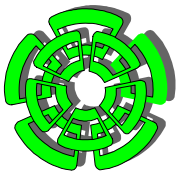


CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Representación y aprendizaje de conocimiento con redes de Petri difusas

TESIS QUE PRESENTA EL:
Ing. Jair Cervantes Canales

PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS

EN LA ESPECIALIDAD DE
CONTROL AUTOMÁTICO

DIRECTORES DE TESIS:

Dr. Wen Yu Liu
Dr. Alejandro J. Malo Tamayo

México, D.F.

MARZO del 2005.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Antecedentes

Las redes de Petri (RdP) fueron introducidas en la literatura en la tesis doctoral de Carl Adam Petri [33] como una herramienta para simular las propiedades dinámicas de sistemas complejos mediante modelos gráficos de procesos concurrentes. Desde entonces su estudio y desarrollo han tenido un auge realmente vigoroso debido fundamentalmente a las numerosas aplicaciones que se les ha encontrado, las cuales incluyen diversas áreas del conocimiento y de la técnica. Se ha demostrado que las RdP son un instrumento adecuado para la representación y análisis de ciertos sistemas, ya que estas tienen la habilidad de representar y analizar de una forma fácil sincronización y concurrencia, donde varios procesos que evolucionan simultáneamente son parcialmente independientes.

Las RdP proveen un conjunto de herramientas con gran capacidad de descripción y operación que son ocupadas para representar *reglas difusas* (reglas que describen la relación difusa entre dos proposiciones). Un conjunto entero de reglas de producción es llamado *una base de reglas*. Además al formalismo de reglas de producción el sistema provee un significado para definir los objetos referidos en las reglas de producción, llamado *declaración de dominio*. La base de reglas y la declaración del dominio constituyen juntos una *base de conocimiento* (KB) del sistema difuso.

En [22] las RdP son usadas para modelar bases de reglas, relacionando simplemente algunos de los elementos (función de marcado) y características (básicamente lugares y transiciones) del formalismo de Petri con los elementos básicos de una KB (proposiciones, grados de verdad e implicaciones). Cuando la complejidad de los sistemas KB incrementa, nuevos elementos son incluidos en la definición inicial de una RdP, los cuales permiten a una RdP representar adecuadamente los rasgos que caracterizan una KB. Este procedimiento nos lleva a la representación de un nuevo modelo el cual es llamado red de Petri difusa (RPD). Las RdP tienen una cualidad inherente en representar lógica de una forma visual e intuitiva y las RPD toman todas estas ventajas de las RdP. De esta manera, la forma de razonamiento de un sistema experto puede ser reducido a simples árboles de nodos si algoritmos de razonamiento basados en RPD son empleados como máquinas de inferencia [7],[5], [6]. Varias generalizaciones de las redes de Petri difusas han sido propuestas en la literatura [47], [48], [3], [23], [30] con el propósito de considerar ciertos problemas concretos vinculados con la representación del conocimiento, razonamiento difuso y teoría de decisión.

1.2. Motivación

Las RPD pueden representar perfectamente reglas de producción difusas con pesos [6][48]. Sin embargo, los pesos en estas no pueden ajustarse de acuerdo a la actualización de conocimiento, en otras palabras, estas no tienen la habilidad de aprender. Por otro lado, las redes de Petri difusas adaptativas [42] poseen la habilidad de aprender de una forma parecida a las redes neuronales. Sin embargo, el conjunto de valores (valores de verdad, factores de certeza, pesos y valores frontera) asociados en la red de Petri difusa para generar un razonamiento son números en [0,1]. Es decir, son asociados unicamente pares $(x, (\mu(x))$ -elemento y grado de pertenencia-) como pesos, valores de verdad, valores frontera y factores de certeza a los lugares y transiciones de la red de Petri (ver figura 1(a)). Por lo cual, es impractico listar todos los pares definiendo el grado de pertenencia [16] de cada elemento al conjunto difuso. Una forma más concreta de definir el grado de pertenencia es expresarla como una función de membresía (función gaussiana, función campana, número difuso

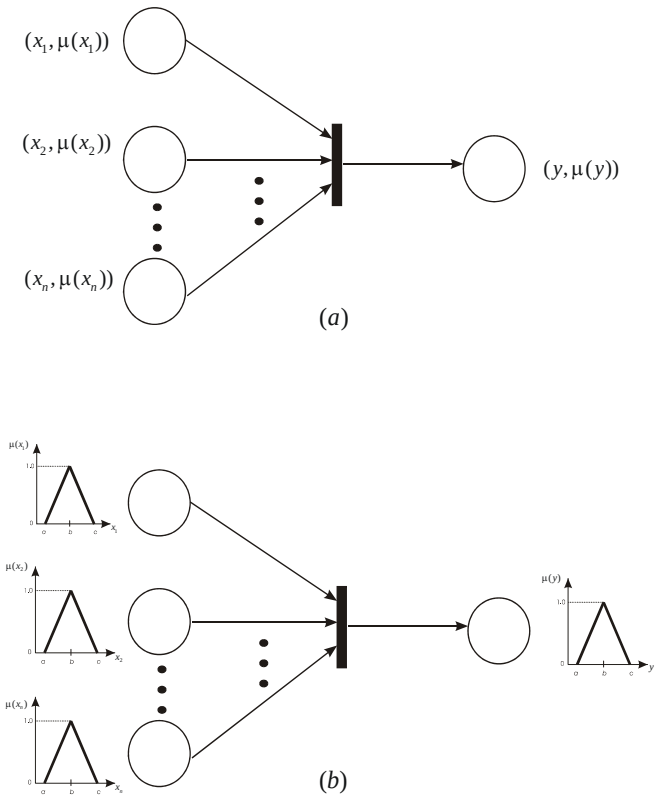


Figura 1.1: (a) Razonamiento de RPD con grados de pertenencia. (b) Razonamiento de RPD con funciones de membresía.

triangular, número difuso trapezoidal, etc.). Si podemos asociar una función de membresía como un valor de verdad a una red de Petri difusa (ver figura 1(b)), entonces el proceso de razonamiento es mucho más concreto y conveniente.

1.3. Planteamiento del Problema

En el trabajo de esta tesis nos ocupamos del modelado y aprendizaje de redes de Petri difusas con funciones de membresía, especificamente números difusos triangulares [20], [16] empleando los formalismos de redes neuronales y redes de Petri.

Los objetivos principales de este proyecto son:

1) Elaborar un algoritmo de razonamiento con propagación hacia adelante y hacia atrás mediante números difusos triangulares utilizando redes de Petri difusas.

2) Elaborar el algoritmo de razonamiento con propagación hacia adelante y valores frontera utilizando números difusos triangulares.

3) Elaborar algoritmos de aprendizaje de pesos mediante redes de Petri difusas con pesos adaptativas modificando los algoritmos de aprendizaje de Widrow Hoff y *Backpropagation (BP)* para trabajar con números difusos triangulares.

1.4. Organización

La tesis esta constituida por cinco capítulos los cuales están organizados de la siguiente forma:

Capitulo 2. Se da una introducción a los conceptos básicos de redes de Petri y Redes Neuronales, también se dan las nociones básicas de conjuntos difusos, funciones de membresía y reglas de producción difusas con la cual se establecen los principios para contruir una nueva estructura para la representación de conocimiento, razonamiento difuso y aprendizaje de conocimiento.

Capitulo 3. Se presentan tres algoritmos de razonamiento difuso. El Primero es un algoritmo de razonamiento difuso hacia atrás con pesos originalmente propuesto en [5]. El algoritmo propuesto en esta tesis es modelado con cuatro tipos de reglas de razonamiento. El segundo algoritmo fue originalmente propuesto en [6] este algoritmo de razonamiento difuso hacia adelante es modelado con cuatro reglas de razonamiento, la diferencia radica en que S.M. Chen calcula el valor de verdad final como el máximo de todas las transiciones disparadas y en el algoritmo propuesto aquí, se calcula el valor de verdad mediante el centro de gravedad de las transiciones disparadas. El tercer algoritmo, se modela el algoritmo con cuatro reglas de razonamiento, este fue propuesto originalmente en [42] y emplea tres tipos de reglas de razonamiento.

Los tres algoritmos de razonamiento difuso propuestos son modificados para trabajar con

números difusos triangulares.

Capitulo 4. Se propone un método de aprendizaje de pesos modificando los algoritmos de Widrow Hoff y el algoritmo *backpropagation* (BP) utilizando redes de Petri difusas con pesos adaptativas, donde los pesos asociados a los lugares son números difusos triangulares.

Capitulo 5. Se desarrollan algunos ejemplos utilizando los algoritmos de razonamiento y aprendizaje propuestos en los capitulos 3 y 4.

Capítulo 2

Conceptos Basicos

En este capitulo introducimos los fundamentos de redes de Petri, incluyendo la definición, terminología básica, reglas de disparo de transiciones, propiedades y métodos de análisis. Por otro lado, damos los fundamentos de las redes neuronales y los tipos de aprendizaje. En este mismo orden, introducimos los principales conceptos y nociones matemáticas de conjuntos difusos, primero vemos los conjuntos difusos como una generalización de conjuntos clásicos generalizando el rango de la función de membresía (o función característica) de {0,1} a un número real en el intervalo [0,1]. Varios conceptos básicos de conjuntos difusos como normalidad, convexidad y números difusos son definidos. Por ultimo introducimos los conceptos de principio de extensión y relaciones difusas así como variables lingüísticas, los cuales son usados en razonamiento difuso.

Ya que la investigación en redes de Petri, redes neuronales y conjuntos difusos es demasiado extensa, es imposible cubrir todos los aspectos de los desarrollos actuales en estos campos. Por lo tanto, el objetivo de este capitulo es proveer una introducción concisa y un resumen de los conceptos básicos centrales para el estudio de estos campos. El material cubierto en este capitulo será usado en los siguientes capítulos.

2.1. Redes de Petri

Las redes de Petri fueron introducidas en la literatura en la tesis doctoral de Carl Adam Petri [33]. Las Redes de Petri son una herramienta gráfica y matemática de modelado [26][32][37] para la descripción formal de sistemas cuya dinámica se caracteriza por concurrencia, sincronización, exclusión mutua y conflictos, las cuales son características típicas de sistemas distribuidos.

El modelado de redes de Petri tiene dos características principales interesantes. Primero, es posible visualizar su comportamiento como concurrencia, paralelismo, sincronización y recursos compartidos. Segundo, existen varios métodos para el análisis de redes de Petri.

Las redes de Petri han sido desarrolladas y extendidas durante años, por lo que varias clases de redes de Petri han sido definidas. Entre ellas, redes de Petri coloreadas por Kurt Jensen[17][18], temporizadas [2][34], estocásticas [9], algebraicas [19], continuas e híbridas [1], difusas [30], etc. En este capitulo hacemos una breve introducción a la teoría de las redes de Petri.

2.1.1. Definición Formal y Fundamentos

Las redes de Petri (RP) se describen como una herramienta de naturaleza gráfica para el diseño y análisis de sistemas dinámicos de eventos discretos [26]. Una red de Petri se representa gráficamente por un *grafo dirigido bipartito*. Los dos tipos de nodos, *lugares y transiciones* (ver figura 2.1) representan, las variables que definen el estado del sistema (lugares) y a sus transformadores (transiciones). Los lugares se representan por círculos, las transiciones por barras y el marcado M se representa por una distribución en los lugares denominados marcas. Una marca se representa gráficamente por un punto en el interior del círculo que define el lugar que lo contiene. Los lugares y transiciones se conectan por arcos dirigidos. Un arco dirigido de un lugar P_i a una transición T_j define un lugar de entrada de la transición. Múltiples entradas a una transición son indicadas por múltiples arcos desde el lugar de entrada a la transición. Un lugar de salida es indicado por un arco desde la transición al lugar. Análogamente, múltiples salidas son representadas por múltiples arcos.

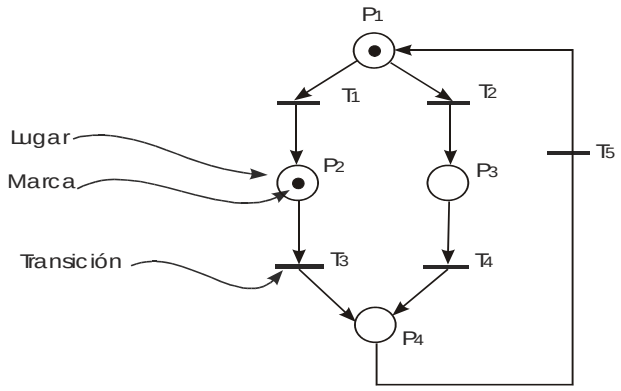


Figura 2.1: Lugares, transiciones y señales en una red de Petri.

El marcado de una red, M, se define por un vector columna en donde los elementos son el número de marcas contenidas en los lugares. El marcado define el estado de la red de Petri. Para la red de Petri dada en la figura 2.1 tenemos: Lugares: $P = \{P_1, P_2, P_3, P_4\}$; Transiciones : $T = \{T_1, T_2, T_3, T_4, T_5\}$; Marcado $m_1 = m_2 = 1$; $m_3 = m_4 = 0$; $M = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T$

Definición Formal.

Definición 2.1 *(Red de Petri Ordinaria) Una Red de Petri Ordinaria [37] (RPO), N, es una cuádruplo $N =< P, T, Pre, Post >$, donde:*

$P = \{p_1, p_2, ..., p_m\}$ es un conjunto finito y no vacío de lugares;

$T = \{t_1, t_2, ..., t_n\}$ es un conjunto finito y no vacío de transiciones;

$P \cap T = \varnothing$ y $P \cup T \neq \varnothing$;

$Pre : P \times T \rightarrow \{0, 1\}$ es el conjunto de lugares de entrada a T;

$Post : T \times P \rightarrow \{0, 1\}$ es el conjunto de lugares de salida de T.

Una RPO marcada es un par $N_m = \langle N, M_0 \rangle$ en el cual N es una red de Petri Ordinaria y M_0 es el marcado inicial.

Definición 2.2 *(Representación matricial) Una RdP N se encuentra definida matricialmente por medio de dos matrices [37]. Sea $n = |P|$ (número de lugares de P) y $m = |T|$ (número de transiciones de T). Se denomina:*

Matriz de incidencia previa a la matriz $C^- = [c_{ij}^-]_{n \times m}$ en la que $c_{ij}^- = Pre(p_i, t_j)$.

Matriz de incidencia posterior a la matriz $C^+ = [c_{ij}^+]_{n \times m}$ en la que $c_{ij}^+ = Post(t_i, p_j)$.

Matriz de incidencia de $N : C = C^+ - C^-$.

Es decir, en las matrices de incidencia las filas representan a los lugares y las columnas a las transiciones.

Ejemplo 1. De la figura 2.1 tenemos que:

$$C^- = Pre(p_i, t_j) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \quad C^+ = Post(t_i, p_j) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Definición 2.3 *Sea N una RdP, donde $t \in T$ y $p \in P$ (según definición 2.1). Se definen los siguientes conjuntos:*

- *Conjunto de lugares de entrada a t : $^\bullet t = \{p \in P \mid Pre(p, t) > 0\}$*
- *Conjunto de lugares de salida de t : $t^\bullet = \{p \in P \mid Post(t, p) > 0\}$*
- *Conjunto de transiciones de entrada a p : $^\bullet p = \{t \in T \mid Post(t, p) > 0\}$*
- *Conjunto de transiciones de salida de p : $p^\bullet = \{t \in T \mid Pre(p, t) > 0\}$*

Definición 2.4 *(Red pura) Una RdP N es una red pura [26] si no existe ninguna transición que tenga un lugar que sea al mismo tiempo de entrada y salida de la transición:*

$$\forall t_j \in T \ , \ \forall p_i \in P, Pre(p_i, t_j) Post(t_j, p_i) = 0 \tag{2.1}$$

Comportamiento Dinámico.

Definición 2.5 *(Transición habilitada) Una transición $t \in T$ está habilitada para un marcado M dado, sii $\forall p \in {}^\bullet t$ se verifica $M(p) \geq Pre(p,t)$.*

Definición 2.6 *(Grado de habilitación) El grado de habilitación de una transición indica el máximo número que la transición puede ser disparada concurrentemente.*

Definición 2.7 *(Regla de evolución del marcado) Si t está habilitada para un marcado M entonces t puede dispararse. En la operación se alcanza un nuevo marcado M', y se denota por $M[t > M'$, el cual resulta de quitar Pre(p, t) marcas de cada lugar $p \in {}^\bullet t$ y añadir $Post(t,p)$ marcas a cada lugar $p \in t^\bullet$. El cambio en el marcado esta dado por la ecuación:*

$$M'(p) = M(p) - Pre(p,t) + Post(t,p), \forall p \in P \tag{2.2}$$

Subclases de Redes de Petri Ordinarias.

Definición 2.8 *(Grafo de estados) Un grafo de estado [26] (GE) o máquina de estados (ME) es una RPO que cumple:*

$$\forall t \in T, |{}^\bullet t| = |t^\bullet| = 1 \tag{2.3}$$

Definición 2.9 *(Grafo marcado) Un grafo marcado [26] (GM) o grafo de sincronización es una RPO que cumple:*

$$\forall p \in P, |{}^\bullet p| = |p^\bullet| = 1 \tag{2.4}$$

Definición 2.10 *(Red de libre elección) Una RdP de libre elección [26] (RLE) es una RPO que cumple:*

$$\forall p \in P, \ si \quad |p^\bullet| > 1, \text{ entonces } \forall t \in p^\bullet, |{}^\bullet t| = 1 \tag{2.5}$$

Definición 2.11 *(Red simple) Una RPO simple [26] (ROS) es una RdP que cumple:*

$$p_1^\bullet \cap p_2^\bullet \neq \varnothing \Rightarrow p_1^\bullet \subseteq p_2^\bullet \ o \ p_1^\bullet \supseteq p_2^\bullet \tag{2.6}$$

para toda $p_1, p_2 \in P$.

2.1.2. Propiedades de las Redes de Petri.

Vivacidad

Definición 2.12 *Una transición t está viva [32] para un marcado inicial dado* M_0*, sii existe una secuencia de disparos a partir de un marcado M sucesor de* M_0 *que comprenda a t :* $\forall M \in M(R, M_0) \quad \exists \sigma : M \xrightarrow{\sigma} M' \ \ tal\ que \ \ t \subset \sigma$.

Una RdP marcada está viva para M_0 sii todas sus transiciones son vivas para M_0. Se puede decir que la propiedad de vivacidad significa la ausencia en el conjunto de alcanzabilidad de un marcado en el que la red se bloquee totalmente, ya que, para que esté viva, todas sus transiciones deben ser disparables desde cualquier marcado alcanzable. Se dice que una RdP marcada está parcialmente viva para M_0 si, tomando como punto de partida cualquier marcado alcanzable a partir de M_0, existe al menos una transición disparable y otra transición no viva. Toda RdP marcada parcialmente viva tiene la posibilidad de evolución global, independientemente de que existan transiciones que no puedan ser disparadas.

Ciclicidad.

Definición 2.13 *Se dice que una RdP posee un comportamiento globalmente cíclico para Mo si existe una secuencia de disparos que permite alcanzar el marcado inicial Mo a partir de cualquier marcado M alcanzable a partir de Mo:*

$$\forall M \in M(R, M_0), \quad \exists \sigma \ tal\ que \ \ M \xrightarrow{\sigma} M_0. \tag{2.7}$$

La ciclicidad o reversibilidad [26] de una RdP marcada garantiza que no existen subconjuntos finales de estados (marcados). Un subconjunto final de estados (marcados) contiene estados (marcados) mutuamente alcanzables entre sí y tales que el estado inicial (marcado inicial) no es alcanzable a partir de ninguno de ellos.

Acotamiento. El significado de esta propiedad es el de asegurar que el sistema que una red representa, posee un *número finito de estados* (si suponemos que cada lugar de la red representa a una variable de estado del sistema y su marcado el valor de dicha variable). Luego la propiedad de acotamiento determina *la finitud del número de estados del sistema* representado por una RdP.

Definición 2.14 *Un lugar p es k-acotado para M_0 sii existe un número entero k tal que $M(p) \leq k$ para cualquier marcado $M \in M(R, M_0)$. Se denomina cota del lugar p al menor entero k que verifica la desigualdad anterior.*

Definición 2.15 *Una RdP marcada es k-acotada para M_0 sii todos sus lugares son k-acotados para M_0:*

$$\forall p \in P \ y \ \ \forall M \in M(R, M_0), M(p) \leq k. \tag{2.8}$$

Merece una consideración especial la *1-acotación*. Si una RdP es 1-acotada para M_0, su marcado es binario (un lugar está o no está marcado) y se dirá que la RdP es binaria para M_0. *Una red segura*, es una RdP 1-acotada. Una RdP es *estructuralmente acotada* si es acotada para cualquier marcado inicial y finito.

Conservatividad Las marcas de una red se pueden entender como recursos del sistema. Normalmente los recursos de un sistema ni se crean ni se destruyen. Cuando las marcas se conservan, tras el disparo de una secuencia de transiciones, se dice que la red es conservativa.

Definición 2.16 *(Red de Petri estrictamente conservativa) Sea $R = (P, T, Pre, Post, M_0)$ se dice que es estrictamente conservativa sii $\forall M' \in M(R, M_0), \Sigma_i M' \ (p_i) = \Sigma_i M(p_i), \ p_i \in P$.*

Esto es, se ha de mantener el número de marcas para cualquier marcado de la red. La definición anterior implica que el número de entradas ha de coincidir con número de salidas, es decir: $(|I(tj)|) = |O(tj)|)$, para cada transición disparable.

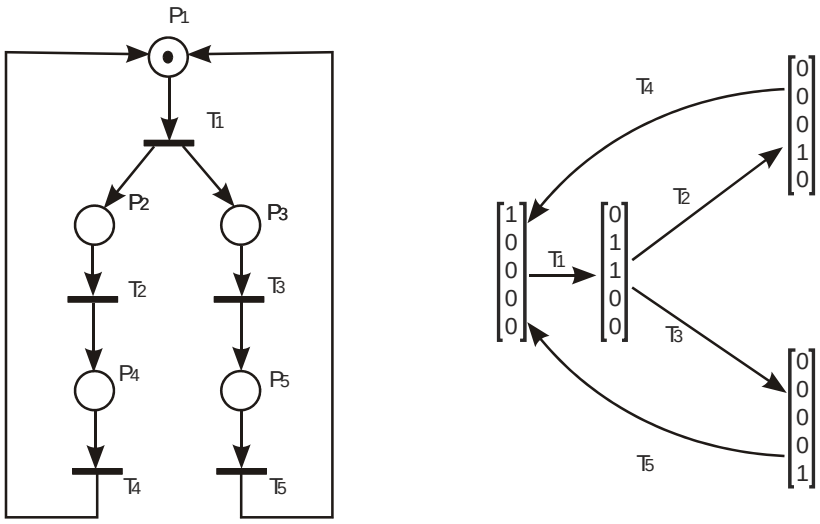


Figura 2.2: Red de Petri y su grafo de alcanzabilidad.

Alcanzabilidad. La alcanzabilidad es una base fundamental para estudiar las propiedades dinámicas de cualquier sistema. al dispararse una transición habilitada, esta cambiará la distribución de las señales (marcado). De esta forma, de una secuencia de disparos resultará una secuencia de marcados, luego un marcado M_n es alcanzable a partir de M_0, si existe una secuencia de disparos que a partir de M_0 nos lleve a M_n. Una secuencia de disparos la denotaremos por $\sigma = t_1, t_2, ..., t_n$. en este caso M_n es alcanzable desde M_0, sii $\exists\sigma$ $t.q.$ $M_0 \left[\sigma\right\rangle M_n$.

2.1.3. Métodos de Análisis.

Las técnicas para el análisis de RdP (análisis cualitativo de redes) se clasifican normalmente en tres grupos.

Técnicas Enumerativas. En primer lugar se tienen las técnicas enumerativas [32][26]. Se basan en la generación del grafo de alcanzabilidad para sistemas limitados o del grafo de cobertura para sistemas no limitados [8]. Estas técnicas se pueden aplicar en teoría,

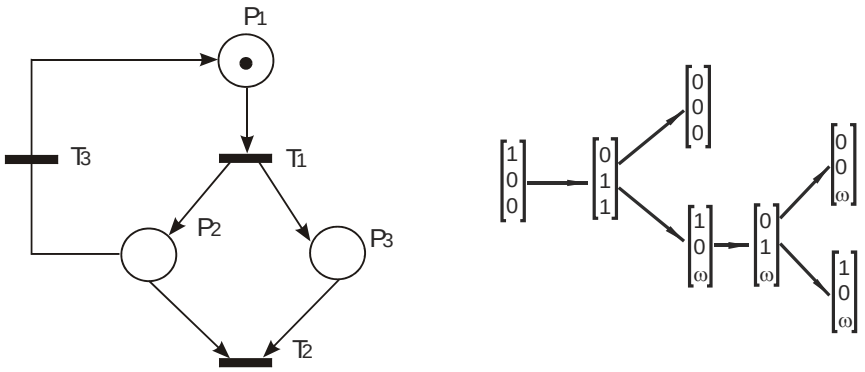


Figura 2.3: Red de Petri y su grafo de cobertura.

pero en la práctica están limitadas a sistemas pequeños debido a su elevada complejidad computacional. En este grafo los nodos corresponden al marcado alcanzable y los arcos corresponden al disparo de las transiciones. En la figura 2.2 se muestra una red de Petri y su grafo de alcanzabilidad (Grafo de marcados). Este grafo puede ser utilizado para mostrar que la red es segura, viva, reversible y que esta tiene dos componentes que se repiten T_1, T_2, T_4 y T_1, T_3, T_5.

En una red no limitada, el número de señales en un lugar puede ser infinito. Un lugar que puede llegar a tener un número infinito de señales se representa por ω. El grafo resultante es llamado grafo de cobertura (ver figura 2.3).

Técnicas de Transformación. En segundo lugar se tienen las técnicas de transformación. En este grupo de técnicas el objetivo es reducir el tamaño de los modelos mediante reglas de reducción que preserven las propiedades que se quieren estudiar. En la figura 2.4 puede observarse un conjunto sencillo de seis reglas de reducción que preservan vivacidad y k-limitación tomadas de [37]. Con este conjunto de reglas es posible reducir la complejidad del cálculo de la vivacidad y limitación de un sistema.

Técnicas Estructurales. En tercer lugar se tienen las técnicas estructurales [37]. En este grupo de técnicas el objetivo es obtener la máxima información del modelo utilizando

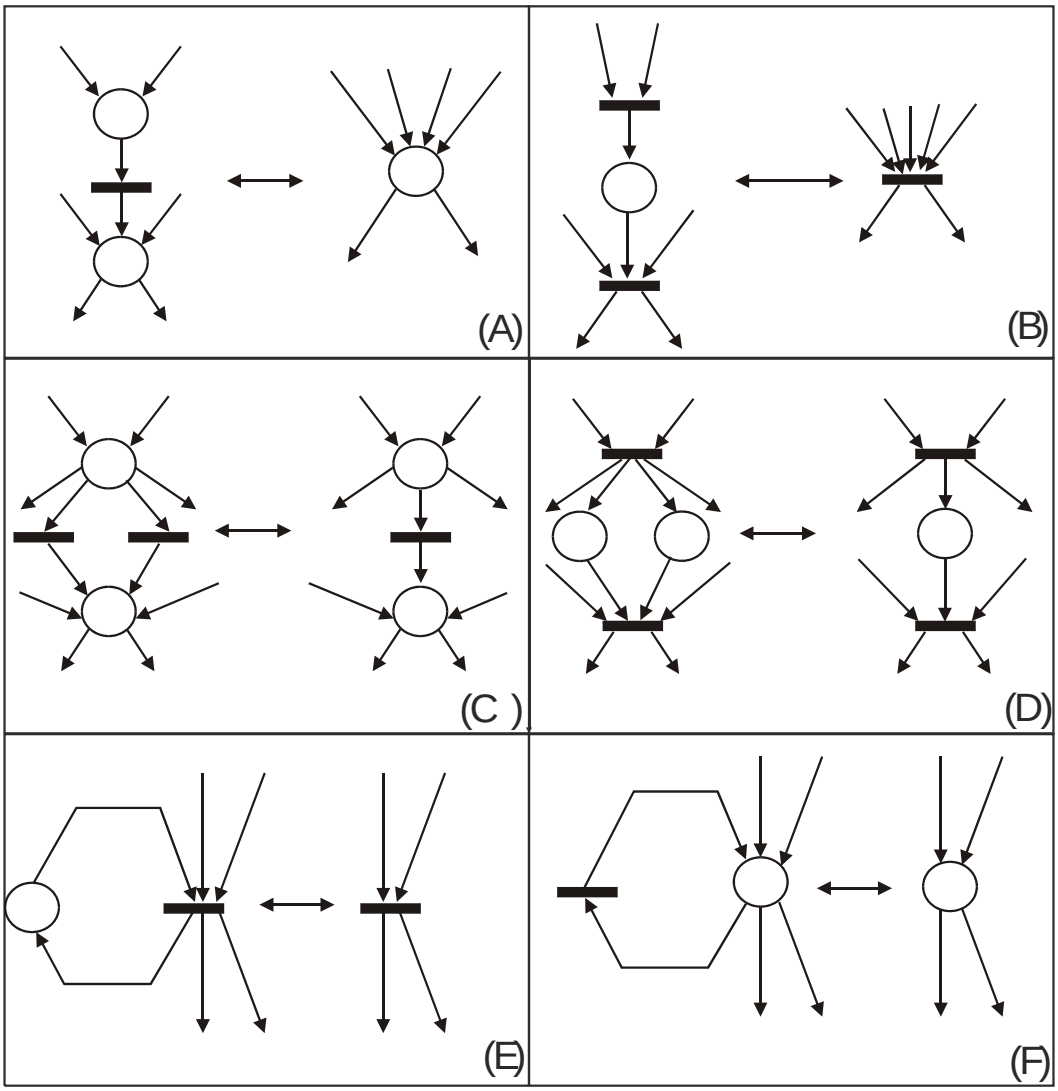


Figura 2.4: Reglas de reducción que presentan vivacidad y k-limitación.

únicamente su estructura y marcado inicial.

Los métodos de álgebra lineal son utilizados para determinar las propiedades de la red. La ecuación de estado de una red de Petri se define como sigue: el marcado M_k se define como un vector columna $m \times 1$. La $j - ésima$ entrada de M_k denota el número de señales en el lugar j inmediatamente después del $k - ésimo$ disparo en la secuencia de disparos. El $k - ésimo$ disparo o vector de control u_k es un vector columna de $n \times 1$, con $n - 1$ $0's$ y una entrada 1 en la $i - ésima$ posición, indicando el disparo de la transición i. Esto es, la $i - ésima$ fila de la matriz de incidencia C denota el cambio de marcado como el resultado del disparo de la transición i, luego la ecuación de estado para la red de Petri se escribe como sigue:

$$M_k = M_{k-1} + C^T u_k, \qquad k = 1, 2, 3, ... \tag{2.9}$$

El marcado de una red de Petri puede ser cambiado cada vez que una transición se dispara. Si no ocurre un bloqueo, el número de disparos es ilimitado. Sin embargo, no todos los marcados pueden ser alcanzados y no todas las secuencias de disparos pueden ser llevadas a cabo. Las restricciones son dadas por los invariantes de la red. Un marcado invariante es obtenido si la suma de los pesos de el marcado de un subconjunto de lugares en una red es siempre constante. Los lugares contenidos en este subconjunto son llamados *componentes conservativos* y el vector que contiene los pesos es *P-Invariante*. Si el disparo de una cierta secuencia de transiciones resulta en el mismo marcado como cuando inicio, la secuencia es llamada *componente repetitivo.* El vector característico de la secuencia de disparos es el *T-Invariante*.

2.2. Redes Neuronales

Una red neuronal artificial (RNA)[15] es una elemento capaz de procesar gran cantidad de información en forma paralela y distribuida, las cuales pueden almacenar conocimiento experimental y tenerlo disponible para su uso [13].

Las redes neuronales simulan el funcionamiento de las redes biológicas mediante modelos matemáticos, los cuales son implementados usando componentes electrónicos o mediante algún software en una computadora. Las redes neuronales tienen las siguientes características: habilidad de aprendizaje, habilidad de procesamiento en paralelo y la propiedad de aproximar cualquier función suave.

Las redes neuronales han sido estudiadas por muchos años, uno de los atributos de estas es que poseen la habilidad de aprender. En 1949 Donald Hebb [14] desarrollo una regla de aprendizaje fisiológica para la modificación de los pesos sinápticos, en 1957 Rosenblatt desarrollo el perceptrón, la mayor parte del trabajo es descrito en el libro "Principles of Neurodynamics" [35]. Uno de los resultados más significativos presentados en este libro fue la regla de entrenamiento del perceptrón. Widrow y Hoff [40] desarrollaron en 1960 el algoritmo del mínimo error cuadrático medio (LMS), usado para formular Adaline (elemento de aprendizaje adaptable) y en (1962) Madaline (múltiple adaline). En 1969 el entusiasmo por las redes neuronales fue incrementado un poco por la publicación del libro de Minsky y Papert "Perceptrons" [25]. En él los autores muestran que existe una interesante clase de problemas (aquellos que no son linealmente separables) los cuales el perceptrón de una sola capa no puede resolver. Grossberg fue poniendo los fundamentos de su teoría de resonancia adaptativa (ART por sus siglas en ingles) [4][11]. Fukushima desarrolló el cognitrón [10]. El algoritmo de *backpropagation* posee paternidad literaria múltiple como es notado por Grossberg en [12]. Este algoritmo fue descubierto por Werbos [39], redescubierto por Parker [29] y descubierto una vez más por Rumelhart, Hinton y Williams [36].

2.2.1. Fundamentos

Las neuronas son la base de las redes neuronales. Una neurona es una unidad procesadora de información que es fundamental para la operación de una red neuronal. La figura 2.5 muestra el modelo de una neurona, los elementos básicos de una neurona son descritos a continuación:

- Un vector de entrada el cual es denotado por $X, X = (x_1, x_2, ..., x_j)$ donde j es el

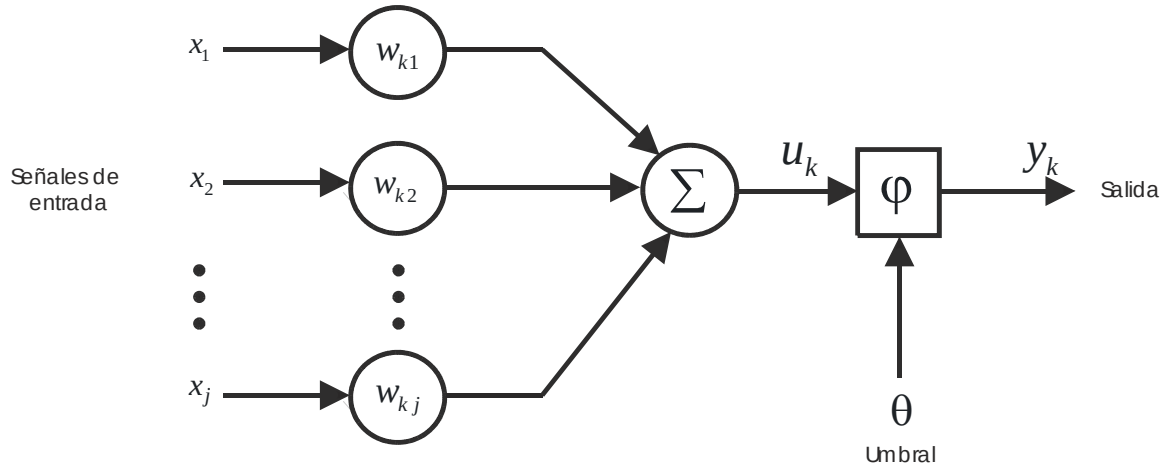


Figura 2.5: Red neuronal.

número de entradas a la RNA y a su vez la dimensión de X, este vector X son los datos con los que va operar la neurona, estas pueden ser dadas del medio ambiente o ser salidas de neuronas anteriores.

- Pesos sinápticos. Al ser capturados los datos de entrada estos son propagados a través de la red, en el proceso de propagación cada componente x_j del vector de entrada X es multiplicada por una variable w_{kj}, la cual aumenta o atenúa la señal de entrada, a w_{kj} se le conoce como peso sináptico o simplemente peso, estos pesos no tienen el mismo valor siempre sino que se van modificando según se requiera para tener un mejor desempeño, posteriormente puede haber una convergencia y entonces estar fijos. Cuando se habla de que una red es capaz de aprender se refiere al hecho de poder modificar sus pesos w_{kj} , el conjunto de pesos genera una matriz W , es decir, w_{kj} es la kj-ésima componente de la matriz de pesos W.

- Un operador suma para sumar las señales de entrada. Esto es, la adición de los productos $x_j * w_{kj}$.

- Una función de activación para limitar la amplitud de la salida de una neurona. Este limite usualmente es el intervalo unitario [0,1] o [-1,1].

En términos matemáticos, una neurona k puede ser descrita por una par de ecuaciones:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \tag{2.10}$$

y

$$y_k = \phi(u_k - \theta_k), \tag{2.11}$$

donde $x_1, x_2, ..., x_j$ son las señales de entrada, $w_{k1}, w_{k2}, ..., w_{kj}$ son los pesos sinápticos de la neurona k, u_k es la salida, mientras que θ_k es el umbral, $\phi(.)$ es la función de activación y y_k es la señal de salida de la neurona.

2.2.2. Aprendizaje de Redes Neuronales

La capacidad de aprendizaje adaptativo es una de las características más atractivas de las redes neuronales. Esto es, aprenden a llevar a cabo ciertas tareas mediante un entrenamiento. Las redes neuronales usan su capacidad de aprendizaje adaptativo para autoorganizar la información que reciben durante el aprendizaje y/o la operación. De acuerdo al tipo de aprendizaje, las redes se pueden subdividir en dos grupos:

- Redes con aprendizaje supervisado. Este tipo de redes se entrenan presentando para cada combinación de entradas, las salidas que se espera ellas produzcan. Los algoritmos de aprendizaje calculan pesos y sesgos nuevos a manera de minimizar el error entre la salida deseada y la obtenida.

- Redes no supervisadas. Los algoritmos de aprendizaje calculan nuevos pesos libremente. Estas redes se utilizan como clasificadores pues se caracterizan por asociar una combinación de entradas especificas con una sola salida.

2.3. Conjuntos Difusos

Un conjunto clásico es un conjunto con una cota rígida, este es caracterizado por y como la totalidad de sus números, objetos o elementos.

Los miembros de un conjunto clásico pueden ser determinados por alguna enumeración o alguna propiedad característica, de esta forma uno puede representar un conjunto A con elementos $a_1, ..., a_{15}$. $A = \{a_1, ..., a_{15}\}$ ó $A = \{a_i \mid 1 \leq i \leq 15\}$. En donde cada miembro a_i puede pertenecer o no pertenecer a un conjunto A. Sin embargo un *conjunto difuso,* es un conjunto sin una cota rígida [16], [49], [44], esto es, la transición de "pertenecer a un conjunto" a "no pertenecer a un conjunto" es gradual, esta transición *"suave"* es caracterizada por *"grados de pertenencia o función de membresía",* las cuales dan flexibilidad a los conjuntos difusos.

La difusividad no viene de la aleatoriedad de los miembros constituyentes de los conjuntos, sino de la naturaleza imprecisa e incierta de pensamientos y conceptos abstractos.

2.3.1. Definiciones Básicas y Terminología.

La lógica difusa asocia incertidumbre a la estructura de un conjunto de datos [21]. Los elementos de un conjunto difuso son pares ordenados que indican el valor del elemento y su grado de pertenencia. Para un conjunto difuso $A = \{(x, \mu_A(x)) \mid x \in X\}$, se tiene que el elemento x pertenece al conjunto A con un grado de pertenencia $\mu_A(x)$, el cual varia entre 0 y 1. Luego un conjunto difuso A es una simple extensión de un conjunto clásico en el cual a la función característica le es permitido tener valores entre 0 y 1, si este valor lo restringimos a únicamente los valores de 0 y 1 entonces A es reducido a un conjunto clásico. A continuación se enumeran algunas de las definiciones básicas de lógica difusa.

