Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Departamento de Computación

Un enfoque evolutivo para el aprendizaje de conocimiento con redes de Petri difusas

Tesis que presenta

Christian Onassis Sánchez Barreto

para obtener el grado de

Maestro en Ciencias

en Computación

Director de la Tesis

Dra. Xiaoou Li

México, D.F. **Septiembre del 2013**

Índice general

Resumen

Desde hace algún tiempo los problemas más recurrentes con los que se enfrenta la industria son la falta de personas capacitadas en campos muy específicos y el elevado costo de sus servicios. Con esto podemos establecer que uno de los bienes mejor valuado es el conocimiento humano, y con esto la capacidad de tomar decisiones. En la actualidad, con la ayuda de personas especializadas podemos crear un sistema que simule la evaluación de los mismos a través del estudio de cierta situación. A esto de le conoce como *sistema experto*.

Los *sistemas expertos* son máquinas que razonan como un experto lo haría en una cierta especialidad o campo. La función de los sistemas expertos no es la de sustituir a los expertos humanos, sino hacer que su trabajo sea más fácil y rápido de solucionarse.

Debido a que el conocimiento en sistemas expertos es confuso y frecuentemente modificado, estos sistemas se han convertido en sistemas dificiles de entender y de gran tamaño. Por tal motivo es muy importante diseñar un marco de inferencia de conocimiento que sea ajustable según la variación del conocimiento como la cognición humana y el pensamiento.

Las *redes de Petri difusas (Fuzzy Petri Nets, en abreviación FPN)* pueden modelar perfectamente *reglas de producción difusas (Fuzzy Production Rules) con pesos*, FPR es una de las representaciones de conocimiento más importante en muchas aplicaciones, los pesos son asignados a las reglas de producción para darles un grado de importancia, esto es, que algunas reglas serán más importantes que otras por lo que deberán tener mayor prioridad.

Basado en el modelo FPN y un mecanismo de disparo de transiciones de FPN, se puede realizar el razonamiento automáticamente. 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.

Las *redes de Petri difusas adaptativas (Adaptive Fuzzy Petri Nets, en abreviación AFPN)* se propusieron para el razonamiento de conocimiento y el aprendizaje, son adecuadas para el conocimiento dinámico, es decir, los pesos de las AFPN se pueden ajustar de forma dinámica según la actualización del conocimiento.

El *algoritmo Backpropagation* es el algoritmo clásico de aprendizaje. Se utiliza siempre para ajustar los parámetros de las funciones de pertenencia y los pesos de las redes. A pesar del éxito del algoritmo Backpropagation para el aprendizaje, este algoritmo posee una serie de deficiencias como: su gran dependencia de parámetros, puede quedar fácilmente atrapado en mínimos locales y generalmente es muy lento en la resolución de búsqueda y optimización, ya que utiliza pasos infinitesimales para alcanzar las soluciones.

Se propone un método de entrenamiento con *algoritmos evolutivos* llamado *AWEA*, es diseñado con el objetivo de eliminar los problemas que sufre el algoritmo Backpropagation. La idea principal, es desarrollar un método alternativo de aprendizaje, este método no reemplazará a los métodos existentes sino que será una alternativa a tener en cuenta por el diseñador de una red en el momento de seleccionar el algoritmo de aprendizaje.

Los resultados obtenidos en esta tesis muestran que el método propuesto es una alternativa interesante para el aprendizaje. El método propuesto es capaz de adaptarse al modelo AFPN y lograr el aprendizaje con las mismas características que el algoritmo Backpropagation. Al mismo tiempo, evita los problemas generales que este algoritmo tiende a tener, ademas converge más rápido que el algoritmo Backpropagation.

Abstract

For some time, the most recurrent problems industry has faced are the lack of qualified people in very specific fields and the high cost of their services. Therefore, we can state that one of the most valued assets is human knowledge, and, as a result, the ability of making decisions. Nowadays, with the help of skilled people we can create a system that simulates the evaluation of the same through the study of a certain situation.

The expert systems are machines that reason as an expert would do in a certain specialty or field. The function of expert systems is not to substitute human experts, but to make their work easier and faster in problem solving.

Due to the fact that knowledge in expert systems is fuzzy and frequently modified, these systems have become large and difficult. For this reason, it is very important to design a frame inference knowledge that is adjustable according to the knowledge variation such as human cognition and thought.

Fuzzy Petri Nets (FPN) can perfectly model Fuzzy Production Rules (FPR) with weights; FPR is one of the most important representations of knowledge in many applications, the weights are assigned to production rules to give them a degree of importance, this means some rules will be more important than others, thus, they must have higher priority.

Based on the FPN model and an FPN transition firing mechanism, automatic reasoning can be made. However, weights in the former cannot be adjusted to the updating of knowledge; in other words, they have no ability to learn.

Adaptive Fuzzy Petri Nets (AFPN) were proposed for reasoning of knowledge and learning; they are suitable for dynamic knowledge, that is, the weights in the AFPN can be adjusted in a dynamic way according to knowledge updating.

Backpropagation algorithm is the classic learning algorithm. It is always used to adjust the parameters of the membership functions and the networks weights. Despite the success of Backpropagation algorithm in learning, this algorithm possesses a series of deficiencies such as: its great parameters dependence, it can be easily trapped in local minimum and it is generally slow in resolution search and optimization since it uses infinitesimal step sizes to reach solutions.

A training method with evolutionary algorithms called AWEA is proposed; it is designed with the objective of avoiding the problems Backpropagation algorithm presents. The main idea is to develop an alternative learning method; this method does not replace the existing methods but will be an alternative for a network designer to take into account when selecting the learning algorithm.

The outcomes obtained in this thesis evince that the approach proposed is an interesting learning alternative. The approach proposed is able to adapt to the AFPN and achieve learning with the same characteristics as the Backpropagation algorithm. At the same time, it avoids the general problems this algorithm tends to have, besides having a faster convergence than the Backpropagation algorithm.

Agradecimientos

Quiero dar gracias a Dios por darme la oportunidad de existir, por guiarme en todo momento y por ayudarme siempre que lo necesito.

A mis padres, Ma. Paz Barreto Sedeño y Fernando Sánchez Rios, por su amor, cariño y comprensión, por enseñarme a ser una persona de bien, a superarme día con día, por la educación que me han dado y los valores que me han inculcado, por ser siempre mis modelos a seguir, muchas gracias.

A mis hermanos, gracias por todos los momentos que hemos pasado juntos, por el apoyo que me han brindado y porque se que aunque tengamos peleas, siempre estarán cuando los necesite.

Muy especialmente a mi novia Mary, por estar conmigo en las buenas y malas, porque siempre me dio ánimos para seguir adelante y principalmente por su amor incondicional. Nada mejor que graduarme hoy 30 de Septiembre, día especial para nosotros porque hoy cumplimos 7 años de estar juntos, feliz aniversario amor.

A la Dra. Li por toda la ayuda que me brindo, por sus comentarios y sus ideas para el desarrollo de mi tesis, por su amabilidad y su buen humor al atenderme, por su gran apoyo y paciencia para que este trabajo pudiera llegar a su fin.

A Sofi por ser una persona noble y gentil, por su apoyo mostrado hacia mi persona y hacia mis compañeros, por el tiempo que me dedico y por esas buenas platicas que tuvimos.

Agradezco al CONACYT por el apoyo económico brindado durante mis estudios de maestría y al CINVESTAV por brindarme una agradable estancia y por ser esa institución de enorme calidad, de la cual me siento orgulloso de haber pertenecido.

Y finalmente a mis amigos de la maestría, que hicieron de la maestría una etapa importante de mi vida, gracias por esos momentos de celebrar, por esos momentos de viajar y por esos momentos de trabajar, gracias.

Índice de figuras

Capítulo 1

Introducción

Los *sistemas basados en conocimiento* representan un paso adelante de los sistemas de información convencionales al pretender representar funciones cognitivas del ser humano como el aprendizaje y el razonamiento. La composición de los sistemas basados en conocimiento consta de: un mecanismo de aprendizaje, una base de conocimientos, un motor de razonamiento y medios de comunicación hombre-maquina.

Un sistema basado en conocimiento o *sistema experto* es un programa que usa conocimiento y procedimientos de razonamiento para resolver problemas lo suficientemente difíciles como para necesitar de un experto para su solución. Se admite que el conocimiento que se puede extraer de los expertos puede ser incompleto, impreciso e incierto.

Debido a que el conocimiento en sistemas expertos es confuso y frecuentemente modificado, estos sistemas se han convertido en sistemas difíciles de entender y de gran tamaño. Por tal motivo es muy importante diseñar un marco de inferencia de conocimiento que sea ajustable según la variación del conocimiento como la cognición humana y el pensamiento.

Las *redes de Petri (Petri Nets, en abreviación PN)* fueron introducidas en la literatura en la tesis doctoral de Carl Adam Petri como una herramienta para simular las propiedades dinámicas de sistemas complejos mediante modelos gráficos de procesos concurrentes. Las PN proveen un conjunto de herramientas con gran capacidad de descripción y operación, ocupadas para representar reglas difusas.

Las *reglas difusas* son el formalismo más popular para representar conocimiento. Las redes de Petri son usadas para modelar bases de reglas, relacionando simplemente algunos de los elementos y características del formalismo de Petri con los elementos básicos de una base de conocimiento.

Las redes de Petri tienen una calidad inherente en la representación lógica de forma intuitiva y visual; las *redes de Petri difusas (Fuzzy Petri Nets, en abreviación FPN)* toman todas las ventajas de las redes de Petri. Las FPN se introdujeron en la literatura como un modelo para representar sistemas basados en conocimiento. Las FPN se usan para representar conocimiento difuso y razonamiento.

Las FPN pueden representar reglas de producción difusas con pesos perfectamente. Sin embargo, a pesar de que estos modelos son flexibles, los pesos en estas redes pueden no ser ajustados de acuerdo a la actualización del conocimiento. En otras palabras, las FPN no tienen la capacidad de aprender. Las *redes de Petri difusas adaptativas (Adaptive Fuzzy Petri Nets, en abreviación AFPN)* poseen la capacidad de aprender de una manera similar a las redes neuronales.

El *algoritmo Backpropagation* es el algoritmo clásico de aprendizaje. Se utiliza siempre para ajustar los parámetros de las funciones de pertenencia y los pesos de las redes. A pesar del éxito del algoritmo Backpropagation para el aprendizaje, este algoritmo posee una serie de deficiencias como: su gran dependencia de parámetros, puede quedar fácilmente atrapado en mínimos locales y generalmente es muy lento en la resolución de búsqueda y optimización, ya que utiliza pasos infinitesimales para alcanzar las soluciones.

Estos problemas muestran la necesidad de buscar un método alternativo de aprendizaje. Este método no reemplazará los métodos existentes sino que será una alternativa a tener en cuenta por el diseñador de una red en el momento de seleccionar el algoritmo de aprendizaje.

Los *algoritmos evolutivos* no fueron diseñados específicamente para la obtención del aprendizaje de conocimiento, sino como algoritmos de búsqueda global, sin embargo, pueden realizar la extracción de conocimiento y específicamente para procesos de inducción de reglas. Aunque esto no los convierte en los mejores algoritmos de inducción de reglas, puesto que no existe ningún algoritmo que sea mejor en todos los casos.

Una característica importante de los algoritmos evolutivos es que realizan una búsqueda global. El hecho de que trabajen con una población de soluciones candidatas más que una solución individual, junto con el uso de operadores estocásticos, reduce la probabilidad de caer en un óptimo local e incrementa la probabilidad de encontrar el máximo global.

Es importante destacar las diversas ventajas que presenta el uso de técnicas evolutivas: simplicidad conceptual, amplia aplicabilidad, superiores a las técnicas tradicionales en muchos problemas del mundo real, tienen el potencial para incorporar conocimiento sobre el dominio y para hibridizarse con otras técnicas de búsqueda/optimización, son robustas a los cambios dinámicos, generalmente pueden auto-adaptar sus parámetros, capaces de resolver problemas para los cuales no se conoce solución alguna, etc.

Se propone un método de entrenamiento con algoritmos evolutivos llamado *AWEA*, es diseñado con el objetivo de eliminar los problemas que sufre el algoritmo Backpropagation. Los resultados obtenidos en esta tesis muestran que el método propuesto es una alternativa interesante para el aprendizaje.

El objetivo principal de este trabajo es la combinación de dos técnicas utilizadas en la construcción de sistemas inteligentes: el modelo AFPN y algoritmos evolutivos. El modelo AFPN empleado para representar y construir una base de reglas y los algoritmos evolutivos para lograr el aprendizaje. Finalmente, basados en la forma en que la AFPN combinada con el AWEA aprende, podamos describir un algoritmo de aprendizaje de los sistemas basados en conocimiento.

1.1. Estado del arte

En esta sección se mencionan algunas técnicas capaces de lograr el aprendizaje. Para empezar, debemos saber que existen algunas extensiones de las redes de Petri utilizadas para la modelación de sistemas complejos, tal es el caso de: *Redes de Petri coloreadas difusas (FCPN)* [1], sus principales ventajas son, la realización del modelado en espacios más pequeños, estructuras de datos más compactos y el procesamiento de información difusa; en sistemas complejos o sistemas basados en reglas, estas ventajas son evidentes. La *red de Petri generalizada asociativa (APN)* [2], APN tiene los mismos inconvenientes que la red de Petri generalizada, los cuales han sido resueltos por extensiones de las mismas, como las redes de Petri coloreadas o las redes de Petri de alto nivel, para el caso de las APN estos problemas se tratarán de resolver desarrollando una estructura jerárquica y de alto nivel.

También existen redes de Petri con la capacidad de la representación y el razonamiento de reglas de producción, a continuación se muestran algunas: Las *redes de Petri de razonamiento difuso (FRPN)* [3], tienen las ventajas gráficas y matemáticas de las PN ordinarias y aborda cuestiones de representar y razonar reglas que contienen literales negativas, las cuales nadie había abordado antes. FRPN es utilizada en sistemas de diagnóstico de fallas basado en reglas, a través de una herramienta flexible y potente, y de interfaz amigable. Todos estos modelos realizan la representación del conocimiento con base a un sistema de reglas.

