EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

"POR MI PATRIA Y POR MI BIEN"

TESIS

ALGORITMO GENÉTICO DE DESCUBRIMIENTO DE CONOCIMIENTO BASADO EN PREDICADOS DE LA LÓGICA DIFUSA ARQUIMEDIANA COMPENSATORIA

Que para obtener el Grado de
Maestro en Ciencias de la Computación

Presenta
Ing. Citlalli Morales de la Cruz
G13071091

Director de Tesis
Dra. Laura Cruz Reyes

Co-director de Tesis
Dr. Rafael Alejandro Espín Andrade

Cd. Madero, Tamaulipas Mayo 2021

Cd. Madero, Tam. 18 de mayo de 2021

OFICIO No. : U.019/21
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. CITLALLI MORALES DE LA CRUZ
No. DE CONTROL G13071091
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ALGORITMO GENÉTICO DE DESCUBRIMIENTO DE CONOCIMIENTO BASADO EN PREDICADOS DE LA LÓGICA DIFUSA ARQUIMEDIANA COMPENSATORIA"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DRA. MARÍA LUCILA MORALES RODRÍGUEZ
SECRETARIO:	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
VOCAL:	DRA. LAURA CRUZ REYES
SUPLENTE:	DR. NELSON RANGEL VALDEZ
DIRECTOR DE TESIS:	DRA. LAURA CRUZ REYES
CO-DIRECTOR DE TESIS:	DR. RAFAEL ESPÍN ANDRADE

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica.
"Por mi patria y por mi bien".

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
MACG 'mdcoa'

Declaración de Originalidad

Yo, Citlalli Morales de la Cruz, en mi calidad de autor manifiesto que este documento de tesis es producto original de mi trabajo y que no infringe derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares. Por lo tanto, la obra es de mi exclusiva autoría y soy titular de los derechos que surgen de la misma.

Asimismo, declaro que en las citas textuales que he incluido y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

En caso de presentarse cualquier reclamación o acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, asumiré toda la responsabilidad y relevo de ésta a mi director de tesis, así como al Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero y a sus respectivas autoridades.

Cd. Madero, Tamaulipas, mayo 2021.

_____

I.S.C. Citlalli Morales de la Cruz

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) y de igual manera agradezco al Instituto Tecnológico de Cd. Madero por las facilidades proporcionadas para el desarrollo de este proyecto.

Le agradezco infinitamente a mi asesora de tesis, la Dra. Laura Cruz Reyes, mi respeto y admiración por su dedicación y enseñanza de igual manera a mi co-director de tesis Dr. Rafael Alejandro Espin Andrade, así como a los integrantes del comité tutorial de esta tesis: Dr. Nelson Rangel Valdéz, Dra. Claudia Guadalupe Gómez Santillán, Dra. María Lucila Morales Rodríguez.

Mi agradecimiento más especial es para dos personas más importantes de mi vida, mis padres por su entusiasmo por la familia, amor incondicional, confianza en mí y toda su enseñanza hacia mí. En cada etapa de mi vida, sus consejos me han hecho mejor, y me enseñó a luchar por mis sueños. Siempre estaré agradecido por todo lo que me den.

A mi hermana, por que su presencia, su cariño, y sus consejos he podido salir adelante eres mi gran motivación para seguir adelante. Gracias por todo lo que hemos compartido.

A todos mis amigos, por todos los momentos que compartimos y que aún faltan, porque su amistad me ha traído muchas alegrías, que siempre han sido una motivación muy especial, gracias por todo lo que hemos vivido juntos.

Resumen

La representación y generación de conocimiento son un problema que tiene como objetivo la recolección de grandes vólumnes de datos para encontrar patrones y poder crear modelos de predicción, que en su paso se encuentra con ciertas limitaciones ya sea por la expresividad de la información, la manera en como se representan y la interpretabilidad.

Descrito lo anterior, existen métodos que se apoyan en técnicas basadas en inteligencia artificial tales como la lógica difusa, las redes neuronales y los algoritmos bioinspirados, entre otros, esto con el objetivo de obtener información valiosa y crear modelos para la toma de decisiones o la comprensión de fenómenos que suceden a nuestro alrededor.

En esta tesis se presenta una modificación de dos algoritmos genéticos desarrollados en (Llorente, 2019) que tienen como objetivo la generación de conocimiento a través de árboles de decisión combinando la programación genética, también se logra una integración de ellos con una nueva metología basada en la Lógica Difusa Arquimediana Compensatoria, la implementación de un nuevo operador genético de mutación y el concepto de exhaustivad para obtener mayor diversidad en la búsqueda de soluciones.

Abstract

The representation and generation of knowledge is a problem that aims to collect large volumes of data to find patterns and to create prediction models, which in its passage is limited by either the expressiveness of the information, the way it is used. how they are represented and interpretability.

As described above, there are methods that rely on techniques based on artificial intelligence such as fuzzy logic, neural networks and bio-inspired algorithms, among others, with the aim of obtaining valuable information and creating models for decision-making or understanding of phenomena that happen around us.

This thesis presents a modification of two genetic algorithms developed in cite llorente whose objective is the generation of knowledge through decision trees combining genetic programming, an integration of them with a new methodology based on Compensatory Archimedean Fuzzy Logic, the implementation of a new genetic mutation operator and the concept of exhaustiveness to obtain greater diversity in the search for solutions.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Dentro de las organizaciones constantemente se requiere tomar decisiones, estás deben estar soportadas con información que esté fundamentada de manera sólida y que haya sido analizada desde distintos puntos de vista para así lograr los objetivos que se quieren alcanzar.

Existen sistemas que apoyan a la toma de decisiones, entre ellos destacan los basados en modelos predictivos, los cuales se encargan de la búsqueda del conocimiento en datos provientes de grandes colecciones de datos; esto con el fin de pronosticar situaciones futuras.

Una rama importante de la informática es la Inteligencia Artificial (IA), que con la combinación de las técnicas que la engloban, proveen sistemas para el apoyo a la toma de decisiones que exploran las dimensiones del problema con la información obtenida y formulan una propuesta de modelo que oriente una gestión del conocimiento de una manera práctica.

En esta tesis se presenta una modificación de dos algoritmos genéticos bajo el nombre de Evolutionary Knowledge based on Compensatory Fuzzy Logic y el Algoritmo Evolutionary Optimization of Generalized Sigmoidal Function desarrollados en (Llorente, 2019) que tienen como objetivo la generación de conocimiento a través de árboles de decisión combinando la programación genética.

Se logra una integración de ellos con una nueva metología basada en la Lógica Difusa Arquimediana Compensatoria, la implementación de una nueva mutación constructiva y el concepto de búsqueda exhaustiva a través del uso de múltiples comodínes para obtener mayor diversidad en labúsqueda de predicados.

1.1. Antecedentes

El presente trabajo pertenece a un proyecto creciente conformado por: la Universidad Autónoma de Coahuila (UAdeC) y el Instituto Tecnológico de Ciudad Madero (ITCM).

Entre los proyectos desarrollados se encuentran los siguientes:

- Algoritmo Virtual Savant basado en lógica difusa compensatoria para el problema de empacado de objetos (Padrón-Tristan, 2020).
- Algoritmo evolutivo para descubrir conocimiento de asociación usando lógica difusa compensatoria (Llorente, 2019).

El objetivo principal es desarrollar productos de investigación para el apoyo para el proceso de toma de decisión y optimización de procesos.

El presente trabajo, busca contribuir en estás áreas a través de la modificación de los algoritmos genéticos Evolutionary Knowledge based on Compensatory Fuzzy Logic y el Algoritmo Evolutionary Optimization of Generalized Sigmoidal Function.

1.2. Problema de investigación

Para el desarrollo de este proyecto se busca generar conocimiento con base en predicados léxicos, que permitan encontrar altos grados de pertenencia bajo el concepto analizado. La formulación es la siguiente:

Dado:

Un conjunto de datos X formado por i objetos con j atributos, se busca generar conocimiento expresado en un conjunto de predicados P tal que cada predicado p es una variable de decisión.

Se busca:

$$\max C_{X_i \in X} f(p, x_i) \tag{1.1}$$

Sujeto a:

$$p > \delta; p \in \Omega \tag{1.2}$$

Donde:

p = es la variable de decisión correspondiente al predicado a evaluar.

$f(p, x_i)$ = valor de verdad del predicado *p* obtenido al evaluar el objeto *i*.

En el caso de p se debe satisfacer un mínimo valor de verdad, denotado por δ que es asignado por un tomador de decisiones (ver Fórmula 1.2). La evaluación de *p* sobre cada x_i se realiza mediante una función f de lógica de predicados basado en Lógica Difusa Arquimediana Compensatoria (LDAC) y su evaluación sobre el conjunto *X* se calcula con la conjunción *C* del valor de verdad de todos los registros de X.

Igualmente en la ecuación 1.2 se establece que p pertecene a región factible Ω, está región está delimitada por el siguiente conjunto de reglas de construcción de un predicado p, expresadas mediante la notación Backus-Naur Form (BNF), permitiendo especificar la gramática con una sintaxis de acuerdo al problema.

1. Predicado:=<Operación>|(<Operación>)

2. Operación:=<Operación_unaria>|(<Operación_unaria>)|
   <Operación_binaria>|(Operación_binaria>)|<Operación_enearia>|(Operación_enearia>)

3. Operación_unaria:= **'NOT'** <Operando> | (**'NOT'** <Operando>)

4. Operación_binaria:=<Operador_binario><Operando><Operando>|
   (<Operador_binario><Operando><Operando>)

5. Operación_enearia:=<Operador_eneario><(Operando)>|
   (<Operador_eneario><(Operando)>)

6. Operador_binario:= **'IMP'|'EQV'**

7. Operador_enerario:= **'AND'|'OR'**

8. Operando:= $'g(x_{i,j})'$|<Predicado>

Donde:

$g(x_{i,j})$ = Función de pertenencia asociada al atributo *j* del objeto *i*.

Cada **Predicado** está compuesto por una **Operación** o un conjunto de ellas; estás se componen de operaciones de tipo **Operación_unaria**, **Operación_binaria** y **Operación_enearia**, el tipo de operación que se elija define el operador lógico a usar, para el caso de las operaciones unarias se utiliza el operador **NOT** que está implícito dentro de la regla de **Operación_unaria**, por otro lado para las operaciones binarias, se establecen en **Operador_binario** dos tipos de operadores lógicos que son **'IMP'** y **'EQV'**, por último, para las operaciones enearias los operadores lógicos que se emplean están definidos en **Operador_eneario**, estos son **'AND'** y **'OR'**. La última regla determina un símbolo terminal que corresponde a la función de pertenencia asociada a un objeto o en su defecto, otro **Predicado**.

Ejemplos:

a) **'NOT'('NOT'** $g(x_{i,j})$ **)** ► $Forma \quad SI \quad aceptada$

b) **'IMP'** $g(x_{i,j})$ $g(x_{i,j})$ ► $Forma \quad SI \quad aceptada$

c) **'EQV'** $g(x_{i,j})$ $g(x_{i,j})$ ► $Forma \quad SI \quad aceptada$

d) **'IMP'** (**'AND'** $g(x_{i,j})$ $g(x_{i,j})$ $g(x_{i,j})$) $g(x_{i,j})$► $Forma \quad SI \quad aceptada$

e) **'EQV'** $g(x_{i,j})$ ► $Forma \quad NO \quad aceptada$

f) **'AND'** $g(x_{i,j})$► $Forma \quad NO \quad aceptada$

g) **'OR'** $g(x_{i,j})$► $Forma \quad NO \quad aceptada$

En los ejemplos los incidos a) hasta el inciso d) las expresiones generadas son aceptadas porque cumplen con las reglas gramáticales, en el caso de las últimas no son aceptadas, por ejemplo en el inciso e) la regla define que el operador **'EQV'** debe ser expresada por un conjunto de Operandos y no solo de uno como es el caso, igualmente para los incisos f) y g).

1.3. Justificación

Dos propuestas basadas en LDC han sido incorporadas en herramientas de apoyo a la decisión: la herramienta Fuzzy tree studio (FTS) y Universe (Meschino, Nabte, Gesualdo, Monjeau, y Passoni, 2014). El objetivo de estas herramientas es convertir un problema en un predicado lógico difuso compensatorio. A Diferencia de FTS, Universe (Espín, González, y Caballero, 2014) aplica un algoritmo genético para optimizar la selección del predicado final. En ninguno de estos trabajos se presenta un análisis comparativo sobre el resultado obtenido. Para abordar esta limitante, (Llorente, 2019) realiza en su tesis de maestría el primer esfuerzo algorítmico documentado con formalidad y con soporte experimental; El algoritmo de Llorente realiza las tareas de evaluación y descubrimiento de predicados mediante dos algoritmos evolutivos: El primer algoritmo está basado en árboles de decisión para navegar entre predicados de LDC, mientras que el segundo algoritmo busca la función de membresía que mejor caracterice a los datos.

Se trabaja con algoritmos genéticos para consolidar el anterior con nuevas características, una nueva mutación, mejor expresividad a través de la implementación de múltiples comodines, y agregando una nueva lógica bajo el nombre Lógica Difusa Compensatoria Arquimediana.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar estrategias para los algoritmos genéticos EK-CFL y AG-GSF, que incrementen la eficiencia y eficacia del proceso de descubrimiento de conocimiento mediante el uso de Lógica Difusa Arquimediana Compensatoria y Programación genética.

1.4.2. Objetivos específicos

- Integrar los algoritmos genéticos EK-CFL y AG-GSF para contar con un algoritmo de descubrimiento de conocimiento de referencia.
- Incrementar el poder para modelar la toma de decisiones mediante el uso de la LDAC, en lugar de la LDC, en el algoritmo de referencia.
- Incrementar la eficacia y eficiencia del algoritmo de referencia, en términos de veracidad de los predicados y tiempo de procesamiento respectivamente, usando estrategias de programación genética que permitan balancear la intensificación y diversificación de la búsqueda
- Incrementar la exhaustividad de la búsqueda para encontrar predicados limitados por la definición actual de predicado simbólico.

1.5. Alcances y limitaciones

Limitaciones:

1. Algoritmo genético es de tipo programación genética.
2. Las dimensiones de las BD analizadas no son de gran escala.

Alcances:

1. Encontrar predicados con LDAC con altos valores de verdad de la proposición universal para una BD dada.

2. El algoritmo de referencia para evaluación es el propuesto en (Llorente, 2019).

1.6. Organización del documento

A continuación se presenta la organización del documento explicando brevemente cada uno de los capítulos que lo componen.

Capítulo 1
Dentro del capítulo se realiza una introducción global y se plantean los objetivos, la justificación, los alcances y las limitaciones para este proyecto.

Capítulo 2
Este capítulo contiene el marco teórico, en donde se revisan los conceptos más relevantes para el soporte del desarrollo de la tesis, tales como los elementos que conforman un predicado, las técnicas de inteligencia artificial para el problema de generación de conocimineto, etc.

Capítulo 3
Dentro de este capítulo se revisa el estado del arte y se analizan los trabajos que están relacionados con este proyecto.

Capítulo 4
Para este capítulo se describe la metodología propuesta llevada a cabo para solucionar el problema de investigación.

Capítulo 5
Por otro lado en este capítulo se exponen los resultados de la evaluación de la metodología y comentarios acerca de esta

Capítulo 6
Finalmente en este capítulo se muestran las conclusiones del trabajo, las aportaciones generadas y los trabajos a futuro.

Capítulo 2

Marco Teórico

Dentro de esta sección se describen los conceptos elementales entre ellos destacan las etapas de la representación del conocimiento, los métodos de cómo abordar este problema tales como la lógica difusa compensatoria (LDC) y lógica difusa arquimediana compensatoria (LDAC) mediante programación genética.

Dados estos conceptos se busca que se comprendan para facilitar la explicación de la metodología de solución que se explica en secciones posteriores a este capítulo del problema presentado en este proyecto.

2.1. Descubrimiento de conocimiento en bases de datos

El descubrimiento de conocimiento en bases de datos (KDD, por sus siglas en inglés) tiene como objetivo la extracción de información de grandes colecciones de datos (Ho, Cheung, y Liu, 2005) en donde las tareas más comunes son la inducción de reglas, encontrar patrones, desarrollar modelos predictivos, clasificación y agrupamiento, etc.

Las áreas de conocimiento involucradas son la inteligencia artificial, el aprendizaje autómatico, estadística, los sistemas de gestión de bases de datos, sistemas para el apoyo en la toma de decisiones, técnicas de visualización de datos, etc., entre ellas, las más frecuentes son (Ho et al., 2005):

- Descriptivas: Se centra en la búsqueda de la caracterización o discriminación de un conjunto de datos. Las técnicas más conocidas son: Agrupamiento o Clustering, Reglas de Asociación, Análisis de Patrones Secuenciales, Análisis de Componentes Principales, Detección de Desviación.
- Predictivas: De acuerdo a una hipótesis se clasifican a nuevos individuos. Los algoritmos principales son: Regresión y Clasificación (Árboles de Decisión, Clasificación Bayesiana, Redes Neuronales, Algoritmos Genéticos, Conjuntos y Lógica Difusa).

Según (Asencios, 2004) el KDD es un proceso iterativo que busca modelos, patrones válidos y útiles, esto con el fin de incorporar los conocimientos y poder realizar una toma de decisiones de acuerdo a la información obtenida, a continuación, en la Figura 2.1 se muestra el proceso de descubrimiento de conocimiento.

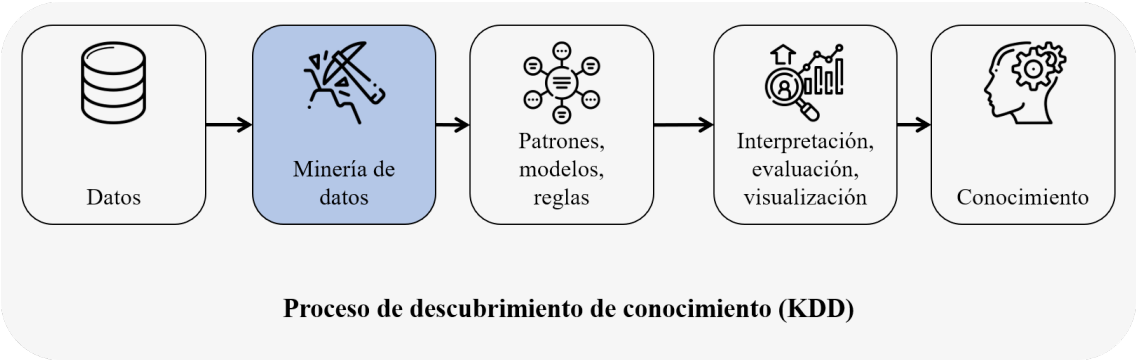

Figura 2.1: Proceso de descubrimiento del conocimiento (KDD)

Una de las etapas a resaltar es la minería de datos (Asencios, 2004) describe que es una etapa dentro del proceso completo del descubrimiento del conocimiento, este intenta obtener patrones o modelos a partir de los datos recopilados. Los algoritmos de en la minería de datos suelen tener tres componentes:

- El modelo, que contiene parámetros que han de fijarse a partir de los datos de entrada.
- El criterio de preferencia, que sirve para comparar modelos alternativos.
- El algoritmo de búsqueda, que viene a ser como cualquier otro programa de inteligencia artificial (IA).

Los resultados que busca obtener la minería de datos son los siguientes:

- Clasificación: Se trata de obtener un modelo que permita asignar un caso de clase desconocida a una clase concreta (seleccionada de un conjunto redefinido de clases).
- Regresión: Se persigue la obtención de un modelo que permita predecir el valor numérico de alguna variable (modelos de regresión logística).
- Agrupamiento (clustering): Hace corresponder cada caso a una clase, en donde las clases se obtienen directamente de los datos de entrada utilizando medidas de similaridad. Es decir, agrupan a los datos bajo diferentes métodos y criterios. Las técnicas más usadas son las clásicas (distancia mínima) y las redes neuronales.
- Resumen: Se obtienen representaciones compactas para subconjuntos de los datos de entrada (análisis interactivo de datos, generación automática de informes, visualización de datos).
- Modelado de Dependencias: Se obtienen descripciones de dependencias existentes entre variables. El análisis de relaciones (por ejemplo las reglas de asociación), en el que se determinan relaciones existentes entre elementos de una base de datos, podría considerarse un caso particular de modelado de dependencias.
- Análisis de Secuencias: Se intenta modelar la evolución temporal de alguna variable, con fines descriptivos o predictivos (redes neuronales multicapas)

2.2. Representación del conocimiento

Uno de los factores imprescindibles para solucionar el proceso de toma de decisiones es la información que se conoce y se tiene acceso; a veces es necesario hacer uso de sistemas de apoyo a la toma de decisiones que a través de técnicas de análisis, tratamiento y manipulación de datos tienen la capacidad de inducir o deducir nuevos conocimientos y buscar la manera de que sean entendibles para el razonamiento humano.

La representación del conocimiento es un concepto sumamente importante en la Inteligencia Artificial (IA); como se mencionó antes, los sistemas toman datos para procesarlos, la representación del conocimiento se centra en la manera de cómo se codifica el conocimiento humano, (Duran y Conesa, 2017) lo definen como la implementación de mecanismos para su representación por medio de símbolos y facilidad al tratamiento de los datos para que los sistemas tengan la posibilidad de razonar sobre ciertos dominios.

Las propiedades para representar el conocimiento de manera apropiada son las siguientes:

- **Representación apropiada:** Debe de ser capaz de ofrecer una solución apropiada de acuerdo al dominio con los datos e información mínima.
- **Inferencia apropiada:** Tiene la habilidad de extraer estructuras nuevas a partir con conocimiento nuevo basado en los datos históricos.
- **Eficiencia inferencial:** Ofrece la posibilidad de añadir técnicas para agilizar el procesamiento de cómputo.
- **Eficiencia de adquisición:** Es fácil añadir conocimiento nuevo.
- **Claridad, naturalidad y modularidad:** Puede identificar el conocimiento representado, con la capacidad de representarse lo más natural posible sin perder su eficiencia y eficacia cuando se fragmente.