Soporte y Fuzzy Singleton. El *soporte* de un conjunto difuso A es el conjunto de todos los puntos $x \in X$ tales que $\mu_A(x) > 0$: $Soporte(A) = \{x \mid \mu_A(x) > 0\}$. Un conjunto difuso cuyo soporte es un único punto en X con $\mu_A(x) = 1$ es llamado un *fuzzy singleton.*

Centro y Conjunto Difuso Normal. El *centro* de un conjunto difuso A se define como el conjunto de todos los puntos $x \in X$ tales que $\mu_A(x) = 1 : Centro(A) = \{x \mid \mu_A(x) = 1\}$. Un conjunto difuso A es *normal* si su centro es no vacío, en otras palabras siempre podemos encontrar un punto $x \in X$ tal que $\mu_A(x) = 1$.

Punto de Cruce. Un punto de cruce de un conjunto difuso A es un punto $x \in X$ en el cual $\mu_A(x) = 0{,}5$: $cruce(A) = \{x \mid \mu_A(x) = 0{,}5\}$

Corte-α y Corte Fuerte-α. El *corte-α* o *conjunto de nivel-α* de un conjunto difuso A es un conjunto no difuso (clásico) definido por: $A_\alpha = \{x \mid \mu_A(x) \geq \alpha\}$. mientras que el *corte fuerte-α* o *conjunto fuerte de nivel-α* se define como: $A'_\alpha = \{x \mid \mu_A(x) > \alpha\}$.

Conjunto Difuso Convexo. Un conjunto difuso A es convexo si sólo si para cualesquiera x_1, x_2 y para cualquier $\lambda \in [0, 1]$, $\mu_A(\lambda x_1 + (1 - \lambda)x_2) \geq \text{mín}\{\mu_A(x_1), \mu_A(x_2)\}$. Alternativamente, A es convexo si todos sus conjuntos de nivel-α son convexos.

Un conjunto clásico $C \in R^n$ es convexo si y sólo si para cualesquiera dos puntos $x_1, x_2 \in C$, su combinación convexa $\lambda x_1 + (1 - \lambda)x_2$ esta acotada en C, donde $0 \leq \lambda \leq 1$.

Números Difusos. Un número difuso A es un conjunto difuso en R normal y convexo.

Ancho de Banda de Conjuntos Difusos Normales y Convexos. Para un conjunto difuso normal y convexo el ancho de banda o ancho se define como la distancia entre los dos únicos puntos de cruce. $ancho\ de\ banda(A) = \mid x_2 - x_1 \mid$, donde $\mu_A(x_1) = \mu_A(x_2) = 0{,}5$

Simetría. Un conjunto difuso A es simétrico si su grado de pertenencia es simétrico alrededor de un cierto punto $x = c$, a saber, $\mu_A(c + x) = \mu_A(c - x)$ para toda $x \in X$.

Abierto por la izquierda, Abierto por la derecha, Cerrado. Un conjunto difuso A es abierto por la izquierda si $\lim_{x \to -\infty} \mu_A(x) = 1$ y $\lim_{x \to +\infty} \mu_A(x) = 0$; es abierto por la derecha si $\lim_{x \to -\infty} \mu_A(x) = 0$ y $\lim_{x \to +\infty} \mu_A(x) = 1$ y cerrado si $\lim_{x \to -\infty} \mu_A(x) = \lim_{x \to +\infty} \mu_A(x) = 0$.

Operaciones Básicas.

Subconjunto. Un conjunto difuso A esta contenido en un conjunto difuso B (o A es un subconjunto de B) si y sólo si $\mu_A(x) \leq \mu_B(x)$ para toda x.

Unión. La unión de dos conjuntos difusos A y B se define como:

$$\mu_{A\cup B} = \text{máx}\{(\mu_A(x), \mu_B(x))\} = \mu_A(x) \vee \mu_B(x)$$

Intersección. La intersección de dos conjuntos difusos A y B se define de la siguiente manera:

$$\mu_{A\cap B} = \text{mín}\{(\mu_A(x), \mu_B(x))\} = \mu_A(x) \wedge \mu_B(x)$$

Complemento. El complemento de un conjunto difuso A, se denota por $\bar{A}(\neg A$, NOT $A)$ y se define como: $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$.

Producto Cartesiano y Co-producto. Seas A y B los conjuntos difusos en X y Y, respectivamente. El producto cartesiano de A y B, denotado por $A \times B$, es un conjunto difuso con grado de pertenencia:

$$\mu_{A\times B}(x, y) = \text{mín}(\mu_A(x), \mu_B(y)).$$

De manera similar el Co-producto $A+B$ es un conjunto difuso con grado de pertenencia:

$$\mu_{A+B}(x, y) = \text{máx}(\mu_A(x), \mu_B(y)).$$

T-norma y T-conorma La intersección de dos conjuntos difusos A y B es especificado por una función T :[0,1]×[0,1]→[0,1], la cual agrega dos funciones de membresía de la siguiente forma:

$$\mu_{A\cap B}(x) = T(\mu_A(x), \mu_B(x)) = \mu_A(x) \,\tilde{*}\, \mu_B(x)$$

donde $\widetilde{*}$ es un operador binario de la función T. A esta clase de operadores usualmente se les conoce como operadores T-norma (norma triangular) y reunen las siguientes propiedades.

Definición 2.17 *T-Norma*

Un operador T-Norma satisface las siguientes propiedades:

$$\begin{aligned} T(0,0) &= 0, T(a,1) = T(1,a) = a && \textit{(Acotado)} \\ T(a,b) &\leq T(c,d) \textit{ si } a \leq c \textit{ y } b \leq d && \textit{(Monotonicidad)} \\ T(a,b) &= T(b,a) && \textit{(Conmutatividad)} \\ T(a,T(b,c)) &= T(T(a,b),c) && \textit{(Asociatividad)} \end{aligned} \tag{2.12}$$

Definición 2.18 *T-conorma*

Un operador T-Conorma (o S-Norma) satisface las siguientes propiedades:

$$\begin{aligned} S(1,1) &= 1, S(0,a) = S(a,0) = a && \textit{(Acotado)} \\ S(a,b) &\leq S(c,d) \textit{ si } a \leq c \textit{ y } b \leq d && \textit{(Monotonicidad)} \\ S(a,b) &= S(b,a) && \textit{(Conmutatividad)} \\ S(a,S(b,c)) &= S(S(a,b),c) && \textit{(Asociatividad)} \end{aligned} \tag{2.13}$$

Parametrización y Formulación de la Función de Membresía.

Un conjunto difuso es completamente caracterizado por su función de membresía (FM) o grado de pertenencia [49], ya que la mayoría de los conjuntos difusos su universo X es la línea real R, seria impractico listar todos los pares definiendo el grado de pertenencia [16]. Una forma de definir la función de membresía más concreta y conveniente es expresarla como una formula matemática. A continuación se describen las clases de funciones de parametrización comúnmente usadas para describir la función de membresía de una y dos dimensiones.

Funciones de Membresía.

FM Triangular. Esta es especificada por tres parámetros {a,b,c} de la forma siguiente:

$$Triangulo(x;a,b,c) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ \dfrac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases} \tag{2.14}$$

Los parámetros $\{a,b,c\}$ (con $a < b < c$) determinan las coordenadas x de las tres esquinas de la FM triangular.

FM Trapezoidal. Esta es especificada por cuatro parámetros $\{a,b,c,d\}$ de la forma siguiente:

$$Trapezoide(x;a,b,c,d) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \dfrac{d-x}{d-c}, & c \leq x \leq d \\ 0, & d \leq x \end{cases} \tag{2.15}$$

Los parámetros $\{a,b,c,d\}$ (con $a < b \leq c < d$) determinan las coordenadas x de las cuatro esquinas de la FM trapezoidal.

FM Gausiana. Esta es especificada por dos parámetros $\{c,\sigma\}$ de la siguiente forma:

$$Gausiano(x;c,\sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \tag{2.16}$$

Una función de membresía gausiana es determinada completamente por c y σ; c representa el centro de la FM, mientras que σ representa el ancho de banda de la FM.

FM de Campana Generalizada. Es especificada por tres parámetros $\{a, b, c\}$:

$$Campana(x; a, b, c) = \frac{1}{1+ \mid \frac{x-c}{a} \mid^{2b}} \tag{2.17}$$

donde el parámetro b es usualmente positivo (si b es negativo la forma de su FM es una campana al revés).

FM Sigmoidal. Se define por:

$$Sig(x; a, c) = \frac{1}{1 + \exp[-a(x-c)]}, \tag{2.18}$$

donde a controla la cuesta al punto de cruce $x = c$. Dependiendo del signo del parámetro a, una sigmoidal FM es abierta por la derecha o abierta por la izquierda.

FM Izquierda-Derecha o FM L-R. Es especificada por tres parámetros $\{\alpha, \beta, c\}$:

$$LR(x; c, \alpha, \beta) = \begin{cases} F_L\left(\frac{c-x}{\alpha}\right) & x \leq c \\ F_R\left(\frac{x-c}{\beta}\right), & x \geq c \end{cases} \tag{2.19}$$

donde $F_L(x)$ y $F_R(x)$ son funciones monótonicamente decrecientes definidas sobre [0,∞), con $F_L(0) = F_R(0) = 1$ y $\lim_{x\to\infty} F_L(x) = \lim_{x\to\infty} F_R(x) = 0$.

Operaciones Aritméticas con Números Difusos.

Un número triangular difuso puede ser parametrizado por una tripleta (a, b, c) donde la función de pertenencia de un número triangular difuso A esta definida por:

$$\mu_A(u) = \begin{cases} 0, & u < a \\ \frac{u-a}{b-a}, & a \leq u \leq b \\ \frac{c-u}{c-b}, & b \leq u \leq c \\ 0, & u > c \end{cases} \tag{2.20}$$

Las operaciones aritméticas entre números difusos triangulares X y Y son definidas como sigue [20][6] :

1. Adición de números difusos triangulares $\oplus$.

$$X \oplus Y = (a_x, b_x, c_x) \oplus (a_y, b_y, c_y) = (a_x + a_y, b_x + b_y, c_x + c_y) \tag{2.21}$$

2. Sustracción de números difusos triangulares $\ominus$.

$$X \ominus Y = (a_x, b_x, c_x) \ominus (a_y, b_y, c_y) = (a_x - c_y, b_x - b_y, c_x - a_y) \tag{2.22}$$

3. Multiplicación de números difusos triangulares $\otimes$.

$$X \otimes Y = (a_x, b_x, c_x) \otimes (a_y, b_y, c_y) = (a_x \times a_y, b_x \times b_y, c_x \times c_y) \tag{2.23}$$

4. División de números difusos triangulares $\otimes$.

$$X \oslash Y = (a_x, b_x, c_x) \oslash (a_y, b_y, c_y) = (a_x/c_y, b_x/b_y, c_x/a_y) \tag{2.24}$$

2.4. Razonamiento Difuso

2.4.1. Principio de Extensión.

El principio de extensión introducido por Zadeh [43] es un concepto básico de la teoría de conjuntos difusos que provee un procedimiento general para extender dominios clásicos de expresiones matemáticas a dominios difusos.

Sea f una función de X a Y y A es un conjunto difuso sobre X definido como:

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + ... + \mu_A(x_n)/x_n. \tag{2.25}$$

Luego, el principio de extensión establece que la imagen de un conjunto difuso A bajo el mapeo $f(.)$ puede ser expresado como un conjunto difuso B,

$$B = f(A) = \mu_A(x_1)/y_1 + \mu_A(x_2)/y_2 + ... + \mu_A(x_n)/y_n \tag{2.26}$$

donde $y_i = f(x_i)$, $i = 1, ..., n$. Si $f(.)$ es un mapeo de varios a uno, entonces existe $x_1, x_2 \in X$, $x_1 \neq x_2$, tal que $f(x_1) = f(x_2) = y^*$, $y^* \in Y$. En este caso, el grado de pertenencia de B con $y = y^*$ es el máximo del grado de pertenencia de A con $x = x_1$ y $x = x_2$, de donde tenemos que:

$$\mu_B(y) = \max_{\substack{x_i \in X \\ f(x_i)=y}} \mu_A(x). \tag{2.27}$$

Suponiendo que la función f es un mapeo de un espacio n-dimensional $X_1 \times X_2 \times ... \times X_n$ a un universo uno-dimensional Y tal que $y = f(x_1, ..., x_n)$, y suponiendo que $A_1, ..., A_n$ son conjuntos difusos en $X_1, ..., X_n$, respectivamente. entonces el principio de extensión afirma que el conjunto difuso B inducido por el mapeo f es definido por:

$$B = \{(y, \mu_B(y)) \mid y = f(x_1, x_2, ..., x_n), (x_1, ..., x_n) \in X\}, \tag{2.28}$$

donde

$$\mu_B(y) = \max_{\substack{(x_1, x_2, ..., x_n) \in X \\ y = f(x_1, x_2, ..., x_n)}} [\min_i \mu_{A_i}(x_i)], \ i = 1, 2, \ldots, n \tag{2.29}$$

Relaciones Difusas. Las relaciones difusas binarias son conjuntos difusos en $X \times Y$ el cual mapea cada elemento en $X \times Y$ a un grado de pertenencia entre 0 y 1.

Sean X y Y dos universos, entonces:

$$R = \{((x, y), \mu_R(x, y)) \mid (x, y) \in X \times Y\} \tag{2.30}$$

es una relación difusa binaria en $X \times Y$.

Composición Max-Min Sean R_1 y R_2 dos relaciones difusas definidas sobre $X \times Y$ y $Y \times Z$, respectivamente. La composición max-min de R_1 y R_2 es un conjunto difuso definido por:

$$R_1 \circ R_2 = \{[(x,z), \max_y \min(\mu_{R_1}(x,y), \mu_{R_2}(y,z))] \mid x \in X,\ y \in Y,\ z \in Z\}, \tag{2.31}$$

o equivalentemente,

$$\mu_{R_1} \circ \mu_{R_2}(x,z) = \max_y \min[(\mu_{R_1}(x,y), \mu_{R_2}(y,z)] = \vee_y[(\mu_{R_1}(x,y) \wedge \mu_{R_2}(y,z)] \tag{2.32}$$

Composición Max-Producto La composición max-producto se define como sigue:

$$\mu_{R_1 \circ R_2}(x,z) = \max_y[(\mu_{R_1}(x,y)\mu_{R_2}(y,z)] \tag{2.33}$$

2.4.2. Variables Lingüísticas

Cuando una variable toma un número como su valor tenemos un marco matemático de trabajo bien establecido para formularlo, pero cuando una variable toma una palabra como su valor no tenemos un marco de trabajo formal para formularlo en teoría de matemática clásica. Para proveer dicho marco de trabajo se introdujo el concepto de variable lingüística, hablando normalmente, si una variable puede tomar palabras en lenguaje natural como su valor se le llama variable lingüística, donde las palabras son caracterizadas por conjuntos difusos definidos en el universo en el cual la variable es definida.

Una *variable lingüística* es caracterizada por una tupla $(x, T(x), X, M)$ en la cual x es el nombre de la variable; $T(x)$ es el *conjunto de términos* de x, esto es, el conjunto de sus *valores lingüísticos* o *términos lingüísticos.*; X es el universo en cuestión y M es una *regla semántica* la cual asocia a cada valor lingüístico su significado $M(A)$, donde $M(A)$ denota un conjunto difuso en X.

Ejemplo 2.1 *Variables lingüísticas y valores lingüísticos.*

Si la "temperatura" es interpretada como una variable lingüística, entonces el termino T(temperatura) será:

$$\begin{aligned} T(x) \;=\; & \{\text{calurosos, no caluroso, muy caluroso, no muy caluroso,...,} \\ & \text{templado, no templado,...,} \\ & \text{frío, no frío, muy frío, más o menos frío, no muy frío,...}\} \end{aligned}$$

donde cada termino en T(temperatura) es caracterizado por un conjunto difuso del universo de discurso X=[0,45]. Normalmente usamos "la temperatura es calurosa" para denotar la asignación del valor lingüístico "caluroso" a la variable lingüística *temperatura*. Por otro lado, cuando la temperatura es interpretada como una variable numérica, usamos la expresión "temperatura = 40" asignando el valor numérico "40" a la variable numérica *temperatura*. La regla semántica define la función de membresía de cada valor lingüístico en el conjunto de términos.

Del ejemplo anterior, podemos observar que el conjunto de términos consiste de varios *términos primarios* (caluroso, templado, frío) modificados por la *negación* ("no") y/o los *modificadores lingüísticos* (muy, más o menos, extremadamente, apenas,...).

Aunque la palabra *modificador* no tiene un significado definido en esencia actúa como un intensificador. Así damos la siguiente definición para los dos *modificadores* más comunes, muy y más o menos.

Definición 2.19 *Sea A un conjunto difuso en U, entonces muy_A es definido como un conjunto difuso en U con la función de membresía*

$$\mu_{muy\_A}(x) = [\mu_A(x)]^2 \tag{2.34}$$

y $más\_o\_menos\_A$ es un conjunto difuso en U con la función de membresía

$$\mu_{más\_o\_menos\_A}(x) = [\mu_A(x)]^{\frac{1}{2}}$$

Concentración y Dilatación de Valores Lingüísticos. Sea A un valor lingüístico caracterizado por un conjunto difuso con grado de pertenencia $\mu_A(.)$. Entonces A^k es interpretado como una versión modificada del valor lingüístico original expresado como:

$$A^k = \int_x [\mu_A(x)]^k / x \tag{2.35}$$

En particular la operación de concentración es definida como:

$$CON(A) = A^2 \tag{2.36}$$

Mientras que la operación de dilatación es expresada por:

$$DIL(A) = A^{0,5} \tag{2.37}$$

Intensificación de Contrastes. La operación de intensificación de contrastes sobre un valor lingüístico A es definido por:

$$INT(A) = \begin{cases} 2A^2, & para\ 0 \leq \mu_A(x) \leq 0{,}5; \\ \neg 2(\neg A)^2, & para\ 0{,}5 \leq \mu_A(x) \leq 1 \end{cases} \tag{2.38}$$

La intensificación de contrastes INT incrementa los valores de $\mu_A(x)$ cuando están arriba de 0.5 y disminuye aquellos que se encuentran abajo de este punto. Luego, la intensificación de contrastes tiene la propiedad de reducir la difusividad de un valor lingüístico A.

Ortogonalidad. Un conjunto de términos $T = t_1, ..., t_n$ de una variable lingüística x sobre el universo X es ortogonal si cumple la siguiente propiedad:

$$\sum_{i=1}^{n} \mu_{t_i}(x) = 1,\ \forall x \in X \tag{2.39}$$

donde los t_i's son conjunto normales y convexos definidos sobre X y estos conjunto difusos hacen el conjunto de términos T.

2.4.3. Reglas Difusas Si-Entonces.

Una regla difusa Si-Entonces (también conocida como regla difusa, implicación difusa o estatuto condicional difuso) asume la forma:

$$Si\ x \text{ es } A \quad Entonces \quad y \text{ es } B \tag{2.40}$$

Donde A y B son valores lingüísticos definidos por conjuntos difusos sobre los universos X y Y respectivamente. A menudo "x es A" es llamado *antecedente* o *premisa*, mientras que "y es B" es llamado *consecuencia* o *conclusión*. Ejemplos de reglas difusas Si-Entonces son empleados en nuestras expresiones lingüísticas diariamente, tales como las siguientes:

- *Si* la presión es alta *Entonces* el volumen es pequeño.
- *Si* la carretera esta resbaladiza *Entonces* manejar es peligroso.
- *Si* el jitomate esta rojo *Entonces* esta maduro.
- *Si* la velocidad es alta *Entonces* frenar poco a poco.

Regla Composicional de Inferencia

La regla composicional de inferencia es una generalización del siguiente procedimiento. Suponiendo que tenemos una curva $y = f(x)$ que regula la relación entre x y y. Cuando $x = a$, entonces a partir de $y = f(x)$ podemos inferir que $y = b = f(a)$; ver figura 2.6. Una generalización del proceso anterior mencionado permitirá a a ser un intervalo y a $f(x)$ ser una función intervalo-valor, como se muestra en la figura 2.7. Para encontrar el intervalo resultante $y = b$ correspondiente al intervalo $x = a$, construimos primero una extensión cilíndrica de a y encontramos su intersección I con la curva del intervalo-valor. La proyección de I sobre el eje y nos proporciona el intervalo $y = b$.

Yendo un paso adelante en nuestra cadena de generalización asumiendo que A' es un conjunto difuso en X y Q es una relación difusa en $X \times Y$. Formando de nuevo una extensión cilíndrica A'_E de A' e interceptando con la relación difusa Q (ver figura 2.8) obtenemos un

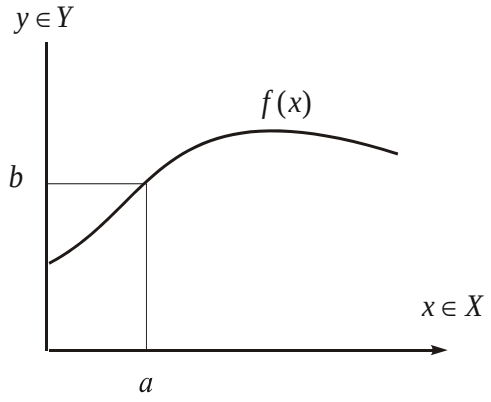


Figura 2.6: Infiriendo $y = b$ de $x = a$ y $y = f(x)$

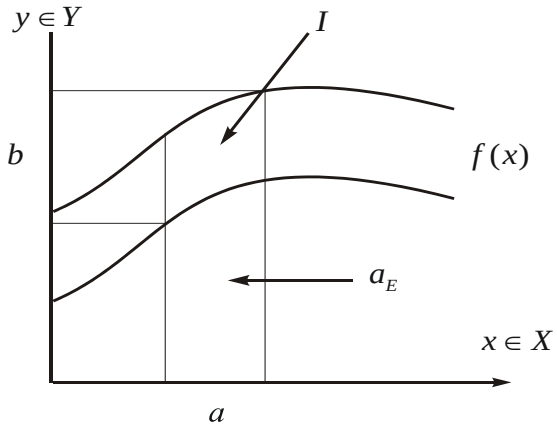


Figura 2.7: Infiriendo el intervalo b del intervalo a y la función valor-intervalo $f(x)$

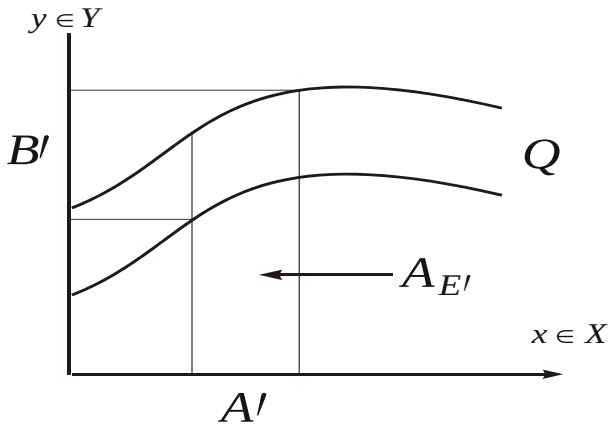


Figura 2.8: Infiriendo el conjunto difuso B' del conjunto difuso A' y la relación difusa Q

conjunto difuso $A'_E \cap Q$ el cual es análogo a la intersección I en la figura 2.7. Entonces, proyectando $A'_E \cap Q$ en el eje-y obtenemos el conjunto difuso B'.

Específicamente, dados $\mu_{A'}$ y μ_Q (x, y) tenemos:

$$\mu_{A'_E}(x, y) = \mu_{A'}(x)$$

y consecuentemente

$$\mu_{A'_E \cap Q}(x, y) = \begin{array}{l} \text{mín}\left[\mu_{A'_E}(x, y), \mu_Q(x, y)\right] \\ \text{mín}\left[\mu_{A'}(x, y), \mu_Q(x, y)\right] \end{array}$$

Finalmente obtenemos B', la proyección de $A'_E \cap Q$ en Y como

$$\mu_{B'}(y) = \begin{array}{l} \text{máx}_{x \in X} \text{mín}\left[\mu_{A'}(x), \mu_Q(x, y)\right] \\ \vee_{x \in X}\left[\mu_{A'}(x) \wedge \mu_Q(x, y)\right] \end{array}$$

Usando la regla composicional de inferencia, podemos formalizar un procedimiento de inferencia sobre un conjunto de reglas difusas Si-Entonces. Este procedimiento de inferencia generalmente es llamado razonamiento aproximado o razonamiento difuso.

Razonamiento Difuso

Como en cualquier otra lógica, las reglas de inferencia de lógica difusa gobiernan la deducción de una proposición q a partir de un conjunto de premisas $\{p_1, p_2, ..., p_n\}$. En lógica difusa, se permite que ambas; premisas y conclusiones sean proposiciones difusas. Además, ya que los resultados inferidos, usualmente deben ser traducidos en términos más significativos (conjuntos difusos) por el uso de aproximación lingüística, la conclusión final obtenida a partir de las premisas $p_1, p_2, ..., p_n$ es, en general, una aproximación en lugar de una consecuencia exacta de $p_1, p_2, ..., p_n$.

La regla básica de inferencia en la lógica tradicional es el *modus ponens*, de acuerdo al cual podemos inferir la verdad de una proposición B a partir de la veracidad de A y la implicación $A \rightarrow B$. Por ejemplo si A es identificado como "El jitomate esta rojo" y B con "El jitomate esta maduro", entonces si es verdad que "El jitomate esta rojo" esto implica que también es verdad que "El jitomate esta maduro". Este concepto es ilustrado de la manera siguiente:

Premisa 1	(Regla)	*Si* x es A *Entonces* y es B
Premisa 2	(Hecho)	x es A
Conclusión:		y es B

Sin embargo, en gran parte del razonamiento humano, el modus ponens es empleado de una manera aproximada. Por ejemplo si tenemos la misma regla de implicación y conocemos que "El jitomate es más o menos rojo" podemos inferir que "El jitomate esta más o menos maduro". Esto es escrito como:

Premisa 1	(Regla)	*Si* x es A *Entonces* y es B
Premisa 2	(Hecho)	x es A'
Conclusión:		y es B'

Donde A' es cercano a A y B' es cercano a B. Cuando A, B, A' y B' son conjuntos difusos de universos apropiados, el procedimiento de inferencia anterior es llamado *razonamiento*

aproximado o *razonamiento difuso*, también llamado, *modus ponens generalizado*, ya que, este es un caso especial del modus ponens.

Sean A, A' y B conjunto difusos de X, X y Y, respectivamente. Asumiendo que la implicación difusa $A \rightarrow B$ es expresada como una relación difusa R sobre $X \times Y$. Entonces el conjunto difuso B inducido por "x es A" y la regla difusa "*Si* x es A *Entonces* y es B" es definido por:

$$\mu_{B'}(y) = \begin{array}{l} \max_x \min[\mu_{A'}(x), \mu_R(x,y)] \\ \vee_x [\mu_{A'}(x) \wedge \mu_R(x,y)], \end{array} \tag{2.41}$$

o, equivalentemente,

$$B' = A' \circ R = A' \circ (A \rightarrow B) \tag{2.42}$$

Podemos usar el procedimiento de inferencia de razonamiento difuso para derivar conclusiones, sabiendo que la implicación difusa $A \rightarrow B$ es definida como una relación difusa binaria.

2.5. Conclusiones

En este capitulo definimos los conceptos básicos de redes de Petri, redes neuronales, conjuntos difusos y razonamiento difuso.

Una red de Petri se representa gráficamente por un grafo dirigido bipartito. Los dos tipos de nodos, lugares y transiciones representan, las variables que definen el estado del sistema (lugares) y a sus transformadores (transiciones). Los lugares se representan por círculos, las transiciones por barras, estos son conectados por arcos que van de lugares a transiciones y de transiciones a lugares. La presencia o ausencia de una señal dentro de un lugar puede indicar cuando una condición asociada con este lugar es verdadera o falsa. Por otro lado, las redes neuronales tienen características como: habilidad de aprendizaje, habilidad de procesamiento en paralelo y la propiedad de aproximar cualquier función suave. Las reglas de producción difusas y el razonamiento difuso son la espina dorsal de los sistemas de inferencia difusos,

los cuales son la herramienta de modelado más importante basados en la teoría de conjuntos difusos.

Capítulo 3

Representación de Conocimiento y Razonamiento con Redes de Petri Difusas

Las reglas Si-Entonces son el formalismo más popular para representar conocimiento. Estas también son llamadas *reglas difusas* (reglas que describen la relación difusa entre dos proposiciones). Un conjunto entero de reglas de producción es llamado *una base de reglas*. Además al formalismo de reglas de producción el sistema provee un significado para definir los objetos referidos en las reglas de producción, llamado *declaración de dominio*. La base de reglas y la declaración del dominio constituyen juntos una *base de conocimiento* (KB) del sistema difuso.

En [22] las RdP son usadas para modelar bases de reglas, relacionando simplemente algunos de los elementos (función de marcado) y características (básicamente lugares y transiciones) del formalismo de Petri con los elementos básicos de una KB (proposiciones, grados de verdad e implicaciones). Cuando la complejidad de los sistemas KB incrementa, nuevos elementos son incluidos en la definición inicial de RdP, los cuales permiten a la RdP representar adecuadamente los rasgos que caracterizan una KB. Este procedimiento nos lleva a la representación de un nuevo modelo el cual es llamado red de Petri difusa (RPD). Las RdP

tienen una cualidad inherente en representar lógica de una forma visual e intuitiva y las RPD toman todas estas ventajas de las RdP. De esta manera, la forma de razonamiento de un sistema experto puede ser reducido a simples árboles de nodos si algoritmos de razonamiento basados en RPD son empleados como máquinas de inferencia. Varias generalizaciones de las redes de Petri difusas han sido propuestas en la literatura sobre redes abstractas con el propósito de considerar ciertos problemas concretos vinculados con la representación del conocimiento, razonamiento difuso y teoría de decisión.

Las redes de Petri difusas (RPD) fueron introducidas a la literatura como un modelo para representar sistemas basados en conocimiento. En [7] Chen, Ke y Chang proponen un algoritmo de razonamiento difuso, en el cual, el valor de verdad de un lugar objetivo puede ser evaluado a partir de un lugar inicial, siempre y cuando el lugar objetivo sea alcanzable a partir de el lugar inicial, Yeung y Tsang proponen en [47] un algoritmo de razonamiento difuso basado en grados de similaridad. Luego, las RPD son usadas para representar conocimiento difuso y razonamiento. Ya que, basados en el disparo de las transiciones de las RPD se pueden desarrollar algoritmos para inferir conocimiento.

En este capitulo, representamos conocimiento utilizando redes de Petri difusas, además utilizando números difusos triangulares inferimos conocimiento mediante tres algoritmos de razonamiento difuso desarrollados en este capitulo.

3.1. Representación de Conocimiento.

A principios de los setentas, Alan Newell y Herb Simon introdujeron la noción de *sistemas de producción* en inteligencia artificial [27] como un modelo psicológico de comportamiento humano. En este modelo, parte del conocimiento es representado en unidades separadas llamadas *producciones* o *reglas de producción.* Estas unidades contienen información concerniente a las acciones que una persona toma al percibir ciertos estímulos del medio ambiente. El modelo de Newell y Simon es parecido a la teoría de memoria de dos procesos en psicología cognoscitiva [24], donde dos diferentes mecanismos de almacenamiento de información sensorial son distinguidos: la memoria a corto plazo y la memoria a largo plazo,

respectivamente. La memoria a corto plazo únicamente contiene una cantidad limitada de información rápidamente decadente. Esta corresponde a la parte de un sistema de producción en el cual los datos de entrada y datos derivados son guardados. La memoria a largo plazo es para almacenamiento de información permanente y corresponde a las reglas base de un sistema de producción en el cual las reglas de producción son especificadas. Desde los años setentas el formalismo de reglas de producción ha sido empleado por algunos investigadores para desarrollar sistemas basados en conocimiento. En estos sistemas, el formalismo de reglas de producción es visto meramente como un lenguaje formal para expresar conocimiento.

3.1.1. Reglas de Producción

Una de las propuestas más populares para representación de conocimiento es el uso de reglas de producción, normalmente llamadas reglas SI-ENTONCES. Estas pueden tomar varias formas, por ejemplo:

Si condición Entonces acción

Si premisa Entonces conclusión

Si p_1 y p_2 son verdaderas Entonces p_3 es verdad

donde p_1, p_2 y p_3 son proposiciones.

Algunos de los beneficios de las reglas Si-Entonces es que son modulares, cada regla define una relativamente pequeña y en principio independiente, pieza de conocimiento.

Sean $p =$"x esta en A" y $q =$ "y esta en B" dos proposiciones, donde A y B son conjuntos clásicos. "p implica a q" o "Si p Entonces q" significa que no puede ocurrir que p sea verdad y que q no lo sea. La interpretación total de la regla de producción "Si p Entonces q" es que, si la regla de producción es verdad, entonces el grado de verdad de q es por lo menos el grado de verdad de p, esto es:

$$\text{Si } p \text{ Entonces } q \text{ es verdad } \Leftrightarrow \tau(p) \leq \tau(q)$$

Cuadro 3.1: Tabla de verdad

p	q	*Si p Entonces q*
1	1	1
0	1	1
0	0	1
1	0	0

Entonces tenemos que:

$$\text{Si} \quad p \quad \text{Entonces} \quad q = \begin{cases} 1 \text{ si } \tau(p) \leq \tau(q) \\ 0 \text{ de otra manera} \end{cases}$$

de donde tenemos la Tabla 3.1. Claramente, el valor de verdad de las proposiciones p, q de estas reglas de producción únicamente puede tomar valores 0 o 1.

3.1.2. Reglas de Producción Difusas

En muchas situaciones, puede ser difícil capturar datos en forma precisa. Una forma para representar propiamente el conocimiento del mundo real bajo la teoría difusa, son las reglas de producción difusas [16][28]. Las reglas de producción difusas (RPDs) son usadas para representar conocimiento difuso, impreciso, conceptos vagos y ambiguos. Las RPDs son usualmente presentadas en la forma de una regla difusa SI-ENTONCES. Una regla de producción difusa es una regla que describe la relación difusa entre dos proposiciones. Sea R_i la i-ésima regla de producción del conjunto $R = \{R_1, R_2, ..., R_n\}$, esta es definida de la siguiente manera:

$$R_i : SI \;\; d_j \;\; Entonces \; d_k \;\; (CF = \mu_i), Th, w \tag{3.1}$$

Donde:

Cuadro 3.2: Escalas de Verdad y sus Intervalos Numéricos

Escala de Verdad	*Intervalo Numérico*
Siempre Verdadero	[1.0 1.0]
Extremadamente Verdadero	[0.95 0.99]
Muy Verdadero	[0.80 0.94]
Considerablemente Verdadero	[0.65 0.79]
Moderadamente Verdadero	[0.45 0.64]
Mas o Menos Verdadero	[0.30 0.44]
Minoritariamente Verdadero	[0.10 0.29]
Minimamente Verdadero	[0.01 0.09]
No Verdadero	[0.00 0.00]

1. d_j y d_k son proposiciones las cuales son usualmente representadas de la siguiente forma:

   El (Atributo) de (Objeto) es (Valor)

   por ejemplo El *precio* del *auto* es *alto*. Se sugiere reemplazar *El (Atributo) de (Objeto)* por una variable. Esto es, el ejemplo mencionado anteriormente puede ser expresado como x es A, donde x es la variable que contiene "*El (Atributo) de (Objeto)*", y A es el valor de la variable x. Esto es, la ecuación 3.1 quedaría de la forma:

   $$R_i : SI \quad \text{“}x \text{ es A”} \quad Entonces \text{ “}y \text{ es B”} \quad (CF = \mu_i), Th, w$$

   donde A y B son variables lingüísticas definidos sobre los universos X y Y respectivamente. Las proposiciones pueden contener variables difusas. La verdad de cada proposición es un número difuso definido en el universo [0,1]. Un ejemplo de escalas de verdad y sus correspondientes intervalos numéricos está dado en la Tabla 3.2.

2. μ_i es el valor del factor de certeza (CF), donde μ_i esta definido en el universo $[0, 1]$ Este representa el grado de certidumbre de la regla R_i.

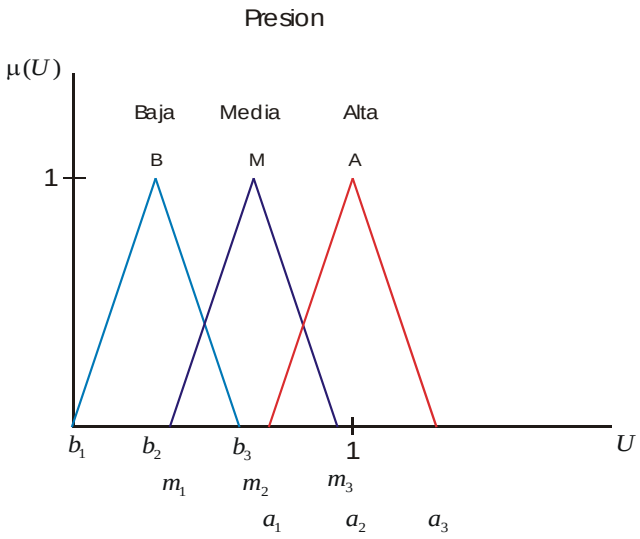


Figura 3.1: Funciones de membresía de presión baja, media y alta

3. Th es el valor frontera

4. w es el peso o grado de importancia de una regla de producción. La anexión de pesos a las proposiciones es de gran importancia, ya que si existen varias proposiciones antecedentes que conlleven a una consecuente, estas proposiciones la mayoría de las veces difieren en importancia, afectando el valor de verdad de la proposición consecuente. El grado de importancia que se adhiera a cada proposición lo vamos a conocer como *peso de la proposición.*

Las función de membresía del conjunto difuso A (*presión alta*), es ilustrada en la figura 3.1 en la cual la presión alta es representada por un número difuso triangular (a_1, a_2, a_3) de la misma forma la presión baja y media son representadas por números difusos triangulares (b_1, b_2, b_3) y (m_1, m_2, m_3) respectivamente. En general las reglas de producción difusas pueden ser categorizadas en 4 tipos los cuales se definen como sigue:

Tipo 1.

$$R_i : SI \;\; d_1 \;\; Entonces \; d_2 \;\; (CF = \mu_i), Th_i = \lambda_i \tag{3.2}$$

Tipo 2.

$$R_i : SI\ d_1\ Y\ d_2\ Y, .., Y\ d_n\ Entonces\ d_z\ (CF = \mu_i), w_1, .., w_{jn}, Th_i = \lambda_i \tag{3.3}$$

Tipo 3.

$$R_i : SI\ d_a\ Entonces\ d_{k1}\ Y\ d_{k2}\ Y, .., Y\ d_{kj}\ (CF = \mu_{k1}, .., \mu_{kj}), Th = \lambda_1, .., \lambda_j \tag{3.4}$$

Tipo 4.

$$R_i : SI\ d_1 O\ d_2\ O, .., O\ d_j\ Entonces\ d_z\ (CF = \mu_1, .., \mu_j), Th = \lambda_1, .., \lambda_j. \tag{3.5}$$

3.2. Representación de Conocimiento con Redes de Petri Difusas.

Es posible usar un modelo de redes de Petri para representar reglas de producción [30]. Una red de Petri se representa gráficamente por un *grafo dirigido bipartito*. Los dos tipos de *nodos, lugares y transiciones* representan, las variables que definen el estado del sistema (lugares) y a sus transformadores (transiciones). Los lugares se representan por círculos, las transiciones por barras. Cada lugar en una red de Petri difusa puede contener o no una señal asociada con un valor de verdad el cual se encuentra definido en el universo [0,1]. Cada transición en una red de Petri es asociada con un valor denominado factor de certeza el cual también se encuentra definido en el universo [0,1]. En una red de Petri difusa, las relaciones de lugares a transiciones y de transiciones a lugares son representadas por arcos dirigidos. La estructura de una red de Petri difusa generalizada se define a continuación.

Definición 3.1 *(Red de Petri Difusa) Una red de Petri difusa (RPD) se define como un tuplo* $RPD = (P, T, D, I, O, f, \alpha, \beta)$ *donde:*

- P Es un conjunto finito y no vacío de lugares;
- T Es un conjunto finito y no vacío de transiciones;

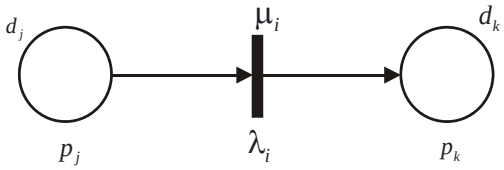


Figura 3.2: Red de Petri de una regla de producción difusa.