Pero las extensiones de redes de Petri que son más importantes para nuestro proyecto de tesis, son las redes de Petri capaces de lograr, no sólo la representación y el razonamiento del conocimiento, sino también el aprendizaje, como son: Las *redes de Petri difusas de alto nivel (HLFPN)* [4] se modelan fácilmente reglas de producción para resolver los problemas de razonamiento. Por lo que las HLFPN ofrecen una mayor flexibilidad, capacidad de aprendizaje puesto que es capaz de modelar tanto estructuras regulares como irregulares, permiten múltiples salidas heterogéneas en caso de que existan, ofrecen una estructura de datos más compacta para las reglas de producción difusas, son capaces de aprender más rápido debido a su reducción estructural. Sin embargo este modelo, sus extensiones, y otros algoritmos

de razonamiento relacionados con el HLFPN aún necesitan seguir siendo estudiados. Las *redes de Petri difusas adaptativas con función triangular (AFPNT)* [5], este modelo surge para la representación del conocimiento y razonamiento, y se basa en el diseño de un marco de conocimientos con inferencia dinámica, que sea ajustable de acuerdo a la variación de conocimiento como la cognición humana y el pensamiento. Se desarrolla un algoritmo de aprendizaje para asegurar la convergencia de los pesos. El problema de este modelo es que aún le falta detalles importantes como, predecir el comportamiento de los sistemas. Las *redes de Petri con aprendizaje difuso (LFPN)* [6], surgen a través del modelo de red basado en el refuerzo de aprendizaje (RL), el cual es aplicado para el control del sistema de un robot. Esta red se centra en la combinación de las FPN con un método de aprendizaje inteligente para la construcción de aprendizaje. El modelo de *redes de Petri con máquinas de aprendizaje (MLPN)* [7], surge de la idea de las *redes neuronales artificiales (RNA)* desarrolladas para sistemas altamente paralelos y distribuidos. Se desarrollaron algoritmos de aprendizaje supervisado y no supervisado con el fin de que sean plenamente entrenables y remediar dificultades de las RNA. MLPN tienen como propósito formar RNA y realizar algunas de sus tarea, tales como, la clasificación de patrones, finalización de patrones, optimización y evaluación de funciones. MLPN es método que ofrece las mismas características que las *redes de Petri difusas de alto nivel (HLFPN)* [4].

Existen otras formas que son utilizadas para la representación del conocimiento, las cuales no incluyen redes de Petri, por eso solo se mencionaran en esta sección para saber que existen y después simplemente no las volveremos a mencionar: También existen formas de representar el conocimiento que no utilizan a las PN, una de las más novedosas y al parecer muy eficiente forma es la del uso de *mapas cognitivos difusos (FCM)* [8], los cuales sirven modelar el comportamiento y el funcionamiento de los sistemas complejos, combinan aspectos de la lógica difusa, redes neuronales, redes semánticas, sistemas expertos y sistemas dinámicos no lineales. Los mapas cognitivos son utilizados eficazmente en la toma de decisiones y en simulaciones de situaciones complejas o de análisis. FCM se utiliza debido a la dificultad de describir todo el sistema por un modelo matemático preciso.

1.2. Motivación

Las *FPN* toman todas las ventajas de las *PN*. Así, el camino de razonamiento de sistemas expertos puede reducirse a simples árboles de poda si se aplican algoritmos de razonamiento basado en *FPN* como un motor de inferencia. Las *FPN* son usadas para representar conocimiento difuso y razonamiento. Ya que, basados en el disparo de las transiciones de las *FPN* se pueden desarrollar algoritmos para inferir conocimiento.

Las redes de Petri difusas pueden representar reglas de producción difusas con pesos perfectamente. 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. Las redes de Petri difusas adaptativas *(AFPN)* poseen la habilidad de aprender de una forma parecida a las redes neuronales.

La programación evolutiva es una abstracción de la evolución en la cual la inteligencia se ve como un comportamiento adaptativo. Su principal característica es el aprendizaje de máquina. Las principales ventajas de las técnicas evolutivas son: la simplicidad conceptual, amplia aplicabilidad, superiores a las técnicas tradicionales en muchos problemas del mundo real, tienen el potencial para incorporar conocimiento sobre el dominio y para hibridizarse con otras técnicas de búsqueda/optimización, pueden explotar fácilmente las arquitecturas en paralelo, son robustas a los cambios dinámicos, generalmente pueden auto-adaptar sus parámetros y son capaces de resolver problemas para los cuales no se conoces solución alguna.

1.3. Planteamiento del problema

Las redes de Petri difusas (FPN) pueden representar reglas de producción difusas con pesos perfectamente, 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, las FPN no tienen la capacidad de aprender. Las redes de Petri difusas adaptativas (AFPN) poseen la habilidad de aprender de una forma parecida a las redes neuronales.

Los algoritmos de aprendizaje de gradiente descendente y el algoritmo Backpropagation, se usan siempre para realizar el aprendizaje. Generalmente debemos hacer algunas modificaciones a estos algoritmos para que el proceso de aprendizaje sea estable. Sin embargo, los algoritmos de gradiente descendente y el algoritmo Backpropagation poseen dos problemas. Primero, suelen quedar atrapados en mínimos locales, generándose de esta manera estimaciones subóptimas de los pesos. Segundo, suelen ser muy lentos para alcanzar la solución.

En este trabajo de tesis tratamos de resolver el modelado y aprendizaje de una base de reglas con redes de Petri difusas adaptativas.

1.4. Contribuciones

De las contribuciones que ofrece en el desarrollo de la presente tesis considero como la más importante: Un modelo extendido de redes de Petri difusas adaptativo, que permita el razonamiento y aprendizaje de conocimiento, y en el cual sea posible realizar la modelación, análisis, simulación y ejecución de sistemas de conocimiento representado por reglas de producción, utilizadas en un mismo ambiente de operación.

Las contribuciones secundarias de nuestro trabajo serían:

- Implementación de un algoritmo de razonamiento difuso con el que se pueda obtener el mapeo entre las reglas de producción ponderadas y el modelo AFPN.
- Implementación de un algoritmo evolutivo para el aprendizaje.
- Implementación de un algoritmo de aprendizaje novedoso en los sistemas basados en conocimiento.

[1]

[1]Un artículo enviado a la conferencia IEEE SMC 2013 (*IEEE International Conference on Systems, Man, and Cybernetics*), Manchester, UK. (Aceptado)

1.5. Organización del documento

La tesis esta constituida por siete capítulos, organizados de la siguiente manera:

Capítulo 2. En este capítulo hablamos de los sistemas basados en conocimiento, de las formas de representación de conocimiento más importantes y de la importancia de la adquisición del conocimiento. *Capítulo 3.* En el capítulo 3 se brinda toda la información necesaria sobre las redes de Petri, su definición, estructuras, tipos, propiedades, métodos de análisis y sus extensiones. *Capítulo 4.* En el capítulo 4 se explica el razonamiento difuso, se establecen las reglas de producción difusas para la representación del conocimiento, se presentan las redes de Petri difusas y su manera de representación del conocimiento, de igual forma se describen las redes de Petri difusas adaptativas y como se logra el razonamiento con estas redes. *Capítulo 5.* En este capítulo se explican la manera clásica de realizar el aprendizaje de conocimiento y se explica como se logra el aprendizaje con la nueva propuesta a través de un algoritmo evolutivo. *Capítulo 6.* Finalmente en el ultimo capítulo, se desarrollan simulaciones de algunos ejemplos comparando nuestro algoritmo con el algoritmo Backpropagation.

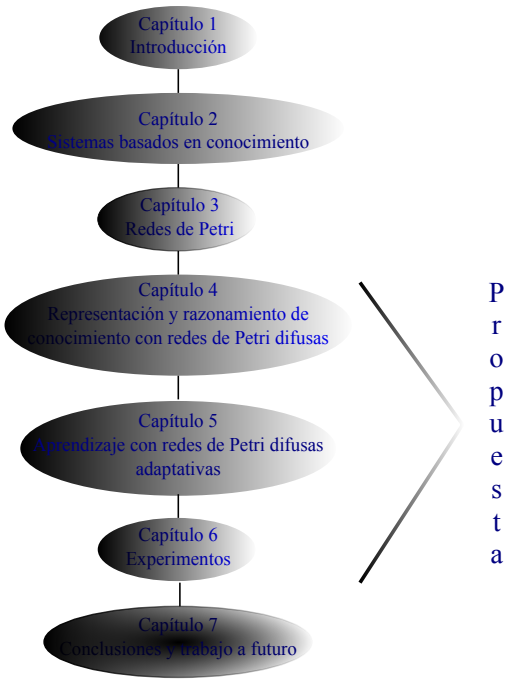

Figura 1.1: *Descripción breve de los capítulos*

Capítulo 2

Sistemas basados en conocimiento

Los sistemas basados en conocimiento van un paso adelante de los sistemas de información convencionales al pretender representar funciones cognitivas del ser humano como el aprendizaje y el razonamiento. Esta clase de aplicaciones descansan en las contribuciones de la inteligencia artificial en general y en la ingeniería del conocimiento en lo particular. Su orientación es la automatización del análisis de problemas, la búsqueda de soluciones, la toma de decisiones y el empleo de conocimiento especializado en un campo específico.

La composición de los sistemas basados en conocimiento consta de: un mecanismo de aprendizaje, una base de conocimientos y de un motor de inferencia. Los sistemas basados en conocimiento se encargan de representar el conocimiento de especialistas de una rama, en la que se procura el aprovechamiento en tareas de diagnóstico, enseñanza y control [9].

A continuación se hace una breve descripción del contenido de este capítulo. En la *sección 2.1* se se definen los sistemas basados en conocimiento, junto con sus principales características, la *sección 2.2* nos introduce un poco a lo que es la representación del conocimiento, las principales formas de representación del conocimiento, así como cual es la que nosotros utilizaremos y finalmente en la *sección 2.3* se explican las fases y métodos para la adquisición del conocimiento.

2.1. Sistema basado en conocimiento o sistema experto

Un *Sistema Basado en Conocimiento* o *Sistema Experto* puede definirse como un sistema informático que simula los procesos de aprendizaje, memorización, razonamiento, comunicación y el comportamiento de un experto humano en una determinada área de especialización dada, suministrando de esta forma, un consultor que pueda sustituirle con unas ciertas garantías de éxito.

Los sistemas expertos permiten el desarrollo de otros sistemas que representan el conocimiento como una serie de reglas. Las distintas relaciones, conexiones y afinidades sobre un tema pueden ser compiladas en un sistema experto pudiendo incluir relaciones altamente complejas y con múltiples interacciones [9].

El contenido de esta sección está distribuido de la siguiente manera. En la *subsección 2.1.1* podemos encontrar las características con las que cuenta un sistema experto en general, para la *subsección 2.1.2* se listan las ventajas de utilizar este tipo de sistemas y en la *subsección 2.1.3* se muestra la arquitectura básica de los sistemas expertos.

2.1.1. Características

Cuando se modelan sistemas expertos se busca que tengan las siguientes características que son propias de los expertos humanos [10]:

- **Habilidad para llegar a una solución de los problemas en forma rápida y certera**. Esta es la habilidad principal que se espera que un experto posea y pueda llevar a cabo. Al mencionar "en forma rápida y certera" obliga a que el experto no solo tenga conocimiento del campo en el que va a trabajar, sino que además tenga experiencia tomando decisiones en él.

- **Habilidad para explicar los resultados a la persona que no cuenta con ese conocimiento**. Esto significa que el experto debe de poder responder en forma clara y certera las preguntas concernientes a las razones de los resultados, el razonamiento derivado de los mismos y las implicaciones subsecuentes. Generalmente las personas que no cuentan con el conocimiento esperan recibir una respuesta más práctica y que se acerque a las condiciones que ellos puedan entender.

- **Habilidad para aprender de las experiencias**. Los expertos deben de aprender tanto de sus propias experiencias como de las experiencias de los demás. Están obligados a estar al día en cuanto a la base de sus conocimientos así como a modificar el proceso de su razonamiento. Los expertos que no se mantienen al día generalmente se vuelven obsoletos.

- **Habilidad de reestructurar el conocimiento para que se adapte al ambiente**. Esto se refiere a que el experto pueda subdividir la base de su conocimiento y usar la porción útil de la misma en la resolución del problema, reduciendo así su tiempo de respuesta. También se refiere a visualizar el problema de distintas perspectivas usando varias porciones del conocimiento y aplicar conocimiento al problema desde distintos niveles.

- **Conciencia de sus limitaciones**. Los expertos pueden evaluar su capacidad para resolver un problema dado y determinar si el mismo se encuentra dentro de sus posibilidades de resolución. Esto también significa que saben cuando referirse a otros expertos.

Las características mencionadas anteriormente le permiten a un sistema experto almacenar datos y conocimiento, sacar conclusiones lógicas, ser capaces de tomar decisiones, aprender, comunicarse con expertos humanos o con otros sistemas expertos, explicar el razonamiento de su decisión y realizar acciones como consecuencia de todo lo anterior.

Aunque también existirán algunas diferencias, las cuales son mostradas a continuación en forma de tabla (ver *cuadro 2.1*):

	Sistema experto	Sistema humano
Conocimiento	Adquirido	Adquirido + Innato
Adquisición del conocimiento	Teórico	Teórico + Práctico
Campo	Único	Múltiples
Explicación	Siempre	A veces
Limitación de capacidad	Sí	Sí, no valuable
Reproducible	Sí, idéntico	No
Vida	Infinita	Finita

Cuadro 2.1: *Diferencias entre un sistema experto y un experto humano*

2.1.2. Ventajas de los sistemas expertos

Las ventajas que se presentan a continuación son en comparación con los expertos humanos:

- Están siempre disponibles a cualquier hora del día y de la noche.
- Mantienen el humor.
- Pueden duplicarse (lo que permite tener tantos sistemas expertos como se necesiten).
- Pueden situarse en el lugar donde sean necesarios.
- Permiten tener decisiones homogéneas efectuadas según las directrices que se les fijen.
- Son fáciles de reprogramar.
- Pueden perdurar y crecer en el tiempo de forma indefinida.
- Pueden ser consultados por personas u otros sistemas informáticos.

