básicas, posteriormente, se muestra la metodología para construir los conjunctores y las t-normas paramétricas en un sólo operador.

5.3.2. Implementación de t-normas básicas

La figura 5.6 muestra los esquemas de la implementación de las t-normas básicas: a) drástica, b) Lukasiewicz, c) producto (observar que el producto digitalizado es un conjunctor), d) nilpotente y e) mínimo, los cuales tienen una entrada más que los diseños mostrados en la implementación de estas t-normas en [**2**,**3**,**4**,**5**], a excepción del mínimo. La entrada *I* es utilizada para especificar la entrada del dominio paramétrico de una t-norma, donde si la t-norma paramétrica a generar tiene una representación de *m*-bits, los valores de la entrada *I* pertenecen al intervalo $[0,2^{m+1}-1]$.

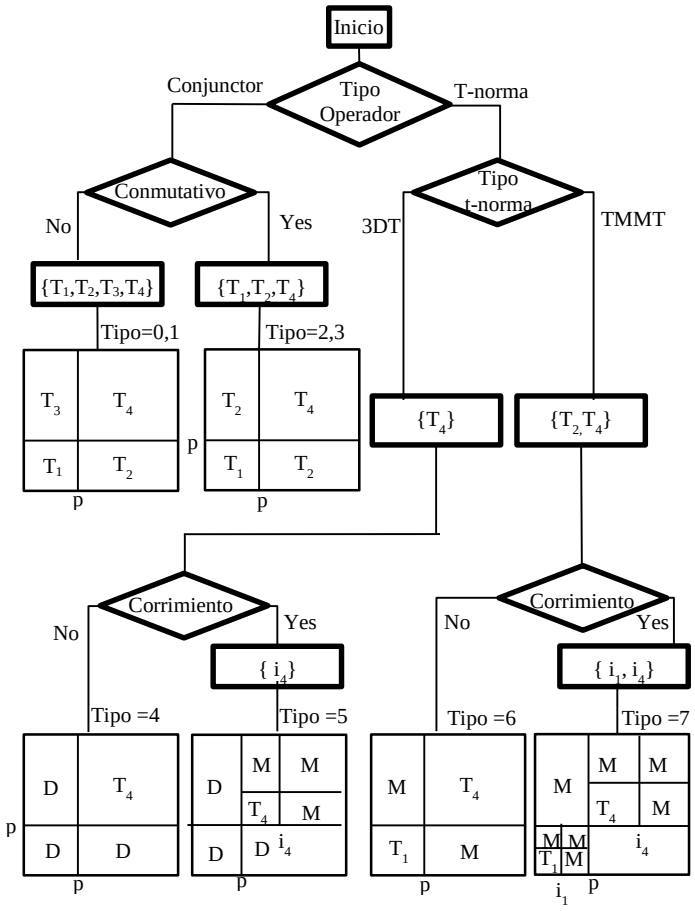


Figura 5.5 Diagrama de flujo para la selección del operador difuso en la implementación en FPGA

Se puede observar igualmente que en las implementaciones de las t-normas básicas son utilizados únicamente circuitos lógicos combinatorios simples, tales como, sumadores, compradores multiplexores, multiplicadores (el cual puede ser implementado ya se utilizando multiplicadores embebidos en el FPGA o mediante un arreglo de sumadores y multiplexores).

Ya que las t-normas básicas digitales son válidas en el intervalo [0, *n*] (en lugar del intervalo unitario), algunas modificaciones en sus definiciones y observaciones se presentan:

a) Debido a que el tamaño de la entrada *I* está en (*m*+1)-bits, las operaciones (circuitos combinatorio) dentro de cada módulo deben ser definidas para (*m*+1)-bits, a pesar de que las entradas *X*, *Y* y la salida T_i están definidas en *m*-bits.

b) Las operaciones de (*m*+1)-bits evitan el sobreflujo en la operación (*X*+*Y*) de las t-normas nilpotente y Lukasiewicz.

c) Se realiza una implementación especial de la t-norma Lukasiewicz, para evitar operaciones de números negativos. Cuando *X*+*Y* < *I*, es decir cuando la operación *X*+*Y*- *I* es menor que cero, el *MSB* (bit más significativo) de la resta mostrada en figura 5.6.b es puesto a 1, y este resultado puede considerarse como incorrecto, por otro lado cuando *X*+*Y* ≥ I, el *MSB* en la operación de resta es puesto a cero. Entonces el bit *MSB* de la operación de la resta se puede tomar como una señal de control de la entrada de selección del multiplexor, para seleccionar las entradas *X*+*Y*-*I* o cero, y generar la salida correcta de la t-norma Lukasiewicz

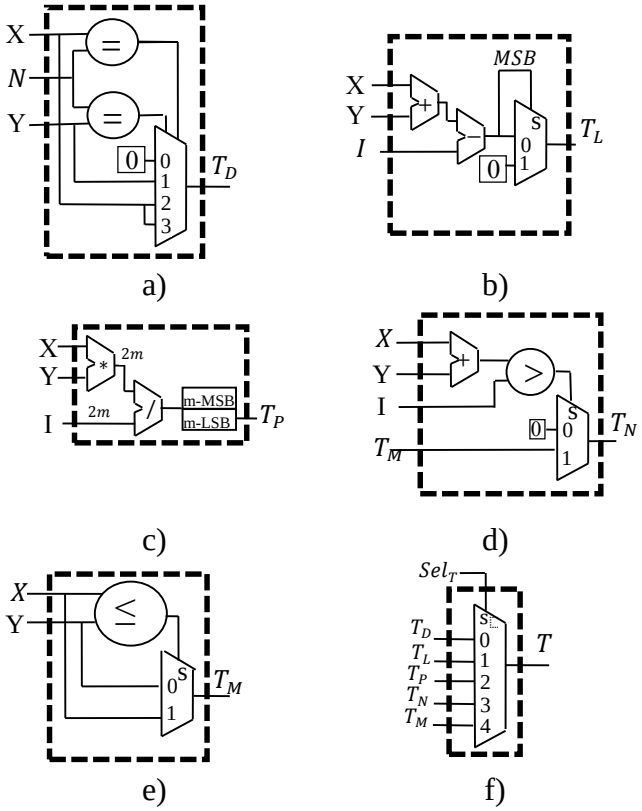


Figura 5.6 Esquemas lógicos de las t.normas básicas: a) Drástico, b) Lukasiewicz; c) Producto; d) Nilpotente; e) Mínimo; f) T-normas multiplexadas

d) Debido a que la multiplicación en representación entera es una operación de *m* a 2*m* bits, el resultado de esta operación debe ser normalizado a enteros de *m*-bits. Entonces la definición de la t-norma T_p es modificada como sigue

$$T_P(x,y) = \frac{x * y}{I}$$

Con esta normalización, se asegura que todos los resultados estén en el intervalo *L*, la salida T_P se le asignan los bits menos significativos de la salida de la división como se muestra en la figura 5.6.c. Las desventajas en esta modificación son, por un lado, el agregar un divisor que conlleva a un costoso consumo de recursos y un tiempo de latencia más lento, como se mostrará en la siguiente sección. Por otro lado, la operación de multiplicación digital no satisface la propiedad asociativa debido a discretización de los resultados, por lo tanto, estrictamente este operador no es una t-norma si no un conjunctor y será referido en esta investigación como una cuasi-t-norma. Por estas razones, en la siguiente sección se presentan los resultados de la implementación de los operadores paramétricos que incluyen y que no incluyen la cuasi-t-norma producto de manera separada para realizar un breve análisis comparativo de estas t-normas.

Para obtener un conjunctor o t-norma paramétrico generada a partir de t-normas básicas, se han multiplexado las diferentes módulos que implementan a las t-normas básicas como se muestra en la figura 5.6.f, utilizando ya sea un multiplexor 4x1 o un multiplexor 5x1 cuando es incluida la t-norma producto. La configuración de las señales de entradas del multiplexor puede ser realizada como se muestra en la tabla 5.4.a o la tabla 5.4.b en el caso que 5 t-normas básicas son consideradas.

Tabla 5.4 Selector de t-normas

a) 4 t-normas básicas

T_i	*T*
0	Drástico
1	Lukasiewicz
2	Nilpotente
3	Mínimo

b) 5 t-normas basicas

T_i	*T*
0	Drástico
1	Lukasiewicz
2	Producto
3	Nilpotente
4	Mínimo

5.3.3. Implementación de conjunctores y t-normas parámetricas

En el diseño de un operador paramétrico unificado para la generación de familias de conjunctores y t-normas paramétricos, es necesario definir algunos parámetros de entradas, los cuales se pueden clasificar como parámetros de ajuste y parámetros de configuración. Los parámetros de ajuste son *p*, i_1 and i_2, los cuales pueden tomar diferentes valores en la definición de una familia de t-normas o conjunctores específica.

