TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLOGICO DE CIUDAD MADERO

División de Estudios de Posgrado e Investigación
Maestría en Ciencias de la Computación

TESIS

ALGORITMO EVOLUTIVO PARA DESCUBRIR CONOCIMIENTO DE ASOCIACIÓN USANDO LÓGICA DIFUSA COMPENSATORIA.

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

Presenta
Carlos Eric Llorente Peralta
G17073003

Director de Tesis:
Dr. Laura Cruz Reyes

Co- Director de Tesis:
Dr. Rafael Espín Andrade

Cd. Madero, Tamaulipas. Junio 2019

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Ciudad Madero

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Cd. Madero, Tams., a 20 de Mayo de 2019

OFICIO No.: U5.071/19
ÁREA: DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS.

ING. CARLOS ERIC LLORENTE PERALTA
No. DE CONTROL G17073003
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 Maestro en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ALGORITMO EVOLUTIVO PARA DESCUBRIR CONOCIMIENTO DE ASOCIACIÓN USANDO LÓGICA DIFUSA COMPENSATORIA "

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE :	DR.	HÉCTOR JOAQUÍN FRAIRE HUACUJA
SECRETARIO:	DR.	JUAN FRAUSTO SOLÍS
VOCAL:	DRA.	LAURA CRUZ REYES
SUPLENTE:	DRA.	MARIA LUCILA MORALES RODRIGUEZ
DIRECTORA DE TESIS :	DRA.	LAURA CRUZ REYES

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"®

DR. JOSÉ AARÓN MELO BANDA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

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

c.c.p.- Archivo
Minuta

JAMB 'JAMF 'jar

Declaración de originalidad

Declaro y prometo que este documento de tesis es producto original de mi trabajo y que no infringe los derechos de terceros, tales como derecho de publicación, derechos de autor, patentes y similares.

Además, 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.

Además, en caso de infracción a los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad de la infracción 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.

Agradecimientos

TABLA DE CONTENIDO

Capítulo 1 Introducción

El uso de lógica como un medio para lograr encontrar sentido a los aspectos que ocurren en la vida en general a existido desde hace mucho tiempo, si no desde el principio del razonamiento humano. Diversas personalidades de la antigüedad han hecho uso de esta metodología con el fin de dar sentido a fenómenos observables.

En la actualidad se hace uso de metodologías lógicas en el entorno computacional, con el fin de desarrollar mecanismos de solución de problemas apoyados en este tipo de procesos lógicos.

Una de estas metodologías es la propuesta por el doctor Lotfi Zadhe la lógica difusa (LD), que propone el uso de una lógica multivaluada en el cual existe una transición gradual en el paso del concepto de cierto a falso, esto nos permite flexibilidad en los resultados.

Posteriormente mediante el uso de la LD el doctor Rafael Espin, propone el uso de una LD en la cual las características puedan compensarse entre si dando paso a una lógica difusa compensatoria (LDC)

En la presente tesis, se aborda el problema de la generación de conocimiento mediante el uso de la metodología de LDC, la cual es mayormente desarrollada por el Dr. Rafael Espín Andrade y su grupo Eureka.

Principalmente se propone un algoritmo evolutivo de LDC que permite el descubrimiento de patrones en conjuntos de datos. Para verificar la calidad de la respuesta se realizan estudios comparativos con otras metodologías y otras aplicaciones que hagan uso de la lógica difusa y la lógica difusa compensatoria.

1.1 Antecedentes

Este proyecto forma parte de un proyecto mayor realizado por investigadores de tres universidades: la Universidad Autónoma de Nuevo León (UANL), la Universidad Autónoma de Chihuahua-Unidad Torres (UAdeC) y el Instituto Tecnológico de Ciudad Madero (ITCM).

La meta central, del proyecto rector es desarrollar un sistema de apoyo a la toma de decisión que incluya como parte de su núcleo los diferentes productos de investigación propuestos por los investigadores. Los trabajos de investigación que han sido realizados incluyen:

a) Un framework para el análisis, diseño y evaluación de algoritmos (Cruz-Reyes. 2016).
b) Un conjunto de algoritmos de optimización competitivos (Cruz-Reyes. 2017).
c) Un sistema de recomendación de acciones eficientes y satisfactorias (Cruz-Reyes. 2015).

El presente trabajo, busca contribuir al crecimiento del núcleo con un algoritmo de descubrimiento de conocimiento basado en lógica difusa compensatoria.

1.2 Descripción del problema

En esta tesis se busca generar conocimiento con base en predicados léxicos, que permitan encontrar altos grados de pertenencia en el concepto analizado. Como es posible observar en el anexo, este problema, que se enuncia a continuación, es al menos tan complejo como los de clase NP-Duro.

Dado:

Un conjunto de datos *D* formado por *i* objetos con *j* atributos, se busca generar conocimiento expresado en un conjunto de predicados *P* tal que se maximice el valor de verdad de cada predicado *p* evaluado, mediante operadores de lógica difusa compensatoria.

Se busca:

$$p = c_{x \in U}\, p(x)$$

Donde:

La construcción del predicado *p*, se llevará a cabo usando la notación Backus-Naur Form (BNF).

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

Operación: = <Operación unaria> | (<Operación_unaria>) | <Operación_binaria> |
(<Operación_binaria>) | <Operación_enearia> | (<Operación_enearia>)

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

Operación_binaria: = <Operando_binario> <Operando> <Operando> |
(<Operando_binario> <operando> <Operando>)

Operación_enearia: = <Operando_eneario> <{Operando}> | (<Operando_eneario>
<{Operando}>)

Operando_binario: = '**IMP'** | **'EQV'**

Operando_eneario: = **'AND'** | **'OR'**

Operando: = **'FPG(j)'** | <Predicado>

Donde:

$$FPG(j) = FPG_T(x_{ij}; \alpha_j, \gamma_j, m_j, m0_j) = T\left(sigm^{m_j}(x_{ij}; \alpha_j, \gamma_j)\left(1 - sigm(x_{ij}; \alpha_j, \gamma_j)\right)^{m0_j - m_j}\right)$$

$$sigm(x_{ij}; \alpha_j, \gamma_j) = \frac{1}{1 + e^{-\alpha_j(x_{ij} - \gamma_j)}}$$

$$\alpha = \frac{\ln(0.99) - \ln(0.01)}{x_{ij} - \gamma_j}$$

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

truevalue(*p*) = grado de pertenencia obtenido al evaluar el predicado *p*.

FPG(*j*) = Función de pertenencia generalizada para cada atributo *j*.

α, ϒ, m, m_0 = Variables que optimizan la Función de pertenencia generalizada (FPG).

1.3 objetivo de la tesis

Objetivo general

Desarrollar un algoritmo evolutivo para el descubrimiento de conocimiento expresado en predicados basado en lógica difusa compensatoria.

Objetivos específicos

1) Desarrollar un algoritmo evolutivo que maximice la veracidad de los predicados.
2) Desarrollar un algoritmo evolutivo competitivo en términos del grado de veracidad de los predicados encontrados
3) Desarrollar un sistema para el descubrimiento de conocimiento, que incorpore como núcleo central el algoritmo evolutivo propuesto.

1.4 Justificación

a) Una alternativa recientemente propuesta para modelar el conocimiento de expertos en forma de predicados es la lógica difusa compensatoria (LDC).
b) De acuerdo con la literatura, mediante el uso de estos predicados de LDC es posible realizar tareas de evaluación, descubrimiento e inferencia de conocimiento, así como otras tareas de aprendizaje automático.
c) Los sistemas de LDC que hasta ahora conocemos no cuentan con un reporte de desempeño, su usabilidad es escasa y su explotación en el manejo de la información para el descubrimiento de conocimiento aún es muy limitada.

d) Para contribuir a la solución de problemas complejos de descubrimiento de conocimiento se propone un algoritmo evolutivo basado en LDC.

1.5 Organización del documento

Capítulo 2 Marco Teórico

En esta sección se detallan los conceptos necesarios para llevar a cabo el proceso de desarrollo de las actividades descritas en esta tesis, conceptos tales como la minería de datos (DM), la lógica difusa compensatoria (LDC), la programación genética (GP), entre otras.

De esta forma se provee de la comprensión necesaria al lector para la correcta interpretación de los procesos detallados en secciones posteriores, ya que la construcción del algoritmo genético elaborado esta basado en estos conceptos.

2.1 Descubrimiento de conocimiento en bases de datos

De acuerdo con lo encontrado en la literatura, el *descubrimiento de conocimiento en bases de datos* (KDD por sus siglas en inglés) es un proceso automatizado donde se combina las operaciones de descubrimiento y análisis. El proceso de extraer conocimiento en grandes volúmenes de datos es un tópico de investigación el cual es clave en los sistemas de base de datos y en la industria; es un área importante y presenta la oportunidad de obtener mayores ganancias (Timaran, 2009).

Algunos autores definen al KDD como “El proceso no trivial de identificación de patrones válidos, novedosos, potencialmente útiles y fundamentalmente entendibles al usuario a partir de los datos” (Timaran et al, 2016). Para llevar a cabo el análisis de los datos, se aplican algoritmos que bajo limitaciones de eficiencia computacional aceptable producen un grupo particular de patrones y/o modelos sobre los datos (Fayyad et al, 1996).

Las áreas de aplicación del KDD son diversas, es posible utilizarlo para descubrir procesos fraudulentos dentro de una compañía, el descubrimiento de relaciones entre enfermedad y sintomatología, la relación entre las causas y sus efectos. Las ventajas que presenta el KDD es el desarrollo de procesos más eficaces debido al tiempo que se reduce al corregir errores, y que la información anticipada que produce, permite reducir las sorpresas y analizar las causas de los efectos obtenidos. Es también de esta manera que se mejoran los procesos que se tienen establecidos, optimizándolos y reduciendo los errores encontrados (Morgado et al,2016).

El proceso de KDD es interactivo e iterativo en involucra al usuario en la toma de decisiones a través de un gran número de pasos (Timaran, 2016), los cuales son (Hernández et al, 2004):

1. **Integración y recopilación de los datos:** En esta parte se determinan las bases de datos a utilizar en el proceso y de donde se obtendrán.
2. **Selección, limpieza y transformación:** Se detectan los errores, información faltante o perdida, construir nuevos atributos y numerar y/o discretizar los atributos.
3. ***Minería de datos*:** En el proceso de minería de datos (DM por sus siglas en ingles), se produce nuevo conocimiento, para lograrlo se construye un modelo que describe los patrones y relaciones entre los datos que pueden usarse para hacer predicciones, para entender mejor los datos y explicar las situaciones pasadas. Esta sección incluye:

    - Elegir el tipo de modelo.
    - Determinar el tipo de tarea de minería más apropiado.
    - Seleccionar el algoritmo de minería que resuelva la tarea y obtenga el modelo que se está buscando.

4. **Evaluación e interpretación:** Se presentan diferentes medidas de evaluación de los modelos, los cuales deben ser precisos, comprensibles e interesantes.
5. **Difusión:** Una vez que el modelo ha sido construido y validado, puede ser utilizado en una variedad de finalidades y deben ser evaluados los resultados obtenidos, de esta manera observamos si debe ser reevaluado, reentrenado o reconstruido completamente.

Es visto que, en DM, los problemas y aplicaciones están normalmente rodeados de factores organizacionales y sociales en las diferentes empresas. Estos factores integran el entorno de DM en los procesos de la vida real y es necesario estudiarlos detenidamente, debido a que estos reflejan las necesidades de las empresas y usuarios, además estos constituyen el entorno de los conocimientos identificados. El conocimiento que resulta de la investigación de los factores y recursos que conforman la organización en los patrones de minería constituye la inteligencia organizacional (Cao L et al, 2010).

En la figura 2-1 la cual es presentada en (Timaran, 2016) se describe el proceso de KDD, que, para el presente trabajo de tesis la de mayor relevancia es la de minería de datos.

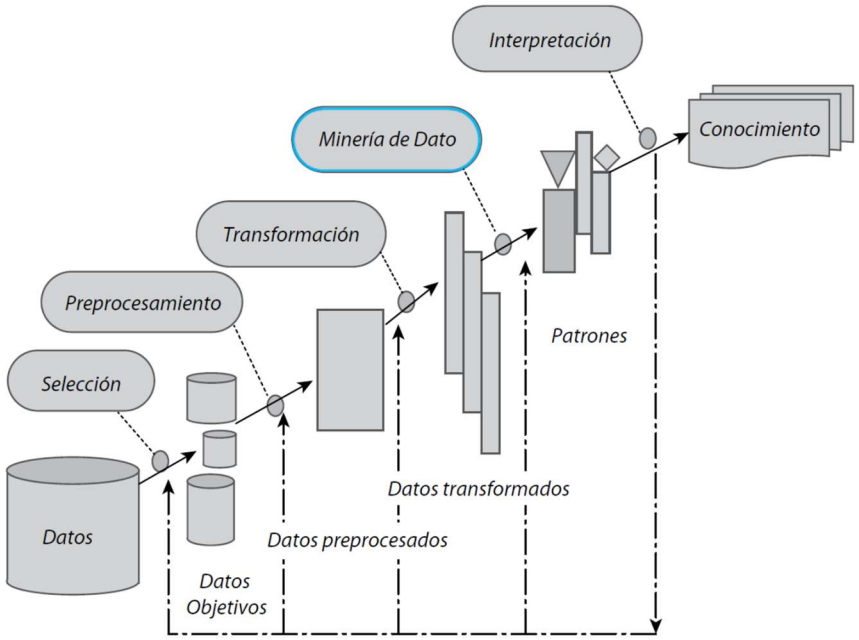

Figura 2.1 *Descripción del proceso de KDD. Timaran et al (2016)*

Existen una serie de elementos que deben tomarse en cuenta para que una organización pueda implementar con éxito el proceso de minería de datos, este conjunto de aspectos más que elementos técnicos, representan cambios organizativos, de mentalidad y culturales dentro de la organización. Steninhoff et al (2012), definen estos aspectos:

a) La aceptación de la responsabilidad para evitar los problemas de forma proactiva mediante la adopción de enfoques y herramientas de minería de datos. Esto es esencialmente una transformación cultural de la organización.
b) Comprender el potencial apoyo que representa la minería de datos a la organización tanto en la gestión diaria como en la toma de decisiones estratégicas.
c) Determinar cómo se va a utilizar el conocimiento resultante del proceso de minería de datos.
d) Compartir experiencias y mejores prácticas.

e) Ver a KDD como un proceso continuo.

2.2 Minería de datos

Dentro de esta etapa, se realiza la búsqueda y descubrimiento de patrones que no son descriptibles a simple vista y que tienen un alto grado de interés, debido a su usabilidad en los procesos y objetivos de la empresa (Quinlan, 1986) (Wang, Iyer y Scott, 1998).

Si bien la minería de datos es parte del proceso de KDD, en gran parte de la literatura son tratados como si fueran lo mismo. De manera concreta, sé describe que el termino minería de datos es manejado por estadísticos, analistas de datos y administradores de sistemas informáticos, como todo el proceso de descubrimiento de descubrimiento, mientras el termino KDD es acuñado principalmente por especialistas en inteligencia artificial (Molina et al, 2004).

Para llevar a cabo el proceso de DM, se hace uso del apoyo de tecnologías que permiten mejorar el desempeño y los resultados obtenidos, dentro de las cuales las más importantes son los métodos estadísticos y el aprendizaje automático, los métodos estadísticos han producido varios paquetes estadísticos que se han ido integrando con las bases de datos a explorar. El aprendizaje automático consiste en obtener reglas de aprendizaje y modelos de los datos, para lo cual es necesaria la ayuda de la estadística (Molina et al, 2004).

Además de las mencionadas se hace uso también de tecnologías como las de visualización, procesamiento paralelo y apoyo a la toma de decisiones. Las técnicas de visualización ayudan a la presentación de los datos lo cual facilita el DM, el procesamiento paralelo ayuda a mejorar el rendimiento y los sistemas de apoyo a la toma de decisiones ayuda a discriminar resultados y proporcionar los resultados esenciales para llevar a cabo las funciones de dirección (Molina et al, 2004).

Dependiendo de si existe una variable objetivo, el proceso de aprendizaje puede clasificarse de la siguiente forma (Versellis, 2009):

Aprendizaje supervisado (predictivo): En este tipo de modelos se hace uso de variables objetivo, variables dependientes o clases, así como también se hace uso de variables denominadas independientes o predictivas, mediante las cuales

se generan patrones que permiten hacer una estimación de valores futuros o desconocidos (Timaran, 2016).

Aprendizaje no supervisado (Descriptivo): En este tipo de metodología, los datos de entrada son analizados de tal manera que se puedan identificar patrones que expliquen o resuman los datos, es decir, son usados con el fin de examinar las propiedades de los datos no para predecir nuevos datos (Timaran, 2016).

Existen también dentro del proceso de DM una serie de tareas las cuales observamos en la figura 2-2.

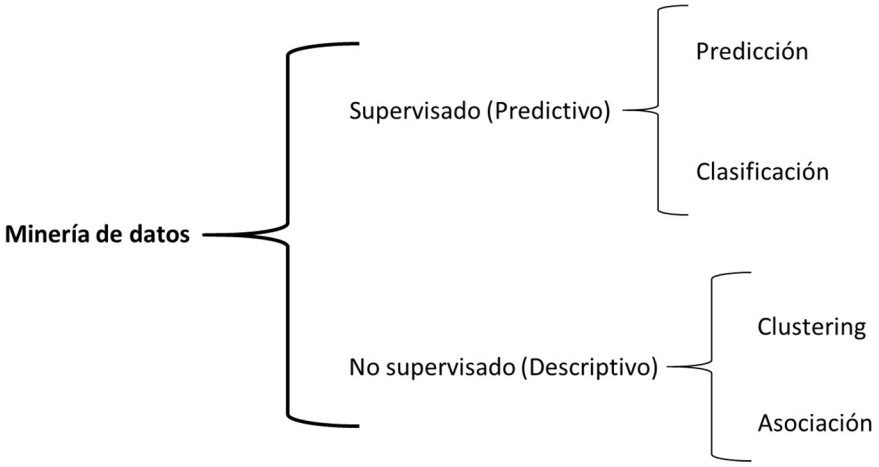

Figura 2.2 Cuadro descriptivo de minería de datos (DM) (Molina et al 2004)

Predicción

En esta tarea se intenta determinar los valores de una o varias variables a partir de un conjunto de datos, la predicción de valores continuos puede realizarse mediante técnicas estadísticas de regresión.
Es posible resolver gran cantidad de problemas mediante el uso de la regresión lineal y pueden resolverse aún más aplicando transformaciones a las variables para que un problema no lineal pueda convertirse a uno lineal (Molina et al, 2004).

Clasificación

Se realiza mediante dos pasos, en el primero se selecciona un atributo que determinara la pertenencia de un registro a una clase, este atributo es llamado atributo clase, el conjunto de registros que permite la construcción del modelo

es llamado conjunto de entrenamiento y se escoge aleatoriamente del total de registros pertenecientes a la base de datos. En el siguiente paso se usan los resultados obtenidos para clasificar un conjunto de registros, el cual es llamado conjunto de prueba y se seleccionan aleatoriamente de la base de datos, estos registros no deben pertenecer al conjunto de prueba inicial. Se estima la efectividad del modelo mediante el porcentaje de aciertos conseguidos al clasificar el conjunto de prueba (Han y Kamber, 2001).