2.1.3. Arquitectura

No existe una estructura de sistema experto común. Sin embargo, la mayoría de los sistemas expertos tienen unos componentes básicos: base de conocimientos, motor de inferencia, base de hechos e interfaz con el usuario. Muchos tienen, además, un módulo de adquisición del conocimiento [11].

La *figura 2.1* muestra la estructura de un sistema experto.

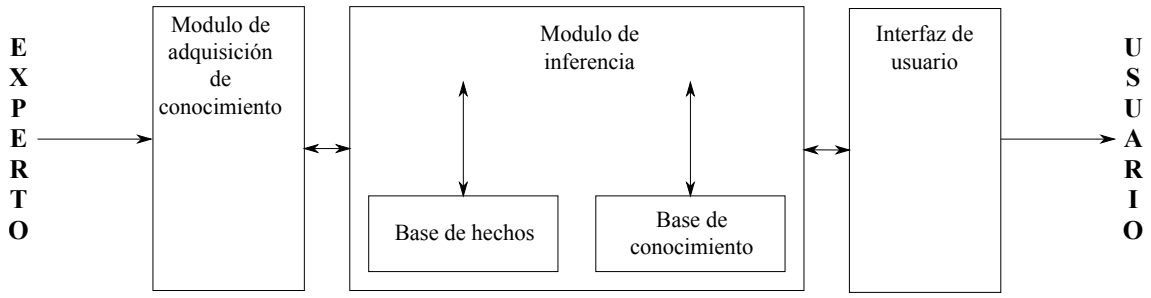

Figura 2.1: *Estructura de un sistema experto*

Base de conocimiento

La base de conocimiento de los sistemas expertos contiene el conocimiento efectivo y heurístico. El conocimiento efectivo es el conocimiento del dominio de la tarea que se comparte ampliamente, encontrado típicamente en libros de texto. El conocimiento heurístico es el conocimiento menos riguroso, más experimental, más critico del funcionamiento.

La base de conocimiento contiene el conocimiento especializado extraído del experto en el dominio. Es decir, contiene conocimiento general sobre el dominio en el que se trabaja. El método más común para representar el conocimiento es mediante reglas de producción.

Una característica muy importante es que la base de conocimientos es independiente del mecanismo de inferencia que se utiliza para resolver los problemas. De esta forma, cuando los conocimientos almacenados se han quedado obsoletos, o cuando se dispone de nuevos conocimientos, es relativamente fácil añadir reglas nuevas, eliminar las antiguas o corregir errores en las existentes. No es necesario reprogramar todo el sistema experto.

Dentro de la base de conocimientos existen tres tipos de conocimientos para construir un sistema experto:

- Reglas.
- Hechos y relaciones entre los componentes.
- Afirmaciones.

Para representar estos tipos de conocimientos en la base de conocimientos, se utilizan tres métodos:

1. **Reglas**. Las reglas son una serie de declaraciones estructurales en forma de oraciones condicionales y están expresadas a través de las estructuras condicionales *Si-Entonces-Sino*. Con ellas se pueden hacer reglas que regulen el resultado de determinada información y proporcionen distintos caminos que nos lleven a un proceso más eficiente. Generalmente son usados en decisiones binarias o más sencillas.

2. **Estructuras**. Las estructuras contienen una jerarquía de componentes y atributos de objetos que pueden ser asignados o heredados de otras estructuras a través de diversos procedimientos. La diferencia entre una estructura y una regla es que una estructura puede representar valores iniciales, apuntadores a otras estructuras o reglas y procedimientos para los cuales se han especificado valores, términos y condiciones de cualquier acción que necesite ser tomada.

3. **Lógica**. Las expresiones lógicas cuentan con predicados, valores y átomos para evaluar hechos del mundo real. El objeto puede ser una constante o una variable que puede cambiar a través del tiempo. Un predicado puede tener uno o más argumentos que son los objetos que describe.

Motor de inferencia

El motor de inferencia es un programa que controla el proceso de razonamiento que seguirá el sistema experto. Utilizando los datos que se le suministran, recorre la base de conocimiento

para alcanzar la solución. La estrategia de control puede ser de encadenamiento progresivo o de encadenamiento regresivo.

Una vez que la base de conocimientos está completa, necesita ser ejecutada por un mecanismo de razonamiento y un control de búsqueda para resolver problemas. El mecanismo de inferencia es la unidad lógica con la que se extraen conclusiones de la base de conocimientos, según un método fijo de solución de problemas que esta configurado imitando el procedimiento humano de los expertos para solucionar problemas. El método más común de razonamiento en los sistemas expertos es la aplicación del *modus ponens*.

Base de hechos

La base de hechos es una parte de la memoria de la computadora que se utiliza para almacenar los datos recibidos inicialmente para la resolución de problemas. Contiene conocimiento sobre el caso concreto en que se trabaja. También se registrarán en ella las conclusiones intermedias y los datos generados en el proceso de inferencia. Al memorizar todos los resultados intermedios, conserva el vestigio de los razonamientos efectuados; por lo tanto, se pueden explicar las deducciones y el comportamiento del sistema.

Módulo de adquisición de conocimiento

El módulo de adquisición del conocimiento permite que se puedan añadir, eliminar o modificar elementos de conocimiento (en la mayoría de los casos reglas) en el sistema experto. Si el entorno es dinámico, entonces este componente es muy necesario, puesto que el sistema funcionará correctamente sólo si se mantiene actualizado su conocimiento. El módulo de adquisición permite efectuar ese mantenimiento, anotando en la base de conocimientos los cambios que se producen.

Las categorías básicas de la investigación en sistemas expertos incluyen: representación del conocimiento, uso del conocimiento (o solución de problemas) y adquisición del conocimiento (es decir, el aprendizaje y descubrimiento del mecanismo).

2.2. Representación del conocimiento

El *conocimiento* es la comprensión adquirida, implica aprendizaje, concienciación y familiaridad con una o más materias; el conocimiento se compone de ideas, conceptos, hechos y figuras, teorías, procedimientos y relaciones entre ellos; y formas de aplicar los procedimientos a la resolución práctica de problemas. El conocimiento que ha de funcionar en un sistema experto es el conocimiento heurístico; el conocimiento heurístico es aquel conocimiento que ayuda a las personas o computadoras a aprender.

La *representación del conocimiento* es un esquema o dispositivo utilizado para capturar los elementos esenciales del dominio de un problema. Una representación manipulable es aquella que facilita la computación. En representaciones manipulables, la información es accesible a otras entidades que usan la representación como parte de un cálculo.

La representación del conocimiento y el razonamiento es un área de la inteligencia artificial cuyo objetivo fundamental es representar el conocimiento de una manera que facilite la inferencia (sacar conclusiones) a partir de dicho conocimiento [12].

El contenido de esta sección es distribuido de la siguiente forma. En la *subsección 2.2.1* se citan las principales forma de representar conocimiento existentes, mientras que en la *subsección 2.2.2* se explica más a detalle una de ellas, las reglas de producción.

2.2.1. Formas para la representación del conocimiento

La forma de representación más usada en los sistemas expertos son las reglas de producción, también llamadas reglas de inferencia. Casi todos los sistemas expertos están basados en este tipo de representación, por lo que en esta tesis se va a utilizar también este tipo de representación de conocimiento, aunque existen otras formas de representación del conocimiento, las cuales no esta demás conocer [12].

Las principales formas para la representación del conocimiento son:

- **Lógica simbólica formal**:

  - *Lógica proposicional.* La lógica proposicional es la más antigua y simple de las formas de lógica. Utilizando una representación primitiva del lenguaje, permite representar y manipular aserciones sobre el mundo que nos rodea. La lógica proposicional permite el razonamiento a través de un mecanismo que primero evalúa sentencias simples y luego sentencias complejas, formadas mediante el uso de conectivos proposicionales, *"AND"*/ *"OR"*.

    Este mecanismo determina la veracidad de una sentencia compleja, analizando los valores de verdad asignados a las sentencias simples que lo conforman. La lógica proposicional permite la asignación de un valor verdadero o falso para la sentencia completa, pero no tiene la facilidad de analizar las palabras individuales que componen la sentencia. La principal debilidad de la lógica proposicional es su limitada habilidad para expresar conocimiento.

  - *Lógica de predicados.* Existen varias sentencias complejas que pierden mucho de su significado cuando se les representa en lógica proposicional. Por esto se desarrolló una forma lógica más general, capaz de representar todos los detalles expresados en las sentencias, esta es la lógica de predicados.

    La lógica de predicados está basada en la idea de que las sentencias realmente expresan relaciones entre objetos, así como también cualidades y atributos de tales objetos. Los objetos pueden ser personas, objetos físicos o conceptos. Tales cualidades, relaciones o atributos, se denominan predicados. Los objetos se conocen como argumentos o términos del predicado.

    Al igual que las proposiciones, los predicados tienen un valor de verdad, pero a diferencia de las proposiciones, su valor de verdad depende de sus términos. Es decir, un predicado puede ser verdadero para un conjunto de términos, pero falso para otro.

- *Reglas de producción.* La regla es la forma más común de representar el conocimiento, debido a su gran sencillez y a que es la formulación más inmediata del principio de causalidad. Una regla consta de un conjuntos de acciones o efectos que son ciertas cuando se cumplen un conjunto de condiciones o causas. La potencia de una regla está en función de la lógica que admita en las expresiones de las condiciones y de las conclusiones.

- **Formas estructuradas**:

  - *Redes asociativas.* Las redes semánticas o redes asociativas, fueron originalmente desarrolladas para representar el significado o semántica de oraciones en inglés, en términos de objetos y relaciones. Actualmente, el término redes asociativas ya no sólo se usa para representar relaciones semánticas, sino también para representar asociaciones físicas o causales entre varios conceptos u objetos.

    Las redes asociativas se caracterizan por representar el conocimiento en forma gráfica. Agrupan una porción de conocimiento en dos partes: objetos y relaciones entre objetos. Los objetos se denominan también nodos (elementos del conocimiento) y las relaciones entre nodos se denominan enlaces o arcos. Cada nodo y cada enlace en una red semántica, deben estar asociados con objetos descriptivos. Estas redes son muy apropiadas para representar conocimiento de naturaleza jerárquica. Su concepción se basa en la asociación de conocimientos que realiza la memoria humana.

  - *Estructuras frame.* Una plantilla (frame) es una estructura de datos apropiada para representar una situación estereotípica. Las plantillas organizan el conocimiento en objetos y eventos que resultan apropiados para situaciones específicas. Una plantilla representa un objeto o situación describiendo la colección de atributos que posee. Cada platilla esta formada por un nombre y por una serie de campos de información o ranuras. Cada ranura puede contener uno o más enlaces. Cada enlace tiene un valor asociado. Varios enlaces pueden ser definidos para cada ranura.

Además los enlaces pueden ser procedimientos que residen en la base de datos y están esperando para ser utilizados cuando se les necesite. A estos procedimientos también se les denomina demons y representan un concepto poderoso en las plantillas, esto es, la habilidad de combinar conocimiento procedimental dentro de la estructura de conocimiento declarativo de la plantilla.

- *Representación orientada a objetos.* Los objetos, son similares a las plantillas. Ambos sirven para agrupar conocimiento asociado, soportan herencia, abstracción y el concepto de procedimientos agregados. La diferencia radica en lo siguiente:

  1. En las plantillas, a los programas y a los datos se les trata como dos entidades relacionadas separadas. En cambio en los objetos se crea una fuerte unidad entre los procedimiento (métodos) y los datos.
  2. Los demons de las platillas sirven sólo para calcular valores para las diversas ranuras o para mantener la integridad de la base de conocimientos cada vez que una acción de alguna plantilla, afecta a otra. En cambio, los métodos utilizados por los objetos son más universales ya que proporcionan cualquier tipo general de cálculo requerido y soportan encapsulamiento y polimorfismo.

  Un objeto es definido como una colección de información que representa una entidad del mundo real y un método es una descripción de cómo debe ser manipulada esta información.

Estas no son las únicas posibilidades de representación de conocimiento. Las representaciones lógicas tienen diversas modalidades como la lógica difusa, la modal, etc. Muchos sistemas utilizan sistemas híbridos que combinan algunas de las aproximaciones anteriores. En cualquier caso, la selección de un formalismo de representación u otro va a condicionar el método de resolución de problemas a utilizar.

Como se mencionó anteriormente, una representación ampliamente usada es la regla de producción, o simplemente regla. En la siguiente sección entraremos más a detalle de lo que son las reglas de producción y de la importancia que estas poseen.

2.2.2. 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* [13].

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 verdadera,

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:

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

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

Cuadro 2.2: *Tabla de verdad*

Entonces tenemos que:

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

de donde tenemos al *cuadro 2.2*. El valor de verdad de las proposiciones p, q de estas reglas de producción únicamente puede tomar valores 0 ó 1.

Las reglas *Si-Entonces* son el formalismo más popular para representar conocimiento. Un conjunto entero de reglas de producción es llamado una base de reglas. Además el formalismo de reglas de producción del 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 del sistema difuso.

2.3. Adquisición del conocimiento

La adquisición de conocimiento es el proceso de recolección de información, a partir de cualquier fuente (experto, libros, revistas, informes), necesaria para construir un sistema basado en conocimiento.

La adquisición de conocimiento no es un paso concreto en la metodología de desarrollo de un sistema basado en conocimiento, sino más bien una tarea que se produce en paralelo a todas las etapas de construcción de estos sistemas (identificación, conceptualización, formalización, implementación, prueba). La adquisición de conocimiento proporciona, a cada etapa, la información que se requiere en cada momento del desarrollo. Por tanto, la recolección de información no se realiza en un único paso aislado; al contrario, forma parte de cada fase.