Los parámetros de configuración T_i son utilizados para especificar las diferentes familias de conjunctores o t-normas que pertenezcan al método definido por el parámetro de configuración *type*. A pesar de que el operador de conjunción difusa paramétrico tiene 8 parámetros de entrada, los valores de estas entradas no siempre son tomadas en cuenta en la definición de una familia de conjunctores o t-normas para algún método especifico.

La tabla 5.5.a muestra los parámetros de entrada que son considerados de acuerdo al método especificado por la entrada *type*, en la tabla 5.5.b se muestra la información acerca del número de bits que se asigna en cada entrada. La entrada type no se incluye en la tabla 5.5a, sin embargo, es claro que esta entrada es necesaria en el esquema general. Las entradas X, Y y la salida Z tampoco se incluyen en la tabla por que estas no son parámetros, si no las entradas y la salida del operador.

Tabla 5.5 Parámetros de entrada considerados en las metodologías de generación de conjunctores y t-normas

a)Parámetros de entrada clasificados por metodologías

Metodología	Parámetros de ajuste	Parametros de configuración
4T	p	T_4, T_3, T_2, T_1
4Tc	p	T_4, T_2, T_1
3DT	p	T_4
3DTs	p, i_1	T_4
TMMT	p	T_4, T_1
TMMTs	p, i_1, i_2	T_4, T_1

b) Número de bits de cada parámetro de entrada

Parámetro de entrada	Número de bits
p	m
i_1, i_2	$m+1$
T_i	2 (or 3)
typ	3

La figura 5.7 muestra los esquemas lógicos de la implementación en FPGA de los seis diferentes métodos generadores de conjunciones difusas paramétricas. Observando estos esquemas se pueden encontrar las siguientes características: a) los esquemas tienen diferentes conjuntos de entradas; b) en todos los esquemas se implementa de un módulo de control, llamado ***Ctrl1*, e**ste módulo es creado para dividir el espacio de entrada *LxL* en cuatro regiones D_i como se especifica en la tabla 5.6, y es implementado a través de comparadores simples; c) la implementación de cualquier método generador de t-normas requiere la adición de dos restadores y un sumador para calcular la t-norma especificada en la región D_4. Por lo tanto, multiplexores de dos entradas son implementados para elegir las entradas y las salidas del módulo t-norma, los cuales son controlados mediante la operación *and* en la salida del módulo ***Ctrl1*,** referida como *xyGEp* (que significa semánticamente si *X* y *Y* son mayores a *p*); d) para la implementación de cualquier metodología generadoras de familias de t-norma con dominio de corrimiento, se creó un módulo de control, llamado ***Ctrl2***, que controla directamente al selector del módulo t-norma, en función de los valores de las entradas D_i, *X, Y, I* and T_i, la creación de dos módulos de control es significativa para reducir la complejidad del diseño.

Tabla 5.6 División de la región *LxL*

Entradas	D_i
x<p & y<p	D_1
x<p & y≥p	D_2
x≥p & y<p	D_3
x≥p & y≥p	D_4

5.3.4. Implementación del operador paramétrico general

La implementación del operador paramétrico que incluye las seis metodologías presentadas en la sección anterior, utiliza la implementación de las partes comunes de las diferentes familias mostradas en la figura 5.7, junto con un módulo de *Ctrl1*, que generaliza las diferencias de todas estas (Ver la figura 5.8). El módulo *Ctrl1* utiliza los valores de las entradas *X, Y,* D_i, i_i, t_i y *type* para

configurar las salidas sel_t, sel_l y *xyGE*, que son conectadas a las entradas del módulo t-norma y los multiplexores como se especifica en la tabla 5.7.

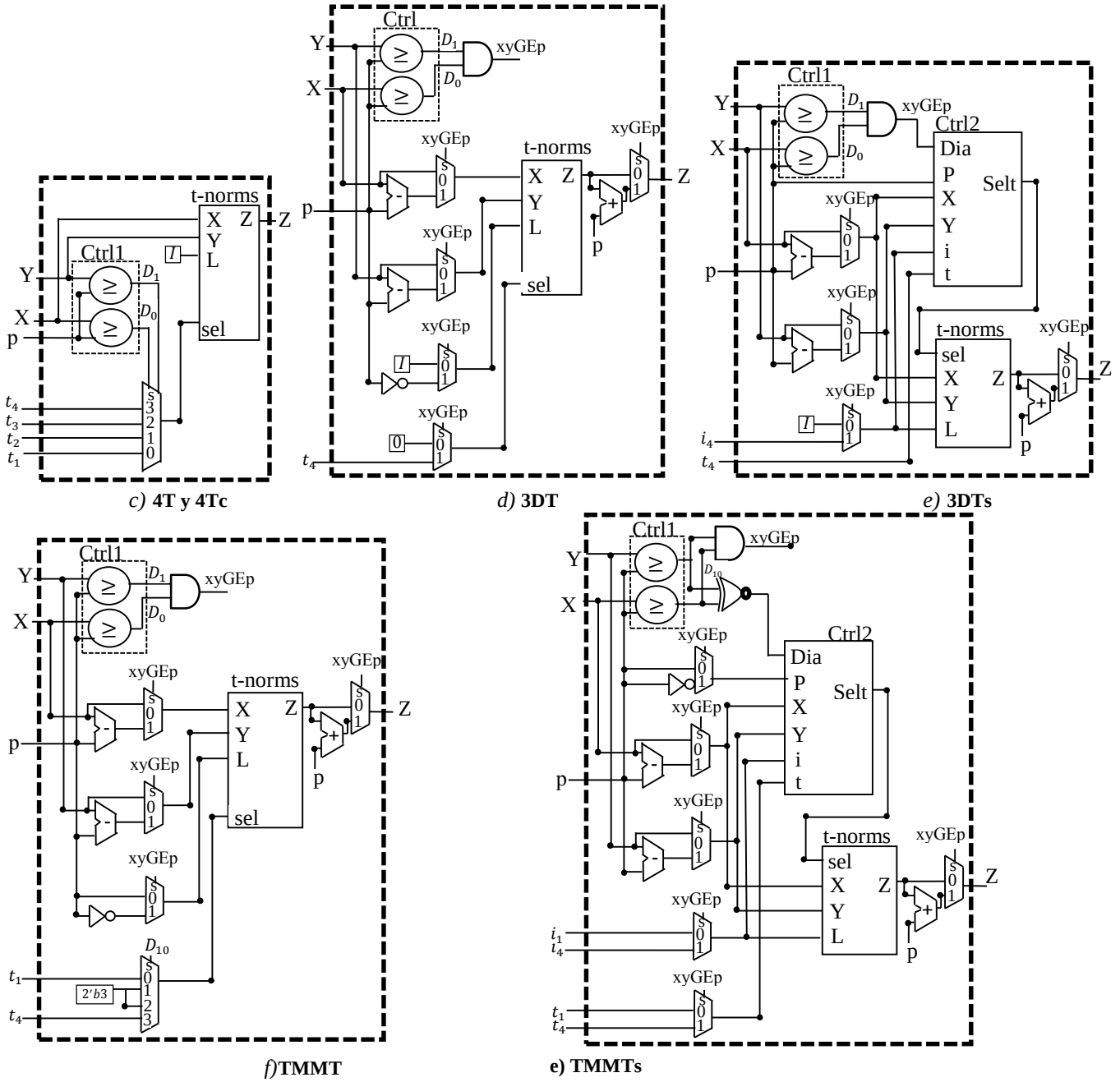


Figura 5.7 Esquemas de los métodos de generación de familias conjunctores y t-normas paramétricos

El parámetro de entrada *p* se utiliza para dividir el espacio de entrada en cuatro secciones: D_1, D_2, D_3 and D_4 como se indicó en la tabla 5.6, donde los valores de p pertenecen al intervalo [0, *n*]. La señal *xyGEp* es puesta en 1, cuando la tupla de entrada (*x*,*y*) se encuentra en la región D_4 y la entrada type es mayor a 3; en otro caso la señal *xyGEp* es puesta a 0. Los selectores de entrada de los módulos t-norma y el multiplexor MUL_L, son configurados según el valor de la señal *type*, que

especifica la metodología a configurar, y de algunos de los valores de los parámetros de entrada i_1, i_4, *X*, *Y*, *p*, según la metodología utilizada.