Agrupación

La agrupación (en inglés clustering), También llamada segmentación se define como el proceso de agrupar objetos físicos o abstractos en clases de objetos similares, es decir, el clustering agrupa un conjunto de datos mediante el principio de maximizar la similitudes intraclase y minimizar las similitudes interclase, de esta manera se construyen particiones significativas de un conjunto de objetos basados en la metodología de divide y conquista en la cual se descompone un sistema de gran escala en pequeños componentes para simplificar el diseño y la implementación (Timaran, 2016).

La meta del clustering en una base de datos es la partición de esta en segmentos o clústeres de registros similares que comparten un numero de propiedades y son considerados homogéneos.

Asociación

Esta tarea descubre patrones en forma de reglas, en los cuales se observan los hechos que ocurren con frecuencia en una base de datos (Timaran, 2016).

Son usadas cuando el objetivo es realizar análisis exploratorio, las reglas descubiertas son utilizadas para predecir comportamientos y permiten descubrir correlaciones y coocurrencias de eventos. Para llevar a cabo su implementación es necesario contar con un archivo de datos histórico sobre el cual llevar a cabo este proceso, por lo general este método de extracción se fundamenta en técnicas estadísticas.

2.3 Lógica difusa compensatoria

La lógica es una de las disciplinas más antiguas de la historia humana, ha habido grandes personajes que han hecho estudios sobre la misma. Entre ellos Leibniz, Boole, Russell, Turing, y muchos otros; a la fecha aún es una disciplina sobre la cual se siguen realizando investigaciones.

En la actualidad se realiza más investigación usando la lógica como base de los procesos computacionales, algunos de los objetivos de estos estudios han sido probar teoremas matemáticos, validar diseños de ingeniería, diagnosticar fallas, codificar y analizar leyes, regulaciones y reglas comerciales. El uso de la metodología lógica por parte de la ingeniería de software es llamada programación lógica.

En esta tesis se hace uso de un método lógico creado por el grupo científico multidisciplinario Gestión Empresarial en la Incertidumbre: Investigación y Servicios del Instituto Superior Politécnico "José Antonio Echeverría", Cujae, en La Habana, Cuba el cual es liderado por el Dr. Rafael Espín Andrade (Cejas, 2011), Éste método llamado lógica difusa compensatoria (LDC), constituye una rama de la Lógica Difusa (LD).

La LD es una disciplina propuesta en los años sesenta por Lotfi Zadeh, cuando se dio cuenta de lo que él llamo el principio de incompatibilidad "*conforme la complejidad de un sistema aumenta, nuestra capacidad para ser precisos y construir instrucciones sobre su comportamiento disminuye hasta el umbral más allá del cual, la precisión y el significado son características excluyentes*" (Pérez P., 2005). En la lógica difusa se hace uso de los conceptos de la lógica y de los *conjuntos de Lukasiewicz* (gobierna los principios de implicación y equivalencia en una lógica trivaluada) mediante la definición de grados de pertenencia (Cejas, 2011).

La LD parte de la idea de que la forma en que se construye el pensamiento humano no es mediante números, si no sobre *etiquetas lingüísticas*. De esta manera se representa el conocimiento común, que es mayoritariamente del tipo lingüístico cualitativo y no necesariamente cuantitativo, mediante un lenguaje matemático a través de conjuntos difusos y funciones características asociadas a ellos. Los términos lingüísticos son inherentemente menos precisos que los datos numéricos, pero expresan el conocimiento en términos más asequibles para la comprensión humana (Pérez P., 2005).

La LD tiene la capacidad de reproducir aceptablemente los modos usuales del razonamiento, considerando que la certeza de una proposición es cuestión de grado, donde la lógica clásica es considerado el caso limite. Por lo cual las características más atractivas de la lógica difusa son su flexibilidad, su tolerancia con la imprecisión, su capacidad para modelar problemas no lineales y su base en el lenguaje natural (Pérez P, 2005).

En las lógicas multivalentes los valores de verdad se corresponden con el intervalo [0,1], la LDC es un nuevo sistema multivalente que rompe con la axiomática tradicional de este tipo de sistemas para lograr un comportamiento semánticamente mejor a los sistemas clásicos (Delgado. 2005).

En LDC se renuncia al cumplimiento de las propiedades clásicas de la conjunción y la disyunción, contraponiendo a éstas la idea de que el aumento o disminución del valor de verdad de la conjunción o la disyunción provocadas por el cambio del valor de verdad de una de sus componentes, puede ser "compensado" con la correspondiente disminución o aumento de la otra (Cejas, 2012). Un crecimiento o decrecimiento en el valor de verdad de la conjunción o disyunción como resultado de un cambio en el valor de verdad de alguna componente, puede ser compensado por el crecimiento o decrecimiento en otra componente. Esta noción hace que la LDC sea una lógica sensible. Existen casos en los que la compensación no es posible. Esto ocurre cuando son violados ciertos umbrales y existe un veto que impide la compensación (Cejas, 2012).

La LDC está formada por una cuarteta de operadores continuos que satisfacen el grupo de axiomas que se muestran en las ecuaciones 1 al 7: conjunción (*c*), disyunción (*d*), orden estricto difuso (*o*) y negación (*n*) (Espín et al, 2011).

1. **Axioma de compensación:**

   $$min(x_1, x_2, \ldots, x_n) \leq c(x_1, x_2, \ldots, x_n) \leq max(x_1, x_2, \ldots, x_n)$$

2. **Axioma de conmutatividad o simetría:**

   $$c(x_1, x_2, \ldots, x_n) = c(x_1, x_2, \ldots, x_n)$$

3. **Axioma de crecimiento estricto:**

Sí $x_1 = y_1, x_2 = y_2, \ldots, x_{i-1} = y_{i-1}, x_{i+1} = y_{i+1}, \ldots, x_n = y_n$ son diferentes de cero, y $x_i > y_i$, entonces, $c(x_1, x_2, \ldots, x_n) > c(x_1, x_2, \ldots, x_n)$

4. **Axioma de veto:**

   Si $x_i = 0$ para algún 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 $o(y, z) \geq 0.5$ entonces $o(x, y) \geq max(o(x, y), o(y, z))$

7. **Leyes de De Morgan:**

$$n(c(x_1, x_2, \ldots, x_n)) = d(n(x_1), n(x_2), \ldots, n(x_n))$$

$$n(d(x_1, x_2, \ldots, x_n)) = c(n(x_1), n(x_2), \ldots, n(x_n))$$

Donde:

o, d, c, n: Se encuentran definidas arriba.

x: es una variable lingüística existente en un conjunto.

i: es la *i_esima* variable.

j: Es la *j_esima* variable.

n: es la última variable del conjunto.

El Axioma de Compensación: Es mediante este axioma, que a una lógica se le otorga el nombre a este tipo de lógica. Se menciona que, para el caso particular de dos componentes, el hecho de que el valor del operador se encuentre entre el mínimo y el máximo, puede interpretarse como que el segundo valor compensa el valor del primero en la veracidad de la conjunción. La idea se generaliza al caso de *n* componentes (Espín et al, 2011).

El Axioma de Conmutatividad o Simetría: Es una propiedad esperada debido a que es natural que el resultado sea independiente del orden en que se tomen los predicados básicos (Espín et al, 2011).

El Axioma de Crecimiento Estricto: En este axioma se dota al sistema de una sensibilidad que hace que cualquier variación en los valores de los predicados básicos modifique el valor de verdad del predicado compuesto, siempre que ninguno de los predicados básicos tenga valor cero (Espín et al, 2011). Como consecuencia de este axioma se tiene además la deseada propiedad de no asociatividad porque no existen operadores compensatorios asociativos, estrictamente crecientes (Dubois y Prade. 1985).

El Axioma de Veto: Alude a su interpretación en el marco de los problemas de decisión; esta propiedad otorga a cualquier predicado básico de una conjunción la capacidad de vetar, es decir la capacidad de impedir cualquier forma de compensación cuando su valor es igual a cero (Espín et al, 2011).

De acuerdo con los axiomas 5 y 6 descritos anteriormente, un orden estricto es un predicado *o: U^2*= [0,1] si se cumplen las dos condiciones siguientes:

a) o (x, y) = n [o (y, x)] (reciprocidad difusa)
b) Si o (x, y) ≥ 0.5 y o (y, z) ≥0.5, entonces o (x, z) ≥máx (o (x, y), o (y, z)) (Transitividad difusa máx-máx.)

La propiedad de transitividad difusa max-max implica en presencia de la reciprocidad difusa la satisfacción de un grupo de propiedades deseables que aportan mayor significación al orden estricto (Espín et al, 2011).

De acuerdo con la definición de los incisos a y b, la función $o(x,y) = 0.5[c(x) - c(y)] + 0.5$ con $n(x) = 1 - x$, es un orden estricto sobre el universo del predicado c. De esta manera, mediante el predicado $o(x,y)$ se evalúa "cuánto mejor es x que y " como c mide la conveniencia de las alternativas x , y para el decisor. Si $o(x,y) = 0.5$, entonces x,y se considerarían indiferentes. Es un instrumento para establecer una relación entre las preferencias del decisor y las veracidades atribuidas a su conocimiento (Espín et al, 2011).

De acuerdo con Espín (2012) una implicación preferida es aquella que parte de definiciones que utilizan combinaciones de los operadores de conjunción, disyunción y negación, para explorar el efecto que esta clase de operadores tienen en el comportamiento de la implicación. De esta manera, la implicación puede ser definida en el caso general como $i_1(x,y) = d(n(x),y)$ o $i_2(x,y) =$

$d\big(n(x), c(x,y)\big)$, de ese modo se generalizan las tablas de verdad de la lógica booleana de dos maneras diferentes.

La equivalencia es consecuentemente definida a partir del operador implicación como $e(x,y) = c\big(i(x,y), i(y,x)\big)$.

Así también se hace uso de los cuantificadores universal y existencial que son definidos a partir de los operadores conjunción y disyunción. Se tiene que cualquiera sea un predicado difuso *p* sobre el universo *U* las proposiciones universal y existencial se definen respectivamente como (Espín et al, 2012):

$$\forall_{x \in U}\, p(x) = \; c_{x \in U}\, p(x) \qquad \text{(Ec. 2.1)}$$

$$\exists_{x \in U}\, p(x) = \; d_{x \in U}\, p(x) \qquad \text{(Ec. 2.2)}$$

Las leyes de De Morgan: Son propiedades esenciales en el comportamiento, que relacionan de una manera natural y universalmente aceptada los operadores conjunción y disyunción.

Los operadores que satisfacen el Axioma 1 se llaman Operadores Compensatorios. Los operadores compensatorios simétricos encontrados en la literatura son los siguientes (Espín et al, 2011):

1) **Operadores de máximo y mínimo**; el primero no satisface los axiomas de Crecimiento Estricto y de Veto; el operador mínimo no satisface el Axioma de Crecimiento Estricto.

2) **Los estadísticos de k-orden**, que incluyen el operador mediano, no satisfacen los Axiomas de Crecimiento Estricto y de Veto.

3) **Las combinaciones de norma y co-norma**, como son los operadores compensatorios exponenciales (incluyendo el llamado Operador de Zimmerman) y los operadores de convexidad lineal, no satisfacen el Axioma de Veto.

4) **La media aritmética** no satisface el Axioma de Veto.

5) **Las medias cuasi – aritméticas**, incluyen, por ejemplo, a la media geométrica. Estos son operadores de la forma $M_f(x_1, x_2, \dots, x_n) = f^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f(x_i)\right)$, donde f es una función continua estrictamente monótona que se extiende, con el uso de los límites correspondientes, a puntos donde no está definida.

2.3.1 Modelación de la vaguedad con etiquetas lingüísticas

En la LDC la modelización de la vaguedad se lleva a cabo a través de *etiquetas lingüísticas*, para representar términos difusos que se encuentran en el lenguaje natural. Con las variables lingüísticas se toma el conocimiento de expertos y se aplica al problema de interés. En *métodos* basados exclusivamente en datos la vaguedad se maneja como una caja negra, como por ejemplo las redes neuronales.

Dichas variables lingüísticas obtienen su valor en escalas proporcionadas por el tomador de decisiones, por ejemplo, la definida en la Tabla 2-1:

Valor de Verdad	Categoría
0	Falso
0,1	casi falso
0,2	bastante falso
0,3	algo falso
0,4	más falso que verdadero
0,5	tan verdadero como falso
0,6	más verdadero que falso
0,7	algo verdadero
0,8	bastante verdadero
0,9	casi verdadero
1	Verdadero

Tabla 2.1 Tabla de ejemplo del modelado de la vaguedad.

2.3.2 Conjuntos difusos

Un conjunto de manera formal puede ser definido como. Si *X* es un conjunto de objetos y *x* es un objeto genérico del conjunto *X,* el set clásico *A*, $A \subset X$, es

definido como una colección de elementos u objetos $x \in X$, dado que cada objeto *x* puede estar o no estar cerca del conjunto *A.* Por definición, la función característica para cada elemento $x \in X$, es posible expresar el conjunto A mediante un conjunto ordenado de pares $(x, 0)\ o\ (x, 1)$ lo cual indica que $x \notin A\ o\ x \in A$ respectivamente (Jang et al, 1997).

En la lógica difusa compensatoria, sin embargo, la función característica por medio de la cual construimos el conjunto difuso toma valores en el rango [0,1], por lo cual se define a un conjunto difuso de la siguiente manera.

Si *X* es una colección de objetos representados genéricamente por *x*, entonces un conjunto difuso *A* en *X* es definido como un conjunto de pares ordenados.

$$A = \{(x, \mu_A(x)) | \ x \in X\}$$

Donde $\mu_A(x)$ es llamada la función de pertenencia (MF por sus siglas en inglés) del conjunto difuso *A*. La MF asigna a cada elemento de *X* un grado de pertenencia entre 0 y 1(Jang et al, 1997).

Para ilustrar el concepto de los conjuntos difusos Zadeh propuso el ejemplo del conjunto de "los hombres altos". Según la teoría de la lógica clásica, al conjunto de hombres altos solo pertenecen los que miden más de una determinada altura y esa altura límite es 1.80 metros, así un hombre es considerado alto cuando mide por ejemplo 1.81 metros y uno bajo cuando mide 1.79 metros. Esto no parece una razón muy lógica para catalogar a un hombre de alto o bajo ya que por ejemplo en el caso expuesto la altura de uno a otro solo se diferencia en 2 centímetros (Pérez P., 2005).

El evaluar si un hombre es alto o bajo, se hace mediante una función que define la transición entre alto a bajo y para ello asigna a las distintas alturas un valor entre 0 y 1. Según sea este valor se considera que se pertenece al conjunto o no.

Aplicando esto al caso anterior, un hombre que mida 1.79 metros se puede decir que pertenece al conjunto de hombres altos con un grado de 0.75 y el hombre que medía 1.81 metros pertenece al conjunto de hombres altos con un grado de 0.8 (Pérez P.,2005).

Si se representa lo anteriormente mencionado en una gráfica se obtendrá que la transición entre alto o bajo con la lógica difusa es una curva con cambios no

abruptos mientras que con la lógica clásica (ver Figura 2-3), el paso de alto a bajo o viceversa es brusco.

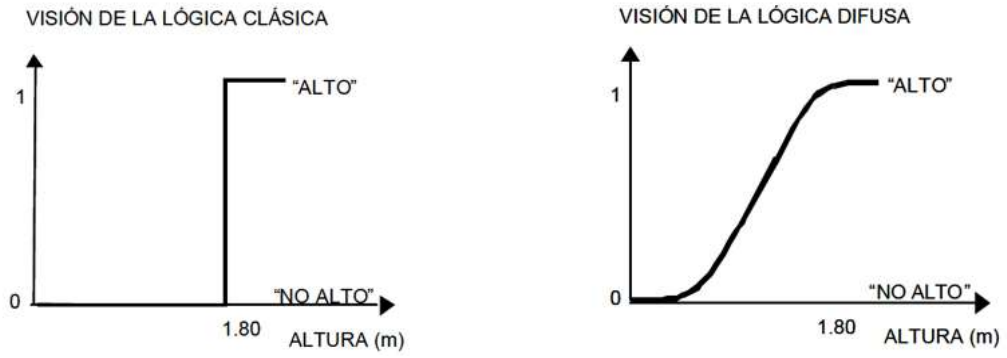

Figura 2.3 Grafica que ejemplifica la diferencia entre lógica clásica y lógica difusa.

El ejemplo anteriormente dado, pertenece a un conjunto difuso continuo, sin embargo, también se pueden aplicar a conjuntos difusos discretos, tal como en el siguiente ejemplo.

Si X = {Tsuru, Chevy, Mazda 3, Sentra, Civic} es el conjunto de autos los cuales una persona puede comprar, el conjunto difuso A = "Auto deseable para comprar" se puede describir de la siguiente manera:

A = {(Tsuru, 0.2) (Chevy, 0.5) (Mazda 3, 0.8) (Sentra, 0.6) (Civic, 0.4)}

En la figura 2-4 se muestra un ejemplo de una función de pertenencia de un conjunto difuso discreto.

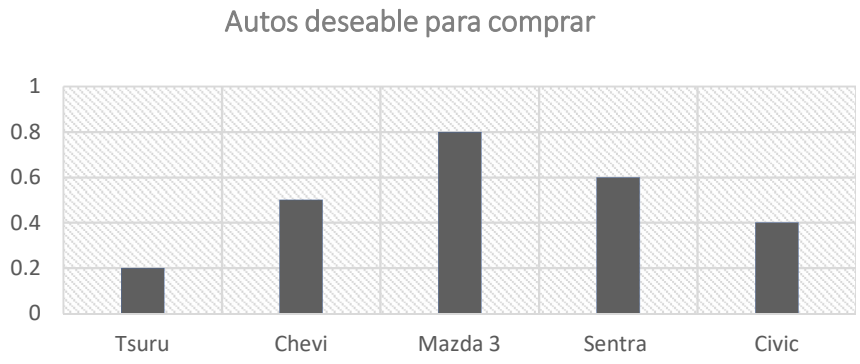

Figura 2.4 Ejemplo de un conjunto discreto.

En definición un conjunto difuso es una característica cualitativa o cuantitativa de algún objeto, a la cual se le otorga un valor de pertenencia en el intervalo [0,1] mediante una función en la cual se calcula el grado de pertenencia o grado de verdad del atributo, siendo llamada esta acción como *normalización lógica multivalente* (NLM) (Espín et al. 2011).

2.3.3 Funciones de pertenencia

Las funciones de pertenencia se nombran usualmente como μ(X) o M(X). Hay ciertas funciones típicas que siempre se suelen usar, tanto por la facilidad de computación que su uso conlleva, como por su estructura lógica para definir su valor lingüístico asociado. Dichas funciones de pertenencia pueden tener diferentes estructuras: rectas, triangulares, sigmoidales, en Z, gaussianas, entre otras (Ceruto et al. 2010,2014)

En esta sección se mencionan algunas de las funciones de pertenencia mencionadas anteriormente.