2.3. Etapas de la representación del conocimiento

Para llevar a cabo la solución a un problema complejo usando de manera intensiva el conocimiento se establece el siguiente proceso:

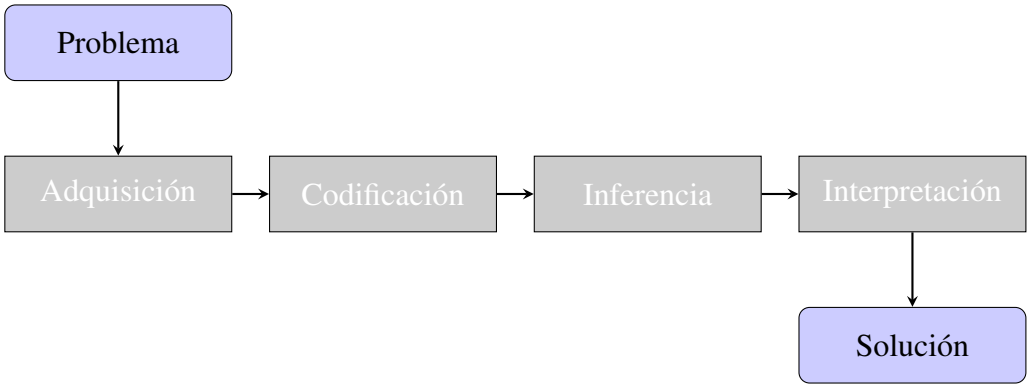

Figura 2.2: Etapas de la representación del conocimiento (Duran y Conesa, 2017)

- **Problema:** Reconocer el problema que se va a resolver y el dominio al que va aplicado.

- **Adquisición:** Establecer el dominio del conocimiento que se va a adquirir y reducir redundancia, información sobrante, dispersión para mejorar el rendimiento del acceso a la información.
- **Codificación:** Elegir que método codificará el conocimiento y hasta qué punto se puede integrar con otras fuentes de información.
- **Inferencia:** Se establece un motor de inferencia que determinará la manera de buscar y extraer conclusiones sobre el problema.
- **Interpretación:** Presenta las conclusiones del problema de manera que sean comprensibles para el usuario objetivo.
- **Solución:** Es el resultado que se obtiene de todo el proceso anterior.

El concepto de representación del conocimiento es uno de los aspectos centrales para la Inteligencia Artificial (IA) aunado con el razonamiento aproximado y el manejo de la incertidumbre. En la siguiente sección se describen técnicas de cómo puede ser abordado dicho concepto.

2.4. Soft computing

Existen diversas técnicas dentro de la computación que facilitan no solo la representación del conocimiento y su razonamiento, sino también su adquisición y tratamiento, ya que modelarlo de manera adecuada proporcionan la clave para proveer soluciones al usuario sobre el problema que se está tratando.

Un inconveniente es que en el mundo real por lo general los datos que se extraen se encuentran matizados, es decir, no poseen valores binarios y eso causa conflicto para poder trabajar con ellos, para ello existe una rama dentro de la IA denominada Soft Computing (SC) o computación blanda que ofrece técnicas para poder trabajar con propiedades como la imprecisión, razonamiento aproximado y la incertidumbre.

El modelo que sigue SC es la mente humana ya que tiene la capacidad de manejar datos parciales, imprecisión y realizar conjeturas; SC se divide en dos categorías como se muestra en la Figura 2.3, el primero denominado Razonamiento aproximado que se basa en fórmulas matemáticas y probabilidad y el segundo Aproximación y búsqueda aleatoria que se enfoca en modelos de computación bioinspirados para la optimización de procesos.

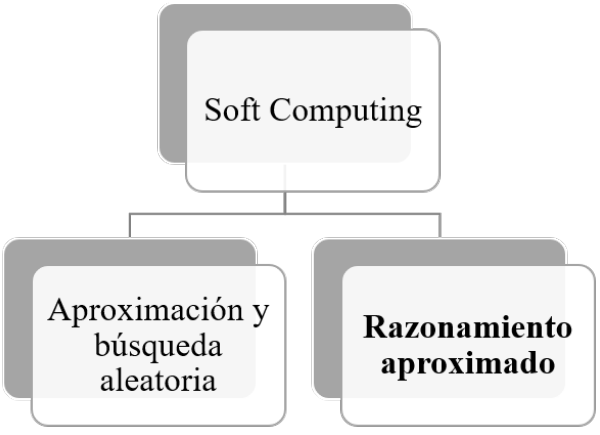

Figura 2.3: Tipos de problemas de Soft Computing (Das et al., 2013)

De los dos tipos de problemas que se pueden abordar con SC descritas anteriormente, existen técnicas que se pueden aplicar para resolverlos, esta taxonomía se muestra en la Figura 2.4. SC es una asociación de diferentes metodologías que involucran los conceptos de verdad parcial, aproximación, incertidumbre, etc., estos métodos poseen la capacidad de aprender de la experiencia, por lo tanto, hace que las técnicas sean únicas a comparación de los métodos analíticos precisos (Sharma y Chandra, 2019).

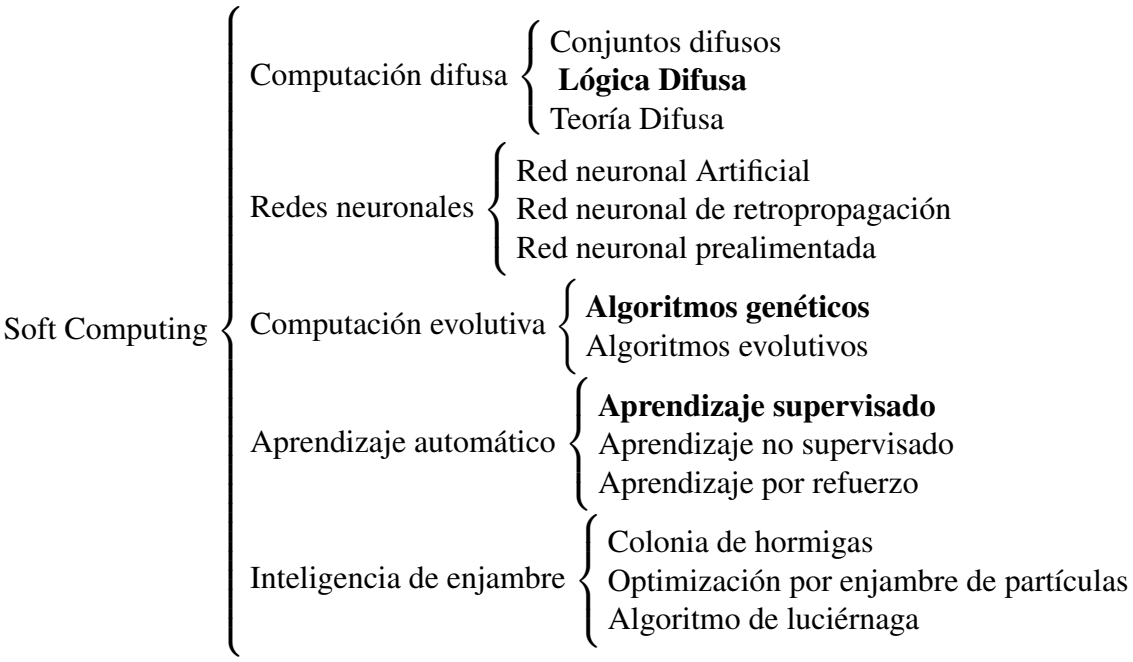

Figura 2.4: Clasificación de técnicas de resolución de problemas de Soft Computing (Sharma y Chandra, 2019)

2.5. Lógica Difusa

La razón por la que la lógica es relevante para la representación y el razonamiento del conocimiento es simplemente que, al menos según un punto de vista, la lógica es el estudio de las relaciones de implicación: lenguajes, condiciones de verdad y reglas de inferencia (Brachman y Levesque, 2004).

La lógica difusa (LD) es un tipo de lógica multivaluada que aparece en 1965 por Lofti A. Zadeh caracterizada por su flexibilidad y la manera en que representa la información del pensamiento humano a través de las variables involucradas en el conjunto de datos usando etiquetas lingüísticas (Zadeh, 1988), y a su vez es permisible para trabajar con información imprecisa creando así un conjunto de proposiciones modelando el razonamiento humano para realizar acciones.

La principal ventaja de un enfoque de representación del conocimiento preferencial basado en lógica difusa sería justamente la oportunidad de utilizar el lenguaje como elemento de comunicación y modelación en el análisis de la decisión, creando un modelo explícito del conocimiento preferencial; posteriormente utilizar la capacidad de inferencia para proponer decisiones que reflejen mejor la preferencia del tomador de decisiones.

Las proposiciones de LD son afirmaciones medidas por un valor de verdad, que se relaciona con el uso de límites definidos bajo el intervalo [0,1]. Estás proposiciones expresan ideas bajo la

percepción de cada individuo (Ross, 2010) y son representadas por declaraciones lingüísticas que se acercan al lenguaje natural que tiende a ser vago e impreciso.

Las proposiciones difusas son asignadas a conjuntos difusos (en donde $\sim$ establece que es difuso). Dado $\underset{\sim}{P}$

$$T(\underset{\sim}{P}) = \mu_{\underset{\sim}{A}}(x), \quad \text{where} \quad 0 \leq \mu_{\underset{\sim}{A}} \leq 1 \tag{2.1}$$

En la ecuación 2.1 se indica el grado de verdad de la proposición $\underset{\sim}{P} : x \in \underset{\sim}{A}$ que equivale al grado de pertenencia de x dentro de un conjunto difuso $\underset{\sim}{A}$.

El modelado de datos a través de manera lingüística es un desafío; en el caso de la LD usar operadores como un sistema lógico sirve para conectar variables y crear enunciados expresados en lenguaje natural. Estos operadores son la conjunción, disyunción, negación y orden estricto (Espín, Bello, Cobo, Marx, y Racet, 2014).

(Espín, González, y Caballero, 2014) establecen que la combinación de la lógica con la toma de decisiones bajo un marco lingüístico, debe ser soportada por un sistema de axiomas que permitan lo siguiente:

- Modelar un tipo de racionalidad compatible con los efectos de razonamiento aproximado y descriptivo (conductual) del veto.
- Manejar la información lingüística de las preferencias del tomador de decisiones.
- Agregar la información de preferencias, incluyendo el manejo de los compromisos y de las situaciones simultáneamente ventajosas y desventajosas del tomador de decisiones (de manera no asociativa).

Este sistema axiomático se denomina Lógica Difusa Compensatoria (LDC) y se describirá en la siguiente sección.

2.6. Lógica Difusa Compensatoria

La lógica difusa compensatoria propone la combinación de lógicas multivalentes soportado por un conjunto de axiomas descritos basados en preferencias expresadas a través de frases complejas que incluyan más de una etiqueta lingüística (Espín, González, y Caballero, 2014). La LDC está conformado es un sistema conformado por un cuarteto de: un operador de conjunción, un operador de disyunción, un operador de negación y un operador de orden difusa estricto que satisfacen axiomas pertenecientes a la lógica y la teoría de la decisión (Espín, Bello, et al., 2014).

Una LDC está formada por un cuarteto de operadores continuos (c, d, o, n); los operadores reciben los nombres de conjunción, disyunción, orden estricto difuso y negación que se definen como sigue: c: $[0,1]^n \to [0,1], d : [0,1]^n \to [0,1], o : [0,1]^n \times [0,1]^n \to [0,1]$ y $n : [0,1] \to [0,1]$, siendo $x = (x_1, x_2, ..., x_n), y = (y_1, y_2, ..., y_n)$ y $z = (z_1, z_2, ..., z_n)$ elementos cualquiera del producto cartesiano $[0,1]^n$, los operadores que forman la cuarteta cumplen los siguientes axiomas:

1. Axioma de compensación:
   $min\{x_1, x_2, ..., x_n\} \leq c\{x_1, x_2, ..., x_n\} \leq max\{x_1, x_2, ..., x_n\}$

2. Axioma de conmutatividad o simetría:
   $c\{x_1, x_2, ..., x_i, x_j, ...x_n\} = c\{x_1, x_2, ..., x_j, x_i, ...x_n\}$

3. Axioma de crecimiento estricto: Si $x_1 = y_1, x_2 = y_2, ..., x_{i-1} = y_{i-1}, x_{i+1} = y_{i+1}, ..., x_n = y_n$ son diferentes de 0 y $x_i > y_i$ entonces $c(x_1, x_2, ..., x_n) > c(y_1, y_2, ..., y_n)$

4. Axioma de veto: Si $x_i = 0$ para un i entonces $c(x) = 0$

5. Axioma de reciprocidad difusa $o(x, y) = n[o(y, x)]$

6. Axioma de transitividad difusa: Si $o(x, y) \geq 0.5$ y Si $o(x, y) \geq 0.5$ entonces $o(x, y) \geq max(o(x, y), o(y, x)$

7. Leyes de Morgan:
   $n(c(x_1, x_2, ..., x_n)) = d(n(x_1), n(x_2), ..., n(x_n))$
   $n(d(x_1, x_2, ..., x_n)) = c(n(x_1), n(x_2), ..., n(x_n))$

El axioma de compensación es el más representativo, ya que este resalta la diferencia de la forma general de la LD (Bouchet, Pastore, Brun, y Ballarin, 2010), la propiedad que refleja se emplea generalmente en la literatura sobre operadores difusos para definir el concepto de operador compensatorio (Vi, Detyniecki, Yager, Prade, y Grabisch, 2001).

El axioma de conmutatividad o simetría es natural que el resultado de la conjunción sea independiente del orden en el que se tomen los predicados básicos.

El axioma de crecimiento estricto otorga la sensibilidad que logra que cualquier cambio en los valores de los predicados básicos modifique el valor de verdad del predicado compuesto, siempre y cuando ninguno de los predicados básicos tenga valor de cero.

El axioma de veto otorga a cualquier predicado básico de una conjunción la capacidad de vetar, es decir, impide cualquier forma de compensación cuando su valor es igual a cero.

2.7. Lógica Difusa Arquimediana Compensatoria

La Lógica Difusa Arquimediana Compensatoria (LDAC) permite la modelación simultánea del pensamiento deductivo y el de toma de decisiones. Esta lógica es la construcción de la unificación de dos tipos diferentes sistemas lógicos difusos llamadas Lógicas Compensatoria y Lógica Arquimediana, las cuales son compatibles (González-Caballero et al., 2015).

Dos formas de definir los sistemas lógicos son el enfoque T-norma (donde $[a_i, b_i]$ con $i = 1, ..., n$ son conjuntos disjuntos del intervalo $[0, 1]$, y para cada i, T_j una t-norma) y T–conorma (donde$[a_i, b_i]$ con $i = 1, ..., n$ son conjuntos disjuntos del intervalo $[0, 1]$, y para cada i, S_j una t-conorma) y la lógica compensatoria difusa (LDC).

2.7.1. Propiedades de t-norma

Una t-norma es una función $T : [0, 1]^2 \rightarrow [0, 1]$, que satisface a los siguientes axiomas:

$T(a,b) = T(b,a)(T1)$	**Conmutatividad**
$T(a,b) \leq T(c,d)$	**Monotonicidad**
Si	
$a \leq c$	
$b \leq d(T2)$	
$T(a,T(b,c)) = T(T(a,,b),c)(T3)$	**Asociatividad**
$T(a,1) = a(T4)$	**Uno como identidad**

2.7.2. Propiedades de t-conorma

Una t-conorma es una función $S : [0,1]^2 \to [0,1]$, que satisface a los siguientes axiomas:

$S(a,b) = S(b,a)ST1)$	**Conmutatividad**
$S(a,b) \leq S(c,d)$	**Monotonicidad**
Si	
$a \leq c$	
$b \leq d(S2)$	
$S(a,S(b,c)) = S(S(a,,b),c)(S3)$	**Asociatividad**
$S(a,1) = a(S4)$	**Uno como identidad**

El orden de los predicados en función de su veracidad se mantiene entre estos dos sistemas difusos, obtenida a través de una función f. Con LDAC Se tiene una lógica de refutación (veracidad de no refutación), las propiedades arquimediana y de idempotencia permiten esta característica (ver Anexo A). Con LDC se tiene una lógica de afirmación dado por los cuantificadores universal y existencial.

El cuantificador universal arquimediano y su cuantificador existencial correspondiente representa la tendencia a no refutar una toma de decisiones representada por un predicado, respectivamente. Equivalente para la LDC, el par formado por un cuantificador universal y un cuantificador existencial basados en LDC representan la afirmación. A continuación, se describen ambos cuantificadores (González-Caballero et al., 2015).

Definición 1:
Sea $X = x_1, ..., x_n \subset [0,1]$ el cuantificador universal para el conjunto finito X y la t-norma arquimediana continua, es definida por $\forall_T x_i \in X = C_T x_1, ..., x_n$, denominado cuantificador universal de no refutación.

Definición 2:
Sea $X = x_1, ..., x_n \subset [0,1]$, el cuantificador universal para el conjunto finito X y el operador de conjunción de un sistema LDC basado en las ecuaciones cuasi aritméticas, es definida por $\forall_C x_i \in X = C_C x_1, ..., x_n$, este cuantificador se denomina como cuantificador universal de afirmación.

Definición 3:
Sea$X = x_1, ..., x_n \subset [0,1]$, el cuantificador existencial para el conjunto finito X y la t-norma T Arquimediana continua, es definida por $\exists_T x_i \in X = D_T x_1, ..., x_n$, que se define como cuantificador existencial de no refutación.

Definición 4:
Sea $X = x_1, ..., x_n \subset [0,1]$, el cuantificador existencial para el conjunto finito X y la, es definida

por $\exists_C x_i \in X = D_C x_1, ..., x_n$, llamado cuantificador existencial de afirmación.

Se tiene en cuenta que las fórmulas para ambos operadores cuantificadores universales $\forall_T$ y $\forall_C$, son básicamente los mismos si son generados por la misma función, excepto por el índice $n = card(X)$, el cual es un número natural. Entre más grande es n, más grande es la evidencia para la persona sobre la falsedad del valor de verdad del predicado universal, y también es el modificador lingüístico necesario para compensación. Por lo tanto, esta relación es la base de la unificación de la teoría de la t-norma y la teoría LDC (González-Caballero et al., 2015).

2.8. Definición de predicado difuso

El concepto de predicado se asocia a la construcción de reglas o proposiciones que se evalúan de acuerdo con las lógicas existentes. Los predicados están conformados mediante etiquetas lingüísticas y se evalúan por medio de operaciones lógicas, sus resultados se obtienen aplicando operadores lógicos y existen diferentes tipos de lógica para realizar dichos cálculos. Los elementos de un predicado son los estados lingüísticos, funciones de pertenencia y operadores lógicos. (Rodríguez-Fdez, Canosa, Mucientes, y Bugarín, 2015).

2.9. Estructura de predicados

La construcción de predicados se basa a partir de una expresión verbal conformada por variables lingüísticas que se traducen al lenguaje de cálculo de predicados (Cejas-Montero, 2011) estos se dividen en predicados simples y compuestos.

Un ejemplo dado por (Cejas-Montero, 2011) para la construcción de predicados compuestos sobre predicados simples se muestra a continuación:

Una tienda gestiona eficientemente el Merchandising en la Villa Panamericana, si cumple el siguiente requisito:

Su Gestión de Inventario es competitiva: Una tienda tiene una Gestión de Inventario competitiva si posee una Rotación Normal, tiene un buen Ritmo de Crecimiento de Rotación y una Cobertura de Inventario apropiada.

Predicados simple

- $R(X)$: "x es una tienda con rotación normal de inventario"
- $C(X)$: "x es una tienda con un buen ritmo de crecimiento de rotación de inventario"
- $CI(X)$: "x es una tienda con una apropiada cobertura de inventario"

Predicado compuesto

$GI(X)$: "x es una tienda con una Gestión de Inventario competitiva.

Con los valores de verdad de los predicados simples se obtiene el cálculo de los valores de verdad de los predicados compuestos para obtener el grado de verdad complejo.

2.9.1. Variables lingüísticas

Para permitir el cálculo de veracidad de una expresión de manera cuantitativa y modelar el conocimiento basado en expertos se hace uso de variables lingüísticas que asocian la información cualitativa a cuantitativa.

Una variable lingüística se caracteriza por valores sintácticos o etiquetas (Espinilla, 2008) está conformada por elementos infinitos como por ejemplo para la variable apariencia está dada por:

$T(Apariencia)$ = hermosa + bonita + adorable + guapo + atractivo + no hermosa + muy bonita + más o menos bonita + no atractivo + muy poco atractivo...

Otro ejemplo está dado por la edad que los términos para su conjunto de datos pueden ser:
$T(Edad)$ = joven + muy joven + no tan joven + mediana edad + adulto + no tan adulto + viejo + Extremadamente viejo...

Que si se representa en un plano es más fácil visualizar (Zadeh, 1975) los límites de cada término dado en el conjunto de datos como se muestra en la Figura 2.5.