Tabla 5.7 Condiciones para la configuración del módulo de control *Ctrl*

type	$x, y \in D_1$			$x, y \in D_2$			$x, y \in D_3$			$x, y \in D_4$		
	sel_T	sel_L	*xyGEp*	sel_T	sel_L	*xyGEp*	sel_T	sel_L	*xyGEp*	sel_T	sel_L	*xyGEp*
0,1	T_1	*n*	*0*	T_2	*N*	*0*	T_3	*n*	*0*	T_4	*n*	*0*
2,3	T_1	*n*	*0*	T_2	*N*	*0*	T_2	*n*	*0*	T_4	*n*	*0*
4	T_D	*n*	*0*	T_D	*N*	*0*	T_D	*n*	*0*	T_4	*n-p*	*1*
5	T_D	*n*	*0*	T_D	*N*	*0*	T_D	*n*	*0*	$[T_4 \vert T_M]^{**}$	i_4	*1*
6	T_1	*p*	*0*	T_M	*N*	*0*	T_M	*n*	*0*	T_4	*n-p*	*1*
7	T_1	i_1	*0*	$[T_1 \vert T_M]^{*}$	*N*	*0*	T_M	*n*	*0*	$[T_4 \vert T_M]^{**}$	i_4	*1*

*La primer t-norma es seleccionada cuando las entradas *x,y* son menores al min (*p*, i_1), T_M es seleccionada en otro caso
**La t-norma T_4 es seleccionada cuando las entradas *x-p*, *y-p* son menores o iguales a min (I-p, i_4), T_M es seleccionada en otro caso

Por simplicidad del diseño, el módulo de control *Ctrl* fue implementado utilizando una descripción por comportamiento, esto es, el comportamiento de la lógica mostrada en la tabla 5.7, fue descrita utilizando sentencias de lenguaje de descripción de hardware Verilog switch-case e if-else [**6**,**7**].

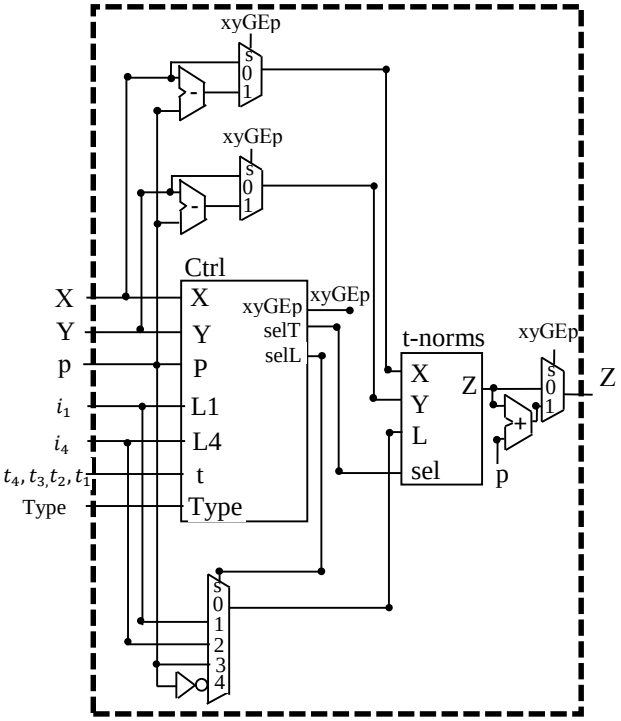


Figura 5.8 Esquema lógico de la implementación del operador paramétrico que unifica las seis metodologías que generan familias de conjunctores y t-normas paramétricas

Nótese que en la implementación de las familias de conjunctores no se está validando los posibles errores de configuración del usuario, esto es, no se verifica que las condiciones $T_1 \leq T_2 \leq T_4$ y $T_1 \leq T_3 \leq T_4$ se satisfagan. Un módulo para validar estas condiciones no es una tarea difícil de implementar, sin embargo, en este trabajo se ha considerado que la implementación de este módulo

afectaría considerablemente el tiempo de latencia del operador, y como está validación es necesaria únicamente en la especificación de la generación de familias conjunctores, se decidió no incluir este módulo en la implementación del operador de conjunción paramétrico, y por lo tanto, la generación de familias de conjunctores validos es responsabilidad del usuario.

La selección de las t-normas básicas en las regiones generadas por *p*, es definida por el usuario o por defecto, esto de acuerdo a la metodología especificada por la entrada *type.* El usuario puede asignas las t-normas básicas especificándose los parámetros de configuración T_1, T_2, T_3, T_4 según la tabla 5.4.a y 5.4.b (dependiendo del número de t-normas básicas implementadas en el operador). Por ejemplo, si el usuario desea definir las t-normas básicas Lukasiewicz y nilpotente en las secciones D_1 and D_4 respectivamente, entonces se deben asignar los valores $T_1 = 1$ and $T_4 = 2$.

Cuando la metodología 4Tc, que genera familias de conjunctores conmutativos es seleccionada, la entrada *type*, se le debe asignar el valor de la entrada T_2 a la entrada T_3, entonces, el valor configurado en entrada T_3 por el usuario no importa, debido a que las regiones D_2 y D_3 se les asigna la t-norma básica configurada en la entrada T_2. Para los casos en que las metodologías asignadas con *type*= 4, 5 que generan t-norms paramétricas, únicamente el valor de T_4 es definido por el usuario, y los valores de las otras entradas T_{i} se les asigna por default el valor T_D. Finalmente, en la selección de las metodologías referenciadas con *type*=6, 7, los valores de las entradas T_2, T_3 no importan, ya que las regiones correspondientes se les asigna la t-norma T_M por defecto.

Como fue mencionado en el capítulo tres, las t-normas paramétricas generadas por las metodologías del tipo 5 and 7, tienen dominios de corrimiento, por lo tanto el diseño considera entradas de (*m*+1)-bits para los especificar los parámetros i_1 y i_4 para configurar los dominios de las t-normas T_1 (sólo para el tipo 7) y T_4 respectivamente.

5.3.5 Resultados

El diseño de la implementación en hardware en FPGA de los operadores paramétricos fueron implementados y verificados en el dispositivo EP4CE115F29C7 de la familia Cyclone IV E de Altera [**8**] utilizando el lenguaje de descripción de hardware Verilog [**6**,**7**] Se implementaron dos diseños con cuatro diferentes tamaños de representación digital, que son *m*=5, 8, 16 y 32 bits cada uno; el primer diseño incluye la cuasi-t-norma producto como una t-norma básica, mientras que el segundo diseño, no incluye esta t-norma.

Se realizó un análisis comparativo del tiempo de latencia y los recursos utilizados en diferentes implementaciones en el dispositivo EP4CE115F29C7, las cuales son: a) las t-normas básicas; b) las seis metodologías de generación de conjunctores y t-normas y; c) El operador paramétrico general de conjunción difusa.

La tabla 5.8 presenta los tiempos de latencia y los recursos utilizados de la implementación de las t-normas básicas, de las cuales se puede notar que todas las t-normas a excepción de la t-norma producto tienen resultados similares para sus respectivas implementaciones. La t-norma producto presenta un tiempo de latencia 40 veces más lento, y requiere de 10 veces más recursos aproximadamente comparadas con las otras t-normas básicas, por lo que sólo es recomendable incluir esta t-norma en el operador de conjunción difusa paramétrico general sólo cuando sea realmente necesario.

En la tabla 5.8 también se muestra los recursos utilizados y los tiempos de latencias del bloque que multiplexa e incluye a las t-normas básicas, y como se puede observar el tiempo de latencia es aproximadamente 14% más lento que el tiempo de latencia de la t-norma básica más lenta, y que el número de recursos utilizados es menor a la suma de los recursos de las t-normas individuales, lo cual es posible debido a que las funciones combinacionales pueden ser mapeadas en las mismas áreas reconfigurables (elementos lógicos) del dispositivo.

Tabla 5.8 Costos de implementación de las t-normas básicas

T-norma o Conjunctor	m-bit	Funciones Combinacionales	Fmax (MHz)	Tiempo de Latencia (ns)
TM, Mínimo	5	9	337.38	2.96
	8	16	307.69	3.25
	16	32	241.66	4.13
	32	64	191.86	5.21
TN, Nilpotente	5	17	317.66	3.15
	8	25	300.48	3.33
	16	49	232.72	4.30
	32	97	182.65	5.47
TL, Lukasiewicz	5	17	268.89	3.72
	8	26	267.31	3.74
	16	50	217.39	4.60
	32	98	170.33	5.87
TD, Drástica	5	9	477.78	2.09
	8	14	415.11	2.41
	16	27	377.07	2.65
	32	54	252.53	3.96
TP, Producto	5	111*(1)	46.9	21.32
	8	279*(1)	23.47	42.61
	16	1079*(2)	8.7	114.94
	32	4296 *(8)	2.91	343.64
Bloque de t-normas multiplexadas	5	35	234.74	4.26
	8	54	201.41	4.96
	16	99	178.67	5.60
	32	189	148.68	6.73

* La información entre paréntesis indica el número de multiplicadores embebidos utilizados

La tabla 5.9 muestra los recursos utilizados y los tiempos de latencia de la implementación en el dispositivo EP4CE115F29C7 de las diferentes metodologías generadoras de familias de t-normas y conjunctores, correspondientes al diseño mostrado en la figura 5.7, cuando la t-normas básica producto no es incluida. Se puede ver que todos los módulos que implementan a las diferentes metodologías generadoras de t-normas tienen tiempos de latencia similares para el mismo tamaño

de *m*-bits y los módulos que implementan a las metodologías 4T y 4Tc, generadores de familias de conjuctores tienen un tiempo de latencia, y el uso de recursos es prácticamente igual al módulo que multiplexa a las t-normas básicas mostrado en la tabla 5.10.