Seguramente la adquisición de conocimientos es la tarea más importante en el desarrollo de sistemas basados en conocimiento. Sin embargo, esta actividad es de momento un campo experimental en el que la inteligencia artificial tiene poco o nada que decir. No existe ningún método completamente automático de adquisición de conocimiento [14].

A continuación se hace una breve descripción del contenido de esta sección. En la *subsección 2.3.1* se explican cuales son las fases que deben seguirse para adquirir conocimiento y en la *subsección 2.3.2* los métodos que existen para lograr la adquisición del conocimiento.

2.3.1. Fases de adquisición del conocimiento

Dado que la tarea de adquisición del conocimiento es una tarea difícil, se han identificado varias etapas en las que se ha de dividir su desarrollo y así permitir abordar esta labor de una manera más sistemática [14].

Existen diferentes versiones sobre la división de esta tarea; la más aceptada en la literatura es la siguiente:

1. Identificación del problema.
2. Conceptualización.
3. Formalización.
4. Implementación.
5. Prueba.

En la siguiente figura (ver *figura 2.2*) podemos ver un esquema de las fases de adquisición del conocimiento.

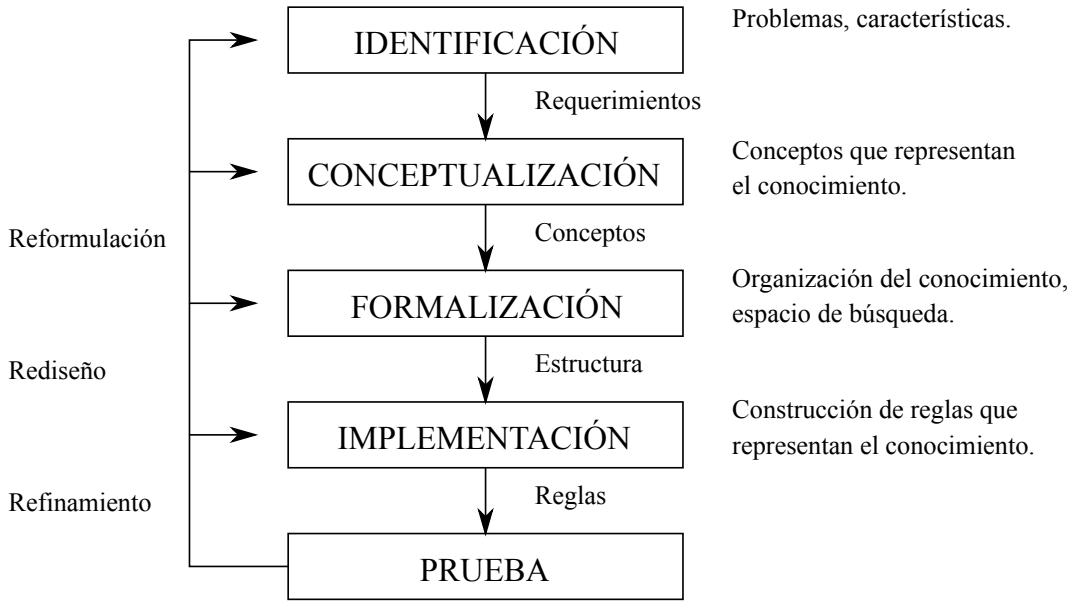

Figura 2.2: *Fases de adquisición del conocimiento*

Identificación

En esta fase se ha de determinar, en primer lugar, si el problema se puede o se debe abordar mediante las técnicas de los sistemas basados en conocimiento. Para que un problema sea adecuado no ha de poder solucionarse de manera algorítmica, ya que si se pudiera de ese modo, no tendría sentido iniciar una labor tan costosa. También ha de ser necesario tener acceso a las fuentes de conocimiento suficientes para completar la tarea. Por último, el problema a tratar ha de tener un tamaño adecuado para que no constituya una tarea inabordable por su complejidad.

El siguiente paso consiste en buscar las fuentes de conocimiento que serán necesarias para el desarrollo del sistema, las más comunes son:

- Expertos humanos en el dominio del problema.
- Libros y manuales que expliquen el problema y técnicas de resolución.
- Ejemplos de casos resueltos.

Éstos últimos serán importantes sobre todo en la última fase de validación, pero se pueden usar también para utilizar técnicas de adquisición automática del conocimiento y obtener de esta manera los elementos básicos que intervienen y sus relaciones.

Con estas fuentes de información se podrán determinar los datos necesarios para la resolución del problema y los criterios que determinan la solución, tanto los pasos que permiten la resolución como su posterior evaluación.

Conceptualización

Antes de entrar en las características globales del problema, es necesario detallar los elementos básicos de éste y descubrir las relaciones entre ellos. En particular, es necesario observar cómo el experto resuelve problemas típicos y abstrae de ellos principios generales que pueden ser aplicados en diferentes contextos.

Hay también que obtener una descomposición del problema en subproblemas, realizando un análisis por refinamientos sucesivos hasta que el ingeniero del conocimiento pueda hacerse una idea de la relación jerárquica de las diferentes fases de resolución hasta los operadores de razonamiento más elementales. Otro elemento necesario es descubrir el flujo de razonamiento en la resolución del problema y especificar cuándo y cómo son necesarios los elementos de conocimiento.

Con esta descomposición jerárquica y el flujo de razonamiento, el ingeniero de conocimiento puede caracterizar los bloques de razonamiento superiores y los principales conceptos que definen el problema. Hará falta distinguir entre evidencias, hipótesis y acciones necesarias en cada uno de los bloques y determinar la dificultad de cada una de las subtareas de resolución. De esta manera se conseguirá captar la estructura del dominio y las diferentes relaciones entre sus elementos.

Formalización

Se han de considerar los diferentes esquemas de razonamiento que se pueden utilizar para modelizar las diferentes necesidades de resolución de problemas identificadas en las fases anteriores.

En este punto, se ha de poder comprender la naturaleza del espacio de búsqueda y el tipo de búsqueda que habrá que hacer. Para ello, se puede comparar ésta con diferentes mecanismos prototípicos de resolución de problemas como la clasificación, abstracción de datos, razonamiento temporal, estructuras causales, etc.

En esta etapa también tendrá que analizarse la certidumbre y completitud de la información disponible , dependencias temporales, o la fiabilidad y consistencia de la información. Se deberá descubrir qué partes del conocimiento constituyen hechos seguros y cuáles no. Para éstos últimos deberá adaptarse alguna metodología de tratamiento de la incertidumbre, de manera que ésta pueda ser modelizada dentro del sistema.

Implementación

En este punto se han de tomar decisiones sobre la especificación del control de la resolución y del flujo de la información. Se deberá tomar decisiones sobre el modo concreto de representar el conocimiento para que se adapte a las estrategias de resolución que se necesiten y las relaciones entre los diferentes conjuntos de conocimiento.

En esta fase se definirán las reglas, e inevitablemente se descubrirán problemas e incompletitudes que obligarán a revisar fases anteriores.

Prueba

Se ha de elegir un conjunto de casos resueltos representativos y se ha de comprobar el funcionamiento del sistema con éstos. En esta fase se descubrirán errores que permitirán corregir análisis anteriores; por lo general aparecerán problemas por falta de reglas, incompletitud, falta de corrección y posibles fallas en el análisis de las reglas pre-establecidas.

2.3.2. Métodos de adquisición del conocimiento

Es necesario distinguir el aprendizaje empírico del sistemático y racional. El primero tiene un carácter espontáneo, cotidiano, particular, relativo y se supone como fuente el uso de los sentidos; el segundo se basa en un procedimiento cuya elaboración es más profunda, incluye a la investigación e incorpora implementación del equipo técnico y especializado.

Es posible considerar al método como una serie de pasos a seguir para lograr el fin, a esto lo podemos remitir a una especie de construcción metodológica en la cual se incluyen procedimientos, técnicas, modelos y teorías que se aplican sistemáticamente durante el desarrollo de una investigación.

Algunos de los métodos más comunes son: el método deductivo, el método inductivo, el método analógico, el método analítico y el método sintético [14].

Método deductivo

Es definido como el método que procede de lo universal a lo particular. Por ejemplo, la lógica y la matemática donde a través de la abstracción sistemática racional y la aplicación de leyes establecidas se llega a la construcción de postulados generales que se consideran científicamente válidos a priori, es decir, independientemente de la experiencia de su aplicación concreta, empírica e inmediata.

Es preciso informar que el uso incorrecto de la deducción puede concluir a que se incurra en conclusiones falsas, aún cuando el razonamiento sea el adecuado. El siguiente es un ejemplo del sistema deductivo correcto y del sistema deductivo incorrecto, vemos que el incorrecto está basado en una teoría mal fundamentada.

Razonamiento deductivo y **correcto**

1. Todo hombre es mortal
2. Sócrates es hombre
3. Por lo tanto: Sócrates es mortal

Razonamiento deductivo e **incorrecto**

1. Todos los gatos son pardos (falso)
2. Tom es un gato
3. Por lo tanto: Tom es pardo

Método inductivo

El método inductivo es aquel método científico que obtiene conclusiones generales a partir de premisas particulares. Se trata del método científico más usual, en el que pueden distinguirse cuatro pasos esenciales: la observación de los hechos para su registro; la clasificación y el estudio de los hechos; la derivación inductiva que parte de los hechos y permite llegar a una generalización.

El razonamiento inductivo puede ser completo (en este caso se acerca a un razonamiento deductivo debido a que sus conclusiones no brindan más datos que los aportados por las premisas) o incompleto (la conclusión trasciende a los datos aportados por la premisa; a medida que hay más datos, habrá una mayor probabilidad de verdad. La verdad de las premisas, de todos modos, no asegura que la conclusión sea verdadera). A continuación un ejemplo del razonamiento inductivo completo comparado con el incompleto:

Razonamiento inductivo y **completo**

1. Kriss y Mary tienen tres perros: Poncho, Hashi y Scooby
2. Poncho es de color negro
3. Hashi es de color negro
4. Scooby es de color negro
5. Por lo tanto, todos los perros de Kriss y Mary son de color negro

Razonamiento inductivo e **incompleto**

1. Poncho es un perro de color negro
2. Hashi es un perro de color negro
3. Scooby es un perro de color negro
4. Por lo tanto, todos los perros son de color negro

Método analógico

Para caracterizar más exactamente el concepto de análogo, deben de considerarse los dos tipos fundamentales de analogías: de atribución y proporcionalidad. En la analogía de atribución se cuenta con un concepto fundamental y otro secundario, uno es considerado como necesario y el otro como contingente, donde el segundo es análogo al primero. En la analogía

de proporcionalidad se conciben igualmente dos conceptos, con cierta discrepancia uno del otro; ambos conceptos difieren de la naturaleza o significado.

Si se considera a la analogía como sinónimo de comparación, el método analógico o comparativo permitirá establecer cierta relación entre dos objetos de aprendizaje o dos conceptos, de modo que a partir del conocimiento del primero se infiera el segundo, ya sea por su atribución o por su proporción.

Método analítico

El método analítico consiste en la operación entre elementos similares muy diferentes, para después sacar una conclusión lógica. Del término análisis significa "separación del todo en sus partes". Cada parte merece un estudio por separado para la comprensión del todo (objeto o concepto). Este tipo de análisis se puede ver al momento de identificar por separado los componentes de un libro:

- *Análisis interno.* Personajes primarios, secundarios y ambientales, trama, clímax, desenlace y moraleja.
- *Análisis externo.* Autor, título, editorial, lugar y año de publicación.

Método sintético

Este método es contrario al de análisis, sin embargo puede considerarse como complementario. Síntesis significa la reunión de las partes de un todo una vez que éstas han sido analizadas y ordenadas. Un ejemplo de síntesis se ve en la química al utilizar dos elementos distintos para formular una molécula única tal es el ejemplo del H_2O (agua) y el CO_2 (bióxido de carbono).

Capítulo 3

Redes de Petri

Las *redes de Petri* son una herramienta gráfica y matemática que ayuda a describir relaciones entre condiciones y eventos presentes en los sistemas del mundo real. Particularmente ideales para modelar características de los sistemas, tales como: operaciones concurrentes, sincronización de procesos, eventos asíncronos, procesos distribuidos, procesos paralelos, procesos no deterministas y/o estocásticos. En este capítulo se presentan aspectos básicos de las redes de Petri que necesitamos conocer [15].

La descripción del contenido de este capítulo viene dada de la siguiente forma. En la *sección 3.1* se da una definición formal de las redes de Petri, después en la *sección 3.2* presentamos cuales son sus componentes principales, la *sección 3.3* nos habla de las reglas básicas de marcado y cuando es que aparece cada una de ellas, la *sección 3.4* nos indica el modo en que se realiza el disparo de una transición, en la *sección 3.5* se presentan los dos tipos de propiedades de las redes de Petri y la definición de cada propiedad, en la *sección 3.6* son listadas las principales ventajas de usar redes de Petri, así como las desventajas de utilizarlas, más adelante en la *sección 3.7* se dan algunos ejemplos de los métodos de análisis y como estos se clasifican, en la *sección 3.8* se da un un ejemplo de como es que se realiza el modelado de una red de Petri, finalmente en la *sección 3.9* se describen las principales extensiones que surgen de las redes de Petri clásicas.

3.1. Definición formal

Una *red de Petri (Petri Net, en abreviación PN)*, es un tipo particular de grafo dirigido bipartito, compuesto por dos tipos de objetos. Estos objetos son *lugares y transiciones* [15].

Una red de Petri, es una estructura algebraica *PN = (P,T,I,O,M0)* donde:

- $P = p_1, p_2, ..., p_m$ es el conjunto finito de lugares.
- $T = t_1, t_2, ..., t_n$ es el conjunto finito de transiciones.
- I: $P \times T \rightarrow \{0, 1\}$ es el conjunto de lugares de entrada a T.
- O: $T \times P \rightarrow \{0, 1\}$ es el conjunto de lugares de salida de T.
- M_0: $P \rightarrow \{0, 1, 2, 3, ...\}$ es el marcado inicial.