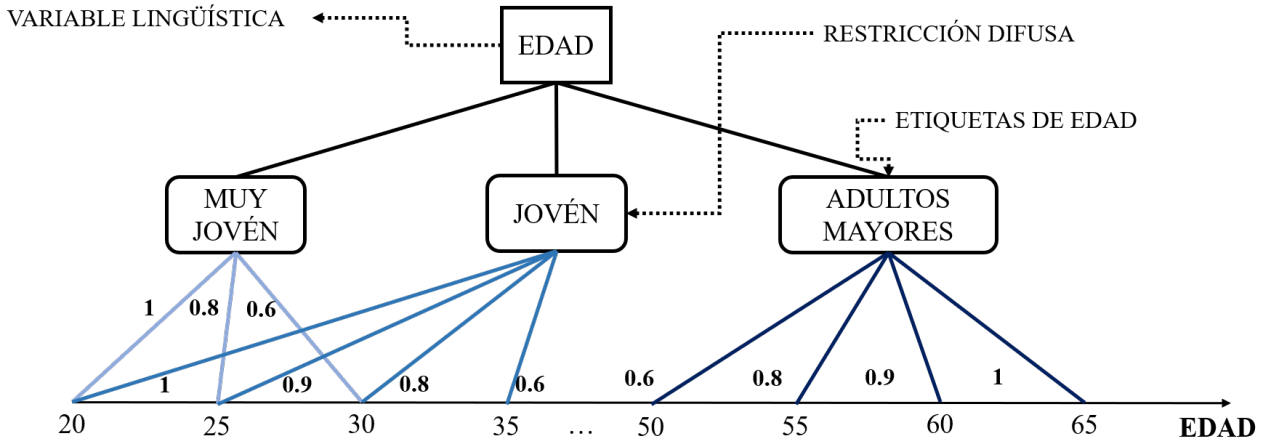

Figura 2.5: Estructura jerárquica de una variable lingüística (Zadeh, 1975)

Los estados lingüísticos son aquellas variables dentro de la lógica difusa cuyos valores son palabras o etiquetas en lugar de números; estas palabras describen elementos que son difíciles de expresar numéricamente (Padrón-Tristan, 2019). Su definición formal es dada por la quinteta $(X, T(X), U, G, M)$ donde:

- X**:** Variable linguística
- $T(X)$**:** El conjunto de etiquetas que X puede tomar
- U**:** El dominio subyacente, los valores que están representados por las etiquetas lingüísticas
- G**:** Gramática para generar etiquetas lingüísticas
- M**:** Reglas semánticas que asocian cada etiqueta de $T(X)$

Los estados lingüísticos ayudan en:

- Granulación, que comprime la información ya que una etiqueta incluye muchos valores posibles,

- Caracterización de situaciones poco definidas y / o complejas,
- Traducción de variables lingüísticas a valores numéricos automáticamente,
- Ampliación de herramientas, donde las herramientas existentes se pueden utilizar para manejar otras variables lingüísticas.

Las variables lingüísticas en LDC tienen su fundamento a partir de los términos lingüísticos que se obtienen, La LDC utiliza la escala de LD, que varía entre 0 y 1 para medir el grado de verdad o falsedad de las proposiciones expresadas en predicados (ver Tabla 2.1).

Tabla 2.1: Valores de verdad (Espín et al., 2012)

Valor de verdad	Categoría
0	Falso
0.1	Casi falso
0.2	Bastante falso
0.3	Algo falso
0.4	Mas falso que verdadero
0.5	Tan verdadero como falso
0.6	Mas verdadero que falso
0.7	Algo verdadero
0.8	Bastante verdadero
0.9	Casi verdadero
1	Verdadero

Los estados lingüísticos están definidos a través de las funciones de pertenencia, las cuáles se explican a continuación.

2.9.2. Funciones de pertenencia

Una función de pertenencia $fA(x)$ en un espacio X caracteriza un conjunto difuso A, donde cada valor x está asociado con un número real dentro del intervalo $[0, 1]$, este número indica cuánto pertenece x al establecer A. Las funciones de pertenencia están definidas por los parámetros de uso, que se ajustan de acuerdo con el entorno del problema a resolver y/o ayuda de un usuario experto (Padrón-Tristan, 2019).

Las funciones de pertenencia se pueden clasificar en funciones discretas y continuas. Las funciones discretas son aquellas que tienen un conjunto finito de soluciones. Por otro lado, las funciones continuas tienen una infinita cantidad de soluciones, estás se encuentran descritas en el Anexo B.

Propiedades de las funciones de pertenencia

Las características más destacables para definir las propiedades de las funciones de pertenencia son las siguientes (Dombi, 1990):

1. Todas las funciones de membresía son continuas.
2. Las funciones de membresía son representadas en intervalos $[a, b]$ a $[0, 1]$, $\mu[a, b] \rightarrow [0, 1]$.

3. Las funciones de pertenencia son (a) monótonamente crecientes, o (b) monótonamente decrecientes, o (c) podrían dividirse en una parte monótona creciente o decreciente.

4. las funciones de pertenencia monótonas en todo el intervalo son (a) funciones convexas o (b) funciones cóncavas, o (c) existe un punto c en el intervalo $[a, b]$ tal que $[a, c]$ es convexo y $[c, b]$ es cóncava (llamadas funciones en forma de S).

5. Las funciones que aumentan monótonamente tienen la propiedad $u(a) = 0$, $u(b) = 1$, mientras que las funciones monótonamente decrecientes tienen la propiedad $u(a) = 1$, $u(b) = 0$.

6. Es muy importante es la forma lineal o linealización de la membresía función.

2.9.3. Operadores lógicos de Lógica Difusa Compensatoria y Lógica Difusa Arquimediana Compensatoria

Una de las bases de las lógicas difusas es el uso de operadores que conectan a las oraciones o declaraciones, los conectivos permiten formar oraciones compuestas, es decir, conformadas por varias proposiciones

Los conjuntos y operadores difusos son sujetos y verbos de la lógica difusa. Al formular las reglas se usan enunciados condicionales de tipo "Si ... entonces" que competen a la lógica difusa, la parte "Si" es el antecedente de la regla y la "Entonces", el consecuente o la conclusión. (Coyaso y Vermonden, 2015).

Cada regla define una superficie de implicación. Para cada posible valor del antecedente se obtiene el conjunto difuso, realizando la implicación y la superposición de todos estos conjuntos difusos para formar la superficie de implicación de esa regla.

A continuación, se muestra la Tabla de operadores lógicos (conectivas lógicas) que se usan para conectar expresiones.

Tabla 2.2: Operadores lógicos (conectivas lógicas)

Operador	**Representación**	**Aridad**
Conjunción	AND ($\wedge$)	2 o más
Disyunción	OR ($\vee$)	2 o mas
Negación	NOT ($\neg$)	1 (unario)
Implicación	IMP ($\rightarrow$)	2 (binario)
Equivalencia	EQV ($\leftrightarrow$)	2 (binario)

Lógica Difusa Compensatoria basada en la Media Geométrica

La definición original de la LDC, denominada Lógica Difusa Compensatoria basada en la Media Geométrica (LDCMG), se basa en la utilización de la media geométrica y su dual en la definición de los operadores conjunción y disyunción. Las ecuaciones 2.2 a 2.5 que se presentan a continuación definen dicha lógica.

Es una cuarteta (c_1, d_1, o_1, n_1) que se define por medio de la media geométrica y su dual como operadores conjuntivo y disyuntivo (Espín, Bello, et al., 2014) junto con orden y la negación.

$$c_1 = (x_1, x_2, ..., x_n) = (x_1, x_2, ..., x_n)^{\frac{1}{n}} \tag{2.2}$$

$$d_1 = (x_1, x_2, ..., x_i, x_j, ...x_n) = 1 - [(1 - x_1), (1 - x_2), ..., (1 - x_n)]^{\frac{1}{n}} \tag{2.3}$$

$$o_1 = (x, y) = 0.5[c_1(x), c_1(y)] + 0.5 \tag{2.4}$$

$$N_1 = (x, y) = 1 - x_i \tag{2.5}$$

Operadores (expresados de manera general en término de dominio-contradominio) de la Lógica Difusa Arquimediana Compensatoria.

Se dice que $L = (c_c, c_T, d_c, d_T, o, n)$ es una Lógica Difusa Compensatoria Arquimediana si cuenta con los siguientes operadores difusos:

1. $c_r : [0,1]^2 \to [0,1]$ es una t-norma Arquimediana que generada por una función f que satisface a al ecuación 1.
2. $d_r : [0,1]^2 \to [0,1]$ satisface $d_t(x_1, x_2) = 1 - c_t(1 - x_1, 1 - x_2)$ para cada $x = (x_1, x_2) \in [0,1]^2$
3. $c_c : [0,1]^n \to [0,1]$ satisface la ecuación 3 para f usada en el punto 1.
4. $d_c : [0,1]^n \to [0,1]$, donde para cada $x = (x_1, x_2, ..., x_n) \in [0,1]^n$, $d_c(x_1, x_2, ..., x_n) = 1 - c_c(1 - x_1, 1 - x_2, ..., 1 - x_n)$ y $c_c(x_1, x_2, ..., x_n) = 1 - d_c(1 - x_1, 1 - x_2, ..., 1 - x_n)$
5. $o : [0,1]^2 \to [0,1]$ es un orden difuso.
6. $n : [0,1] \to [0,1]$ es el operador de negación con $n(x) = 1 - x$

Ejemplo
Una lógica difusa arquimediana compensatoria se obtiene de la lógica probabilística, y LDCMG en donde la definición de los operadores se resume en la Tabla 2.3.

Tabla 2.3: Operadores del sistema de lógica probabilística y la LDCMG (González-Caballero et al., 2015)

Operador	Sistema de lógica probabilística	Sistema GMBCL
Conjunción	$c(x_1, x_2) = x_1 * x_2$	$c(x) = \sqrt[n]{\prod_{i=1}^{n} x_1}$
Disyunción (dual de la conjunción)	$d(x_1, x_2) = x_1 + x_2 - x_1 * x_2$	$d(x) = 1 - \sqrt[n]{\prod_i = 1^n (1 - x_i)}$
Negación	$1 - x$	1-x
Implicación $(d(n(x_1), x_2))$	Implicación de Reinchenbach $i(x_1, x_2) = 1 - x_2 + x_1 x_2$	Implicación Natural $i(x_1, x_2) = 1 - \sqrt{x_1 + x_1 x_2}$
Equivalencia $c(i(x_1, x_2), i(x_2, x_1))$	$i(x_1, x_2) * i(x_2, x_1)$	$\sqrt{i(x_1, x_2) * i(x_2, x_1)}$

Los operadores de conjunción y disyunción correspondientes a ambos sistemas lógicos pueden ser generados por el mismo generadores aditivos, $f(u) = -ln(1 - u)$ para conjunción y

$f^{(-1)}(u) = -e^{-u}$.

Por lo tanto, este ejemplo de LDAC es generado por el par de funciones $-ln(u)$ y $ln(1-u)$, ambas satisfaciendo la proposición: Toda lógica continua arquimediana y su lógica difusa compensatoria correspondiente son compatible para cada fórmula proposicional formada recursivamente por átomos fórmulas, conjunción, disyunción, negación, implicación natural y operadores de equivalencia (González-Caballero et al., 2015) y es compatible calcular con lógica probabilística o LDCMG, en el sentido para mantener el orden de los predicados, que son formado por los operadores de la Tabla 2.3.

2.10. Algoritmos evolutivos

La evolución natural de las especies se describe como un proceso de aprendizaje para que las especies aumenten su aptitud, logrando de esta manera una mejor supervivencia dentro del ambiente; esta teoría aplicada a los algoritmos evolutivos ayuda a la solución de tareas de optimización o aprendizaje cumpliendo principalmente estás características (Yu y Gen, 2010):

- Población base: Los algoritmos evolutivos mantienen un grupo de soluciones denominado población.
- Orientado a la aptitud: Cada elemento de la población se llama individuo que tiene una representación genética denominada código y se evalúa su rendimiento a través de la aptitud.
- Impulso por variaciones: Los individuos se someten a una serie de operaciones de variación para imitar la genética dentro de un espacio de soluciones.

2.10.1. Algoritmos genéticos

Los algoritmos genéticos (AG) son métodos adaptativos, generalmente usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio supervivencia del más apto.

En (Golberg, 1989) define a los algoritmos genéticos como algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural combinando la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, de manera aleatoria, para representar así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas .

Para alcanzar la solución a un problema se parte de un conjunto inicial de individuos, llamado población inicial, generado de manera aleatoria. Cada uno de estos individuos representa una posible solución al problema. Estos individuos evolucionarán tomando como base los esquemas propuestos por Darwin (Darwin, 2016) sobre la selección natural, y se adaptarán en mayor medida tras el paso de cada generación a la solución requerida.

Operador de Selección
Puesto que se trata de imitar lo que ocurre en la naturaleza, se ha de otorgar un mayor número de oportunidades de reproducción a los individuos más aptos (Gestal, 2013). Por lo tanto, la selección de un individuo estará relacionada con su valor de ajuste. No se debe sin embargo eliminar por

completo las opciones de reproduccióon de los individuos menos aptos, pues en pocas generaciones la población se volvería homogénea.

Operador de cruza
El cruce es el operador genético principal: se necesitan dos individuos, llamados padres, y produce uno o dos nuevos individuos, llamados descendientes, combinando partes de los padres. En su forma más simple, el operador trabaja intercambiando subcadenas antes y después de un punto de cruce seleccionado al azar (Affenzeller, Winkler, Wagner, y Beham, 2009).

Operador de mutación
El tercer operador genético, la mutación, es esencialmente una modificación arbitraria que ayuda a prevenir la convergencia prematura al muestrear aleatoriamente nuevos puntos en el espacio de búsqueda. En el caso de cadenas de bits, la mutación se aplica simplemente volteando bits al azar en una cadena con una cierta probabilidad llamada tasa de mutación (Affenzeller et al., 2009).

A continuación, se describe la secuencia del AG:

1. Crear una población aleatoria de N cromosomas, cada uno con tamaño de m bits.
2. Cada cromosoma es evaluado y se le asigna una aptitud $f(x)$.
3. Se seleccionan los cromosomas con base a su aptitud.
4. Se aplican un conjunto de operadores genéticos a cada par de cromosomas con una probabilidad de cruza p_c seleccionando aleatoriamente el punto de cruce.
5. Se aplica el operador genético de mutación al nuevo cromosoma con una probabilidad p_m
6. Se agrega el nuevo individuo a la nueva población
7. Se reemplaza la población anterior con la nueva
8. Se evalúa el criterio de parada; de lo contrario se repite el paso 2.

El concepto fundamental del AG es el cromosoma, que tiene como objetivo codificar la información a través de cadenas de símbolos (por lo general binarios), las cuales pueden ser operadas por operadores genéticos, de esta manera las soluciones evolucionan hacia una solución mejorada (Ghanea-Hercock, 2003).

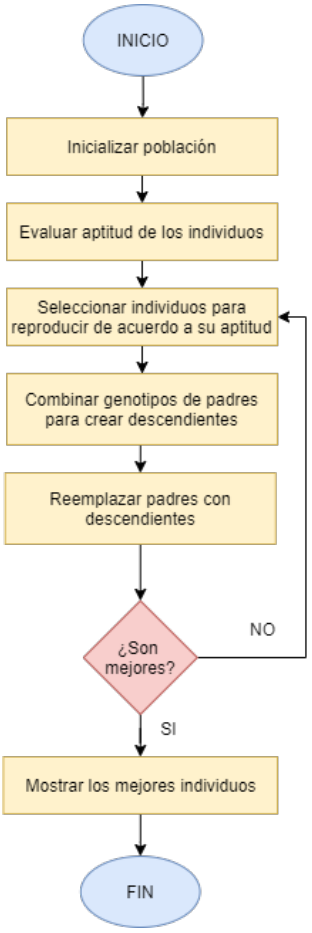

Figura 2.6: Secuencia de operaciones de un Algoritmo genético (Ghanea-Hercock, 2003)

Algoritmos genéticos para optimización de predicados

En esta sección se presenta la funcionalidad de dos algoritmos de referencia para el desarrollo de este proyecto; en donde el primero lleva el proceso de descubrimiento de conocimiento expresado en LDC y el segundo analiza los datos de cada atributo asignando una Función de Pertenencia Generalizada (FPG) para mejorar su valor.

Algoritmo Evolutionary Knowledge based on Compensatory Fuzzy Logic (Llorente, 2019)

A través del algoritmo genético que hace uso de las metodologías de programación genética llamado Evolutionary Knowledge based on Compensatory Fuzzy Logic (EK-CFL), que realiza descubrimiento de conocimiento expresado en predicados de LDC , se lleva a cabo la evolución de los predicados descubre reglas en instancias de datos usando varias metodologías de LDC, el funcionamiento general se describe en el algoritmo 1, el cual se presenta a continuación.

El algoritmo de descubrimiento de conocimiento expresado en predicados de LDC nombrado EK-CFL (Evolutionary Knowledge based on Compensatory Fuzzy Logic), a través de un algoritmo genético que hace uso de las metodologías de programación genética, se lleva a cabo la evolución de los predicados descubre reglas en instancias de datos usando varias metodologías de LDC, el funcionamiento general se describe en el algoritmo 1, el cual se presenta a continuación.

Algorithm 1: Algoritmo genético de descubrimiento de predicados KD-CFL

Entrada: Set de datos, conjunto de funciones de pertenencia
Salida : Conjunto de predicados de LDC y su valor de verdad

Generar la población inicial
$poblacion \leftarrow random\_poblacion()$
Evalúa el fitness de cada individuo de la población
$fitness\_individuo \leftarrow calcula\_fitness(individual)$
$evaluar(poblacion)$
do
    Selecciona los individuos a modificar
    $torneo\_seleccion(poblacion)$
    Cruza de individuos seleccionados
    $nuevo\_hijo \leftarrow cruza(padre1, padre2)$
    Muta a lo individuos seleccionados
    $nuevo\_hijo \leftarrow muta(padre1)$
    Reemplaza los mejores individuos
    $poblacion \leftarrow (poblacion, nuevo\_hijo)$
    Si hay funciones de pertenencia generalizadas
    if $FPG == true$ **then**
        Optimiza las FPG,s
        $EOF - GSF(individuals)$
        $evaluar(poblacion)$
while $mejoresNindividuos = falseorgeneracion < max\_generaciones$ *Mientras no se consigan los individuos con un mínimo valor de verdad seleccionado o no se llegue al máximo de generaciones*
Se regresa la lista de los mejores individuos o el mejor individuo en caso de no alcanzar el valor de verdad seleccionado
return $mejores \quad N \quad individuos \quad or \quad mejor\_individuo$

Generación de la población

Para la generación de la población del Algoritmo 1 se hacen uso de dos metodologías: árbol de crecimiento (Grow tree) y árbol completo (full tree) siempre y cuando el individuo no exceda la profundidad que se especifica por el usuario; esto debido a que ninguno de los métodos crea árboles diversos por lo que se hace una combinación llamada rampa de mitad y mitad (ver Algoritmo 2

Algorithm 2: Algoritmo generador de la población inicial (método rampa de mitad y mitad

Entrada: Lista de funciones, lista de operadores, profundidad, método de construcción
Salida : Expresión en cadena

if $profundidad = 0$ *or* $metodo = crecimiento$ *and* $rand() < \frac{|term\_set|}{|term\_set|+|func\_set|}$ **then**
    $expr \leftarrow choose\_random\_element(term\_set)$
else
    $funcion \leftarrow choose\_random\_element(func\_set)$
    for $i = 1$ **to** $(arity(funcion)$ **do**
        arg $i \leftarrow gen\_rnd\_expr(func\_set, term\_set, profundidad - 1, metodo)$
    end
end

Los datos de entrada que recibe el algoritmo son: operadores logicos $(func\_set)$ y las etiquetas lingüísticas a usar $(term\_set)$ o un límite de profundidad del árbol (profundidad), el ultimo valor requerido es el tipo de construcción a usar (método).

En la línea uno se válida que no sea el nivel máximo de crecimiento o que el método seleccionado sea de crecimiento y que el valor seleccionado aleatoriamente este dentro de los conectores lógicos. De cumplirse estas condiciones se selecciona una etiqueta lingüística. De lo contrario se selecciona un valor perteneciente a los conectores lógicos en la línea 4.

Si se ha seleccionado un conector lógico, las operaciones de selección de parámetros continua hasta que se cumple con la aridad del operador lógico esto en la línea 6; una vez terminado se guarda la cadena construida en $expr$ y se regresa la cadena.

Selección por torneo
El método de selección es mediante torneo (Golberg, 1989) en donde la idea se basa en elegir un número determinado de individuos al azar de una población (con o sin reemplazo), se selecciona al mejor individuo de este grupo para su posterior procesamiento genético y se repite tantas veces como desee (generalmente hasta que se llene el grupo de apareamiento). Los torneos suelen celebrarse entre parejas de individuos con un tamaño de $s = 2$, aunque se pueden utilizar y analizar torneos más grandes.

Cruza de un punto
La cruza se realiza mediante el intercambio de ramas de dos árboles de manera que se mantenga la corrección sintáctica, es decir, los programas descendientes son siempre programas válidos que se pueden evaluar. Primero se seleccionan dos padres, luego se escoge un subárbol aleatorio de cada padre, que luego se intercambia en la ubicación del nodo correspondiente en cada individuo descendiente (Ghanea-Hercock, 2003).

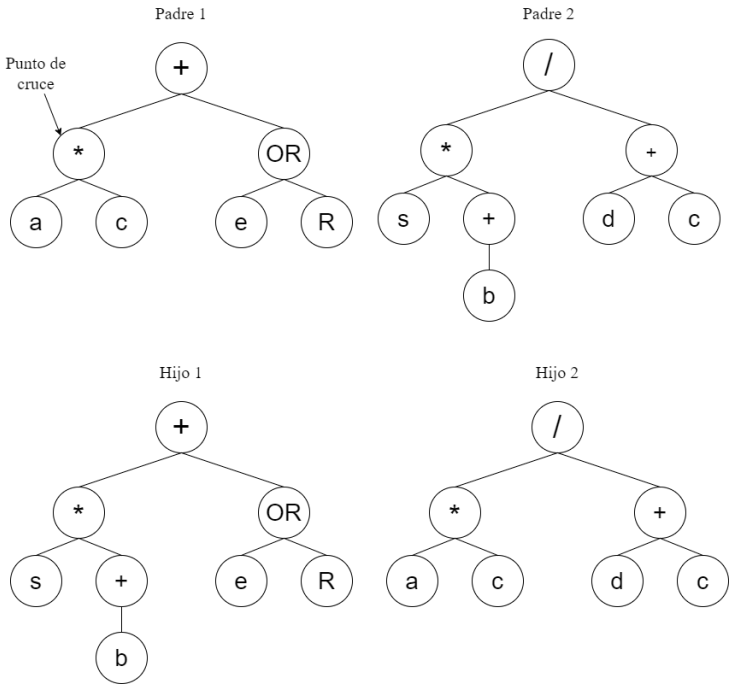

Figura 2.7: Ejemplo de una cruza de árboles (Ghanea-Hercock, 2003)

Mutación por subárboles
La mutación usada es mutación por subárboles que consiste en seleccionar un punto de manera aleatoria en el árbol padre, para después ser sustituido por un subárbol generado de manera aleatoria, también está operación es conocida como cruza "headless chicken"(Angeline, 1997).