Podemos concluir de las tablas 5.8 y 5.9 que el tiempo de latencia en la implementación de las metodologías generadoras de conjunctores es 30% más lento que la t-norma básica más lenta (T_L, Lukasiewicz) y que los recursos utilizados en la implementación de 4T y 4Tc son menores a la suma de los recursos utilizados al implementar las t-normas básicas, lo cual es posible debido a la misma razón explicada en la multiplexión de las t-normas básicas mencionada en el párrafo anterior. Los módulos que implementan las metodologías que generan familias de t-normas paramétricas (3DT, 3DTs, TMMT, TMMTs) utilizan tres veces más recursos que las implementación de conjunctores paramétricos, y su tiempo de latencia es dos veces más lento.

Tabla 5.9 Costo de implementación de las metodologías que generan familias de conjunctores y t-normas

T-normas o Conjunctores	m-bits	Funciones Combinacionales	Fmax (MHz)	Tiempo de Latencia (ns)
4T, y 4Tc	5	46	192.53	5.19
	8	74	190.77	5.24
	16	134	164.28	6.09
	32	257	128.83	7.76
3DT	5	91	113.52	8.81
	8	141	98.02	10.20
	16	272	76.81	13.02
	32	533	59.42	16.83
3DTs	5	115	106.87	9.36
	8	180	98.99	10.10
	16	344	78.2	12.79
	32	668	60.64	16.49
TMMT	5	92	105.21	9.50
	8	141	105.57	9.47
	16	272	76.64	13.05
	32	533	62.10	16.10
TMMTs	5	129	110.38	9.06
	8	202	98.91	10.11
	16	381	73.13	13.67
	32	739	63.67	15.71

En la tabla 5.10 se muestran los valores de los recursos utilizados y el tiempo de latencia en la implementación del operador de conjunción difusa general (que incluye la descripción de todas las metodologías), a) incluyendo y b) no incluyendo el cuasi t-norma básico producto. Sin la cuasi-t-norma producto los recursos utilizados son menores al 1% del total de los recursos del dispositivo [**9**], generando al igual que las implementaciones anteriores, una descripción totalmente combinatoria, es decir, los resultados son obtenidos en un ciclo de reloj, con una *fmax*=60.42 MHz para el caso de la representación de 32 bits.

Comparando estos resultados con los módulos que implementan a las seis metodologías por separado, se puede determinar que estos tienen un tiempo de latencia similar, y que los recursos

utilizados por el operador general sólo son un 14% mayor que los utilizados en la implementación de módulo que implementa a la metodología más compleja, es decir la metodología TMMTs. Por lo que se puede concluir a partir de los resultados de tabla 5.10, que no hay una degradación en el tiempo de latencia, y además, que los recursos necesarios son eficientes en la implementación del operador general paramétrico. Estos resultados de implementación pueden atribuirse a las siguientes causas: a) que los módulos que implementan a las diferentes metodologías tienen estructuras de descripción similares; b) a la optimización realizada por el sintetizador de Altera y; c) al mapeo de las diferentes funciones combinacionales en la misma área reconfigurable.

En la tabla 5.10, también se muestran tanto el tiempo de latencia, así como los recursos utilizados al implementar el operador general incluyendo la cuasi-t-norma producto, y como era de esperarse, el costo de la implementación del operador que incluye al producto es mucho mayor que cuando no se incluye. Sin embargo, su implementación es factible ya que con la inclusión del producto, los recursos utilizados no sobrepasan el 4% de los recursos disponibles en el dispositivo y su tiempo de latencia está en el orden de los nanosegundos.

Tabla 5.10 Recursos y tiempo de latencia utilizados en la implementación del operador general paramétrico

	Con la cuasi-t-norma producto				**Sin la cuasi-t-norma producto**				**Total en FPGA**
Categoría	n=5	n=8	n=16	n=32	n=5	n=8	n=16	n=32	
Fuciones Combinacionales totales	242 (.2%)	467 (.4%)	1424 (1.2%)	5068 (4.4%)	162 (0.1%)	239 (0.2%)	442 (0.4%)	849 (0.7%)	114,480
Multiplicadores embebidos de 9-bits	1 (.2%)	1 (.2%)	2 (.4%)	8 (1.5%)	0	0	0	0	532
Fmax (MHz)	35.3	19.78	7.99	2.79	102.27	75.78	75.78	60.42	
Tiempo de latencia (ns)	28.33	50.56	125.16	358.42	9.52	10.94	13.90	24.24	
PIN	40	68	116	212	46	64	112	400	529

Finalmente, se presenta una comparación de los resultados obtenidos en este trabajo con los resultados presentados en [**10**], en el cual fueron presentados los costos de la implementación en hardware de una familia generadora de conjunctores paramétricos. En [**10**] se reportó que la implementación fue realizada en el FPGA 3E3S500EFG320-5 de Xilinx, utilizando 382 compuertas y un tiempo de latencia de 14.03 ns en una representación de 8 bits. Para una mejor comparación, en este trabajo, se implementó nuevamente ese operador en el FPGA de Altera utilizando las t-normas básicas presentadas en la figura 5.6 y los esquemas sugeridos en ese trabajo.

En la tabla 5.11 se presentan los costos en la implementación de la familias de conjunctores paramétricos presentado en [10], así como los costos de la implementación de algunas de las familias generadoras de conjunctores y t-normas paramétricas presentadas en este trabajo. Igualmente de la tabla 5.12 se puede inferir: a) que los conjunctores generados por las familias 4T y 4Tc tienen los tiempos de latencia más rápidos y con menor consumo de recursos; b) que la implementación del conjunctor paramétrico de [10] tienen un tiempo de latencia similar a las implementaciones de las familias de los t-normas paramétricas presentadas en este trabajo.

En este trabajo no vimos necesaria la implementación en hardware de las t-normas paramétricas tradicionales de Yager, Hamacher, Dombi, etc, analizadas ampliamente en la literatura, citando algunas en [11,12,13], ya que hemos deducido que estas no tienen una implementación tan eficiente en hardware como las presentadas en ese capítulo debido principalmente a: a) estas requieren de la implementación tanto la t-normas producto (del cual ya se ha hecho ver su desventaja) e incluso otras funciones matemáticas aún más complejas y; b) su descripción en hardware no es adecuada en una representación de números digital. Sin embargo, este análisis puede considerarse como un trabajo futuro, con el objetivo de realizar un análisis comparativo.

Tabla 5.11 Costo de la implementación individual de algunas familias de conjunctores y t-normas paramétricos

T-norma o Conjunctor	Familia	Bits	Funciones Combinacionales	Fmax (MHz)	Tiempo de Latencia (ns)
Operador de [**10**]		5	50	134.26	7.45
		8	82	121.97	8.20
		16	162	90.63	11.03
		32	322	68.67	14.56
Lukasewicz	3DT	5	80	129.99	7.69
		8	121	114.82	8.71
		16	239	82.67	12.10
		32	462	65.92	15.17
Nilpotente	3DTs	5	111	115.17	8.68
		8	177	107.09	9.34
		16	340	80.76	12.38
		32	673	61.9	16.16
Lukasewicz - Nilpotente	TMMT	5	81	120.98	8.27
		8	126	108.67	9.20
		16	246	83.79	11.93
		32	486	65.56	15.25
Drástico-Lukasewicz	TMMTs	5	107	138.2	7.24
		8	156	115.19	8.68
		16	302	98.67	10.13
		32	596	74.34	13.45
LLLM	4Tc	5	36	207.51	4.82
		8	59	196.46	5.09
		16	115	179.82	5.56
		32	451	135.78	7.36

Para verificar la implementación en hardware de los diseños presentados se realizó el siguiente proceso: a) los modelos fueron simulados utilizando el simulador de Model-Sim de Mentor Graphics para la representación de 5 bits ($n \in [0\text{-}31]$); b) los diseños fueron sintetizados en el FPGA EP4CE115F29C7, utilizando la tarjeta de desarrollo DE2; c) Para la verificación de los resultados se utilizó el enfoque de verificación llamado hardware in loop (HIL), en el cual la tarea básica fue enviar todos los posibles valores de entrada (X,Y) desde un software en Matlab a la implementación hecha en el FPGA, y la salida Z del FPGA fue regresada al programa de Matlab para su análisis, para mayor detalle del enfoque HIL, obteniendo los mismos resultados correctos que los obtenidos en las simulaciones; d) los datos obtenido del FPGA fueron graficados utilizando el programa de Matlab (y que se presentan en la figura 5.8) para las implementaciones de los módulos con

representaciones numéricas de *n*=8, 16 and 32, sólo algunos valores de entrada-salida fueron verificados.