Sigmoidal o sigmoidea: 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 2-5), con su correspondiente función de pertenencia:

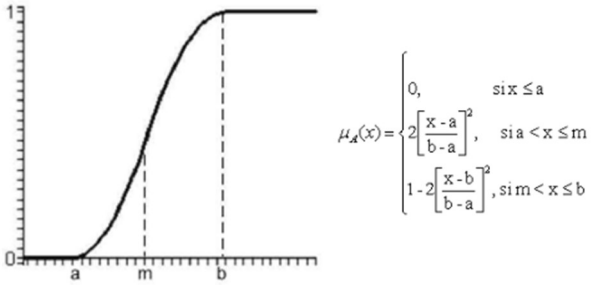

Figura 2-5 Grafica de una función de pertenencia sigmoidal.

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 2-6).

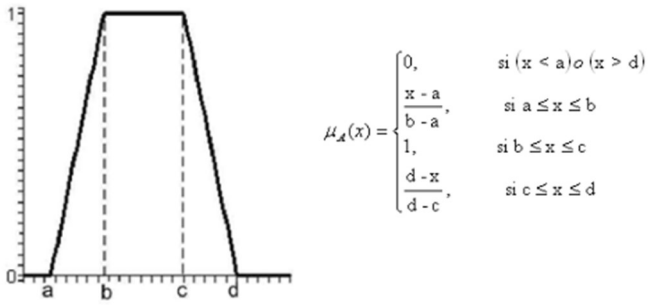

Figura 2-6 Grafica de una función de pertenencia trapezoidal

Gaussiana: esta 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* mas estrecha es la campana (ver figura 2-7).

$$\mu_A(x) = e^{-k(x-m)^2}$$

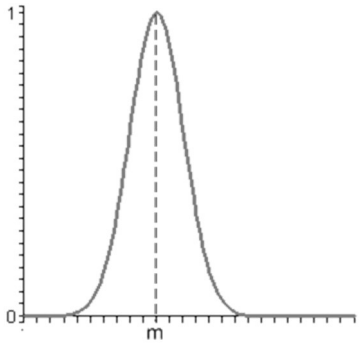

Figura 2-7 Grafica de una función de pertenencia gaussiana

2.3.4 Función de pertenencia generalizada

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 et al, 2018).

Drakopolus y su teorema de burbuja sigmoidal forma la base para aproximar cada función de membresía por sigmoides (González et al, 2018).

En González et al (2018), se define una familia parametrizada de funciones de membresía la cual depende de tres parámetros $\propto > 0, \gamma \in \mathbb{R}$ y se fija a $m_0 > 0$.

Algunas de las ventajas mencionadas son (González et al, 2018):

a) 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.
b) Los miembros de los FPG,s son modificados por etiquetas lingüísticas. Esta propiedad aumenta la expresividad.
c) 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.
d) Sus parámetros tienen un significado como se presenta en la teoría de Dombi. Aunque es menos evidente y difícil de calcular.
e) Es posible que las FPG,s 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 $\alpha > 0$, $\gamma \epsilon \mathbb{R}$ es la siguiente (González et al, 2018):

$$Sigm(x: \alpha, \gamma) = \frac{1}{1 + e^{-\alpha (x-\gamma)}}$$

Donde:

$$\propto = \frac{\log(0.99) - \log(0.01)}{\gamma - \beta}$$

Y una función de pertenencia generalizada es definida como sigue (González et al 2018):

$$FPG(x{:}\,\alpha,\gamma,m) = \frac{Sigm(x{:}\,\alpha,\gamma)^m * (1 - Sigm(x{:}\,\alpha,\gamma)^{1-m}}{M}$$

Donde:

$$M = m^m * (1-m)^{1-m}$$

Por medio de esta, se toma un conjunto de datos difusos y se aproxima variando los valores de $\alpha, \beta\ y\ m$ (González et al, 2018).

En la Figura 2.8 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).

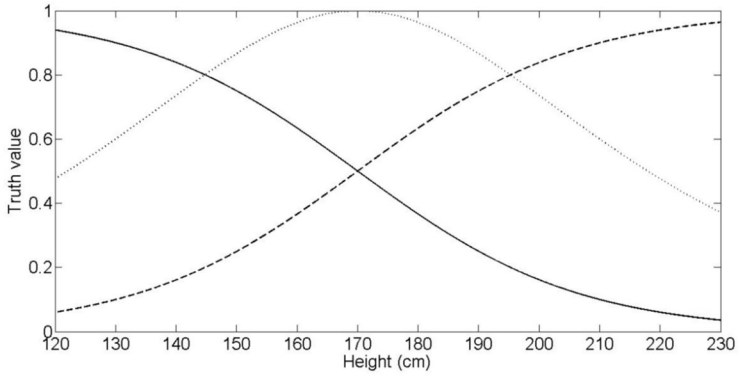

Figura 2.8 familia de funciones de pertenencia formadas a partir de una FPG.

2.3.5 Lógica de predicados

El instrumento fundamental de comunicación humana es el lenguaje, formado por frases de tipo interrogativo, imperativo y declarativo. Estas últimas presentan en muchos casos un grado de veracidad.

En muchos de los casos estas declaraciones son expresadas en forma de proposiciones las cuales son expresiones que pueden ser evaluadas como verdaderas o falsas en lenguajes naturales. Las proposiciones solo pueden ser expresiones declarativas.

Existen las proposiciones simples o también pueden ser llamados predicados simples en las cuales no es posible encontrar otras proposiciones, además las proposiciones compuestas o predicados compuestos son aquellas están conformadas de varias preposiciones simples y conectores lógicos.

Para ejemplificar lo que son los predicados simples se usa el atributo de altura en un grupo de hombres, a este atributo le asignamos una etiqueta lingüística tal como hombre_alto, de tal manera la hombre_alto(x) se determina como un predicado simple o átomo, si tomando una serie de atributos y asignándosele etiquetas lingüísticas y uniendo estas a partir de un conector lógico que en este caso es conjunción (^) se genera el ejemplo del predicado compuesto hombre saludable come_sano(x) ^ ejercicio_regular(x) ^ duerme_bien(x).

Una expresión que representa una proposición bien escrita se llama formula bien formada (FBF). Por otra parte, una fórmula es una tautología si es verdadera para todas sus interpretaciones, también es llamada formula valida, por otro lado, una fórmula es una contradicción si es falsa para todas sus interpretaciones. También llamada inconsistente (Frausto et al, 2000).

Dentro de las principales estructuras que podemos encontrar en la lógica de predicados están los predicados en forma normal conjuntiva (FNC) **v**(*v1*, *v2*, *v3*) y los predicados en forma normal disyuntiva (FND) ^(*v1*, *v2*, *v3*), en el cual en las estructuras de FNC el principal operador lógico es el operador de conjunción (^) y para las FND es el operador de disyunción (v) (ver figura 2.9).

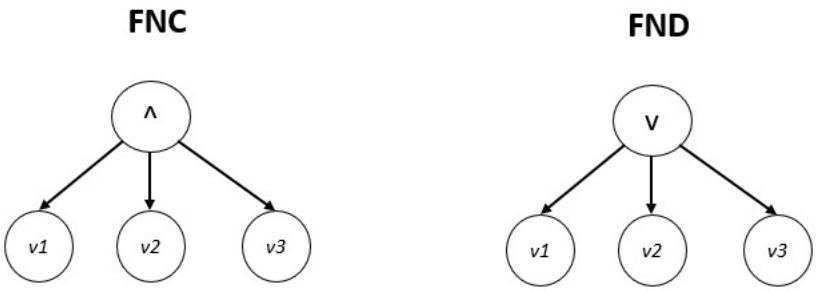

Figura 2.9 Representación de predicados lógicos en FNC y FND.

En el cálculo de predicados es posible indicar postulados sobre objetos individuales, se pueden indicar postulados relacionando varios objetos, se usan cuantificadores existenciales y universales.

Metodología funcional: realizar una tabla de verdad de todas las opciones mencionadas.

Metodología formal: utilizar una teoría formal que permita mediante un esquema axiomático, realizar deducciones.

Todas las lógicas proposicionales tienen el mismo problema de que la evaluación funcional tiene complejidad exponencial (Frausto et al, 2000).

Para el caso del cálculo de predicados la situación es aún peor, ya que el número de interpretaciones depende no solo del número *n* de símbolos de predicados, sino también de todos los posibles valores de todas las variables que intervengan puesto que el número de interpretaciones de las variables es en general infinito, no existe una manera de determinar todas las posibles interpretaciones de las fórmulas que intervienen en ninguna lógica de predicados (Frausto et al, 2000).

Para determinar ello la LD puede aportar mucho pues la vaguedad y la incertidumbre son los objetos de su modelado. Una propiedad esencial de esta lógica es el *principio de gradualidad* el cual afirma que una proposición puede ser verdadera y falsa a la vez, siempre que se le asigne un grado de verdad y de falsedad (Ceruto et al, 2010).

Precisamente la lógica de predicados estudia las frases declarativas con un grado de detalle, considerando la estructura interna de las proposiciones. Está basada en la idea de que las sentencias realmente expresan relaciones entre objetos, así como también cualidades y atributos de tales objetos.

En esencia un predicado es una función del universo X en el intervalo [0,1] y las operaciones de conjunción (^), disyunción (v), negación (¬), implicación (→) y equivalencia (↔), se definen en el dominio [0,1].

Ellos, junto con otros operadores, garantizan la combinación efectiva de elementos intangibles valorados a través de expertos considerando escalas categoriales de veracidad, con información cuantitativa, que aporta valores de verdad a través de predicados definidos convenientemente a partir de tal información.

Los predicados se pueden representar de diferentes formas, una de ellas es mediante árboles. Por ejemplo, un predicado se puede representar utilizando un árbol general (para evitar asociatividad) donde cada nodo puede ser un operador.

2.3.6 la lógica difusa y la modelación de la decisión

El uso de una diversidad de operadores con propiedades que generalizan la Lógica Bivalente, pareciera ser la manera natural de modelar problemas de decisión a partir del lenguaje. Sin embargo, esta manera de abordar las decisiones no proporciona la mejor base para utilizar la capacidad de la Lógica Difusa para la transformación del conocimiento y las preferencias del decisor en fórmulas lógicas. El uso del lenguaje como elemento de comunicación entre un analista y un decisor apunta más al uso de una combinación armónica de operadores, que hacia el uso de sólo uno de ellos (Ceruto et al. 2010) (Espín et al. 2011).

Existen dos características que principalmente dificultan el uso de los enfoques lógicos en la modelación de la decisión.

a) La propiedad asociativa de los operadores conjunción y disyunción utilizados.
b) La ausencia total de compensación de los valores de verdad de los predicados simples cuando se calcula la veracidad de los predicados compuestos haciendo uso de los operadores.

La asociatividad característica de una gran parte de los operadores utilizados para la agregación determina que árboles de predicados, que representan preferencias diferentes, produzcan valores de verdad iguales de sus predicados compuestos. Bajo la propiedad de asociatividad representada mediante la figura 2-9, observamos cómo estos árboles representarían las mismas preferencias, algo inapropiado en un modelo de toma de decisiones (Guzmán et al, 2010).

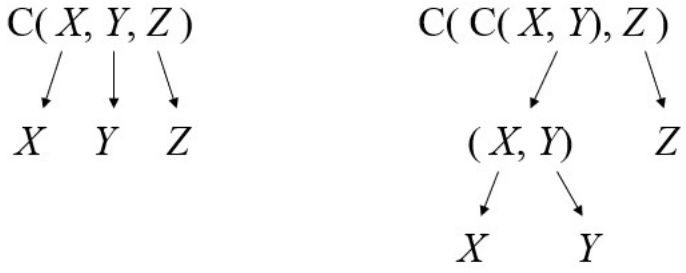

Figura 2-9 Arboles de representación de asociatividad.

Para ejemplificar lo expuesto se evaluarán los dos árboles mediante el uso de la lógica difusa la cual hace uso de operadores t-norma y t-conorma y la lógica difusa compensatoria que hace uso de operadores compensatorios.

Lógica difusa

Árbol 1

$$c(v_1 = 0.7, v_2 = 0.8, v_3 = 0.2)$$

$$0.7 \times 0.8 \times 0.2 = 0.112$$

Árbol 2

$$c(c(v_1 = 0.7, v_2 = 0.8), v_3 = 0.2)$$

$$c\big((0.7 \times 0.8 = 0.56), v_3 = 0.2\big)$$

$$0.56 \times 0.2 = 0.112$$

Lógica difusa compensatoria

Árbol 1

$$c(v_1 = 0.7, v_2 = 0.8, v_3 = 0.2)^{1/3}$$

$$(0.7 \times 0.8 \times 0.2)^{1/3}$$

$$(0.112)^{1/3} = 0.482$$

Árbol 2

$$c\left(c(v_1 = 0.7, v_2 = 0.8)^{1/2}, v_3 = 0.2\right)^{1/2}$$

$$c\left(\left((0.7 \times 0.8)^{1/2} = 0.7483\right), v_3 = 0.2\right)^{1/2}$$

$$c(0.7483 \times 0.2)^{1/2}$$

$$(0.0149)^{1/2} = 0.3868$$

Es posible observar la forma en que trabajan ambas lógicas, se observa en el primer árbol que la capacidad de compensación nos permite evaluar cada predicado simple mediante un operador que compensa los valores, a diferencia de la LD en la cual no existe este tipo de compensación, además al analizar el segundo árbol observamos que mediante LD obtenemos el mimo resultado que en el árbol 1, aunque evidentemente hay diferencia entre los arboles evaluados, en LDC observamos que al comparar el primer y el segundo árbol el grado de

aridad impacta en la evaluación del resultado, permitiendo tener grados de importancia para cada uno de los sub-predicados.

2.4 Programación genética

Como se observa en la naturaleza, las estructuras con más éxito son las que lidian con su entorno, sobreviven y crecen a un ritmo mayor. Es decir, las estructuras son el resultado de la aptitud. Las estructuras son creadas a partir de los procesos de recombinación genética y mutación de esta manera se entiende que la aptitud (en inglés fitness) define la estructura, los programas computacionales se encuentran entre las estructuras más complejas creadas por el hombre, de tal manera que se ha buscado responder una de las preguntas centrales en ciencia de la computación atribuida a Arthur Samuel en los años 50. ¿Cómo pueden las computadoras aprender a resolver problemas sin ser programadas explícitamente? (Koza, 1998).

La programación genética (GP por sus siglas en inglés) es una metodología, en la cual a través del uso de técnicas de computación evolutiva se dota a las computadoras de la capacidad de resolver problemas automáticamente. Desde sus inicios la GP ha sido estudiada, con el fin de llevar a cabo la resolución de una amplia gama de problemas prácticos, la cual ha producido una serie de resultados competitivos para el ser humano. En el nivel más abstracto, GP es una metodología sistemática la cual es independiente del dominio, logrando de esta manera la resolución automática de problemas computacionales a partir de una declaración de alto nivel, en la figura 2-10 se ve el flujo de la GP (Poli R. et al, 2008).

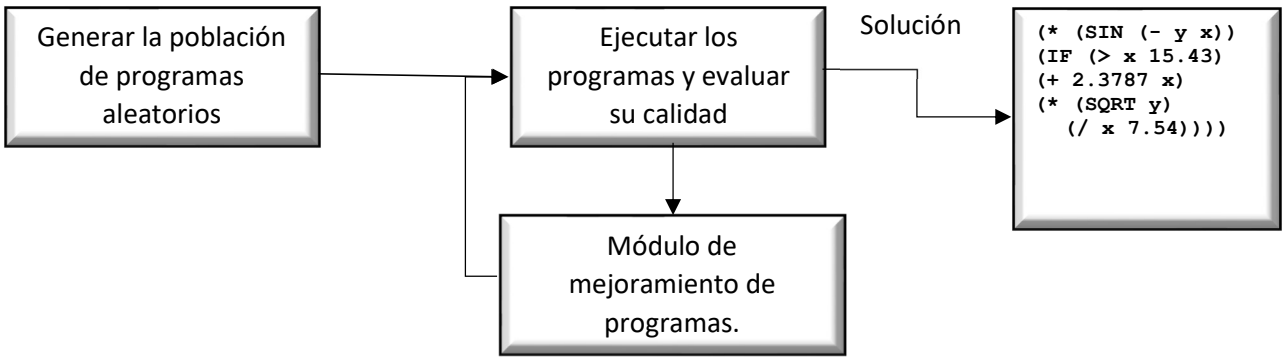

Figura 2-10 Flujo de control básico para la programación genética, donde la supervivencia del mas apto se utiliza para encontrar soluciones (Poli R. et al, 2008)

La GP lleva a cabo la evolución de programas, en la cual generación tras generación se transforman estocásticamente las poblaciones de programas en nuevas poblaciones, donde se espera que los resultados sean mejores, debido a que existe un factor de aleatoriedad, no se pueden garantizar los resultados. Sin embargo, es gracias a esta aleatoriedad que es posible escapar de trampas generadas por métodos determinísticos (Poli R. et al, 2008).

Las principales operaciones genéticas que se utilizan para crear nuevos programas a partir de los existentes son (Poli R. et al, 2008):

a) Cruza: mediante la selección de partes elegidas al azar de dos programas padres seleccionados se realiza una combinación de estas, obteniendo como resultado un programa hijo.
b) Mutación: se lleva a cabo la selección aleatoria de un programa padre, del cual se selecciona una parte al azar creando así un programa nuevo.

En programación genética, los programas se expresan como arboles de sintaxis en lugar de líneas de código usando como ejemplo la siguiente expresión $Var_1 \wedge Var_2 \vee Var_3$ las variables son las hojas del árbol (Var_1, Var_2, Var_3), cuyo nombre en GP es terminales. De la misma forma, los operadores lógicos $(\wedge,\vee)$ son nodos internos llamados funciones. La unión de estas funciones y terminales conforman un conjunto primitivo de un sistema GP (Poli R., 2008).

Es común en la literatura de GP representar las expresiones en notación prefija, ya que en esta notación se hace más sencillo observar la relación entre sub-expresiones y sus correspondientes subárboles. Tomando como ejemplo $\big((Var_1 \wedge Var_2) \vee (Var_1 \wedge Var_3)\big)$ se convierte en $\big(\vee (\wedge\ Var_1\ Var_2)(\wedge\ Var_1\ Var_3\ )\big)$ el cual se aprecia en el árbol de la figura 2-11.

De acuerdo con Janikow C. (1995) hacer uso de árboles de decisión es una de las metodologías más populares para la adquisición de conocimiento simbólico. El conocimiento resultante expresado en forma de árboles simbólicos y el uso

de mecanismos de inferencia fácilmente entendibles ha sido elogiado por su fácil comprensión.

Esto es debido a que se codifican relaciones semánticas explícitas entre atributos, que permiten a los usuarios comprender fácilmente el comportamiento de los modelos generados (Karabadji et al. 2017).

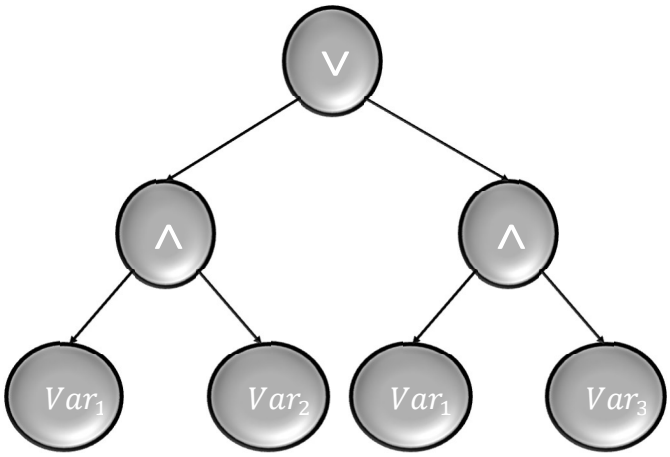

Figura 2-11 Árbol de decisión de GP que representa la formula $(\vee (\wedge\ Var_1\ Var_2)(\wedge\ Var_1\ Var_3\ ))$.