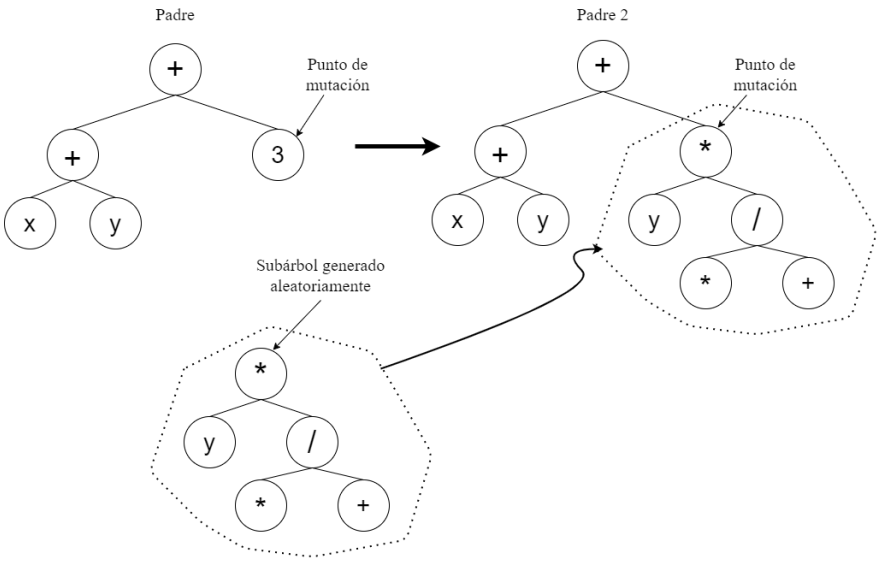

Figura 2.8: Ejemplo de una mutación de árboles (Ghanea-Hercock, 2003)

Algoritmo Evolutionary Optimization of Generalized Sigmoidal Function(Llorente, 2019)

El algoritmo genético de optimización de parámetro de la función de pertenencia generalizada nombrado AG-GSF (Evolutionary Optimization of Generalized Sigmoidal Function) (ver Algoritmo 3) está basada en la función sigmoidal generalizada (GSF) que es mediante la cual descubrimos las mejores configuraciones para los parámetros.

Algorithm 3: Algoritmo de optimización de parámetros EO-GSF

1 El uso de algoritmos genéticos permite descubrir predicados difusos sobre un espacio de búsqueda y
Entrada: Predicado de LDC, Número de generaciones
Salida : Predicado de LDC con igual o mayor valor de verdad
2 N es el número de individuos por generación
$xBest : TipoSolucion \leftarrow EA(N : integer; f : tipoFuncionObjetivo)$
3 var
4 Población
$xg, x * g : array[1, ..., N]$ of $tipoSolucion$
5 **Begin**
Inicializar el número de generación
$g \leftarrow 1$
6 Inicializar la población
$x \leftarrow inicializar(N)$
7 **repeat**
8 Seleccionar los mejores individuos
$x \leftarrow seleccionar(x, f)$
9 Alterar los individuos seleccionados
$x := alterar(x)$
10 $g \leftarrow g + 1$
11 **until** *terminación*
12 Seleccionar la mejor solución encontrada
$xbest := seleccionar(x, f)$
13 **End**

Generación de la población aleatoria
La función de pertenencia generalizada (FPG) se compone de tres parámetros los cuales son β que representa el valor mínimo antes de cero, γ que define el valor difuso 0.5 y m que muestra la tendencia de la función a ser una sigmoidal una sigmoidal negativa o una función convexa.

El cálculo de los valores de cada uno de los atributos hace uso de las siguientes fórmulas:

$$\beta = Rand(Min, ..., Max) < \gamma \tag{2.6}$$

$$\gamma = Rand(Min, ..., Max) > \beta \tag{2.7}$$

$$m = Rand(0, 1) \tag{2.8}$$

El atributo β se selecciona aleatoriamente de los datos, siempre y cuando sea menor que γ, así también γ se selecciona aleatoriamente de entre los datos y debe ser mayor que β para el caso de m se le asigna un valor aleatorio entre 0 y 1.

La representación del cromosoma se representa de tal manera que los estados lingüísticos que estén siendo optimizados se integren en un cromosoma principal (ver en Figura 2.9). En donde cada trío de alelos está compuesto por los parámetros mencionados en el párrado anterior.

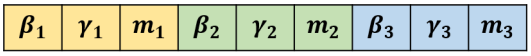

Figura 2.9: Ejemplo de un cromosoma de optimización de predicados de LDC (Llorente, 2019)

Método de selección
Para el método de selección se emplea el de selección por estado uniforme, propuesta por Whitley (Whitley, 1989) y se usa en algoritmos genéticos no generacionales, en los cuales solo unos cuantos individuos son reemplazados en cada generación.

Análisis de la selección de Estado uniforme (Coello, 2004):

- Mecanismo especializado de selección
- Su complejidad es $O(nlogn)$
- Los AGs no generacionales no son muy comunes en aplicaciones de optimización, aunque sí pueden utilizarse.

Algorithm 4: Algoritmo selección uniforme

Entrada: Número de generaciones, porcentaje de cruza y mutación
Salida : mejores N individuos

1 Denominaremos G a la población original de un AG
2 Seleccionar R individuos $1 \leq R < M$ de entre los más aptos
3 Efectuar cruza y mutación a los R individuos seleccionados. Llamaremos H a los hijos
4 Elegir al mejor individuo en H (o a los μ mejores)
5 Reemplazar los μ peores individuos de G por los μ mejores individuos de H

Método de cruza por barajeo
Este método es parecido al método de cruza acentuada. Primero se seleccionan aleatoriamente un número n de alelos del primer padre y se sustituye en el hijo. Por consiguiente, se elige un número aleatorio n de alelos del segundo padre y se sustituye en el hijo, esto se realiza hasta terminar con la creación del hijo.

Algorithm 5: Algoritmo cruza por barajeo

Entrada: Cadenas de texto que representan un árbol de decisión
Salida : Cadena de texto combinada que representa un árbol de decisión

1 **for** $i = 0$ **to** $length(parent1)$ **do**
2 $cartas$ = $random()$ * $lenght(parent1)$
3 **for** $j = 0$ **to** $j < cartas$ && $i < length(parent1)$ **do**
4 $hijo[i]$ = $parent1[i]$
5 **end**
6 $cartas$ = $random()$ * $lenght(parent1)$
7 **for** $j = 0$ **to** $j < cartas$ && $i < length(parent1)$ **do**
8 $hijo[i]$ = $parent2[i]$
9 **end**
10 **if** $i < length(parent1)$ **then**
11 i–;
12 **end**

Método de mutación
Este operador de mutación modifica directamente la representación interna de un número, expresado en codificación binaria. Para ello se selecciona un bit al azar para negarlo y así alterar el valor original. En este caso los valores son reales, pero de igual manera, existen dentro de un rango de valores, por lo cual el alelo seleccionado se muta variando el valor por un valor existente entre su rango mínimo y máximo.

Algorithm 6: Algoritmo mutación generación de subárbol

Entrada: Cadenas de texto que representan un árbol de decisión
Salida : Cadena de texto modificada

$mutaciones$ = $random() * length(parent)$
for $i = 0$ **to** $length(parent1)$ **do**
  $parent[i]$ = $random() * (maximo - minimo) + minimo$ *
end

2.10.2. Programación genética para optimización de predicados

La programación genética (PG) surge como una evolución de los algoritmos genéticos tradicionales, manteniendo el mismo principio de selección natural. De esta manera se busca resolver los problemas mediante la inducción de programas y algoritmos que los resuelvan. Es en esta posibilidad donde la PG muestra potencial, puesto que permite desarrollar de forma automatizada programas, entendiéndose éstos de una forma amplia, es decir, como algoritmos en los que, a partir de una serie de entradas, se genera una serie de salidas(Gestal, Rivero, Rabuñal, Dorado, y Pazos, 2010).

La PG es una extensión de los algoritmos genéticos en la que la población está formada por programas de computación representados en forma de árboles sintácticos (Ceruto Cordovés, Rosete-Suárez, y Espin Andrade, 2010). En la PG se escogen las acciones de los programas (funciones) y los datos sobre los que actúan los programas (terminales), y se opera con ellos hasta encontrar una estructura que represente una solución óptima al problema (Koza, 1990), (Koza, 1992), (Koza, 2007), (O'Reilly, Yu, Riolo, y Worzel, 2006).

2.10.3. Representación de predicados con árboles n-arios

Los predicados difusos están conformados por operadores lógicos y estados lingüísticos; se tiene en cuenta la aridad (ver Tabla 2.2) que es la cantidad de estados o variables lingüísticas que puede admitir cada operador para conformar los predicados difusos.

Una expresión que representa una proposición bien escrita se llama fórmula bien formada (FBF)l dada la definición como sigue:

Se dice que una cadena λ de C es una fórmula bien formada (FBF) si y solo si λ puede obtenerse mediante las siguientes reglas (Mora Espinosa y Nieto Sánchez, 2019):

- λ es una letra proposicional.
- Si λ es fbf, entonces $\neg(\lambda)$ es fbf.
- λ es equivalente a una de las fórmulas $(\alpha) \wedge (\delta)$, o $(\alpha) \vee (\delta)$, o $(\alpha) \rightarrow (\delta)$, o $(\alpha) \leftrightarrow (\delta)$ con α y δ

Entonces el conocimiento expresado en predicados puede ser representando mediante árboles de decisión construidos a través de programación genética en donde los nodos hoja son estados lingüísticos y los nodos internos son operadores de LDC; además la profundidad del nodo se cálcula mediante la cantidad de nodos que deben ser atravesados para alcanzar al nodo raíz del árbol.

Ejemplo de representación del predicado en expresión:

$estado1 \rightarrow estado2$ que en palabras se traduce a *estado1* **Implica** *estado2*

Ejemplo de representación del predicado en árbol:

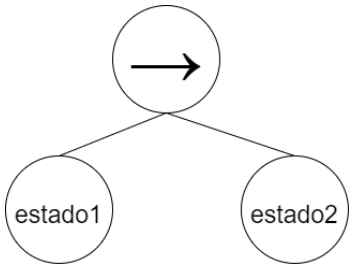

Figura 2.10: Ejemplo de la reresentación de un predicado en árbol.

Construcción de árboles (Métodos árbol de crecimiento y árbol completo)

En (Poli, Langdon, McPhee, y Koza, 2008) se describen dos tipos de construcción de árboles: árbol de crecimiento (grown tree) y árbol completo (full tree); el primero se caracteriza debido a que las construcciones pueden ser diferentes, es decir, que las ramas no necesitan alcanzar la profundidad definida inicialmente, al contrario del segundo método, que se asocia a que todas las hojas deben estar en la misma profundidad.

A continuación, se muestran los algoritmos 7 y 8 para la generación de la población en (Llorente, 2019).

Algorithm 7: Método grow_tree

Entrada: profundidad anterior del árbol, profundidad actual
Salida : árbol de desición

```
probability= rand(100)
if probability ≥ 80 || depth == max_depth then
    ▷ sí el número es mayor que 80 o el nodo es un nodo hoja
    Buffer[position] = rand(linguistic_state)        ▷ selecciona estado linguistico
    position++
    depth++
else
    Buffer[position] = rand(operator_LDC)        ▷ selecciona un operador de LDC
    Position++                                   ▷ avanza a una localidad vacia
    depth++
end
grow_tree(depth,position)
```

Si bien estos métodos son fáciles de implementar y usar, se suele dificultar el control de las distribuciones estadísticas de propiedades importantes como los tamaños y formas de los árboles generados.

Algorithm 8: Método complete_tree

Entrada: predicado, profundidad anterior del árbol, posición actual disponible en el predicado
Salida : predicado modelado como árbol de decisión y última *posición* del predicado

```
1  if Depth = Depth_max then
2      ▷ sí la profundidad es para un nodo hoja agrega variable
3      Buffer[position] = rand(linguistic_state)   ▷ selecciona y almacena estado linguistico
4      return (Buffer, Position)
5  else
6          ▷ si la profundidad es para un nodo interior agrega operador y su aridad
7      Buffer[position] = rand(operator_LDC)   ▷ selecciona un operador de LDC
8      Position++                              ▷ avanza a una localida vacia
9      Buffer[position] = aridad = rand(5)+2   ▷ selecciona aridad
10         ▷ llama a complete_tree() para construir los subárboles hijos del nodo actual
11     for i = 1 to aridad do
12         (Buffer,position) = complete_tree(buffer,Depth-1,position+1)
13         return (Buffer,position)
14     end
15 end
```

En la construcción de la población el algoritmo en (Llorente, 2019) se genera la mitad de los árboles construidos con árboles de crecimiento y la otra mitad son arboles completos, permitiendo tener una mayor variedad de selección y evitar que buenas construcciones no sean tomadas en cuenta debido a las limitantes propias de cada metodología.

2.11. Intensificación y diversificación en algoritmos genéticos

Uno de los principales retos para que un algoritmo genético funcione de manera eficaz es mantener un correcto equilibrio entre los conceptos de intensificación y diversificación también conocidos como explotación y exploración respectivamente, en (Martínez, 2008) se establecen los conceptos de la siguiente manera:

- Diversificación: Este concepto establece la capacidad de explorar las diferentes regiones del espacio de soluciones (S). Esto para encontrar soluciones fiables, es decir, que no dependan de las condiciones de aplicación, como las características concretas del problema de optimización a tratar o particularidades del generador de números aleatorios que se usen, etc.

- Intensificación: se define como la capacidad de explotar la información adquirida de las regiones del espacio de búsqueda visitadas para alcanzar soluciones precisas en las regiones visitadas, siempre y cuando las soluciones presenten buena calidad.

2.12. Descubrimiento de conocimiento a partir del uso de predicados

El uso de predicados difusos para el problema de descubrimiento de conocimiento es divido por una serie de tareas, la primera llamada evaluación de predicados difusos, la segunda denominada descubrimiento de predicados lógicos difusos y la tercera, inferencia basada en clasificación.

Se debe tener en cuenta que la complejidad del problema de descubrimiento de conocimiento está determinado por lo siguiente:

- La cantidad de variables involucradas, que a su vez depende de la cantidad de cláusulas del predicado.
- Los posibles valores que puede tomar cada variable.

2.12.1. Evaluación de predicados difusos

La evaluación de los predicados está basada en la obtención del cálculo de los valores de verdad dado un predicado que es descubierto de un conjuto de datos. Esta tarea se encarga de evaluar el predicado a partir de los valores de las funciones de pertenencia, que son descubiertos por sistemas o herramientas e incluso por personas expertas que ya hayan especificado el valor de los parámetros de esas funciones.

2.13. Descubrimiento de predicados difusso

El objetivo del descubrimiento de predicados difusos es crear relaciones entre los estados lingüísticos a partir de un conjunto de datos, siempre y cuando cumplan con las especificaciones del usuario. La búsqueda se puede realizar mediante metodologías heurísticas para la búsqueda de predicados difusos y para el ajuste de los parámetros de las funciones de pertenencia definidas en los estados lingüísticos.

2.14. Inferencia basada en clasificación

La tarea de inferencia primero descubre predicados difusos dentro de un conjunto de datos en el que se han definido los estados lingüísticos de variables de condición y decisión. Los predicados difusos obtenidos son utilizados para inferir los valores de las variables de decisión a partir de otro set de datos en el que sólo se conocen las variables de condición.

Para el proceso de toma de decisiones, dentro de las etapas de Minería de datos se encuentra la clasificación que se encarga de catalogar un dato dentro de una de las clases asumiendo que estás son disjuntas (Riquelme Santos, Ruiz, y Gilbert, 2006), esta tarea requiere un control adecuado de los datos que se dispongan ya que es un reto debido al aumento de volumen de las bases de datos.

Capítulo 3

Estado del arte

En esta sección se mencionan las investigaciones y el uso del software que se usó para las diferentes tareas con respecto al problema de generación de conocimiento y su representación; está dividido en dos secciones en donde el primero describe metodologías para la generación de predicados, como en sistemas de apoyo para la toma de decisiones y el segundo en la inferencia basada en clasificación con predicados de LDC.

3.1. Generación de conocimiento para la toma de decisiones

A continuación, se describen metodologías para generar conocimiento a través de reglas de inferencia o predicados difusos basadas en algoritmos genéticos basados en LDC permitiendo modelar el conocimiento de expertos (que por simplicidad se denominará algoritmo de referencia) y por último software comercial.

3.1.1. Algoritmos de optimización

Los siguientes algoritmos que son descritos en el marco teórico; el primer algoritmo como se ha mencionado antes descubre predicados basados en la lógica difusa compensatoria y el segundo optimiza los parámetros de las funciones de pertenencia; para ambos algoritmos las instancias utilizadas fueron Tinto, Order y Bupa.

Algoritmo Evolutionary Knowledge based on Compensatory Fuzzy Logic

El algoritmo de descubrimiento de conocimiento expresado en predicados de LDC nombrado EK-CFL (Evolutionary Knowledge based on Compensatory Fuzzy Logic), a través de un algoritmo genético que hace uso de las metodologías de programación datos usando varias metodologías de LDC, el funcionamiento general de genética, se lleva a cabo la evolución de los predicados y descubre reglas en instancias.

Uno de sus experimentos tuvo como objetivo generar predicados con Lógica Difusa Compensatoria (ver Figura 3.1), haciendo una comparación con el software Eureka Universe, para después evaluarlos por el cuantificador de universalidad (forAll).

Predicados descubiertos		
Mínimo valor de verdad	**Predicados construidos por Universe**	**Predicados construidos por Eureka**
0.99	NOT (IMP "mid residual_sugar" "low citric_acid")	(IMP(NOT(IMP"midchlorides" "lowresidual_sugar"))"lowsulphates")
0.98	NOT (IMP "high residual_sugar" "low alcohol")	(OR(IMP"midchlorides" "lowchlorides")"lowresidual_sugar")
0.97	AND (AND "low residual_sugar" "mid free_sulfur_dioxide" "mid residual_sugar" "high alcohol" "high density" "mid alcohol" "low volatile_acidity" "low fixed_acidity" "low total_sulfur_dioxide" "high residual_sugar" "mid chlorides" "high pH" "low chlorides") (AND "low pH" "low residual_sugar" "mid sulphates" "high total_sulfur_dioxide" "mid chlorides" "mid density" "low chlorides" "mid citric_acid")	(IMP"midsulphates" (NOT"midresidual_sugar"))
0.96	NOT (IMP "high residual_sugar" "low free_sulfur_dioxide")	(IMP(AND"highchlorides" (AND"middensity" "highchlorides" "midresidual_sugar" "highpH" "midchlorides" "midtotal_sulfur_dioxide"))"lowalcohol")
0.95	NOT (IMP "mid sulphates" "low total_sulfur_dioxide")	(OR"lowalcohol" (IMP"midvolatile_acidity" "midcitric_acid")(NOT"midchlorides")(EQV"midchlorides" "lowresidual_sugar"))

Figura 3.1: Extracto de predicados construidos por el algoritmo EK-CFL (Llorente, 2019)

Algoritmo Evolutionary Optimization of Generalized Sigmoidal Function

En esta sección se presenta el algoritmo genético de optimización de parámetro de la función de pertenencia generalizada nombrado EO-GSF (Evolutionary Optimization of Generalized Sigmoidal Function) en, la cual está basada en la función sigmoidal generalizada (GSF) que es mediante la cual se descubren las mejores configuraciones para los parámetros.

N°	Predicados generados
1	(IMP (AND "new citric_acid" "new free_sulfur_dioxide" "new alcohol" "new fixed_acidity" "new pH" "new residual_sugar" "new volatile_acidity" "new sulphates" "new density" "new chlorides" "new total_sulfur_dioxide") "new quality")
2	(IMP (NOT "new chlorides") "new quality")
3	(IMP (OR "new pH" "new density" "new fixed_acidity" "new chlorides") "new quality")
4	(IMP (OR "new volatile_acidity" "new density") "new quality")
5	(IMP (NOT "new sulphates") "new quality")
6	(IMP (OR "new chlorides" "new total_sulfur_dioxide" "new sulphates" "new citric_acid" "new fixed_acidity" "new free_sulfur_dioxide" "new pH" "new density" "new residual_sugar") "new quality")
7	(IMP (OR "new sulphates" "new volatile_acidity" "new residual_sugar" "new pH") "new quality")
8	(IMP (OR "new alcohol" "new residual_sugar" "new volatile_acidity" "new sulphates" "new pH" "new fixed_acidity" "new citric_acid" "new density" "new total_sulfur_dioxide") "new quality")
9	(IMP (OR "new fixed_acidity" "new pH") "new quality")
10	(IMP (NOT "new volatile_acidity") "new quality")
11	(IMP (OR "new sulphates" "new residual_sugar" "new free_sulfur_dioxide" "new chlorides") "new quality")
12	(IMP (OR "new residual_sugar" "new total_sulfur_dioxide" "new pH" "new citric_acid" "new density" "new free_sulfur_dioxide" "new alcohol") "new quality")
13	(IMP (IMP "new chlorides" "new volatile_acidity") "new quality")
14	(IMP (IMP "new total_sulfur_dioxide" "new density") "new quality")
15	(IMP (OR "new citric_acid" "new free_sulfur_dioxide" "new alcohol" "new fixed_acidity" "new pH" "new residual_sugar" "new volatile_acidity" "new sulphates" "new density" "new chlorides" "new total_sulfur_dioxide") "new quality")

Figura 3.2: Extracto de predicados construidos por el algoritmo EK-CFL para optimizar su función de pertenencia (Llorente, 2019)

De los predicados presentados en la Figura 3.2 se calcula el mejor valor de verdad a través de la optimización de sus funciones de pertenencia.