Tabla 5.12 Configuración del operador general de conjunción paramétrica generado de familias de t-normas y conjunctores paramétricos

Superficie de figura 5.8	Tipo	Metodología	t-normas utilizadas	sel_i $(octal)$	P	i_1	i_4
a)	0,1	4T	$T_1 = T_D, T_2 = T_L, T_3 = T_N, T_4 = T_M$	0134	11	X	X
b)	2,3	4Tc	$T_1 = T_D, T_2 = T_L, T_4 = T_M$	01X4	10	X	X
c)	4	3DT	$T_4 = T_P$	0002	11	X	X
d)	4	3DT	$T_4 = T_D$	XXX0	11	X	X
e)	5	3DTs	$T_4 = T_L$	XXX0	11	X	15
f)	5	3DTs	$T_4 = T_L$	XXX0	11	X	25
g)	6	TMMT	$T_1 = T_L, T_4 = T_D$	1XX0	11	X	X
h)	6	TMMT	$T_1 = T_L, T_4 = T_L$	1XX1	11	X	X
i)	7	TMMTs	$T_1 = T_L, T_4 = X$	1XXX	31	22	X
j)	7	TMMTs	$T_1 = T_L, T_4 = T_M$	1XX4	25	35	x
k)	7	TMMTs	$T_1 = T_L, T_4 = T_N$	1443	11	15	25
l)	7	TMMTs	$T_1 = T_D, T_4 = T_L$	1XX3	11	18	15

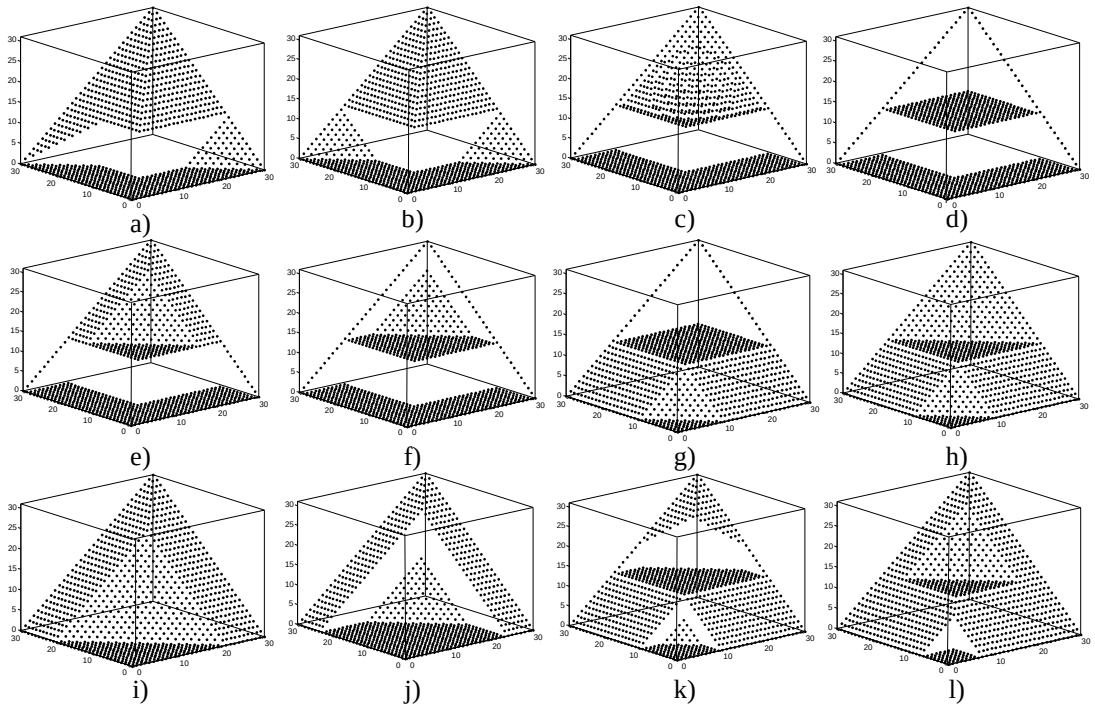


Figura 5.9 Superficies de las t-normas y conjunctores correspondientes a la tabla 5.13

La figura 5.9 muestra las superficies correspondientes a las t-normas y conjunctores paramétricos especificados en la tabla 5.12, indicando los valores de las entradas paramétricas. Hay que notar que las celdas con valor X, corresponde a la condición de no importa, por lo que sus valores no son considerados. Con la figura 5.9, y la tabla 5.12, se intentan mostrar casos más

generales de la configuración del operador general paramétrico, ya que es impráctico mostrar cada una de las familias que se pueden generar de este operador.

5.3.6. Observaciones

Se presentó la implementación en un FPGA de un operador de conjunción difusa paramétrico generalizado que puede ser configurado como un conjunctor o una t-norma en un sistema de inferencia difusa. El diseño presentado está basado en de las t-normas básicas drástica, Lukasiewicz, producto, nilpotente y mínimo. Este diseño se incluyen a) cinco t-normas básicas, b) diferentes métodos de agregación de t-normas que son: el método de suma ordinal de t-normas, el método de suma ordinal de t-subnormas basadas en t-normas básicas, el método de suma monotónica de t-normas, y una metodología basada en la extensión de t-normas básicas a través de la t-norma drástica [**14**,**15**,**2**,**16**].

La implementación eficiente en un esquema de varias metodologías generadoras de familias de t-normas y conjunctores paramétricos fue realizada utilizando una representación unificada de versiones simplificadas de las diferentes metodologías de agregación de t-normas consideradas en [**15**,**16**].

Este diseño tiene una implementación eficiente en el FPGA con similar tiempo de latencias, que la implementación por separado de las diferentes metodologías presentadas y los recursos utilizados son menores al 1% del total de recursos del FPGA EP4CE115F29C7.

Este diseño puede ser extendido de manera directa para incluir la metodología de suma monotónica (*p*,*I*-*p*) presentada en [**3**], haciendo una simple modificación en el módulo de control.

Se presentaron los resultados de la implementación de este operador considerando representaciones digitales menores o iguales a 32 bits, debido a que los autores consideramos que si se desea una mayor precisión en la representación es mejor utilizar la representación de punto flotante.

El diseño presentado no incluye un módulo de validación para asegurar que las configuraciones en la implementación de conjunctores por el usuario sean válidas, y esto fue debido a que tal módulo hace más lento el diseño del operador general. Para resolver este problema, se recomiendan dos posibles soluciones: a) realizar la validación en software o b) implementar el módulo de validación en hardware; un diseño con validación fue presentado en [**2**,**17**] y puede servir como referencia.

Como se mostró en capítulo 3, el diseño presentado puede ser configurado para obtener 135 operador diferentes operadores de conjunción o t-normas paramétricas.

5.4. Implementación del algoritmo de evolución diferencial utilizando representación de punto flotante de doble precisión

5.4.1. Módulos de punto flotante de FPGA de Altera

Altera provee una librería de acceso libre de bloques de propiedad intelectual (IP) llamados *Megafunciones* [**8**,**9**]. Entre estas Megafunciones de Altera podemos encontrar bloques parametrizables para la implementación de operaciones de punto flotante que se incluirán para el desarrollo de este trabajo. La tabla 5.13 muestra el número de recursos utilizados por los operadores de suma, comparación y multiplicación para operandos de punto flotante de doble precisión, es decir Megafunciones para operaciones aritméticas de punto flotante de 64 bits que adecuadas para el dispositivo EP4CE115F29C7.

Tabla 5.13 Características de recursos de los operadores de punto flotante de las megafunciones

Mega-función	Latencia	Elementos Lógicos	Registros Lógicos	Multiplicadores embebidos	F_{MAX} (MHz)
FPMULT	5	552	530	18	102
FPCOMP	1	176	2	-	-
FPAddSub	7	1534	584	-	105

Como se muestra en la tabla 5.13 las operaciones aritméticas utilizando la representación de punto flotante deben realizarse utilizando una implementación secuencial. Por lo que en este trabajo, se presenta un esquema lógico secuencial para realizar la implementación del algoritmo de evolución diferencial mediante una máquina de estados

5.4.2. Diseño en FPGA del algoritmo de evolución diferencial.

Como el algoritmo de evolución diferencial realiza operaciones de punto flotante únicamente en la generación del vector de prueba, entonces sólo es necesario un módulo de punto flotante. De la tabla 5.13 se observa que el módulo de comparación *FTCOMP*, es combinacional, ya este utiliza la misma lógica que los comparadores de enteros. El diseño de la implementación en hardware del DEA se describe a continuación.

El esquema del diseño de la implementación del DEA consiste de los siguientes módulos: a) un módulo de almacenamiento que contenga la información de los individuos, llamado *PMem*; b) un módulo para almacenar la aptitud de cada individuo de acuerdo a la función objetivo, *FXMem*; c) el módulo de evaluación de aptitud, *Fitness*; d) el módulo generador de vector de prueba, *CrossOver*; e) cuatro módulos generadores de números aleatorios; f) un módulo de control a través de una máquina de estados, para controlar la secuencia de ejecución del DEA (otra forma de realizar el control es mediante un procesador).