2.4.1 Población inicial aleatoria

Existen varias formas de generar una población aleatoria, sin embargo, para este caso se hará mención de dos tipos de metodologías de construcción de árboles de decisión los cuales son el método de árbol de crecimiento (Grow tree) y el de árbol completo (Full tree).

En ambas metodologías cada individuo se genera de manera que no excedan la profundidad máxima especificada por el usuario. La profundidad de un nodo es la cantidad de nodos que se deben atravesar para llegar al nodo comenzando desde el nodo raíz del árbol. La profundidad del árbol es definida por el nodo hoja de más profundidad del árbol (Poli R. et al, 2008).

En el método de árbol completo todas las hojas tienen la misma profundidad, los nodos se toman aleatoriamente del conjunto de funciones hasta que se alcanza la profundidad máxima del árbol En la figura 2-12 (Poli R. et al, 2008). Se aprecia la construcción de un árbol completo en 7 tiempos

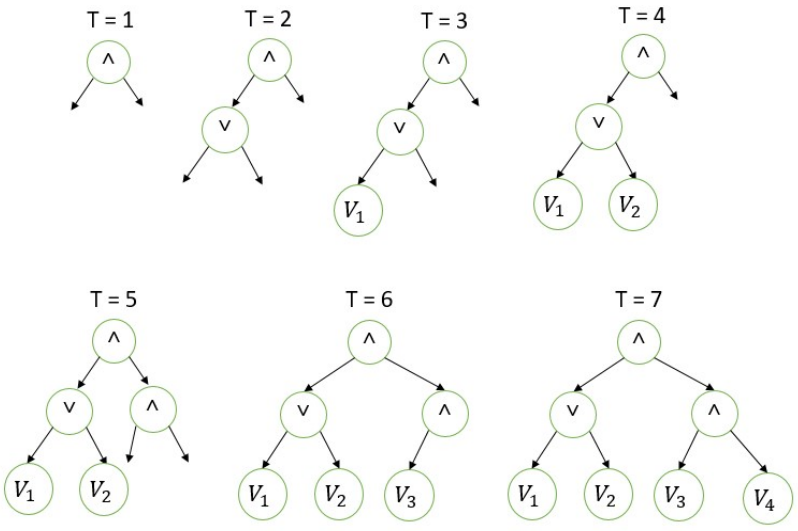

Figura 2-12 Construcción del árbol completo (full) en siete tiempos.

Este tipo de árboles es recomendado cuando las funciones tienen una misma *aridad* (número de argumentos necesarios para que dicho operador o función se pueda calcular), sin embargo, esta estructura genera construcciones algo limitadas.

El método de construcción de crecimiento permite inicializar árboles en los cuales el tamaño y la forma de los árboles son más variados. Ya que permiten elegir del conjunto total de variables y funciones en cualquier punto de la construcción a excepción del nodo inicial y las hojas, esto es realizado mientras no se alcance el límite de profundidad del árbol. En la figura 2-13 (Poli R. et al, 2008). Se ejemplifica la construcción de un árbol mediante este método en cinco tiempos.

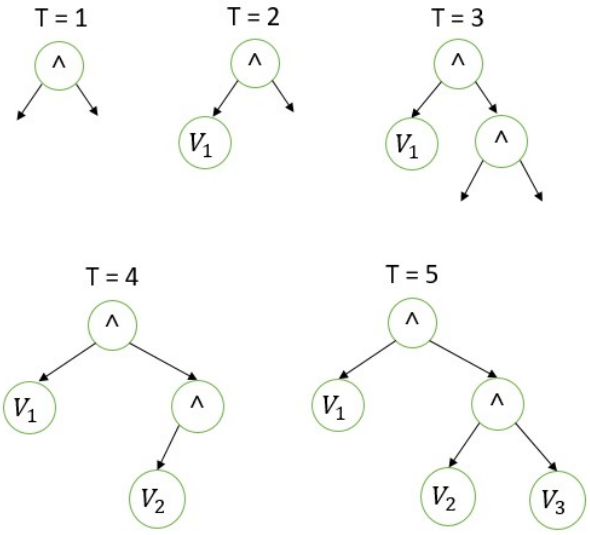

Figura 2-13 Construcción de árbol de crecimiento (Grow) en cinco tiempos.

Debido a que ni el método de árbol de crecimiento ni el método de árbol completo proporcionan una gran variedad de tamaños o formas por sí solos, es necesaria una combinación llamada rampa de mitad y mitad. Mediante este tipo de construcción la mitad de la población inicial se construye utilizando el método de árbol completo y la otra mitad se construye usando el método de árbol de crecimiento. Esto se hace usando un rango de límites de profundidad para ayudar a garantizar que generemos árboles con una variedad de tamaños y formas (Poli R. et al, 2008).

El algoritmo 2.1 sigue el enfoque “rampa mitad y mitad” (Poli R. et al, 2008).

Método: gen_rnd _expr(*func_set, term_set, profundidad - 1, método*)

1. **if** *profundidad* = 0 **or** (*método* = crecimiento **and** rand () < $\frac{|\ term\ set|}{|term\ set|+|func\ set|}$)
2. *expresión* = choose_random_element(*term_set*)
3. **else**
4. *función* = choose_random_element(*func_set*)
5. **for** *i* = 1 **to** arity(*función*) **do**
6. arg *i* = gen_rnd_expr(*func_set, term_set, profundidad - 1, método*);
7. **end for**
8. *expr* = (*func, arg 1, arg 2, ..., arg arity(función)*);
9. **end if**
10. **return** (expr)

Algoritmo 2.1 Algoritmo generador de la población inicial (método rampa de mitad y mitad).

Para tal algoritmo se reciben dos grupos de datos los cuales son los conectores lógicos (*func_set*) y las etiquetas lingüísticas a usar (*term_set*) y un limite de profundidad del árbol (*profundidad)*, el ultimo valor requerido es el tipo de construcción a usar (*método)*.

En la línea uno se verifica que no sea el nivel máximo de crecimiento o que el método seleccionado sea de crecimiento o 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.

2.4.2 Método de selección

Como sucede con la mayoría de los algoritmos evolutivos, en GP los operadores genéticos se aplican a individuos seleccionados probabilísticamente en función de su aptitud. De tal manera los mejores individuos tienen mejores probabilidades de tener descendencia que los inferiores. Existe una cantidad variada de métodos de selección, tal como los basados en selección proporcional método de ruleta, sobrante estocástico, universal estocástica, etc. (Poli R. et al 2008).

En este caso se menciona la metodología de selección por torneo.

La idea básica del método de selección por torneo es realizar la selección con base en comparaciones directas de los individuos (Coello Carlos, 2017).

Hay dos versiones de la selección mediante torneo:

a) Determinística (ver algoritmo 2.2)
b) Probabilística

Método tournament_selection(*poblation*)

```
1. While( p < number_poblation)
2. {
3.     Random_list = random_distribution(poblation);
4.     Parents =select_parents(random_list, number_parents);
5.     Best_parent = best_selection(parents)
6. }
```

Algoritmo 2.2 Método de selección por torneo determinista

En este algoritmo se lleva a cabo un numero de selecciones igual al numero de individuos existentes en la población total, en los cuales siempre se seleccionan los individuos de mejor aptitud.

En la línea tres se muestra como se baraja la población inicial de manera que la selección sea totalmente aleatoria, en la cuatro se selecciona un numero de padres (por lo general dos) de la lista aleatoria y se guarda en padres.

Una vez realizada esta acción se seleccionan los mejores individuos.

2.4.3 Método de cruza

Los métodos de cruza usados en la creación de algoritmos genéticos son variados, estos dependen del tipo de representación del cromosoma, del tipo de datos contenidos, de la complejidad de la estrategia de cruza.

En el uso de árboles de decisión, El método de cruza más utilizada es la cruza de subárboles. Dado dos padres, se hace el cálculo de la longitud de ambos y se seleccionan aleatoriamente puntos de cruza, posteriormente se crea la descendencia reemplazando el subárbol enraizado en el punto de cruce en una copia del primer padre con una copia del subárbol enraizado en el punto de cruce en el segundo padre (Poli R. et al, 2008). Es posible observar este proceso en la fig. 2.14 (Poli R. et al, 2008).

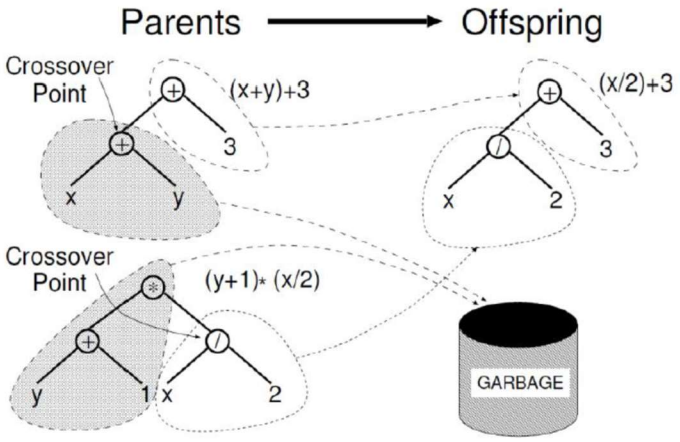

Figura 2.14 Representación grafica del método de cruza de subárbol (Poli R. et al, 2008).

En el algoritmo 2.3 demuestra la forma en que se lleva a cabo la operación de cruza.

Método: cruza_sub-arboles (*parent1, parent2*)
Entrada (cadenas de texto que representan arboles de decisión)
Salida (cadena de texto creada a partir de los datos iniciales)

1. *crosspoint* =select-subtree (*parent1*, rand ()); //Selecciona subárbol
2. *crosspoint2* = select-subtree (*parent2*, rand ()) ;//Selecciona subárbol
3. *subtree1* = traverse-tree(*crosspoint*); //recorre el subárbol
4. *subtree2* = traverse-tree(*crosspoint2*); //recorre el subárbol
5. delete-subtree(*subtree1*); //elimina el subárbol del árbol principal
6. *tree* = copy-subtree (*parent1*, *subtree2*, *crosspoint*); //remplaza el subárbol
7. **return** (*tree*); //regresa el nuevo árbol

Algoritmo 2.3 Algoritmo que efectúa la cruza de subárboles en dos árboles de decisión seleccionados

En el algoritmo anterior se describe como se seleccionan subárboles de los arboles padres a través de la elección de puntos de cruza completamente aleatorios, donde una vez realizado se recorren los subárboles para definir su tamaño, posteriormente se elimina el subárbol del *parent1* y se copia en este espacio el subárbol del *parent2* una vez realizadas estas acciones se obtiene un árbol hijo completamente nuevo.

Posteriormente los hijos obtenidos, remplazan a otros miembros de la población inicial, mediante el método de selección de torneo negativo. Esta estrategia de cruza nos permite seleccionar en varias ocasiones los mismos padres y no obtener el mismo hijo, también es posible cruzar varios subárboles por vez, pero esta estrategia no es comúnmente usada (Poli R. et al, 2008).

Mediante el uso de esta estrategia si no se cuenta con un método de selección uniforme, y tomando en cuenta las estructuras de los árboles, la mayor parte de subárboles seleccionados son nodos hoja. Esto genera que el intercambio genético entre arboles sea muy pequeño, es por tal necesaria la implementación de una estrategia que permita seleccionar en la mayor parte de las veces una operación de función, que un nodo hoja (Poli R. et al, 2008).

2.4.4 Método de mutación

El método utilizado más comúnmente en la mutación de subárbol es la llamada mutación de subárbol. En este método se selecciona aleatoriamente un punto de mutación en un árbol y sustituye el subárbol enraizado allí con un subárbol generado aleatoriamente.

La mutación del subárbol a veces se implementa como un cruce entre un programa y un programa aleatorio generado recientemente. Un ejemplo de esto se puede observar en la figura 2.15 (Poli R. et al, 2008). Esta operación también se conoce como cruce “headless chicken” (Poli R. et al, 2008).

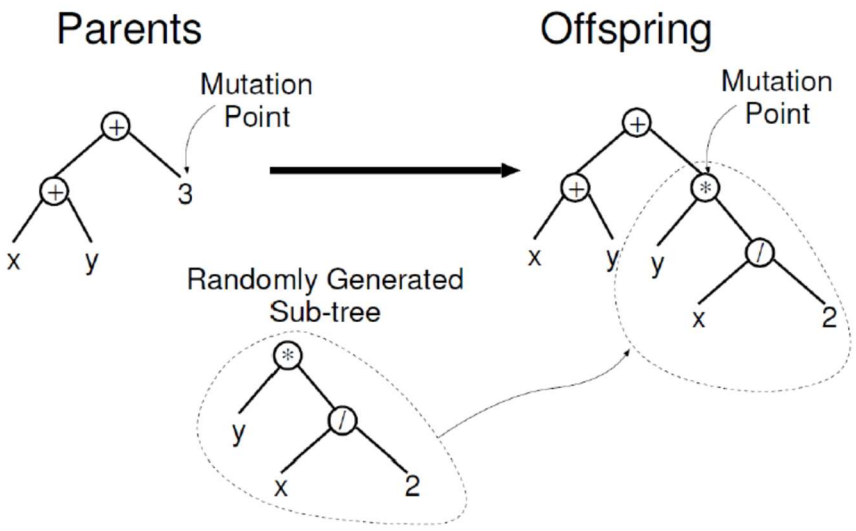

Figura 2.15 Representación grafica del método headless chicken (Poli R. et al, 2008).

Otra estrategia de mutación muy comúnmente usada es la de mutación de un punto, en la cual las funciones son cambiadas por funciones equivalentes pertenecientes al conjunto de funciones y las variables son cambiadas por otras variables de conjunto de variables elegidas al azar (Poli R. et al, 2008). La mutación puntual con el uso de probabilidades permite que varios nodos muten de manera independiente.

En el algoritmo 2.4 se representa el método de mutación:

Método mutación headless(*parent*)

Entrada (cadena de texto que representa un árbol de decisión)
Salida (cadena de texto que representa un árbol de decisión)

1. *longitud_parent* = longitud (*parent*);
2. *punto_mutación* = **rand**(*longitud_parent*);
3. *nuevo_subárbol* = crear_subárbol ();
4. *nuevo_árbol* = remplazar (nuevo_subárbol, punto_mutación);
5. **return**(nuevo_arbol);

Algoritmo 2.4 Método de mutación de arboles de decisión.

En este algoritmo se determina la longitud del padre recibido y posteriormente se selecciona un punto donde se llevará a cabo la mutación, una vez determinado se crea el nuevo subárbol y este se remplaza en el punto de cruza seleccionado generando así un nuevo hijo.

Mediante el uso de esta metodología cada nodo es considerado y, considerando una probabilidad, es alterado como se explicó anteriormente. Esto permite que varios nodos muten independientemente en una aplicación de mutación puntual (Poli R. et al, 2008).

Capítulo 3 Estado del arte

En el siguiente capítulo, se mencionan investigaciones que fueron desarrolladas por el grupo eureka, en estos experimentos se hace uso de una variedad de lógicas difusas compensatorias (LDC) con el fin de evaluar una serie de tareas las cuales son la evaluación de predicados difusos, el descubrimiento de predicados lógicos difusos compensatorios (PLDC) y también la implementación de una función de pertenencia generalizada (FPG) la cual nos permite crear funciones de membresía a partir de los datos contenidos en los almacenes de datos usados sin usar el conocimiento de un experto.

Así también se observan las diferentes metodologías heurísticas que fueron estudiadas por este grupo, y a través de esto vemos los resultados que lograron obtener y cuales fueron los resultados publicados.

En el desarrollo de la investigación, se encontró software diseñado para el uso de la LDC del cual se hace mención y se analizan los aspectos de cada una de estas aplicaciones para poder describir sus capacidades.

3.1 Evaluación de predicados lógicos difusos compensatorios

En la publicación del ingeniero Jesús Cejas Montero del 2009 de nombre la lógica difusa compensatoria se evalúa el manejo de la mercadotecnia de un grupo de tiendas a través de un predicado de LDC en las cuales las características son expresadas a través de conjuntos difusos y posteriormente evaluados mediante una metodología de LDC.

De acuerdo con los resultados se expresa que en el uso de la LDC se pueden desarrollar modelos a través de la ingeniería del conocimiento, así como la experiencia. Y que estos modelos nos permiten tomar decisiones coherentes mediante la interacción de directivos y expertos dentro de la organización. Así también, se toma a la LDC como un paso importante en la creación de un cálculo representado léxicamente.

En el 2010 en se publica el documento de investigación aplicación de la lógica difusa compensatoria en la selección de ofertas de armaduras ópticas por los autores Gil Guzmán, Kety M.; Chao Bataller, Adrian; Muñoz Gutiérrez, Salvador; Espín Andrade, Rafael A. en el cual mediante el uso de un modelo de

lógica difusa compensatoria se analizan una serie de factores y características que son relevantes en la operación de compra.

Para determinar la calidad de las armaduras se diseña un modelo expresado mediante un predicado de lógica difusa compensatoria (LDC) y a través del uso de este modelo evalúan una serie de armaduras ópticas cuyos resultados son mostrados en la figura 3.1.

Figura 3.1 Análisis de calidad de una serie de armaduras ópticas mediante un predicado de LDC (Gil Guzmán, et al. 2010).

De acuerdo con sus los resultados expresados, este experimento les permitió determinar que no se lleva a cabo el correcto análisis de proveedores, que a partir de un modelo de LDC es posible establecer un orden lógico de importancia con respecto a la calidad de cada armadura y que este modelo mejora la toma de decisiones con respecto a el proceso de compra de las armaduras ópticas.

En la publicación de nombre diseño e implementación informática de un cuadro de mando integral para el hotel villa la granjita de hostal masscotte cuyos autores son Ms. C. Pablo Michel Marín Ortega, Dr. C. Lourdes García Ávila, Dr. C. Rafael Espín Andrade, Prof. Dr. Jorge Marx Gómez. En el cual se hace

uso de la LDC para definir un indicador de control de gestión, que se encarga de medir el cumplimiento de la estrategia, a partir de la compensación de los indicadores definidos en un cuadro de mando integral.

Se menciona que mediante la aplicación de este método se logro identificar elementos que forman parte de la información necesaria para llevar a cabo el proceso de toma de decisiones brindando también a los directivos un mayor conocimiento de las áreas de la empresa.

En el documento un sistema lógico para el razonamiento y la toma de decisiones 2011, elaborado por el doctor Rafael Alejandro Espín Andrade, Eduardo Fernández González y Erick González Caballero en el cual se hace uso de la LDC como una herramienta que favorezca la toma de decisiones, en este caso se hace uso de la lógica difusa compensatoria basada en la media geométrica (LDCBMG).