Predicado	Eureka-Universe	Universe
1	**0.976430137**	0.81161477
2	0.982110969	**0.98876695**
3	**0.90305892**	0.79506811
4	**0.916270852**	0.84122238
5	0.973208652	**0.97360045**
6	**0.866627112**	0.73201431
7	**0.909053824**	0.81927653
8	**0.874235399**	0.72556138
9	**0.918507379**	**0.844661268**
10	0.966643202	**0.967475837**
11	**0.916384717**	0.870367931
12	**0.885982721**	0.748333851
13	**0.933014655**	0.902903347
14	**0.92712822**	0.863925856
15	**0.861977535**	0.707722381
Promedio	**0.920708**	**0.83950102**
DS	**0.03978909**	**0.090873919**

Figura 3.3: Extracto de predicados construidos por el algoritmo EK-CFL para optimizar su función de pertenencia (Llorente, 2019)

Los resultados obtenidos de la Figura 3.3, fueron evaluados por la prueba de Wilcoxon, para determinar si estadísticamente existe diferencia entre uno y otro en donde se concluye algoritmo Eureka-Universe es superior a Universe, mediante la obtención de un P-Value de **0.0035**.

3.1.2. Algoritmo multiobjetivo para extraer predicados difusos

Una metodología para el descubrimiento de predicados difusos es propuesto en (Osorio Roig et al., 2015) utilizando metaheurísticas y aborda las limitaciones de solo evaluar una sola función objetivo a múltiples objetivos; la métrica que se usa es el valor de verdad que mide la cantidad de tuplas que satisfacen un predicado.

El método de referencia es llamado FuzzyPred que es propuesto en (Ceruto Cordovés et al., 2010); la manera en como trabaja el método es obteniendo un conjunto de predicados difusos y es modelado como un problema de optimización combinatoria de un solo objetivo, evaluado mediante una única métrica de calidad conocida como el valor de verdad del predicado.

El trabajo de (Osorio Roig et al., 2015) implementa otras métricas lo que lo convierte en multiobjetivo,las métricas que se añaden son soporte, este indica en cuantas tuplas está presente el predicado su valor oscila entre cero y uno, lift conjuntivo donde establece el nivel de dependencia de los predicados simples cuando se relacionan a través de un operador conjunción y lift disyuntivo cuando el nivel de dependencia se relacionan a través de un operador de disyunción.

MFuzzyPred como es llamado el método, construye una solución global que está conformado con los mejores predicados, a su vez, cada predicado es representado en un cromosoma expresado por la fórmula 3.1:

$$T = A * B + 2 \tag{3.1}$$

Donde:

- A: Representa la cantidad de atributos de la vista minable.
- B: La cantidad máxima de cláusulas del predicado
- Las dos posiciones del final indican la cantidad real de cláusulas y el tipo de forma normal utilizada en el predicado que son la Forma Normal Conjuntiva (FNC) y Disyuntiva (FND).

La construcción del cromosoma es de longitud fija y su codificación está dada por valores enteros generados aleatoriamente de acuerdo a una escala definida:

Escala:

- 0 = no está en el predicado
- 1 = está en el predicado
- 2 = aparece negada
- 3 = aparece con el modificador muy

Un ejemplo de construcción de la solución se da a continuación en la Figura 3.4, en donde se muestra el genotipo y el fenotipo que se interpreta como sigue: En la primera cláusula la variable X_2 aparece en conjunción con la variable X_3 negada; en la segunda cláusula la variable X_0 aparece en conjunción con la variable X_1 la cual está afectada por el modificador muy.

a)

X_0	X_1	X_2	X_3	X_0	X_1	X_2	X_3	CRC	FN
0	0	1	2	1	3	0	0	2	DFN

b)

$$(X_2 \ \wedge \ \neg X_3) \vee (X_0 \ \wedge \ \neg X_1)$$

Figura 3.4: a) Genotipo del método MFuzzyPred; b) Fenotipo del método MFuzzyPred (Osorio Roig et al., 2015)

La configuración de los algoritmos son los siguientes: Escalador de Colinas Estocástico Multiobjetivo (ECMO), Búsqueda Tabú Multiobjetivo (BTMO, la lista Tabú es de tamaño 20), Recocido Simulado Multiobjetivo (RSMO, con un alfa de 0.9, temperatura inicial 20, y temperatura final 0), Escalador de Colinas Estocástico con Reinicio Multiobjetivo (ECMOR, con tamaño de la vecindad del estado actual de 2).

Para la experimentación los operadores de Lógica Difusa que se usan son: Min-Max (Zadeh), Media Geométrica y su Dual (Lógica Difusa Compensatoria) y las base de datos que se usaron fueron: Basketball que contiene 96 instancias y 14 atributos y Stock que contiene 950 instancias y 20 atributos, ambas disponibles en el repositorio UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/).

El objetivo del caso de estudio es analizar el comportamiento de los atributos (etiquetas lingüísticas) según los operadores de lógica difusa (Zadeh y LDC) y la conectiva CNF. Por otra parte se analiza el trayecto de los estados según las evaluaciones de soporte, el valor de verdad, lift conjuntivo y lift disyuntivo

Tabla 3.1: Resultado de las métricas de calidad para la base de datos Stock(Osorio Roig et al., 2015)

Algoritmos	Valor de verdad	Soporte	Lift conjuntivo	Lift disyuntivo
ECMO	0.681	0.704	0.782	0.994
BTMO	0.589	0.613	0.708	0.993
RSMO	0.638	0.666	0.752	0.993
ECMOR	0.775	0.819	0.985	0.974
Promedio	**0.670**	**0.7005**	**0.8067**	**0.9885**

Tabla 3.2: Resultado de las métricas de calidad para la base de datos Stock(Osorio Roig et al., 2015)

Algoritmos	Valor de verdad	Soporte	Lift conjuntivo	Lift disyuntivo
ECMO	0.561	0.601	0.684	0.982
BTMO	0.500	0.536	0.607	0.978
RSMO	0.559	0.592	0.659	0.982
ECMOR	0.596	0.733	0.913	0.992
Promedio	**0.554**	**0.6155**	**0.7157**	**0.9835**

El fin de la experimentación demuestra que es es posible obtener conocimiento a través del uso de algoritmos metaheurísticos, aunque está construcción solo se limita a dos formas: la Forma Normal Conjuntiva (FNC) y Disyuntiva (FND), centrándose solamente en la calidad de los predicados encontrados.

3.1.3. Sistemas de apoyo para la toma de decisiones

Software Fuzzy Tree Studio

Fuzzy Tree Studio (FTS) es un producto de software concebido como un sistema de soporte de decisiones. Aborda los conceptos de lógica difusa mediante una interfaz gráfica de usuario amigable. El objetivo principal es ayudar en el análisis de datos mediante la aplicación de conocimientos expertos previos que permitan evaluar y comparar alternativas (Passoni, Meschino, Gesualdo, y Monjeau, 2021)

FTS está conformado por tres partes principales: el editor y visualizador de árboles, el generador de datos y el analizador de consultas.

Etapas del funcionamiento del software:

- Generación de datos: el usuario puede importar datos de archivos o escribirlos en una tabla. Los datos pueden ser numéricos o palabras clave, opciones de la lista de opciones. Los datos deben ser coherentes con respecto al árbol diseñado.

- Evaluación del árbol:con base en un conjunto de datos, se otorga el grado de verdad de la raíz para cada caso. Pero además, se entrega información gráfica y numérica útil, para permitir el análisis de los casos particulares.

No obstante, a partir de su frecuente utilización se han encontrado algunas deiciencias en su funcionamiento desde el punto de vista informático:

- Interfaz de usuario poco amigable
- Muy complicado para usuarios con menor experiencia.
- Errores al cargar los datos de los archivos .TXT para procesarlos.
- Imposibilidad de presentar los predicados en un árbol de decisión.
- Procedimiento muy complicado para insertar los predicados compuestos

Eureka Universe

El software universe (Eurekas, 2021) tiene como fin en un principio implementar las metodologías de lógica difusa y lógica difusa compensatoria para el tratamiento de datos. El software es ejecutado desde línea de comandos tanto para Windows y para Mac, que también puede ser ejecutada en Linux.

Universe es un sistema único que permite resolver tareas y combinarlas en soluciones para la toma de decisiones, basado en lenguaje natural.

Las tareas que lleva a cabo este software son:

- Evaluación: Calcula los valores de verdad de un predicado difuso de un set de datos.
- Descubrimiento: Busca relaciones entre los estados lingüísticos de un set de datos (predicados difusos) que cumplan con las especificaciones del usuario.
- Inferencia: Realiza primeramente una tarea de descubrimiento en un set de datos en el que se han definido los estados lingüísticos de variables de condición y decisión. Los predicados difusos obtenidos son utilizados para inferir los valores de las variables de decisión a partir de otro set de datos en el que sólo se conocen las variables de condición.

3.2. Inferencia basada en clasificación con predicados de Lógica Difusa Compensatoria

Algoritmo Virtual Savant basado en Lógica Difusa Compesantoria para el problema de empacado de objetos

Para contribuir a la solución del problema de empacado de objetos en contenedores (BPP, por su sigla en inglés) En (Padrón-Tristan, 2019) se propone el método VS-LDC, el cual aprende el comportamiento de un algoritmo de referencia aplicado a la solución de instancias de BPP con solución conocida. Posteriormente, VS-LDC aplica el modelo aprendido a la solución de nuevas instancias de mayor dimensión. El enfoque seguido, conocido como Virtual Savant, fue concebido

para el cómputo paralelo.

Su metodología se basa en un convertidor de instancias de optimización para la clasificación y viceversa, un clasificador probabilístico basado en lógica difusa compensatoria, y heurísticas de refinamiento de la solución.

3.3. Comentarios finales

El proyecto se basa en la integración de los algoritmos EK-CFL y EO-GSF que se presentan como algoritmo de referencia para mayor sencillez, principalmente este solo cuenta con la implementación de un solo asterisco para la búsqueda de predicados, en comparación al propuesto que implementa múltiples para ofrecer un enfoque con mayor exhaustividad en el espacio de soluciones.

Se ha demostrado que usar algoritmo metaheurísticos son un método de solución para abordar el problema de obtención de conocimiento ya que que brinda resultados competitivos sin embargo se observa que su estructura es limitada. Por otra parte, se implementa una mutación constructiva para mejorar la expresividad de los predicados difusos descubiertos, también se observa que los proyectos explorados fungen como sistemas de apoyo para la toma de decisiones. En la tabla 3.5 se realiza esta comparación.

Trabajos	**Algoritmo de optimización**	**Sistema de apoyo para la toma de decisiones**	**Expresividad**	**Exhaustividad**
Algoritmo de referencia (Llorente, 2019)	Módulo separados	✓	✓	Uso de 1 asterisco
MFuzzyPred	✓	✓	X	X
Fuzzy Tree Studio	X	✓	X	X
Algoritmo Virtual Savant basado en LDC para el problema de BPP (Tristan, 2019)	✓	✓	X	Uso de 1 asterisco
Eureka Universe	✓	✓	X	Uso de 1 asterisco
Algoritmo propio	Integrados	✓	✓	Múltiples asteriscos

Figura 3.5: Comparación de proyectos con el estado del arte

Capítulo 4

Metodología de solución

De acuerdo con los capítulos anteriores en donde se han expuesto los conceptos relevantes para el desarrollo de esta tesis, así como una revisión de propuestas para la solución del problema de generación de conocimiento con predicados difusos, dentro de este capítulo se detalla el proceso para ofrecer una contextualización de lo que se propone.

Se describe la arquitectura general del algoritmo integrado y en secciones posteriores la propuesta de la metodología de solución con la integración de dos métodos: una mutación constructiva y un método de incremento de la exhaustividad del espacio de soluciones para la búsqueda de predicados basado en la lógica difusa arquimediana compensatoria.

La lógica difusa arquimediana compensatoria como se menciona en la sección 2.7, es una metodología multivalente que satisface las características de enfoque descriptivo de la toma de decisiones y el enfoques normativos para la toma de decisiones.

4.1. Propuesta de solución

Para contribuir a la solución de generación de conocimiento se integra el algoritmo de referencia (Llorente, 2019) aplicando nuevos métodos para diversidad y exhaustividad en la búsqueda y optimización de predicados.

4.2. Arquitectura del algoritmo propuesto Eureka Universe Archimedean Compensatory Fuzzy Logic with Genetic Programming

El método de descubrimiento de predicados propuestos consta de dos algoritmos genéticos anidados, en donde el primero busca predicados de alto valor de verdad y el segundo optimiza funciones de pertenencia ajustándolas a los datos, es decir, se pueden generar predicados para después ser evaluados como también poder optimizar los parámetros de los predicados descubiertos.

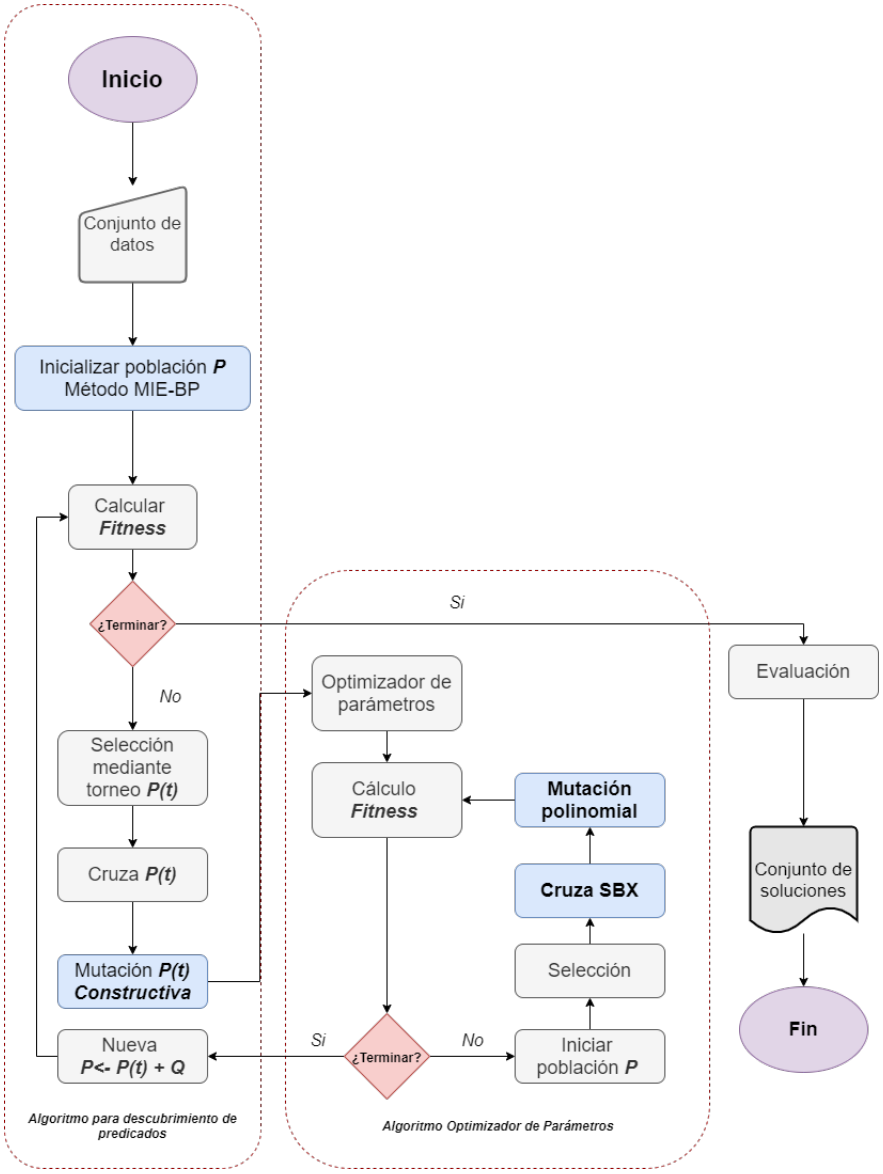

Figura 4.1: Diagrama de flujo del algoritmo EU-ACFL-GP

Como se observa en la figura 4.1 el algoritmo está anidado por los dos algoritmos, el primero genera una población de soluciones, es decir, realiza un descubrimiento de predicados y está conectado con el algoritmo optimizador de parámetros; comienza inicializando una población P de manera aleatoria en donde se establece por el usuario la configuración de los asteriscos tales como las variables lingüísticas, operadores lógicos y la profundidad.

La generación de la población se realiza mediante el algoritmo 2 que es el método de rampa; después cada individuo es evaluado por su aptitud; si la aptitud es buena se realiza la evaluación del predicado; de lo contrario se entra al primer algoritmo para evolucionar los predicados y después analizar los datos para mejorar los parámetros de la función de pertenencia.

Operador de cruza Simulated Binary Crossover

El algoritmo optimizador de parámetros utiliza el operador Simulated Binary Crossover (SBX). Incluye un parámetro que calcula la extensión asociada con las distribuciones de probabilidad utilizadas para la generación de los hijos y su grado de diversidad inducido por estos operadores se ajusta cambiando el valor de estos parámetros.

El factor de dispersión (Deb y Beyer, 2001) se define en relación de la diferencia en los valores de los hijos respecto a los valores de los padres como se define en la ecuación 4.1:

$$\beta = \left| \frac{x_i^{2,t+1} - x_i^{1,t+1}}{x_i^{(2,t)} - x_i^{(1,t)}} \right| \tag{4.1}$$

De acuerdo con (Sánchez, Lozano, y Herrera, 2007) se define una distribución de probabilidad de los hijos basada en la distancia de los padres en donde los padres si están muy cerca el uno del otro, los hijos generados podrían distribuirse alrededor de los padres. De lo contrario, si los padres están alejados entre sí, los hijos apenas se distribuirán a su alrededor.

La distribución de probabilidad utilizada para crear una solución secundaria se deriva para tener un poder de búsqueda similar al de un cruce de un solo punto en los algoritmos genéticos codificados en binario.

$$P(\beta) = \begin{cases} 0.5(n+1)\beta^n & \text{si } \beta \leq 1; \\ 0.5(n+1)\frac{1}{\beta^{n+2}} & de \quad lo \quad contrario \end{cases} \tag{4.2}$$

La Figura 4.2 muestra la distribución de probabilidad anterior dados los valores para $n = 2$ y 5 para crear soluciones descendientes a partir de dos soluciones padres donde el padre 1 tieen un valor de $x_i^{(1,t)} = 2.0$ y $x_i^{(2,t)} = 2.0$ en el espacio real. En las expresiones anteriores, la distribución índice n es cualquier número real no negativo. Un gran valor de n da una mayor probabilidad de crear cerca soluciones padres que por el contrario un pequeño valor de n permite seleccionar soluciones distantes como soluciones descendientes. Utilizando Ecuación 4.2 se calcula β_q.

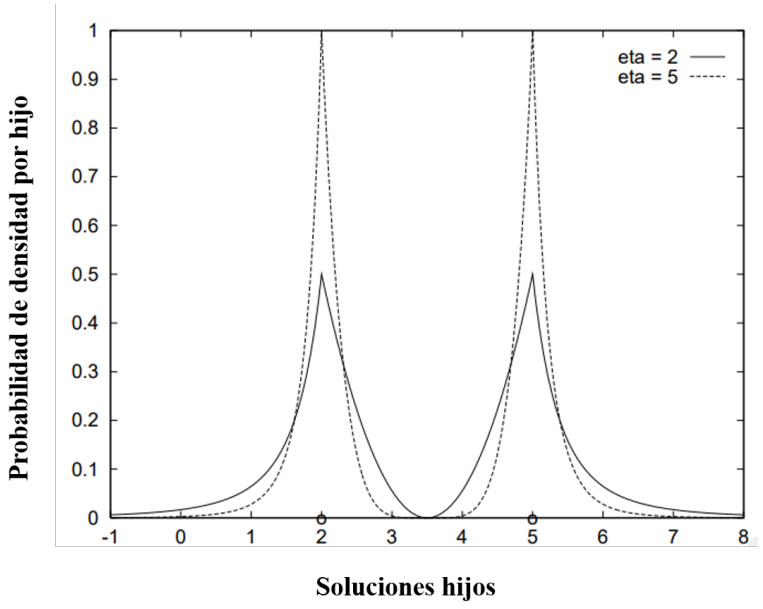

Figura 4.2: Probabilidad de distribución para la creación de soluciones descendientes de variables continuas. (Deb y Beyer, 2001)

El algoritmo determina que cada gen del hijo se genera en el vecindario del correspondiente gen de uno de los padres; los pasos de este proceso se ilustran en el algoritmo 9.

Algorithm 9: Operador de cruza SBX

Escoger un número aleatorio $\upsilon \in$ [0,1]
Calcular la probabilidad de distribución β_q **if** $\upsilon \leq 0.5$ **then**
  $(2\upsilon)^{\frac{1}{n+1}}$
else
  $(\frac{1}{2(1-\upsilon)})^{\frac{1}{n+1}}$
end
Calcular la solución del hijo 1
$x_i^{(1,t+1)} = 0.5[(1+\beta_q)x_i^{(1,t)} + (1-\beta_q)x_i^{(2,t)}]$
Calcular la solución del hijo 2
$x_i^{(2,t+1)} = 0.5[(1-\beta_q)x_i^{(1,t)} + (1+\beta_q)x_i^{(2,t)}]$

Se observa que las soluciones de dos hijos son simétricas con respecto a las soluciones de los padres. Esto se usa deliberadamente para evitar un sesgo hacia cualquier solución principal en particular en una sola operación de cruce. Otro aspecto de este operador de cruza es, que para un fijo, las soluciones descendientes tienen una extensión que es proporcional a la de las soluciones principales.