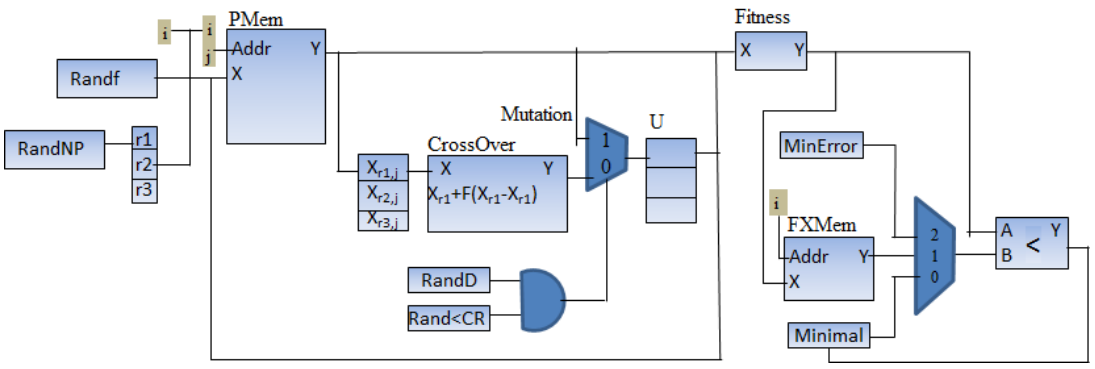


Figura 5.10 Esquema lógico de la implementación del algoritmo de evolución diferencial

La figura 5.10 muestra los módulos para realizar las operaciones necesarias en el DEA, también son mostrados los registros: *i*, *j*, utilizados para direccionar PMem y FXMex, tres registros para almacenar los índices r_1, r_2, r_3, tres registros de 64-bits para almacenar los valores de los atributos X_{r1}, X_{r2} and X_{r3} y un archivo de n registros de 64-bits para almacenar cada atributo del individuo de prueba. Algunos comparadores y multiplexores utilizados en el DEA no son presentados en este esquema general por motivos de simplificación del esquemático. A continuación, se presenta una descripción detallada de cada módulo.

5.4.3. Módulos de memoria

Módulo PMem

Este módulo es implementado utilizando la descripción lógica de una memoria de acceso aleatorio (RAM, por sus siglas en inglés), para almacenar los valores de la población de la generación actual. El tamaño de la memoria está determinado por el producto del tamaño de la población *NP* por la dimensión del problema *n*, por lo que el tamaño de la RAM queda expresada como:

$$PMem[TAM] = NP * D \ palabras$$

donde cada palabra es especificada por 8 bytes (64-bits), entonces el tamaño de *PMem* puedes ser especificado como sigue

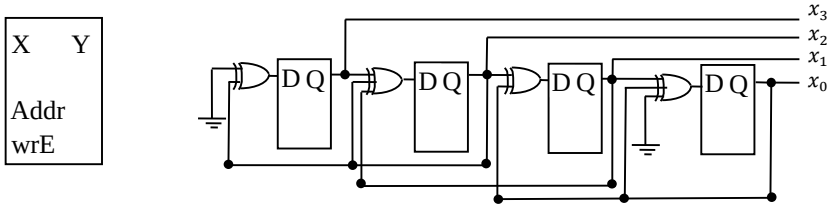
$$PMem[TAM] = NP * D * 8 \ \ bytes$$

Módulo FXMem

Este módulo es implementado de manera similar a *PMem*, con la diferencia de que el tamaño de *FXMem* sólo está en función del parámetro *NP*, debido a que sólo es necesario almacenar un valor de aptitud por cada individuo. Las expresiones que determinan el tamaño de la memoria *FXMem[TAM]* son:

$$FXMem[TAM] = NP\ words$$
$$FXMem[TAM] = NP * 8\ \ bytes$$

La figura 5.11.a muestra el diagrama lógico de la implementación de una *RAM*, donde *X* es la entrada de datos, *Y* la salida de datos, *Addr* es la entrada de dirección, y *wrE* se utiliza para habilitar a la memoria en modo de escritura, es decir para cuando se quiera almacenar un dato en la memoria.



a) Memoria b) Módulo generador de números aleatorios

Figura 5.11 Esquema lógico de los módulos a) RAM y b) Generador de números aleatorios de 4 celdas

5.4.4. Generador de números aleatorios

El circuito de un autónoma celular [**18**], ha sido ampliamente utilizado para generar números pseudoaleatorios. El módulo implementado en este diseño trabaja con dos reglas donde la primera está definida por $a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t)$ y la segunda regla es definida por $a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t)$, donde $a_i(t+1)$ representa el estado siguiente de la celda-i, $a_{i-1}(t)$ representa el estado actual de la celda-(*i*-1), también referenciado como vecino izquierdo de la celda celda-*i*, $a_i(t)$ representa el estado actual de la celda-*i*, $a_{i+1}(t)$ representa el estado actual de la celda-*(i*+1) o vecino derecho de la celda-*i* y $\oplus$ representa a operación lógica xor. Un autómata celular de *n* células, *n*-CA, puede generar 2^m-1 diferentes pseudo números aleatorios, donde *m* es el número de celdas. El esquema para *m*=4 celdas se presenta en la figura 5.11.b.

La implementación del DEA contiene tres diferentes módulos que generan valores enteros en los rangos: [0-*NP*], [0,*n*] y [0-127], para generar los números que pertenezcan al tamaño de la población, la dimensionad de la función, y una discretización de 128 valores del intervalo [0,1] respectivamente. Por otro lado, se implementa un módulo generador de número aleatorios de punto flotante de doble precisión que se utiliza para generar los valores de población inicial.

5.4.5. Módulo generador de vector de prueba

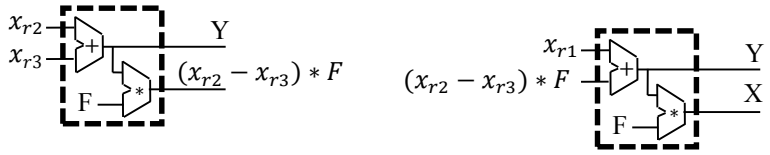
Las operaciones de punto flotante sólo son realizadas para generar el vector de prueba, de *n* dimensiones, según el algoritmo de evolución diferencial presentado en el capítulo 2.

$$v_{i,j} = x_{i,r1} + F * \left(x_{i,r2} - x_{i,r3}\right)$$

Donde $x_{i,r1}$, $x_{i,r2}$, $x_{i,r3}$ y F son valores de punto flotante. Las siguientes tres operaciones binarias de punto flotante se implementan:

$$R1 = (x_{i,r2} - x_{i,r3})$$
$$R2 = F * R1$$
$$v_{i,j} = x_{i,r1} + R2$$

Esta secuencia de operaciones se implementan mediante las megafunciones *FPMult* y *FPAddSub* mostradas en la tabla 5.13. Por lo tanto, el módulo *CrossOver* tiene una latencia de 22 ciclos de reloj para completar una operación. La figura 5.12 muestra el esquema de operador de cruza.



a) Modulo de Cruza: Etapa 1 b) Modulo de cruza: Etapa 2

Figura 5.12 Implementación del módulo de cruza

5.4.6. Módulo de funciones de aptitud

El módulo de funciones de aptitud depende de la aplicación a optimizar, por lo que este módulo cambia de una aplicación a otra, el cual puede representar a un modelo matemático, una red neuronal, un sistema de inferencia difusa, entre otros. En este trabajo se ha implementado un conjunto de seis funciones matemáticas que son utilizadas como funciones de prueba o pruebas de escritorio tradicionalmente utilizadas para la evaluación del rendimiento de algoritmos metaheurísticos de búsqueda. La tabla 5.14 y la figura 5.13, muestran la definición y los diagramas a bloques de estas funciones respectivamente.