En este caso se mide la competitividad de cuatro empresas fabricantes de adhesivos tisulares a través del modelado lógico difuso (ver tabla 3.1) de los conceptos que se consideran determinantes que para este caso son la variabilidad de su línea de productos vl(x), su fortaleza en el mercado m(x), el estado de su economía S(x), su tecnología T(x), si se tiene una línea de productos fuertes l(x), el grado de investigación en desarrollo de sus productos i(x), lo cual determina el grado de competitividad C(x).

Es posible observar en la tabla las calificaciones de cada aspecto tomado en cuenta y observar que la que mejor califica es la empresa C en la cual vemos que es más cierto que falso que es competitiva, de igual forma observamos que B es ligeramente mas cierto que falso que es competitiva y de A y D podemos afirmar que su competitividad es de baja a nula.

Se observa que la LDC se usa como una herramienta de evaluación multicriterio, adecuándose a las situaciones en las que el problema se puede describir de forma verbal, lo cual nos permite usar el lenguaje como elemento clave de comunicación en la construcción de modelos semánticos que facilitan la evaluación, la toma de decisiones y el descubrimiento de conocimiento.

Empresa x	vl(x)	m(x)	S(x)	T(x)	l(x)	I(x)	C(x)
A	0.23	0.1	0.5	0.516	0.234	0.058	0.246
B	0.77	0.4	0.611	0.682	0.627	0.393	0.545
C	0.92	0.8	0.812	0.584	0.903	0.815	0.728

D	0.39	1	0	0.763	0.58	0.336	0

Tabla 3.1 Resultados obtenidos mediante el uso de la LDCBMG en la evaluación de la competitividad

Mediante los experimentos revisados encontramos que la LDC es una herramienta que nos permite tomar el conocimiento léxico de personas expertas relacionadas con el tema de evaluación y poder expresarlos en forma de modelos de evaluación, lo cual nos sirve como herramienta en la toma de decisiones y en la generación de conocimiento.

3.2 Descubrimiento de conocimiento mediante el uso de la lógica difusa compensatoria

En la tesis de maestría de Taymi Ceruto Cordovés método para obtener predicados difusos a partir de datos utilizando metaheurísticas se hace uso de diversas metodologías metaheurísticas para lograr descubrir predicados difusos a partir de instancias de datos.

Se menciona el uso de la herramienta ICPRO y la creación de una herramienta mediante el apoyo de la biblioteca de clases BiCIAM, para la experimentación se hace uso de las instancias economía mexicana y diabetes.

Se realizaron experimentos mediante el uso de algoritmos de búsqueda aleatoria (BA), escalador de colinas (HC por sus siglas en ingles) y de un algoritmo genético (GA por sus siglas en ingles). Y las estructuras de construcción de predicados se basan en la forma norman conjuntiva (FNC) y la forma normal disyuntiva (FND).

Dentro de los resultados publicados con la instancia de economía mexicana, se encuentra que para los predicados en FNC el algoritmo de escalador de colinas tiene mejor comportamiento, sin embargo, se menciona que tiende a comportarse mal debido a la existencia de mesetas. En el caso de los predicados en FND se encuentra que el algoritmo de mejor desempeño es el algoritmo genético (ver figura 3.2).

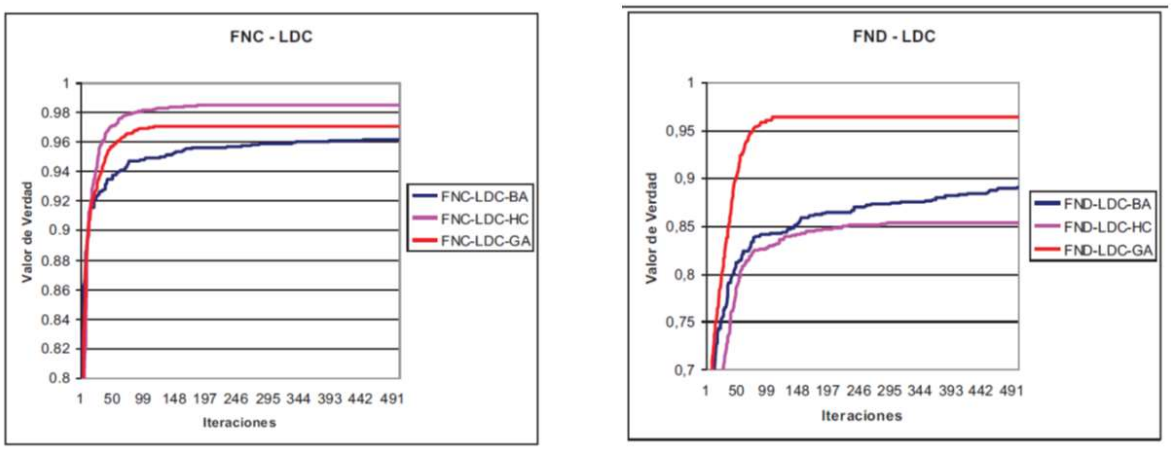

Figura 3.2 Resultado observados en la instancia economía mexicana Taymi Ceruto 2012.

En el caso de estudio de la instancia diabetes los resultados muestran que para los predicados en FNC el algoritmo heurístico con mejor comportamiento es el HC, mientras que para los predicados con estructura en FND los resultados fueron poco concluyentes, en la figura 3.3 se observan los resultados mostrados.

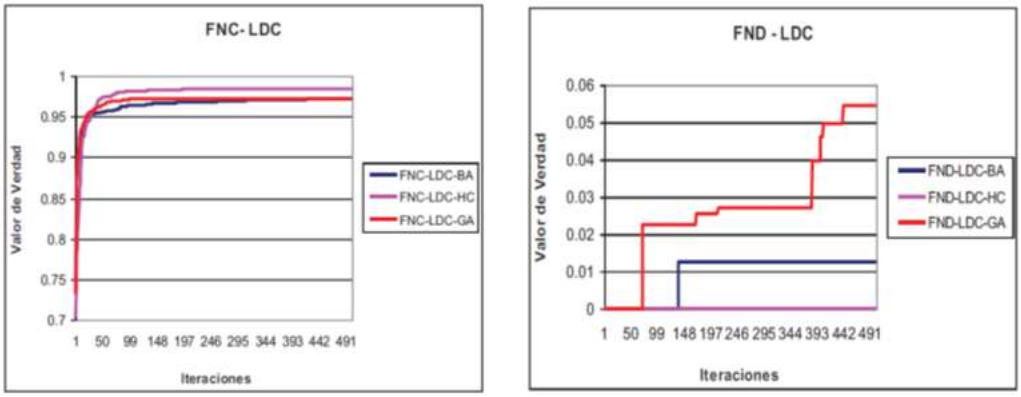

Figura 3.3 Resultados observadas en la instancia diabetes. Taymi Ceruto, 2012.

En el caso de la instancia de diabetes, se realiza una prueba de la cantidad de tiempo requerido para llegar a una solución óptima, en cuyos resultados muestra que tanto para estructuras de FNC y FND los mejores tiempos los alcanza mediante un GA (ver tabla 3.2).

Algoritmo	Tiempo (milisegundos)	Cantidad tuplas	Predicados
HC	30611	100	2
HC	342217	7672	1
BA	23354	100	2
BA	270383	7672	1
GA	8722	100	2
GA	85556	7672	1

Tabla 3.2 Resultados de experimentos de tiempos de ejecución Taymi Ceruto 2012.

Dentro de las conclusiones planteadas se plantea que los algoritmos actuales de aprendizaje computacional no son capaces de resolver el problema abarcado en la investigación y además que el método propuesto tiene como característica que el aprendizaje puede ser no supervisado.

Posteriormente en el 2013 Taymi Ceruto Cordovés, Alejandro Rosete Suárez y Rafael Espín-Andrade publican el articulo búsqueda de predicados en una base de datos utilizando metaheurísticas.

Para este caso se hizo uso de los algoritmos escalador de colinas búsqueda aleatoria y un algoritmo genético, y se utilizo la instancia de economía mexicana para llevar a cabo la experimentación.

Como resultado de los experimentos se encuentra que los predicados descubienrtos mediante el uso de estas metodologías muestran altos valores de verdad, correspondiendo cada uno de estos a valores superiores a .9 donde el máximo valor posible a encontrar es 1.

En este documento los predicados construidos están en FNC y FND el aprendizaje realizado es no supervisado.

En el documento Mfuzzypred: Algoritmo multiobjetivo para extraer predicados difusos del 2015 redactada por Dailé Osorio Roig1, Orenia Lapeira Mena, Taymi Ceruto Cordovés y Alejandro Rosete Suárez, en el cual se propone un método para el descubrimiento de predicados lógicos difusos a través de métodos heurísticos.

Como métodos de evaluación de la calidad del predicado, se utilizan además de el valor de verdad el cual es un cuantificador que representa la cantidad de tuplas

que satisfacen un predicado, tres cuantificadores más, los cuales son el soporte, que indica en cuantas tuplas está presente el predicado su valor oscila entre cero y uno, el lift conjuntivo, que mide el nivel de dependencia de los predicados simples cuando se relacionan a través de un operador conjunción y el lift disyuntivo, que mide el nivel de dependencia de los predicados simples cuando se relacionan a través de un operador de disyunción.

Los algoritmos usados son el escalador de colinas estocastico multiobjetivo (ECMO), búsqueda tabú multiobjetivo (BTMO), recocido simulado multiobjetivo (RSMO) y el escalador de colinas con reinicio estocastico multiobjetivo (ECMOR).

Para la cual fueron usadas las instancias basketball y stock, ambas disponibles en el repositorio UC Irvine Machine Learning Repository.

En la tabla 3.3 y 3.4 se muestran los resultados que alcanzaron:

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 3.3 Resultados obtenidos en la instancia Basketball Osorio 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

Tabla 3.4 Resultados obtenidos en la instancia Stock Osorio et al, 2015.

Se puede decir de este trabajo, que el uso de la LDC con el fin de encontrar predicados lógicos difusos puede llevarse a cabo y es mediante la inclusión de mas cuantificadores, que dan aun mayor soporte a sus resultados.

En este capítulo se observa que se han realizado experimentos, con el fin de demostrar que es posible llevar a cabo la obtención de conocimiento mediante el uso de algoritmos metaheurísticos, sin embargo, las construcciones de estos predicados difusos aún están muy limitadas, y no se hace un análisis real de los datos aportados por los algoritmos heurísticos usados, sino que se centra en la calificación de los predicados encontrados. Así también no se detalla cómo se llevaron a cabo las construcciones de estos algoritmos heurísticos y solo se menciona su uso y resultados.

3.3 Descubrimiento de predicados lógicos difusos compensatorios usando una función de pertenencia generalizada

En el documento de trabajo de investigación de 2018 de nombre uso de la generalización para el descubrimiento de conocimiento en un framework de lógica difusa compensatoria arquimediana redactado por Rafael A Espin Andrade, Erick Gonzalez caballero y Witold Predicz, en el cual se habla acerca de las funciones de pertenencia que permiten la transformación de los datos en conjuntos difusos, en este documento se menciona que si bien existe información acerca de las funciones de pertenencia, no se define en que casos de modelado pueden ser utilizados.

Es por este caso que se aborda el uso de una función de pertenencia generalizada (FPG), que pueda tomar los datos y modelar funciones de pertenencia a partir de estos, y si bien existen otras funciones de pertenencia de este tipo, se dice que es difícil modelar diferentes semánticas mediante su uso.

Para este caso se modela un PLDC con el fin de usar una FPG para cada uno de los predicados simples, de forma que el modelo obtenido sea completamente optimizado con esta metodología.

A través de el uso de la FPG, se obtuvieron funciones de pertenencia con diferente forma oscilando en funciones continuas de forma sigmoidal positiva, sigmoidal negativa o funciones convexas, además se hace posible el uso de los modificadores muy, extremadamente, etc.

Y en general, se menciona que debido al uso de la FPG es posible dotar al modelo de una gran expresividad de los resultados en lenguaje natural.

Mediante el uso de una FPG, es posible obtener mejoras a los predicados descubiertos, en otro de los casos es posible llevar el proceso de minería de datos sin necesidad de la modelación del problema por parte de expertos.

3.4 Software de lógica difusa compensatoria

Software Icpro

Este software fue presentado por primera vez en el 2008 por profesores investigadores de la Universidad de Mar del Plata, Argentina; dirigidos por el Ing. Gustavo Meschino. Se conceptualiza como un framework de análisis de datos con técnicas de Inteligencia Computacional (Bataller. 2014).

El sistema permite crear un nuevo proyecto que puede ser de los siguientes tipos: de lógica de predicados, de mapas autoorganizados o de agrupamiento K-medias. Asociado a esta investigación el proyecto que más se utilizó fue el de lógica de predicados; con el objetivo de insertar los predicados, tanto simples como compuestos, asociados al modelo basado en LDC.

Su utilización ha sido muy frecuente tanto a nivel empresarial como docente, obteniéndose resultados satisfactorios (Bataller. 2014). No obstante, a partir de su frecuente utilización se han encontrado algunas deficiencias en su funcionamiento desde el punto de vista informático.

Entre los problemas más frecuentes en la utilización del ICpro se encontraban:

- 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.
- Imposibilidad de construir varios modelos en un mismo proyecto.
- Imposibilidad de calcular el valor del parámetro alfa necesario para procesar predicados que utilizan conjuntos difusos para obtener el valor de verdad.
- Procedimiento muy complicado para insertar los predicados compuestos.

Software Fuzzy Tree Studio (FTS)

Este software posee un módulo cuyo objetivo es el de ayudar al usuario formalizar y calcular el valor de verdad de predicados implementando Lógica Difusa para cuantificar el valor de verdad de predicados parciales y operar adecuadamente con ellos, generalizando los conceptos de la Lógica de Predicados tradicional (Bataller. 2014).

Se hizo énfasis en la interfaz de usuario, en la búsqueda de lograr un software altamente amigable y fácil de utilizar. El propósito final es el de apoyar a los decisores en el análisis de datos, la generación de inteligencia y la evaluación y comparación de alternativa (Bataller. 2014).

Funciones:

- El sistema permite a los usuarios diseñar diagramas que representen árboles de predicados de Lógica Difusa, ofreciendo herramientas gráficas para su confección.
- Una vez finalizada esta etapa, se hace posible la generación de datos de entrada desde diversas fuentes, verificando su coherencia respecto al modelo propuesto.
- Para concluir el proceso, el sistema es capaz de evaluar el modelo y brindar información gráfica y completa sobre los resultados obtenidos.
- Esta solución de software es escalable. En una posterior etapa, se piensan incorporar algunas técnicas de optimización como algoritmos genéticos, que permitan perfeccionar el diseño del modelo.

Características del usuario:

- Conocimientos relacionados a la Lógica Difusa y el diseño de predicados.
- Respecto a la interacción con el sistema son suficientes nociones básicas sobre manejo de aplicaciones Windows.

Requisitos de funcionalidad:

- El sistema permite administrar proyectos destinados al diseño de árboles de decisión basados en Lógica Difusa y su posterior evaluación.
- El sistema ofrece herramientas gráficas para diseñar diagramas que representen predicados de Lógica Difusa.

- El sistema permite la utilización de predicados compuestos. Cada predicado debe estar caracterizado por un operador lógico y, a su vez, vinculado con uno o más predicados simples.
- El sistema permite exportar los diagramas diseñados a formato BMP (Windows Bitmap), PNG (Portable Network Graphics), PDF (Portable Document Format) y XPS (XML Paper Specification).
- El sistema permite ingresar conjuntos de datos desde distintas fuentes, que puedan ser utilizados para evaluar el predicado.
- El sistema permite evaluar el modelo del predicado difuso diseñado a partir de los conjuntos datos de entrada ingresados al proyecto.

Software Universe

El software universe se diseño, con el fin de implementar las metodologías de lógica difusa y lógica difusa compensatoria para el tratamiento de datos. Este se ejecuta desde un entorno de linea de comandos, para la cual existe una versión para Windows y para Mac, esta ultima también puede ser ejecutada en Linux.

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.

Para llevar a cabo las tareas de descubrimiento e inferencia, universe hace uso de algoritmos genéticos para efectuar la búsqueda.

Así también, universe permite trabajar con el siguiente tipo de lógicas:

1. Lógica de Zadeh.
2. Lógica Probabilística.
3. Lógica Compensatoria Basada en la Media Geométrica.
4. Lógica Compensatoria Basada en la Media Aritmética.

5. Lógica Compensatoria Basada en la Media Quasi-Aritmética.
6. Lógica Compensatoria Arquimediana.

Universe hace uso de un archivo de parámetros para llevar a cabo cada una de las tareas.

En la tabla 3.5 se hace una comparación de las características del software presentado.

Software							
	Disponibilidad	tareas			Interfaz amigable	Algoritmo de optimización	Manejo de datos
		E	D	I			
IcPRO	No	si	no	no	no	no	Si
Fuzzy Tree Studio	Si	si	no	no	si	no	Si
Universe.	Si	si	si	si	no	si	No

Tabla 3.5 Comparación de las características del software de LDC

Capítulo 4 Metodología

En este capítulo se describen los procesos que dan origen al presente proyecto de tesis, el cual es la creación de un algoritmo evolutivo que nos permita obtener conocimiento de un entorno de datos mediante el uso de la lógica difusa compensatoria (LDC).

Como en anteriores capítulos se ha observado, la LDC es una metodología lógica multivalente la cual nos permite obtener conocimiento expresado en predicados, y nos permite modelar el conocimiento de expertos de la misma manera, de tal forma que todo lo expresado en lenguaje natural puede ser transformado a un predicado de LDC.

De acuerdo con lo descrito en el capitulo dos, algunas de las LDC existentes en la literatura son la basada en la media geométrica (LDCBMG), la basada en la media aritmética (LDCBMA), la arquimediana (LDCA), etc.

Los predicados simples (PS) se componen de estados lingüísticos, lo cual quiere decir que para cada atributo del dataset se denomina mediante una o mas etiquetas lingüísticas y una función de pertenencia para cada etiqueta, mediante la función de pertenencia se genera el conjunto difuso y se realiza la modelación de la vaguedad. Por ejemplo, para el atributo edad, se pueden establecer etiquetas tales como joven, niño, anciano, muy anciano, muy niño, etc. Y modelar mediante funciones de pertenencia como la triangular, sigmoidal, función z, etc. Así también estos predicados simples se pueden conectar mediante operadores lógicos (OPL) los cuales son los operadores de conjunción, disyunción, implicación, equivalencia y negación. Los predicados simples evaluados a través de estos operadores generan predicados de LDC compuestos.

En la figura 4.1 se observa lo descrito en los párrafos anteriores, con el fin de generar una conceptualización general.