Operador de mutación polinomial

La manera en como trabaja el operador de mutación polinomial es a través de una solución padre dada, está genera soluciones con una probabilidad alta cercanos a esa solución. Se usa una distribución de probabilidad polinomial que es calculada con la ecuación 4.3.

$$x_i^{(t+1)} = x_i^{(t)} + (x_i^{(U)} - x_i^{(L)}) * \delta_i \tag{4.3}$$

Donde $u_i < 0.5$, la δ_i es igual a:

$$(2 * u_i)^{\frac{1}{(n+1)}} - 1 \tag{4.4}$$

Y si $u_i \geq 0.5$, entonces δ_i resulta:

$$1 - [2 * (1 - u_i)]^{\frac{1}{(n+1)}} \tag{4.5}$$

Donde $u_i \sim \cup[0,1]$ y n es un parámetro que controla la variabilidad de la perturbación.

4.2.1. Propuesta para el incremento de la búsqueda de predicados

El método de incremento de la exhaustividad de la búsqueda de predicados, MIE-BP, es propuesto para la guía de búsqueda de predicados mediante el uso de una estructura de árbol flexible que permite la incorporación de múltiples componentes de búsqueda, llamados comodines. Esta estrategia amplía el espacio de búsqueda para incluir más predicados factibles con la finalidad de incrementar la exhaustividad de la búsqueda, y en consecuencia facilitar el descubrimiento de conocimiento oculto en los datos.

Uso del predicado simbólico en la búsqueda de conocimiento

El predicado simbólico es una forma de representación de un predicado difuso en el que se puede utilizar el asterisco * como comodín en sustitución de estados lingüísticos o predicados difusos, de forma similar a las expresiones regulares.

Usar solo * representa el predicado simbólico más general que se puede especificar en una tarea de descubrimiento, para generar poblaciones aleatorias de predicados difusos.

Ejemplo. Predicado simbólico general

En la Tabla 4.1 se muestra los elementos que participan para el descubrimiento de predicados difusos.

Tabla 4.1: Configuración del predicado simbólico

Elementos	**Valores**
Predicado	"*"
Operadores	(AND OR IMP EQV NOT)
Variables	estado1, estado2,estado3, estado4

A continuación, se muestran algunos ejemplos de los predicados difusos que pudieran generarse:

- (AND estado1 estado3 estado2 estado4)
- (IMP estado2 estado1)
- (OR estado3 estado4)
- (NOT estado2)

Algoritmo Método de Incremento de la Exhaustividad de la Búsqueda de Predicados

En el siguiente algoritmo se muestra la estructura del método MIE-BP, recibe como entrada una serie de elementos que son, la profundidad, lista de estados que corresponden a las etiquetas del conjunto de datos y el número máximo de elementos que pueden tener por nodo; la salida obtenida será un árbol.

Algorithm 10: Método MIE-BP

Entrada: $profundidad$, $lista\_estados$, $lista\_operadores$, $max\_elementos$
Salida : $arbol$

Validar la profundidad actual
if $profundidad < (max\_profundidad)$ **then**
    Escoger un operador aleatoriamente
    $arbol \leftarrow random(lista\_operadores)$
    El número máximo de elementos se recorre en función del operador lógico seleccionado
    for n **to** $random(aridad(arbol), max\_elementos$ **do**
        Se agrega un hijo de acuerdo con los estados disponibles y se actualiza la profundidad
        $arbol.anadirHijo(generarSolucion(lista\_estados, profundidad + 1)$
    end
    return $arbol$
else
    return $random(arbol)$
end

Un ejemplo de la implementación del algoritmo se muestra con el siguiente caso: $(AND \quad *_1, *_2, *_3)$, en la Figura 4.3 se muestra la configuración de cada asterisco y en la figura 4.4 la representación de los predicados difusos en forma de árbol.

ID asterisco	**"$*_1$"**	**"$*_2$"**	**"$*_3$"**
Operadores	AND, OR, NOT	AND, OR, NOT	AND, OR, NOT
Variables	free_sulfur_dioxide, density	Sulphates, alcohol	total_sulfur_dioxide, volatile_acidity, citric_acid
Pc (profundidad comodín)	prof -1	prof -1	prof -1

Figura 4.3: Configuración de comodines para el predicado $AND \quad *_1, *_2, *_3$

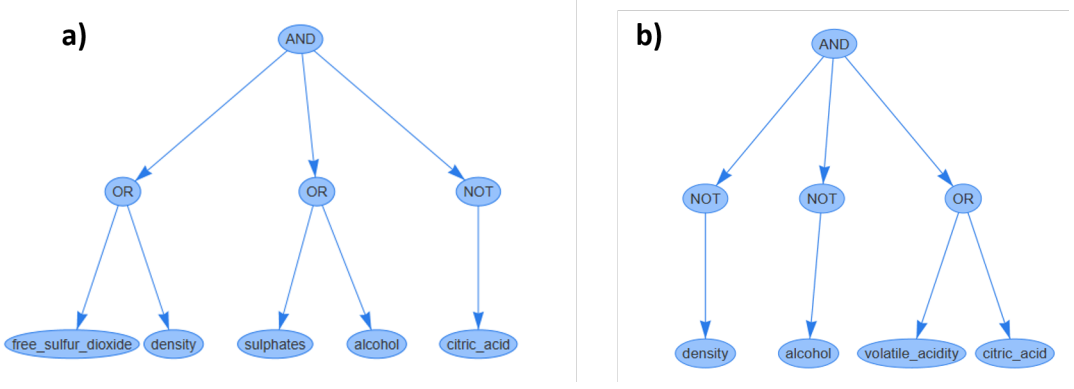

Figura 4.4: Representación de dos soluciones de comodines con múltiples asteriscos AND $*_1, *_2, *_3$

4.2.2. Propuesta para la diversidad en la búsqueda de predicados difusos

La diversidad evitar la convergencia prematura, es decir, no se estanca en zonas en donde no se encuentra el óptimo global.

Mutación constructiva

Dentro de la mutación constructiva, se eligen aquellos individuos que quedan después de las operaciones de selección y cruza. Luego, se aplica la búsqueda local hill climbing en ellos, es decir, se generan individuos hasta que se obtienen mejores individuos que los padres. La ventaja de usar la mutación constructiva es que se reduce la naturaleza destructiva de la operación de mutación transfiriendo solo a los mejores individuos que los padres a la siguiente generación. Otra ventaja es que se elige los padres mutantes después del cruce y la selección, así los individuos que no son buenos para producir mejores descendientes que los padres se modifican en la mutación (Bhardwaj, Tiwari, Varma, y Krishna, 2014).

En el algoritmo 11 se muestra el procedimiento:

Algorithm 11: Mutación constructiva

1 **Begin**
2 Generar una población inicial (k)
3 Seleccionar aleatoriamente los porcentajes de la población inicial para cruza (P_c), mutación (P_m) y reproducción (P_r).
4 Agrupar la población total que queda después de la reproducción $(N_c$ y $N_m)$ en pares.
5 Aplicar el método Hill Climbing que es tomar la pareja de padres y generar descendientes hasta que podamos mejorar a los padres.
6 Repetir este proceso para todos los pares.
7 Seleccionar a los mejores individuos (P_c) y eliminar los individuos restantes para la próxima generación.
8 Repita los pasos 2 a 7, hasta que alcanzar el número requerido de generaciones o porcentaje de satisfacción requerido.
9 **End**

En la figura 4.5 se muestra un ejemplo del proceso de esta mutación.

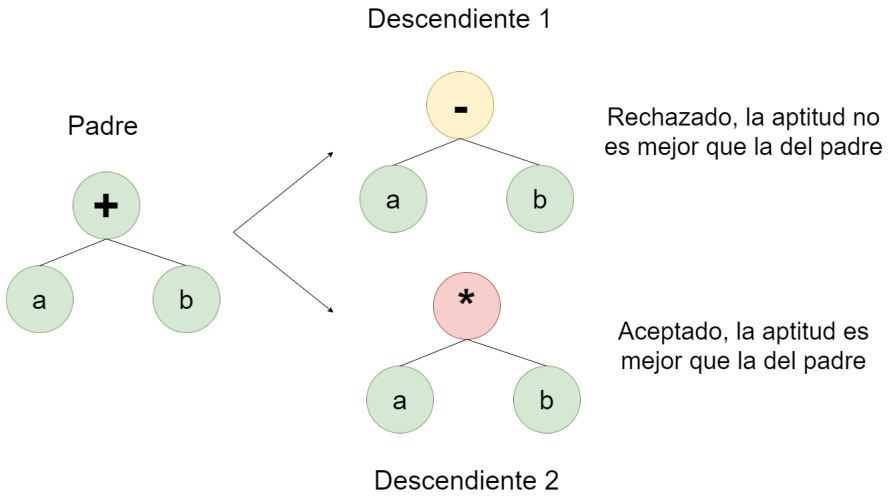

Figura 4.5: Mutación constructiva

Para mejorar la calidad de las soluciones se propone una mutación constructiva en donde se eligen aquellos individuos que quedan después de la selección y cruza. Luego se aplica el método Hill-Climbing escalando la búsqueda en ellos, es decir, generar mejores hijos que los padres. La ventaja de usar la mutación constructiva es que se reduce la naturaleza estructurante de la operación de mutación transfiriendo solo los mejores individuos que los padres para la próxima generación.

Algorithm 12: Propuesta de Mutación constructiva

Entrada: arbol actual $cTree$; conjunto de nodos editables E; conjunto de estados linguisticos L; conjunto de operadores O.
Salida : mejor arbol $temporalBestTree$

Inicializar una variable temporal $temporalBestTree \leftarrow cTree$
for i **to** $|E|$ **do**
  Elija aleatoriamente un nodo editable $nodeMutation \leftarrow random(E)$
  Inicializar una variable temporal $fitnessTemporalBestTree \leftarrow -1$
  Se crea un subárbol con estados lingüísticos y operadores disponibles. $subTree \leftarrow createSubTree(L, O)$
  Agregar un subárbol sobre la mutación del nodo $newTree \leftarrow addSubTree(nodeMutation, subTree)$
  if *fitness(newTree)* >*fitness(temporalBestTree)* **then**
    $temportalBestTree \leftarrow newTree$
    break
  else
    $fitness(newTree) > fitnessTemporalBestTree$
  end
  $temporalBestTree \leftarrow newTree$
  $fitnessTemporalBestTree \leftarrow fitness(temporalBestTree)$
end
return $temporalBestTree$

4.3. Métricas para la evaluación de algoritmos

Las medidas para evaluar el desempeño son las siguientes:

Valor de verdad

La función de evaluación es aquella por medio de la cual se obtiene el valor de pertenencia global de cada predicado descubierto, esto significa hacer uso de la constante de universalidad expresada en las operaciones de conjuntos, y por medio de esta determinar que tanto el individuo cumple con las reglas para todos los objetos pertenecientes a la instancia de datos (Llorente, 2019).

$$\underset{p_L \in P_L}{maxtruevalue}(p_L) = C_{x \in \cup} P_l(x) \tag{4.6}$$

Donde:

P = Universo de predicados que pueden ser generados mediante el uso de operadores y variables.
p= Predicado seleccionado del universo de predicados.
U = Conjunto de objetos que conforman la instancia.
x = Objeto que forma parte del conjunto de objetos.
$\cup$ = operador de conjunción lógica de cada registro que evalúa el predicado.
L = Metodología de LDC que evalúa el predicado.

Exactitud de clasificación

La exactitud es una métrica para evaluar modelos de clasificación. Es es la fracción de predicciones que el modelo realizó correctamente que se calcula de la siguiente manera (Developers, 2020):

$$exactitud = \frac{N\acute{u}mero \quad de \quad predicciones \quad correctas}{N\acute{u}mero \quad total \quad de \quad predicciones} \tag{4.7}$$

Tiempo de ejecución

Se hace el uso del módulo Time que regresa el tiempo en segundos desde, se realiza una estimación a través de la media de un número n de ejecuciones del algoritmo.

Capítulo 5

Experimentación y resultados

En este capítulo se presenta un conjunto de experimentos realizados para evaluar la calidad del algoritmo propuesto EU-ACFL-GP con el uso de la lógica difusa arquimediana compensatoria. Se contrasta su desempeño contra la integración de los algoritmos EK-CFL y AG-GSF, que por simplicidad se denominará algoritmo de referencia.

5.1. Diseño experimental general

A continuación se muestra la estructura de

5.1.1. Configuración del entorno para la experimentación

La experimentación se hizo en una máquina con las siguientes características: marca DELL con un procesador Intel Core i5-9500M CPU con 3.00GHz, memoria RAM de 12GB y Sistema Operativo Windows 10.

5.1.2. Descripción de las instancias utilizadas:

A continuación, en la Tabla 5.1 se muestran las instancias (NSF, 1987) utilizadas en la experimentación. Donde se indica el nombre, su descripción y la cantidad de atributos y objetos de cada una.

Tabla 5.1: Descripción de las instancias

Nombre de la instancia	Descripción	No. atributos	No. objetos
iris	Incluye tres especies de iris. Una especie de flor es linealmente separable de las otras dos, pero las otras dos no son linealmente separables entre sí.	6	150
tinto	Describe características relevantes sobre el vino tinto y vino blanco.	12	1599
cancer	Las características se calculan a partir de una imagen digitalizada de un aspirado con aguja fina (FNA) de una masa mamaria. Describen las características de los núcleos celulares presentes.	32	569
diabetes	Los conjuntos de datos consisten en varias variables predictoras médicas y una variable objetivo. Las variables predictoras incluyen el número de embarazos que ha tenido la paciente, su IMC, nivel de insulina, edad, etc.	9	768
glass	Este es un conjunto de datos de identificación de Glass de UCI. Contiene 10 atributos, incluida la identificación. La respuesta es tipo de vidrio (7 valores discretos).	10	214

5.2. Experimento 1. Comparación de los algoritmos EU-ACFL-GP_v1 (diversificación) en el algoritmo de referencia

5.2.1. Propósito del experimento

Realizar una comparación entre los algoritmos existentes midiendo la calidad de los predicados descubiertos a través del cuantificador de universalidad (forAll) en las instancias mostradas en la Tabla 5.1. Se muestra la configuración del algoritmo en la Tabla 5.2 aplicando las instancias.

5.2.2. Configuración experimental (Específico)

Tabla 5.2: Parámetros de configuración

Configuración	
Tamaño de la población	50
Porcentaje de cruza	95 %
Porcentaje de mutación	5 %
Iteraciones	30

5.2.3. Resultados y análisis

En las siguientes Tablas 5.3, 5.4 y 5.5 se muestra las 30 ejecuciones realizadas en cada uno de los algoritmos, en ellas se ve a grandes rasgos las diferencias entre los resultados, así como su promedio por cada instancia evaluada.

Tabla 5.3: Valor de verdad en el algoritmo de referencia

	conjuntos de instancias de clasificación				
No. de corrida	Cancer	Glass	Iris	Tinto	Diabetes
1	0.523840	0.775873	0.665038	0.704736	0.467525
2	0.649902	0.840690	0.743698	0.790549	0.546669
3	0.769418	0.905580	0.824974	0.859086	0.605588
4	0.860768	0.951358	0.855437	0.897511	0.645336
5	0.899727	0.971072	0.883727	0.939838	0.687443
6	0.918521	0.971002	0.895359	0.946431	0.708386
7	0.928784	0.960795	0.908029	0.974716	0.722338
8	0.951204	0.972886	0.916063	0.969845	0.729295
9	0.956050	0.982478	0.924801	0.983328	0.739488
10	0.955290	0.986799	0.928933	0.997351	0.752313
11	0.977382	0.992017	0.939652	0.989450	0.746184
12	0.977285	0.992539	0.921610	0.987647	0.751701
13	0.980171	0.986741	0.920178	0.981813	0.764505
14	0.975711	0.990354	0.923245	0.974865	0.766916
15	0.975619	0.992970	0.935195	0.984227	0.749097
16	0.968929	0.996267	0.938426	0.982561	0.766781
17	0.964333	0.996342	0.938114	0.988857	0.773939
18	0.966566	0.997476	0.927106	0.994874	0.761773
19	0.961404	0.990685	0.922437	0.985674	0.764278
20	0.967947	0.986352	0.935442	0.990768	0.766032
21	0.959430	0.991659	0.942345	0.986654	0.775688
22	0.972308	0.995625	0.938645	0.984233	0.770030
23	0.976712	0.995602	0.936230	0.984765	0.769298
24	0.961218	0.992013	0.940492	0.988927	0.775865
25	0.982665	0.989200	0.926967	0.984044	0.770277
26	0.988431	0.999817	0.907192	0.974648	0.767936
27	0.975524	0.990741	0.928009	0.976363	0.769500
28	0.988713	0.994992	0.927233	0.985959	0.766625
29	0.987499	0.994255	0.936861	0.989953	0.771277
30	0.988067	0.996037	0.939033	0.972419	0.769887
Promedio	**0.930314**	**0.972674**	**0.905682**	**0.958403**	**0.730732**

Tabla 5.4: Valor de verdad en el algoritmo EU-ACFL-GP

	Conjunto de instancias de clasificación				
No. de corrida	Cancer	Glass	Iris	Tinto	Diabetes
1	0.873651	0.997617	0.813628	0.919844	0.900343
2	0.873987	0.997617	0.831272	0.920747	0.903341
3	0.876492	0.999286	0.834067	0.921918	0.904777
4	0.877628	0.999403	0.850330	0.927582	0.914810
5	0.882652	1.000000	0.853176	0.930723	0.915477
6	0.884851	1.000000	0.853176	0.932594	0.933813
7	0.885400	1.000000	0.856316	0.935271	0.939660
8	0.894407	1.000000	0.862228	0.937940	0.944815
9	0.901533	1.000000	0.862228	0.946665	0.945242
10	0.906391	1.000000	0.863447	0.949159	0.949693
11	0.910214	1.000000	0.896812	0.952629	0.950899
12	0.910400	1.000000	0.905522	0.962328	0.956186
13	0.913596	1.000000	0.917654	0.969300	0.956233
14	0.929682	1.000000	0.928377	0.976348	0.956275
15	0.935691	1.000000	0.933769	0.980941	0.961353
16	0.936205	1.000000	0.933773	0.986976	0.962667
17	0.959072	1.000000	0.936379	0.995215	0.965139
18	0.970911	1.000000	0.940792	0.995562	0.969855
19	0.976895	1.000000	0.951028	0.996287	0.978898
20	0.983955	1.000000	0.978647	0.996488	0.987462
21	0.986384	1.000000	0.978820	0.996576	0.993160
22	0.994363	1.000000	0.987773	0.996808	0.998605
23	0.995907	1.000000	0.995568	0.996863	0.998875
24	0.996013	1.000000	0.995822	0.997714	0.999115
25	0.997669	1.000000	0.997431	0.999155	0.999299
26	0.998727	1.000000	0.999751	0.999431	0.999934
27	0.999859	1.000000	0.999751	0.999960	0.999943
28	0.999967	1.000000	0.999941	0.999980	0.999994
29	0.999994	1.000000	1.000000	1.000000	1.000000
30	1.000000	1.000000	1.000000	1.000000	1.000000
Promedio	**0.941750**	**0.999797**	**0.925249**	**0.970700**	**0.962862**

Tabla 5.5: Valor de verdad algoritmo EU-ACFL-GP (mutación constructiva)

	Conjuntos de instancias de clasificación				
No. de corrida	Cancer	Glass	Iris	Tinto	Diabetes
1	0.915558	1.000000	0.847086	0.887003	0.926297
2	0.921940	0.999996	0.850891	0.889582	0.926297
3	0.927973	0.999865	0.852122	0.891098	0.928476
4	0.943515	0.999764	0.858797	0.894091	0.928886
5	0.949447	1.000000	0.858797	0.897544	0.931326
6	0.950951	1.000000	0.870580	0.898757	0.933699
7	0.953056	1.000000	0.884463	0.904589	0.940760
8	0.953553	1.000000	0.884845	0.909247	0.945492
9	0.960346	1.000000	0.893772	0.914195	0.947127
10	0.963458	1.000000	0.921369	0.934991	0.965078
11	0.966517	1.000000	0.926952	0.935388	0.965078
12	0.977465	1.000000	0.935042	0.940635	0.969193
13	0.983081	1.000000	0.946011	0.946569	0.969360
14	0.986224	1.000000	0.947702	0.947871	0.976405
15	0.986463	1.000000	0.949303	0.950591	0.979006
16	0.992010	1.000000	0.956510	0.951140	0.981441
17	0.996881	1.000000	0.969471	0.951540	0.989348
18	0.997188	1.000000	0.971423	0.959709	0.990968
19	0.997630	1.000000	0.973496	0.960304	0.991621
20	0.997814	1.000000	0.980438	0.992598	0.992840
21	0.998945	1.000000	0.993083	0.992598	0.994620
22	0.999043	1.000000	0.993851	0.994129	0.995837
23	0.999928	1.000000	0.993851	0.995945	0.997625
24	0.999932	1.000000	0.994860	0.996165	0.999027
25	1.000000	1.000000	0.999745	0.997226	0.999848
26	1.000000	1.000000	0.999944	0.999315	1.000000
27	1.000000	1.000000	0.999997	0.999872	1.000000
28	1.000000	1.000000	1.000000	0.999924	1.000000
29	1.000000	1.000000	1.000000	1.000000	1.000000
30	1.000000	1.000000	1.000000	1.000000	1.000000
Promedio	**0.977297**	**0.999988**	**0.941813**	**0.951087**	**0.972189**

En la Tabla 5.6 se muestra una comparación de manera resumida de los promedios obtenidos de cada algoritmo.