Tabla 5.14 Funciones matemáticas de prueba

	Función	Ecuación
1	Modelo esfera	$f_1(x) = \sum_{i=1}^{D} x_i^2$
2	Problema Schwefel's 2.22	$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$
3	Problema Schwefel's 1.2	$f_3(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$
4	Problema Schwefel's 2.21	$f_4(x) = \max\{\lvert x_i \rvert, 1 \leq i \leq D\}$
5	Función Generalizada Rosenbrock's	$f_5(x) = \sum_{i=1}^{D} \lvert 100(x_{i+1-} x_i^2)^2 + (x_i - 1)^2 \rvert$
6	Funciónn Step	$f_6(x) = \sum_{i=1}^{D} \lvert (x_i + 0.5) \rvert^2$

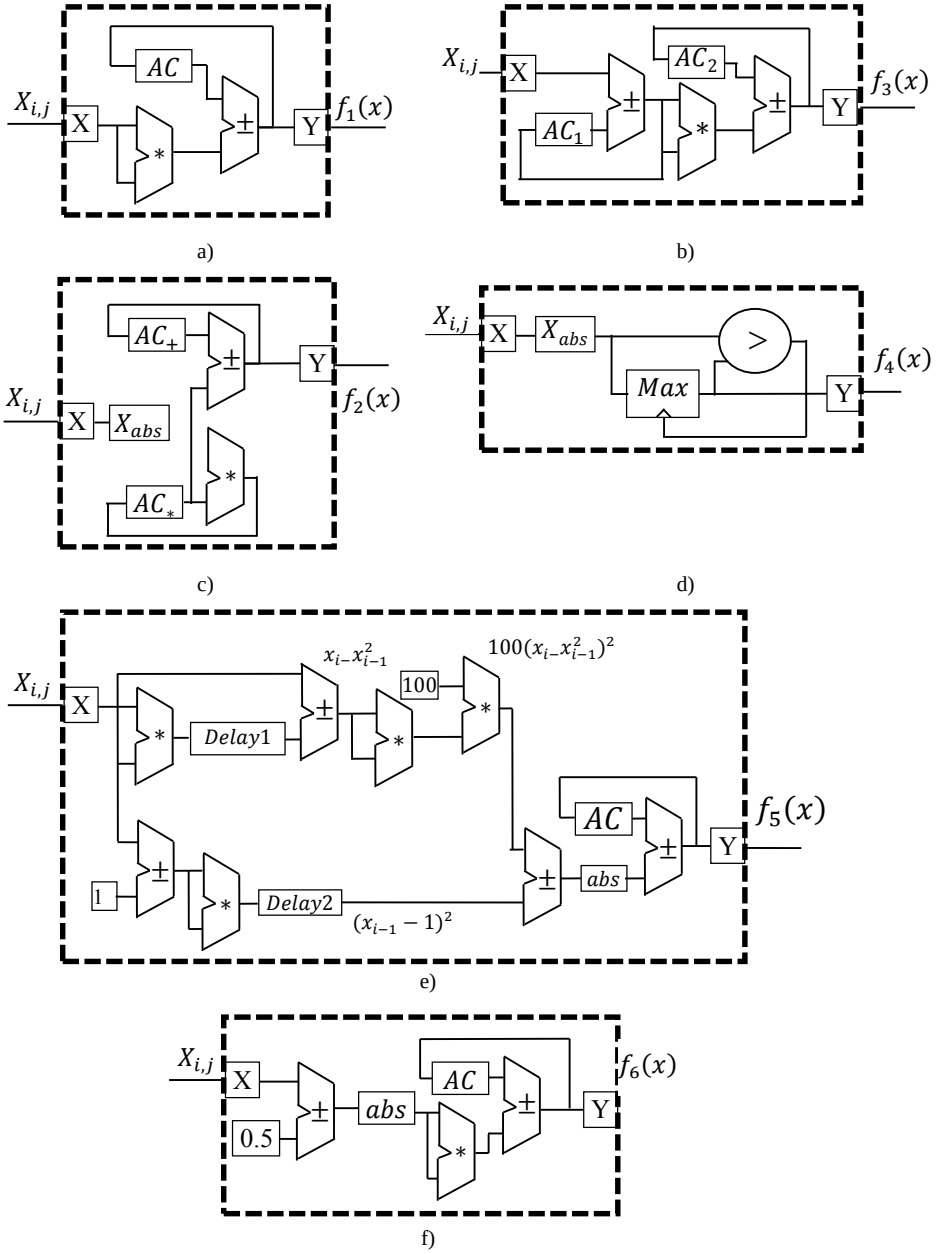


Figura 5.13 Implementación de las funciones objetivos de prueba

5.4.7. Módulo de control

Cada uno de los módulos descritos anteriormente contienen entradas de control, que son manejadas por este módulo implementado mediante una máquina de estados, con el objetivo de realizar el correcto funcionamiento del algoritmo.

5.4.8. Resultados

Los resultados presentados en la tabla 5.15 muestran los recursos consumidos en la implementación del DEA cuando la función esfera es implementada (*f1*) con una población de *NP*=128 elementos y una dimensionalidad *n*=32 en el dispositivo FPGA EP4CE115F29C7, de la familia cyclone IV de Altera. En la columna resultados se presenta el número de recursos utilizados, con referencial al número total de recursos disponibles por el dispositivo para diferentes categorías.

Tabla 5.15 Recursos consumidos en la implementación del DEA

Categoría	Resultados
Total de Funciones combinatorias (TCF)	10,330/114,480 (9%)
Registros Lógicos Dedicados (DLR)	5,366 /114,480 (5%)
Memoria total en bits (TMb)	8480/3,981,312 (<1%)
Multiplicadores embebidos 9-bits (EM9)	72/512(14%)
Fmax (MHz)	95MHz

La tabla 5.16 muestra los recursos utilizados por las diferentes funciones mostradas en la figura 5.13. Para evaluar el tiempo de latencia de la implementación del DEA, se realizaron dos implementaciones del conjunto de funciones de prueba, es decir, las funciones matemáticas de *f1* a *f6*, con los siguientes valores en los parámetros: a) *NP*= 128 y *n*= 32 y b) *NP*= 16 y *n*= 4, que se muestran en la tabla 5.17

Tabla 5.16 Resursos utilizados en la implementación de las funciones objetivo

Category	$\mathbf{f_1}$	$\mathbf{f_2}$	$\mathbf{f_3}$	$\mathbf{f_4}$	$\mathbf{f_5}$	$\mathbf{f_6}$
TCF	2405	2915	4397	213	8501	4281
DLR	1182	1494	1929	74	4551	1880
TMb	71	54	140	0	293	113
EM9	18	18	18	0	72	18
$F_{max(MHz)}$	95	95	70	122	71	80
Latencia	8clk*W	11clk* W+10clk	12clk*W	3clk*W	17clk*W	12clk*W

Los valores utilizados en los parámetros *CR* y *F* son tomados del análisis presentado en [**19**], donde se mostró que con estos valores, se obtiene el óptimo con el menor número generaciones. Los valores mostrados en la tabla 5.17, como son el número de generaciones promedio, *AveGen*, el tiempo promedio, *AveTime*, y el error, fueron obtenidos después de 20 corridas del algoritmo utilizando como condiciones de paro un error de $1e^{-12}$ y/o el número máximo de generaciones igual a 20,000.

5.4.9. Observaciones

Se presentó el diseño del algoritmo de evolución diferencial en un FPGA de Altera siguiendo un flujo secuencial, que utiliza tres parámetros de configuración definidos como parámetros de cruza-mutación, escalamiento y tamaño de población

Tabla 5.17 Tiempos consumidos de las diferentes funciones de objetivo

a) *NP*=128,*W*=32

f(x)	CR	F	AveGen	AveTime(s)	Error
$f_1(x)$	0.9	0.7	11571	26.67	$1.00E^{-12}$
$f_2(x)$	0.2	0.7	2802	7.7484	$1.00E^{-12}$
$f_3(x)$	0.9	0.7	20000	442.0546	0.0911
$f_4(x)$	0.8	0.7	20000	91.8609	0.089
$f_5(x)$	0.9	0.7	20000	119.9898	$1.45E^{-06}$
$f_6(x)$	0	0.7	1521	7.4476	$2.57E^{-13}$

b) *NP*=16, *W*=4

f(x)	CR	F	AveGen	AveTime	Error
$f_1(x)$	0.9	0.7	121	6ms	1.00E-12
$f_2(x)$	0.2	0.7	250	16ms	1.00E-12
$f_3(x)$	0.9	0.7	144	75ms	1.00E-12
$f_4(x)$	0.8	0.7	501	53ms	1.00E-12
$f_5(x)$	0.9	0.7	1683	.24s	1.00E-12
$f_6(x)$	0	0.7	101	12ms	1.00E-12

El diseño presentado no explota el enfoque paralelo porque se ha considerado que esta técnica es dependiente de la aplicación. Sin embargo, es posible notar que el paralelismo es más adecuado en la implementación de la función objetivo, ya que ésta presenta un evidente cuello de botella en muchas aplicaciones y su implementación es directa.

En este trabajo se describe la implementación del DEA considerado en [**20**], sin embargo, como existen diferentes modificaciones del DEA, teniendo como principales variantes: a) mayor número de individuos que participan en el proceso de la generación del vector de prueba, que normalmente incrementan en un número par de individuos, con el objetivo de obtener una mejor exploración del espacio de búsqueda; b) el uso del mejor individuo de la población como el principal ancestro, el cual conlleva a una mejor explotación de un subespacio (local) del espacio de búsqueda; c) la manera en la cual el operador de mutación-cruza es implementado. Para mayor detalle ver [**19**,**21**], siendo evidente que la implementación en FPGA de estas variantes es directa.