D Es un conjunto finito y no vacío de proposiciones;

I Es una función de entrada, $I : T \rightarrow P^{\infty}$;

O Es una función de salida, $O : T \rightarrow P^{\infty}$;

f Es una función que asigna a cada transición un número difuso en el universo [0,1].

α Es una función que asocia a cada lugar un número difuso en el universo [0,1] $\alpha : P \rightarrow$ [0,1];

β Es una función que asocia a un lugar p_i una proposición d_i.

Definición 3.2 *(Red de Petri difusa con Pesos) Una red de Petri Difusa con Pesos (RPDP) se define como un tuplo* $RPDP = (P, T, D, Th, I, O, W, F, f, \alpha, \beta, \gamma, \theta)$ *donde:*

$P, T, D, I, O, f, \alpha, \beta$. se definen de igual forma que en la definición anterior

$Th = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ conjunto finito de valores frontera

$W = \{w_1, w_2, \ldots, w_r\}$es un conjunto finito de pesos

$F = \{f_1, f_2, \ldots, f_s\}$ conjunto de conjuntos difusos

γ es una función que asigna a cada transición un valor frontera λ_i

θ es una función que asigna a cada lugar un peso, $\theta : P \rightarrow W$

Sean p_j y p_k lugares y t_i una transición. Si $p_j \in I(t_i)$, entonces existe un arco dirigido que va del lugar p_j a la transición t_i. Si $p_k \in O(t_i)$,entonces existe un arco dirigido que va de la transición t_i al lugar p_k. Si $f(t_i) = \mu_i$, donde μ_i es un número difuso definido en el universo $[0, 1]$, entonces la transición t_i se dice que esta asociada con un número difuso μ_i. Si $\beta(p_i) = d_i$ donde $d_i \in D$, entonces el lugar p_i se dice que esta asociado con una proposición d_i.

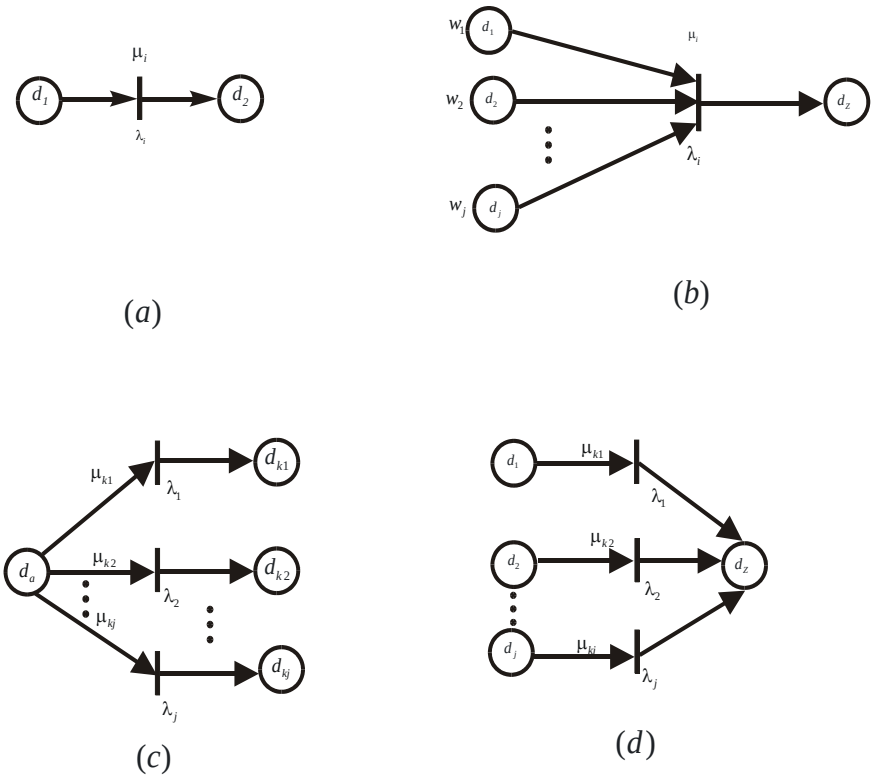


Figura 3.3: Redes de Petri de las principales reglas de producción difusas.

Si una RPD contiene algunas señales en algunos lugares, entonces es llamada red de Petri difusa marcada, donde una señal en el lugar p_i esta representada por un punto etiquetado $\overset{\alpha(p_i)}{\bullet}$ y el valor de la señal en el lugar $p_i, p_i \in P$ esta denotado por $\alpha(p_i)$, donde $\alpha(p_i)$ es un número difuso. Si $\alpha(p_i) = y_i$, donde y_i es un número difuso y $\beta(p_i) = d_i$. entonces el valor de verdad de la proposición d_i es y_i. Si $W(p_i) = w_i$ donde w_i es un número difuso definido en el universo [0,1] y $\beta(p_i) = d_i$, entonces el lugar p_i se dice que esta asociado con el peso w_i indicando el peso de la proposición d_i.

Si una RPDP contiene algunas señales en algunos lugares, entonces es llamada red de Petri difusa con pesos marcada.

Una transición t_i dispara removiendo las señales de los lugares de entrada y depositando una señal dentro de cada lugar de salida. El disparo de las reglas de producción difusas puede ser considerado como disparo de transiciones.

Podemos usar redes de Petri difusas para representar reglas de producción difusas de una base de reglas. Por ejemplo, la regla de producción difusa de la ecuación 3.1 puede ser modelada por una red de Petri difusa, como lo muestra la figura 3.2, donde μ_i es el valor de factor de certeza (CF) el cual indica el grado de certeza de la regla R_i, y μ_i esta definido en el universo [0,1].

Si la porción antecedente o consecuente de una regla de producción difusa contiene conectores "AND" o "OR"(Y o O), entonces esta es llamada una regla de producción difusa compuesta. Cualquier regla de producción difusa compuesta pueden ser incluida dentro de los cuatro tipos de reglas de producción difusas definidos en ecs. (3.2), (3.3), (3.4) y (3.5) y estas a su vez pueden ser modeladas por redes de Petri difusas como lo muestran las figuras [3.3(a)], [3.3(b)], [3.3(c)] y [3.3(d)] respectivamente.

Ejemplo

Suponiendo que deseamos verificar la calidad de los engranes de una fresa, los productos mal maquinados son rechazados o enviados a un proceso de remaquinado. Las propiedades en que se basan para determinar la calidad de tales engranes son únicamente *tolerancia* y *acabado* y mediante estas dos propiedades clasifica a los engranes en las cuatro clases

siguientes:

1. **Clase A**. Si diámetro interior y exterior están dentro de la tolerancia permitida y además el engrane tiene un buen acabado.

2. **Clase B**. Si el diámetro interior y exterior esta dentro de la tolerancia permitida y el engrane tiene un mal acabado.

3. **Clase C**. Si el diámetro interior esta dentro de la tolerancia, el diámetro exterior sobrepasa la tolerancia o si el diámetro exterior esta dentro de la tolerancia, el diámetro interior sobrepasa la tolerancia o si el diámetro interior y exterior sobrepasan la tolerancia y tiene un mal acabado o un buen acabado.

4. **Clase D**. Si el diámetro interior o exterior es menor que la tolerancia.

Suponiendo que la tolerancia permitida para el diámetro interior y diámetro exterior es de $\pm$ 0,03125,, entonces tenemos las apreciaciones siguientes:

1. $A = (DIR - DIP = \pm$ 0,03125)
2. $A\prime = (DER - DEP = \pm$ 0,03125)
3. $B = (DIR - DIP >$ 0,03125)
4. $B\prime = (DER - DEP >$ 0,03125)
5. $C = (DIR - DIP <-$ 0,03125)
6. $C\prime = (DER - DEP <-$ 0,03125)

Donde DIR es el diámetro interior real o diámetro interior de la pieza maquinada y DIP es el diámetro interior pretendido, de la misma forma DER es el diámetro exterior real ó diámetro exterior de la pieza maquinada y DEP es el diámetro exterior pretendido. Una vez tomadas en cuenta estas apreciaciones, mediante las siguientes reglas se modela el proceso de selección de calidad de engranes.

R_1:IF DI=A and DE=A$'$ THEN TOL=Acep-A

R_2:IF DI=A and DE=B$'$ THEN TOL=Acep-B

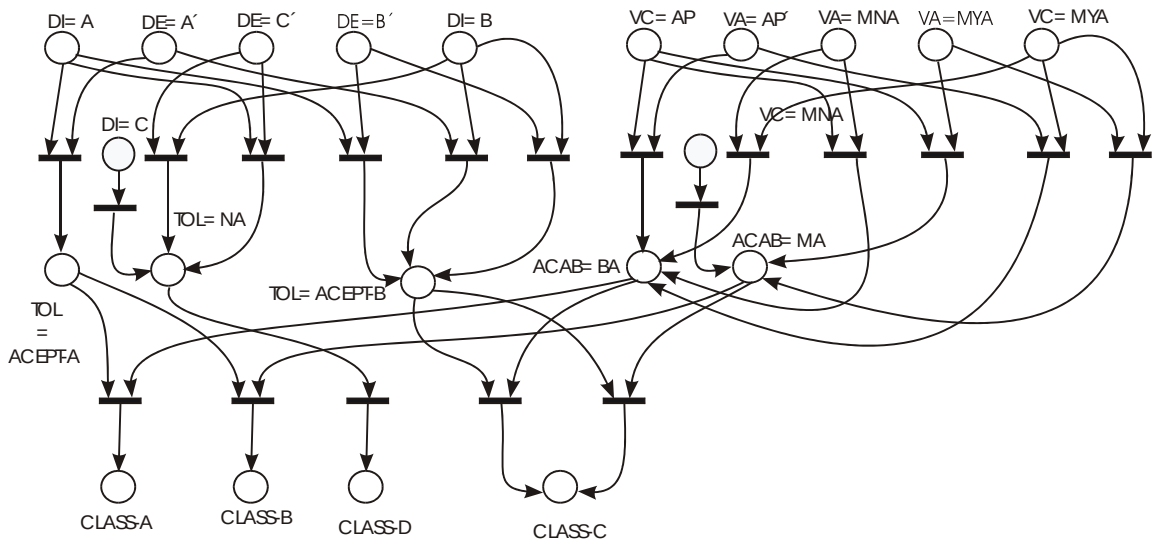


Figura 3.4: Red de Petri Difusa del proceso de inspección.

R_3:IF DI=A and DE=C′ THEN TOL=NA

R_4:IF DI=B and DE=A′ THEN TOL=Acep-B

R_5:IF DI=B and DE=B′ THEN TOL=Acep-B

R_6:IF DI=B and DE=C′ THEN TOL=NA

R_7:IF DI=C THEN TOL=NA

R_8:IF DI=C THEN TOL=NA

R_9:IF DI=C THEN TOL=NA

R_{10}:IF VC=AP and VA=AP THEN Acab=BA

R_{11}:IF VC=AP and VA=MNA THEN Acab=BA

R_{12}:IF VC= AP and VA= MYA THEN Acab=MA

R_{13}:IF VC= MNA THEN Acab=MA

R_{14}:IF VC= MNA THEN Acab=MA

R_{15}:IF VC= MNA THEN Acab=MA

R_{16}:IF VC= MYA and VA=AP THEN Acab=BA

R_{17}:IF VC= MYA and VA=MNA THEN Acab=BA

R_{18}:IF VC= MYA and VA=MYA THEN Acab=MA

R_{19}:IF TOL=Acep-A and Acab=BA THEN RI=TA

R_{20}:IF TOL=Acep-A and Acab=MA THEN RI=TB

R_{21}:IF TOL=Acep-B and Acab=BA THEN RI=TC

R_{22}:IF TOL=Acep-B and Acab=MA THEN RI=TC

R_{23}:IF TOL= NA and Acab=BA THEN RI=TD

R_{24}:IF TOL= NA and Acab=MA THEN RI=TD

Mediante las reglas de producción anteriores y basándonos en las cuatro reglas de producción básicas podemos modelar la red de Petri (ver figura 3.4).

3.3. Razonamiento con Redes de Petri Difusas

Por muchos años, uno de los tópicos de investigación de inteligencia artificial (IA) ha sido el desarrollo de notación lo suficientemente precisa para representación de conocimiento y razonamiento. Varios algoritmos de razonamiento difuso han sido propuestos por diversos autores en [48] Yeung y Tsang proponen un algoritmo de razonamiento difuso basado en grados de similaridad usando redes de Petri difusas, sin embargo, los factores de certeza de las reglas de producción difusas y los pesos de las proposiciones están restringidos y estos son representados por valores reales entre cero y uno. En [7] Chen, Ke y Chang proponen un algoritmo de razonamiento difuso, en el cual, el valor de verdad de un lugar objetivo puede ser evaluado a partir de un lugar inicial, siempre y cuando el lugar objetivo sea alcanzable a partir de el lugar inicial. En [5] Chen presenta un algoritmo de razonamiento difuso hacia atrás, en él, el valor de verdad de cualquier lugar objetivo puede ser evaluado, elaborando un árbol de nodos con los lugares alcanzables e inmediatamente alcanzables hacia atrás a partir del lugar objetivo hasta llegar a un conjunto de lugares iniciales. Una vez construido el árbol de nodos se pide al usuario introduzca los valores de verdad de los lugares iniciales y con ellos se calcula el valor de verdad del lugar objetivo. En [6] Chen perfecciona su algoritmo asociando pesos a los lugares. Aun más, los valores de verdad asociados a las proposiciones ya no están restringidos a valores entre cero y uno, Estos son representados por números

difusos. En [42] se introducen las redes de Petri adaptativas, las cuales además de contar con las propiedades de las redes de Petri poseen la habilidad de aprender como las redes neuronales. En esta sección se estudian las características principales de tres algoritmos de razonamiento difuso, técnicas de razonamiento así como las ventajas y desventajas de cada algoritmo.

3.3.1. Algoritmo de Razonamiento Hacia atrás (RDHAP).

El algoritmo de razonamiento difuso hacia atrás fue propuesto en [5]. Este algoritmo puede realizar razonamiento difuso hacia atrás para evaluar el grado de verdad de cualquier proposición especificada por el usuario. Sin embargo, las técnicas de evaluación empleadas por Chen pueden llegar a dar resultados inconclusos, ya que el grado de verdad de las proposiciones consecuentes es evaluado multiplicando el grado de verdad máx mín de las proposiciones antecedentes por el factor de certeza de la regla. El producto puede llegar a ser muy pequeño cuando existen varios niveles de razonamiento. Cuando se compara un valor de verdad consecuente con su valor frontera, el valor de verdad puede llegar a ser más pequeño que el valor frontera y entonces provocar un resultado inconcluso. Además en el algoritmo propuesto por Chen las reglas de producción no tienen asociados pesos a sus proposiciones y los valores que maneja tanto en valores de verdad, factores de certeza y valores frontera son valores reales definidos en [0,1] y no números difusos.

El algoritmo que presentamos puede realizar razonamiento difuso hacia atrás para evaluar el grado de verdad de cualquier proposición especificada por el usuario. Asumiendo que deseamos conocer el valor de verdad de la proposición d_s, en este caso el lugar p_s es llamado el lugar objetivo. El algoritmo puede realizar razonamiento difuso hacia atrás generando un grafo Y-O y preguntando al usuario el grado de verdad de otras proposiciones las cuales pueden inferir en el grado de verdad de la proposición d_s. El algoritmo de razonamiento difuso hacia atrás es expresado por una grafo difuso Y-O. Cada nodo en el grafo lo denotaremos por una tripleta (p_k,CIAA (p_k),$\alpha(p_k)$), donde $p_k \in P$, CIAA(p_k) denota el conjunto de lugares inmediatamente alcanzables hacia atrás a partir de p_k, $\alpha(p_k)$ denota el grado de verdad

de la proposición d_k, donde $\alpha(p_k)$ es un número difuso definido en el universo [0,1]. Sea CF_{xy} el factor de certeza asociado a la transición que se encuentra entre los lugares p_x y p_y. Asumiendo que un número triangular difuso puede ser parametrizado por una tripleta (a, b, c) donde la función de pertenencia de un número triangular difuso A esta definida por las ecuaciones (2.14) y (2.20).

Técnicas de Razonamiento Difuso.

Los tipos de reglas que usaremos en este algoritmo, así como el calculo del valor de verdad de las proposiciones son mostradas a continuación:

Tipo 1.

$$R_i : SI\ d_1\ Entonces\ d_2\ (CF = \mu_i) \tag{3.6}$$

Asumiendo que el valor de μ_i así como el valor de verdad de d_1 y d_2 son números difusos triangulares, esto es, $\mu_i = (a_{\mu_i}, b_{\mu_i}, c_{\mu_i})$, $\alpha(d_1) = (a_{d1}, b_{d1}, c_{d1})$. Entonces, el valor de verdad de la proposición d_2 puede ser evaluado y este es igual a:

$$\alpha(d_2) = (a_{d2}, b_{d2}, c_{d2}) \text{ donde : } \begin{array}{l} a_{d2} = a_{d1} \cdot a_{\mu_i} \\ b_{d2} = b_{d1} \cdot b_{\mu_i} \\ c_{d2} = c_{d1} \cdot c_{\mu_i} \end{array} \tag{3.7}$$

El proceso de razonamiento de este tipo de regla puede ser modelado por una red de Petri difusa marcada como lo muestra la figura 3.5 (a) y (b).

Tipo 2.

$$R_i : SI\ d_1\ Y\ d_2\ Y, .., Y\ d_j\ Entonces\ d_z\ (CF = \mu_i), w_1, w_2, .., w_j \tag{3.8}$$

Asumiendo que $w_{1,}w_2, \ldots, w_j$ son los pesos asociados a las proposiciones $d_1, d_2, \ldots, d_j$, donde $w_i = (a_{wi}, b_{wi}, c_{wi})$ y estos son números triangulares difusos, $\alpha(d_1), \alpha(d_2), \ldots, \alpha(d_j)$ son los valores de verdad de las proposiciones $d_1, d_2, \ldots, d_j$. Entonces el valor de verdad de la proposición d_z puede ser calculado basados en [20] y este es igual a:

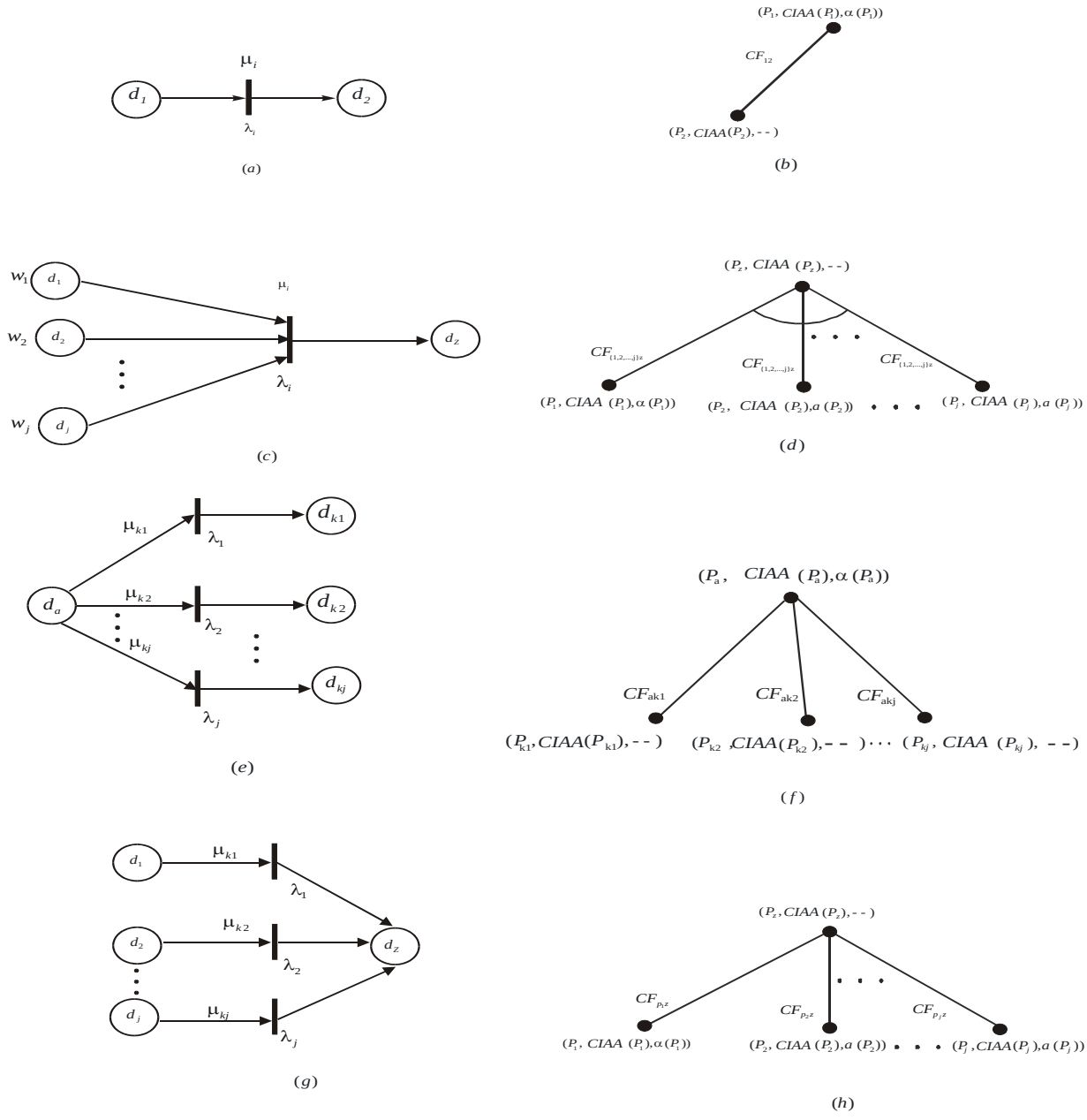


Figura 3.5: Representación de las reglas de producción difusas.

$$\alpha(d_z) = (a_{dz}, b_{dz}, c_{dz}) \text{ donde: } \begin{array}{l} a_{dz} = \left[\dfrac{\sum_{j=1}^{n} (a_{d_j} \cdot a_{w_j})}{\sum_{j=1}^{n} (c_{w_j})} \right] \cdot a_{\mu i} \\ b_{dz} = \left[\dfrac{\sum_{j=1}^{n} (b_{d_j} \cdot b_{w_j})}{\sum_{j=1}^{n} (b_{w_j})} \right] \cdot b_{\mu i} \\ c_{dz} = \left[\dfrac{\sum_{j=1}^{n} (c_{d_j} \cdot c_{w_j})}{\sum_{j=1}^{n} (a_{w_j})} \right] \cdot c_{\mu i} \end{array} \tag{3.9}$$

El proceso de razonamiento de este tipo de regla puede ser modelado por una red de Petri difusa marcada como lo muestra la figura 3.5 (c) y (d).

Tipo 3.

$$R_i : SI\ d_a\ Entonces\ d_{k1}\ Y\ d_{k2}\ Y, .., Y\ d_{kj}\ (CF = \mu_{k1}, .., \mu_{kj}), Th = \lambda_1, .., \lambda_j \tag{3.10}$$

Asumimos que $\alpha(d_a)$ es el valor de verdad de la proposición d_a. Entonces el valor de verdad de las proposiciones d_{k1}, d_{k2},...d_{kj} puede ser calculado basados en [20] y este es igual a:

$$\alpha(d_{k_j}) = \ (a_{dk_j}, b_{dk_j}, c_{dk_j})\ \ donde: \begin{array}{l} a_{dk_1} = a_{d_a} \cdot a_{\mu_{k1}},\ b_{dk_1} = b_{d_a} \cdot b_{\mu_{k1}},\ c_{dk_1} = c_{d_a} \cdot c_{\mu_{k1}} \\ a_{dk_2} = a_{d_a} \cdot a_{\mu_{k2}},\ b_{dk_2} = b_{d_a} \cdot b_{\mu_{k2}},\ c_{dk_2} = c_{d_a} \cdot c_{\mu_{k2}} \\ \vdots \\ a_{dk_n} = a_{d_a} \cdot a_{\mu_{kn}},\ b_{dk_n} = b_{d_a} \cdot b_{\mu_{kn}},\ c_{dk_n} = c_{d_a} \cdot c_{\mu_{kn}} \end{array} \tag{3.11}$$

El proceso de razonamiento de este tipo de regla puede ser modelado por una red de Petri difusa marcada como lo muestra la figura 3.5 (e) y (f).

Tipo 4.

$$R_i : SI\ d_1 O\ d_2\ O, .., O\ d_j\ Entonces\ d_z\ (CF = \mu_1, .., \mu_j), Th = \lambda_1, .., \lambda_j. \tag{3.12}$$

Asumiendo que $\alpha(d_1), \alpha(d_2), ..., \alpha(d_j)$ son los valores de verdad de las proposiciones $d_1, d_2, .., d_j$, CF_1,CF_2, ..,CF_j son los factores de certeza asociados a las transiciones, entonces el valor de verdad de la proposición d_z puede ser calculado y este es igual a:

$$\alpha(d_z) = (a_{dz}, b_{dz}, c_{dz}) \text{ donde: } \begin{array}{l} a_{dz} = \left[\dfrac{\sum_{j=1}^{n} (a_{d_j} \cdot a_{\mu_j})}{\sum_{j=1}^{n} (c_{\mu_j})} \right] \\ b_{dk} = \left[\dfrac{\sum_{j=1}^{n} (b_{d_j} \cdot b_{\mu_j})}{\sum_{j=1}^{n} (b_{\mu_j})} \right] \\ c_{dk} = \left[\dfrac{\sum_{j=1}^{n} (c_{d_j} \cdot c_{\mu_j})}{\sum_{j=1}^{n} (a_{\mu_j})} \right] \end{array} \tag{3.13}$$

El proceso de razonamiento de este tipo de regla puede ser modelado por una red de Petri difusa marcada como lo muestran las figuras 3.5 (g) y 3.5 (h).

Algoritmo de Razonamiento Difuso Hacia Atrás.

ENTRADA: la proposición objetivo d_j.

SALIDA: el valor de verdad de la proposición d_j.

Paso 1. Marcar el nodo raíz $(p_j, CIAA(p_j), --)$ como un nodo no terminal, donde:

p_j es el lugar objetivo

$CIAA(p_j)$ es el conjunto de lugares inmediatamente alcanzable hacia atrás de p_j.

el símbolo "–" denota que el valor de verdad de la proposición d_j es desconocido.

Paso 2. Se selecciona un nodo no terminal $(p_i, CIAA(p_i), --)$. **Si** $CIAA(p_i) = \varnothing$, **Entonces** se marca el nodo como nodo terminal. De otra manera, **Si** $CIAA(p_i) = \{p_w, ..., \{p_a, p_b, ..., p_k\}, ..., p_z, ..\}$ **Entonces** se incrementa el grafo Y-O como en la figura 3.6, donde $\mathrm{CF}_{wi} = (a_{\mu wi}, b_{\mu wi}, c_{\mu wi})$ denota el valor del factor de certeza asociado a la

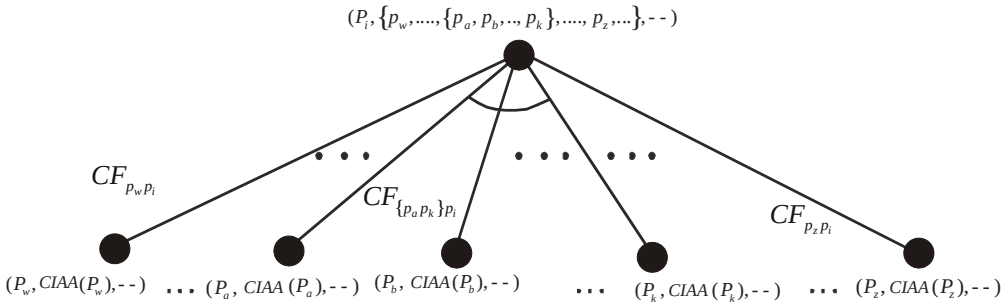


Figura 3.6: Grafo AND-OR.

transición que se encuentra entre los lugares p_w y p_i, $\mathrm{CF}_{\{a,b,..,k\}i}$ denota el valor del factor de certeza asociado a la transición que se encuentra entre los lugares $\{p_a, p_b, .., p_k\}$ y p_i.

Paso 3. Si no existe un nodo no terminal **Entonces** ir al paso 4 de otra manera ir al paso 2.

Paso 4. Para cada nodo terminal $(p_s, \varnothing, --)$ en el grafo difuso generado Y-O, preguntar al usuario el valor de $y_s = (a_{\alpha s}, b_{\alpha s}, c_{\alpha s})$ que es el valor de verdad de la proposición d_s.

Paso 5. Evaluar el valor de $\alpha(p_k)$ para cada nodo no terminal $(p_k, CIAA(p_k), \alpha(p_k))$ en el grafo difuso generado Y-O basado en los tipos de reglas expuestas anteriormente.

Ejemplos:

Ejemplo 3.1 *Veamos el siguiente ejemplo, el cual fue presentado en [42]. Sean* d_1, d_2, d_3, d_4, d_5 *y* d_6 *seis proposiciones, entre las cuales existen las siguientes reglas de producción difusas con pesos:*

R_1:SI d_1 *Entonces* d_4 (μ_1)

R_2:SI d_2 *Y* d_4 *Entonces* d_5 (w_2, w_4, μ_3)

R_3:SI d_3 *O* d_5 *Entonces* d_6 (μ_2, μ_4)

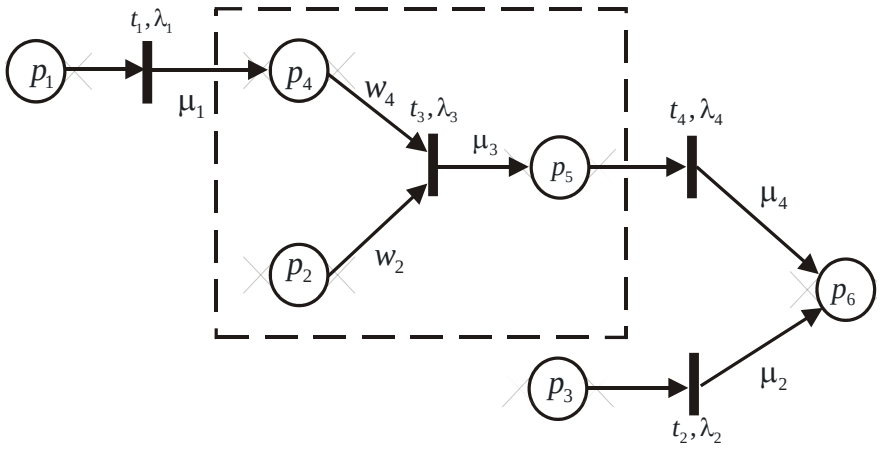


Figura 3.7: Red de Petri que representa las reglas de producción del Ejemplo 3.1.

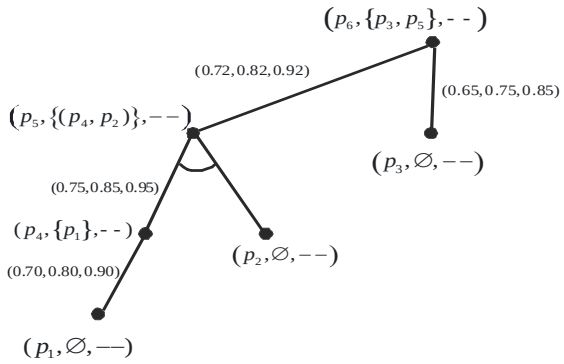
Estas reglas de producción pueden ser representadas por una red de Petri como se muestra en la figura 3.7, donde P=$\{p_1, p_2, p_3, p_4, p_5, p_6\}$, T= $\{t_1, t_2, t_3, t_4\}$, D=$\{d_1, d_2, d_3, d_4, d_5, d_6\}$, $W_I = \{w_2, w_4\}$, $W_O = \{\mu_1, \mu_2, \mu_3, \mu_4\}$. Con tres proposiciones de entrada (p_1, p_2, p_3) y tres proposiciones consecuentes, los datos son dados como:

μ_1 =0.70 0.80 0.90 μ_2 =0.65 0.75 0.85 μ_3 =0.75 0.85 0.95 μ_4 =0.72 0.82 0.92
w_2 =0.27 0.37 0.47 w_4 =0.53 0.63 0.73.

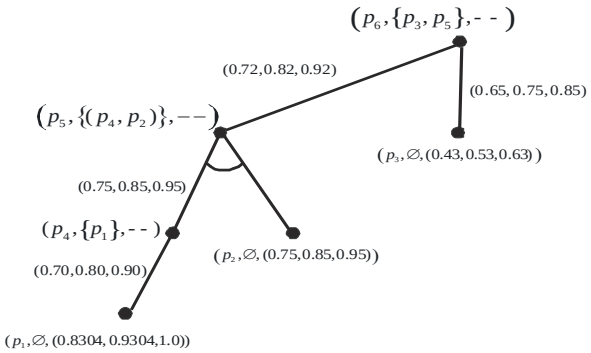
Asumiendo que deseamos conocer el valor de verdad de la proposición p_5 empleamos el algoritmo presentado de donde tenemos los siguientes resultados:

1. Siguiendo los pasos 1,2 y 3 del algoritmo de razonamiento difuso hacia atrás, generamos el grafo difuso Y-O mostrado en la figura 3.8(a).

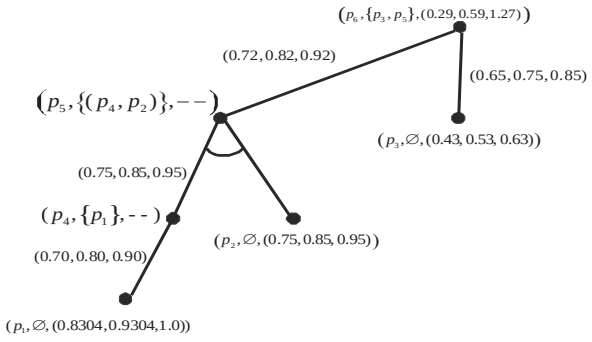
2. De la figura 3.8 (a), podemos observar que existen tres nodos terminales. Realizando el paso 4 del algoritmo el sistema pedirá al usuario introduzca el valor de verdad de las proposiciones d_1, d_2, d_3. suponiendo que el usuario introduce los valores de verdad $\alpha(d_1) = (0.8304 \;\; 0.9304 \;\; 1.0)$, $\alpha(d_2) = (0.75 \;\; 0.85 \;\; 0.95)$, $\alpha(d_3) = (0.43 \;\; 0.53 \;\; 0.63)$, entonces el nodo terminal quedaría como en la figura 3.8 (b)



(a)



(b)



(c)

Figura 3.8: Grafo AND-OR del Ejemplo 3.1.

3. Realizando el paso 5 del algoritmo tenemos los siguientes resultados:

   - Evaluando el valor de verdad de la proposición d_4 tenemos:

     $a_{d4} = a_{d1} \cdot a_{\mu 1} = (0.8304) \cdot (0.70) = 0.58$

     $b_{d4} = b_{d1} \cdot b_{\mu 1} = (0.9304) \cdot (0.80) = 0.74$

     $c_{d4} = c_{d1} \cdot c_{\mu 1} = (1.0) \cdot (0.90) = 0.90$

   - Evaluando el valor de verdad de la proposición d_5 tenemos:

     $$a_{d5} = \left[\frac{(a_{d4} \cdot a_{w4}) + (a_{d2} \cdot a_{w2})}{c_{w4} + c_{w2}}\right] \cdot a_{\mu 3} = 0.32$$

     $$b_{d5} = \left[\frac{(b_{d4} \cdot b_{w4}) + (b_{d2} \cdot b_{w2})}{b_{w4} + b_{w2}}\right] \cdot b_{\mu 3} = 0.66$$

     $$c_{d5} = \left[\frac{(c_{d4} \cdot c_{w4}) + (c_{d2} \cdot c_{w2})}{a_{w4} + a_{w2}}\right] \cdot c_{\mu 3} = 1.31$$

   - Evaluando el valor de verdad de la proposición d_6 tenemos:

     $$a_{d6} = \left[\frac{(a_{d5} \cdot a_{\mu 4}) + (a_{d3} \cdot a_{\mu 2})}{c_{\mu 4} + c_{\mu 2}}\right] = 0.29$$

     $$b_{d6} = \left[\frac{(b_{d5} \cdot b_{\mu 4}) + (b_{d3} \cdot b_{\mu 2})}{b_{\mu 4} + b_{\mu 2}}\right] = 0.59$$

     $$c_{d6} = \left[\frac{(c_{d5} \cdot c_{\mu 4}) + (c_{d3} \cdot c_{\mu 2})}{a_{\mu 4} + a_{\mu 2}}\right] = 1.27$$

4. En la figura 3.8(c) podemos observar que el nodo raíz del grafo difuso And-Or es (p_6,$\{p_3, p_5\}$,(0.29 0.59 1.27)), *i.e.*, el grado de verdad de la proposición d_6 es el número difuso triangular (0.29 0.59 1.27), ya que el universo en el cual se encuentra definida es U=[0,1] la función de membresía del valor de verdad difuso de la proposición d_6 se muestra en la figura usando una línea negra remarcada.

3.3.2. Algoritmo de Razonamiento Difuso con Pesos (RDPP)

El algoritmo presentado en esta sección es una extensión del presentado en [6] con la diferencia de que en la regla de tipo 4, Chen calcula el valor de verdad final como el máximo

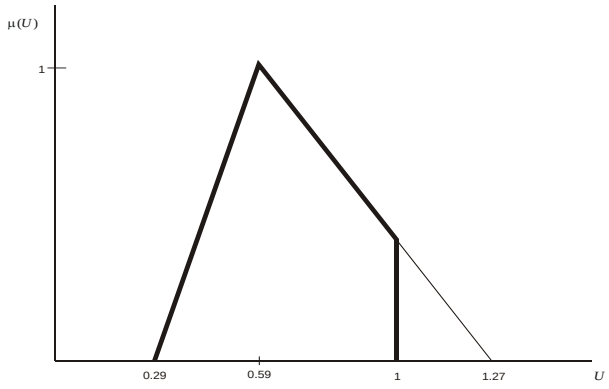


Figura 3.9: Valor de verdad de la proposición d_6.

de todas las transiciones disparadas. En este algoritmo se calcula el valor de verdad mediante el centro de gravedad de las transiciones disparadas. Además, a diferencia del algoritmo de RDHA el valor de verdad final se calcula como el valor de verdad final de cada ruta y no como el valor final del conjunto de rutas.

Técnicas de Razonamiento Difuso con Pesos.

Las técnicas de razonamiento difuso con pesos son iguales a las presentadas en las ecuaciones (3.6), (3.8), (3.10) y (3.12). El calculo de los valores de verdad consecuentes son calculados con las ecuaciones (3.7), (3.9), (3.11) y (3.13), las figuras 3.16, 3.17, 3.18 y 3.19 ilustran el calculo de los valores de verdad consecuentes.

Algoritmo de Razonamiento Difuso con Pesos.

Sea t_a una transición y p_i, p_j y p_k tres lugares en una RPDP. Si $p_i \in I(t_a)$ y $p_k \in O(t_a)$, entonces p_k es llamado inmediatamente alcanzable desde p_i. Si p_k es inmediatamente alcanzable desde p_i y p_j es inmediatamente alcanzable desde p_k, entonces p_j se dice que es alcanzable desde p_i. El conjunto de lugares que son inmediatamente alcanzables desde un lugar p_i se dice que es un conjunto inmediatamente alcanzable de p_i y es denotado por

$CIA(p_i)$. El conjunto de lugares que son alcanzables desde un lugar p_i se dice conjunto alcanzable de p_i y es denotado por $CA(p_i)$. Sea t_a una transición y sea p_i y p_k lugares en una RPDP. Si $p_i \in I(t_a)$ y $p_k \in I(t_a)$ entonces p_i y p_k son llamados lugares adyacentes con respecto a t_a.

Asumimos que el factor de certeza (CF) es un número difuso definido en [0,1], esto es, CF_1 lo podemos parametrizar como $(a_{\mu 1}, b_{\mu 1}, c_{\mu 1})$

ENTRADA: El valor de verdad difuso y_s de la proposición d_s, donde y_s es un número difuso definido en $[0, 1]$.

SALIDA: El valor de verdad difuso de la proposición d_j.