3.2. Componentes fundamentales

Lugares y transiciones. Representan eventos concatenados, donde los lugares indican pre y post condiciones en la ejecución de una transición.

Arcos. Indican la dirección de la secuencia de eventos, n arcos igualmente dirigidos, se pueden representar como una flecha simple etiquetada con el número n, que indica su peso.

Tokens. Un token puede ser asignado a un lugar de la PN para denotar el estado del sistema modelado. Los tokens son usados para definir la ejecución de una PN, por lo tanto, la cantidad y la posición de los tokens puede cambiar durante la ejecución.

Para apreciar mejor cada elemento de la PN, se muestra la *figura 3.1:*

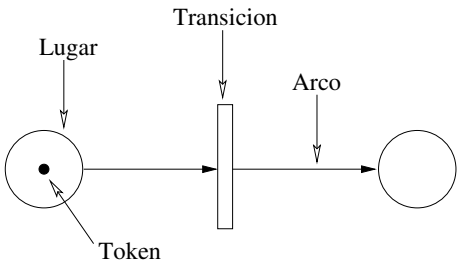

Figura 3.1: *Elementos de una red de Petri*

3.3. Marcado

El marcado es la asignación de tokens a los lugares de una PN. Considera la distribución de tokens en los lugares para cada evento de una secuencia dada.

En una PN el marcado cambia según tres reglas básicas [15]:

- Una transición está habilitada si todos los lugares de entrada están marcados, es decir, si los lugares poseen un número mayor o igual de marcas que el peso de los arcos que los unen a la transición.

- Una transición habilitada puede o no dispararse dependiendo del tipo de transición (según el sistema que represente) y de la ocurrencia del evento asociado.

- El disparo de una transición habilitada consiste en quitar marcas de cada uno de los lugares de entrada (tantas como el peso de los arcos correspondientes) y añadir marcas a cada uno de los lugares de salida (tantas como el peso de los arcos correspondientes).

3.4. Ejecución

La ejecución de una PN es controlada por el número de tokens y su distribución en la red. Los tokens se alojan en los lugares y controlan el disparo de las transiciones que forman la red. Cambiando la distribución de los tokens en los lugares, podremos estudiar el comportamiento dinámico que puede alcanzar el sistema en diferentes estados.

Una PN es ejecutada a través del disparo de transiciones *(ver figura 3.2)*, que se llevan a cabo con la habilitación de la regla y posteriormente se aplica la regla de disparo, las cuales gobiernan el flujo de los tokens por la red [16].

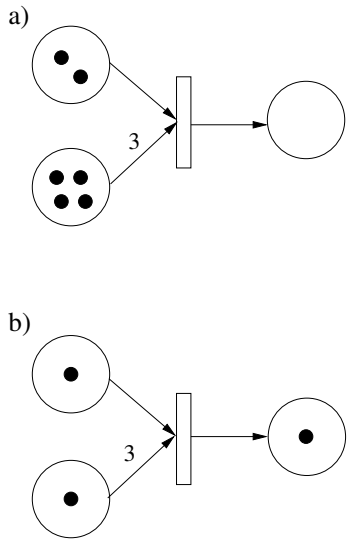

Figura 3.2: *a) Red de Petri antes del disparo.*
b) Red de Petri después del disparo.

3.5. Propiedades

Como una herramienta matemática, las PN presentan algunas propiedades. Estas, cuando son interpretadas en el contexto del sistema modelado, permiten al diseñador del sistema identificar la presencia o ausencia de características funcionales específicas en el dominio de aplicación del sistema que se está diseñando [15].

A continuación se hace una breve descripción del contenido de esta sección. En la *subsección 3.5.1* podemos encontrar propiedades de comportamiento que dependen del marcado inicial. Mientras que en la *subsección 3.5.2* nos encontramos con las propiedades estructurales independientes del marcado inicial.

3.5.1. De comportamiento

- *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 $\alpha = t_1, t_2,...,t_n$.

- *Vivacidad*. Una transición t está viva para un marcado inicial dado M_0, si existe una secuencia de disparos a partir de un marcado M sucesor de M_0 que cumpla con t:

  $\forall M \in M(R, M_0)$, $\exists \alpha$: $M\alpha \rightarrow M_0$ tal que $t \subset \alpha$.

  Una PN marcada está viva para M_0 si 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 (deadlock), ya que, para que este viva, todas sus transiciones deben ser disparables desde cualquier marcado alcanzable.

  Se dice que una PN 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 PN marcada parcialmente viva tiene la posibilidad de evolución global, independientemente de que existan transiciones que no puedan ser disparadas.

- *Ciclicidad*. Se dice que una PN posee comportamiento globalmente cíclico para M_0 si existe una secuencia de disparos que permite alcanzar el marcado inicial M_0 a partir de cualquier marcado M alcanzable a partir de M_0:

  $\forall M \in M(R, M_0)$, $\exists \alpha$ tal que $M\alpha \rightarrow M_0$.

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

- *Acotamiento*. El propósito de esta propiedad es asegurar que el sistema que una red representa posea 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).

Un lugar p es k-acotado para M_0 si 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.

Una PN marcada es k-acotada para M_0 si todos sus lugares son k-acotados para M_0:

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

Se dice que la red está k-acotada si para todo marcado alcanzable tenemos que ningún lugar tiene un número de marcas mayor que k.

- *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. 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 el número de salidas.

3.5.2. Estructurales

- *Pura*. Una PN es una red pura si no existe ninguna transición que tenga un lugar que sea al mismo tiempo de entrada y salida de la transición.

- *Acotada estructuralmente*. Una PN está acotada estructuralmente si está acotada para cualquier marcado inicial finito.

  Un lugar p en una PN se dice no acotado estructuralmente si existe un marcado M y una secuencia de disparos α desde M tal que p no esté acotado.

- *Estructuralmente viva*. Una PN está estructuralmente viva si existe algún marcado inicial para el que esté viva.

- *Completamente controlable*. Una PN se dice completamente controlable si cualquier marcado es alcanzable desde cualquier otro marcado.

- *Estructuralmente conservativa.* Una PN es estructuralmente conservativa si para cualquier marcado inicial M_0 y un marcado alcanzable $M \in R(M_0)$, existe un vector x (nx1) tal que $x_i \neq 0$, para cualquier i = 1, 2,..., n y $x^t M = x^t M_0$.

- *Parcialmente repetitiva.* Una PN es (parcialmente) repetitiva si existe un marcado finito M_0 y una secuencia de disparos tal que (alguna) toda transición ocurra un número infinito de veces en α.

- *Parcialmente consistente.* Una PN es (parcialmente) consistente si existe un marcado finito M_0 y una secuencia de disparos cíclica desde M_0 a M_0 tal que (alguna) toda transición ocurra al menos una vez en α.

3.6. Ventajas y desventajas

Las ventajas de las *redes de Petri* son:

- El sistema completo frecuentemente es fácil de entender debido a la naturaleza gráfica y precisa del esquema de representación.

- El comportamiento del sistema se puede analizar usando la teoría de *Red de Petri*.

- Puesto que las redes de Petri pueden ser sintetizadas usando los enfoques ascendente (bottom-up) y descendente (top-down), es posible especificar sistemáticamente aquellos sistemas cuyo comportamiento es conocido o fácilmente verificable.

Y sus desventajas son las siguientes:

- Las redes de Petri generales no pueden modelar ciertas situaciones de prioridad.

- El problema de alcanzabilidad en las PN es de tiempo exponencial y es un fuerte consumidor de espacio.

3.7. Métodos de análisis

Los métodos para analizar las redes de Petri pueden clasificarse en los siguientes grupos: árbol de alcanzabilidad, matriz de incidencia y técnicas de simplificación [15].

El contenido de esta sección está distribuido de la siguiente manera. En la *subsección 3.7.1* se da la definición del primer método de análisis, el árbol de alcanzabilidad, así como un ejemplo para apreciar su funcionamiento, en la *subsección 3.7.2* se muestra la definición y un ejemplo de la matriz de incidencia y finalmente en la *subsección 3.7.3* se describe la estrategia de simplificación con su respectivo ejemplo para dar un mejor entendimiento.

3.7.1. Árbol de alcanzabilidad

Este método representa el conjunto de todas las marcas alcanzables de la marca inicial M_0. Partiendo del estado inicial M_0, se generan todos los estados alcanzables desde éste mediante el disparo de una transición. A partir de cada estado se vuelve a repetir el proceso, apareciendo, en consecuencia, un grafo en forma de árbol con una estructura infinita.

Para representar esta estructura infinita con un árbol finito, se deja de expandir el árbol cuando se alcanza un marcado frontera (hojas del árbol) como se muestra en la *(figura 3.3)*.

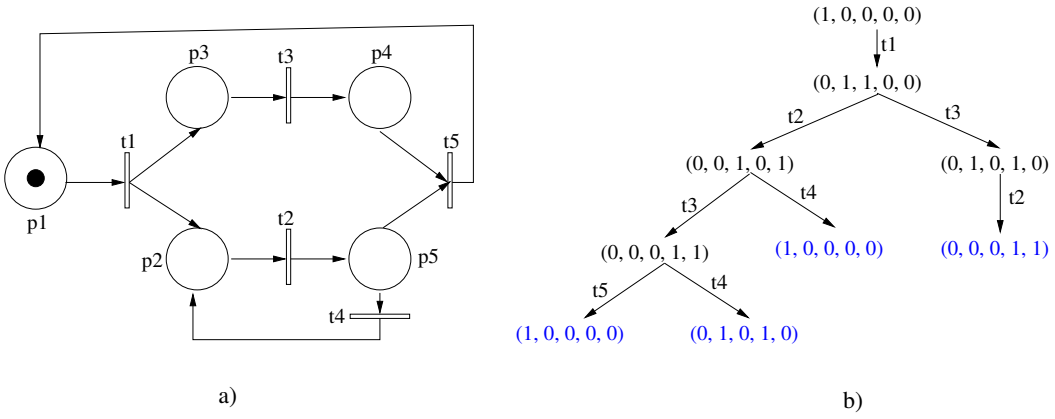

Figura 3.3: *a) Red de Petri.*
b) Árbol de alcanzabilidad

3.7.2. Matriz de incidencia

Una PN con n plazas y m transiciones se representa por dos matrices de incidencia de dimensión $m \times n$ que representan las conexiones entre los lugares de la red *(ver figura 3.4)*.

La matriz de incidencia previa C-: C-(j, i) = I(p_i, t_i).

La matriz de incidencia previa C+: C+(j, i) = O(p_i, t_i).

Se define la matriz de incidencia, C, como C = C+ - C-.

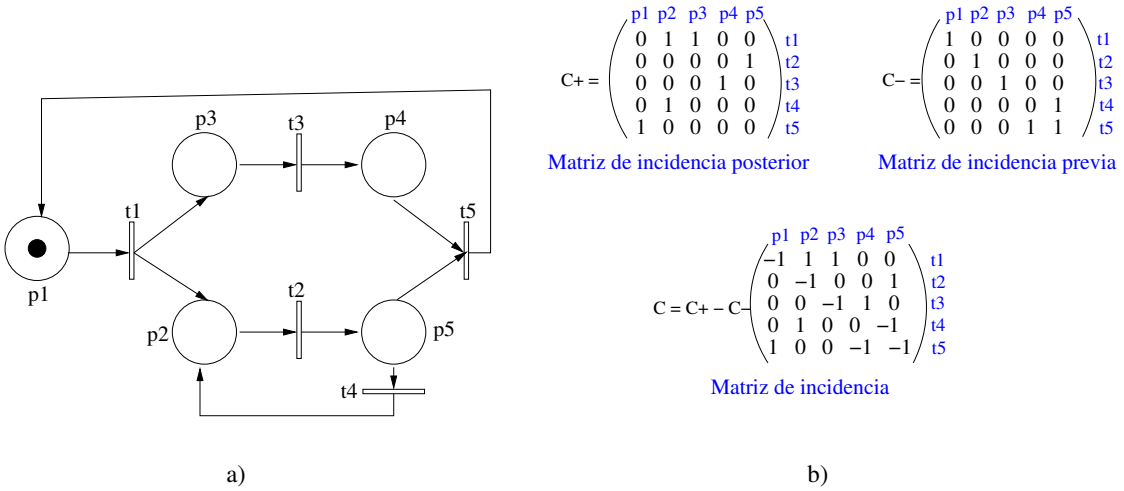

Figura 3.4: *a) Red de Petri.*
b) Matriz de incidencia

3.7.3. Técnica de simplificación

Una estrategia para facilitar el análisis, es reducir el sistema a un modelo más simple el cual pueda conservar sus propiedades de análisis. La *figura 3.5* muestra distintas transformaciones que conservan propiedades de análisis según la especificación en cada caso.

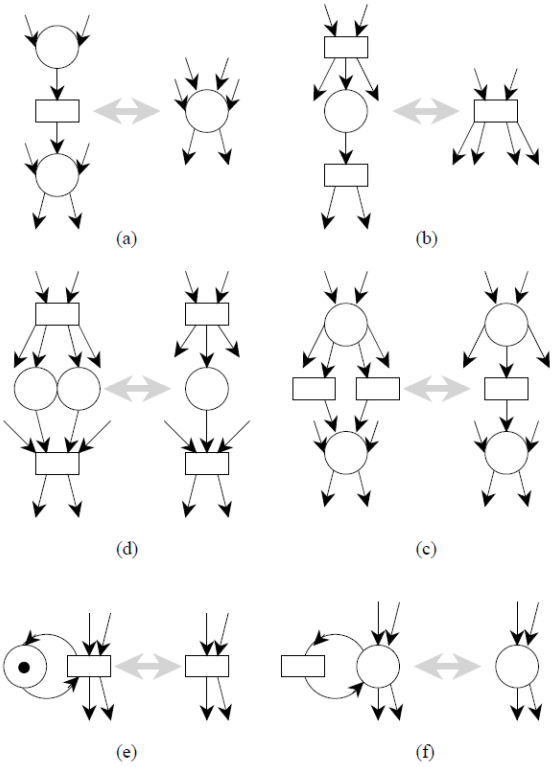

Figura 3.5: *Transformaciones que conservan sus propiedades*

3.8. Modelación con redes de Petri

La construcción de un modelo de PN normalmente se realiza a través del análisis, partiendo de una especificación de requerimientos. El análisis de una PN permite desarrollar el modelo y verificar si éste se encuentra bajo las especificaciones requeridas, por lo que es de suma importancia su estudio.