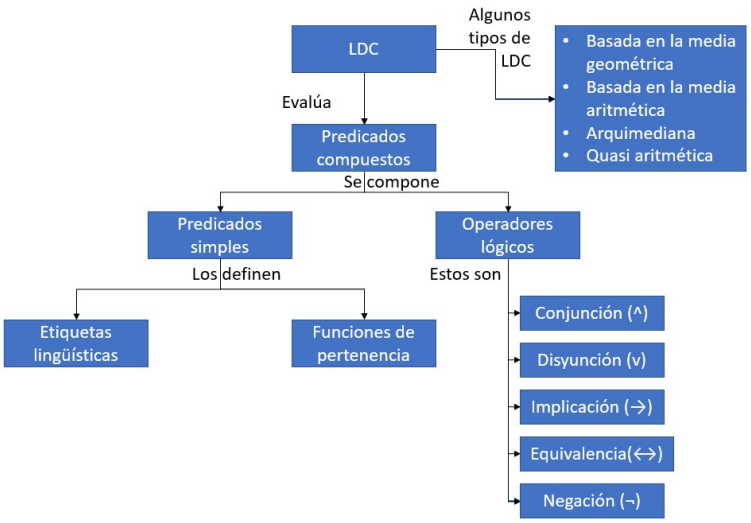

Figura 4.1 Mapa conceptual de la LDC

Otro de los aspectos que seria importante destacar es que el conocimiento expresado en predicados de LDC son generados a través de arboles de decisión, en los cuales se seleccionan de manera aleatoria los PS y los OPL a excepción de en el caso de los nodos raíz, en el cual solo se seleccionan OPL. En la figura 4.2, se observa el proceso de creación de este tipo de árboles.

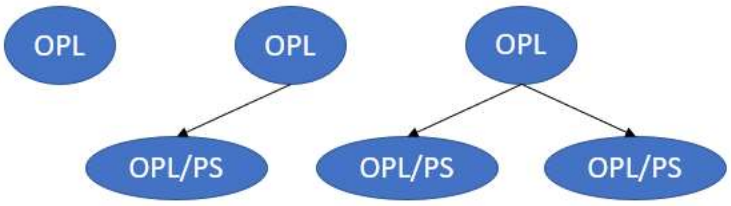

Figura 4.2 Construcción de un predicado de LDC a partir de un árbol de decisión

Estos son algunos aspectos que el autor piensa son necesarios aclarar para este capítulo, con el fin de hacer más sencilla la comprensión de la metodología en general.

En los siguientes párrafos se comienza con la descripción del funcionamiento del algoritmo nombrado Eureka-Universe. En el cual cada módulo de este se dedica a una tarea en específico.

Para poner este algoritmo en contexto en la figura 4.3 se desarrolla la funcionalidad de Eureka-Universe desde la vista de las conceptualizaciones que fueron usadas para lograr llevar a cabo el algoritmo propuesto.

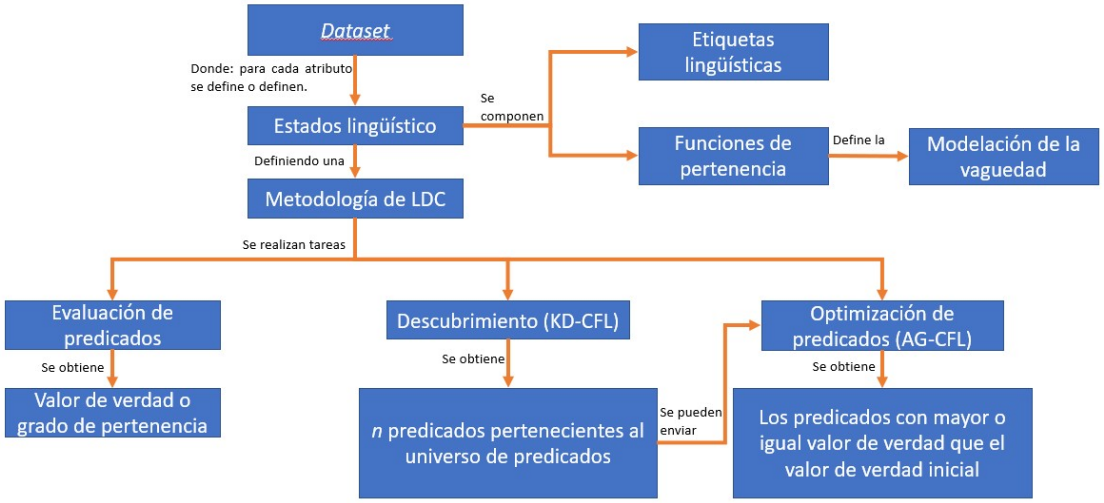

Figura 4.3 Esquema del funcionamiento del algoritmo Eureka-Universe

Es posible observar que para cada conjunto de datos recibidos se denominan los estados lingüísticos para cada atributo, se selecciona la metodología de LDC con la que se trabajara (LDCBMG, LDCBMA, etc.) y posteriormente se selecciona la tarea a realizar.

Para la tarea de evaluación se debe introducir el predicado del cual se desea conocer el valor de verdad con respecto al conjunto de datos, en la tarea de descubrimiento se generan predicados que existen en el universo de predicados posibles, todos estos predicados son FBF, y en la tarea de optimización de predicados, el predicado puede ser dado por el usuario o también ir tomando predicados del universo de predicados y buscar mejorarlos mediante la auto adaptación a través de los datos del conjunto.

En la sección 4.1 se describe el funcionamiento del algoritmo de descubrimiento de conocimiento expresado en predicados de LDC nombrado EK-CFL (Evolutionary Knowledge based on Compensatory Fuzzy Logic), en el cual a travez de un algoritmo genético que hace uso de las metodologías de programación genética se lleva a cabo la evolución de los predicados y se

obtiene de esta manera el conocimiento existente en las instancias de datos analizadas.

En la sección 4.2 se analiza el algoritmo nombrado EO-GSF (Evolutionary Optimization of Generalized Sigmoidal Function). En el cual se propone el uso de un algoritmo genético que se encarga de analizar los datos pertenecientes al atributo al que se ha asignado una FPG y por medio de la variación de parámetros de la FPG mejorar el valor de verdad recibido.

Debido a que la metodología de evaluación es solo un sub modulo que opera en función de la LDC seleccionada y no existe mayor problema en su ejecución, no se determina una sección para esta, sin embargo para tener una idea de su funcionamiento, este puede ser observado en el anexo -----.

Cada uno de estos módulos se pueden ejecutar de manera independiente el uno del otro, por lo cual se pueden llevar a cabo las tareas seleccionadas y posteriormente usar ese nivel de información en la siguiente tarea. De la misma manera, como estos módulos serán independientes las modificaciones en los mismos no impactara en gran manera en el algoritmo general.

En el algoritmo 4.1 nombrado Eureka-Universe se observa el funcionamiento general de este con respecto a los módulos que lo componen.

Método Eureka-Universe.

```
1:  Eureka_Universe(Dataset, memberfunction_list){
2:  Read("opción");
3:  Case 1
4:      Evaluate(predicate);
5:  Case 2
6:      KD-CFL(linguistic_states, );
7:  Case 3
8:      EO-GSF(predicate list);
9:  Return ("Predicates-CFL-list"); }
```

Algoritmo 4.1 Algoritmo general Eureka-Universe

4.1 Algoritmo genético de descubrimiento de predicados EK-CFL

En esta sección se describe el funcionamiento del algoritmo EK-CFL, el cual es un algoritmo genético que descubre reglas en instancias de datos usando varias metodologías de LDC, el funcionamiento general se describe en el algoritmo 4.2, el cual se presenta a continuación.

Method EK-CFL

```
1: EK-CFL(Dataset,memberfunction_list)
2: {
3:  poblation = random_poblation();
4: fitness_individuos = calcula_fitness(individual); // Genera la población inicial
5: evaluate(poblation); //Evalua el fitness de cada individuo de la población
6: do{
7:      tournament_selection(poblation); //Selecciona los individuos a modificar
8:      new_child[i] = cruze(parent1, parent2); //Cruza los individuos
        seleccionados
9:      new_child[i] = mute(parent1); //muta los individuos seleccionados
10:     poblation = replace(poblation, new_child); //Remplaza los mejores
        individuos
11:     if(FPG = true) //Si hay funciones de pertenencia generalizadas
12:        EO-GSF(individuals); //Optimiza las FPG,s
13:     evaluate(poblation);
14: }While(Best N individuals = false or generations < max_generations);
    //Mientras no se consigan los individuos con un mínimo valor de verdad seleccionado
    o no se llegue al máximo de generaciones
15: Return(Best N individuals or Best individual); //se regresa la lista de los
    mejores individuos o el mejor individuo en caso de no alcanzar el valor de verdad
    seleccionado
16:    }
```

Algoritmo 4.2 Algoritmo de descubrimiento de predicados

En la figura 4.4 se describe el comportamiento de este algoritmo de manera gráfica, además es posible observar que lo que hacemos en este proceso de KD es construir arboles de decisión de manera aleatoria los cuales al paso de cada generación se irán modificando mediante operaciones genéticas, esperando

obtener predicados de LDC con altos valores de pertenencia al conjunto buscado

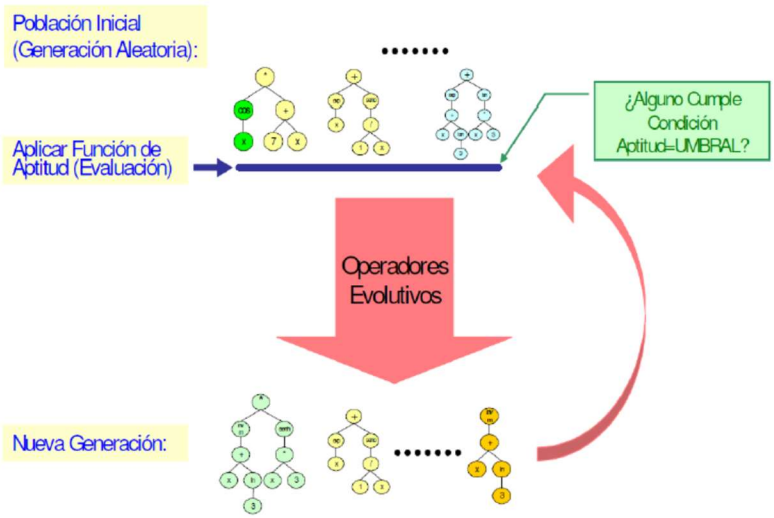

Figura 4.4 Ejemplo grafico de un algoritmo evolutivo con uso de árboles de decisión Poli R, et al. 2008

4.1.1 Generación de la población aleatoria

El primer paso en la construcción de un algoritmo genético es la creación de la población la cual se generará de manera aleatoria. En la literatura se presentan diferentes métodos de generar esta población.

En la presente tesis la construcción de individuos se hace mediante el uso de árboles de decisión de programación genética tal como se vio en el capítulo 2.4. En poli R. et al, 2008 se mencionan dos tipos de metodologías de construcción de árboles de decisión los cuales son el método de árbol de crecimiento (grow tree) y el de árbol completo (full tree).

Para este caso, en los arboles de decisión construidos los nodos hoja son estados lingüísticos y los nodos internos son operadores de LDC. En ambas metodologías cada individuo se genera de manera que no excedan la profundidad máxima especificada por el usuario. La profundidad de un nodo es la cantidad de nodos que se deben atravesar para llegar al nodo comenzando desde el nodo raíz del árbol. La profundidad del árbol es definida por el nodo hoja de más profundidad del árbol (Poli R. et al, 2008).

En el método de construcción de un árbol completo (full tree), la construcción del árbol de decisión se lleva a cabo rama a rama, en cada nodo interno se seleccionan los operadores de LDC que hayan sido especificados por el usuario estos operadores son los de conjunción (∧), disyunción (∨), implicación (→), equivalencia (↔) y negación (¬), Los cuales son tomados de manera aleatoria. Una vez que se han determinado los nodos internos en la rama que se está generando, se comienza con la selección de los estados lingüísticos que al igual que en las operaciones anteriores son seleccionados de manera aleatoria, en la figura 4.5 se puede observar un ejemplo de lo mencionado.

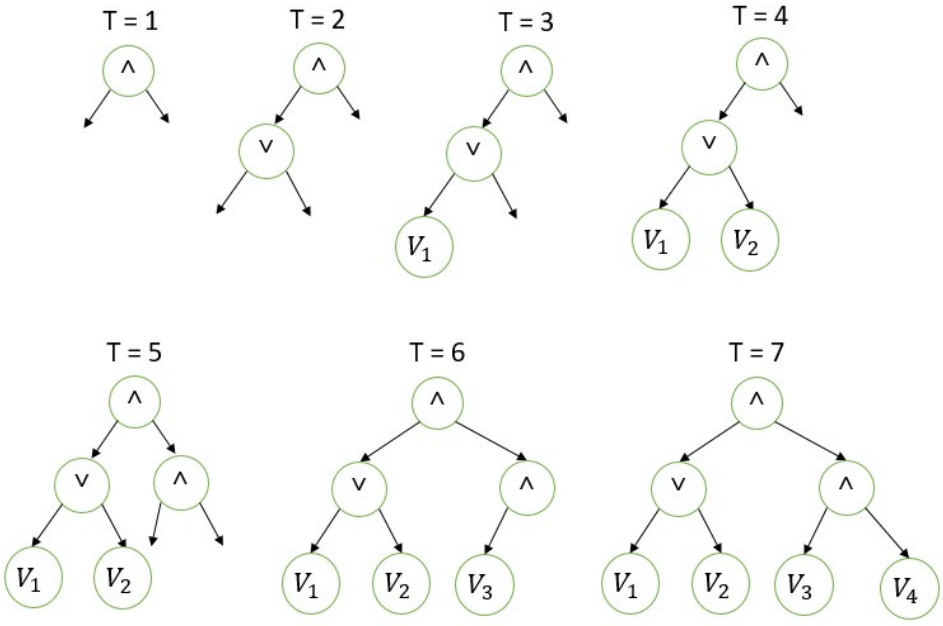

Figura 4.5 Construcción de árbol completo en 7 pasos

Encontramos en Poli R. et al, 2008. Que este tipo de árboles es recomendado cuando las funciones tienen una misma *aridad* (número de argumentos necesarios para que dicho operador o función se pueda calcular), sin embargo, este tipo de construcciones deja de lado otro tipo de posibles construcciones.

En el algoritmo 4.3, se describe como se lleva a cabo la construcción de los arboles completos.

```
Method Complete_tree

1:  Complete_tree(Depth, Position)
2:  {
3:  if (Depth = Depth_max) // Si la profundidad es un nodo hoja
4:  {
5:      Buffer[position] = rand(linguistic_state); //selecciona un estado lingüístico y lo guarda en
        cadena.
6:      Position++;
7:      Depth++;
8:  }
9:  Else //Si el nodo no es un nodo hoja
10:{
11:     Buffer[Position] = rand(operator_LDC); //selecciona un operador de LDC
12:     Position++;
13:     Depth++;
14:}
15:Complete_tree(Depth, Position); //llama a complete tree hasta terminar de construir el
   arbol
16:}
```

Algoritmo 4.3 Algoritmo de creación de arboles completos (Full tree)

En el caso de los árboles de crecimiento (grow tree) las construcciones pueden darse de formas variadas, debido a que cada una de las ramas no necesariamente necesitan alcanzar la profundidad definida por el usuario.

En cada nodo construido tanto los operadores de LDC como los estados lingüísticos pueden ser seleccionados, aun así, todos los nodos hojas deben ser estados lingüísticos, dicho de otra forma, una vez seleccionado un estado lingüístico se finaliza la profundidad máxima de esa rama, de la misma manera cuando se alcanza la profundidad definida por el usuario, la selección se hará solo del conjunto de estados lingüísticos. En la Figura 4.6 se ejemplifica la construcción de un árbol de crecimiento en 5 tiempos.

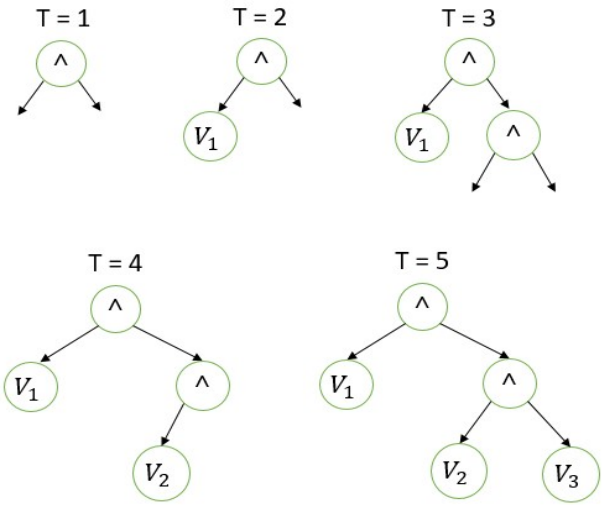

Figura 4.6 Árbol de crecimiento construido en 5 pasos

En el algoritmo 4.4 se puede observar cómo se lleva a cabo la construcción de un árbol de crecimiento.

Metodo Grow_tree

```
17:Complete_tree(Depth, Position)
18:{
19:Probability = rand(100); //genera un numero entre 0 y 99
20:if (Probability > 80 || Depth==Max_Depth) //si el numero es mayor que 80 o el nodo es un
    nodo hoja
21:{
22:     Buffer[position] = rand(linguistic_state); //Se selecciona un estado linguistico
23:     Position++;
24:     Depth++;
25:}
26:Else //En otro caso
27:{
28:     Buffer[Position] = rand(operator_LDC); //Se selecciona un operador de LDC
29:     Position++;
30:     Depth++;
31:}
32:Grow_tree(Depth, Position);
33:}
```

Algoritmo 4.4 algoritmo de construcción de árboles de crecimiento

De acuerdo con lo encontrado en la literatura ni el método de árbol de crecimiento ni el método de árbol completo proporcionan una gran variedad de tamaños o formas por sí solos, es por lo tanto que es necesaria una combinación llamada rampa de mitad y mitad (Poli R. et al, 2008).

En la construcción de la población el algoritmo propuesto genera la mitad de los arboles construidos con árboles de crecimiento y la otra mitad son arboles completos, lo cual nos permite tener una mayor variedad de selección y evitar que buenas construcciones no sean tomadas en cuenta debido a las limitantes propias de cada metodología.

En el estudio de las maneras de representar un árbol de decisión se llevaron a cabo diferentes pruebas, al final se optó por lo recomendado en la literatura, cada individuo es representado mediante una cadena en notación prefija debido a la sencillez de sus construcciones y su comprensibilidad, así también, fue necesario el uso de auxiliares en operadores de LDC cuya aridad puede ser de 2 a *n* variables posibles. En la figura 4.7 se hace uso de los arboles observados en la figura 4.5 y 4.6 para representarlos por medio de una cadena en notación prefija.

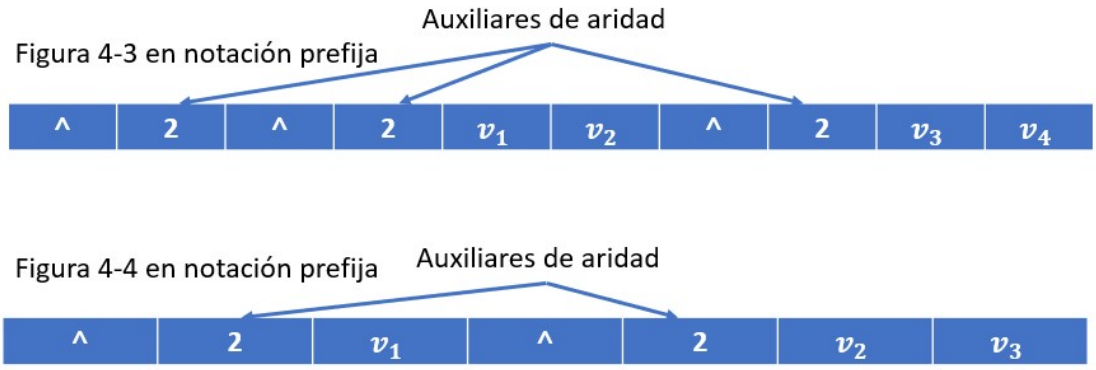

Figura 4.7 Representación en notación prefija de arboles de las figuras 4.5 y 4.6

4.1.2 Método de selección

Existe una cantidad variada de métodos de selección, tal como los basados en selección proporcional (método de ruleta, sobrante estocástico, universal estocástica, etc.), mediante torneo, mediante selección uniforme, etc.