Paso 1 $(p_s, \alpha(p_s), CIA(p_s))$ es el nodo raíz, donde:

p_s es el lugar inicial y $\beta(p_s) = d_s$;

$\alpha(p_s) = y_s$, que es el valor de verdad de la proposición d_s;

$CIA(p_s)$ es el conjunto inmediato alcanzable del lugar p_s

Paso 2 Seleccionamos un nodo no-terminal $(p_i, \alpha(p_i), CIA(p_i))$

Si $CIA(p_i) = \emptyset$ ó $\forall p_k \in CIA(p_i)$ el lugar objetivo $p_j \notin CA(p_k)$ **Entonces** marcarlo como nodo terminal

Si El lugar objetivo $p_j \in CIA(p_i)$ y $CF = \mu$ **Entonces** creamos un nuevo nodo $(p_j, \alpha(p_j), CIA(p_j))$ en el árbol y un arco etiquetado μ es dirigido desde el nodo $(p_i, \alpha(p_i), CIA(p_i))$ a el nodo $(p_j, \alpha(p_j), CIA(p_j))$, donde $\alpha(p_j) = (a_{\alpha j}, b_{\alpha j}, c_{\alpha j})$ y estos a su vez se calculan como $a_j = a_{\alpha i} \cdot a_{\mu i}$, $b_j = b_{\alpha i} \cdot b_{\mu i}$, $c_k = c_{\alpha i} \cdot c_{\mu i}$.

En este caso marcamos el nodo $(p_i, \alpha(p_i), CIA(p_i))$ como un nodo terminal y el nodo $(p_j, \alpha(p_j), CIA(p_j))$ es llamado nodo exitoso.

Para cada lugar $p_k \in CIA(p_i)$, donde $p_k \neq p_i$,

Si $AP_{ik} = \emptyset$ (esto es p_i no tiene lugares adyacentes) y el lugar objetivo $p_j \in CA(p_k)$, $CF_{ik} = \mu$ y p_k no aparece en ningún nodo sobre el camino entre el nodo raíz $(p_s, \alpha(p_s), CIA(p_s))$ y el nodo seleccionado $(p_i, \alpha(p_i), CIA(p_i))$, **Entonces**, crear

un nuevo nodo $(p_k, \alpha(p_k), CIA(p_k))$ en el árbol y un arco etiquetado μ es dirigido desde el nodo $(p_i, \alpha(p_i), CIA(p_i))$ a el nodo $(p_k, \alpha(p_k), CIA(p_k))$, donde $\alpha(p_j)$ es calculado como una regla de Tipo 1. En este caso el nodo $(p_k, \alpha(p_k), CIA(p_k))$ es llamado nodo no-terminal.

De otra manera.

Si $AP_{ik} = \{p_a, p_b, ..., p_z\}$ y el lugar objetivo $p_j \in CA(p_k)$, **Entonces**, requerir al usuario introducir el valor de verdad de las proposiciones $d_a, d_b, ..., d_z$ respectivamente.

Si $CF_{ik} = \mu$, **Entonces**, crear un nuevo nodo $(p_k, \alpha(p_k), CIA(p_k))$ en el árbol y un arco etiquetado μ es dirigido desde el nodo $(p_i, \alpha(p_i), CIA(p_i))$ a el nodo $(p_k, \alpha(p_k), CIA(p_k))$, donde ,$\alpha(p_k) = (a_{\alpha k}, b_{\alpha k}, c_{\alpha k})$, donde $a_{\alpha k}, b_{\alpha k}$ y $c_{\alpha k}$ se calculan de la misma forma que una regla de Tipo 2. En este caso el nodo $(p_i, \alpha(p_i), CIA(p_i))$ es un nodo terminal, mientras que el nodo $(p_k, \alpha(p_k), CIA(p_k))$ es un nodo no-terminal.

Paso 3 **Si** no existe un nodo no terminal, **Entonces**, ir al Paso 4, **De otra manera**, ir al Paso 2.

Paso 4 **Si** no hay nodos exitosos (si no existe relación antecedente consecuente de la proposición d_s a d_j) **Entonces**, parar, **De otra manera**, el camino desde el nodo raíz a el nodo exitoso es llamado Forma de Razonamiento.

Sea Q un conjunto de nodos exitosos, tal que:

$$Q = \{(p_j, s_1, CIA(p_j))(p_j, s_2, CIA(p_j)), ..., (p_j, s_m, CIA(p_j))\}$$

donde s_i es el número triangular difuso de la i-ésima ruta.

Ejemplos:

Ejemplo 3.2 *Sean $d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8$ y d_9 nueve proposiciones, entre las cuales existen las siguientes reglas de producción difusas con pesos:*

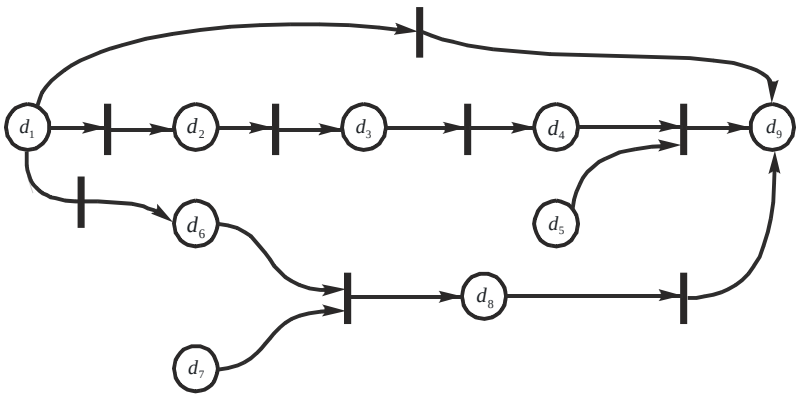


Figura 3.10: Red de Petri difusa representando las reglas de producción del Ejemplo 3.2.

R_1:SI d_1 Entonces d_2 [CF=(0.80 0.90 1.0)]

R_2:SI d_2 Entonces d_3 [CF=(0.80 0.90 1.0)]

R_3:SI d_3 Entonces d_4 [CF=(0.80 0.90 1.0)]

R_4:SI d_4 Y d_5 Entonces d_9 [CF=(0.70 0.80 0.90)]

R_5:SI d_1 Entonces d_6 [CF=(0.85 0.95 1.0)]

R_6:SI d_6 Y d_7 Entonces d_8 [CF=(0.70 0.80 0.90)]

R_7:SI d_8 Entonces d_9 [CF=(0.60 0.70 0.80)]

R_8:SI d_1 Entonces d_9 [CF=(0.60 0.70 0.80)]

Estas reglas pueden ser modeladas por una red de Petri difusa mostrada en la figura 3.10. Asumiendo que deseamos conocer el valor de verdad de la proposición d_9 y teniendo los pesos de las proposiciones $w_4 = (0.80\ \ 0.90\ 1.0)$, $w_5 = (0.60\ \ 0.70\ \ 0.80)$, $w_6 = (0.50\ \ 0.60\ 0.70)$, $w_7 = (0.70\ \ 0.80\ \ 0.90)$, $w_1 = w_2 = w_3 = w_8 = w_9 = (1.0,1.0,1.0)$ así como los valores de verdad de las proposiciones $p_1 = (0.80\ \ 0.90\ \ 1.0)$, $p_5 = (0.70\ \ 0.80\ \ 0.90)$, $p_7 = (0.70\ 0.80\ \ 0.90)$, $p_8 = (0.75\ \ 0.85\ \ 0.95)$ empleando el algoritmo de propuesto tenemos el árbol de la figura 3.11. De donde tenemos los siguientes resultados:

$Q = \{(p_9, (0.30\ \ 0.58\ \ 1.10), \_\_\_), (p_9(0.22\ \ 0.46\ \ 0.90)), (p_9(0.48\ \ 0.63\ \ 0.80))\}$.

Ya que $s_i \in [0, 1]$ los valores de verdad difusos de las rutas que van a la proposición d_9 quedarían como se muestra en la figura 3.12

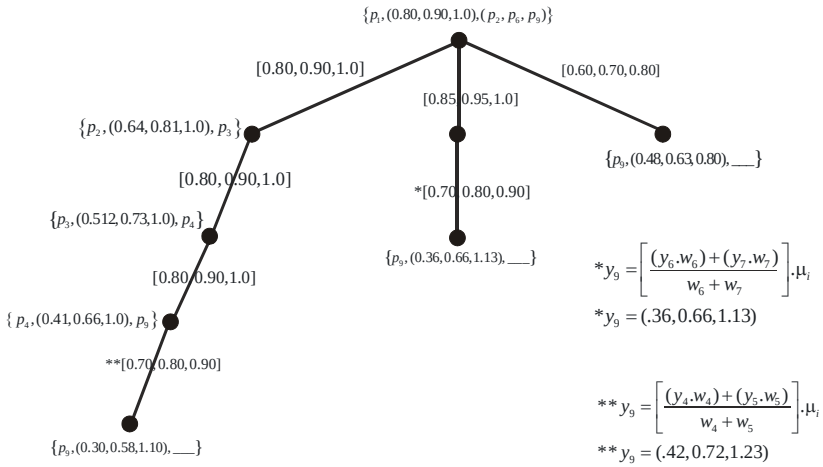


Figura 3.11: Arbol de nodos del Ejemplo 3.2.

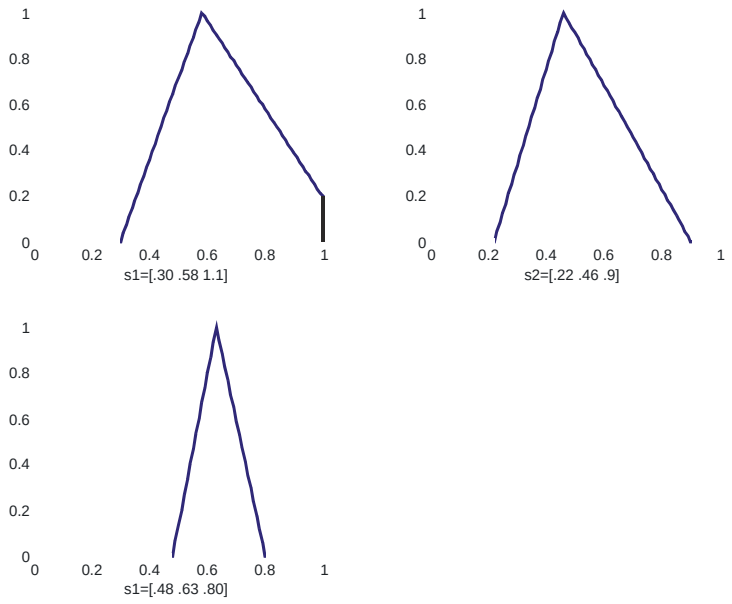


Figura 3.12: Funciones de membresía de las rutas de la proposición d_9.

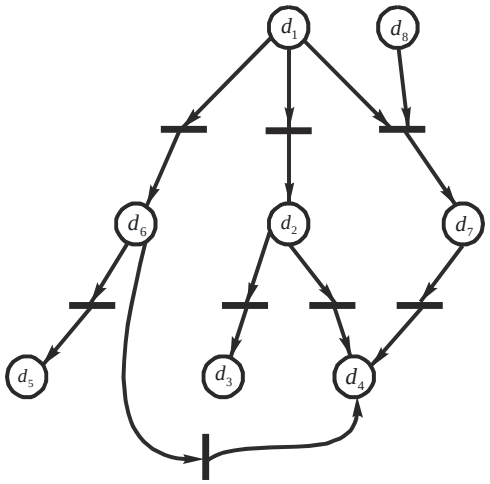


Figura 3.13: Red de Petri difusa de las reglas de producción del Ejemplo 3.3.

Ejemplo 3.3 *Sean $d_1, d_2, d_3, d_4, d_5, d_6, d_7$ y d_8 ocho proposiciones, entre las cuales existen las siguientes reglas de producción difusas con pesos:*

R_1:SI d_1 Entonces d_2 [CF=(0.75 0.85 0.95)]

R_2:SI d_2 Entonces d_3 [CF=(0.70 0.80 0.90)]

R_3:SI d_2 Entonces d_4 [CF=(0.70 0.80 0.90)]

R_4:SI d_1 Entonces d_6 [CF=(0.80 0.90 1.0)]

R_5:SI d_6 Entonces d_4 Y d_5 [CF=(0.85 0.95 1.0)]

R_6:SI d_8 Y d_1 Entonces d_7 [CF=(0.80 0.90 1.0)]

R_7:SI d_7 Entonces d_4 [CF=(0.80 0.90 1.0)]

Estas reglas pueden ser modeladas por una red de Petri difusa mostrada en la figura 3.13

Asumiendo que deseamos conocer el valor de verdad de la proposición d_4 y teniendo los pesos de las proposiciones $w_1 = (0.40 \;\; 0.50 \;\; 0.60)$, $w_8 = (0.60 \;\; 0.70 \;\; 0.80)$, $w_2 = w_3 = w_4 = w_5 = w_6 = w_7 = (1.0 \;\; 1.0 \;\; 1.0)$ así como los valores de verdad de las proposiciones $p_1 = (0.70 \;\; 0.80 \;\; 0.90)$, $p_8 = (0.80 \;\; 0.90 \;\; 1.0)$ empleando el algoritmo propuesto tenemos el árbol de la figura 3.14. De donde tenemos los siguientes resultados:

$Q = \{(p_4, (0.37 \;\; 0.54 \;\; 0.77), \_\_\_), \; (p_4(0.48 \;\; 0.68 \;\; 0.90)\_\_\_), \; (p_4(0.34 \;\; 0.69 \;\; 1.34)\_\_\_)\}$.

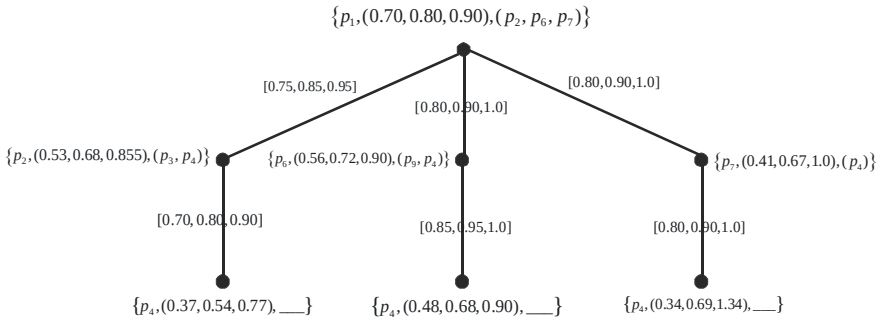


Figura 3.14: Árbol de nodos del Ejemplo 3.3.

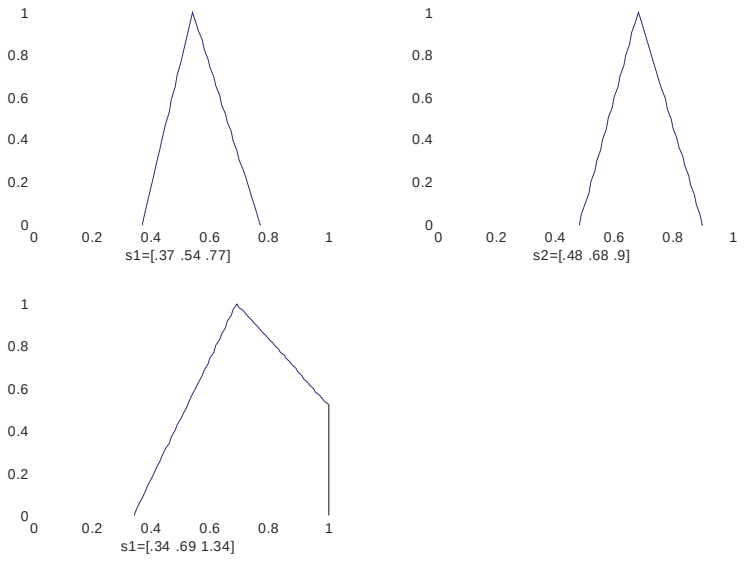
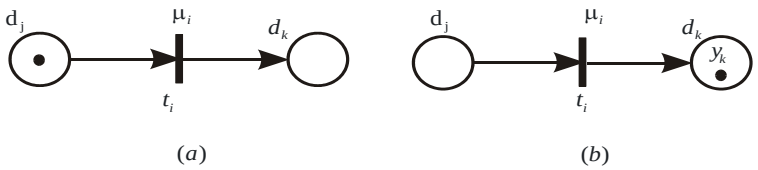


Figura 3.15: Funciones de membresía de las rutas de la proposición d_4.



$$y_k = (a_{\alpha k}, b_{\alpha k}, c_{\alpha k}) \ \ donde: \quad \begin{aligned} a_k &= a_{\alpha j} \cdot a_{\mu i} \\ b_k &= b_{\alpha j} \cdot b_{\mu i} \\ c_k &= c_{\alpha j} \cdot c_{\mu i} \end{aligned}$$

Figura 3.16: Disparo de una RPD del *Tipo 1.* (a) Antes del disparo de la transición t_i. (b) Después del disparo de la transición t_i.

Ya que $s_i \in [0, 1]$ los valores de verdad difusos de las rutas que van a la proposición d_4 quedarían como se muestra en la figura 3.15

3.3.3. Algoritmo de Razonamiento Difuso con Pesos y Valores Frontera (RDPF)

El algoritmo presentado en esta sección es una extensión del presentado en [42] con la diferencia de que los valores de verdad, factores de certeza, pesos y valores frontera son valores en [0,1] y no números difusos. En esta sección se presenta una algoritmo de razonamiento difuso trabajando con números difusos triangulares.

Técnicas de Razonamiento Difuso con Pesos y Valores Frontera.

Las técnicas de razonamiento difuso con pesos son iguales a las presentadas en las ecuaciones (3.6), (3.8), (3.10) y (3.12). Los cuatro Tipos de razonamiento difuso con pesos pueden ser modelados con redes de Petri difusas con pesos de la forma siguiente:

1. Usando una RPD el proceso de razonamiento difuso con pesos del Tipo 1 puede ser modelado como en la figura 3.16.

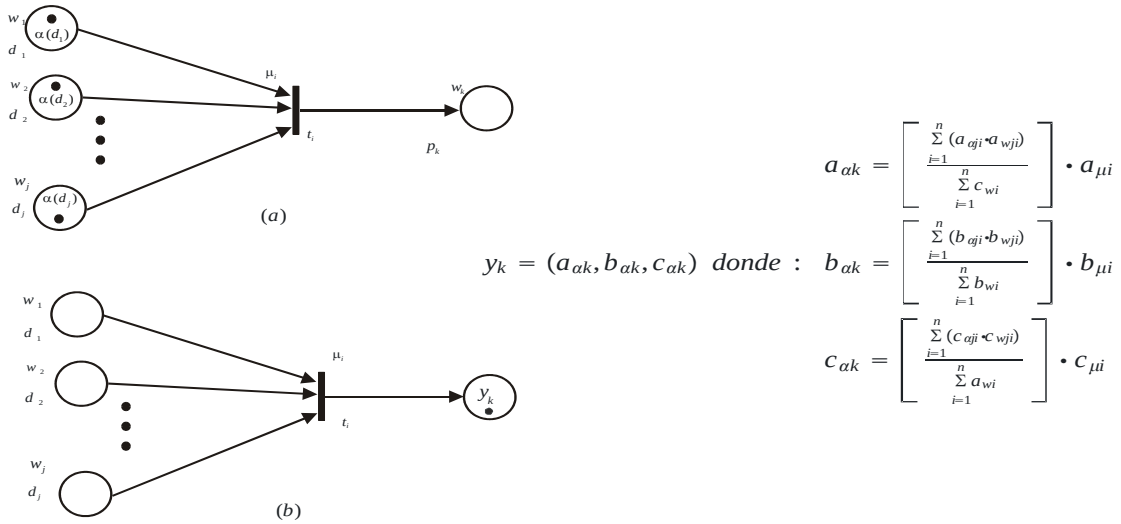


Figura 3.17: Disparo de una RPD del *Tipo 2.* (a) Antes del disparo de la transición t_i. (b) Después del disparo de las transición t_i.

2. Usando una RPD el proceso de razonamiento difuso con pesos del Tipo 2 puede ser modelado como en la figura 3.17.

3. Usando una RPD el proceso de razonamiento difuso con pesos del Tipo 3 puede ser modelado como en la figura 3.18.

4. Usando una RPD el proceso de razonamiento difuso con pesos del Tipo 4 puede ser modelado como en la figura 3.19.

Dividiendo el conjunto de lugares P en tres partes $P = P_I \cup P_{Int} \cup P_O$, donde P es un conjunto de lugares de una RPDA; $P_I = \{p \in P \mid {}^\bullet p = 0\}$, si $p \in P_I$ este lugar es llamado lugar de entrada; $P_{Int} = \{p \in P \mid {}^\bullet p \neq 0 \;\; y \;\; p^\bullet \neq 0\}$, si $p \in P_{Int}$ este lugar es llamado lugar interior; $P_O = \{p^\bullet = 0\}$, si $p \in P_O$ este lugar es llamado lugar de salida.

Definición 3.3 *(lugar fuente y lugar pozo) Un lugar p es llamado un lugar fuente si no tiene transiciones de entrada, y es llamado lugar pozo si no tiene transiciones de salida.*

Un lugar fuente corresponde a una precondición mientras que un lugar fuente corresponde a una proposición consecuente.

Definición 3.4 *(Ruta) Dado un lugar p_i, una cadena de transiciones $t_1, t_2, ..., t_n$ es llamada una ruta de p_i si p_i puede llevar una señal a partir de un grupo de lugares fuente $p_{i1}, p_{i2}, .., p_{in}$ hasta un conjunto de lugares pozo $(p_{j1}, p_{j2}, .., p_{jn})$ mediante el disparo de estas transiciones. Esto es, si p_{jn} es alcanzable a partir de p_{in} entonces se dice que existe una ruta entre estos dos lugares.*

Si una cadena de transiciones dispara en secuencia, esta es llamada ruta activa.

Definición 3.5 *$\forall t \in T$, t es habilitada si $\forall p_i \in I(t), \alpha(p_i) > 0, i = 1, 1, ..., n$. Donde $\alpha(p_i)$ es un número difuso triangular [20].*

Definición 3.6 *Sean A y B números difusos triangulares parametrizados por (a_1, a_2, a_3) y (b_1, b_2, b_3) respectivamente, se dice que $A > B$ si y solamente si $a_1 > b_1, a_2 > b_2$ y $a_3 > b_3$. De otra manera se dice que A y B no son comparables.*

Definición 3.7 *Cuando una transición t es habilitada, esta produce un nuevo valor de verdad $y(t)$, donde: $y(t)$ es un número difuso triangular, siendo el valor de verdad consecuente es calculado como en las ecuaciones (3.7), (3.9), (3.11) y (3.13) si $y(t) > Th(t)$. De lo contrario $y(t) = [0\ 0\ 0]$.*

Algoritmo de Razonamiento Difuso.

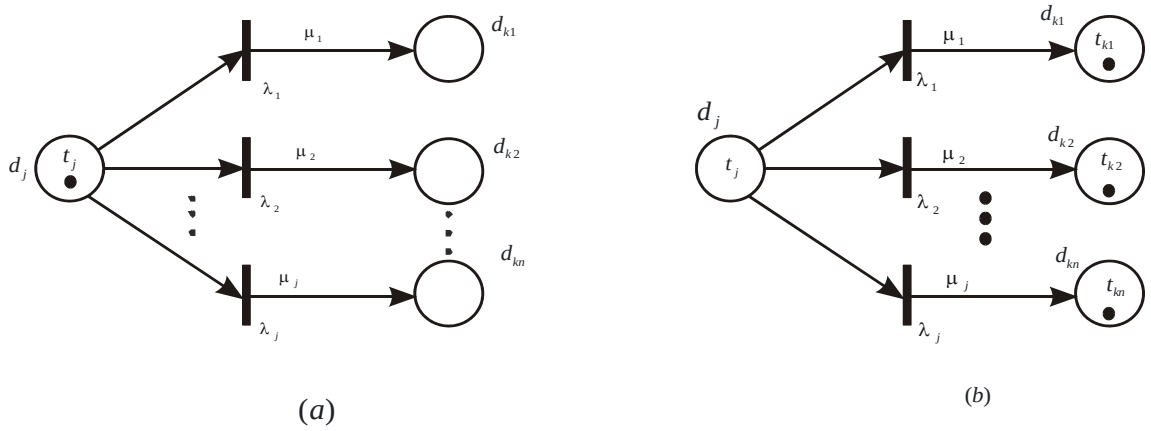
ENTRADA: Los valores de verdad de un conjunto de proposiciones antecedentes.

SALIDA: Los valores de verdad de un conjunto de proposiciones consecuentes.

Paso 1. Construir el conjunto de lugares de entrada P_{UI} donde $P_{UI} = \{p \in P \mid {}^\bullet p = \emptyset\}$

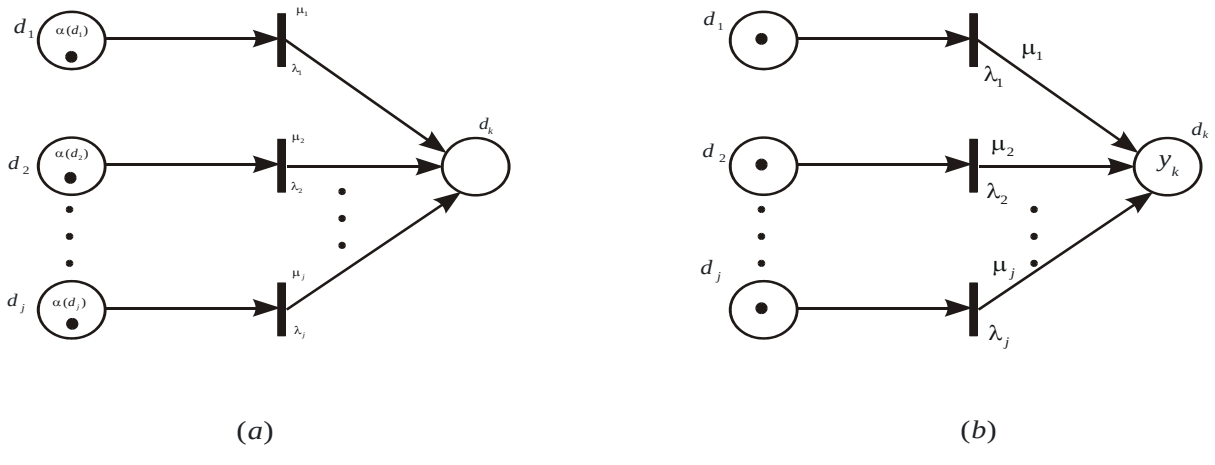
Paso 2. Construir un conjunto de transiciones iniciales habilitadas $T_{Initial}$, donde:

$$T_{Initial} = \{t \in T \mid {}^\bullet t \cap P_{UI} \neq \emptyset \ \ y \ {}^\bullet t \cap P_{Int} \ = \emptyset\}$$



$$a_{\alpha k_1} = a_{\alpha j} \cdot a_{\mu 1},\ b_{\alpha k_1} = b_{\alpha j} \cdot b_{\mu 1},\ c_{\alpha k_1} = c_{\alpha j} \cdot c_{\mu 1}$$
$$a_{\alpha k_2} = a_{\alpha j} \cdot a_{\mu 2},\ b_{\alpha k2} = b_{\alpha j} \cdot b_{\mu 2},\ c_{\alpha k2} = c_{\alpha j} \cdot c_{\mu 2}$$
$$\vdots$$
$$a_{\alpha k_n} = a_{\alpha j} \cdot a_{\mu n},\ b_{\alpha kn} = b_{\alpha j} \cdot b_{\mu n},\ c_{\alpha kn} = c_{\alpha j} \cdot c_{\mu n}$$

Figura 3.18: Disparo de una RPD del *Tipo 3.* (a) Antes del disparo de las transiciones $t_{j1}, t_{j2}, .., t_{jn}$. (b) Después del disparo de las transiciones $t_{j1}, t_{j2}, .., t_{jn}$.



$$y_k = (a_{\alpha k}, b_{\alpha k}, c_{\alpha k}) \ \ donde: \quad a_{\alpha k} = \frac{a_{\alpha j1} \cdot a_{\mu_i}}{\sum_{i=1}^{n} c_{wji}} + \frac{a_{\alpha j2} \cdot a_{\mu_i}}{\sum_{i=1}^{n} c_{cwji}} + \ldots + \frac{a_{\alpha jn} \cdot a_{\mu_i}}{\sum_{i=1}^{n} c_{cwji}}$$

$$b_{\alpha k} = \frac{b_{\alpha j1} \cdot b_{\mu_i}}{\sum_{i=1}^{n} b_{wji}} + \frac{b_{\alpha j2} \cdot b_{\mu_i}}{\sum_{i=1}^{n} b_{cwji}} + \ldots + \frac{b_{\alpha jn} \cdot b_{\mu_i}}{\sum_{i=1}^{n} b_{cwji}}$$

$$c_{\alpha k} = \frac{c_{\alpha j1} \cdot c_{\mu_i}}{\sum_{i=1}^{n} a_{wji}} + \frac{c_{\alpha j2} \cdot c_{\mu_i}}{\sum_{i=1}^{n} a_{cwji}} + \ldots + \frac{c_{\alpha jn} \cdot c_{\mu_i}}{\sum_{i=1}^{n} a_{cwji}}$$

Figura 3.19: Disparo de una RPD del *Tipo 4.* (a) Antes del disparo de las transiciones $t_{j1}, t_{j2}, .., t_{jn}$. (b) Después del disparo de las transiciones $t_{j1}, t_{j2}, .., t_{jn}$.

Paso 3. Encontrar la transición habilitada actual $T_{Current}$ donde:

$$T_{Current} = \{t \in T_{Initial} \mid \forall p_i \in^{\bullet} t,\ m(p_i) > 0 \ \text{ y } \ y(t) > Th(t)\}$$

Paso 4. Calcular los nuevos valores de verdad de acuerdo a las técnicas de razonamiento difuso presentadas

Paso 5. Remover y adicionar las señales de acuerdo a las técnicas de razonamiento difuso.

Paso 6. Sea $T = T - T_{Current}$, $P = P - {}^{\bullet}T_{Current}$.

Paso 7. Ir al Paso 3 y repetirlo hasta que $T_{Current} = 0$

Ejemplos:

Ejemplo 3.4 *Como ejemplo usaremos el sistema de admisión de alumnos graduados de preparatoria, el cual fue presentado en [19] y modelado con redes de Petri algebraicas. Una universidad usa el siguiente criterio para evaluar las solicitudes de ingreso:*

1. Un solicitante con punto promedio de calidad (grade point average) GPA >3.50, un buen expediente de exámenes del graduado (graduate record examination) GRE y fuertes (buenas) referencias será aceptado y premiado con un asistente(ACA).

2. Un solicitante con un GPA >3.50 y con un buen GRE o fuertes referencias será aceptado (ACB).

3. Un solicitante con un GPA >3.00 pero ≤3.50, un buen GRE, y fuertes referencias será aceptado (ACB).

4. Un solicitante con GPA >3.00 pero ≤3.50 con un buen GRE o fuertes referencias será aceptado con condiciones (ACC).

5. Un solicitante con GPA ≤3.00 pero con un buen GRE y fuertes referencias será aceptado con condiciones (ACC).

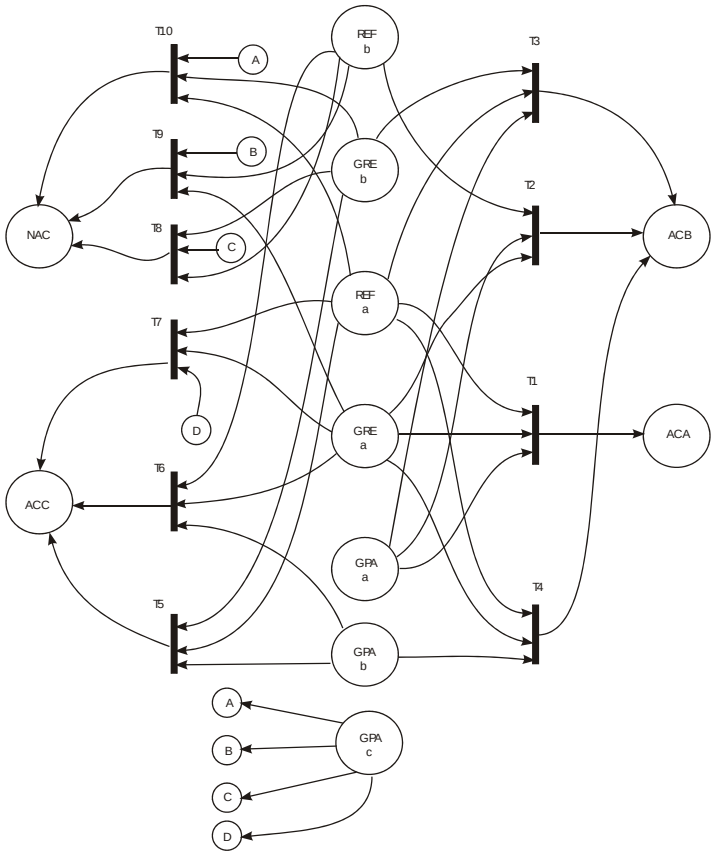


Figura 3.20: Red de Petri difusa del Ejemplo 3.4

6. De cualquier otra forma un solicitante no será aceptado (NAC).

Trasladando el criterio de evaluación a reglas de producción tenemos:

$$
\begin{aligned}
R_1 &: Si\ GPA > 3.50\ Y\ GRE\ bueno\ Y\ referencias\ fuertes\ Entonces\ ACA \\
R_2 &: Si\ \ GPA > 3.50\ Y\ GRE\ bueno\ Y\ referencias\ débiles\ Entonces\ ACB \\
R_3 &: Si\ GPA > 3.50\ Y\ GRE\ malo\ Y\ referencias\ fuertes\ Entonces\ ACB \\
R_4 &: Si\ 3.5 \geq GPA > 3\ Y\ GRE\ bueno\ Y\ referencias\ fuertes\ Entonces\ ACB \\
R_5 &: Si\ 3.5 \geq GPA > 3\ Y\ GRE\ malo\ Y\ referencias\ fuertes\ Entonces\ ACC \\
R_6 &: Si\ 3.5 \geq GPA > 3\ Y\ GRE\ bueno\ Y\ referencias\ débiles\ Entonces\ ACC \\
R_7 &: Si\ GPA \leq 3.00\ Y\ GRE\ bueno\ Y\ referencias\ fuertes\ Entonces\ ACC \\
R_8 &: Si\ GPA \leq 3.00\ Y\ GRE\ malo\ Y\ referencias\ fuertes\ Entonces\ NAC \\
R_9 &: Si\ GPA \leq 3.00\ Y\ GRE\ bueno\ Y\ referencias\ débiles\ Entonces\ NAC \\
R_{10} &: Si\ GPA \leq 3.00\ Y\ GRE\ malo\ Y\ referencias\ débiles\ Entonces\ NAC
\end{aligned}
$$

Si estas a su vez las trasladamos a reglas de producción difusas, entonces estas quedarían de la siguiente forma:

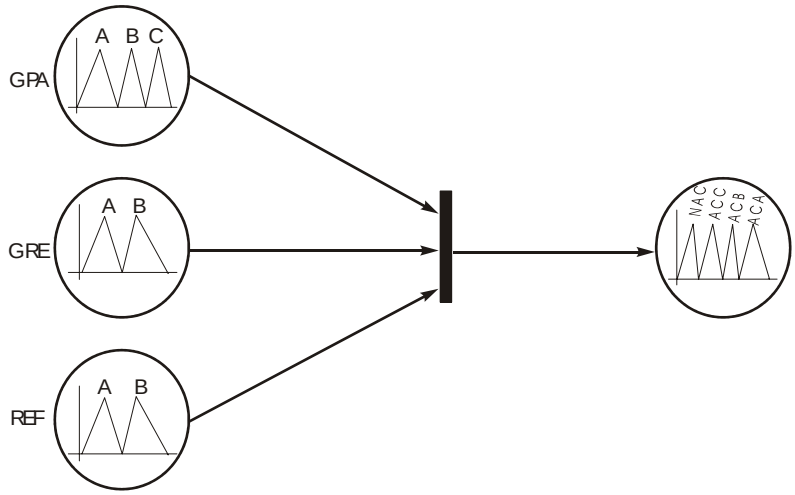


Figura 3.21: Red de Petri con funciones de membresía.

$$
\begin{aligned}
R_1 &: Si\ d_1\ Y\ d_2\ Y\ d_3\ Entonces\ d_8, (\mu_1{=}.8\ .9\ 1.0), \lambda_1 = [0.46\ 0.79\ 1.33] \\
R_2 &: Si\ \ d_1\ Y\ d_2\ Y\ d_5\ Entonces\ d_9, (\mu_2{=}.8\ .9\ 1.0), \lambda_1 = [0.38\ 0.67\ 1.17] \\
R_3 &: Si\ d_1\ Y\ d_3\ Y\ d_4\ Entonces\ d_9, (\mu_3{=}.8\ .9\ 1.0), \lambda_1 = [0.38\ 0.67\ 1.17] \\
R_4 &: Si\ d_6\ Y\ d_2\ Y\ d_3\ Entonces\ d_9, (\mu_4{=}.8\ .9\ 1.0), \lambda_1 = [0.38\ 0.67\ 1.17] \\
R_5 &: Si\ d_6\ Y\ d_2\ Y\ d_5\ Entonces\ d_{10}, (\mu_5{=}.8\ .9\ 1.0), \lambda_1 = [0.35\ 0.63\ 1.10] \\
R_6 &: Si\ d_6\ Y\ d_3\ Y\ d_4\ Entonces\ d_{10}, (\mu_6{=}.8\ .9\ 1.0), \lambda_1 = [0.35\ 0.63\ 1.10] \\
R_7 &: Si\ d_7\ Y\ d_2\ Y\ d_3\ Entonces\ d_{10}, (\mu_7{=}.8\ .9\ 1.0), \lambda_1 = [0.35\ 0.63\ 1.10] \\
R_8 &: Si\ d_7\ Y\ d_2\ Y\ d_5\ Entonces\ d_{11}, (\mu_8{=}.8\ .9\ 1.0), \lambda_1 = [0.20\ 0.44\ 0.88] \\
R_9 &: Si\ d_7\ Y\ d_3\ Y\ d_4\ Entonces\ d_{11}, (\mu_9{=}.8\ .9\ 1.0), \lambda_1 = [0.20\ 0.44\ 0.88] \\
R_{10} &: Si\ d_7\ Y\ d_4\ Y\ d_5\ Entonces\ d_{11}, (\mu_{10}{=}.8\ .9\ 1.0), \lambda_1 = [0.20\ 0.44\ 0.88]
\end{aligned}
$$

en donde el valor de λ_i es el valor mínimo con el cual se cumple la regla R_i y este lo calculamos con el valor de verdad más bajo de cada categoría.

Con las reglas de producción difusas construimos la red de Petri difusa de la figura 3.20. Claramente, el comportamiento de esta depende de sólo tres parámetros y las variables lingüísticas de cada uno de estos, de una forma más explícita esta podría ser ilustrada como en la figura 3.21.

Suponiendo que tenemos 4 grupos de alumnos cuyos valores de GPA, GRE y REF están contenidos en las siguientes funciones de membresía:

$$\begin{aligned} & & \alpha(d_1) =& (0.89 \;\; 0.95 \;\; 1.0) \\ Grupo\ 1: & & \alpha(d_2) =& (0.8 \;\; 0.9 \;\; 1.0) \\ & & \alpha(d_3) =& (0.85 \;\; 0.95 \;\; 1.0) \\ \\ & & \alpha(d_6) =& (0.80 \;\; 0.82 \;\; 0.88) \\ Grupo\ 2: & & \alpha(d_2) =& (0.81 \;\; 0.91 \;\; 1.0) \\ & & \alpha(d_3) =& (0.85 \;\; 0.95 \;\; 1.0) \\ \\ & & \alpha(d_1) =& (0.90 \;\; 0.95 \;\; 1.0) \\ Grupo\ 3: & & \alpha(d_2) =& (0.3 \;\; 0.5 \;\; 0.7) \\ & & \alpha(d_3) =& (0.86 \;\; 0.96 \;\; 1.0) \\ \\ & & \alpha(d_1) =& (0.4 \;\; 0.6 \;\; 0.7) \\ Grupo\ 4: & & \alpha(d_2) =& (0.72 \;\; 0.82 \;\; 1.0) \\ & & \alpha(d_3) =& (0.74 \;\; 0.85 \;\; 1.0) \end{aligned}$$

Teniendo en cuenta que los pesos asociados a las proposiciones $d_1, d_2, d_3, d_4, d_5, d_6, d_7$ son $w_6 = w_7 = w_1 = (0.8 \;\; 0.9 \;\; 1.0)$ y $w_4 = w_5 = w_2 = w_3 = (0.5 \;\; 0.6 \;\; 0.7)$. Determinar si estos grupo de alumnos son aceptados o no, y si son aceptados determinar si se les asigna un asistente o estarán condicionados.