Tabla 5.6: Promedio de los valores de verdad las instancias por algoritmo

Instancia	Algoritmo referencia	Algoritmo EU-ACFL-GP	Algoritmo EU-ACFL-GP (mutación constructiva)
glass.txt	0.972673883	0.999797397	**0.99998750**
diabetes.txt	0.730732325	0.962862082	**0.972188579**
cancer.txt	0.930313919	0.941749893	**0.977297275**
tinto.txt	0.958403019	**0.970700163**	0.951087229
iris.txt	0.905682407	0.925249268	**0.941813321**

Se observa que en la mayoría de las instancias los mejores resultados marcados en negrita son obtenidos a través del algoritmo EU-CFL-GP con la implementación de la mutación constructiva, cabe destacar que la experimentación se hizo bajo el concepto de la LDC.

5.3. Experimento 2. Comparación de los algoritmos EU-ACFL-GP_v2 (intensificación y diversificación) y EU-ACFL-GP_v1

5.3.1. Propósito del experimento

Comparar el algoritmo EU-ACFL-GP contra el algoritmo EU-ACFL-GP con la implementación del método MIE-BP mediante los predicados descubiertos a través de los valores de verdad obtenidos, el número de variables que aparecen en el predicado, el número de estructuras y el tiempo de procesamiento.

5.3.2. Configuración experimental (Específico)

Tabla 5.7: Parámetros de configuración

Configuración	
Tamaño de la población	50
Porcentaje de cruza	95 %
Porcentaje de mutación	5 %
Iteraciones	30

5.3.3. Resultados y análisis

De acuerdo como se muestra en el algoritmo del método MIE-BP propone la posibilidad de configurar múltiples asteriscos para la búsqueda de predicados.

Se establen dos casos para evaluar el comportamiento de los algoritmos:

- Caso 1: (IMP * *)
- Caso 2: (AND * * *)

Para ser evaluadas las experimentaciones se lleva a cabo una prueba de hipótesis Wilcoxon en donde se establecen las siguiente hipótesis:

Hipótesis:

- H0: No hay diferencias estadísticamente significativa entre los algoritmos EU-ACFL-GP y EU-ACFL-GP con MIE-BP
- H1: hay diferencias estadísticamente significativa entre los algoritmos EU-ACFL-GP y EU-ACFL-GP con MIE-BP

Con un nivel de significancia $\alpha = 0.05$

Tabla 5.8: Promedio de valor de verdad (Caso 1 y 2)

	Caso 1		Caso 2	
Instancia	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**
iris	0.971546245	0.975213332	0.986548887	0.985647224
wine	0.963211716	0.999999041	0.957323384	0.977526308
cancer	0.975440978	0.998284476	0.999852231	0.999999999
diabetes	0.965474545	0.985471264	0.975469877	0.984545774
glass	0.961457825	0.988888421	0.954784655	0.987468894

En la Tabla 5.8 se muestran las instancias evaluadas a través de dos algoritmos EU-CFL-GP y EU-CFL-GP con MIE-BP en donde los valores de verdad para el caso 1 son mayores para el algoritmo 2; en el caso 2 se muestra cierta similitud de valores de verdad para ambos algoritmos.

La prueba de hipótesis se realizó mediante el software STAC (Rodríguez-Fdez et al., 2015) para el caso uno indica que H0 es rechazada porque el valor de p-value obtenido es: 0.024023 mientras que en el caso dos la H0 es aceptada ya que el valor de p-value es de 0.13801.

Tabla 5.9: Promedio de Variables encontradas por algoritmo (Caso 1 y 2)

	Caso 1		Caso 2	
Instancia	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**
iris	4	3	4	5
wine	8	5	5	7
cancer	3	5	2	3
diabetes	6	5	14	17
glass	8	5	4	7

En la Tabla 5.9 se observan el promedio de las variables que fueron encontradas en los primeros 10 resultados de la última iteración de las instancias

La prueba de hipótesis arrojó para el caso uno indica que H0 es aceptada porque el valor de p-value obtenido es: 0.138010 esto indica que no existe diferencia, mientras que en el caso dos la H0 es rechazada ya que el valor de p-value es de 0.04122683.

Tabla 5.10: No. Estructuras encontradas en los predicados descubiertos (Caso 1 y 2)

	Caso 1		Caso 2	
Instancia	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**
iris	7	1	10	3
wine	6	7	6	5
cancer	8	10	10	12
diabetes	8	6	2	5
glass	3	5	4	6

En la Tabla 5.10 se observan el número de estructuras que fueron encontradas en los primeros 10 resultados de la última iteración de las instancias

La prueba de hipótesis arrojó para el caso uno indica que H0 es aceptada porque el valor de p-value obtenido es: 0.8907458 esto indica que no existe diferencia, igualmente para caso dos la H0 es aceptada ya que el valor de p-value es de 0.6844698

Tabla 5.11: Promedio de tiempo en minutos (Caso 1 y 2)

	Caso 1		Caso 2	
Instancia	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**	**EU-ACFL-GP**	**EU-ACFL-GP con MIE-BP**
iris	0.262225982	0.286133188	0.24683333	0.225411456
wine	4.33333	2.398316423	3.845086667	2.897319833
cancer	5.8974475	4.894342466	4.897456796	3.895475424
diabetes	1.469265532	0.883058467	1.44023451	1.419137567
glass	0.589754524	0.457622014	0.45798614	0.447745642

En la Tabla 5.11 se muestra el tiempo de procesamiento para los dos casos establecidos para los dos algoritmos. Se observa a simple vista que el algoritmo con el método incorporado logra mantener un tiempo debajo del algoritmo EU-ACFL-GP.

La prueba de hipótesis arrojó para el caso uno indica que H0 es aceptada porque el valor de p-value obtenido es: 0.079615 esto indica que no existe diferencia, sin embargo, en el caso dos la H0 es rechazada ya que el valor de p-value es de 0.043114446783075355

5.4. Experimento 3. Clasificación de predicados

5.4.1. Propósito del experimento

Medir la precisión de clasificación a partir del descubrimiento de predicados la lógica difusa arquimediana compensatoria.

5.4.2. Configuración experimental (Específico)

Tabla 5.12: Parámetros de configuración

Configuración	
Tamaño de la población	50
Porcentaje de cruza	95 %
Porcentaje de mutación	5 %
Iteraciones	30

5.4.3. Resultados y análisis

En primera instancia se realiza un descubrimiento de predicados para la clase diabético y no diabético obteniendo los siguientes dos resultados en la Tabla 5.13 que serán evaluados:

Tabla 5.13: Predicados descubiertos para la instancia diabetes

Valor de verdad	Predicado
0.966930208	$(EQV(AND"Blood\_Pressure""BMI""Glucose\_Concentration""Age")"diabetes")$
0.0.970397876	$(EQV(AND"Diabetes_Pedigree""Blood\_Pressure""BMI""Age")"no\_diabetes")$

Los parámetros de los estados lingüísticos para el predicado 1 son los siguientes:

Tabla 5.14: Parámetros de estados lingüísticos Diabético

Estado	**Beta**	**Gamma**	**m**
Blood_Pressure	73.5387	95.7273	0.6753
BMI	65.1591	65.6305,	0.7304
Glucose_Concentration	127.0959	128.04701,	0.5526
Age	60.73863	66.8899,	0.0882

Los parámetros de los estados lingüísticos para el predicado 2 son los siguientes:

Tabla 5.15: Parámetros de estados lingüísticos No Diabético

Estado	**Beta**	**Gamma**	**m**
Diabetes_Pedigree	0.9991	1.0259	0.1238
Blood_Pressure	54.7185	66.3199,	0.3429
BMI	66.3369	66.7637	0.5464
Age	65.6886	68.6883	0.8413

Representación gráficas de las variables lingüísticas Diabético

Un diabético se caracteriza por tener presión arterial media entre 73.5387-95.7273, un índice de masa corporal (BMI) medio entre 65.1591-65.6305, una concentración de glucosa media entre 127.0959-128.04701 y una edad creciente entre 60.73863 66.8899.

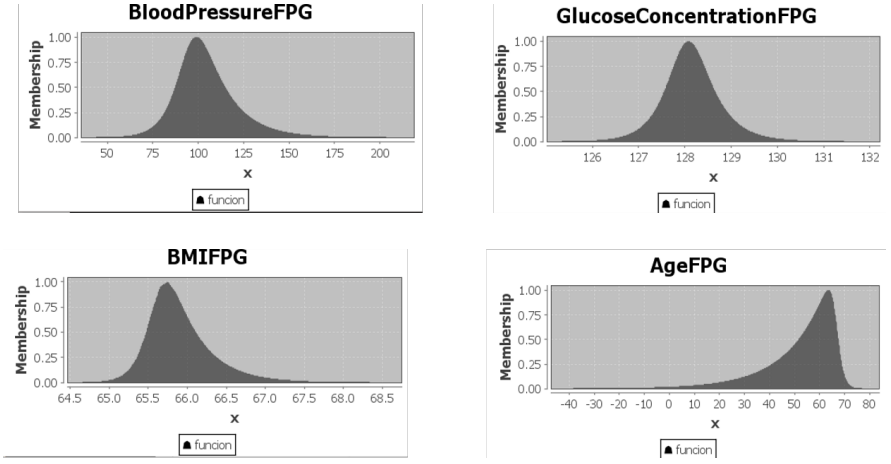

Figura 5.1: Funciones de pertenencia para las variables lingüísticas del predicado Diabetico

Representación gráficas de las variables lingüísticas NO Diabético

Una persona no diabética se caracteriza por tener presión media entre 73.5387-95.7273, un índice de masa corporal (BMI) medio entre 65.1591-65.6305, una concentración de glucosa media entre 127.0959-128.04701 y una edad creciente entre 60.73863 66.8899.

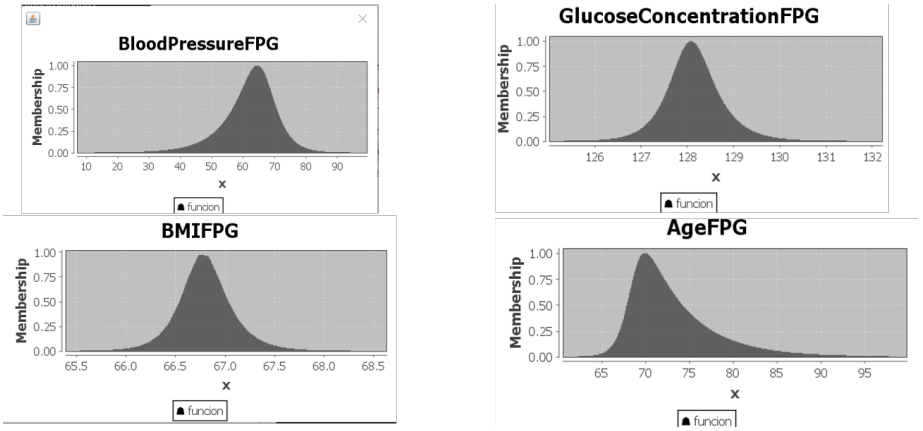

Figura 5.2: Funciones de pertenencia para las variables lingüísticas del predicado NO Diabetico

Evaluación de precisión

Se comparó el antecedente de cada predicado con respecto a su valor de verdad, si el resultado de diabético es mayor al del resultado de los registros de No diabético entonces se asigna 1, de lo contrario 0; para esta experimentación se obtuvo una precisión de 60 %.

Capítulo 6

Conclusiones y trabajo futuro

Este capítulo se describen las principales aportaciones de este proyecto. También se presentan las contribuciones y algunas posibilidades de trabajo futuro.

6.1. Conclusiones

Esta tesis forma parte del estudio de la lógica difusa compensatoria (LDC) y la lógica difusa arquimediana compensatoria (LDAC) ambas lógicas multivalentes capaces de abordar información imprecisa y vaga; ambas metodologías expresan el conocimiento a través de predicados difusos que funcionan con la conexión de operadores lógicos y variables lingüísticas asociadas a los conjuntos de datos.

Se realizó una revisión teoríca de elementos como los algoritmos genéticos, la programación genética para acoplar los árboles de decisión a la construcción y representación de predicados difusos usando los operadores lógicos como conectores y los estados lingüísticos como los nodos hoja.

Se llevó a cabo la mejora de conceptos de diversidad al implementar una mutación constructiva y en cuestión de exhaustividad para la generación y descubrimiento de predicados diversos; por otra parte se evalúa el tiempo de ejecución ya que forma parte de un sistema en apoyo a la toma de decisiones por tal motivo los tiempos de respuesta deben ser cortos para el usuario.

6.2. Contribuciones

Dentro de este proyecto de tesis se cumplieron los siguientes objetivos específicos:

1. Este documento de tesis describe la integración de los algoritmos genéticos EK-CFL y AG-GSF para contar con un algoritmo de descubrimiento de conocimiento de referencia (ver sección 4.2).

2. Incrementar la eficacia y eficiencia del algoritmo de referencia, en términos de veracidad de los predicados y tiempo de procesamiento respectivamente, usando estrategias de programación genética que permitan balancear la intensificación y diversificación de la búsqueda.

   De acuerdo con las experimentaciones obteniendo:

Los resultados de la experimentación muestran que el algoritmo EU-CFL-GP con la incorporación de la mutación constructiva mejora al algoritmo de referencia, ya que presenta una mejora del 8 % en el 80 % de las instancias y una mejora del 6 % con respecto a todas las instancias (ver sección 5).

3. Incrementar la exhaustividad de la búsqueda para encontrar predicados limitados por la definición actual de predicado simbólico (ver 5).

6.3. Difusión de la investigación

A partir del trabajo realizado se derivaron las siguientes participaciones en eventos y publicaciones:

Artículo en capítulos de libro:

- Edgar Ramiro Pedraza-Hernández, Citlalli Morales-de-la-Cruz, Laura Cruz-Reyes, Rafael Alejandro Espín-Andrade, Cesar Medina-Trejo, Claudia Gómez-Santillán. (2020). Estudio comparativo de métodos difusos aplicados a clasificación de datos. En *La investigación científica y tecnológica impulsando la creatividad para innovar.* (pp.536-546).Academia Mexicana Multidisciplinaria A.C

Participación en eventos:

- Exposición del trabajo .Estudio comparativo de métodos difusos aplicados a clasificación de datos"para el evento *Academia Mexicana Multidisciplinaria.* Agosto 2020.
- Exposición del trabajo "Genetic knowledge discovery algorithm based on CFL predicates"para el evento *International Virtual Workshop on Business Analytics Eureka 2021.* Junio 2021.

6.4. Trabajo futuro

1. Obtener diversidad en los predicados obtenidos utilizando estrategias parecidas a la búsqueda tabú, o utilizando funciones de optimización alternativas.
2. Precisar mejor el rol del algoritmo genético de búsqueda de las funciones de pertenencia, los parámetros de la configuración, y explorar otras formas de búsqueda y/o optimización para este paso de la búsqueda de los parámetros de las funciones de pertenencia.

Referencias

Affenzeller, M., Winkler, S., Wagner, S., y Beham, A. (2009, 03). Genetic algorithms and genetic programming.

Angeline, P. J. (1997). Subtree crossover: Building block engine or macromutation. *Genetic programming*, *97*, 9–17.

Asencios, V. V. (2004). Data mining y el descubrimiento del conocimiento. *Industrial Data*, *7*(2), 83–86.

Bhardwaj, A., Tiwari, A., Varma, M. V., y Krishna, M. R. (2014). Classification of eeg signals using a novel genetic programming approach. En *Proceedings of the companion publication of the 2014 annual conference on genetic and evolutionary computation* (pp. 1297–1304).

Bouchet, A., Pastore, J., Brun, M., y Ballarin, V. (2010). Logica difusa compensatoria basada en la media aritmética y su aplicación en la morfología matemática difusa. *Facultad de Ingeniería, Universidad Nacional de Mar del Plata, Argentina*.

Brachman, R., y Levesque, H. (2004). *Knowledge representation and reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Cejas-Montero, J. (2011). La lógica difusa compensatoria/the compensatory fuzzy logic. *Ingeniería Industrial*, *32*(2), 157–161.

Ceruto Cordovés, T., Rosete-Suárez, A., y Espin Andrade, R. (2010, 01). Obtención de predicados difusos a partir de datos utilizando metaheurísticas. *Editor invitado*, *1*, 29.

Coello, C. A. C. (2004). Introducción a la computación evolutiva. *México*.

Coyaso, F. J. R., y Vermonden, A. (2015). Lógica difusa para la toma de decisiones y la selección de personal. *Universidad & Empresa*, *17*(29), 239–256.

Darwin, C. (2016). *On the origin of species, 1859.* Routledge.

Das, S. K., Kumar, A., Das, B., y Burnwal, A. (2013). On soft computing techniques in various areas. *Computer Science & Information Technology (CS & IT)*, *3*, 59–68.

Deb, K., y Beyer, H.-G. (2001). Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary computation*, *9*(2), 197–221.

Developers, G. (2020). *Clasificación: Exactitud.* Descargado 2021-03-15, de https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=es-419

Dombi, J. (1990). Membership function as an evaluation. *Fuzzy sets and systems*, *35*(1), 1–21.

Duran, J., y Conesa, J. (2017). *Representació del conocimiento, febrero 2013.* Universitat Oberta de Catalunya.

Espín, R. A., Alfonso, D., y Cejas, J. (2012). Aplicación de la lógica difusa compensatoria en el sector empresarial. *Revista Dyna*, *87*(3), 3.

Espín, R. A., Bello, R., Cobo, A., Marx, J., y Racet, A. (2014). *Soft computing for business intelligence*. Springer.

Espín, R. A., González, E., y Caballero, E. (2014). Un sistema lógico para el razonamiento y la toma de decisiones: la lógica difusa compensatoria basada en la media geométrica. *Investigación operacional*, *32*(3), 230–245.

Espinilla, M. (2008). *Modelos linguíasticos difusos aplicados a la evalución sensorial.* Escuela Politécnica Superior de Jaén.

Eurekas. (2021). *Eurekas community.* Descargado 2021-03-15, de https://www.eurekascommunity.org/eureka-universe-eu/understanding-eu/guide

Gestal, M. (2013, 08). Introduccion a los algoritmos geneticos.

Gestal, M., Rivero, D., Rabuñal, J., Dorado, J., y Pazos, A. (2010). *Introducción a los algoritmos genéticos y la programación genética*. Universidade da Coruña.

Ghanea-Hercock, R. (2003). *Applied evolutionary algorithms in java*. Springer Science & Business Media.

Golberg, D. (1989). Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, *1989*(102), 36.

González-Caballero, E., Espín, R. A., Pedrycz, W., y Fernández, E. (2015). Archimedean-compensatory fuzzy logic systems. *International Journal of Computational Intelligence Systems*. doi: 10.1080/18756891.2015.1129591

González-Caballero, E., Espín, R. A., Pedrycz, W., Martínez, L., y Guerrero-Ramos, L. (2021, 02). Continuous linguistic variables and their applications to data mining and time series prediction. *International Journal of Fuzzy Systems*. doi: 10.1007/s40815-020-00968-w

Ho, T., Cheung, D., y Liu, H. (2005). Advances in knowledge discovery and data mining. En *9th pacific-asia conference, vietnam.*

Koza, J. R. (1990). *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems* (Vol. 34). Stanford University, Department of Computer Science Stanford, CA.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection* (Vol. 1). MIT press.

Koza, J. R. (2007). Introduction to genetic programming. En *Proceedings of the 9th annual conference companion on genetic and evolutionary computation* (pp. 3323–3365).

Llorente, C. (2019). *Métodos de representación, búsqueda e inferencia para analítica de datos mediante predicados: un enfoque neuro difuso basado en lógica difusa arquimediana compensatoria y aprendizaje profundo* (Tesis de Master no publicada). Instituto Tecnológico de Ciudad Madero (ITCM).

Martínez, C. G. (2008). *Algoritmos genéticos locales*. Universidad de Granada.

Meschino, G. J., Nabte, M., Gesualdo, S., Monjeau, A., y Passoni, L. I. (2014). Fuzzy tree studio: a tool for the design of the scorecard for the management of protected areas. En *Soft computing for business intelligence* (pp. 99–112). Springer.

Mora Espinosa, C. F., y Nieto Sánchez, J. C. (2019). *Lógica matemática*. Universidad Central.

NSF. (1987). *Uc irvine machine learning repository, center for machine learning and intelligent systems.* UC Irvine Machine Learning Repository. ([Web; accedido el 06-08-2020])

O'Reilly, U.-M., Yu, T., Riolo, R., y Worzel, B. (2006). *Genetic programming theory and practice ii* (Vol. 8). Springer Science & Business Media.

Osorio Roig, D., Lapeira Mena, O., Ceruto Cordovés, T., y Rosete Suárez, A. (2015). Mfuzzy-pred: Algoritmo multi-objetivo para extraer predicados difusos. *Revista Cubana de Ciencias Informáticas*, *9*, 61–75.

Padrón-Tristan, F. (2019). Use of compensatory fuzzy logic fot knowledge discoverey applied to the warehouse order picking problem for real-time order batching..

Padrón-Tristan, F. (2020). *Algoritmo de virtual savant basado en lógica difusa compensatoria para problemas de empacado de objetos* (Tesis de Master no publicada). Instituto Tecnológico de Ciudad Madero (ITCM).

Passoni, L., Meschino, G., Gesualdo, S., y Monjeau, A. (2021, 06). Fuzzy tree studio: Una herramienta para el diseÑo del tablero de mando para la gestiÓn de Áreas protegidas. En

(p. 10-11).

Poli, R., Langdon, W. B., McPhee, N. F., y Koza, J. R. (2008). *A field guide to genetic programming*. Lulu. com.

Riquelme Santos, J. C., Ruiz, R., y Gilbert, K. (2006). Minería de datos: Conceptos y tendencias. *Inteligencia Artificial: Revista Iberoamericana de Inteligencia Artificial, 10 (29), 11-18.*.

Rodríguez-Fdez, I., Canosa, A., Mucientes, M., y Bugarín, A. (2015). STAC: a web platform for the comparison of algorithms using statistical tests. En *Proceedings of the 2015 ieee international conference on fuzzy systems (fuzz-ieee).*

Ross, T. (2010). *Fuzzy logic with engineering applications*. John Wiley & Sons Ltd.