En este caso se hace uso de la metodología de selección por torneo.

La idea básica del método es seleccionar con base en comparaciones directas de los individuos (Coello Carlos, 2017).

Hay dos versiones de la selección mediante torneo:
• Determinística
• Probabilística

4.1.3 Método de cruza

Los métodos de cruza usados en la creación de algoritmos genéticos son variados, estos dependen del tipo de representación del cromosoma, del tipo de datos contenidos, de la complejidad de la estrategia de cruza.

En el uso de árboles de decisión, El método de cruza más utilizada es la cruza de subárboles. Dado dos padres, se hace el cálculo de la longitud de ambos y se seleccionan aleatoriamente puntos de cruza, Luego, crea la descendencia reemplazando el subárbol enraizado en el punto de cruce en una copia del primer padre con una copia del subárbol enraizado en el punto de cruce en el segundo padre (Poli R. et al, 2008). Es posible observar este proceso en la fig. 4.8 (Poli R. et al, 2008).

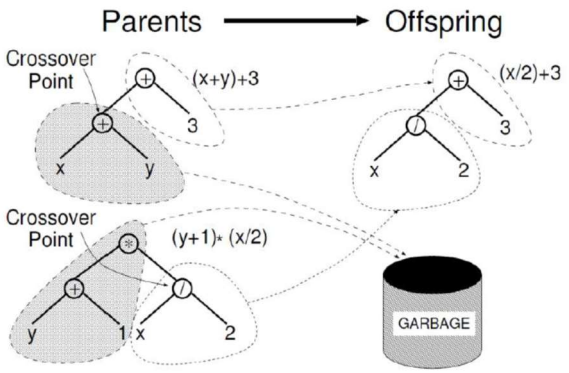

Figura 4.8 Método cruza de subárboles

4.1.4 Método de mutación

El método utilizado más comúnmente en la mutación de subárbol es la llamada generación de subárbol. En este método se selecciona aleatoriamente un punto de mutación en un árbol y sustituye el subárbol enraizado allí con un subárbol generado aleatoriamente.

La mutación del subárbol a veces se implementa como un cruce entre un programa y un programa aleatorio generado recientemente. Un ejemplo de esto se puede observar en la figura 4.9 (Poli R. et al, 2008). Esta operación también se conoce como cruce "headless chicken" (Poli R. et al, 2008).

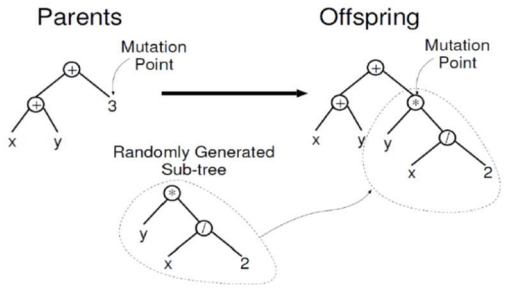

Figura 4.9 mutación mediante el método Headless Chiken Poli R. et al, 2008.

4.2 Algoritmo de optimización de parámetros EO-GSF

Tal como se ha mencionado en capítulos anteriores la forma en que se crean los conjuntos difusos es a través del uso de funciones de membresia, que toma los valores del atributo y llevan a cabo una normalización lógica multivalente en el intervalo [0,1]. Para llevar esta operación a cabo, es necesario tener conocimiento respecto al tema y poder así llevar a cabo un modelado correcto de los atributos del problema.

Sin embargo, han propuesto el uso de una función de pertenencia generalizada, la cual a través del uso de la función sigmoidal, genera una serie de familias de funciones que no dependen del conocimiento de un experto, sino que, mediante los datos genera funciones de membresía que permiten mejorar la calidad de los predicados y obtener altos grados de verdad.

En esta sección se presenta el algoritmo genético de optimización de parámetro de la función de pertenencia generalizada, la cual esta basada en la función sigmoidal generalizada (GSF) que es mediante la cual descubrimos las mejores configuraciones para los parámetros.

En el algoritmo 4.5 se observa el funcionamiento de este.

Algoritmo 2 EO-GSF

{*xbest* : TipoSolucion} = EA(*N*: integer; *f* : TipoFuncionObjetivo)
/**N* es el número de individuos por generación*/
var
x*g*, **x**∗*g* : **array** [1, . . . ,*N*] **of** TipoSolucion; // Población;
begin
g := 1; // Inicializar el número de generación
{**x**} := *inicializar*(*N*); // Inicializar la población

repeat
{**x**∗} := *seleccionar*(**x**, *f*); // Seleccionar los mejores individuos
{**x**} := *alterar*(**x**∗); // Alterar los individuos seleccionados
g := *g*+1;
until terminacion

{*xbest*} := *seleccionar*(**x**, *f*); //Seleccionar la mejor solución encontrada
End

Algoritmo 4.5 Algoritmo de optimización de parámetros

4.2.1 Generación de la población aleatoria

Para este caso, 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.

Cada estado lingüístico definido a partir de una FPG se compone de estos tres parámetros los cuales son iterados aleatoriamente buscando la mejor configuración para el estado, lo cual nos permita encontrar la mejor configuración para el predicado de LDC, con mayor valor de verdad.

En este caso la representación del cromosoma será llevado a cabo, de tal manera que los estados lingüísticos que estén siendo optimizados se integren en un cromosoma principal. En este caso el estado 1 se coloca en los primeros 3 alelos del cromosoma, el segundo en los siguientes tres alelos, formando grupos de 3 para cada representación.

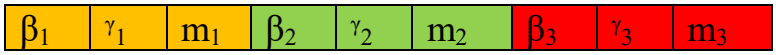

Figura 4. 10 cromosoma de optimización de predicados de LDC mediante una FPG

Para calcular los valores de cada uno de los atributos que serán incluidos en la representación se hace uso de las siguientes formulas:

$$\beta = Rand(Min, \ldots., Max) < \gamma$$

$$\gamma = Rand(Min, \ldots., Max) > \beta$$

$$m = Rand(0,1)$$

Esto quiere decir que el atributo β se selecciona aleatoriamente de los datos, mas debe ser 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.

4.2.2 Método de selección

Para el método de selección usado en este algoritmo genético, usamos el de selección por estado uniforme.

Esta técnica fue propuesta por Whitley y se usa en AGs no generacionales, en los cuales solo unos cuantos individuos son reemplazados en cada generación (Coello, 2017).

En general, la técnica resulta útil cuando los miembros de la población resuelven colectivamente (y no de manera individual) un problema.

En el algoritmo 4.6 se observa el funcionamiento de selección por estado uniforme.

Metodo seleccion

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

Algoritmo 4.6 Método de selección por estado uniforme

4.2.3 Método de cruza

El método de cruza utilizado es inspirado en la manera en cómo se revuelven las cartas en un juego de barajas, parecido también al método de cruza acentuada. Por medio de la cual se seleccionan aleatoriamente un numero *n* de alelos del primer padre y se sustituyen en el hijo. A continuación, se seleccionan

un numero aleatorio *n* de alelos del segundo padre y se sustituyen en el hijo, esto se realiza hasta terminar el proceso de creación del hijo.

En el algoritmo 4.7 se observa el método de cruza.

```
1. Metodo: Cruza (parent1, parent2) {
2. For (i=0; i<length(parent1); i++) {
3. cartas = random () * (lenght (parent1));
4. For (j=0; j<cartas && i<length(parent1); i++,j++){
        i. hijo[i] = parent1[i];}
5. cartas = random () * (lenght (parent));
6. For (j=0; j<cartas && i<length(parent1); i++, j++){
        i. hijo[i] = parent2[i];}
7. if (i< length(parent1))
        i. i--;
8. }}
```

Algoritmo 4.7 Método de cruza por barajeo.

4.2.4 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.

En el algoritmo 4.8 se observa el funcionamiento de este.

```
Method: mutacion(parent){
mutaciones = random() * lenght(parent);
For(i=0;i<mutaciones; i++){
        Parent[i] = random () * (máximo –
minimo ) + minimo;}
}
```

Algoritmo 4.8 Algoritmo de método de mutación

4.3 Función de evaluación

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.

Dependiendo del tipo de operación difusa usada, variara la fórmula de obtención del grado de universalidad de las reglas encontradas, sin embargo, tal como se ha mencionado esta se basa en la operación de conjunción de los valores de pertenencia de cada uno de los elementos del registro, es por tal que la manera de representarlo es la siguiente:

$$\underset{p_L \in P_L}{maxtruevalue(p_L)} = c_{x \in U}\, p_L(x)$$

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.
C = operador de conjunción lógica de cada registro que evalúa el predicado.
L = Metodología de LDC que evalúa el predicado.

Capitulo 5 Experimentación y resultados

En esta sección se presentan los experimentos realizados para llevar a cabo la evaluación del núcleo denominado Eureka-Universe, los cuales permiten lograr una comprensión de cada uno de los procesos por separado, así como también se describen las instancias utilizadas en los experimentos de manera que en la descripción de cada experimento solo se hará mención de la instancia usada y se detalla información que es usada de manera general en cada uno de los experimentos.

El equipo usado en cada uno de los experimentos es una maquina Dell con un procesador Intel Core i7-7500u con 2.7 Ghz *2, una memoria Ram de 16 Gb, y un sistema operativo Windows 10. El lenguaje usado para la compilación del programa es java, mediante el uso de NetBeans.

La información de las instancias se presenta en la tabla 5-1.

Nombre	Descripción	Atributos	Objetos
Tinto.txt	Esta instancia está compuesta de un conjunto de vinos con diferente grado de calidad.	12	1599
Order.txt	Esta instancia está compuesta de un conjunto de objetos de almacén y su correspondiente información de ubicación.	11	45
Bupa.txt	Esta instancia contiene análisis sanguíneos que determinan el grado de alcohol en la sangre	5	345

Tabla 5.1 Características de las instancias usadas para experimentación

Es también pertinente mencionar que para cada uno de los experimentos se realizó un total de 30 pruebas para cada configuración, de tal manera que los resultados obtenidos puedan ser observados de manera objetiva.

5.1 Experimento 1: Generación de predicados de lógica difusa compensatoria

Objetivo: Generar predicados lógicos difusos mediante el uso del software Universe que usa la metodología de árboles completos y el algoritmo EK-CFL que usa el método rampage, evaluando los resultados mediante el cuantificador de universalidad y comparando la sencillez de los predicados construidos.

Configuración del experimento

Mediante el uso de la instancia Tinto.txt, se llevó a cabo la generación de predicados de LDC buscando satisfacer un mínimo valor de verdad (evaluado mediante el cuantificador de universalidad) para cada una de las pruebas realizadas.

En la tabla 5.2 se muestran los valores de algunos de los atributos antes llevar a cabo la normalización lógica multivalente (NLM) en el intervalo [0,1].

fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides
11.6	0.580	0.66	2.2	0.074
10.4	0.610	0.49	2.1	0.2
7.4	1.185	0.00	4.3	0.097
10.4	0.440	0.42	1.5	0.145
8.3	1.020	0.02	3.4	0.084
7.6	1.580	0.00	2.1	0.137
6.8	0.815	0.00	1.2	0.267
7.3	0.980	0.05	2.1	0.061
7.1	0.875	0.05	5.7	0.082
6.7	0.760	0.02	1.8	0.078

Tabla 5.2 Atributos de la instancia vinos sin normalizar

En la tabla 5.3 se muestran los valores de los atributos después de la NLM mediante el uso de diversas funciones de pertenencia.

fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides
0.83094904	0.66735145	0.22034484	0.97703806	0.02391793
0.6225671	0.75286821	0.66580406	0.97856489	0.16231351

0.09714866	0.99989062	0.99820022	0.90954692	0.03450749
0.6225671	0.22214419	0.81659014	0.98584279	0.07284904
0.1961736	0.99891279	0.9977364	0.94815478	0.02806388
0.11432238	0.99999955	0.99820022	0.97856489	0.06446379
0.05867524	0.98144911	0.99820022	0.98850613	0.36782297
0.08945497	0.99810393	0.99680797	0.97856489	0.01941196
0.07569759	0.99186966	0.99680797	0.78376771	0.02718234
0.05384673	0.96093423	0.9977364	0.98257319	0.02549932

Tabla 5.3 Atributos después de la NLM

En la tabla 5.4 se describe la configuración usada en los algoritmos genéticos de Universe y EK-CFL y el mínimo valor de verdad buscado en cada prueba.

Configuración del algoritmo genético.	
Población	100
Profundidad del árbol	3
Porcentaje de cruza	95%
Porcentaje de mutación	5%
Numero de generaciones.	100
Valores mínimos de verdad	0.99, 0.98, 0.97, 0.96, 0.95

Tabla 5.4 Configuración de los algoritmos usados para el proceso de KD

Resultados

En la tabla 5.5 se muestra el resultado promedio para cada uno de los valores de verdad buscados en cada uno de los experimentos realizados, de la misma forma en la tabla 5.6 se observan algunos de los predicados construidos.

Resultados		
Mínimo valor de verdad	**Valores de verdad encontrados por Universe**	**Valores de verdad encontrados por Eureka**
0.99	0.97365	0.99185
0.98	0.937841	0.99364
0.97	0.926196	0.99129

0.96	0.944852	0.96223
0.95	0.964967	0.97610

Tabla 5.5 Valores de verdad alcanzados por los algoritmos mediante la experimentación

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"))

Tabla 5.6 Predicados construidos por los algoritmos en la experimentación

Conclusión del experimento

Se observa en el experimento que el promedio general de Universe es un valor de verdad de 0.9495 y el obtenido a través de Eureka-Universe es de 0.9830 lo que muestra una diferencia de 0.03352 grados de verdad. De la misma forma Universe genera una desviación estándar de 0.01951, por otro lado, Eureka-Universe tiene una desviación estándar de 0.01359, a través de lo cual no se muestra una diferencia significativa, en lo que respecta a la construcción de predicados, se observa una mayor capacidad de construcción por parte del método rampage, y en la mayoría de las ocasiones la capacidad de alcanzar los valores de verdad requeridos por el tomador de decisiones.

5.2 Experimento 2: Optimización de predicados lógicos difusos compensatorios

Objetivo: En este experimento, se genera un grupo de predicados de LDC mediante el uso de FPG,s sin usar la información de expertos y se mejora su valor de verdad, mediante la optimización de los parámetros α, γ y m de una función de pertenencia generalizada, por medio de la cual se lleva a cabo la NLM para cada atributo (véase tabla 5-3). posteriormente, se hace una comparación de los resultados obtenidos por el software Universe y Eureka-Universe.

Configuración del experimento

En la tabla 5.7, se observa el grupo de predicados generados de manera completamente aleatoria usando como base la instancia Tinto.txt (véase tabla 5.2), los cuales posteriormente serán optimizados mediante el uso de los algoritmos genéticos existentes en Universe y Eureka-Universe, cuya configuración se observa en la tabla 5.8.

Predicados generados
(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")
(IMP (NOT "new chlorides") "new quality")
(IMP (OR "new pH" "new density" "new fixed_acidity" "new chlorides") "new quality")
(IMP (OR "new volatile_acidity" "new density") "new quality")

(IMP (NOT "new sulphates") "new quality")
(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")
(IMP (OR "new sulphates" "new volatile_acidity" "new residual_sugar" "new pH") "new quality")
(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")
(IMP (OR "new fixed_acidity" "new pH") "new quality")
(IMP (NOT "new volatile_acidity") "new quality")
(IMP (OR "new sulphates" "new residual_sugar" "new free_sulfur_dioxide" "new chlorides") "new quality")
(IMP (OR "new residual_sugar" "new total_sulfur_dioxide" "new pH" "new citric_acid" "new density" "new free_sulfur_dioxide" "new alcohol") "new quality")
(IMP (IMP "new chlorides" "new volatile_acidity") "new quality")
(IMP (IMP "new total_sulfur_dioxide" "new density") "new quality")
(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")

Tabla 5.7 Predicados usados en la experimentación

Configuración del algoritmo genético Universe y Eureka-Universe	
Población de FPG por predicado	50
Generaciones para FPG	50
Porcentaje de cruza	95 %
Porcentaje de mutación	5%

Tabla 5.8 Configuración usada para Universe y Eureka-Universe

En el capítulo 4.2, se explica el funcionamiento del algoritmo genético usado por Eureka-Universe nombrado EO-GSF.

Resultados

En la tabla 5.9 se observan los resultados promedio obtenidos para cada predicado con el que se experimentó, observando que hay un grado de diferencia entre los resultados obtenidos mediante Universe y los obtenidos mediante Eureka-Universe.

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

Tabla 5.9 Valores de verdad alcanzados por Universe y Eureka-Universe

Mediante los resultados obtenidos, se usó la prueba de wilcoxon, para determinar si estadísticamente existe diferencia entre uno y otro.

Para llevar a cabo esta prueba se usó la plataforma STAC (Rodríguez et al, 2015), la cual nos dice que estadísticamente existe evidencia para concluir que el algoritmo Eureka-Universe es superior a Universe, mediante la obtención de un P-Value de 0.0035.

Conclusiones del experimento

Mediante el uso del algoritmo genético nombrado EO-GSF, que lleva a cabo la optimización de predicados mediante el uso de FPG, s. se observa que es posible construir estados lingüísticos sin necesidad de la información de un experto u otras herramientas existentes para la creación de funciones de pertenencia.

También se observa que el algoritmo EO-GSF logró mejores resultados que los encontrados por medio de Universe, sin embargo, en los predicados más sencillos, Universe mostro un mejor desempeño.

5.3 Experimento 3 Solución del problema de order picking mediante LDC

Objetivo: Resolver el problema de integración de pedidos "Order Picking", mediante el uso de LDC y FPG, s. así como verificar la interpretabilidad de los resultados.

Configuración del experimento

En la tabla 5.10 se observa un ejemplo del contenido de la instancia order.txt, que es la usada en el siguiente experimento.

vol 1	sp dist 1	aisles 2	vol 2	sp dist 2	batched
1	37.18	1	1	105.17	1
1	37.18	2	3	117.2	0
1	37.18	1	2	36.63	1
1	37.18	1	2	6.66	1

Tabla 5.10 Ejemplos del contenido de Order.txt

Para este caso llevaron a cabo dos actividades de descubrimiento. La primera actividad es encontrar un predicado que implique la integración de un segundo pedido a un lote actual, la estructura para esta tarea se define como:

"*" IMPLICA "*integrar pedidos*"

Donde IMPLICA es un operador de lógica difusa compensatoria.

La segunda acción que se realiza es buscar un predicado que implique que no se lleva a cabo la integración del pedido en un lote, al igual que en el anterior la estructura se define de forma que el predicado no sea integrado.

"*" IMP "no integrar pedidos".