El comportamiento dinámico de la RPDPF se explica como sigue:

- Del primer grupo la transición t_1 de la red de Petri difusa de la figura 3.20 se habilita. Al dispararse los valores de verdad de las proposiciones d_1, d_2 y d_3 son removidos y se adiciona una señal en el lugar p_8 (ACA) cuyo valor de verdad es (0.51 0.84 1.33).

- Del segundo grupo, la transición t_4 de la red de Petri difusa de la figura 3.20 se habilita. Al dispararse los valores de verdad de las proposiciones d_6, d_2 y d_3 son removidos y se adiciona una señal en el lugar p_9 (ACB) cuyo valor de verdad es (0.49 0.79 1.26).

- Del tercer grupo, la transición t_3 de la red de Petri difusa de la figura 3.20 se habilita. Al dispararse los valores de verdad de las proposiciones d_1, d_4 y d_3 son removidos y se adiciona una señal en el lugar p_9 (ACB) cuyo valor de verdad es (0.43 0.74 1.21).

- Del segundo grupo, la transición t_7 de la red de Petri difusa de la figura 3.20 se habilita. Al dispararse los valores de verdad de las proposiciones d_7, d_2 y d_3 son removidos y se adiciona una señal en el lugar p_{10} (ACC) cuyo valor de verdad es (0.35 0.66 1.16).

Valores Lingüísticos y Valores Numéricos Del ejemplo anterior podemos observar que existen 10 formas posibles en las cuales se pueden presentar las reglas de producción con valores lingüísticos y estas son:

Aceptado con asistente (ACA)	GPA $Bueno$ $a)$ GRE $Bueno$ REF $Fuertes$		
Aceptado (ACB)	GPA $Bueno$ $b)$ GRE $Bueno$ REF $débiles$	GPA $Bueno$ $c)$ GRE $Malas$ REF $Fuertes$	GPA $Regular$ $d)$ GRE $Bueno$ REF $Fuertes$
Aceptado con condiciones (ACC)	GPA $Regular$ $e)$ GRE $Bueno$ REF $Debiles$	GPA $Regular$ $f)$ GRE $Malas$ REF $Fuertes$	GPA $Malo$ $g)$ GRE $Bueno$ REF $Fuertes$
No aceptado (NAC)	GPA $Malo$ $h)$ GRE $Malas$ REF $Fuertes$	GPA $Malo$ $i)$ GRE $Bueno$ REF $débiles$	GPA $Malo$ $j)$ GRE $Malas$ REF $débiles$

convirtiendo cada valor lingüístico a un número difuso triangular tenemos:

Variable	Valor lingüístico	Valor numérico
GPA	Bueno	[0.875 1.0 1.0]
	Regular	[0.75 0.812 0.875]
	Malo	[0.1 0.425 0.75]
GRE	Buenas	[0.7 0.85 1.0]
	Malas	[0.3 0.4 0.5]
REF	Fuertes	[0.7 0.85 1.0]
	Débiles	[0.3 0.4 0.5]

Claramente a cada valor lingüístico le corresponde un valor numérico, trasladando los incisos a), b), ...,j) a redes de Petri difusas del tipo 2 y siendo los pesos w_1 =[0.8 0.85 0.9], w_2 =[0.4 0.5 0.6] y w_3 =[0.4 0.5 0.6] y el factor de certeza μ_i =[0.7 0.8 0.9] evaluamos el valor de verdad consecuente de cada una de ellas, de donde tenemos:

Aceptado con asistente (ACA)	*a*) [0.42 0.73 1.18]
Aceptado (ACB)	*b*) [0.36 0.63 1.01] *c*) [0.36 0.63 1.01] *d*) [0.38 0.66 1.11]
Aceptado con condiciones (ACC)	*e*) [0.33 0.56 0.94] *f*) [0.33 0.56 0.94] *g*) [0.21 0.52 1.05]
No aceptado (NAC)	*h*) [0.16 0.42 0.88] *i*) [0.16 0.42 0.88] *j*) [0.10 0.32 0.71]

Esto es, un alumno será aceptado con asistente sólo si al evaluar los valores numéricos de las variables lingüísticas GPA, GRE y REF mediante una red de Petri, el valor de la proposición consecuente es mayor que [0.42 0.73 1.18]. Un alumno será aceptado sin asistente si el valor de la proposición consecuente es mayor que el mínimo de las tres formas posibles,

esto es, si es mayor que mín[[0.36 0.63 1.01], [0.36 0.63 1.01], [0.38 0.66 1.11]] =[0.36 0.63 1.01]. Un alumno será aceptado con condiciones si el valor de la proposición consecuente es mayor que mín[[0.33 0.56 0.94], [0.33 0.56 0.94], [0.21 0.52 1.05]] =[0.21 0.52 0.94]. Cualquier alumno que presente un valor numérico menor que el anterior no será admitido.

3.4. Comparación Entre los Algoritmos Propuestos.

Mediante el algoritmo de razonamiento difuso con pesos (ARDP) presentado podemos determinar cuando existe una relación antecedente-consecuente de una proposición d_a a una proposición d_z, donde $d_a \neq d_z$. Más aún, si el grado de verdad de la proposición d_a es conocido y d_z es una proposición consecuente de d_a entonces el grado de verdad de la proposición d_z puede ser calculado, la principal diferencia entre este algoritmo y el de razonamiento hacia atrás (ARDHA) radica en el hecho de que en el ultimo podemos evaluar el valor de verdad de cualquier proposición especificada por el usuario, mientras que en el ARDP, si no existe una relación antecedente-consecuente de una proposición d_a conocida a una proposición d_z que se desea conocer, entonces, no se puede evaluar el valor de verdad de la proposición d_z, esto es, el valor de verdad de la proposición d_z sólo se puede evaluar si esta pertenece al conjunto de lugares alcanzables a partir de d_a. Además en el algoritmo de razonamiento difuso hacia atrás el valor de la proposición final se calcula como el valor de verdad final del conjunto de rutas y en el algoritmo de razonamiento difuso con pesos se calcula el valor de verdad para cada ruta.

Por el contrario, la principal característica de estos dos algoritmos es que no se asocian valores frontera a las transiciones, lo cual significa que cualquier lugar p_k tal que $p_k \in {}^{\cdot}t_j$ (p_k pertenece al conjunto de lugares de entrada de t_j) que se encuentre marcado con un valor de verdad y_k, por muy pequeño que este sea, la transición t_j estará habilitada y esta podrá ser disparada.

En el algoritmo de razonamiento difuso con pesos y valores frontera (ARDPF) presentado se asocian valores frontera a las transiciones, es decir, se condiciona una transición t_j habilitada por los lugares de entrada $p_{k_1}, p_{k_2}, ..., p_{k_n}$ a dispararse únicamente cuando $p_{k_i} \geq \lambda$

$(i = 1, 2, .., n)$ donde λ es el valor frontera. Luego, asociar valores frontera a las transiciones nos lleva a una forma más eficiente de razonamiento. En el ejemplo siguiente se aprecian las diferencias ya descritas.

Ejemplo 3.5 *Sean* $p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}$ *y* p_{15} *proposiciones, las cuales están relacionadas por las siguientes reglas de producción difusas:*

R_1 : Si p_1 *Entonces* p_7 (λ_1, μ_1);
R_2 : Si p_2 *Entonces* p_9 $(w_1 = w_2, \lambda_2, \mu_2)$;
R_3 : Si p_3 *Entonces* p_9 $(w_2 = w_1, \lambda_2, \mu_2)$;
R_4 : Si p_4 Y p_5 Y p_6 *Entonces* p_{10} $(w_3, w_4, w_5, \lambda_3, \mu_3)$;
R_5 : Si p_7 Y p_8 Y p_9 *Entonces* p_{12} $(w_6, w_7, w_8, \lambda_4, \mu_4)$;
R_6 : Si p_9 O p_{10} Entonces p_{13} $(w_9, w_{10}, \mu_5, \mu_6, \lambda_5, \lambda_6)$;
R_7 : Si p_{12} O p_{13} O p_{14} *Entonces* p_{15} $(w_{11}, w_{12}, w_{13}, \lambda_7, \lambda_8, \lambda_9, \mu_7, \mu_8, \mu_9)$.

Con estas reglas construimos la red de Petri difusa la cual se muestra en la figura 3.22. De donde podemos apreciar que tenemos nueve proposiciones de entrada ($p_1, p_2, p_3, p_4, p_5, p_6, p_8, p_{10}$ y p_{13}) y seis proposiciones consecuentes ($p_7, p_9, p_{11}, p_{12}, p_{14}$ y p_{15}), los pesos asociados a las proposiciones, así como los factores de certeza y valores frontera asociados a las transiciones son dados en la Tabla 3.3.

Usando los tres algoritmos de razonamiento difuso presentados, el conjunto de valores de salida (valores de verdad de las proposiciones consecuentes) puede ser calculado a partir de los valores de entrada (ver Tabla 3.4). En la Tabla 3.5 se muestran los resultados de los tres algoritmos.

Se puede observar en la tabla que algunos datos son 0, lo cual significa que los valores frontera asociados a las transiciones son mayores que los valores de verdad asociados a las proposiciones de entrada a la transición. Por ejemplo en el grupo 3 $\alpha(p_1)$ =[0.5 0.6 0.7], mientras que el valor frontera es [0.75 0.85 0.95] luego $\alpha(p_1) < \lambda_1$ por lo que la transición t_1 no puede ser disparada y por ello el valor de verdad de $\alpha(p_7) = 0$.

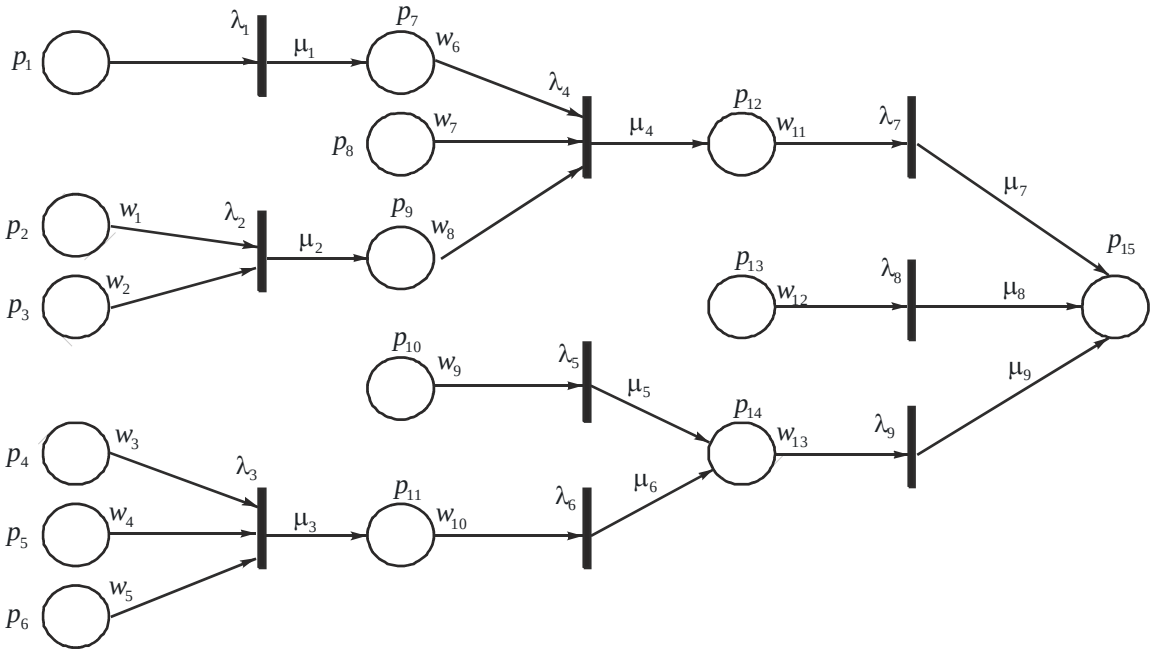


Figura 3.22: Red de Petri difusa del problema

Cuadro 3.3: Valores frontera, factores de certeza y pesos de RPD del ejemplo 3.4

μ_i	W_i	Th_i
μ_1 =(0.80 0.90 1.0)	w_1 =(0.70 0.80 0.90)	λ_1 =(0.75 0.85 0.95)
μ_2 =(0.80 0.90 1.0)	w_2 =(0.70 0.80 0.90)	λ_2 =(0.70 0.80 0.90)
μ_3 =(0.85 0.95 1.0)	w_3 =(0.65 0.75 0.85)	λ_3 =(0.60 0.70 0.80)
μ_4 =(0.80 0.90 1.0)	w_4 =(0.80 0.90 1.0)	λ_4 =(0.40 0.50 0.60)
μ_5 =(0.85 0.95 1.0)	w_5 =(0.70 0.80 0.90)	λ_5 =(0.35 0.45 0.55)
μ_6 =(0.85 0.95 1.0)	w_6 =(0.80 0.90 1.0)	λ_6 =(0.30 0.40 0.50)
μ_7 =(0.80 0.90 1.0)	w_7 =(0.85 0.95 1.0)	λ_7 =(0.40 0.50 0.60)
μ_8 =(0.70 0.80 0.90)	w_8 =(0.90 0.95 1.0)	λ_8 =(0.60 0.70 0.80)
μ_9 =(0.85 0.95 1.0)		λ_9 =(0.40 0.50 0.60)

Cuadro 3.4: Valores de verdad antecedentes de la RPD del ejemplo 3.4.

Grupo	p_1	p_2	p_3	p_4	p_5	p_6	p_8	p_{10}	p_{13}
1	(.85 .95 1.0)	(.80 .90 1.0)	(.85 .95 1)	(.7 .8 .9)	(.55 .65 .75)	(.4 .5 .6)	(.5 .6 .7)	(.85 .95 1)	(.8 .9 1)
2	(.50 .60 .70)	(.40 .50 .60)	(.4 .5 .6)	(.5 .6 .7)	(.55 .65 .75)	(.5 .6 .75)	(.3 .4 .5)	(.3 .4 .5)	(.5 .6 .7)
3	(.70 .80 .90)	(.60 .70 .80)	(.5 .6 .7)	(.5 .6 .7)	(.3 .4 .5)	(.5 .6 .7)	(.3 .4 .5)	(.3 .4 .5)	(.8 .9 1)
4	(.85 .95 1.0)	(.80 .90 1.0)	(.75 .85 .95)	(.75 .85 .95)	(.85 .95 1)	(.9 .95 1)	(.3 .4 .5)	(.3 .4 .5)	(.9 .95 1)
5	(.75 .85 .95)	(.85 .95 1.0)	(.6 .7 .8)	(.55 .65 .75)	(.90 .95 1)	(.8 .9 1)	(.95 1 1)	(.8 .9 1)	(.8 .9 1)

Cuadro 3.5: Valores de verdad consecuentes de la RPD del ejemplo 3.4.

Grupo		p_7	p_9	p_{11}	p_{12}	p_{14}	p_{15}
1	RPDHA	(.68 .85 1.0)	(.63 .83 1.0)	(.36 .61 .95)	(.41 .68 1.06)	(.51 .78 1.15)	(.45 .78 1.32)
	RPDP	(.68 .85 1.0)	(.63 .83 1.0)	(.36 .61 .95)	(.41 .68 1.06)	(.51 .78 1.15)	(.56 .72 .9)
	RPDPVF	(.68 .85 1.0)	(.66 .92 1.2)	(.14 .23 .35)	(.41 .71 1.15)	(.36 .47 .58)	(.31 .51 .87)
2	RPDHA	(.4 .54 .7)	(.32 .45 .6)	(.34 .58 .94)	(.23 .41 .71)	(.27 .49 .84)	(.26 .49 .92)
	RPDP	(.4 .54 .7)	(.32 .45 .6)	(.34 .58 .94)	(.23 .41 .71)	(.27 .49 .84)	(.35 .48 .63)
	RPDPVF	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)
3	RPDHA	(.56 .72 .9)	(.44 .58 .75)	(.28 .5 .8)	(.29 .5 .84)	(.24 .45 .76)	(.34 .6 1.06)
	RPDP	(.56 .72 .9)	(.44 .58 .75)	(.28 .5 .8)	(.29 .5 .84)	(.24 .45 .76)	(.56 .72 .9)
	RPDPVF	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(0 0 0)	(.19 .27 .38)
4	RPDHA	(.68 .85 1)	(.62 .78 .97)	(.55 .87 1.2)	(.36 .61 .97)	(.36 .63 1)	(.42 .72 1.2)
	RPDP	(.68 .85 1)	(.62 .78 .97)	(.55 .87 1.2)	(.36 .61 .97)	(.36 .63 1)	(.63 .76 .9)
	RPDPVF	(.68 .85 1)	(.62 .78 .97)	(.55 .87 1.2)	(.29 .51 .87)	(.23 .43 .74)	(.21 .28 .38)
5	RPDHA	(.6 .76 .95)	(.58 .74 .9)	(.5 .8 1.18)	(.48 .75 1.1)	(.55 .85 1.28)	(.48 .83 1.4)
	RPDP	(.6 .76 .95)	(.58 .74 .9)	(.5 .8 1.18)	(.48 .75 1.1)	(.55 .85 1.28)	(.47 .8 1.28)
	RPDPVF	(.6 .76 .95)	(.34 .47 .62)	(.39 .6 .88)	(.34 .52 .76)	(.5 .75 1.1)	(.34 .54 .85)

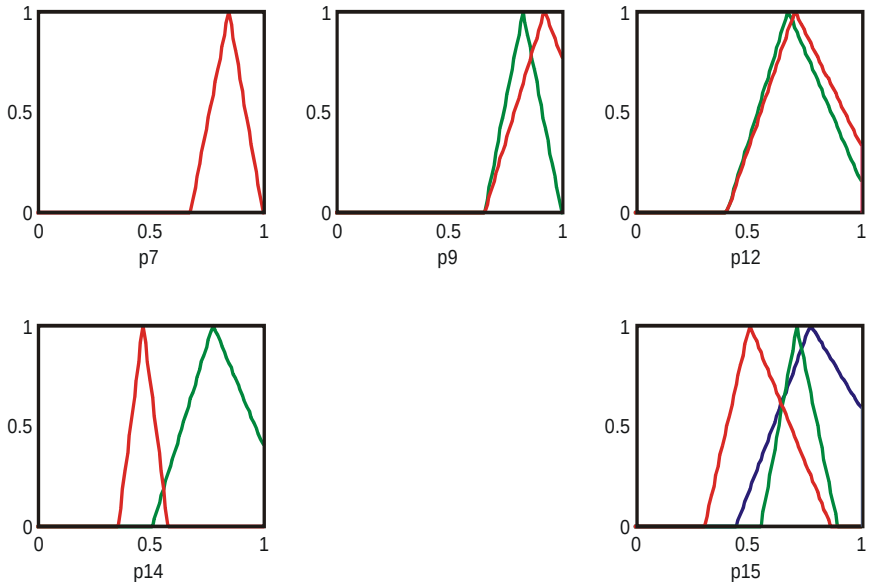


Figura 3.23: Valores de Verdad de las proposiciones consecuentes del grupo 1.

3.5. Conclusión

En este capitulo se presentaron tres algoritmos de razonamiento difuso. El *algoritmo de razonamiento difuso hacia atrás* propuesto por [5], *algoritmo de razonamiento difuso con pesos* [6] y *algoritmo de razonamiento difuso con pesos y valores frontera* [42] la diferencia de los originalmente propuestos radica en que los desarrollados en este capitulo los valores de verdad, factores de certeza y valores frontera son números difusos triangulares. Además, se asocian pesos a las proposiciones los cuales también son números difusos triangulares.

Capítulo 4

Aprendizaje con Redes de Petri Difusas Adaptativas

La teoría de RPD provee herramientas efectivas para manipular información imprecisa o vaga. Algunos modelos de redes de Petri difusas han sido propuestos para soportar razonamiento difuso y tomar decisiones [7], [23], [47] y [48]. Otras aplicaciones de las RPD en sistemas basados en conocimiento son evaluación de inconsistencias [38], aprendizaje de conocimiento [41], [42], etc.

Las redes de Petri difusas pueden representar reglas de producción difusas con pesos perfectamente [6][48]. Sin embargo, aunque estos modelos son muy flexibles los pesos en estos no pueden ajustarse de acuerdo a la actualización de conocimiento, en otras palabras, estas no tienen la habilidad de aprender, algunas redes de Petri difusas con la habilidad de aprender han sido propuestas en la literatura. En [23] Carl G. Looney propone un algoritmo para ajustar los valores frontera de una red de Petri difusa, sin embargo, el ajuste de los pesos se realiza mediante un test y no se asegura la convergencia del algoritmo. En [30] Pedricz y Gomide proponen una red de Petri difusa generalizada (GFPN) en el cual la dinámica de los lugares de entrada y salida de una GFPN puede ser transformado en una red neuronal con neuronas lógicas Or/And. En [41] X. Li y F. Lara definen las redes de Petri difusas adaptativas (AFPN), estas poseen la habilidad de aprender de una forma parecida

a las redes neuronales. En [42] X. Li, W. Yu y F. Lara proponen reglas de razonamiento más generalizadas para las AFPN. Además, se desarrolla el algoritmo *backpropagation (BP)* para aprendizaje de conocimiento bajo condiciones generalizadas.

En este capitulo se analiza el aprendizaje con redes de Petri difusas con pesos adaptativas (RPDA) utilizando números difusos triangulares. Las definiciones básicas de conjuntos difusos y operaciones con números difusos triangulares descritas en el capitulo 2 y definidas en [20],[43] y [16] son utilizadas en nuestra RPDA. El proceso de aprendizaje de pesos de una red se desarrolla mediante dos algoritmos de aprendizaje. El algoritmo de aprendizaje de Widrow-Hoff para redes de una sola capa [40], [13] y el algoritmo *backpropagation* para redes multicapa [36], [13].

4.1. Algoritmos de Aprendizaje en Redes Neuronales

4.1.1. Algoritmo de Widrow-Hoff.

La regla de aprendizaje LMS minimiza el error cuadrático medio $e^2(k)$ generado por la salida deseada y la obtenida , el cual es definido por

$$(t(k) - a(k))^2 = e^2(k) \tag{4.1}$$

donde en cada iteración tenemos un gradiente estimado de la forma:

$$\nabla e^2(k) \tag{4.2}$$

los primeros R elementos de $\nabla e^2(k)$ son derivadas con respecto a los pesos de la red, mientras que el elemento $(R+1)$ es la derivada con respecto a b.

$$\left[\nabla e^2(k)\right]_j = \frac{\partial e^2(k)}{\partial w_j} = 2e(k)\frac{\partial e(k)}{\partial w_j}, j = 1, 2, ..., R, \tag{4.3}$$

y

$$\left[\nabla e^2(k)\right]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b} \tag{4.4}$$

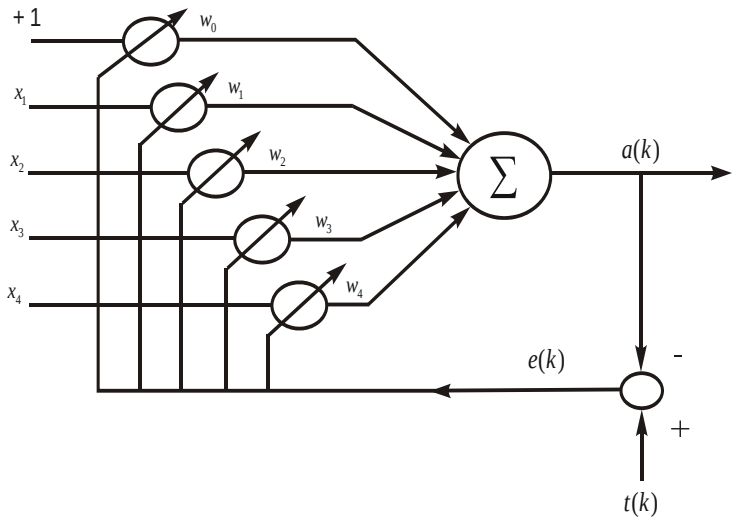


Figura 4.1: Aprendizaje mediante algoritmo de Widrow-Hoff

Evaluando la derivada parcial de $e(k)$ con respecto a los pesos, tenemos:

$$\begin{aligned} \frac{\partial e(k)}{\partial w_j} &= \frac{\partial [t(k)-a(k)]}{\partial w_j} = \frac{\partial}{\partial w_j}\left[t(k) - w^T p(k) + b\right] \\ \frac{\partial e(k)}{\partial w_j} &= \frac{\partial}{\partial w_j}\left[t(k) - \left(\sum_{i=1}^{R} w_i p_i(k) + b\right)\right] \end{aligned} \tag{4.5}$$

donde $p_i(k)$ es definido como el i-ésimo elemento del vector de entrada en la k-ésima iteración. Simplificando tenemos:

$$\frac{\partial e(k)}{\partial w_j} = -p_j(k) \tag{4.6}$$

de una forma similar, obtenemos el elemento final del gradiente

$$\frac{\partial e(k)}{\partial b} = -1 \tag{4.7}$$

ya que $p_j(k)$ y 1 son elementos del vector de entrada X, por lo tanto el gradiente del error cuadrático en la iteración k puede ser escrito como

$$\nabla e^2(k) = -2e(k)X(k) \tag{4.8}$$

Teniendo en cuenta que el algoritmo de gradiente descendiente con constante de aprendizaje es:

$$x_{k+1} = x_k - \alpha \nabla e^2(k) \mid_{x=x_k} \tag{4.9}$$

si substituimos la ecuación 4.8 tenemos que:

$$x_{k+1} = x_k + 2\alpha e(k)X(k) \tag{4.10}$$

luego

$$w(k+1) = w(k) + 2\alpha e(k)p(k) \tag{4.11}$$

y

$$b(k+1) = b(k) + 2\alpha e(k) \tag{4.12}$$

4.1.2. Algoritmo *Backpropagation* (*BP*)

El algoritmo *Backpropagation* se utiliza para ajustar los pesos y sesgos de una red con el fin de minimizar la suma del cuadrado de los errores de la red.

El algoritmo BP es un método iterativo de optimización de gradiente descendente, cuyos detalles se presentan a continuación.

Para una neurona j en una capa oculta o en la salida, la señal de salida es:

$$o_j = f(\sum_{i=1}^{n} w_{ij} o_i - b_j) \tag{4.13}$$

donde:

f es la función de activación de la neurona

w_{ij} son los pesos de las conexiones entre la neurona considerada j, y la neurona i, perteneciente a la capa precedente

o_i es la salida de la neurona i de la capa precedente

b_j es el sesgo de la neurona j

Además, se define:

$$net_j = \sum_{i=1}^{n} w_{ij} o_i - b_j \tag{4.14}$$

la salida de la neurona j, entonces es dada por:

$$o_j = f\left(net_j\right) \tag{4.15}$$

para el aprendizaje, el valor de $-bj$ se considera como un peso correspondiente a la conexión de la neurona j con una supuesta neurona de la capa precedente cuya salida es constante e igual a uno.

El algoritmo *Backpropagation* permite determinar los valores de los pesos para los cuales la función de error es mínima. Esto no siempre se logra, ya que muchas veces el algoritmo converge a mínimos locales y no a mínimos globales, o simplemente no converge.

Se considera una red con M neuronas en la capa de salida y suponiendo que se dispone de un conjunto de aprendizaje con P patrones, uno de los cuales, denominado p, tiene salidas dadas por:

$$t_p = [t_{p1}, t_{p2}, ..., t_{pM}]$$

El error cuadrático tiene, para este patrón, la siguiente expresión

$$E_p = \frac{1}{2} \sum_{i=1}^{n} (t_{pi} - o_{pi})^2 \tag{4.16}$$

Los valores t_{pi} representan las salidas deseadas ante las entradas correspondientes al patrón p. Cuando dicho patrón es presentado a la red, los pesos se modifican según la regla iterativa derivada del método de optimización según el gradiente, con lo cual el peso w_{ij} según la ecuación es:

$$\Delta w_{ij}(h) = \eta \left(-\frac{\partial E_p}{\partial w_{ij}} \right) = \eta \left(-\frac{\partial E_p}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} \right) \tag{4.17}$$

donde η es la tasa de aprendizaje (constante de proporcionalidad) $(0 < \eta < 1)$. En general, los pesos se inicializan entre cero y uno aleatoriamenté.

Se define el parámetro δ_j como:

$$\delta_j = -\frac{\partial E_p}{\partial w_{ij}} = -\frac{\partial E_p}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}} \tag{4.18}$$

Para calcular las derivadas es necesario tener en cuenta que la función de activación debe ser continua suponiendo que nuestra función de activación escogida es una sigmoide logarítmica, la derivada es:

$$\frac{df(x)}{dx} = \frac{d}{dx} \left(\frac{1}{1+e^{-x}} \right) = \frac{1}{1+e^{-x}} \left(1 - \frac{1}{1+e^{-x}} \right) = f(x)(1-f(x)) \tag{4.19}$$

Para una neurona j en la capa de salida se tiene entonces,

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \tag{4.20}$$

Para una neurona en la capa oculta o en la capa de entrada, se tiene:

$$\delta_j = o_j (1 - o_j) \sum_{k=1}^{m} \delta_k w_{jk} \tag{4.21}$$

donde el contador k cubre las neuronas de la capa posterior a la j.

Entonces la corrección de los pesos se comienza por la capa de salida y se propaga hacia atrás hasta llegar a la capa de entrada.

Con esto el término 4.17 se puede expresar como:

$$\Delta w_{ij} = \eta \delta_j o_i \tag{4.22}$$

Considerando los P patrones de que se dispone y con los cuales se realizara el aprendizaje, la expresión para el error total, o error de ajuste, es la siguiente:

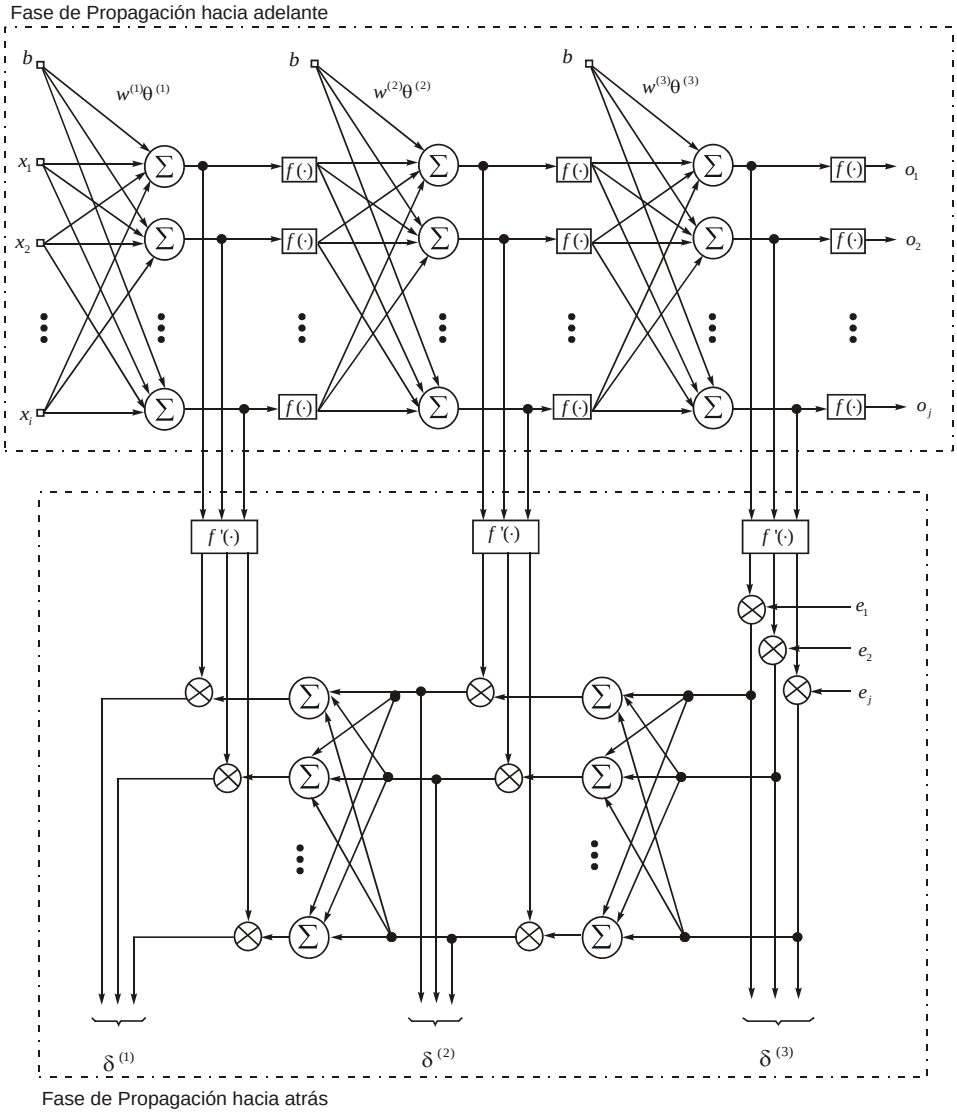


Figura 4.2: Fase de Propagación hacia adelante y hacia atrás del algoritmo BP.

$$E = \sum_{p=1}^{m} E_p = \sum_{p=1}^{m} \left(\frac{1}{2} \sum_{i=1}^{n} (t_{pi} - o_{pi})^2 \right) \tag{4.23}$$

En general, el aprendizaje se considera acabado cuando el valor de E es menor o igual que un límite preestablecido.

En resumen, la actualización de pesos mediante al algoritmo de retropropagación es:

1. *Inicialización.* Se comienza inicializando con una configuración razonable de una red neuronal, fijando todos los pesos y umbrales a números pequeños aleatorios que estén uniformemente distribuidos.

2. *Presentación de ejemplos de entrenamiento.* Presentar las red con una época de ejemplos de entrenamiento. Para cada ejemplo en el conjunto, desarrollar la secuencia de computo hacia adelante y hacia atrás descritos en los puntos 3 y 4.

3. *Computo hacia adelante.* Sea $(x(k), d(k))$ una muestra de aprendizaje en la época, con el vector de entrada $x(k)$ aplicado a la capa de entrada de nodos sensores y el vector de respuesta deseado $d(k)$ presentado a la capa de salida de nodos computacionales. Computar los campos locales inducidos y las señales de funciones de la red procediendo hacia delante en la red, capa por capa. El campo local inducido $v_j^{(l)}(k)$ para la neurona j en la capa l es

$$v_j^{(l)}(k) = \sum_{i=0}^{m_o} w_{ji}^{(l)}(k) y_i^{(l-1)}(k) \tag{4.24}$$

Donde $y_i^{(l-1)}(k)$ es la señal de salida de la neurona i en la capa anterior $l-1$ en la iteración k y $w_{ji}^{(l)}(k)$es el peso sináptico de la neurona j en la capa l que es alimentada de la neurona i en la capa $l-1$. Para $i=0$ tenemos $y_0^{(l-1)}(k) = +1$ y $w_{j0}^{(l)}(k) = b_j^{(l)}(k)$ es aplicado a la neurona j en la capa l. Asumiendo el uso de una función sigmoide, la señal de salida de la neurona j en la capa l es:

$$y_j^{(l)} = \varphi_j(v_j(k))$$

si la neurona j se encuentra en la primer capa oculta, usamos:

$$y_j^{(0)} = x_j(k)$$

donde $x_j(k)$ es el $j - ésimo$ elemento del vector de entrada $x(k)$. Si la neurona j está en la capa de salida, usamos

$$y_j^{(L)} = o_j(k)$$

computamos la señal de error como

$$e_j(k) = d_j(k) - o_j(k)$$

donde $d_j(k)$ es el $j - ésimo$ elemento del vector de respuesta deseado d_k.

4. *Cómputo hacia atrás.* Computar los δ_s (gradientes locales) de la red definidos como

$$\begin{aligned} \delta_j^{(l)}(k) &= e_j^{(L)}(k)\varphi_j'\left(v_j^{(l)}(k)\right) && \text{para la neurona } j \text{ en la capa de salida } L \\ \delta_j^{(l)}(k) &= \varphi_j'\left(v_j^{(l)}(k)\right)\sum_m \delta_m^{(l+1)}(k)w_{kj}^{(l+1)}(k) && \text{para la neurona } j \text{ en la capa oculta } l \end{aligned}$$

donde el apóstrofe en φ_j' denota diferenciación con respecto al argumento. Ajustar los pesos sinápticos de la red en la capa l de acuerdo a la regla delta generalizada:

$$w_{ji}^{(l)}(k+1) = w_{ji}^{(l)}(k) + \eta\delta_j^{(l)}(k)y_i^{(l-1)}(k)$$

donde η es el parámetro de la velocidad de aprendizaje.

5. *Iteración.* Iterar los cómputos hacia delante y hacia atrás de los puntos 3 y 4 presentando nuevas épocas de ejemplos de entrenamiento a la red hasta que se consiga el criterio de paro.

4.2. Redes de Petri Difusas Adaptativas (RPDA).

Las redes de Petri difusas con habilidad para aprender fueron propuestas en [42], las cuales son llamadas redes de Petri difusas adaptativas (RPDA), estas se definen a continuación:

Definición 4.1 *Una red de Petri difusa con pesos adaptativa se define como una tupla* $RPDA = (P, T, D, I, O, \alpha, \beta, Th, W)$, *donde:*

P *y* T *son conjuntos no vacíos, finitos y disjuntos de lugares y transiciones*

$D = \{d_1, d_2, ..., d_n\}$*Conjunto finito de proposiciones;*

$I =$*Función de entrada,* $I : T \rightarrow P^{\infty}$, *a cada transición le asocia bolsas de lugares;*

$O =$*Función de salida, a cada transición le asocia bolsas de lugares;*

$\alpha =$ *Función de asociación,* $\alpha : P \rightarrow [0, 1]$ *Mapea lugares a números reales entre 0 y 1;*

$\beta =$ *Función de asociación,* $\beta : P \rightarrow D$, *a cada lugar le asocia una proposición D.*

$Th : T \rightarrow [0, 1]$ *es una función que asigna a cada transición un valor frontera* $\lambda_i, \lambda_i \in [0, 1]$

$W = W_I \cup W_O$. $W_I : I \rightarrow [0, 1]$ *y* $W_O : O \rightarrow [0, 1]$, *son conjuntos de pesos de entrada y pesos de salida (factores de certeza) los cuales asignan pesos a todos los arcos de la red.*

Si $p_j \in I(t_i)$, entonces existe un arco dirigido que va del lugar p_j a la transición t_i. Si $p_k \in O(t_i)$,entonces existe un arco dirigido que va de la transición t_i al lugar p_k. Si $W_O = \mu_i$, donde μ_i es un número difuso triangular, entonces la transición t_i se dice que esta asociada con un número difuso μ_i. Si $W_I = w_i$, donde w_i es un número difuso triangular. Entonces el lugar p_i se dice que esta asociado con un número difuso w_i. Si $\beta(p_i) = d_i$ donde $d_i \in D$, entonces el lugar p_i se dice que esta asociado con una proposición d_i.

Definición 4.2 *Si una RPDA contiene algunas señales en algunos lugares, entonces esta es llamada RPDA marcada, donde una señal en el lugar* p_i *esta representada por un punto etiquetado* $\overset{\alpha(p_i)}{\cdot}$ *y el valor de la señal en el lugar* $p_i, p_i \in P$ *esta denotado por* $\alpha(p_i)$, *donde* $\alpha(p_i)$ *es un número difuso triangular. Si* $\alpha(p_i) = y_i$ *y* $\beta(p_i) = d_i$. *entonces el grado de verdad de la proposición* d_i *es* y_i.

Los cuatro tipos de reglas de producción definidas en Ecs.(3.2), (3.3), (3.4) y (3.5) pueden ser representadas por redes de Petri difusas adaptativas como lo muestra la figura 4.3.