En primera instancia, el modelado con PN depende de una interpretación del funcionamiento lógico del sistema, que por lo general provee un conjunto de especificaciones de naturaleza informal. En base a esto se desarrolla un bosquejo, que gradualmente se debe complementar, adicionando características, hasta hacer reflejar de manera adecuada el sistema en cuestión.

Para ejemplificar la aplicación del modelado, utilizaremos la siguiente base de reglas, tomada de la tesis doctoral de Joselito Medina Marín [17] y la descripción es la siguiente:

Ejemplo de modelado

Tenemos 3 lugares que representan a los eventos de la base de reglas (ver *figura 3.6*):

$P_0 = PRIMA$,

$P_1 = EMPLEADO$,

$P_2 = VENTAS$.

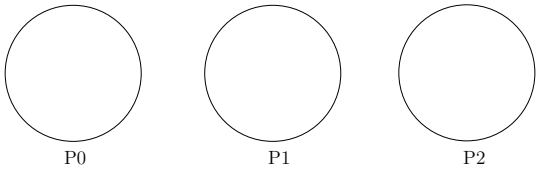

Figura 3.6: *Lugares representados por eventos de la base de reglas*

Regla 1: Cuando la cantidad de la *PRIMA* de un *EMPLEADO* se modifica, si la cantidad es mayor que $100.00, entonces el rango del *EMPLEADO* se incrementa en uno.

Si cantidad de *PRIMA* > 100 *Entonces* actualizar rango del *EMPLEADO* en $+1$. Podemos ver como a través de una condición (transición en la red de Petri) se modela el paso de un estado a otro (ver *figura 3.7*).

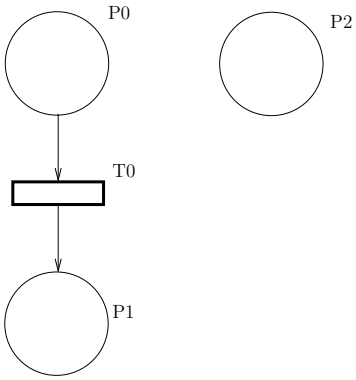

Figura 3.7: *Modelado de la* ***Regla 1***

Regla 2: Cuando el rango de un *EMPLEADO* se modifica, si el rango es mayor que el nivel 5, entonces la *PRIMA* del *EMPLEADO* se incrementa en diez veces el nivel del rango.

Si rango del *EMPLEADO* > 5 *Entonces* actualizar la cantidad de *PRIMA* *10. A través de una nueva regla el modelado nos muestra como se puede regresar al estado anterior (ver *figura 3.8*).

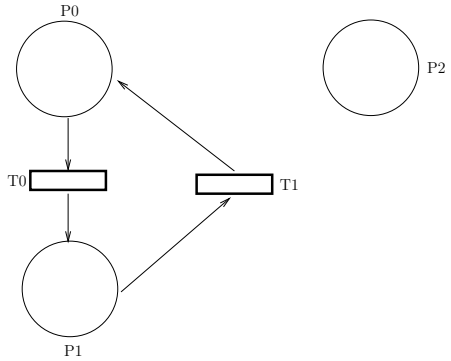

Figura 3.8: *Modelado de la* ***Regla 2***

Regla 3: Cuando se obtienen las *VENTAS* del mes y el número de éstas es superior a 50, entonces el rango del *EMPLEADO* se incrementa en un nivel.

Si número de *VENTAS* > 50 *Entonces* actualizar el rango del *EMPLEADO* +1. Ahora al modelar la nueva regla se observa como ya existe una relación entre los 3 eventos de la base de reglas inicial (ver *figura 3.9*).

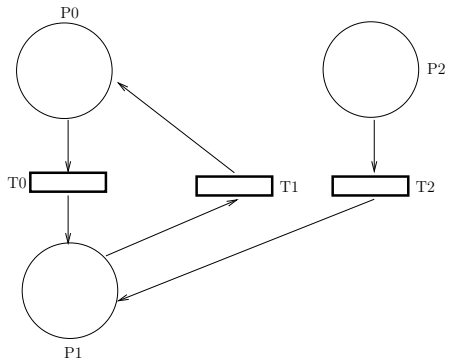

Figura 3.9: *Modelado de la* ***Regla 3***

3.9. Extensiones

Actualmente, se han desarrollado una gran variedad de tipos de redes de Petri (para poder aplicarse a problemas más específicos), existen distintas clasificaciones para ellas, en esta tesis se utiliza la descrita por *Aalst* [18]. En esta clasificación se distinguen las redes de bajo nivel (redes lugar/transición) y las extensiones más comunes (coloreadas, estocásticas y jerárquicas).

Las redes de Petri clásicas permiten modelar estados, eventos y condiciones, sincronizaciones y paralelismo, entre otras características del sistema. Sin embargo, las redes de Petri que describen el mundo real tienden a ser complejas y extremadamente grandes. Mas aún, las redes de Petri clásicas no permiten la modelación de datos y tiempo. Para solucionar este problema, se han propuesto muchas extensiones [16].

Las extensiones se pueden agrupar en las siguientes cuatro categorías:

- Redes de Petri temporizadas.
- Redes de Petri con arcos inhibidores.
- Redes de Petri coloreadas.
- Redes de Petri jerárquicas.

Las *redes de Petri temporizadas* le agregan al modelo el concepto de tiempo y pueden dividirse, a su vez, en dos clases: *redes de Petri de tiempo determinista (redes de Petri regulares)* las cuales incluyen a las redes de Petri que tienen asociadas un tiempo de disparo determinado en sus transiciones, lugares o arcos y *redes de Petri de tiempo estocástico (redes de Petri estocásticas)* que incluyen a las redes de Petri que tienen asociadas un tiempo de disparo estocástico en sus transiciones, lugares o arcos.

Las *redes con arcos inhibidores* pueden chequear si un lugar está vacío, lo que les da la potencia de las máquinas de Turing (una vez contemos con al menos dos de dichos arcos). Una extensión similar es la que viene dada por los *arcos de transferencia*, que mueven todos

los tokens que se encuentren en un lugar a otro, o los *arcos reset* o de puesta a cero, que eliminan todos los tokens que haya en un lugar.

Otras generalizaciones extienden la naturaleza de los tokens que pueden contener los lugares. La más conocida es la de las *redes de Petri coloreadas*. En ellas los tokens dejan de ser indistinguibles, portando un valor de un cierto tipo, de manera que las transiciones no sólo eliminan los tokens de las precondiciones y los crean en las postcondiciones, sino que transforman su valor. Es fácil ver que las redes de Petri coloreadas son Turing completas, ya que una transición podría transformar la entrada de un token precondición según cualquier función recursiva, para producir un token con el correspondiente valor de postcondición.

Las *redes de Petri jerárquicas* proporcionan, como su nombre lo indica, una jerarquía de subredes para tratar el problema de tamaño que se tiene cuando se modelan sistemas reales. De tal forma que la construcción de un sistema grande, se basa en un mecanismo de estructuración de dos o más procesos, representados por subredes.

En general, las redes de Petri pueden verse como una estructura especial para estudiarse como ejercicio intelectual, también se les puede ver como otro autómata capaz de aceptar o generar lenguajes formales; o (la más relevante para nuestro estudio) como un esquema de representación para describir, analizar y sintetizar diferentes sistemas *en tiempo real*.

Capítulo 4

Representación y razonamiento de conocimiento con redes de Petri difusas

Las *reglas difusas* son el formalismo más común para representar conocimiento. Las redes de Petri son utilizadas para modelar bases de reglas, relacionando algunos de sus elementos (función de marcado) y características (lugares y transiciones) con elementos de una base de conocimientos (proposiciones y grados de verdad). Cuando la complejidad de los sistemas aumenta, es necesario incluir nuevos elementos a la definición inicial de red de Petri, y así poder representar los rasgos de la base de conocimientos adecuadamente, este proceso nos lleva a un nuevo modelo llamado red de Petri difusa. Las redes de Petri difusas son usadas para representar conocimiento y razonamiento, ya que basados en el disparo de las transiciones de las redes de Petri difusas, se pueden desarrollar algoritmos para inferir conocimiento [19].

A continuación se hace una breve descripción del contenido de este capítulo. En la *sección 4.1* se explica las dos partes principales de los sistemas de inferencia difusos: el razonamiento difuso y las reglas de producción difusas, para la *sección 4.2* se presentan las redes de Petri difusas y se brinda una muestra como es que estas representan conocimiento, en la *sección 4.3* se introducen las redes de Petri adaptativas las cuales pueden realizar la actualización del conocimiento y de esta manera poder modelar sistemas de conocimientos y finalmente en la *sección 4.4* se da una breve explicación del proceso de razonamiento de conocimiento.

4.1. Razonamiento difuso

El *razonamiento* en un sentido amplio es la facultad humana que permite resolver problemas. En un sentido más restringido, se llama *razonamiento* al proceso mental de realizar una inferencia de una conclusión a partir de un conjunto de reglas. Se observa la dinámica del razonamiento como facultad de la especie humana que le permite entender el medio, usando esa facultad de forma consciente y evolutiva.

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 basada en la teoría de conjuntos difusos. El razonamiento difuso, también conocido como razonamiento aproximado, es un procedimiento de inferencia que deriva conclusiones a partir de un conjunto de reglas difusas *Si-Entonces* y hechos conocidos [20].

El contenido de esta sección está distribuido de la siguiente forma. En la *subsección 4.1.1* se brinda una definición de los sistemas de inferencia difusa, mientras que en la *subsección 4.1.2* se presentan las reglas de producción difusas y se muestra un pequeño ejemplo para su entendimiento.

4.1.1. Sistemas de inferencia difusa

Los sistemas de inferencias difusas son un marco de trabajo de cómputo muy popular basado en los conceptos de teoría de conjuntos difusos, reglas difusas *Si-Entonces* y razonamiento difuso. Esto puede construir aplicaciones exitosas en una extensa variedad de campos, tales como: control automático, clasificación de datos, análisis de decisiones, sistemas expertos difusos, visión por computadora y robótica.

Los sistemas de inferencia difusa tienen una estrecha relación con los conceptos difusos tales como conjuntos difusos, variables lingüísticas y demás. Los sistema de inferencia difusa más populares que se pueden encontrar son: los de tipo Mamdani (con fuzzificador y defuzzificador) y los de tipo Sugeno [20].

Debido a su naturaleza multidisciplinaria, los sistemas de inferencia difusa están asociados con diferentes nombres, tales como sistemas difusos basados en reglas, sistemas expertos difusos, modelado difuso, memoria asociativa difusa, controladores difusos y simples. Los sistemas de inferencias difusas pueden tomar entradas difusas o nítidas, pero las salidas que producen son siempre conjuntos difusos. Algunas veces es necesario tener salidas nítidas, especialmente cuando el sistema de inferencia difusa es usado como un controlador.

Un sistema de inferencia difusa con entradas y salidas nítidas implementa un mapeo no lineal desde su espacio de entrada hacia su espacio de salida. El mapeo es realizado por un conjunto de reglas *Si-Entonces*, cada una de las cuales describe el comportamiento local del mapeo. En particular, el antecedente de una regla define la región difusa del espacio de entrada, mientras que el consecuente especifica la salida en la región difusa.

4.1.2. Reglas de producción difusas

Estas también son llamadas *reglas difusas* (reglas que describen la relación difusa entre dos proposiciones). 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 [21].

Las reglas de producción difusas son usadas para representar conocimiento difuso, impreciso, conceptos vagos y ambiguos. La estructura de las reglas de producción difusas es usualmente presentada en la forma de una regla difusa *Si-Entonces* aunque también se podría representar como una regla especial de la forma *Si todos-Entonces* o *Si alguno-Entonces*, las cuales equivalen a reglas con cláusulas de las condiciones conectadas con *"AND"* y *"OR"* respectivamente.

Las proposiciones pueden contener variables difusas. El grado de 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 a continuación en el *cuadro 4.1.*

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]
Más o menos verdadero	[0.30 0.44]
Minioritariamente verdadero	[0.10 0.29]
Minimamente verdadero	[0.01 0.09]
No verdadero	[0.00 0.00]

Cuadro 4.1: *Escalas de verdad y sus intervalos numéricos*

Sea R un conjunto de reglas difusas $R = R_1, R_2, ..., R_n$. La fórmula general de la regla de producción difusa i_{th} está dada la siguiente manera:

$$R_i\text{: } Si\ d_j\ Entonces\ d_k\ (CF = \mu_i), \lambda$$

donde d_j y d_k son proposiciones las cuales tienen asociadas un valor determinado: grado de verdad (el grado de verdad de cada proposición es un valor real entre cero y uno). λ es el umbral de la regla, μ_i es el valor del factor de certeza (CF), $\mu_i \epsilon [0, 1]$. Representa la veracidad de la regla. Entre más grande el valor más se cree en la regla y $1 \leq i \leq n$.
Por ejemplo, la siguiente es una regla de producción:

$$R_i\text{: } Si \text{ el día es soleado } Entonces \text{ el cielo es azul } (CF = 0.80), 0.20.$$

Sea λ un valor umbral, donde $\lambda \epsilon [0, 1]$, si el grado de verdad de la proposición d_j de la fórmula general es y_i, donde $y_i \epsilon [0, 1]$, entonces:

- *Si* $y_j \geq \lambda$, *Entonces* la fórmula general será disparada. Obteniendo el grado de verdad de la proposición $d_k = y_i{}^* \mu_i$.