A partir de estas búsquedas se seleccionan catorce de los predicados construidos y se evalúa su eficacia mediante un conjunto de prueba.

Resultados

En la tabla, 5.11 se muestra un conjunto de predicados descubiertos y su efectividad al ser evaluados mediante un conjunto de prueba.

Predicados	Éxitos	Porcentaje de éxito
"sp dist 2 C" **AND** "added aisles C" **AND** "common aisles A" **IMPLY** "it is not batched"	44	97.78%
"sp dist 1 C" **AND** "added aisles B" **IMPLY** "it is not batched"	42	93.33%
"sp dist 1 B " **AND** "vol 1 C" **AND** "added aisles B" **AND** "aisles 1 B" **AND** "added aisles A" **AND** "vol 2 C" **IMPLY** "it is not batched"	41	91.11%
"common aisles C" **AND** "vol 2 B" **AND** "added aisles B" **IMPLY** "it is not batched"	42	93.33%
"vol 2 B" **AND** "added aisles A" **AND** "batch distance B" **IMPLY** "it is not batched"	41	91.11%
"added aisles A" **AND** "aisles 1 C" **IMPLY** "it is not batched"	42	93.33%
"added aisles A" **AND** "aisles 1 C" **IMPLY** "it is not batched"	42	93.33%
"vol 2 A" **AND** "added aisles C" **AND** "common aisles B" **AND** "sp dist 2 C" **AND** "added aisles A" **AND** "common aisles A" **IMPLY** "it is not batched"	42	93.33%
“vol 2 A" **AND** "added aisles A" **IMPLY** "it is not batched"	44	97.78%
"aisles 2 B" **OR** "added aisles B" **IMPLY** "it is not batched"	43	95.56%
added aisles A" **AND** "vol 1 A"**IMPLY** "it is not batched"	43	95.56%
"added aisles B" **AND** "sp dist 1 A" **IMPLY** "it is not batched")	43	95.56%
"common aisles A" **AND** "sp dist 1 A" **AND** "added aisles A" **IMPLY** "it is not batched"	43	95.56%
"sp dist 2 A" **AND** "added aisles A" **AND** "aisles 2 C" **IMPLY** "it is not batched"	41	91.11%

Tabla 5.11 Predicados lógicos descubiertos para la integración de pedidos

Como es posible observar los predicados con mejor porcentaje de solución, son aquellos en los que el resultado implica no integrar el objeto al pedido.

De acuerdo con la tabla 5.11 el predicado con mejores resultados obtenidos es el siguiente:

"sp dist 2 C" **AND** "added aisles C" **AND** "common aisles A" **IMPLY** "it is not batched"

En la tabla 5.12 se observan los valores de los parámetros γ, β y m de la FPG para el predicado optimizado.

Etiqueta lingüística	atributo	Función de membresía	Parámetros de la función		
			ϒ	β	*M*
sp dist 2 C	sp dist 2	FPG	96.7524717	20.6009427	0.95879975
common aisles A	common aisles	FPG	1.7689514	0.01233985	0.916713
added aisles C	added aisles	FPG	1.24203125	0.22520622	0.98364062

Tabla 5.12 Atributos optimizados para los estados lingüísticos usados para el problema de order picking

De acuerdo con los resultados, el predicado descubierto se interpreta de la siguiente manera:

si la distancia recorrida para recoger el segundo pedido supera los 96 metros, y los pasillos comunes en el primer y segundo orden son dos o más, y los pasillos adicionales visitados para recoger hasta el segundo orden es más de uno, entonces implica que la segunda orden no se integra con la primera.

Conclusiones del experimento

De acuerdo con lo anterior observamos que el uso de la metodología de lógica difusa compensatoria es aplicable al problema de integración de pedidos, y que gracias a la capacidad que tiene la lógica difusa compensatoria para entregar información, podemos usar esa capacidad para interpretar los resultados, generando así predicados altamente comprensibles para el usuario final.

5.4 Experimento 4: Inferencia usando LDC

Objetivo: mediante el uso de la lógica difusa compensatoria arquimediana (LDCA) se busca reproducir el experimento del reporte técnico de Espín y

González (2018) en el cual mediante el predicado Mcv ^ Alkphos ^ Sgpt ^ Sgot ^ Gammagt ↔ Driks, se realizan operaciones de inferencia, en el cual cada uno de sus atributos se define mediante una FPG arquimediana.

Configuración del experimento

Mediante el uso de la instancia Bupa.txt vista en la tabla 5.13, se busca optimizar el predicado Mcv ^ Alkphos ^ Sgpt ^ Sgot ^ Gammagt ↔ Driks usando el algoritmo genético EO-GSF, posteriormente se usara un conjunto de prueba, mediante el cual se buscara inferir los datos.

Mcv	Alkphos	Sgpt	Sgot	Gammagt	Driks
85	92	45	27	31	0
85	64	59	32	23	0
86	54	33	16	54	0
91	78	34	24	36	0
87	70	12	28	10	0
98	55	13	17	17	0

Tabla 5.13 contenido de la instancia Bupa

En la tabla 5.14 se observa la configuración los algoritmos genéticos usados para la optimización del predicado y para la inferencia, en ambos casos el algoritmo genético se configuro con los mismos parámetros.

Configuración del Algoritmo de optimización y del algoritmo de inferencia.	
Población	300
Generaciones	200
% cruza	0.6

%mutación	0.4

Tabla 5.14 Configuración de el algoritmo

El conjunto de prueba a usar consta de 34 registros seleccionados aleatoriamente de la instancia Bupa.txt. Se realizaron dos experimentos diferentes, uno donde se buscaba un alto grado de alcohol en la sangre y el segundo donde se buscaba que el grado de alcohol en la sangre fuera bajo.

Resultados

En la tabla 5.15 se observan los resultados de la optimización del predicado cuando la optimización está orientada a un alto grado de alcohol en la sangre, a su vez en la tabla 5.17 se observa lo mismo, pero en este caso la optimización busca un bajo grado de alcohol en la sangre.

Por otro lado, en las tablas 5.16 y 5.18. Se muestran los resultados inferidos para cada uno de los 34 registros, en estas tablas se muestra el número de registro, el valor inferido y el valor real del registro.

	Mcv	Alkphos	Sgpt	Sgot	Gammagt	Drinks	ML	True value
α	0.05769	0.05675	0.05117	0.05677	0.0529	0.05957		
γ	83.81	43.05	55.5	60.22	54.97	50	4	0.8889
m	0.92207	0.90501	0.51620	0.13642	0.49227	1		

Tabla 5.15 Estados lingüísticos optimizados para alto grado de alcohol en la sangre

No	Inferido	Real	No	Inferido	Real	No	Inferido	Real	No	Inferido	Real
1	0.134	6	10	0.3	0.5	19	0.362	4	28	0.098	2
2	0.084	2	11	0.004	3	20	0.484	4	29	0.122	5
3	0.214	4	12	0.054	0.5	21	0.128	9	30	0.122	0.5
4	0.314	0.5	13	0.156	2	22	0.352	4	31	0.132	4
5	0.136	5	14	0.032	7	23	0.004	4	32	0.044	2
6	0.046	8	15	0.178	6	24	0.1	6	33	0.042	3
7	0.012	0	16	0.284	0.5	25	0.22	0.5	34	0.646	0.5
8	0.364	4	17	0.004	12	26	0.14	4			
9	0.406	8	18	0.4	3	27	0.084	0.5			

Tabla 5.16 Tabla comparativa de los datos inferidos contra los reales

	Mcv	Alkphos	Sgpt	Sgot	Gammagt	Drinks	ML	True value
α	0.05676	0.05053	0.05649	0.05231	0.0523	0.05967		
γ	55.14	38.62	52.06	14.37	91.27	50	2	0.7778
m	0.5937	0.4258	0.2570	0.5598	0.0744	0.0		

Tabla 5.17 Estados lingüísticos optimizados para bajo grado de alcohol en la sangre

No	Inferido	Real	No	Inferido	Real	No	Inferido	Real	No	Inferido	Real
1	0.06	6	10	0.122	0.5	19	0.032	4	28	0.372	2
2	0.252	2	11	0.35	3	20	0.312	4	29	0.062	5
3	0.46	4	12	0.336	0.5	21	0.222	9	30	0.098	0.5
4	0.008	0.5	13	0.162	2	22	0.092	4	31	0.078	4
5	0.696	5	14	0.106	7	23	0.34	4	32	0.014	2
6	0.088	8	15	0.222	6	24	0.002	6	33	0.238	3
7	0.18	0	16	0	0.5	25	0.048	0.5	34	0.354	0.5
8	0.056	4	17	0.34	12	26	0.136	4			
9	0.028	8	18	0.052	3	27	0.036	0.5			

Tabla 5.18 Tabla comparativa de los datos inferidos contra los reales

De acuerdo con lo reportado en Espín y González (2018) el mejor resultado obtenido en el predicado es de 0.8673 orientado a un grado de alcohol, mientras que en el experimento que se compara al suyo el resultado obtenido es de 0.8889, también se menciona que el tiempo de solución mediante el uso de SQP es del orden de horas.

Respecto a la inferencia, menciona que se hizo una comparación con weka, en la cual se superó a los algoritmos mostrados en la tabla 5.19.

Method	**MAE in half-pint**
The proposed method	0.31670
ZeroR method	2.6283
M5P	2.4065
Multilayer Perceptron	3.2768

Tabla 5.19 Resultados reportados en Espin y Gonzalez 2018.

Conclusiones del experimento

Se puede decir que la construcción de funciones de pertenencia a partir de una LDCA en general da la oportunidad de crear una gran variedad de funciones.

Sin embargo, en el presente, no se obtuvieron los resultados esperados, por lo que es necesario investigar más acerca de metodologías de inferencia.

Conclusiones

En los capítulos anteriores se ha estudiado a la lógica difusa compensatoria (LDC), se ha observado su similitud con la lógica difusa propuesta por el Dr. Lotfi Zadhe y se observan las diferencias metodológicas en ambas. Así también se ha comentado acerca de los resultados presentados por algunos investigadores con respecto a la LDC y en sus estudios han presentado a la LDC como una metodología lógica multivaluada, la cual puede representar el conocimiento de lenguaje natural en modelos lógicos que permiten evaluar problemas de decisión.

Así también se menciona su uso en el procedimiento de machine learning (ML), en el cual en un entorno de datos se buscan predicados lógicos difusos compensatorios que nos permitan identificar patrones de comportamiento y también realizar conexiones lógicas entre atributos, estas conexiones o la calidad de los predicados descubiertos es medido a través de una constante de universalidad.

Para realizar este proceso de ML diversos autores hicieron uso de varias metodologías heuristicas comparando sus resultados, de acuerdo a lo entendido mediante estos se entiende que en términos de tiempo el algoritmo con mejor funcionamiento es el algoritmo genético (GA), que si bien no ofrece el mejor resultado, sus resultados son competitivos.

Dicho lo anterior, se comprende el objetivo principal del desarrollo de esta tesis el cual es proveer un algoritmo evolutivo de descubrimiento de conocimiento mediante el uso de la LDC. Ya que el proceso de ML mediante un GA permite obtener buenos resultados en una cantidad aceptable de tiempo, también se entiende que el uso de una lógica multivaluada como la LDC nos permite modelar el conocimiento en un modelo que utiliza el lenguaje natural, lo cual permite que los resultados encontrados puedan ser altamente expresivos y ricos en su presentación, así como también ser capaces de usar el conocimiento de expertos para el modelo de solución.

Hablando de como se evalua la calidad del predicado, podemos decir que el valor de verdad a pesar de parecer un valor probabilístico, no se toma de la misma manera, por ejemplo, si estuviéramos evaluando un lote de leche y esperando determinar su frescura, un valor probabilístico de .8 quiere decir que 80 envases de cada 100 son frescos, por el contrario un avalor de verdad de .8

quiere decir que se observa que de acuerdo a sus características 80 partes de 100 de cada bote es leche fresca, lo cual nos permite ver que cualquiera de los botes esta en buen estado.

Por lo tanto se toma como objetivo encontrar predicados que tengan altos valores de verdad, lo cual nos permita determinar un grado de pertenencia alto, el cual nos permita decir que existe evidencia de que estos predicados pertenecen al conjunto buscado en un alto grado, sin embargo, hay que tomar en cuenta que quien determina el valor de predicado ideal es siempre el tomador de decisiones y por lo general pasando un grado de verdad de .5 se puede considerar que se comienza a cumplir con el objetivo deseado.

Si bien también hay que aclarar que este valor de verdad cuando es bajo no indica que un predicado descubierto sea malo, si no que simplemente de acuerdo con el conjunto con el que se esta comparando no se aprecia pertenencia, para ejemplificar lo dicho podríamos decir que si estuviéremos clasificando especies de flores de acuerdo a las características observadas por cada especie, un alto grado de verdad con una especie por ejemplo un valor de verdad de 0.9 quiere decir que esas características pertenecen a la especie en 90 partes de 100, sin embargo si tenemos un valor de verdad de 0.1 significa que las características pertenecen a la especie en 10 partes, lo cual permite declarar que no pertenece a la especie o que es una especie diferente.

De acuerdo con este aspecto se observa que aunque los valores de verdad altos nos permiten verificar que el grado de pertenencia de un predicado a un conjunto es alto, es también observable que los valores de verdad bajos también ofrecen conocimiento observado en el entorno de datos.

Otro de los objetivos mencionados en la tesis fue el de proveer un algoritmo que fuera competitivo en términos de valores de verdad. Lo cual es posible observarse en el desarrollo de los experimentos realizados, que nos permiten observar y comparar la calidad de los predicados descubiertos por la plataforma Universe contra la versión Eureka-Universe. La cual ofrece resultados competitivos en tiempos adecuados.

Y en el ultimo de los aspectos este provee un algoritmo que mediante el uso de la LDC, nos permite llevar a cabo el proceso de descubrimiento de conocimiento.

El desarrollo de la presente tesis, documenta la forma en que se llevo a cabo el desarrollo de cada uno de los aspectos que integran el proyecto principal, permitiendo al lector observar las características del algoritmo propuesto y las metodologías usadas para su construcción. Así también permite al lector generar nuevas ideas en cuanto al desarrollo del mismo.

Sin más que agregar se finaliza el presente con un agradecimiento especial a aquellos que se toman el tiempo de leer la presente redacción.

Bibliografía

A. Bouchet, J. Pastore, M. Brun y V. Ballarin. Lógica 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.*

Adrián Chao Bataller (2014), Metodología para la gestión del conocimiento y la toma de decisiones basado en lógica difusa compensatoria*., Instituto Superior Politécnico José Antonio Echeverría trabajo de maestría.*

Benito, T. y Duran, M. (2008), lógica Borrosa *Universidad de Carlos III*.

Cruz-Reyes, L., et al.(2015), Simplification of Decision Rules for Recommendation of Projects in a Public Project Portfolio. *In Melin, Patricia, Castillo, Oscar and Kacprzyk, Janusz (eds), Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization, Studies in Computational Intelligence, Springer International Publishing, 2015.*

Cruz-Reyes, L. et al.(2016) A Statistical Tool for Comparison of Heuristics. *(Eds.), Handbook of Research on Military, Aeronautical, and Maritime Logistics and Operations 2016.*

Cruz-Reyes L., et al.(2017), Incorporation of implicit decision-maker preferences in Multi-Objective Evolutionary Optimization using a multi-criteria classification method. *Applied Soft Computing, volume 50 2017*.

Delgado, T.(2005). Capacity-building: spatial data infrastructure readiness index – IDE, P*roceedings of 8th Un Regional Cartographic Conference for the Americ, 2005a.*

Dubois, D. y Prade, H.(1985), A review of fuzzy set aggregation connectives, *Information Sciences, 1985, vol. 36 (1-2), p.85-121, ISSN 0020-0255.*

Erick González Caballero, Rafael Alejandro Espín Andrade (2010). Solución de juegos cooperativos n-personales basada en lógica difusa compensatoria. *Revista investigación operacional, Vol., 31 , No. 1, 45-60.*

Erick González-Caballeroa, Rafael A. Espín-Andrade, Witold Pedryczc, Liliana Guerrero Ramos, (En proceso de publicación). Continuous Linguistic Variables for Data Mining.

Espín, R. y Fernández, E. (2009), La Lógica Difusa Compensatoria: Una Plataforma para el Razonamiento y la Representación del Conocimiento en un Ambiente de Decisión Multicriterio,″ *Análisis Multicriterio para la Toma de Decisiones: Métodos y Aplicaciones., Coedición: editorial Plaza y Valdés / Editorial Universidad de Occidente,*

Gil Guzmán, Kety M.; Chao Bataller, Adrián; Muñoz Gutiérrez, Salvador; Espín Andrade, Rafael A (2010). Aplicación de la lógica difusa compensatoria en la selección de ofertas de armaduras ópticas. *Ingeniería Industrial, vol. XXXI, núm. Febrero, 2010, pp. 1-9*

Ivet Cabanas Moreda(2007), Sistema de Ayuda a la Decisión basado en la Lógica Difusa Compensatoria. Aplicación en el Grupo Empresarial de la Industria Portuaria. I*nstituto Superior Politécnico "José Antonio Echeverría" Ciudad de la Habana.*

Jesús Cejas-Montero (2011). La lógica difusa compensatoria. *Ingeniería Industrial/ISSN 1815-5936/Vol. XXXII/No. 2/mayo-agosto/2011*

Rafael Alejandro Espín Andrade, Adolfo Alberto Vanti (2005). Administración lógica: un estudio de caso en una empresa de comercio exterior. *BASE – Revista de Administración y contabilidad de Unisino.*

Rafael Alejandro Espín Andrade, Eduardo Fernández González y Erick González Caballero (2011). Un sistema lógico para el razonamiento y la toma de decisiones: la lógica difusa compensatoria basada en la media geométrica. *Revista investigación operacional VOL., 32 , NO. 3, 230-245*.

Taymi Ceruto Cordovés, Alejandro Rosete Suárez, Rafael Espín Andrade (2010). *O*btención de predicados difusos a partir de datos utilizando metaheurísticas*, RIIO, Edición N° 1, diciembre de 2010, Año 1, pp. 29 –37, ISSN 2145 – 9517.*

Taymi Ceruto Cordovés, (2012) Método para obtener predicados difusos a partir de datos utilizando metaheurísticas*., Instituto Superior Politécnico José Antonio Echeverría (CUJAE). – Tesis (Maestría).*

Taymi Ceruto Cordovés, Orenia Lapeira Mena, Alejandro Rosete Suárez, Rafael Espín Andrade (2013). Discovery of fuzzy predicates in database*, Eureka-2013. Fourth International Workshop Proceedings.*

Taymi Ceruto, Orenia Lapeira, Annika Tonch, Claudia Plant,Rafael Espín, and Alejandro Rosete (2014), Mining Medical Data to Obtain Fuzzy Predicates. *M. Bursa et al. (Eds.): ITBAM 2014, LNCS 8649, pp. 103–117*

T. Ceruto, O. Lapeira and A. Rosete. Medidas de calidad para predicados difusos en forma normal conjuntiva y disyuntiva. *Ingeniería e investigación Vol. 34 No. 3, diciembre - 2014 (63-69).*

Zadeh, L. (1965). Fuzzy Sets. Information and Control, *1965, vol. 8, p.338-353, ISSN 0019-9958*