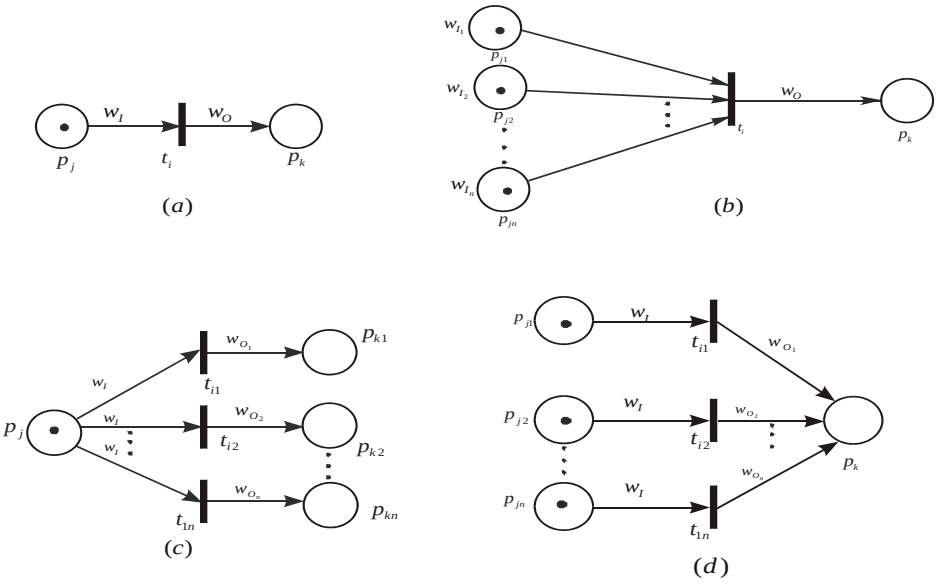


Figura 4.3: Modelado de reglas de producción difusas con redes de Petri difusas adaptativas.

Dividimos el conjunto de lugares P en tres partes $P = P_I \cup P_{Int} \cup P_O$, donde P es un conjunto de lugares de una RPDA; $P_I = \{p \in P \mid {}^\bullet p = 0\}$, si $p \in P_I$ este lugar es llamado lugar de entrada; $P_{Int} = \{p \in P \mid {}^\bullet p \neq 0 \;\; y \;\; p^\bullet \neq 0\}$, si $p \in P_{Int}$ este lugar es llamado lugar interior; $P_O = \{p^\bullet = 0\}$, si $p \in P_O$ este lugar es llamado lugar de salida.

4.3. Aprendizaje para RPDA con el Algoritmo Widrow-Hoff

4.3.1. Aprendizaje para RPDA sin Valores Frontera

A. Aproximación del disparo.

El proceso de razonamiento de redes de una sola capa es analizado en esta subsección mediante las siguientes definiciones:

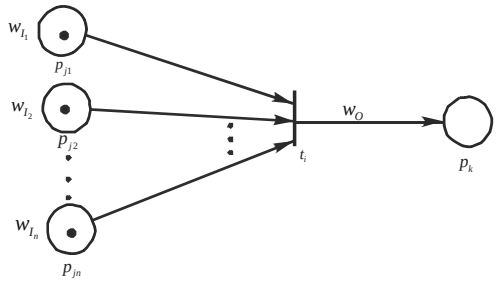


Figura 4.4: Red de Petri difusa Adaptativa sin valores frontera de un caso 2.

Sea $I(t) = \{p_{I1}, p_{I2}, ..., p_{In}\}$ lugares de entrada y $W_I = [w_{I1}, w_{I2}, ..., w_{In}], [\lambda_1, \lambda_2, ..., \lambda_n]$ sus correspondientes pesos y valores frontera y sea $O(t) = \{p_{O1}, p_{O2}, ..., p_{Om}\}$ el conjunto de lugares de salida.

Definición 4.3 *Cuando una transición t es habilitada, esta produce un nuevo valor de verdad $\alpha(p_j)(t)$.*

$$\alpha(p_j)(t) = \begin{cases} a_{\alpha(p_j)} = \dfrac{\sum_{i=1}^{n}(a_{\alpha i} \cdot a_{wi})}{\sum_{i=1}^{n} c_{wi}} \\ b_{\alpha(p_j)} = \dfrac{\sum_{i=1}^{n}(b_{\alpha i} \cdot b_{wi})}{\sum_{i=1}^{n} b_{wi}} \\ c_{\alpha(p_j)} = \dfrac{\sum_{i=1}^{n}(c_{\alpha i} \cdot c_{wi})}{\sum_{i=1}^{n} a_{wi}} \end{cases} \tag{4.25}$$

Podemos usar una función continua $\alpha(p_j)(t)(x)$ para aproximar $\alpha(p_j)(t)$:

$$\alpha(p_j)(t)(x) = x_\varphi . F_\varphi(x); \varphi = a, b, c \tag{4.26}$$

donde:

$$x_\varphi = \sum_{i=1}^{n}(\varphi_{\alpha i} \cdot \varphi_{wi})$$

donde F(x) es una función sigmoide que aproxima la frontera de t.

$$F_{\varphi}(x) = \frac{\frac{a_{\mu_i}}{\Sigma_i\ c_{wi}}}{1 + e^{-\beta(x_a)}}; \varphi = a$$
$$F_{\varphi}(x) = \frac{\frac{b_{\mu_i}}{\Sigma_i\ b_{wi}}}{1 + e^{-\beta(x_b)}}; \varphi = b$$
$$F_{\varphi}(x) = \frac{\frac{c_{\mu_i}}{\Sigma_i\ a_{wi}}}{1 + e^{-\beta(x_c)}}; \varphi = c$$

claramente, si β es lo suficientemente grande, $e^{-b(x_{\varphi})} \approx 0$. entonces $F(x) \approx \frac{\varphi_{\mu_j}}{\Sigma_i\ b_{wi}}$.

Definición 4.4 *Cuando una transición es habilitada, entonces es disparada tomando lugar la transmisión de señales de la siguiente manera:*

1. *Si un lugar p_{ok} tiene una sola transición de entrada t, una señal con valor*

$$[a_{w_{ok}} a_{\alpha(t)}, b_{w_{ok}} b_{\alpha(t)}, c_{w_{ok}} c_{\alpha(t)}]$$

es depositada en cada lugar de salida p_{ok}, $k = 1, 2, ..., m$ y todas las señales en $p_{ij}, j = 1, 2, ..., n$ son removidas.

2. *Si un lugar (p_{Ok}) tiene más de una transición de entrada y más de una de estas se dispara, es decir, existe más de una ruta activa al mismo tiempo, entonces un nuevo valor de verdad $\alpha(p_{Ok})$ es calculado como el centro de gravedad de las transiciones disparadas:*

$$\alpha(p_{Ok}) = (a_{\alpha p_{Ok}}, b_{\alpha p_{Ok}}, c_{\alpha p_{Ok}})$$

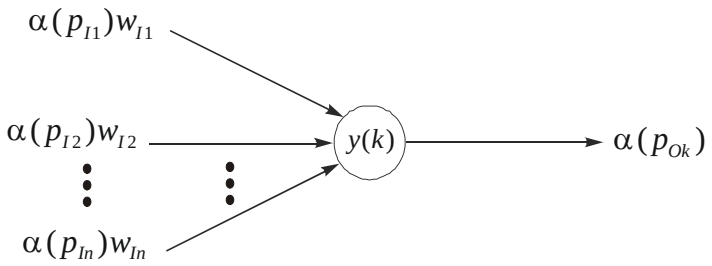


Figura 4.5: Red neuronal de una red de Petri difusa adaptativa.

- *donde:*

$$a_{\alpha p_{Ok}} = \frac{\sum_{i=1}^{n}(a_{\alpha i} \cdot a_{\mu i})}{\sum_{i=1}^{n} c_{\mu i}}$$

$$b_{\alpha p_{Ok}} = \frac{\sum_{i=1}^{n}(b_{\alpha i} \cdot b_{\mu i})}{\sum_{i=1}^{n} b_{\mu i}}$$

$$c_{\alpha p_{Ok}} = \frac{\sum_{i=1}^{n}(c_{\alpha i} \cdot c_{\mu i})}{\sum_{i=1}^{n} a_{\mu i}}$$

cuando t_j *dispara,* $t_j \in {}^{\cdot}p_{ok}$

B. Aprendizaje

Tomando una regla de producción difusa del Tipo 2 como una ilustración para mostrar el procedimiento de aprendizaje de conocimiento usando una RPDA. El Tipo 2 puede ser trasladado como una estructura de una red neuronal como se muestra en la figura 4.5. Donde $y(k)$ se define como sigue:

$$y_{\varphi}(k) = f_{\varphi}(x) \cdot x_{\varphi}(k) \tag{4.27}$$

donde:

$$x_\varphi(k) = W\varphi^T(k)P_\varphi(k) = \sum_{i=1}^{n}(\varphi_{\alpha i}(k)\cdot\varphi_{wi}(k)), \quad \begin{array}{l} f_\varphi(k)(x) = \dfrac{\dfrac{a_{\mu_i}}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x_a)}}; \varphi = a \\ f_\varphi(k)(x) = \dfrac{\dfrac{b_{\mu_i}}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x_b)}}; \varphi = b \\ f_\varphi(k)(x) = \dfrac{\dfrac{c_{\mu_i}}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x_c)}}; \varphi = c \end{array} \tag{4.28}$$

β constante que ajusta la pendiente de $f_\varphi(x)$
$W_\varphi(k)$ vector de pesos $W_\varphi(k) = [\varphi_{w_1}(k), \varphi_{w_2}(k), .., \varphi_{w_n}(k)]^T$
$P_\varphi(k)$ vector de valores de verdad $P_\varphi = [\varphi_{\alpha_1}(k), \varphi_{\alpha_2}(k), .., \varphi_{\alpha_n}(k)]^T$

Por lo tanto, la ley de aprendizaje de Widrow-Hoff puede ser aplicada como:

$$W(k+1) = W(k) + 2\delta e(k)P(k), \qquad e(k) = t(k) - y(k) \tag{4.29}$$

donde $t(k)$ es la salida deseada y el vector de pesos $W(k)$ es calculado recursivamente.

4.3.2. Aprendizaje para RPDA con Valores Frontera

A. Aproximación del disparo.

El proceso de razonamiento es el mismo que utilizamos en el algoritmo *backpropagation* desarrollado en las ecuaciones (4.32), (4.33), (4.34), (4.35), (4.36) y (4.37).

B. Aprendizaje

Una vez más, trasladando una regla de producción difusa del Tipo 2 a una estructura de una red neuronal como se muestra en la figura 4.5. Donde $y(k)$ en este caso se define como sigue:

$$y_\varphi(k) = f_\varphi(x).x_\varphi(k) \tag{4.30}$$

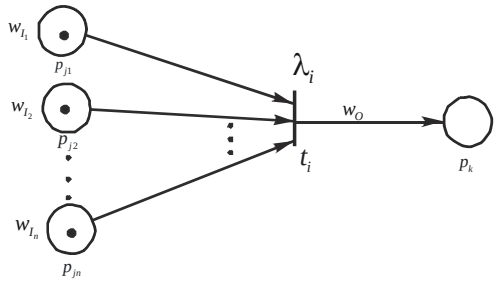


Figura 4.6: Red de Petri difusa con pesos adaptativa con valor frontera λ_i asociado a la transición t_i de un Caso 2.

donde:

$$x_\varphi(k) = W\varphi^T(k)P_\varphi(k) = \sum_{i=1}^{n}(\varphi_{\alpha i}(k)\cdot\varphi_{wi}(k)),$$

$$f_\varphi(k)(x) = \frac{\frac{a_{\mu_i}}{\Sigma_i\ c_{wi}}}{1+e^{-\beta\left(x_\varphi - a_{Th(t)}\right)}}; \varphi = a$$

$$f_\varphi(k)(x) = \frac{\frac{b_{\mu_i}}{\Sigma_i\ b_{wi}}}{1+e^{-\beta\left(x_\varphi - b_{Th(t)}\right)}}; \varphi = b$$

$$f_\varphi(k)(x) = \frac{\frac{c_{\mu_i}}{\Sigma_i\ a_{wi}}}{1+e^{-\beta\left(x_\varphi - c_{Th(t)}\right)}}; \varphi = c$$

β constante que ajusta la pendiente de $f_\varphi(x)$
$W_\varphi(k)$ vector de pesos $W_\varphi(k) = [\varphi_{w_1}(k), \varphi_{w_2}(k), .., \varphi_{w_n}(k)]^T$
$P_\varphi(k)$ vector de valores de verdad $P_\varphi = [\varphi_{\alpha_1}(k), \varphi_{\alpha_2}(k), .., \varphi_{\alpha_n}(k)]^T$

(4.31)

Por lo tanto, la ley de aprendizaje de Widrow-Hoff puede ser aplicada como en la Ec. (4.29).

4.4. Aprendizaje para RPDA con el Algoritmo *BP*

A. Aproximación del disparo El proceso de razonamiento en una red neuronal multicapa es analizado en esta subsección. A continuación de introduce el proceso de razonamiento mediante las siguientes definiciones:

Definición 4.5 *$\forall t \in T$, t es habilitada si $\forall p_i \in I(t), \alpha(p_i) > 0, i = 1, 2, ..., n$. Donde $\alpha(p_i)$ es un número difuso triangular.*

Definición 4.6 *Sean A y B números difusos triangulares parametrizados por (a_1, a_2, a_3) y (b_1, b_2, b_3) respectivamente, se dice que $A > B$ si y solamente si $a_1 > b_1, a_2 > b_2$ y $a_3 > b_3$. De otra manera se dice que A y B no son comparables.*

Definición 4.7 *Cuando una transición t es habilitada, esta produce un nuevo valor de verdad $\alpha(p_j)(t)$.*

$$\alpha(p_j)(t) = \begin{cases} a_{\alpha(p_j)} = \dfrac{\sum_{i=1}^{n}(a_{\alpha i} \cdot a_{wi})}{\sum_{i=1}^{n} c_{wi}}, \sum_{i=1}^{n}(a_{\alpha i} \cdot a_{wi}) > a_{Th(t)} \quad y \\ b_{\alpha(p_j)} = \dfrac{\sum_{i=1}^{n}(b_{\alpha i} \cdot b_{wi})}{\sum_{i=1}^{n} b_{wi}}, \sum_{i=1}^{n}(b_{\alpha i} \cdot b_{wi}) > b_{Th(t)} \quad y \\ c_{\alpha(p_j)} = \dfrac{\sum_{i=1}^{n}(c_{\alpha i} \cdot c_{wi})}{\sum_{i=1}^{n} a_{wi}}, \sum_{i=1}^{n}(c_{\alpha i} \cdot c_{wi}) > c_{Th(t)} \\ 0 \quad ,\sum_{i=1}^{n}(a_{\alpha i} \cdot a_{wi}) < a_{Th(t)} \quad o \\ 0 \quad ,\sum_{i=1}^{n}(b_{\alpha i} \cdot b_{wi}) < b_{Th(t)} \quad o \\ 0 \quad ,\sum_{i=1}^{n}(c_{\alpha i} \cdot c_{wi}) < c_{Th(t)} \end{cases} \tag{4.32}$$

Podemos usar una función continua $\alpha(p_j)(t)(x)$ para aproximar $\alpha(p_j)(t)$:

$$\alpha(p_j)(t)(x) = x_\varphi . F_\varphi(x); \varphi = a, b, c \tag{4.33}$$

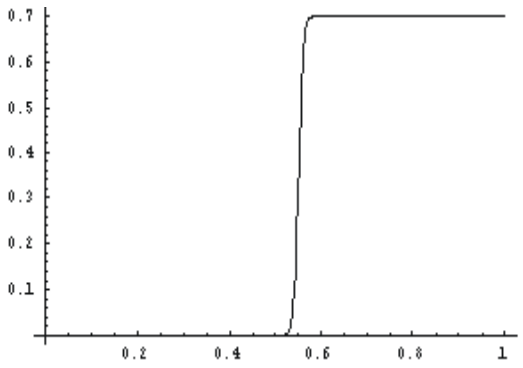


Figura 4.7: Funcion sigmoide.

donde:

$$x_\varphi = \sum_{i=1}^{n} (\varphi_{\alpha i} \cdot \varphi_{wi}) \tag{4.34}$$

donde F(x) es una función sigmoide que aproxima la frontera de t.

$$\begin{aligned} F_a(x) &= \frac{\frac{a_{\mu_i}}{\Sigma_i\ c_{wi}}}{1 + e^{-\beta\left(x - a_{Th(t)}\right)}} \\ F_b(x) &= \frac{\frac{b_{\mu_i}}{\Sigma_i\ b_{wi}}}{1 + e^{-\beta\left(x - b_{Th(t)}\right)}} \\ F_c(x) &= \frac{\frac{c_{\mu_i}}{\Sigma_i\ a_{wi}}}{1 + e^{-\beta\left(x - c_{Th(t)}\right)}} \end{aligned} \tag{4.35}$$

Claramente, si β es lo suficientemente grande, cuando $x_\varphi > \varphi_{Th(t)}$, $e^{-b\left(x-\varphi_{Th(t)}\right)} \approx 0$, entonces $F(x) \approx \frac{\varphi_{\mu_j}}{\Sigma_i\ b_{wi}}$, cuando $x_\varphi < \varphi_{Th(t)}$, $e^{-b\left(x-\varphi_{Th(t)}\right)} \to \infty$, entonces $F(x) \approx 0$.

En la figura 4.7 $\frac{\varphi_{\mu_j}}{\Sigma_i\ b_{wi}}$ =0.70, $\varphi_{Th(t)}$ =0.55. Luego para cualquier valor de x_φ >0.55 $F(x)$ ≈0.70 y para cualquier valor de x_φ <0.55 $F(x) \approx 0$.

Definición 4.8 *Cuando una transición es habilitada, entonces es disparada tomando lugar la transmisión de señales de la siguiente manera:*

1. *Si un lugar p_{ok} tiene una sola transición de entrada t, una señal con valor*

$$[a_{w_{Ok}} a_{\alpha(t)}, b_{w_{Ok}} b_{\alpha(t)}, c_{w_{Ok}} c_{\alpha(t)}]$$

es depositada en cada lugar de salida p_{Ok}, $k = 1, 2, ..., m$ y todas las señales en $p_{Ij}, j = 1, 2, ..., n$ son removidas.

2. *Si un lugar (p_{Ok}) tiene más de una transición de entrada y más de una de estas se dispara, es decir, existe más de una ruta activa al mismo tiempo, entonces un nuevo valor de verdad $\alpha(p_{Ok})$ es calculado como el centro de gravedad de las transiciones disparadas:*

$$\alpha(p_{Ok}) = (a_{\alpha p_{Ok}}, b_{\alpha p_{Ok}}, c_{\alpha p_{Ok}}) \tag{4.36}$$

donde:

$$\begin{aligned} a_{\alpha p_{Ok}} &= \frac{\sum_{i=1}^{n}(a_{\alpha i} \cdot a_{\mu i})}{\sum_{i=1}^{n} c_{\mu i}} \\ b_{\alpha p_{Ok}} &= \frac{\sum_{i=1}^{n}(b_{\alpha i} \cdot b_{\mu i})}{\sum_{i=1}^{n} b_{\mu i}} \\ c_{\alpha p_{Ok}} &= \frac{\sum_{i=1}^{n}(c_{\alpha i} \cdot c_{\mu i})}{\sum_{i=1}^{n} a_{\mu i}} \end{aligned} \tag{4.37}$$

cuando t_j dispara, $t_j \in {}^{\cdot}p_{ok}$

De acuerdo a las definiciones arriba mencionadas, una transición t es habilitada si todos los lugares de entrada tienen señales, si el factor de certeza producido por estas es más grande que el valor frontera, entonces la transición t es disparada.

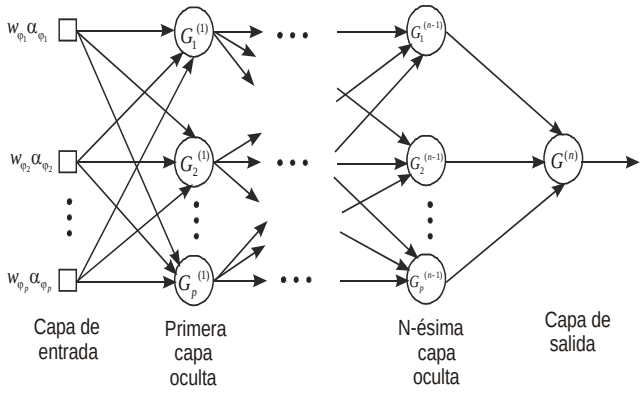


Figura 4.8: Red neuronal multicapa con $n-1$ capas ocultas y una neurona de salida.

B. Razonamiento hacia adelante (feedforward) Tomando una regla de producción difusa del Tipo 2 como una ilustración para mostrar el procedimiento de aprendizaje de conocimiento usando una RPDA. El Tipo 2 puede ser trasladado como una estructura de una red neuronal como se muestra en la figura 4.8. donde $G^{(i)}$ se define como sigue:

$$G_{\varphi}^{(i)}(x) = f_{\varphi}(x).x_{\varphi} \tag{4.38}$$

donde:

$$x_{\varphi} = W_{\varphi}^{T}\Lambda_{\varphi} = \sum_{i=1}^{n}(\varphi_{\alpha i}\cdot\varphi_{wi}), \quad \begin{aligned} f_a(x) &= \frac{\frac{a_{\mu_i}}{\Sigma_i\ c_{wi}}}{1+e^{-\beta\left(x_{\varphi}-a_{Th(t)}\right)}} \\ f_b(x) &= \frac{\frac{b_{\mu_i}}{\Sigma_i\ b_{wi}}}{1+e^{-\beta\left(x_{\varphi}-b_{Th(t)}\right)}} \\ f_c(x) &= \frac{\frac{c_{\mu_i}}{\Sigma_i\ a_{wi}}}{1+e^{-\beta\left(x_{\varphi}-c_{Th(t)}\right)}} \end{aligned} \tag{4.39}$$

β constante que ajusta la pendiente de $f_{\varphi}(x)$
W_{φ} vector de pesos $W_{\varphi} = [\varphi_{w_1}, \varphi_{w_2}, .., \varphi_{w_n}]^T$
Λ_{φ} vector de valores de verdad $\Lambda_{\varphi} = [\varphi_{\alpha_1}, \varphi_{\alpha_2}, .., \varphi_{\alpha_n}]^T$

La función continua $f(x)$ puede aproximar un factor lógico si μ, b y λ son seleccionados de manera adecuada. Para un lugar p, existen algunas rutas que van desde un conjunto de lugares fuente a este. Los pesos de estas rutas pueden ser estimados mediante el algoritmo *backpropagation* desarrollado. A lo largo de una ruta de seleccionada Ω, el proceso de propagación hacia adelante es tal que dados los valores de entrada U y los pesos w_i, la salida $O_\varphi(\Omega)$ puede ser expresada como:

$$O_\varphi(\Omega) = G_\varphi^{(n)} \left\{ W_\varphi^{(n)} G_\varphi^{(n-1)} [W_\varphi^{(n-1)} G_\varphi^{(n-2)} ... W_\varphi^{(2)} G_\varphi^{(1)} \times (W_\varphi^{(1)} U_\varphi)] \right\} \tag{4.40}$$

donde $G_\varphi^{(k)}(k = 1...n)$ es función de activación de la k-ésima capa, $W_\varphi^{(k)}(k = 1...n)$ es el vector de pesos de la k-ésima capa.

C. Backpropagation Si el vector de salida deseado es O_φ^*, entonces el error es calculado como:

$$e_n = O_\varphi(\Omega_i) - O_\varphi^*$$

El algoritmo de aprendizaje es el mismo que el algoritmo de *backpropagation* de redes neuronales multicapa, en el cual:

- Los pesos en la capa de salida son actualizados como

$$W_\varphi^{(n)}(k+1) = W_\varphi^{(n)}(k) - \gamma_i e_{\varphi n} \Lambda_\varphi^{(n)} \tag{4.41}$$

donde:

$\Lambda_\varphi^{(n)}$	$n - ésima$ capa de entrada
$\gamma_i > 0$	Ganancia adaptativa
$W_\varphi^{(n)}(k)$	Vector de pesos

$$e_{\varphi i} = e_{\varphi i+1} \dot{G}_\varphi{}^{(i+1)}(x) W_\varphi^{(i)} \tag{4.42}$$

Los pesos en las capas ocultas son actualizados como:

$$\begin{aligned} W_{\varphi}^{(n-1)}(k+1) &= W_{\varphi}^{(n-1)}(k) - \gamma_{n-1}\dot{G_{\varphi}}^{(n-1)} e_{\varphi n-1}\Lambda_{\varphi}^{(n-1)} \\ &\vdots \\ W_{\varphi}^{(2)}(k+1) &= W_{\varphi}^{(2)}(k) - \gamma_2 \dot{G_{\varphi}}^{(2)} e_{\varphi 2}\Lambda_{\varphi}^{(2)} \\ W_{\varphi}^{(1)}(k+1) &= W_{\varphi}^{(1)}(k) - \gamma_1 \dot{G_{\varphi}}^{(1)} e_{\varphi 1}\Lambda_{\varphi}^{(1)} \end{aligned} \tag{4.43}$$

Donde $\dot{G}$ es la derivada de la función no lineal G

$$\begin{aligned} \dot{G_{\varphi}} := \frac{d}{dx}\left[\frac{\frac{a_{\mu_i}}{\sum_i c_{wi}}}{1+e^{-\beta\left(x_a - a_{Th(t)}\right)}}\right] &= \frac{a_{\mu_i} x_a \beta e^{-\beta\left(x_a - a_{Th(t)}\right)}}{\left(1+e^{-\beta\left(x_a - a_{Th(t)}\right)}\right)^2 \sum_i c_{w_i}} + \\ &+ \frac{a_{\mu_i}}{\left(1+e^{-\beta\left(x_a - a_{Th(t)}\right)}\right)\sum_i c_{w_i}}; \varphi = a \\ \dot{G_{\varphi}} := \frac{d}{dx}\left[\frac{\frac{b_{\mu_i}}{\sum_i b_{wi}}}{1+e^{-\beta\left(x_b - b_{Th(t)}\right)}}\right] &= \frac{b_{\mu_i} x_b \beta e^{-\beta\left(x_b - b_{Th(t)}\right)}}{\left(1+e^{-b\left(x_b - b_{Th(t)}\right)}\right)^2 \sum_i b_{w_i}} + \\ &+ \frac{b_{\mu_i}}{\left(1+e^{-\beta\left(x_b - b_{Th(t)}\right)}\right)\sum_i b_{w_i}}; \varphi = b \\ \dot{G_{\varphi}} := \frac{d}{dx}\left[\frac{\frac{c_{\mu_i}}{\sum_i a_{wi}}}{1+e^{-\beta\left(x_c - c_{Th(t)}\right)}}\right] &= \frac{c_{\mu_i} x_c \beta e^{-\beta\left(x_c - c_{Th(t)}\right)}}{\left(1+e^{-\beta\left(x_c - c_{Th(t)}\right)}\right)^2 \sum_i a_{w_i}} + \\ &+ \frac{c_{\mu_i}}{\left(1+e^{-\beta\left(x_c - c_{Th(t)}\right)}\right)\sum_i a_{w_i}}; \varphi = c \end{aligned} \tag{4.44}$$

Luego, el algoritmo de aprendizaje de una $RPDA$ quedaría de la siguiente manera:

Paso 1 Se seleccionan un conjunto de pesos iniciales

Paso 2 Para conjunto de datos de entrada, encontrar las rutas activas y marcarlas estas como $\Omega_1, \Omega_2, ...\Omega_k$.

Paso 3 Siguiendo cada ruta activa, de acuerdo al algoritmo de razonamiento, calcular las salidas correspondientes.

Paso 4 Calcular el error, seleccionar γ_n y usar Ecs. (4.43) para ajustar los pesos de cada ruta.

4.5. Conclusión

En este capitulo introducimos las redes de Petri difusas adaptativas con números difusos triangulares (RPDA). Las reglas de producción de una base de reglas son modeladas por RPDA. Al igual que otros modelos de redes de Petri difusas, las RPDA pueden ser usadas para representar conocimiento y razonamiento, pero las RPDA tienen una importante ventaja: los pesos asociados a los lugares son ajustables. Basados en la regla de disparo, una modificación de los algoritmos de aprendizaje de Widrow-Hoff y *Backpropagation* es desarrollado para asegurar la convergencia de los pesos.

Capítulo 5

Ejemplos

En este capitulo se desarrollan tres ejemplos para ilustrar los procesos de razonamiento difuso con pesos y aprendizaje de pesos utilizando los resultados obtenidos en los capitulos anteriores

5.1. Aprendizaje con el Algoritmo de Widrow-Hoff

Ejemplo 5.1 *El siguiente sistema modela el encendido de una bomba centrifuga, esta se enciende si se cumplen las siguientes condiciones:*

1) El tinaco (tanque alto) deberá estar casi vacío;

2) La cisterna (tanque bajo) deberá estar semillena;

3) Debe existir suministro eléctrico.

Las dos condiciones iniciales se evalúan primero y si estas se cumplen la consecuente se evalúa con la tercera condición. Si alguna de estas tres condiciones no se cumple entonces la bomba no se enciende.

Modelando el sistema con reglas de producción tenemos d_1, d_2, d_3, d_4, d_5 *y* d_6 seis proposiciones de un sistema experto Γ_1, entre las cuales existen las siguientes reglas de producción difusas con pesos:

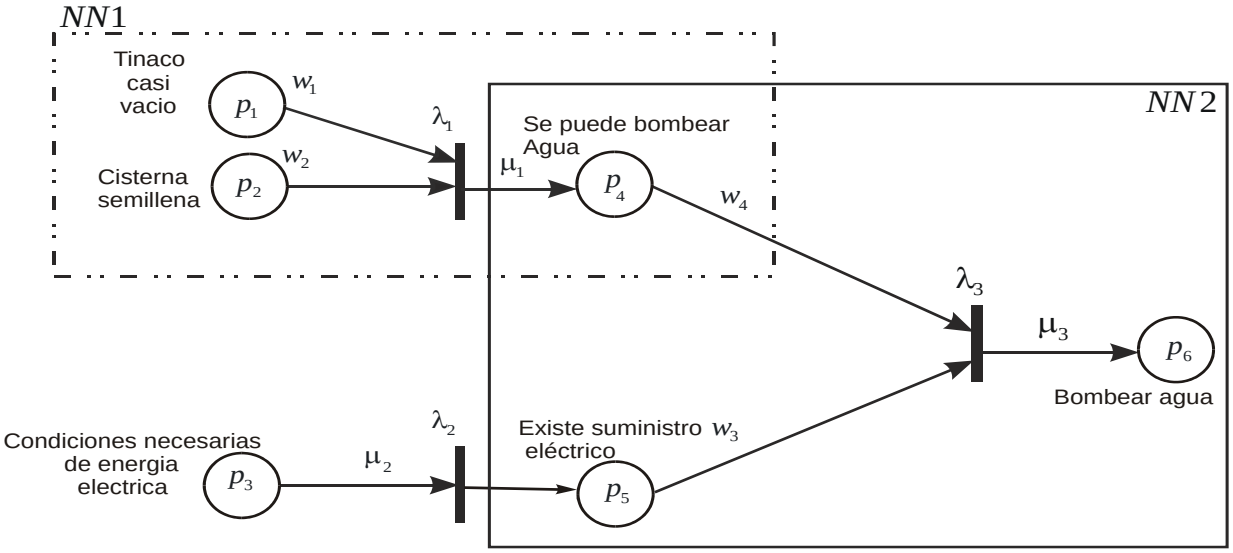


Figura 5.1: Red de Petri difusa del Ejemplo 5.1

R_1 :SI d_1(Tinaco está casi vacío) Y d_2 (Cisterna está semillena) Entonces d_4 (Se puede bombear agua) $(w_1, w_2), \mu_1, \lambda_1$;

R_2 :SI d_3 (Condiciones necesarias de energía eléctrica) Entonces d_4 (Existe suministro eléctrico) μ_2, λ_2;

R_3 :SI d_4 (Se puede bombear agua) Y d_5 (Condiciones necesarias de energía eléctrica) Entonces d_6 (Bombear agua) $(w_4, w_5), \mu_3, \lambda_3$;

Con las reglas de producción difusas anteriores podemos representar conocimiento. Usando la definición de una RPDA nuestra red de Petri del ejemplo quedaria de la siguiente manera:

$$RPDA = \{P, T, D, I, O, \alpha, \beta, Th, W\} \tag{5.1}$$

donde $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}$, $T = \{t_1, t_2, t_3\}$, $D = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$, $Th = \{\lambda_1, \lambda_2, \lambda_3\}$, $W_I = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$, $W_O = \{\mu_1, \mu_2, \mu_3\}$. Esta puede ser modelada como se muestra en la figura 5.1.

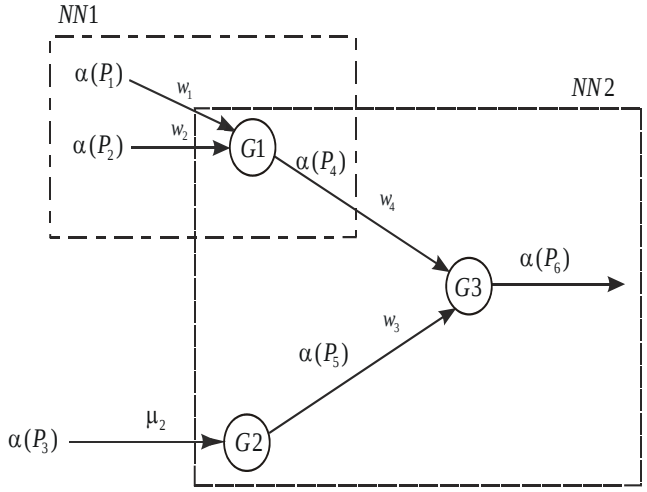


Figura 5.2: Conversión de la red de Petri difusa adaptativa de la figura 5.1 en una red neuronal.

Asumiendo que conocemos los factores de certeza, los valores frontera, pesos iniciales y valores de verdad de las proposiciones antecedentes, los cuales están representados por las siguientes funciones de membresía

$\mu_1 = [0.8\ 0.9\ 1.0]$	$\lambda_1 = [0.40\ \ 0.47\ \ 0.55]$
$\mu_2 = [0.7\ 0.8\ \ 0.9]$	$\lambda_2 = [0.50\ \ 0.60\ \ 0.70]$
$\mu_3 = [0.85\ \ 0.95\ \ 1.0]$	$\lambda_3 = [0.40\ \ 0.50\ \ 0.60]$

y

$w_1^* = [0.8\ \ 0.9\ \ 0.95]$	$w_1 = [0.3\ \ 0.4\ \ 0.5]$	$\alpha(p_1) = [0.45\ \ 0.5\ \ 0.6]$
$w_2^* = [0.71\ \ 0.83\ \ 0.95]$	$w_2 = [0.12\ \ 0.23\ \ 0.34]$	$\alpha(p_2) = [0.5\ \ 0.6\ \ 0.7]$
$w_3^* = [0.38\ \ 0.49\ \ 0.60]$	$w_3 = [0.3\ \ 0.4\ \ 0.5]$	$\alpha(p_3) = [0.9\ \ 0.95\ \ 1.0]$
$w_4^* = [0.45\ \ 0.56\ \ 0.77]$	$w_4 = [0.8\ \ 0.85\ \ 0.90]$	

Aplicando el algoritmo de razonamiento difuso desarrollado en el capitulo anterior el cual es descrito por las ecuaciones (4.32), (4.33), (4.34), (4.35), (4.36) y (4.37) tenemos que los valores de verdad de las proposiciones consecuentes son: $\alpha(p_4) = [0.30\ \ 0.49\ \ 0.81]$,

$\alpha(p_5) = [0.63 \;\; 0.76 \;\; 0.90]$ y $\alpha(p_6) = [0.23\ 0.58\ 1.4]$

Si los pesos deseados son desconocidos, entonces podemos usar el algoritmo propuesto para aprender los pesos. Las partes adaptativas de la RPDA de la figura 5.1 pueden ser transformadas como lo muestra la figura 5.2.

Aplicando la ley de aprendizaje de Widrow-Hoff con $t_{1(k)} = [0.30 \;\; 0.49 \;\; 0.81]$ y $t_{2(k)} = [0.23\ 0.58\ 1.4]$, $y_1(k) = [\alpha(p_4)]$, $y_2(k) = [\alpha(p_6)]$; $P_{1(k)} = [\alpha(p_1), \alpha(p_2)]$, $P_{2(k)} = [\alpha(p_3), \alpha(p_4)]$, $\beta = 20$. Los resultados del aprendizaje son mostrados en la figura 5.3, como se puede ver en esta figura los pesos convergen a sus valores reales cuando $k > 40$.

5.2. Aprendizaje con el Algoritmo *Backpropagation*

Ejemplo 5.2 *El sistema experto* Γ_2 *evalúa sólo dos conjuntos (existen más causas) de fallos por las cuales un motor de inyección electrónica tiene un bajo rendimiento por litro en base a las siguientes reglas de producción difusas con pesos:*

Sean $d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}$ y d_{12} doce proposiciones de un sistema experto Γ_2.