- *Si* $y_j < \lambda$, *Entonces* la fórmula general no se dispara.

En el ejemplo previo, si el grado de verdad de la proposición *el día es soleado* es 0.90 y el umbral $\lambda = 0.20$, entonces la regla R_1 puede ser disparada, ya que $0.90 \geq 0.20$. Por lo que el grado de verdad de la proposición *el cielo es azul* es 0.90*0.80 = 0.72. Esto indica que la probabilidad de que el cielo este azul es de 0.72.

4.2. Redes de Petri difusas

Podemos utilizar una *red de Petri difusa (Fuzzy Petri Nets, en abreviación FPN)* para representar las reglas de producción. Una FPN difiere de una red de Petri tradicional, principalmente en que los lugares están marcados con tokens asociados con un valor de verdad entre [0,1] y cada transición está asociada con un factor de certeza también entre [0,1] [22].

La estructura de una *FPN* generalizada se define a continuación. Una FPN, se define como una tupla *FPN = (P,T,I,O,M0,f,α,β)* donde:

- $P = p_1, p_2, ..., p_m$ es el conjunto finito de lugares.
- $T = t_1, t_2, ..., t_n$ es el conjunto finito de transiciones.
- I: $P \times T \rightarrow \{0, 1\}$ es el conjunto de lugares de entrada a T.
- O: $T \times P \rightarrow \{0, 1\}$ es el conjunto de lugares de salida de T.
- M_0: $P \rightarrow \{0, 1, 2, 3, ...\}$ es el marcado inicial.
- f es una función que asigna a cada transición un número difuso en el universo [0,1].
- α es una función de asociación, un mapeo de lugares a valores reales entre cero y uno, α: P$\rightarrow$[0,1].
- β es una función que asocia a un lugar p_i una proposición d_i, β: P$\rightarrow D$.

A continuación se hace una breve descripción del contenido de esta sección. En la *subsección 4.2.1* se describe la representación del conocimiento a través de las redes de Petri difusas, mostrando como estas pueden representar a las reglas de producción difusas.

4.2.1. Representación del conocimiento con redes de Petri difusas

Es posible utilizar un modelo de redes de Petri para representar reglas de producción. 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 [22].

En una FPN, las relaciones de lugares a transiciones y de transiciones a lugares, son representadas por arcos dirigidos. Basados en el ejemplo de la sección anterior, la regla de producción difusa:

$$R_i\text{: } Si\ d_j\ Entonces\ d_k\ (CF = \mu_i)$$

puede ser modelada como se muestra en la *figura 4.1.*

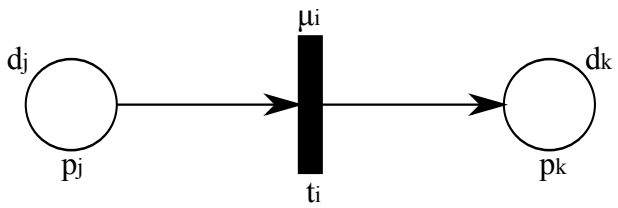

Figura 4.1: *FPN de una regla de producción difusa*

$$FPN = P, T, D, I, O, f, \alpha, \beta,$$
$$P = p_1, p_2,$$
$$T = t_1,$$
$$D = \textit{el día es soleado, el cielo es azul},$$
$$I(t_1) = p_1 \text{ y } O(t_1) = p_2,$$
$$f(t_1) = 0.80,$$
$$\alpha(p_1) = 0.90 \text{ y } \alpha(p_2) = 0.72,$$
$$\beta(p_1) = \textit{el día es soleado} \text{ y } \beta(p_2) = \textit{el cielo es azul}.$$

Una transición t_i está habilitada para disparar, si para todo $p_j \epsilon I(t_i)$, $\alpha(p_j) \geq \lambda$, donde λ es un valor umbral y $\lambda \epsilon [0, 1]$. El valor del token en un lugar de salida t_i es calculado como se muestra en la *figura 4.2.*

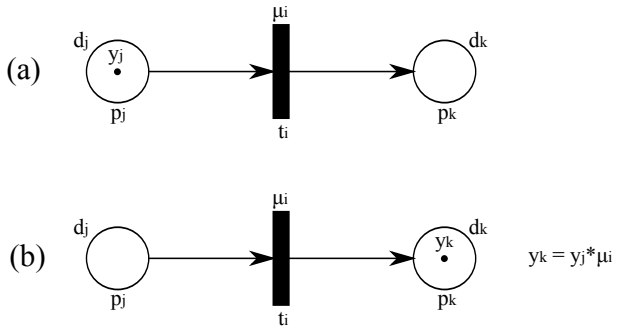

Figura 4.2: *Disparo de una FPN. a) Antes de disparar la transición t_i. b) Después de disparar la transición t_i.*

La regla de producción difusa compuesta puede ser distinguida por cuatro tipos de reglas:

- **Tipo 1:** *Si* d_{j1} AND d_{j2} AND...AND d_{jn} *Entonces* d_k $(CF = \mu_i)$. Este tipo de regla es modelado por una FPN marcada como se muestra en la *figura 4.3*.

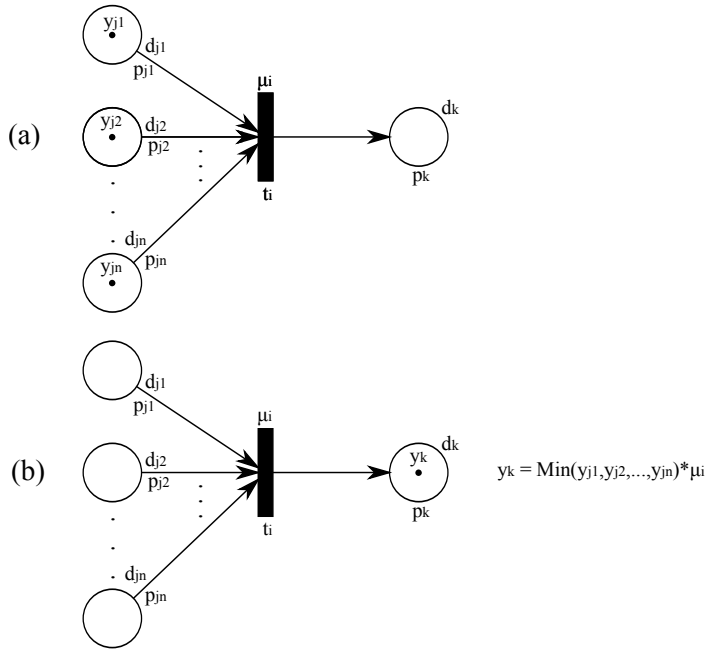

Figura 4.3: *FPN marcada de tipo 1. Antes y después de disparar la transición t_i.*

- **Tipo 2:** *Si* d_j *Entonces* d_{k1} AND d_{k2} AND...AND d_{kn} $(CF = \mu_i)$. Este tipo de regla es modelado por una FPN marcada como se muestra en la *figura 4.4*.

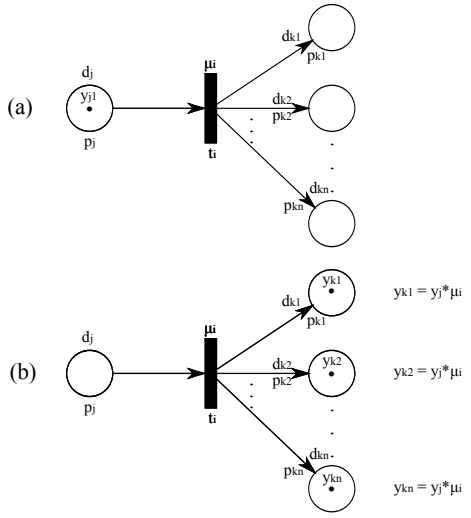

Figura 4.4: *FPN marcada de tipo 2. Antes y después de disparar la transición* t_i.

- **Tipo 3:** *Si* d_{j1} OR d_{j2} OR...OR d_{jn} *Entonces* d_k $(CF = \mu_i)$. Este tipo de regla es modelado por una FPN marcada como se muestra en la *figura 4.5*.

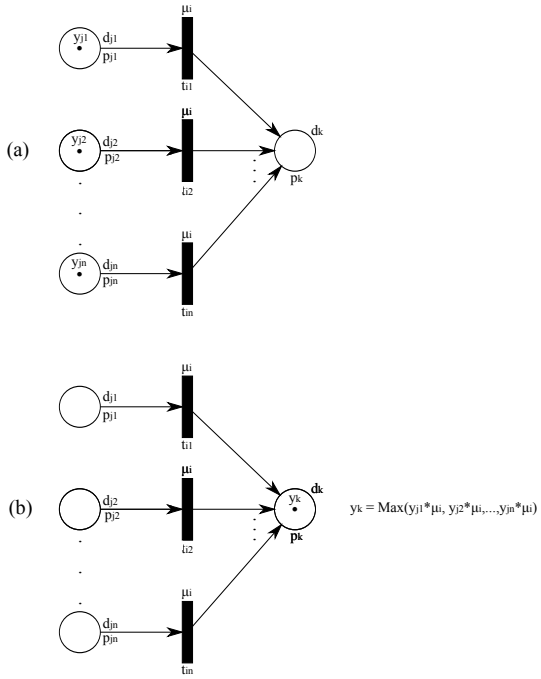

Figura 4.5: *FPN marcada de tipo 3. Antes y después de disparar la transición* t_i.

- **Tipo 4:** *Si* d_j *Entonces* d_{k1} OR d_{k2} OR...OR d_{kn} $(CF = \mu_i)$. Este tipo de reglas pueden ser un inconveniente para el control deductivo porque no hacen implicaciones específicas, por lo cual, ya no se mencionaran en lo que resta de la tesis.

4.3. Redes de Petri difusas adaptativas

En el modelado del sistema de conocimiento, se requiere la actualización del conocimiento, por lo que diversos autores han propuesto diversos tipos de modelos. En este caso se presentan las *redes de Petri difusas adaptativas (Adaptive Fuzzy Petri Nets, en abreviación AFPN)*, las cuales introducen el concepto adaptable a las FPN [23].

Una AFPN, se define como una tupla *AFPN = (P, T, D, I, O, M0,* α*,* β*, Th, W)* donde:

- P y T son conjuntos no vacíos, finitos y disjuntos de lugares y transiciones.
- $D = d_1, d_2, ..., d_n$ es el conjunto finito de proposiciones.
- I: $P \times T \rightarrow \{0, 1\}$ es el conjunto de lugares de entrada a T.
- O: $T \times P \rightarrow \{0, 1\}$ es el conjunto de lugares de salida de T.
- M_0: $P \rightarrow \{0, 1, 2, 3, ...\}$ es el marcado inicial.
- f es una función que asigna a cada transición un número difuso en el universo [0,1].
- α es una función de asociación, un mapeo de lugares a valores reales entre cero y uno, α: P$\rightarrow$[0,1].
- β es una función que asocia a un lugar p_i una proposición d_i, β: P$\rightarrow$ D.
- Th: T $\rightarrow [0, 1]$ es una función que asigna a cada transición un valor frontera λ_i, $\lambda_i \epsilon [0, 1]$.
- $W = W_I \cup W_O$. W_I: $I \rightarrow [0, 1]$ y W_O: $O \rightarrow [0, 1]$ son conjuntos de pesos de entrada y de salida (factores de certeza) los cuales asignan pesos a todos los arcos de la red.

La descripción del contenido de esta sección viene dada de la siguiente forma. En la *subsección 4.3.1* se describe la representación del conocimiento a través de las redes de Petri difusas adaptativas, y nuevamente, al igual que con las redes de Petri difusas, se muestra como estas pueden representar reglas de producción difusas.

4.3.1. Representación del conocimiento con redes de Petri difusas adaptativas

Las AFPN son el modelo de una clase particular de reglas de producción difusas, estas reglas se les conoce como reglas de producción difusas con pesos [23].

Definidas en tres tipos de reglas:

- **Tipo 1:** Una regla de producción difusa simple

$$R\text{: } Si\ a\ Entonces\ c, Th(t) = \lambda, W_O(t, p_j) = \mu, W_I(p_i, t) = \omega$$

Su gráfica correspondiente se muestra en la siguiente figura, (ver *figura 4.6*).

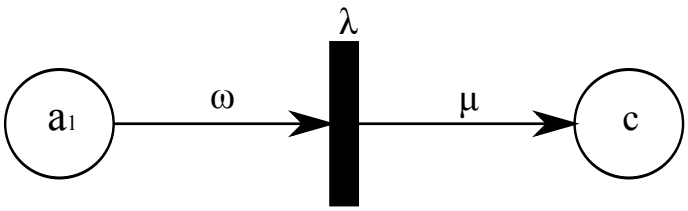

Figura 4.6: *AFPN de tipo 1.*

- **Tipo 2:** Una regla conjuntiva compuesta

$$R\text{: } Si\ a_1\ AND\ a_2\ AND ... AND\ a_n\ Entonces\ c,$$
$$Th(t_i) = \lambda_i, W_O(t_i, p_j) = \mu_i, W_I(p_i, t_i) = \omega_i, i = 1, 2, ..., n$$

Su gráfica correspondiente se muestra en la siguiente figura, (ver *figura 4.7*).

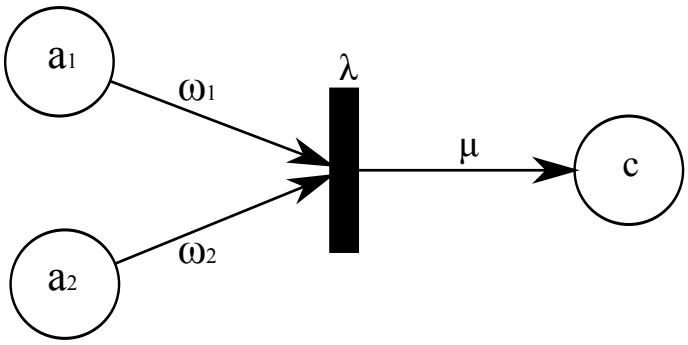

Figura 4.7: *AFPN de tipo 2.*

- **Tipo 3:** Una regla disyuntiva compuesta

$$R\text{: } Si\ a_1\ OR\ a_2\ OR ... OR\ a_n\ Entonces\ c,$$
$$Th(t_i) = \lambda_i, W_O(t_i, p_j) = \mu_i, W_I(p_i, t_i) = \omega_i, i = 1, 2, ..., n$$

Su gráfica correspondiente se muestra en la siguiente figura, (ver *figura 4.8*).