Sánchez, A. M., Lozano, M., y Herrera, F. (2007). Algoritmos genéticos para codificación real con operador de cruce híbrido con múltiples descendientes: 2blx0. 5-2fr0. 5-2pnx3-2sbx0. 01. En *Vi congreso español sobre metaheurísticas, algoritmos evolutivos y bioinspirados* (pp. 411–418).

Sharma, D., y Chandra, P. (2019). A comparative analysis of soft computing techniques in software fault prediction model development. *International Journal of Information Technology*, *11*(1), 37–46.

Vi, D., Detyniecki, M., Yager, D., Prade, R., y Grabisch, M. (2001, 06). Mathematical aggregation operators and their application to video querying.

Whitley, L. D. (1989). The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. En *Icga* (Vol. 89, pp. 116–123).

Yu, X., y Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer Science & Business Media.

Zadeh, L. A. (1975). The concept of a linguistic variable and its applications to approximate reasoning. *Information Sciente*.

Zadeh, L. A. (1988). Fuzzy logic. *Computer*, *21*(4), 83–93.

Anexo A

Estudio de la Lógica Difusa Arquimediana Compensatoria (ACFL) y la Variable Lingüística Continúa Generalizada (GCLV)

Lógica Difusa Arquimediana Compensatoria

Esta lógica consiste en una t-norma y una t-conorma Arquimediana generadas por una función, que sirven como generadores de operadores compensatorios. Además, incluyen el operador de negación más habitual y una orden difusa.

El cuantificador universal de Arquimediano y su cuantificador existencial correspondiente representan la tendencia a no refutar una toma de decisiones representada por un predicado, respectivamente. De manera equivalente para la Lógica Difusa Compensatoria (LDC), el par formado por un cuantificador universal LDC y un cuantificador existencial LDC representa la afirmación.

Propiedades de t-norma

Una t-norma es una función $T : [0,1]^2 \to [0,1]$, que satisface a los siguientes axiomas:

$T(a,b) = T(b,a)(T1)$	**Conmutatividad**
$T(a,b) \leq T(c,d)$	**Monotonicidad**
Si	
$a \leq c$	
$b \leq d(T2)$	
$T(a,T(b,c)) = T(T(a,,b),c)(T3)$	**Asociatividad**
$T(a,1) = a(T4)$	**Uno como identidad**

Propiedades de t-conorma

Una t-conorma es una función $S : [0,1]^2 \to [0,1]$, que satisface a los siguientes axiomas:

$S(a,b) = S(b,a)ST1)$	**Conmutatividad**
$S(a,b) \leq S(c,d)$	**Monotonicidad**
Si	
$a \leq c$	
$b \leq d(S2)$	
$S(a,S(b,c)) = S(S(a,,b),c)(S3)$	**Asociatividad**
$S(a,1) = a(S4)$	**Uno como identidad**

Hay una propiedad clásica que coincide con la t-noram y t-conorma, se refiere a las leyes de Morgan.

$\neg T(a,b) = S(\neg a, \neg b) y \neg S(a,b) = T(\neg a, \neg b)$ y el operador de negación usual es $\neg a = 1 - a$

Una t-norma $T : [0,1]^2 \to [0,1]$ es una t-norma continua si y unicamente si para cada par de secuencias convergentes $\{x_n\}_{n=1}^{\infty}$ y $\{x_n\}_{n=1}^{\infty}$

Propiedades Arquimedianas

1. Cada t-norma $T : [0,1]^2 \to [0,1]$, satisface $T(x,y) \leq min\{x,y\}$
2. Cada t-conorma $S : [0,1]^2 \to [0,1]$, satisface $S(x,y) \geq max\{x,y\}$
3. $min(x,y)$ es la unica t-norma continua idempotente y la máxima función de la familia de operadores t-norma.
4. $max(x,y)$ es la única t-conorma continua idempotente y la mínima función de la familia de operadores t-conorma.
5. Una consecuencia de las propiedades precedentes es que cada t-norma (t-conorma) pertenece a solo uno de dos subconjuntos, el singleton de la t-norma mínima (la t-conorma máxima) o el conjunto de normas.
6. Para cada t-norma Arquimediana continua hay una función continua no creciente $f : [0,1] \to [0+\infty]$ satisfaciendo $f(1) = 0$, tal que:

$$T(x_1, ..., x_n) = f^{(-1)}(\sum_{i=1}^{n} f(x_i)) \tag{6.1}$$

   Donde:

$$f^{(-1)}(z) = \begin{cases} f^{-1}(z) & \text{si } z \in \text{ [0,f(0)]} \\ 0 & \text{si } (f(0), +\infty) \end{cases} \tag{6.2}$$

7. Existe una propiedad equivalente para las t-conormas.

Sea n un operador de negación de $[0,1]$ a $[0,1]$ o un operador estrictamente decreciente que cumple $n(n(x) = x, n(0) = 1$ y $n(1) = 0$.

Lógica Arquimediana

Una lógica Arquimediana es un trío (c,d,n) de operadores, donde c es una t-norma Arquimediana continua, d es su correspondiente t-conorma Arquimediana continua y n es su operador de negación.

Se establece que $x = (x_1, x_2, ..., x_n)$, $y = (y_1, y_2, ...y_n)$, $z = (z_1, z_2, ..., z_n)$ son cualquier elemento del producto Cartesiano.

Lógica Difusa Compensatoria

Es un 4-tupla de operadores continuos (c,d,o,n), c y d desde $[0,1]^n$ entre $[0,1]$, el operador o de $[0,1]^2$ a $[0,1]$ y n un operador de negación, satisfaciendo los siguientes axiomas.

1. Axioma de compensación:
   $min\{x_1, x_2, ..., x_n\} \leq c\{x_1, x_2, ..., x_n\} \leq max\{x_1, x_2, ..., x_n\}$
2. Axioma de conmutatividad o simetría:
   $c\{x_1, x_2, ..., x_i, x_j, ...x_n\} = c\{x_1, x_2, ..., x_j, x_i, ...x_n\}$
3. Axioma de crecimiento estricto: Si $x_1 = y_1, x_2 = y_2, ..., x_{i-1} = y_{i-1}, x_{i+1} = y_{i+1}, ..., x_n = y_n$ son diferentes de 0 y $x_i > y_i$ entonces $c(x_1, x_2, ..., x_n) > c(y_1, y_2, ..., y_n)$
4. Axioma de veto: Si $x_i = 0$ para un i entonces $c(x) = 0$

5. Axioma de reciprocidad difusa $o(x,y) = n[o(y,x)]$
6. Axioma de transitividad difusa: Si $o(x,y) \leq 0.5$ y Si $o(x,y) \leq 0.5$ entonces $o(x,y) \leq max(o(x,y), o(y,x)$
7. Leyes de Morgan:
   $n(c(x_1, x_2, ..., x_n)) = d(n(x_1), n(x_2), ..., n(x_n))$ y $n(d(x_1, x_2, ..., x_n)) = c(n(x_1), n(x_2), ..., n(x_n))$

El operador d satisface las propiedades bajo:

1. Propiedad compensatoria:
   $min\{x_1, x_2, ..., x_n\} \leq d(x_1, x_2, ..., x_n\}) \leq max\{x_1, x_2, ..., x_n\}$
2. Propiedad de simetría:
   $dx_1, x_2, ..., x_i, ...x_j, ...x_n = dx_1, x_2, ..., x_j, ...x_i, x_n$
3. Propiedad de crecimiento estricto Sí $x_1 = y_1, y_2, ..., x_{i-1}, x_{i+1}, ..., x_n = y_n$ son diferentes de 1 y $x_i > y_i$ entonces $d(x_1, x_2, ..., X_n) > d(y_1, y_2, ..., y_n)$
4. Axioma de veto Si $x_i = 1$ para un i entonces $d(x) = 1$
5. $c(a, a, ..., a) = a, d(a, a, ..., a) = a$, idempotencia de los operadores c y d.

Los operadores c y d son llamados como conjunción y disyunción, respectivamente. El operador o es llamado orden estricto difuso.

Una familia de la LDC es la media cuasi aritmética, una de ellas por ejemplo la media geométrica. Los operadores son de la siguiente forma:

$$M_f = x_1, x_2, ..., x_n = f^{-1}(\frac{\sum_{i=1}^{n} f(x_i)}{n}) \tag{6.3}$$

f es una función estrictamente monótona continua, la cual extiende de puntos no definidos usando su límite correspondiente. Estos operadores satisfacen los axiomas 1-3. En la conjunción, si se tiene que para toda $i \in 1, 2, ..., n$, $lim_{x_i \to 0} M_f = x_1, x_2, ..., x_n$ =0, se obtiene el axioma 4, entonces se obtienen las ecuaciones 6.4, 6.5 y 6.6

$$d(x_1, x_2, ..., x_n) = 1 - f^{-1}(\frac{\sum_{i=1}^{n} f(1 - x_i)}{n}) \tag{6.4}$$

$$n(x) = 1 - x \tag{6.5}$$

$$o(x,y) = 0.5[c(x) - c(y)] + 0.5 \tag{6.6}$$

Que se refiere a la Lógica Compensatoria basada en la Medica Cuasi Aritmética

Operadores (expresados de manera general en término de dominio-contradominio) de la LDAC

Se dice que $L = (c_c, c_T, d_c, d_T, o, n)$ es una Lógica Difusa Compensatoria Arquimediana si cuenta con los siguientes operadores difusos:

1. $c_r : [0,1]^2 \rightarrow [0,1]$ es una t-norma Arquimediana que generada por una función f que satisface a al ecuación 1.

2. $d_r : [0,1]^2 \rightarrow [0,1]$ satisface $d_t(x_1, x_2) = 1 - c_t(1 - x_1, 1 - x_2)$ para cada $x = (x_1, x_2) \in [0,1]^2$

3. $c_c : [0,1]^n \rightarrow [0,1]$ satisface la ecuación 3 para f usada en el punto 1.

4. $d_c : [0,1]^n \rightarrow [0,1]$, donde para cada $x = (x_1, x_2, ..., x_n) \in [0,1]^n$, $d_c(x_1, x_2, ..., x_n) = 1 - c_c(1 - x_1, 1 - x_2, ..., 1 - x_n)$ y $c_c(x_1, x_2, ..., x_n) = 1 - d_c(1 - x_1, 1 - x_2, ..., 1 - x_n)$

5. $o : [0,1]^2 \rightarrow [0,1]$ es un orden difuso.

6. $n : [0,1] \rightarrow [0,1]$ es el operador de negación con $n(x) = 1 - x$

Tabla 6.1: Operadores del sistema de lógica probabilística y la GMBCL

Operador	Sistema de lógica probabilística	Sistema GMBCL
Conjunción	$c(x_1, x_2) = x_1 * x_2$	$c(x) = \sqrt[n]{\prod_{i=1}^{n} x_1}$
Disyunción (dual de la conjunción)	$d(x_1, x_2) = x_1 + x_2 - x_1 * x_2$	$d(x) = 1 - \sqrt[n]{\prod_i = 1^n (1 - x_i)}$
Negación	$1 - x$	1-x
Implicación $(d(n(x_1), x_2))$	Implicación de Reinchenbach $i(x_1, x_2) = 1 - x_2 + x_1 x_2$	Implicación Natural $i(x_1, x_2) = 1 - \sqrt{x_1 + x_1 x_2}$
Equivalencia $c(i(x_1, x_2), i(x_2, x_1))$	$i(x_1, x_2) * i(x_2, x_1)$	$\sqrt{i(x_1, x_2) * i(x_2, x_1)}$

Variable Lingüística Continúa Generalizada

Para poder explicar la forma en que se genera una Variable Lingüística Continúa Generalizada (GCLV), es necesario explicar las funciones pertenecientes a la misma.

De acuerdo con una lógica difusa compensatoria L y su generador de funciones f, se define un modificador lingüístico generalizado $m(x, L)$ mediante la siguiente ecuación (Espín et al.,en proceso):

$x_L^a = f^{-1}(af(x))$ donde $a \epsilon \mathbb{R}^+$

Sustituyendo $f(x)$ por $-g(ln(x))$ como el modificador lingüístico generalizado, puede llamarse a g función generadora secundaria de L. Además puede notar se que si $a = 1$ y $f(x) = 0$ y g es una función impar entonces $x_L^a = f^{(-1)}(0) = e^{-g^{-1(0)}}$.

Las funciones que se usan como estos modificadores los observamos en la Tabla 1 donde $n \epsilon \mathbb{N}$: También hay que observar que, x_L^a generaliza la ecuación:

$C_T = \underbrace{(x, x, .., x)}$ n veces $= f^{(-1)} \underbrace{(f(x) + f(x) + ... + f(x))}$ n veces $= f^{(-1)}(nf(x)), donde\, n \epsilon \mathbb{N}$

Además, si L es una ACLF y g es su función generadora secundaria, se obtiene una función sigmoidal generalizada.

Tabla 6.2: Funciones generadoras

f	f^{-1}
$f(x) = -ln(x)$	$f^{-1}(x) = e^{-x}$
$f(x) = -ln^n(x)$	$f^{-1}(x) = e^{-\sqrt[n]{x}}$
$f(x) = -arcsinh(ln(x))$	$f^{-1}(x) = e^{-sinh(x)}$
$f(x) = -log_n(x)$	$f^{-1}(x) = e^{-ln(n)x}$

$$S_g(x) = \frac{1}{1 + e^{-g(-1)(x)}} \tag{6.7}$$

Y de acuerdo con lo anterior se define una función sigmoidal generalizada parametrizada donde $\gamma \epsilon \mathbb{R}, \alpha > 0$ son los parámetros.

$$S_g(x; \alpha, \gamma) = \frac{1}{1 + e^{-g^{(-1)}(\alpha(x-\gamma))}} \tag{6.8}$$

Una vez visto lo anterior, una GCLV con los parámetros $\gamma \epsilon \mathbb{R}, \alpha > 0$ y $m \epsilon$ [0,1] generados por la función generadora secundaria g de una lógica L Arquimediana-Compensatoria, se define de la siguiente manera (Espín et al., en proceso):

$$GCLV_L(x; \alpha, \gamma, m) = \frac{C_T(S_g(x; \alpha, \gamma)^m_L, (1 - S_g(x; \alpha, \gamma))^{1-m}_L)}{M} \tag{6.9}$$

Donde C_T es una t-norma arquimediana continua en L y M es el máximo de $C_T(S_g(x; \alpha, \gamma)L^m, (1 - S_g(x; \alpha, \gamma))L^{1-m})$ en $\mathbb{R}$. $S_g(x; \alpha, \gamma)L^m$ y $(1 - S_g(x; \alpha, \gamma))L^{1-m}$ son funciones lingüísticas generalizadas sobre $S_g(x; \alpha, \gamma)$ y $1 - S_g(x; \alpha, \gamma)$.
Además si L es una ACFL y g su función generadora secundaria,, entonces la t-norma generada por g es la siguiente ecuación:

$$C_T(x_1, x_2) = e^{-g^{(-1)}(-g(ln(1-x_1))-g(ln(1-x_2)))} \tag{6.10}$$

De acuerdo con esta ecuación y a una lógica L podemos obtener lo siguiente:

$$d_T(x_1, x_2) = 1 - e^{-g^{(-1)}(-g(ln(1-x_1))-g(ln(1-x_2)))} \tag{6.11}$$

$$c_C(x_1, x_2, ..., x_n) = e^{-g^{(-1)}\left(\frac{-\sum_{i=1}^n g(ln(x_i))}{n}\right)} \tag{6.12}$$

$$d_C(x_1, x_2, ..., x_n) = 1 - e^{-g^{(-1)}\left(\frac{-\sum_{i=1}^n g(ln(1-x_i))}{n}\right)} \tag{6.13}$$

Otra de las ecuaciones que es importante mencionar es el cuantificador de universalidad que para un predicado p de una lógica L se calcula de la siguiente forma:

$$\forall = f^{-1}\left(\frac{\sum_{i=1}^n f(p_i)}{n}\right) \tag{6.14}$$

Anexo B

En esta sección se presentan las funciones de pertenencia:

Sigmoidal o sigmoidea

Está definida por sus límites inferior a, superior b y el valor m o punto de inflexión, tales que $a < m < b$. El crecimiento es más lento cuanto mayor sea la distancia. $(a - b)$ Para el caso concreto de $m = (a + b)/2$, que es lo usual, se obtiene la siguiente gráfica (ver Figura 6.1), con su correspondiente función de pertenencia:

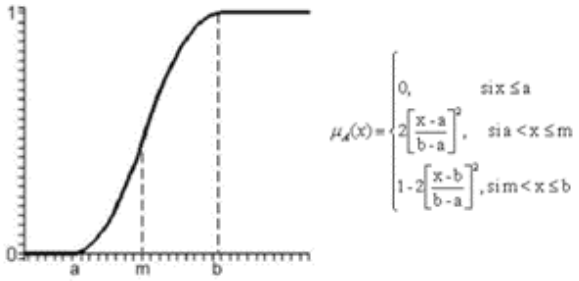

Figura 6.1: Gráfica de una función de pertenencia sigmoidal.

Trapezoidal

Definida por sus límites inferior a, superior d, y los límites de soporte inferior b y superior c, tal que $a < b < c < d$, cuya representación y función de pertenencia pueden generalizarse de la siguiente manera (ver Figura 6.2).

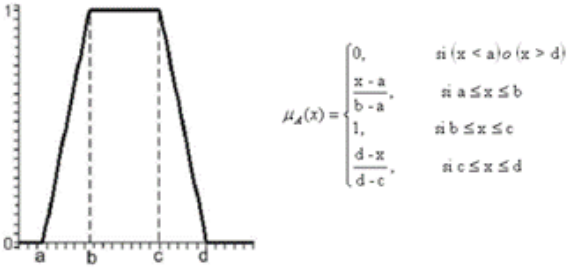

Figura 6.2: Gráfica de una función de pertenencia sigmoidal.

Gaussiana

Está es definida a partir del valor medio m y el valor parámetro k>0. Esta es la típica función de campana de gauss y cuanto mayor es el valor de k más estrecha es la campana (ver figura 6.3).

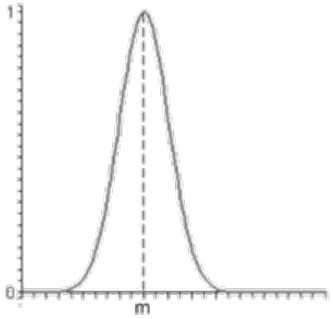

Figura 6.3: Gráfica de una función de pertenencia sigmoidal.

Función de pertenencia generalizada (FPG)

En lo relacionado con funciones de pertenencia a lo largo de la historia el estudio de estas ha sido limitado en comparación con otros temas existentes en conjuntos difusos, existe en si una descripción de un gran número de funciones, sin embargo, en el desarrollo de herramientas difusas, no se considera la función más adecuada para una aplicación dada (González-Caballero, Espín, Pedrycz, Martínez, y Guerrero-Ramos, 2021).

Drakopolus y su teorema de burbuja sigmoidal forma la base para aproximar cada función de membresía por sigmoides.

(González-Caballero et al., 2021) define una familia parametrizada de funciones de membresía la cual depende de tres parámetros $\alpha > 0$, $\gamma \in R$ y se fija a $m_0 > 0$.

Algunas de las ventajas mencionadas son:

- Contiene funciones de al menos tres tipos de formas, una es una función de membresía estrictamente creciente, otra es una función estrictamente decreciente y la tercera es una convexa que aumenta estrictamente en su primera parte y disminuye estrictamente en su segunda parte. Estos diferentes tipos de formas permiten representar diferentes valores lingüísticos.
- Los miembros de los FPG, son modificados por etiquetas lingüísticas. Esta propiedad aumenta la expresividad.
- La FPG aprovecha las posibilidades como aproximador universal propio de la función sigmoidal. Cada función de membresía continua puede ser aproximada por los miembros de la familia de funciones.
- Sus parámetros tienen un significado como se presenta en la teoría de Dombi. Aunque es menos evidente y difícil de calcular.
- Es posible que las FPG, se cumplan las condiciones sugeridas por Valente de Oliveira para garantizar la semántica.
- La forma de calcular una función sigmoidal que se crea con los parámetros $\alpha > 0$, $\gamma \in R$ es la siguiente:

$$Sigm(x; \alpha, \gamma) = \frac{1}{1 + e^{-g^{(-1)}(\alpha(x-\gamma))}} \tag{6.15}$$

Donde:

$$\alpha = \frac{log(0.99) - log(0.01)}{(\gamma - \alpha)} \tag{6.16}$$

Y una función de pertenencia generalizada es definida como sigue:

$$FPG(x; \alpha, \gamma, m) = \frac{Sigm(x; \alpha, \gamma)^m \times (1 - Sigm(x; \alpha, \gamma)^{1-m})}{M} \tag{6.17}$$

Donde:

$$M = m^m \times (1 - m)^{1-m} \tag{6.18}$$

Por medio de esta, se toma un conjunto de datos difusos y se aproxima variando los valores de α, β y m.

En la Figura 6.4 se puede observar, como el conjunto difuso de nombre altura, es mostrada en un grado de alto en la línea punteada (Sigmoidal positiva), medio en la línea estrellada (Gaussiana) y bajo en la línea solida (Sigmoidal negativa).

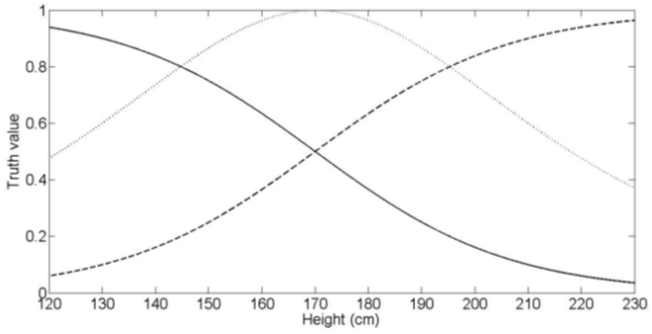

Figura 6.4: Gráfica de una función de pertenencia sigmoidal.