R_1 :SI d_1 y d_2 y d_3 Entonces d_6 $(w_1, w_2, w_3), \mu_1, \lambda_1$;

R_2 :SI d_4 Entonces d_8 ; μ_2, λ_2;

R_3 :SI d_5 y d_6 y d_7 Entonces d_{10} $(w_5, w_6, w_7), \mu_3, \lambda_3$;

R_4 :SI d_8 y d_9 Entonces d_{11} $(w_8, w_9), \mu_4, \lambda_4$;

R_5 : SI d_{10} o d_{11} Entonces d_{12} ; $\mu_5, \mu_6, \lambda_5 \lambda_6$

donde:

d_1 = Baja potencia

d_2 = Funcionamiento brusco

d_3 = Marcha mínima errática

d_4 = Jaloneo

d_5 = Filtros de combustible en buen estado

d_6 = Flujo de inyector alterado

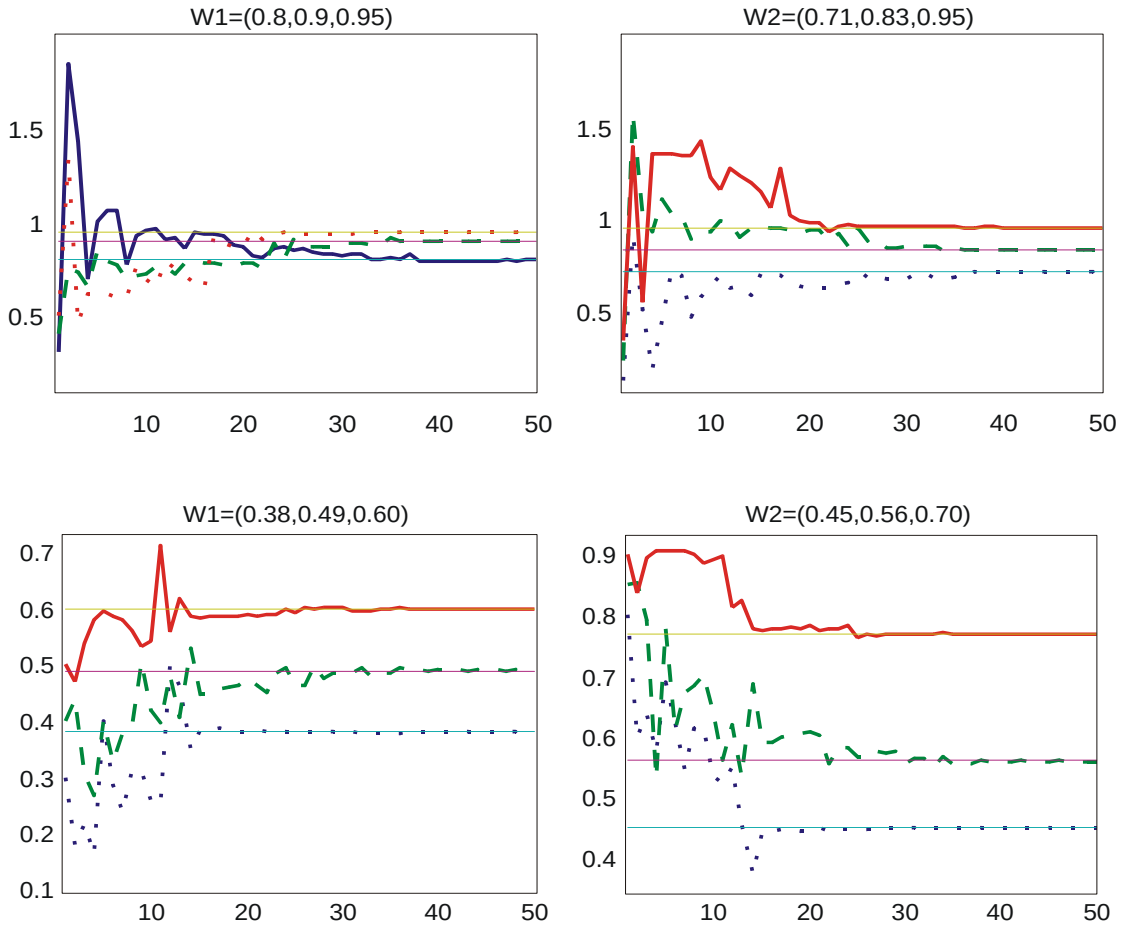


Figura 5.3: Aprendizaje de pesos

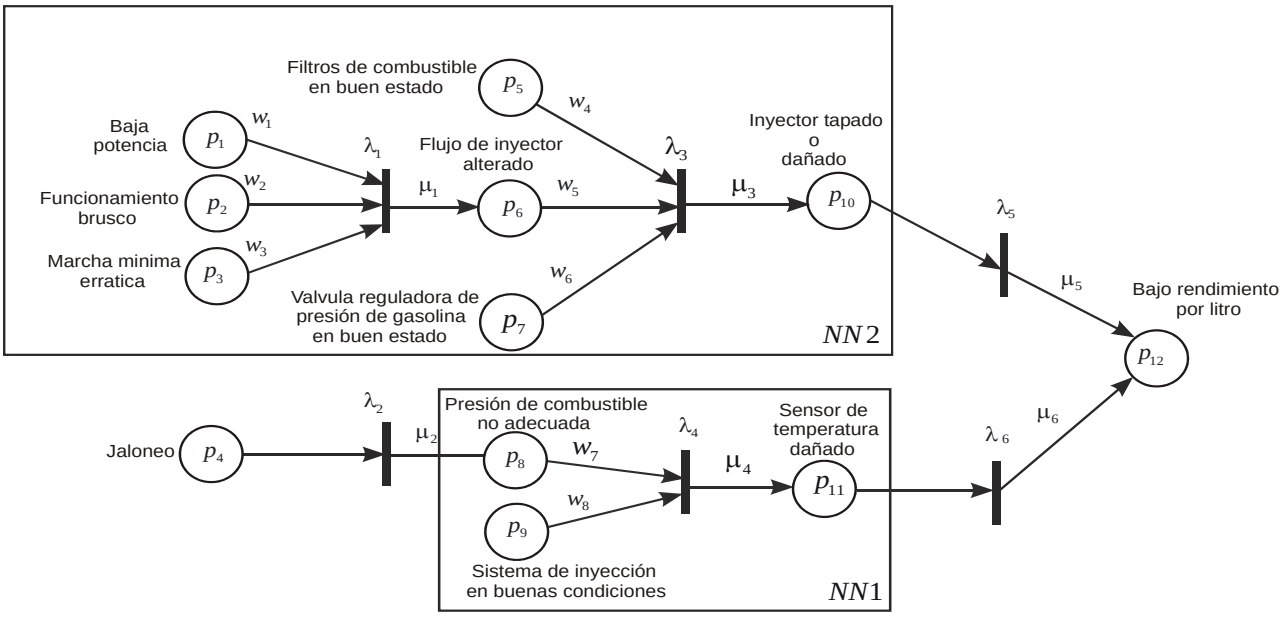


Figura 5.4: Red de Petri difusa adaptativa del Ejemplo 5.2.

d_7 = Válvula reguladora de presión de gasolina en buen estado

d_8 = Presión de combustible no adecuada

d_9 = Sistema de inyección en buenas condiciones

d_{10} = Inyector tapado o dañado

d_{11} = Sensor de temperatura dañado

d_{12} = Bajo rendimiento por litro

Este ejemplo incluye tres tipos de reglas de producción, las reglas 1,3 y 4 son reglas de tipo 2, la regla de producción 2 es una regla de tipo 1, mientras que la regla de producción 5 es de tipo 4. A continuación mostramos el razonamiento difuso y el algoritmo de aprendizaje de pesos.

Trasladando las reglas de producción difusas en una RPDA esta quedaría como (ver figura 5.4) sigue:

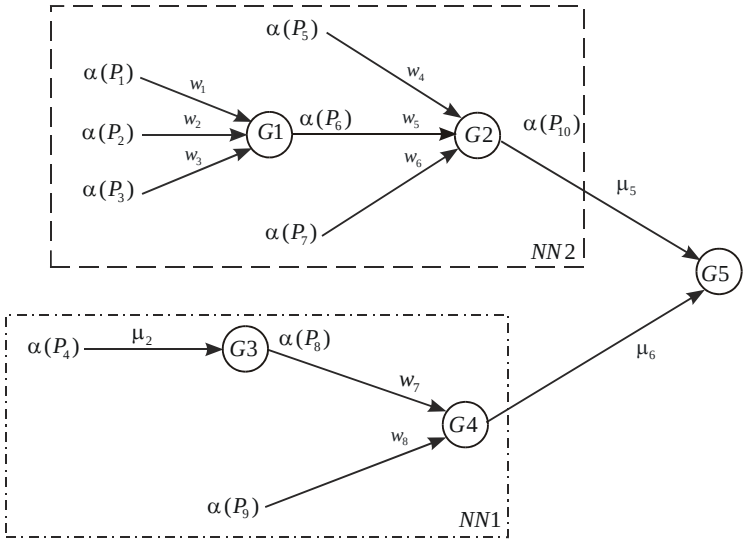


Figura 5.5: Conversión de la red de Petri difusa adaptativa de la figura 5.4 en una red neuronal.

$$RPDA = \{P, T, D, I, O, \alpha, \beta, Th, W\} \tag{5.2}$$

donde $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}\}$, $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $D = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}\}$, $Th = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\}$, $W_I = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8\}$, $W_O = \{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6\}$

Con siete proposiciones de entrada $(p_1, p_2, p_3, p_4, p_5, p_7, p_9)$ y cinco proposiciones consecuentes, los datos son dados como sigue:

$\mu_1 = [0.80 \;\; 0.90 \;\; 1.0]$	$\lambda_1 = [0.31 \; 0.45 \; 0.58]$
$\mu_2 = [0.70 \;\; 0.80 \;\; 0.90]$	$\lambda_2 = [0.30 \; 0.40 \;\; 0.50]$
$\mu_3 = [0.85 \;\; 0.95 \;\; 1.0]$	$\lambda_3 = [0.25 \;\; 0.35 \;\; 0.45]$
$\mu_4 = [0.90 \;\; 0.95 \;\; 1.0]$	$\lambda_4 = [0.33 \;\; 0.43 \;\; 0.53]$
$\mu_5 = [0.85 \;\; 0.95 \;\; 1.0]$	$\lambda_5 = [0.20 \;\; 0.30 \;\; 0.40]$
$\mu_6 = [0.80 \;\; 0.90 \;\; 1.0]$	$\lambda_6 = [0.21 \;\; 0.33 \;\; 0.45]$

El modelo de RPDA mostrado en la figura 5.4 puede ser convertido en una red neuronal como lo muestra la figura 5.5. Simplificando la red neuronal en dos subredes NN1 y NN2, como lo muestran los recuadros punteados de la figura 5.5. En donde, la subred neuronal NN1 es una red de una sola capa y la subred NN2 es una red multicapa.

Para este ejemplo entrenaremos las dos redes independientemente

Para la Ruta Activa NN1

Para G4 usamos las sigmoides

$$F_{4a}(x) = \frac{\frac{0.70}{\Sigma_i\ c_{wi}}}{1 + e^{-\beta(x-0,3)}}$$
$$F_{4b}(x) = \frac{\frac{0.80}{\Sigma_i\ b_{wi}}}{1 + e^{-\beta(x-0,4)}}$$
$$F_{4c}(x) = \frac{\frac{0.90}{\Sigma_i\ a_{wi}}}{1 + e^{-\beta(x-0,5)}}$$

para aproximar los valores frontera, con $\beta = 20$.

Para la transición t_4, la función argumento $G(x)$ es:

$$x = \alpha(P_8)w_7 + \alpha(P_9)w_{8.}$$

Usando el algoritmo de razonamiento difuso, un conjunto de datos de salida (valores de verdad de proposiciones consecuentes) puede ser calculado por medio de los datos de entrada (valores de verdad de proposiciones antecedentes).

Si los pesos son desconocidos, las redes neuronales pueden ser usadas para estimar los pesos. Asumiendo que los pesos ideales son números triangulares difusos definidos como:

$$w_7^* = [0.47\ \ 0.57\ \ 0.67],\ w_8^* = [0.29\ \ 0.39\ \ 0.49].$$

Si las entradas $\alpha(P_4)$ y $\alpha(P_9)$ son dados aleatoriamente, podemos obtener una salida $\alpha(P_{11})$ por medio de las reglas de producción del sistema experto Γ_2. Dadas condiciones

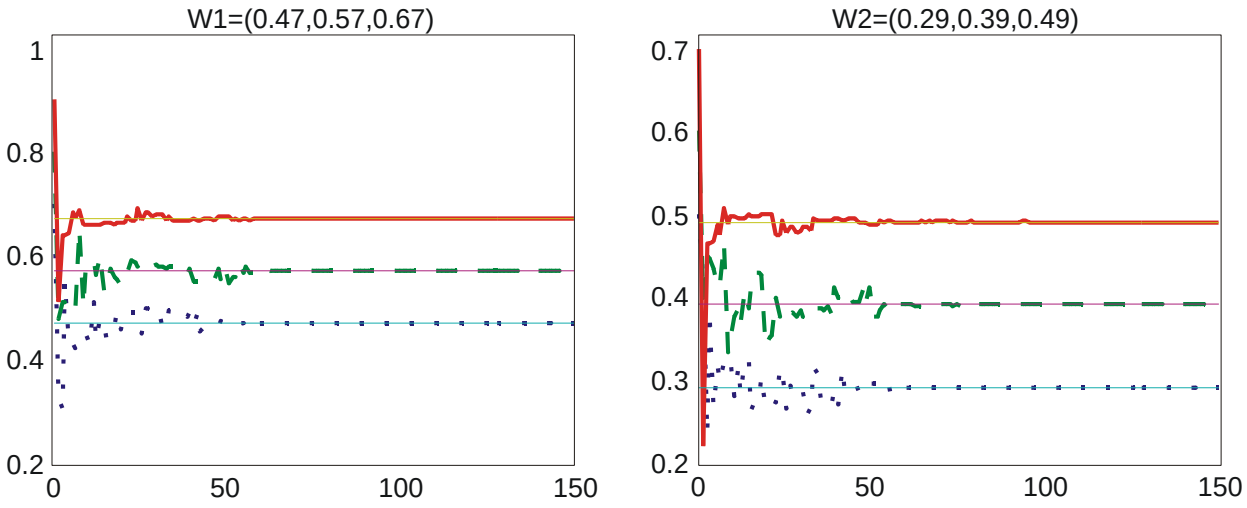


Figura 5.6: Resultados de aprendizaje del Ejemplo 5.2

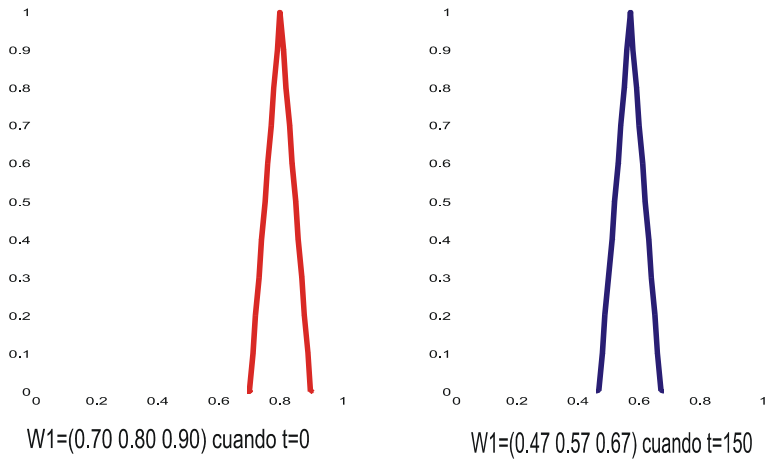


Figura 5.7: Números difusos antes y después del aprendizaje de pesos de W1.

iniciales para $w_7 = [0.7 \;\; 0.8 \;\; 0.9]$ y $w_8 = [0.5 \;\; 0.6 \;\; 0.7]$ evaluadas con las mismas entradas a la red neuronal, el error entre la salida de la red neuronal ($\alpha'(P_{11})$) y la salida del sistema experto Γ_2 ($\alpha(P_{11})$) puede ser usado para modificar los pesos usando la siguiente ley de aprendizaje:

$$\begin{aligned} W_\varphi(k+1) &= W_\varphi(k) + \dot{G}\delta e_\varphi(k)\Lambda_\varphi(P(k)) \\ e_\varphi(k) &: = \alpha(P_{11})(k) - \alpha'(P_{11})(k) \end{aligned} \tag{5.3}$$

donde δ es el instante de aprendizaje, $W_\varphi(k) = [\varphi_{w_7}(k), \varphi_{w_8}(k)]$, $\Lambda_\varphi(P(k)) = [\varphi_{P_8}(k), \varphi_{P_9}(k)]$. Después del proceso de entrenamiento ($k > 60$) los pesos convergen a los valores reales. La figura 5.6 muestra los resultados de la simulación, Mientras que la figura 5.7 muestra el peso W1 cuando $k = 0$ y cuando $k = 150$.

Para la Ruta Activa NN2

Para G2 usamos las sigmoides

$$F_{1a}(x) = \frac{\dfrac{0.8}{\Sigma_i \; c_{wi}}}{1 + e^{-\beta(x-0.31)}} \qquad F_{2a}(x) = \frac{\dfrac{0.85}{\Sigma_i \; c_{wi}}}{1 + e^{-\beta(x-0.25)}}$$

$$F_{1b}(x) = \frac{\dfrac{0.90}{\Sigma_i \; b_{wi}}}{1 + e^{-\beta(x-0.45)}} \quad y \quad F_{2b}(x) = \frac{\dfrac{0.95}{\Sigma_i \; b_{wi}}}{1 + e^{-\beta(x-0.35)}}$$

$$F_{1c}(x) = \frac{\dfrac{1.0}{\Sigma_i \; a_{wi}}}{1 + e^{-\beta(x-0.58)}} \qquad F_{2c}(x) = \frac{\dfrac{1.0}{\Sigma_i \; a_{wi}}}{1 + e^{-\beta(x-0.45)}}$$

para aproximar los valores frontera, con $\beta = 20$.

Para la transición t_1, la función argumento $G_1(x)$ es:

$$x = \alpha(P_1)w_1 + \alpha(P_2)w_2 + \alpha(P_3)w_3$$

mientras que para la transición t_3, la función argumento $G_2(x)$ es:

$$x = \alpha(P_5)w_4 + \alpha(P_6)w_5 + \alpha(P_7)w_6$$

Usando el algoritmo de razonamiento difuso, un conjunto de datos de salida (valores de verdad de proposiciones consecuentes) puede ser calculado por medio de los datos de entrada (valores de verdad de proposiciones antecedentes).

Si los pesos son desconocidos, las redes neuronales pueden ser usadas para estimar los pesos. Asumiendo que los pesos ideales son números triangulares difusos definidos como:

$$\begin{aligned} w_1^* &= [0.30 \;\; 0.50 \;\; 0.70] \\ w_2^* &= [0.35 \;\; 0.45 \;\; 0.55] \\ w_3^* &= [0.55 \;\; 0.65 \;\; 0.75] \\ w_4^* &= [0.60 \;\; 0.70 \;\; 0.80] \\ w_5^* &= [0.60 \;\; 0.80 \;\; 0.95] \\ w_6^* &= [0.20 \;\; 0.40 \;\; 0.60]\,. \end{aligned}$$

Si las entradas $\alpha(P_1)$, $\alpha(P_2)$, $\alpha(P_3)$, $\alpha(P_5), \alpha(P_6)$ y $\alpha(P_7)$ son dados aleatoriamente, podemos obtener una salida $\alpha(P_{10})$ por medio de las reglas de producción del sistema experto Γ_2. Dadas condiciones iniciales para los pesos

$$\begin{aligned} w_1 &= [0.70 \;\; 0.80 \;\; 0.90] \\ w_2 &= [0.10 \;\; 0.20 \;\; 0.30] \\ w_3 &= [0.40 \;\; 0.50 \;\; 0.60] \\ w_4 &= [0.60 \;\; 0.70 \;\; 0.80] \\ w_5 &= [0.50 \;\; 0.60 \;\; 0.75] \\ w_6 &= [0.20 \;\; 0.25 \;\; 0.35]\,. \end{aligned}$$

Si estos pesos se evalúan con las mismas entradas a la red neuronal, el error entre la salida de la red neuronal ($\alpha'(P_{10})$) y la salida del sistema experto Γ_2 ($\alpha(P_{10})$) puede ser usado para modificar los pesos usando la siguiente ley de aprendizaje:

$$\begin{aligned} W_\varphi(k+1) &= W_\varphi(k) + \dot{G}\delta e_\varphi(k)\Lambda_\varphi(P(k)) \\ e_\varphi(k) &: = \alpha(P_{10})(k) - \alpha'(P_{10})(k) \end{aligned} \tag{5.4}$$

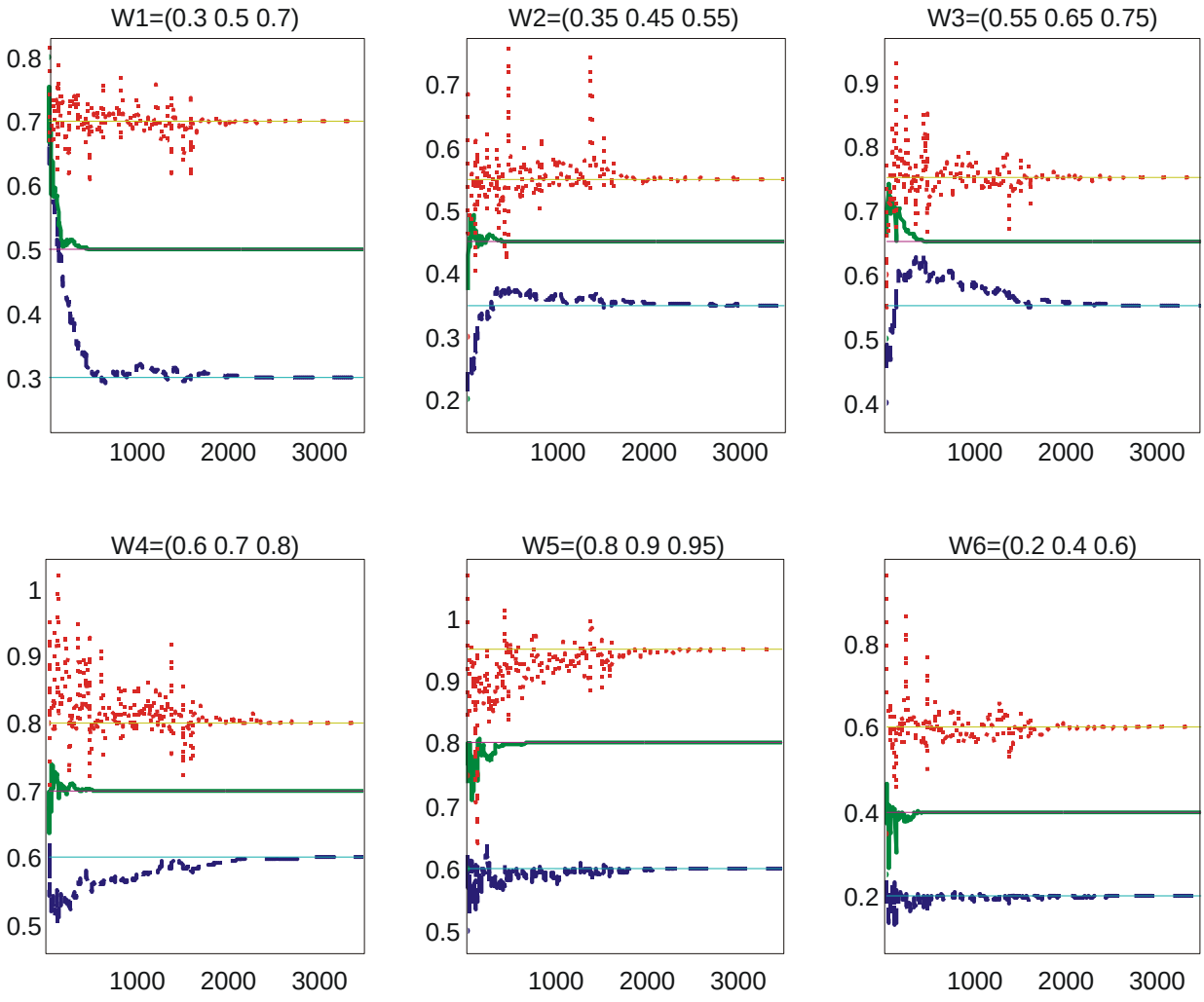


Figura 5.8: Aprendizaje de pesos para red multicapa.

donde δ es el instante de aprendizaje, $W_\varphi(k) = [\varphi_{w_7}(k), \varphi_{w_8}(k)]$, $\Lambda_\varphi(P(k)) = [\varphi_{P_8}(k), \varphi_{P_9}(k)]$. Después del proceso de entrenamiento ($k > 2000$) los pesos convergen a los valores reales. La figura 5.8 muestra los resultados de la simulación.

Ejemplo 5.3 *Consideremos el ejemplo para determinar el precio de venta de un auto de segunda mano, donde son esquematizados cuatro pasos de razonamiento (ver figura 5.9) que envuelven los siguientes 12 atributos:*

- Cuarto paso:

  Distancia recorrida por el auto (KM)

  Año del auto (MA)

  Sexo del conductor (SC)

  Edad del conductor (EC)

  Tipo de auto (TA)

  Poder del motor (PM)

- Tercer paso:

  Grado de uso del auto (UA)

  Mantenimiento del auto (MNA)

  Forma de manejo (FM)

- Segundo paso:

  Estado del auto (EA)

  Precio de un auto nuevo (VAN)

- Primer paso:

  Precio del auto de segunda mano. (VAU)

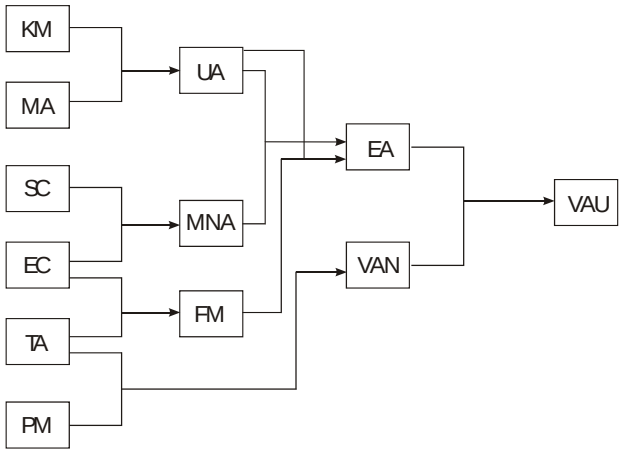


Figura 5.9: Esquema de cuatro pasos de razonamiento.

Entre los atributos anteriores existen las siguientes reglas de producción difusas, mediante las cuales se modela la red de Petri difusa de las figuras 5.10 y 5.11.

R_1:IF KM= Grande y MA= Reciente THEN UA= Alto

R_2:IF KM= Grande y MA= Viejo THEN UA= Alto

R_3:IF KM= Medio y MA= Reciente THEN UA= Medio a Alto

R_4:IF KM= Medio y MA= Viejo THEN UA= Poco

R_5:IF KM= Pequeño y MA= Reciente THEN UA= Poco

R_6:IF KM= Pequeño y MA= Viejo THEN UA= Poco

R_7:IF SC= Hombre y EC= Joven THEN MC= Malo

R_8:IF SC= Hombre y EC=Edad media THEN MC= Bueno

R_9:IF SC= Hombre y EC=Viejo THEN MC= Regular a Bueno

R_{10}:IF SC= Mujer THEN FM= Peligrosa

R_{11}:IF TA= Compacto y PM= Alto THEN VAN= Medio a Caro

R_{12}:IF TA= Compacto y PM= Bajo THEN VAN= Barato

R_{13}:IF TA= Pickup y PM= Alto THEN VAN= Medio a Caro

R_{14}:IF TA= Pickup y PM= Bajo THEN VAN= Barato

R_{15}:IF TA= Auto Urbano y PM= Alto THEN VAN= Medio a Caro

R_{16}:IF TA= Auto Urbano y PM= Bajo THEN VAN= Barato

R_{17}:IF TA= Deportivo y PM= Alto THEN VAN= Caro

R_{18}:IF TA= Deportivo y PM= Bajo THEN VAN= Medio a Caro

R_{19}:IF TA= De Lujo THEN VAN= Caro

R_{20}:IF TA= Camioneta y PM= Alto THEN VAN= Caro

R_{21}:IF TA= Camioneta y PM= Bajo THEN VAN= Precio Medio a Caro

R_{22}:IF UA= Alto y FM= Peligrosa THEN EA= Malo

R_{23}:IF UA= Alto y MNA= Malo THEN EA= Malo

R_{24}:IF UA= Alto y MNA= Regular a Bueno THEN EA= Regular

R_{25}:IF UA= Alto y MNA= Bueno THEN EA= Regular

R_{26}:IF UA= Medio a Alto y FM= Peligrosa THEN EA= Malo

R_{27}:IF UA= Medio a Alto y MNA= Malo THEN EA= Malo

R_{28}:IF UA= Medio a Alto y MNA= Regular a Bueno THEN EA= Regular

R_{29}:IF UA= Medio a Alto y MNA= Bueno THEN EA= Regular

R_{30}:IF UA= Poco y FM= Peligrosa THEN EA= Regular

R_{31}:IF UA= Poco y MNA= Malo THEN EA= Regular

R_{32}:IF UA= Poco y MNA= Regular a Bueno THEN EA= Bueno

R_{33}:IF UA= Poco y MNA= Bueno THEN EA= Bueno

R_{34}:IF EA= Malo y VAN= Barato THEN VAU= Barato

R_{35}:IF EA= Malo y VAN= Medio a Caro THEN VAU= Barato

R_{36}:IF EA= Malo y VAN= Caro THEN VAU= Precio Medio

R_{37}:IF EA= Regular y VAN= Barato THEN VAU= Barato

R_{38}:IF EA= Regular y VAN= Medio a Caro THEN VAU= Precio Medio

R_{39}:IF EA= Regular y VAN= Caro THEN VAU= Medio a Caro

R_{40}:IF EA= Bueno y VAN= Barato THEN VAU= Barato

R_{41}:IF EA= Bueno y VAN= Medio a Caro THEN VAU= Precio Medio

R_{42}:IF EA= Bueno y VAN= Caro THEN VAU= Medio a Caro

De la red de Petri de la figura 5.10 tenemos las proposiciones asociadas a los lugares las cuales se muestran en la 5.1.

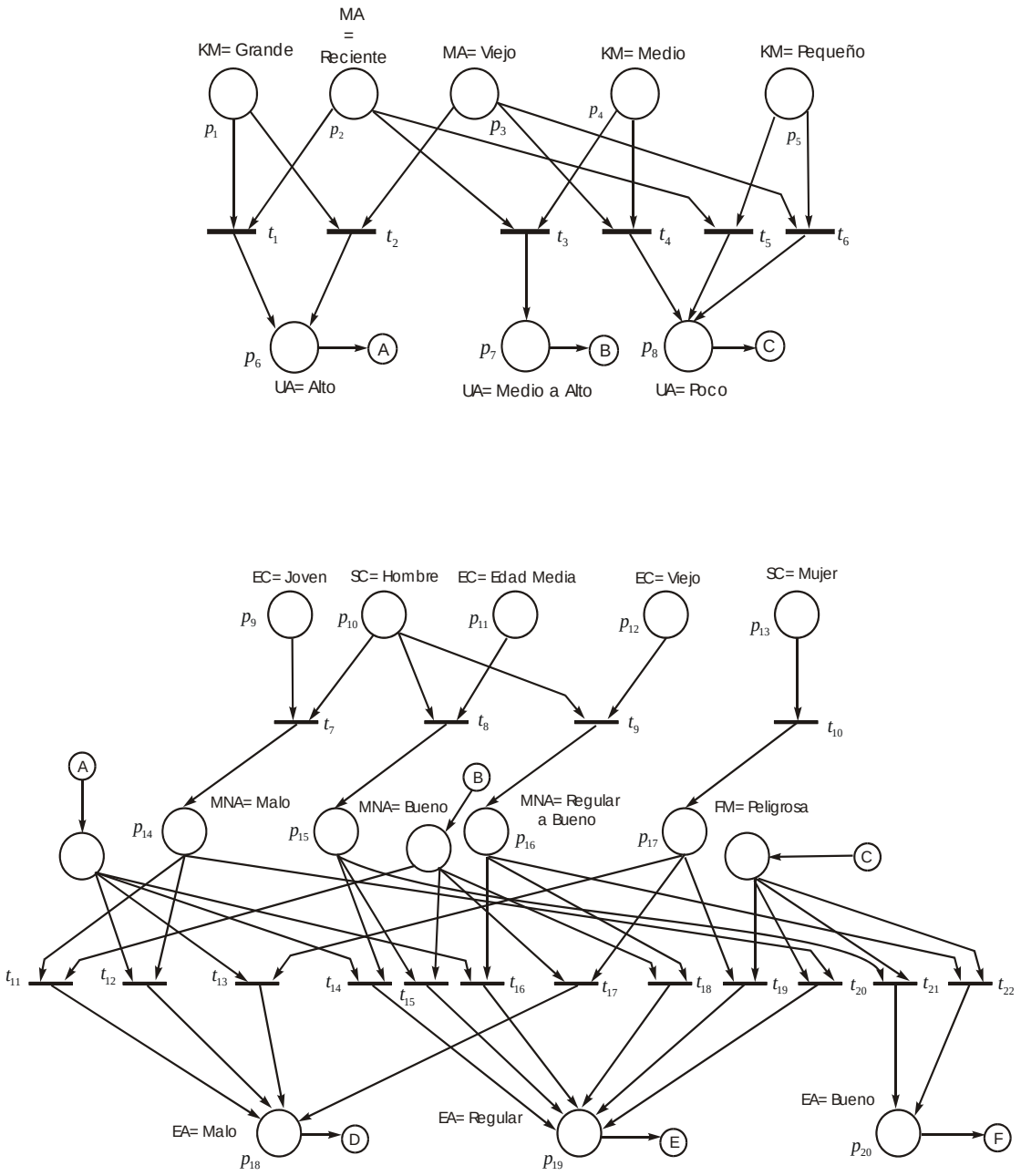


Figura 5.10: Red de Petri difusa (parte (a)) del Ejemplo 5.3

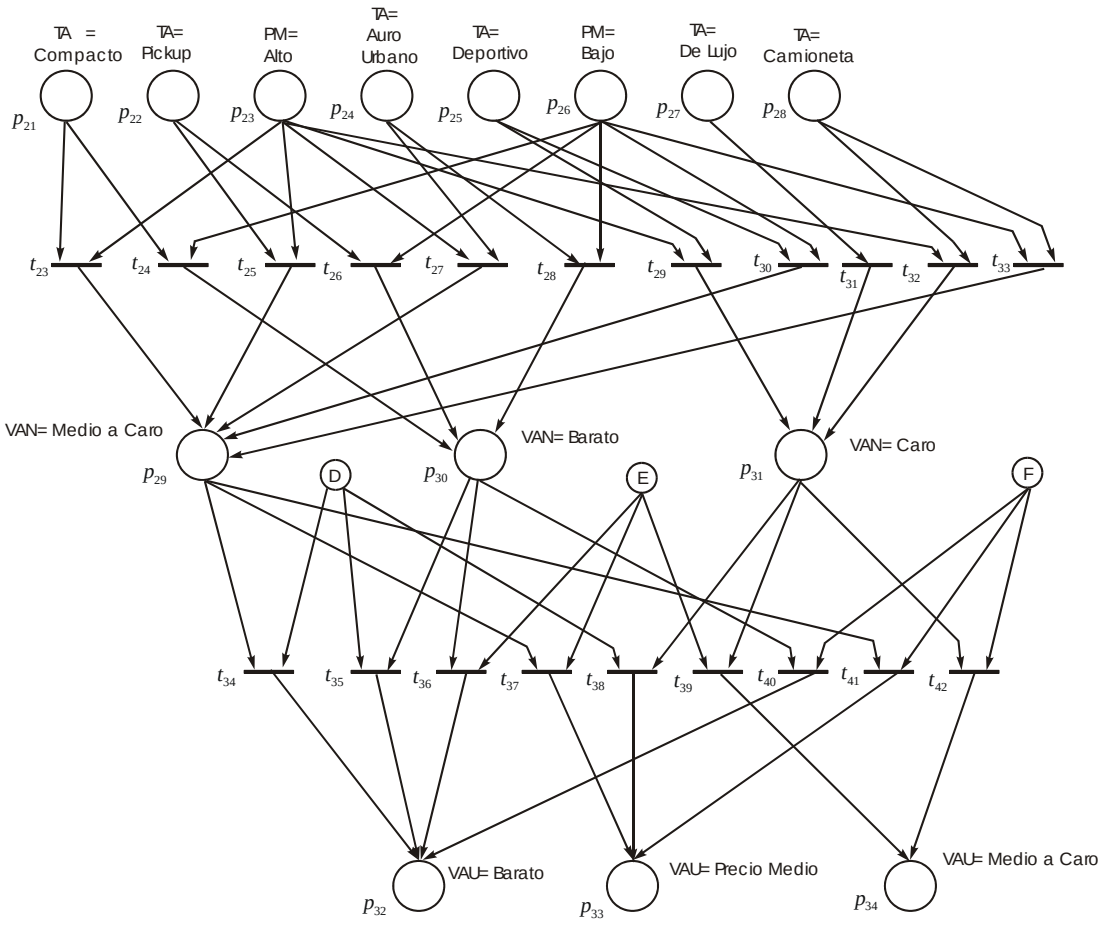


Figura 5.11: Red de Petri difusa (parte (b)) del Ejemplo 5.3

Cuadro 5.1: Proposiciones asociadas a los lugares

Lugar p_i	Proposición $\beta(p_i)$	Lugar p_i	Proposición $\beta(p_i)$
p_1	KM=Grande	p_{18}	EA=Malo
p_2	MA=Reciente	p_{19}	EA=Regular
p_3	MA=Viejo	p_{20}	EA=Bueno
p_4	KM=Medio	p_{21}	TA=Compacto
p_5	KM=Pequeño	p_{22}	TA=Pickup
p_6	UA=Alto	p_{23}	PM=Alto
p_7	UA=Medio a Alto	p_{24}	TA=Auto Urbano
p_8	UA=Poco	p_{25}	TA=Deportivo
p_9	EC=Joven	p_{26}	PM=Bajo
p_{10}	SC=Hombre	p_{27}	TA=De Lujo
p_{11}	EC=Edad Media	p_{28}	TA=Camioneta
p_{12}	EC=Viejo	p_{29}	VAN=Medio a Caro
p_{13}	SC=Mujer	p_{30}	VAN=Barato
p_{14}	MNA=Malo	p_{31}	VAN=Caro
p_{15}	MNA=Bueno	p_{32}	VAU=Barato
p_{16}	MNA=Regular a Bueno	p_{33}	VAU=Precio Medio
p_{17}	FM=Peligrosa	p_{34}	VAU=Precio Medio a Caro

Convirtiendo cada valor lingüístico de la tabla a su correspondiente número difuso triangular tenemos:

Variable	Valor Lingüístico	Valor Numérico
KM	Grande	[0.1 0.2 0.3]
	Medio	[0.5 0.6 0.7]
	Pequeño	[0.8 0.9 1.0]
MA	Reciente	[0.8 0.9 1.0]
	Viejo	[0.2 0.3 0.4]
SC	Hombre	[0.8 0.9 1.0]
	Mujer	[0.5 0.6 0.7]
EC	Joven	[0.5 0.6 0.7]
	Edad Media	[0.9 0.95 1.0]
	Viejo	[0.7 0.8 0.9]
TA	Compacto	[0.7 0.8 0.9]
	Pickup	[0.7 0.8 0.9]
	Auto-Urbano	[0.7 0.8 0.9]
	Deportivo	[0.8 0.9 1.0]
	Camioneta	[0.8 0.9 1.0]
	De Lujo	[0.9 0.95 1.0]
PM	Bajo	[0.5 0.6 0.7]
	Alto	[0.8 0.9 1.0]

De la tabla se observa que algunos valores lingüísticos como "grande" de la variable "KM" poseen un valor numérico más pequeño en comparación con el valor lingüístico "pequeño" de la misma variable. Ya que, para obtener el valor del auto usado (VAU) tenemos ajustar los valores numéricos de acuerdo a las características "deseables", en este caso, el que un auto tenga un gran kilometraje recorrido es una característica "no deseable" por ello, el valor numérico asociado a esta.