El diseño e implementación en FPGA del DEA fue publicado en la revista "Acta politécnica Hungarica" [**22**].

5.5. Referencias

1. Cortés-Antonio P., B.: Hardware Design of Digital Parametric Conjunctors and t-Norms. International Journal of Fuzzy Systems 17(4), 559-576 (2015)
2. Cortés-Antonio P., B.: FPGA implementation of (p)-monotone sum of basic t-norms., Barcelona, Spain (August 2010)
3. Cortés-Antonio P., B.: FPGA implementation of (p, I-p) – Monotone Sum of Basic t-norm., san francisco, CA, USA (May 2011)

4. Cortés-Antonio P., B.: FPGA implementation of fuzzy system with parametric membership functions and parametric conjunctions. Lecture Notes in Computer Science (9th Mexican International Conference on Artificial Intelligence) 6438 LNAI(PART 2 Advances in Soft Computing), 487-499 (2010)
5. Cortés-Antonio P., B.: FPGA Iimplementation of Parametric Fuzzy Mamdani Systems., Prague, Czech Republic (August 2010)
6. Kilts, S.: Advanced FPGA Design: Architecture, Implementation, and Optimization. (2007).
7. Palnitkar, S.: Verilog HDL: A Guide to Digital Design and Synthesis. Verilog HDL: A Guide to Digital Design and Synthesis (2003).
8. Inc, A.: Introduction to Altera IP Cores. Available at: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_intro_to_megafunctions.pdf
9. Inc, Altera: Cyclone IV Device Handbook. Available at: https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf
10. Rudas I.J., B.: Digital fuzzy parametric conjunctions for hardware implementation of fuzzy systems., pp.157-166 (2009).
11. Zhang X., H.: A fuzzy logic system based on Schweizer-Sklar t-norm. Science in China, Series F: Information Sciences 49(2), 175-188 (2006).
12. Rybalov A., K.: Parameterized uninorm and absorbing norm and their application for logic design. In : Electrical \& Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of, pp.1-5 (2012)
13. Kóczy, L.: A Note on Hamacher-Operators. In : Advances in Soft Computing, Intelligent Robotics and Control. Springer (2014) 159-163
14. Mayor G., T.: 7 Triangular norms on discrete settings. Logical, Algebraic, Analytic and Probabilistic Aspects of Triangular Norms, 189 (2005)
15. Batyrshin I., R.: Parametric t-norms in reconfigurable digital fuzzy systems., Berkeley, CA, USA (August 2012)
16. Batyrshin I., R.: On the monotone sum of basic t-norms in the construction of parametric families of digital conjunctors for fuzzy systems with reconfigurable logic. Knowledge-Based Systems 38, 27-36 (January 2013)
17. Cortes Antonio P., B.: FPGA implementation of fuzzy Mamdani system with parametric conjunctions generated by monotone sum of basic t-norms. Polibits 44, 53-58 (2011)
18. Hortensius P.D., M.: Parallel Random Number Generation for VLSI Systems Using Cellular Automata. IEEE Transactions on Computers 38(10), 1466-1473 (1989)
19. Mezura-Montes, E., Vel, Coello Coello, C.: A comparative study of differential evolution variants for global optimization. In : Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp.485-492 (2006)
20. Storn R., P.: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341-359 (1997)

21. Qin A.K., H.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 13(2), 398-417 (2009)
22. Cortés-Antonio P., R.-G.: Design and Implementation of Differential Evolution Algorithm on FPGA for Double-Precision Floating-Point Representation. Acta Polytechnica Hungarica 11(4), 139-153 (2014)

Conclusiones

Se propusieron diferentes metodologías para la generación de operadores de conjunción difusa paramétrico para genera al menos 135 familias de operadores paramétricos basándose en las metodologías de: a) suma monotónica de t-normas; b) suma ordinal de t-normas y; c) la extensión de la t-norma drástica, que tienen como principio común la simplicidad matemática en su implementación para que sean adecuadas a integrarse en soluciones de sistemas difusos de bajos recursos computacionales.

Se propuso una regla de aprendizaje para entrenar un sistema de inferencia difusa Sugeno primer orden con operadores de conjunción paramétricos que ajusta los parámetros del modelo ANFIS, incluyendo los parámetros de operadores de conjunción difusa propuestos. Basada en la regla de aprendizaje de ANFIS que de manera tradicional utiliza los métodos de optimización del gradiente de mayor descenso, el estimador de mínimos cuadrados, y se incorporó el algoritmo de evolución diferencial. Se mostró la mejora en la capacidad de aproximación del modelo ANFIS tradicional teniendo mejoras relativas de hasta el 80% para diferentes funciones de prueba. Sin embargo, debe hacerse notar que al agregar un mayor de libertad a la arquitectura ANFIS, y proponiendo una optimización que utiliza una metaheurística libre de gradiente, como lo es el algoritmo de evolución diferencial, el tiempo de aprendizaje de los parámetros de la red se incrementa considerablemente.

Se propusieron diseños novedosos para la implementación en FPGA de métodos de inteligencia computacional:

1. Las metodologías generadoras de familias de operadores de conjunción difusa paramétricos y un diseño generalizado que integra.
2. La arquitectura ANFIS-Sugeno de primer orden, con funciones de membresías triangulares e integrando los operadores de conjunción difusa paramétricos desarrollados en este trabajo.
3. El algoritmo de evolución diferencial con representación de punto flotante de doble precisión.

Todos estos métodos pueden integrarse como módulos en diferentes soluciones del cómputo inteligente.

Trabajos publicados

Durante el desarrollo de esta investigación se han desarrollado las siguientes publicaciones en revistas y congresos internacionales:

Revistas JCR y CONACYT

1. Cortes Antonio P., B.: FPGA implementation of fuzzy Mamdani system with parametric conjunctions generated by monotone sum of basic t-norms. Polibits 44, 53-58 (2011)

2. Batyrshin I., R.: On the monotone sum of basic t-norms in the construction of parametric families of digital conjunctors for fuzzy systems with reconfigurable logic. Knowledge-Based Systems 38, 27-36 (January 2013)

3. Cortés-Antonio P., R.-G.: Design and Implementation of Differential Evolution Algorithm on FPGA for Double-Precision Floating-Point Representation. Acta Polytechnica Hungarica 11(4), 139-153 (2014)

4. Cortés-Antonio P., B.: Hardware Design of Digital Parametric Conjunctors and t-Norms. International Journal of Fuzzy Systems 17(4), 559-576 (2015)

Congresos internacionales

5. Cortés-Antonio P., B.: FPGA implementation of fuzzy system with parametric membership functions and parametric conjunctions. Lecture Notes in Computer Science (9th Mexican International Conference on Artificial Intelligence) 6438 LNAI(PART 2 Advances in Soft Computing), 487-499 (2010).
6. Cortés-Antonio P., B.: FPGA implementation of (p, I-p) – Monotone Sum of Basic t-norm., san francisco, CA, USA (May 2011).
7. Cortés-Antonio P., B.: FPGA Implementation of (p)-monotone sum of basic t-norms., Barcelona, Spain (August 2010).
8. Cortés-Antonio P., B.: FPGA Implementation of Parametric Fuzzy Mamdani Systems., Prague, Czech Republic (August 2010).
9. Batyrshin I., R.: Parametric t-norms in reconfigurable digital fuzzy systems., Berkeley, CA, USA (August 2012).

Trabajos futuros

Los resultados presentados en el capítulo 4 de la regla de aprendizaje serán enviados para su publicación en una revista JCR.

Extender la regla de aprendizaje propuesta, a las otras metodologías generadoras de familias de operadores paramétricos de intersección difusa propuestas en el capítulo tres, y finalmente presentar una metodología que entrene al operador generalizado que implementa a las 135 familias de operadores.

Desarrollar un análisis comparativo temporal y de aproximación de la regla de aprendizaje propuesta para la arquitectura ANFIS-Sugeno de primer orden con operadores paramétricos de conjunción difusa que hagan uso de las conjunciones propuestas contra las t-normas definidas con funciones complejas que han sido presentadas en la literatura de la lógica difusa.

Desarrollar un análisis comparativo temporal y de aproximación la regla de aprendizaje propuesta para la arquitectura ANFIS-Sugeno de primer orden con operadores paramétricos de conjunción sustituyendo el algoritmo de evolución diferencial con otras metaheurísticas propuestas en la literatura de optimización de metaheurísticas.

Diseñar e implementar en FPGA la regla de aprendizaje propuesta para un para modelos difusos Sugeno de primer orden con operadores paramétricos de conjunción: a) controlado por una máquina de estados y; b) controlado por un procesador embebido.

Implementar soluciones en software y hardware de la regla de aprendizaje propuesta para la arquitectura ANFIS-Sugeno de primer orden utilizando operadores de conjunción difusa paramétrica en problemas de aplicación emergentes tales como son el Control, la Medicina, la Agricultura entre otros.