Suponiendo que tenemos nueve autos usados con las características mostradas en la Tabla

Cuadro 5.2: Valores lingüísticos de la proposiciones antecedentes

Auto	KM	MA	SC	EC	TA	PM
1	Grande	Reciente	Hombre	Joven	De Lujo	Alto
2	Medio	$Viejo$	Hombre	Edad Media	Pickup	Alto
3	Pequeño	Reciente	Mujer		Camioneta	Alto
4	Grande	$Viejo$	Mujer		Compacto	Bajo
5	Medio	Reciente	Hombre	Edad Media	Deportivo	Alto
6	Pequeño	$Viejo$	Hombre	Viejo	Auto Urbano	Alto
7	Pequeño	Reciente	Hombre	Edad Media	Camioneta	Alto
8	Grande	$Viejo$	Hombre	Joven	Pickup	Bajo
9	Grande	Reciente	Mujer		Auto Urbano	Bajo

Cuadro 5.3: Valores Numéricos de la proposiciones antecedentes

Auto	KM	MA	SC	EC	TA	PM
1	[.1 .2 .3]	[.8 .9 1]	[.8 .9 1]	[.5 .6 .7]	[.9 .95 1]	[.8 .9 1]
2	[.5 .6 .7]	[.2 .3 .4]	[.8 .9 1]	[.9 .95 1]	[.7 .8 .9]	[.8 .9 1]
3	[.8 .9 1]	[.8 .9 1]	[.5 .6 .7]		[.8 .9 1]	[.8 .9 1]
4	[.1 .2 .3]	[.2 .3 .4]	[.5 .6 .7]		[.7 .8 .9]	[.5 .6 .7]
5	[.5 .6 .7]	[.8 .9 1]	[.8 .9 1]	[.9 .95 1]	[.8 .9 1]	[.8 .9 1]
6	[.8 .9 1]	[.2 .3 .4]	[.8 .9 1]	[.7 .8 .9]	[.7 .8 .9]	[.8 .9 1]
7	[.8 .9 1]	[.8 .9 1]	[.8 .9 1]	[.9 .95 1]	[.8 .9 1]	[.8 .9 1]
8	[.1 .2 .3]	[.2 .3 .4]	[.8 .9 1]	[.5 .6 .7]	[.7 .8 .9]	[.5 .6 .7]
9	[.1 .2 .3]	[.8 .9 1]	[.5 .6 .7]		[.7 .8 .9]	[.5 .6 .7]

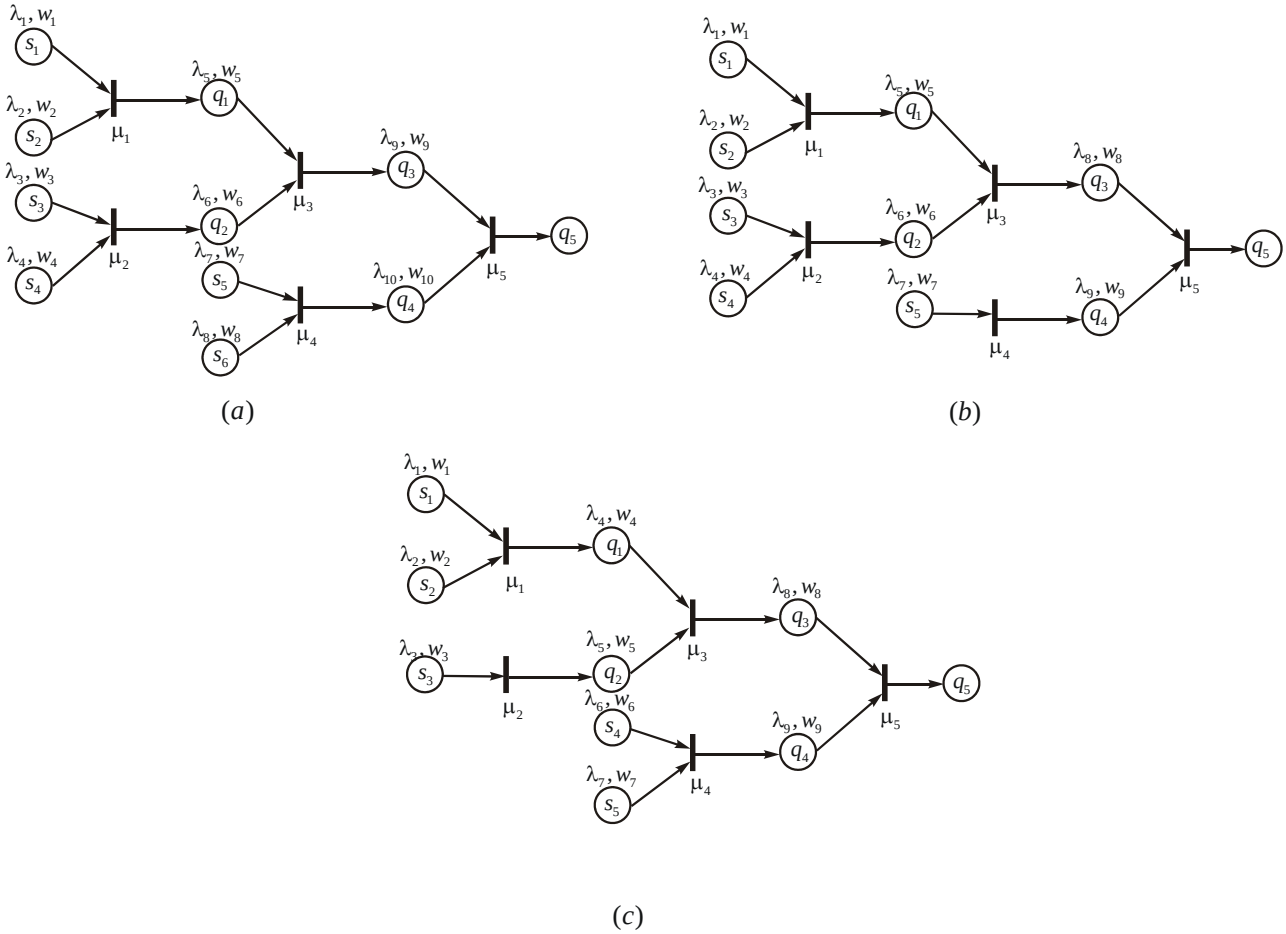


Figura 5.12: Trayectorias recorridas en la red de Petri

Cuadro 5.4: Valores lingüísticos de la proposiciones consecuentes

Auto	UA	MNA	FM	EA	VAN	VAU
1	Alto	Malo		Malo	Caro	Medio
2	Poco	Bueno		Bueno	Medio a Caro	Medio
3	Poco		Peligrosa	Regular	Caro	Medio a Caro
4	Alto		Peligrosa	Malo	Barato	Barato
5	Medio a Alto	Bueno		Regular	Caro	Medio a Caro
6	Poco	Regular a Bueno		Bueno	Medio a Caro	Medio
7	Poco	Bueno		Bueno	Caro	Medio a Caro
8	Alto	Malo		Malo	Barato	Barato
9	Alto		Peligrosa	Malo	Barato	Barato

5.4. Evaluando el valor de verdad de la proposición final para cada uno de ellos, tenemos que la trayectoria recorrida en la red de Petri de la figura 5.10 es más simple (ver figura 5.12). Específicamente para el ejemplo 1 la trayectoria recorrida en la red de Petri original es como el inciso (b), para ejemplos 3,4 y 9 la trayectoria recorrida es como en el inciso (c) y para ejemplos 2,5,6,7 y 8 la trayectoria recorrida es similar al inciso (a). Los valores de verdad de las proposiciones consecuentes y sus respectivos valores lingüísticos son mostrados en la Tabla 5.4 y Tabla 5.5 respectivamente.

Las tres trayectorias o caminos mostrados en la figura 5.12 pueden ser trasladadas redes neuronales multicapa como se muestra en la figura 5.13. Entrenando cada una de las redes tenemos:

Para la Ruta Activa T1

Para $G5$ usamos las sigmoides:

Cuadro 5.5: Valores Numéricos de la proposiciones consecuentes

Auto	UA	MNA	FM	EA	VAN	VAU
1	[.24 .45 .78]	[.41 .68 1.08]		[.2 .51 1.2]	[.72 .85 1]	[.33 .63 1.32]
2	[.23 .42 .72]	[.53 .83 1.26]		[.23 .57 1.32]	[.47 .76 1.2]	[.24 .6 1.49]
3	[.49 .81 1.28]		[.4 .54 .7]	[.27 .6 1.29]	[.5 .81 1.26]	[.27 .64 1.52]
4	[.088 .21 .44]		[.4 .54 .7]	[.15 .34 .75]	[.38 .63 1.02]	[.19 .45 1.06]
5	[.39 .65 1.07]	[.53 .83 1.26]		[.28 .67 1.54]	[.5 .81 1.26]	[.27 .67 1.66]
6	[.33 .57 .94]	[.47 .76 1.2]		[.24 .60 1.42]	[.47 .76 1.2]	[.25 .62 1.55]
7	[.49 .81 1.28]	[.53 .81 1.26]		[.5 .83 1.26]	[.31 .73 1.68]	[.28 .69 1.73]
8	[.088 .21 .44]	[.41 .68 1.08]		[.15 .41 1.02]	[.38 .63 1.02]	[.19 .48 1.21]
9	[.24 .45 .78]		[.4 .54 .7]	[.19 .45 .97]	[.38 .63 1.02]	[.2 .49 1.18]

$$F_{1a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{1b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{1c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{2a}(x) = \frac{\frac{0.85}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.25)}}; F_{2b}(x) = \frac{\frac{0.95}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.35)}}; F_{2c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.45)}};$$

$$F_{3a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{3b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{3c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{4a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{4b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{4c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{5a}(x) = \frac{\frac{0.85}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.25)}}; F_{5b}(x) = \frac{\frac{0.95}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.35)}}; F_{5c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.45)}}.$$

Con las cuales aproximamos los valores frontera, usando $\beta = 15$ y η =0.5.

Para las transiciones $t_1, t_2, ..., t_5$ la función argumento $G_i(x); i = 1, .., 5.$ es:

$$\begin{aligned} x_1 &= \alpha(P_1)w_1 + \alpha(P_2)w_2 \\ x_2 &= \alpha(P_3)w_3 + \alpha(P_4)w_4 \\ x_3 &= \alpha(P_5)w_5 + \alpha(P_6)w_6 \\ x_4 &= \alpha(P_7)w_7 + \alpha(P_8)w_8 \\ x_5 &= \alpha(P_9)w_9 + \alpha(P_{10})w_{10}. \end{aligned}$$

Asumiendo que los pesos ideales son números triangulares difusos definidos como:

$$\begin{aligned} w_1^* &= [0.50 \;\; 0.60 \;\; 0.70] & w_6^* &= [0.70 \;\; 0.80 \;\; 0.90] \\ w_2^* &= [0.60 \;\; 0.70 \;\; 0.80] & w_7^* &= [0.45 \;\; 0.55 \;\; 0.65] \\ w_3^* &= [0.45 \;\; 0.55 \;\; 0.65] & w_8^* &= [0.55 \;\; 0.65 \;\; 0.75] \\ w_4^* &= [0.70 \;\; 0.80 \;\; 0.90] & w_9^* &= [0.65 \;\; 0.75 \;\; 0.85] \\ w_5^* &= [0.85 \;\; 0.90 \;\; 0.95] & w_{10}^* &= [0.60 \;\; 0.70 \;\; 0.80]. \end{aligned}$$

Si las entradas $\alpha(P_1)$, $\alpha(P_2), ...,$ $\alpha(P_{10})$ son dadas aleatoriamente, podemos obtener una salida $\alpha(P_{11})$ por medio del algoritmo de razonamiento difuso. Dadas condiciones iniciales para los pesos

$$\begin{aligned} w_1 &= [0.70 \;\; 0.80 \;\; 0.90] & w_6 &= [0.20 \;\; 0.30 \;\; 0.40] \\ w_2 &= [0.10 \;\; 0.20 \;\; 0.30] & w_7 &= [0.80 \;\; 0.90 \;\; 1.0] \\ w_3 &= [0.40 \;\; 0.50 \;\; 0.60] & w_8 &= [0.90 \;\; 0.95 \;\; 1.0] \\ w_4 &= [0.60 \;\; 0.70 \;\; 0.80] & w_9 &= [1.0 \;\; 1.0 \;\; 1.0] \\ w_5 &= [0.05 \;\; 0.10 \;\; 0.15] & w_{10} &= [1.0 \;\; 1.0 \;\; 1.0]. \end{aligned}$$

Calculando el error entre la salida de la red neuronal $\alpha(p_{11})$ y la salida del sistema experto $\Gamma_3(\alpha(p_{11}))$ podemos modificar los pesos para la capa de salida usando la siguiente ley de aprendizaje:

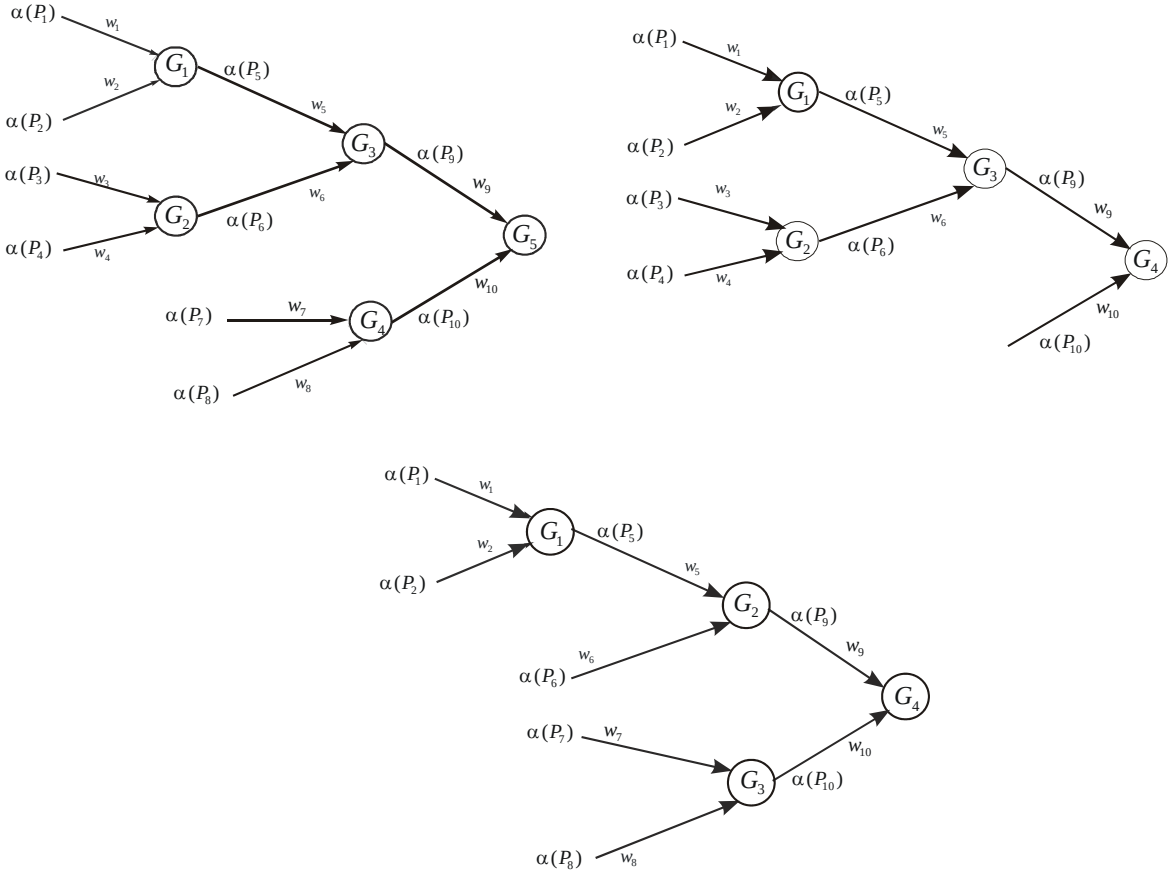


Figura 5.13: Redes neuronales multicapa

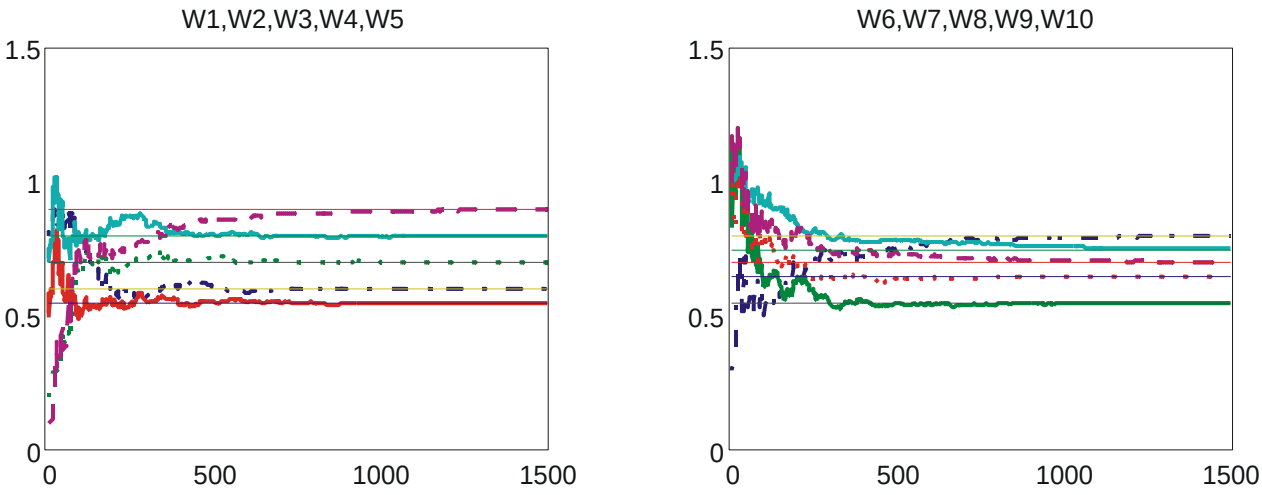


Figura 5.14: Resultados del aprendizaje del Ejemplo 5.3 (Ruta activa T1).

$$\begin{aligned} W_\varphi(k+1) &= W_\varphi(k) + \dot{G}\delta e_\varphi(k)\Lambda_\varphi(P(k)) \\ e_\varphi(k) &: = \alpha(P_{11})(k) - \alpha'(P_{11})(k) \end{aligned}$$

donde δ es el instante de aprendizaje, $W_\varphi(k) = [\varphi_{w_9}(k), \varphi_{w_{10}}(k)]$, $\Lambda_\varphi(P(k)) = [\varphi_{P_9}(k), \varphi_{P_{10}}(k)]$. Después del proceso de entrenamiento ($k > 1300$) los pesos convergen a los valores reales. La figura 5.14 muestra los resultados de la simulación para el parámetro b de cada uno de los pesos.

Para la Ruta Activa T2

Para $G'4$ usamos las sigmoides:

$$F_{1a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{1b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{1c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{2a}(x) = \frac{\frac{0.85}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.25)}}; F_{2b}(x) = \frac{\frac{0.95}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.35)}}; F_{2c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.45)}};$$

$$F_{3a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{3b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{3c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{5a}(x) = \frac{\frac{0.85}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.25)}}; F_{5b}(x) = \frac{\frac{0.95}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.35)}}; F_{5c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.45)}}.$$

Con las cuales aproximamos los valores frontera, usando $\beta = 15$.

Para las transiciones $t_1, t_2, ..., t_5$ la función argumento $G_i(x); i = 1, .., 5.$ es:

$$\begin{aligned} x_1 &= \alpha(P_1)w_1 + \alpha(P_2)w_2 \\ x_2 &= \alpha(P_3)w_3 + \alpha(P_4)w_4 \\ x_3 &= \alpha(P_5)w_5 + \alpha(P_6)w_6 \\ x_4 &= \alpha(P_9)w_9 + \alpha(P_{10})w_{10} \end{aligned}$$

Asumiendo que los pesos ideales son números triangulares difusos definidos como:

$$\begin{aligned} w_1^* &= [0.50 \quad 0.60 \quad 0.70] \quad & w_5^* &= [0.85 \quad 0.90 \quad 0.95] \\ w_2^* &= [0.60 \quad 0.70 \quad 0.80] \quad & w_6^* &= [0.70 \quad 0.80 \quad 0.90] \\ w_3^* &= [0.45 \quad 0.55 \quad 0.65] \quad & w_9^* &= [0.65 \quad 0.75 \quad 0.85] \\ w_4^* &= [0.70 \quad 0.80 \quad 0.90] \quad & w_{10}^* &= [0.60 \quad 0.70 \quad 0.80] \end{aligned}$$

Si las entradas $\alpha(P_1)$, $\alpha(P_2)$, ..., $\alpha(P_6), \alpha(P_9), \alpha(P_{10})$ son dadas aleatoriamente, podemos obtener una salida $\alpha(P_{11})$ por medio del algoritmo de razonamiento difuso. Dadas condiciones iniciales para los pesos

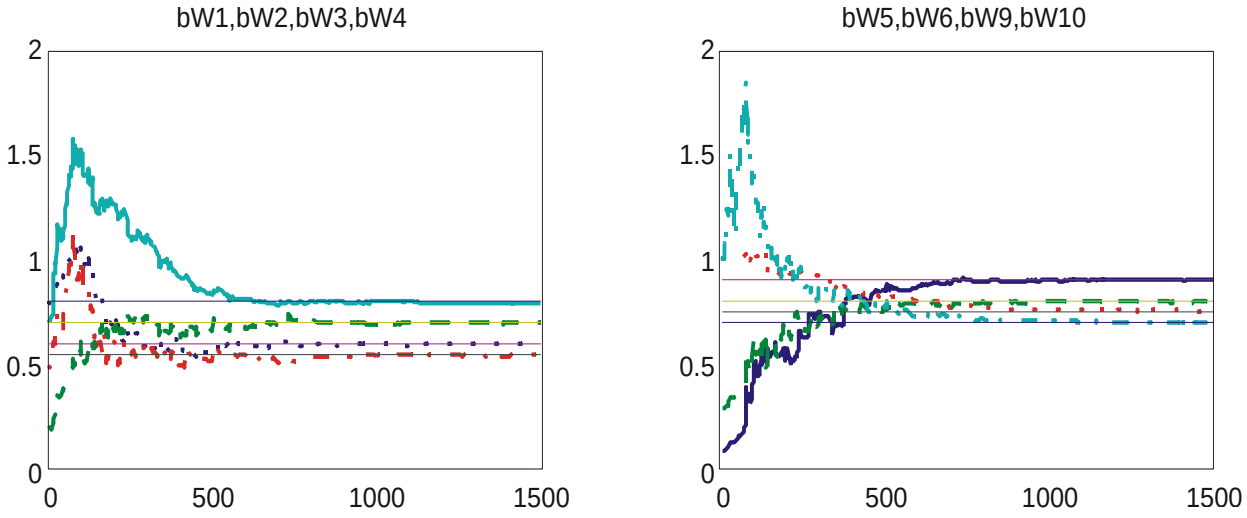


Figura 5.15: Resultados del aprendizaje del Ejemplo 5.3 (Ruta activa T2).

$$
\begin{aligned}
w_1 &= [0.70 \quad 0.80 \quad 0.90] & w_5 &= [0.05 \quad 0.10 \quad 0.15] \\
w_2 &= [0.10 \quad 0.20 \quad 0.30] & w_6 &= [0.20 \quad 0.30 \quad 0.40] \\
w_3 &= [0.40 \quad 0.50 \quad 0.60] & w_9 &= [1.0 \quad 1.0 \quad 1.0] \\
w_4 &= [0.60 \quad 0.70 \quad 0.80] & w_{10} &= [1.0 \quad 1.0 \quad 1.0]
\end{aligned}
$$

Si calculamos el error entre la salida de la red neuronal $\alpha(p_{11})$ y la salida del sistema experto $\Gamma_3(\alpha(p_{11}))$ podemos modificar los pesos usando la siguiente ley de aprendizaje para la capa de salida:

$$
\begin{aligned}
W_\varphi(k+1) &= W_\varphi(k) + \dot{G}\delta e_\varphi(k)\Lambda_\varphi(P(k)) \\
e_\varphi(k) &: = \alpha(P_{11})(k) - \alpha'(P_{11})(k)
\end{aligned}
$$

donde δ es el instante de aprendizaje, $W_\varphi(k) = [\varphi_{w_9}(k), \varphi_{w_{10}}(k)]$, $\Lambda_\varphi(P(k)) = [\varphi_{P_9}(k), \varphi_{P_{10}}(k)]$. Después del proceso de entrenamiento ($k > 1000$) los pesos convergen a los valores reales. La figura 5.15 muestra los resultados de la simulación para el parámetro b de cada uno de los pesos.

Para la Ruta Activa T3

Para $G''4$ usamos las sigmoides:

$$F_{1a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{1b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{1c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{2a}(x) = \frac{\frac{0.85}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.25)}}; F_{2b}(x) = \frac{\frac{0.95}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.35)}}; F_{2c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.45)}};$$

$$F_{3a}(x) = \frac{\frac{0.8}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.31)}}; F_{3b}(x) = \frac{\frac{0.90}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.45)}}; F_{3c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.58)}};$$

$$F_{5a}(x) = \frac{\frac{0.85}{\Sigma_i\ c_{wi}}}{1+e^{-\beta(x-0.25)}}; F_{5b}(x) = \frac{\frac{0.95}{\Sigma_i\ b_{wi}}}{1+e^{-\beta(x-0.35)}}; F_{5c}(x) = \frac{\frac{1.0}{\Sigma_i\ a_{wi}}}{1+e^{-\beta(x-0.45)}}.$$

Para aproximar los valores frontera, usando $\beta = 10$.

Para las transiciones $t_1, t_2, ..., t_4$ la función argumento $G_i(x); i = 1, .., 4.$ es:

$$\begin{aligned} x_1 &= \alpha(P_1)w_1 + \alpha(P_2)w_2 \\ x_3 &= \alpha(P_5)w_5 + \alpha(P_6)w_6 \\ x_2 &= \alpha(P_7)w_7 + \alpha(P_8)w_8 \\ x_4 &= \alpha(P_9)w_9 + \alpha(P_{10})w_{10}. \end{aligned}$$

Asumiendo que los pesos ideales son números triangulares difusos definidos como:

$$\begin{aligned} w_1^* &= [0.50 \ \ 0.60 \ \ 0.70] \quad w_7^* = [0.45 \ \ 0.55 \ \ 0.65] \\ w_2^* &= [0.60 \ \ 0.70 \ \ 0.80] \quad w_8^* = [0.60 \ \ 0.65 \ \ 0.70] \\ w_5^* &= [0.85 \ \ 0.90 \ \ 0.95] \quad w_9^* = [0.65 \ \ 0.75 \ \ 0.85] \\ w_6^* &= [0.70 \ \ 0.80 \ \ 0.90] \quad w_{10}^* = [0.60 \ \ 0.70 \ \ 0.80]. \end{aligned}$$

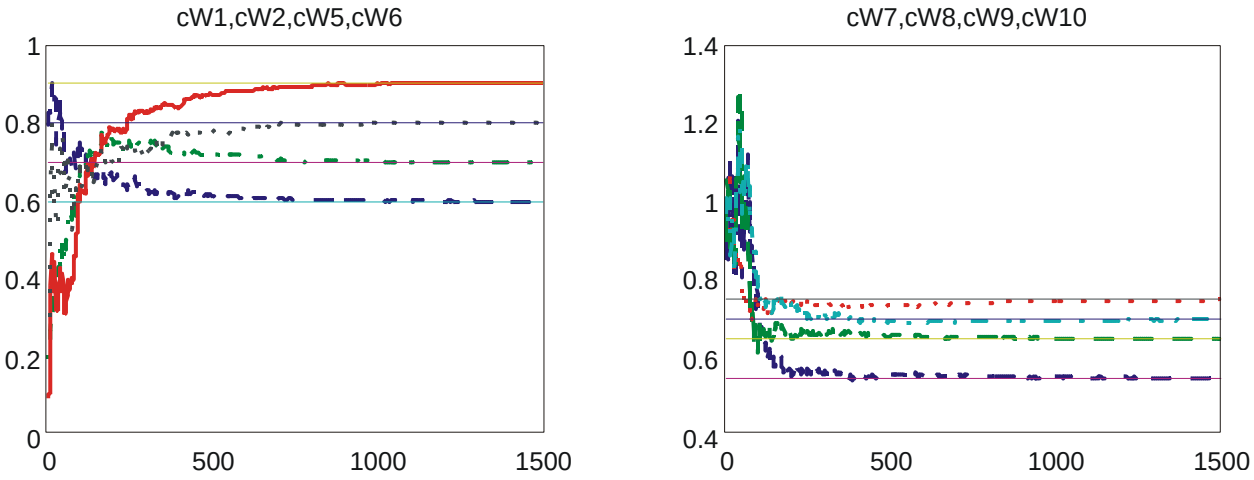


Figura 5.16: Resultados del aprendizaje del Ejemplo 5.3 (Ruta activa T3).

Si las entradas $\alpha(P_1)$, $\alpha(P_2)$, $\alpha(P_5)$, ..., $\alpha(P_{10})$ son dadas aleatoriamente, podemos obtener una salida $\alpha(P_{11})$ por medio del algoritmo de razonamiento difuso. Dadas condiciones iniciales para los pesos

$$\begin{aligned} w_1 &= [0.70 \quad 0.80 \quad 0.90] \quad & w_7 &= [0.80 \quad 0.90 \quad 1.0] \\ w_2 &= [0.10 \quad 0.20 \quad 0.30] \quad & w_8 &= [0.90 \quad 0.95 \quad 1.0] \\ w_5 &= [0.05 \quad 0.10 \quad 0.15] \quad & w_9 &= [1.0 \quad 1.0 \quad 1.0] \\ w_6 &= [0.20 \quad 0.30 \quad 0.40] \quad & w_{10} &= [1.0 \quad 1.0 \quad 1.0]\,. \end{aligned}$$

Si calculamos el error entre las salidas de la red neuronal $\alpha(p_{11})$ y la salida del sistema experto $\Gamma_3(\alpha(p_{11}))$ podemos modificar los pesos usando la siguiente ley de aprendizaje para la capa de salida:

$$\begin{aligned} W_\varphi(k+1) &= W_\varphi(k) + \dot{G}\delta e_\varphi(k)\Lambda_\varphi(P(k)) \\ e_\varphi(k) &: = \alpha(P_{11})(k) - \alpha'(P_{11})(k) \end{aligned}$$

donde δ es el instante de aprendizaje, $W_\varphi(k) = [\varphi_{w_9}(k), \varphi_{w_{10}}(k)]$, $\Lambda_\varphi(P(k)) = [\varphi_{P_9}(k), \varphi_{P_{10}}(k)]$. Después del proceso de entrenamiento ($k > 1100$) los pesos convergen a los valores reales.

La figura 5.15 muestra los resultados de la simulación para el parámetro b de cada uno de los pesos.

5.3. Conclusiones

Se desarrollaron tres ejemplos, el primer ejemplo se mostró el aprendizaje de pesos mediante el algoritmo de Widrow-Hoff los dos ejemplos restantes utilizando el algoritmo Backpropagation modificado se mostró que los pesos asociados a las proposiciones convergen a sus valores iniciales.

Capítulo 6

Conclusiones y Trabajo Futuro

6.1. Contribución de la Tesis

En el *Capitulo 3* se presentaron diferentes algoritmos para representación de conocimiento y razonamiento mediante redes de Petri difusas, los cuales se describen a continuación:

1. El primero es un algoritmo de razonamiento difuso hacia atrás, el cual fue originalmente propuesto en [5] sin embargo, las técnicas de evaluación empleadas por Chen pueden llegar a dar resultados inconclusos, ya que el grado de verdad de las proposiciones consecuentes es evaluado multiplicando el grado de verdad máx mín de las proposiciones antecedentes (iniciales) por el factor de certeza de la regla. El producto puede llegar a ser muy pequeño cuando existen varios niveles de razonamiento. Por lo cual, cuando se compara un valor de verdad consecuente con su valor frontera, el valor de verdad puede llegar a ser más pequeño que el valor frontera y entonces provocar un resultado inconcluso. Además, en la RPD propuesta por Chen las reglas de producción no tienen asociados pesos a sus proposiciones y los valores de verdad, factores de certeza y valores frontera son valores reales en [0,1] y no números difusos.

   Mediante el algoritmo propuesto el valor de verdad de cualquier lugar objetivo puede ser evaluado, elaborando un árbol de nodos con los lugares alcanzables e inmediatamente

alcanzables hacia atrás partiendo del lugar objetivo hasta llegar a un conjunto de lugares iniciales. Una vez construido el árbol de nodos se pide al usuario introduzca los valores de verdad de los lugares iniciales y con ellos se calcula el valor de verdad del lugar objetivo, a diferencia del algoritmo propuesto en [5], los valores de verdad, factores de certeza y valores frontera son números difusos triangulares. Aunado a esto, se asocian pesos a las proposiciones los cuales también son números difusos triangulares.

2. El segundo algoritmo propuesto es una modificación de [6], el cual, fue inicialmente propuesto por el mismo autor en [7], en este algoritmo el valor de verdad de un lugar objetivo puede ser evaluado a partir de un lugar inicial, siempre y cuando el lugar objetivo sea alcanzable a partir del lugar inicial. La diferencia entre este algoritmo y el propuesto radica en las reglas de producción difusas, específicamente para los Tipos 2 y 4. Chen calcula el valor de verdad para estos casos como el máximo de todas las transiciones disparadas. En este algoritmo propuesto se calcula el valor de verdad mediante el centro de gravedad de las transiciones disparadas.

3. Por ultimo el tercer algoritmo presentado en esta sección es una extensión del presentado en [42] con la diferencia de que los valores de verdad, factores de certeza, pesos y valores frontera son valores reales en [0,1] y no números difusos. El algoritmo presentado trabaja con números difusos triangulares.

En el *Capitulo 4* analizamos el aprendizaje con redes de Petri difusas con pesos adaptativas (RPDA), estas fueron originalmente definidas en [42]. Sin embargo, los valores de verdad y pesos asociados a los lugares, así como los factores de certeza y valores frontera asociados a las transiciones son restringidos a valores reales en [0,1]. En el algoritmo que proponemos, utilizamos números difusos triangulares y nos basamos en las definiciones básicas de conjuntos difusos y operaciones con números difusos triangulares definidas en [20],[43] y [16] para realizar el proceso de razonamiento. El proceso de aprendizaje de pesos de la red de Petri difusa propuesta se desarrolla mediante dos algoritmos de aprendizaje. El algoritmo de aprendizaje de Widrow-Hoff para redes de una sola capa y el algoritmo de retropropagación (backpropagation) para redes multicapa.

6.2. Trabajo Futuro

Como perspectivas de trabajo consideramos los siguientes puntos:

- Por medio de los ejemplos presentados se observo que el algoritmo backpropagation es muy efectivo. Ya que, después del proceso de entrenamiento, los pesos del sistema experto convergen a sus valores reales. Sin embargo, cuando la complejidad de la red de Petri aumenta, el número de iteraciones para entrenar nuestra red se incrementa y el tiempo de aprendizaje se eleva exponencialmente. Con el propósito de disminuir el tiempo de aprendizaje se puede proponer un algoritmo de aprendizaje en el cual el parámetro de aprendizaje no sea fijo, sino que cambie dependiendo de la complejidad de la ruta en la red de Petri o de la iteración que se este evaluando.

- Otra alternativa interesante de investigación es considerar el proceso de razonamiento y aprendizaje con funciones gausianas en lugar de números difusos triangulares.

- Todo este panorama muestra una apertura interesante hacia el posterior trabajo de predicción del comportamiento de sistemas expertos usando RPDA.

Bibliografía

[1] Alla, H. y David, R., "Continuous and hybrid Petri nets", *Journal of Circuits, Systems, and Computers*, 8(1), 159–188, 1998.

[2] Berthomieu,B. y Diaz, M., "Modeling and Verification of Time Dependent Systems Using Time Petri Nets," *IEEE Transactions on Software Engineering*, Vol. 17, No. 3, pp 259-273, Marzo 1991.

[3] Bugarin A.J. y Barro S., "Fuzzy Reasoning Supported by Petri Nets," *IEEE Trans. Fuzzy Systems*, pp. 135-150, 2(2), 1994

[4] Carpenter, G. and Grossberg, S. . "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing*, 37:54-115, 1987.

[5] Chen, S.M."Fuzzy Backward Reasoning Using Fuzzy Petri Nets", *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, Vol 30, No. 6, Dec. 2000.

[6] Chen, Shyi Ming, "Weighted Fuzzy Reasoning Using Weighted Fuzzy Petri Nets", *IEEE Transactions on Knowledge and Data Engineering*, Vol 14. No 2, March/April 2002.

[7] Chen, S.M., Ke, J.S. y Chang, J.F. , "Knowledge Representation Using Fuzzy Petri Nets", *IEEE Transactions on Knowledge and Data Engineering*, Vol 2, No. 3, Sept. 1990.

[8] Finkel, A. ,"The minimal coverability graph for Petri nets.", In G. Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674, *Lecture Notes in Computer Science*, pp. 210-243. Springer-Verlag, Berlin, 1993.

[9] Florin, G., Fraize,C. y Natkin, S. , "Stochastic Petri Nets: Properties, Applications and Tools", *Microelectronics and Reliability*, No.4, Vol 31, pp. 669-697, 1991.

[10] Fukushima, K. . "Cognitron: a self-organizing multilayered neural network", *Biological Cybernetics*, 20:121-136, 1975.

[11] Grossberg, S. "Competitive learning: from interactive activation to adaptive resonance", *Cognitive Science*, 11:23-63, 1987.

[12] Grossberg, S. "Nonlinear neural networks: Principles, mechanisms, and architectures", *Neural Networks*, 1:17-61, 1988.

[13] Haykin, S., *Neural Networks, A comprehensive Foundation*, Second Edition, Prentice Hall, 1999.

[14] Hebb, D.O. . "*The Organization of behaviour*". John Wiley, 1949.

[15] Hilera, Jose, Martinez, Victor, "*Redes Neuronales Artificiales, Fundamentos, modelos y applicaciones*", Addison-Wesley lberoamericana, Ra-Ma, 1995.

[16] Jang, J.S.R., Sun, C.T., Mizutani, E., "*Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Machine Intelligence*". Prentice Hall, New York, 1997.

[17] Jensen,Kurt, "Coloured Petri Nets: Central Models and Their Properties," *Advances in Petri Nets* 1986, Part I, *LNCS* Vol. 254, Springer Verlag, 1987.

[18] Jensen,Kurt, "*Coloured Petri Nets*", Springer Verlag, 1996.

[19] Kan, Chieh-Ying y He, Xudong, "A Method for Constructing Algebraic Petri Nets", *Journal of systems Software*,35:15-27, 1996

[20] Kaufmann, A. and Gupta, M.,"*Fuzzy Mathematical Models in Engineering and Management Science*", New York: Elsevier, 1988.

[21] Lin, Chin Teng y George Lee, C.S.,"*A Neuro Fuzzy Sinergism To Intelligent Systems*", Prentice Hall PTR, Upper Saddle River, New Jersey, 1996.

[22] Looney, C.G. "Fuzzy Petri Nets for Rule-Based Decision Making," *IEEE Trans. Syst. Man Cybern.*, Vol SMC-18, pp. 178-183,1988.

[23] Looney, C.G. "Fuzzy Petri Nets and Applications," *Fuzzy Reasoning and Information, Decision and Control Systems*, S. G. Tzafestas and A. N. Venetsanopoulos, Eds. Norwell, Ma: Kluwer, 1994.

[24] Mandler, G., "Recognizing: The judgment of prior occurrence", *Psychological Review*, 87, 252-271, 1980.

[25] Minsky, M. and Papert, S. . *Perceptrons*. MIT Press, 1969.

[26] Murata, Tadao, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, No. 4, Vol. 77, pp. 541-580, 1989.

[27] Newell, A., & Simon, H. A., *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[28] Negoita, C.V., "*Expert Systems and Fuzzy Systems*". Massachussetts: Bejamin/Cummings, 1985

[29] Parker, D.B. *Learning-logic*. Technical Report 581-64, Once of Technology Licensing, Stanford University, 1982.

[30] Pedricz, W. y Gomide,F., "A Generalized Fuzzy Petri Net Model," *IEEE Trans. Fuzzy Systems*, Vol. 2, No 4, pp. 295-301, 1994.

[31] Pedrycz,W., "Fuzzy multimodels," *IEEE Trans. on Fuzzy Systems*, 4, pp.139-148. 1996.

[32] Peterson, J. L., "*Petri Net Theory and the Modeling of Systems*", Prentice Hall, 1981

[33] Petri, C.A., "*Kommunikation mit Automaten*". Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. También, Traducción al Ingles, "*Communication with Automata.*" New York: Griffiss Air Force Base. Tech. Rep. RADC-TR-65-377, vol. 1, Suppl. 1, 1966.

[34] Ramamoorty, C. y Ho, G., "Performance Evaluation of Asynchronous Concurrent Systems Using Petri nets", *IEEE Transaction of software engineering*, SE-6(5), 440-449, 1980

[35] Rosenblatt, F. "*Principles of Neurodynamics*". Spartan Books, 1962.

[36] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. ,"*Learning representations by back-propagating errors*". Nature, 323:533-536, 1986.

[37] Silva, Manuel, "*Las Redes de Petri en la Automatica y la Informatica*", Editorial AC, 1985.

[38] Scarpelli H. y Gomide S.,"A High-Level Fuzzy Net Approach for Discovering Potential Inconsistencies in Fuzzy Knowledge Bases," *Fuzzy Sets and Systems*, pp. 175-193, Vol. 19, 1994.

[39] Werbos, P. . "*Beyond regression: New tools for prediction and analysis in the behavioural sciences*". PhD thesis, Harvard University, Cambridge, MA., 1974.

[40] Widrow, B. y M.E. Hoff, Jr."Adaptive Switching Circuits," *Proc. 1960 IRE West. Elect. Show Conv. Rec.*, Part 4, 96-104, New York. 1960.

[41] Xiaoou Li y Felipe Lara Rosano,"Adaptive Fuzzy Petri Nets for Dynamic Knowledge Representation and Inference," *Expert Syst. Applicat.*, Vol 19, no. 3, 2000.

[42] Xiaoou Li, Wen Yu y Felipe Lara Rosano, "Dynamic Knowledge Inference and Learning Under Adaptive Fuzzy Petri Net Framework", *IEEE Transactions on Systems and cybernetics-Part C*, Vol. 30, No. 4, Noviembre 2000.

[43] Zadeh L.A.,"Fuzzy Sets as a Basis For a Theory of Possibility", *Fuzzy Sets Systems* 1(1):3-28, 1978.

[44] Zadeh L.A.,"Knowledge representation in fuzzy logic", IEEE Trans. on Knowledge and Data Engineering 1, 89-100, 1989.

[45] Zadeh L.A.,"Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Trans. Syst. Man Cybern.*, 3(1): pp.28-44, Enero 1973.

[46] Zadeh L.A.,"Fuzzy Sets", *Information and Control.*, 8:338-353, 1965.

[47] Yeung, D. S. and Tsang, E. C. C., "Fuzzy Knowledge Representation and Reasoning Using Petri Nets", *Expert Systems Applications*, Vol 7, pp. 182-190, 1994.

[48] Yeung, D. S. and Tsang, E. C. C., "A Multilevel Weighted Fuzzy Reasoning Algorithm for Expert Systems", *IEEE Transactions on Systems, Man, and Cybernetics-Part A*, Vol 28, No. 2, March. 1998.

[49] Zimmermann, H. J., "*Fuzzy Set Theory and its Applications*," Boston, Mass.:Kluwer Nijhoff, 1991.