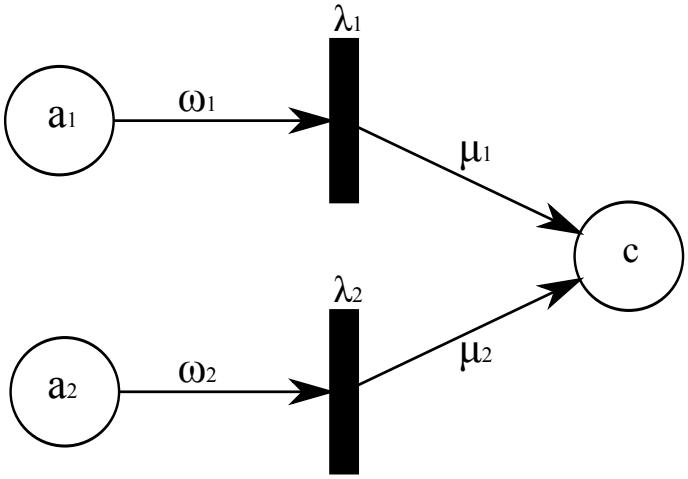

Figura 4.8: *AFPN de tipo 3.*

4.4. Razonamiento de conocimiento

En primer lugar, damos algunas definiciones básicas que son útiles para explicar la regla de disparo de la transición de cada AFPN [23].

Posición de origen, posición final: una posición p es llamada una posición de origen sino tiene transiciones de entrada. Y llamada posición final sino tiene transiciones de salida. Una posición de origen corresponde a una proposición antecedente de la WFPR, y una posición final corresponde a una consecuente. Por ejemplo, en la *figura 4.9*, P_1, P_2, P_3 y P_5 son posiciones de origen, mientras que P_6 es una posición final.

Ruta: Dada una posición p, una cadena de transiciones t_1, t_2,...,t_n es llamada una ruta para p, si p puede obtener un token de disparo a través de esta cadena de transiciones en secuencia de un grupo de posiciones de origen. Para una posición p, es posible que sean más de una ruta. Por ejemplo, en la *figura 4.9*, t_1t_3 es una ruta para P_6, t_2 es otra ruta para P_6.

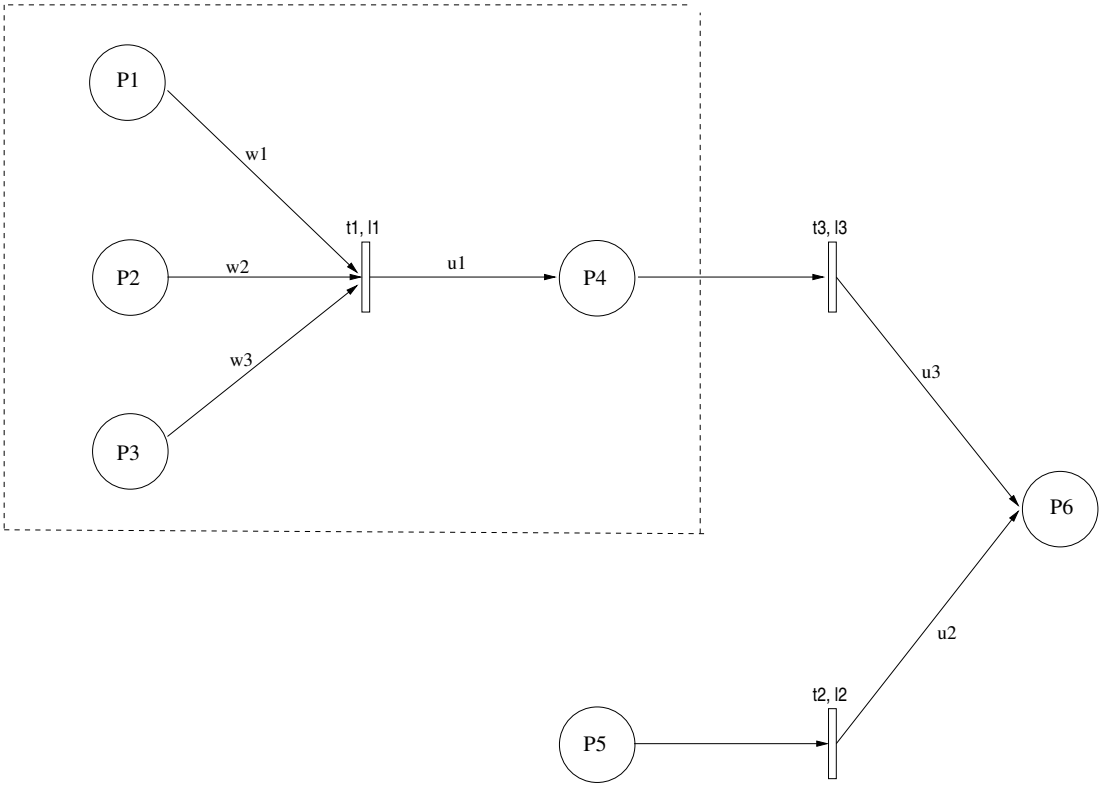

Figura 4.9: *Ejemplo de una AFPN*

Dividimos el conjunto de posiciones P en tres partes $P = P_{UI} \cup P_{int} \cup P_O$, donde P es el conjunto de posiciones de la AFPN; $P_{UI} = \{p \epsilon P \mid {}'p = \phi\}$, $p \epsilon P_{UI}$ es llamada una posición de entrada de usuario; $P_{int} = \{p \epsilon P \mid {}'p \neq \phi \ and \ p' \neq \phi\}$, $p \epsilon P_{int}$ es llamada una posición interior; $P_O = \{{}'p = \phi\}$, $p \epsilon P_O$ es llamada una posición de salida. ϕ es el conjunto vacío.

Una transición t está habilitada si todos sus lugares de entrada tienen tokens, si el factor de certeza es mayor que su umbral, entonces t dispara, así una AFPN puede ser implementada. De este modo, a través de transiciones disparadas, los factores de certeza pueden ser razonados desde un conjunto de proposiciones antecedentes conocido a un conjunto de proposiciones consecuentes paso a paso.

La marca de un lugar $m(p)$ está definida como el factor de certeza del token en ella. Cuando t está activada, se produce un nuevo factor de certeza $CF(t)$.

$$CF(t) = \begin{cases} \sum_j m(p_{Ij}) * \omega_{Ij} & \sum_j m(p_{Ij}) * \omega_{Ij} > Th(t) \\ 0 & \sum_j m(p_{Ij}) * \omega_{Ij} < Th(t) \end{cases}$$

Utilizamos una función continua $CF(t)(x)$ para aproximar $CF(t)$

$$CF(t)(x) = x * F(x)$$

donde

$$x = \sum_j m(p_{Ij}) * \omega_{Ij}$$

$F(x)$ es una función sigmoide que se aproxima al umbral de t

$$F(x) = 1/(1 + e^{-b(x - Th(t))})$$

donde b es un instante. Si b es suficientemente grande, cuando $x > Th(t)$, $e^{-b(x-Th(t))}) \approx 0$, entonces $F(x) \approx 1$ y cuando $x < Th(t)$, $e^{-b(x-Th(t))}) \to \infty$, entonces $F(x) \approx 0$.

En base a esta teoría de razonamiento, surge el siguiente algoritmo de razonamiento difuso (ver *algorimo 4.1*).

Algoritmo 4.1 Algoritmo de Razonamiento Difuso

Entrada: Factores de certeza de un conjunto de proposiciones antecedentes (corresponden a P_{UI}).

Salida: Factores de certeza de un conjunto de proposiciones consecuentes (corresponden a $P_{int} \cup P_O$).

Construir el conjunto de posiciones de entrada del usuario P_{UI}.

Construir el conjunto de transiciones iniciales habilitadas $P_{inicial}$.

Buscar la transición actual habilitada.

Mientras $T_{actual} \neq \phi$ **hacer**

Buscar la transición actual habilitada.

Calcular los nuevos factores de certeza producidos por las transiciones disparadas.

Efectuar la transmisión de tokens.

$T = T - T_{actual}$, $P = P - T_{actual}$.

Capítulo 5

Aprendizaje con redes de Petri difusas adaptativas

El algoritmo Backpropagation es el algoritmo clásico de aprendizaje. Se utiliza siempre para ajustar los parámetros de las funciones de pertenencia y los pesos de las redes. A pesar del éxito del algoritmo Backpropagation para el aprendizaje, este algoritmo posee una serie de deficiencias como: su gran dependencia de parámetros, puede quedar fácilmente atrapado en mínimos locales y generalmente es muy lento en la resolución de búsqueda y optimización, ya que utiliza pasos infinitesimales para alcanzar las soluciones [23].

El método de entrenamiento con algoritmos evolutivos AWEA, es diseñado con el objetivo de eliminar los problemas que sufre el algoritmo Backpropagation. La idea principal, es desarrollar un método alternativo de aprendizaje, este método no reemplazará a los métodos existentes sino que será una alternativa a tener en cuenta por el diseñador de una red en el momento de seleccionar el algoritmo de aprendizaje.

A continuación se hace una breve descripción del contenido de este capítulo. En la *sección 5.1* se explica la forma clásica de aprendizaje, a través del algoritmo Backpropagation, en la *sección 5.2* se definen los algoritmos evolutivos y la importancia de estos en la obtención de aprendizaje y finalmente en la *sección 5.3* se describe el método de aprendizaje utilizado, el algoritmo AWEA.

5.1. Aprendizaje clásico

El método de entrenamiento más utilizado es el método de gradiente descendente. Este método define una función $E(W)$ que proporciona el error que comete la red en función del conjunto de pesos sinápticos W. El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error, aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno [23].

A continuación se hace una breve descripción del contenido de esta sección. En la *subsección 5.1.1* se describe detalladamente como se realiza el aprendizaje de conocimiento de manera clásica, es decir, con el algoritmo Backpropagation y en la *subsección 5.1.2* se menciona cuales son las deficiencias de este algoritmo, lo que nos brinda la motivación necesaria para buscar un método alternativo en la adquisición de conocimiento.

5.1.1. Aprendizaje de conocimiento utilizando el algoritmo Backpropagation

El algoritmo Backpropagation se utiliza para ajustar los pesos y sesgos de una red con el fin de minimizar la suma de los errores de la red. El algoritmo Backpropagation es un método iterativo de optimización de gradiente descendente. El método Backpropagation, basado en la generalización de la regla delta, a pesar de sus limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las redes neuronales [24].

Para el desarrollo del algoritmo Backpropagation se hacen las siguientes suposiciones:

- El modelo AFPN de un sistema experto ha sido desarrollado.
- En el modelo AFPN, Th y W_O son conocidos.
- Un conjunto de factores de certeza P_{UI} y P_O está dado.

En está ocasión tomamos la regla de producción difusa con pesos (WFPR) de tipo 2 como una ilustración para mostrar el proceso de aprendizaje de conocimiento utilizado en la AFPN. LA regla de tipo 2 puede ser trasladada en una AFPN. Esta estructura AFPN puede ser trasladada a una estructura de red neuronal, definiendo a G como:

$$\mathrm{G}(x) := \mathrm{f}(x)^{*}x,$$

con $x = W^T \wedge = \sum_{i=1}^{n} \alpha_i \omega_i$; $\mathrm{f}(x) = \mu \ / \ 1{+}e^{-b(x-\lambda)}$ es una función sigmoide, b es una constante la cual se ajusta a $\mathrm{f}(x)$, W es el factor de pesos, $W := [\omega_1, \omega_2, ..., \omega_n]^T$, α es el vector de salida de la capa previa, $\wedge := [\alpha_1, \alpha_2, ..., \alpha_n]^T$. La función $\mathrm{f}(x)$ continua puede aproximar un umbral si μ, b y λ son valores apropiados seleccionados, ver la *figura 5.1* donde $\mu = 0.8$, $b = 200$ y $\lambda = 0.5$.

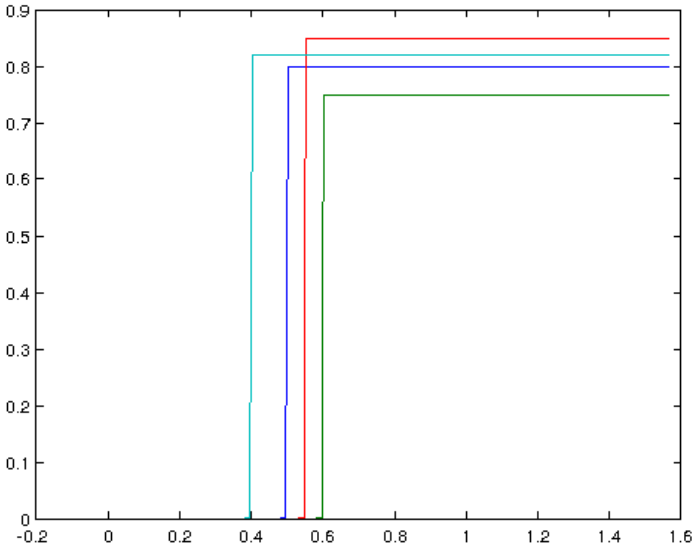

Figura 5.1: *Ejemplo de función sigmoide*

Para un lugar p, existen algunas rutas de aprendizaje las cuales pertenecen a un conjunto de lugares fuente. Los pesos en estas rutas pueden ser entrenados de acuerdo al algoritmo Backpropagation de esta sección. A lo largo de la ruta seleccionada Ω, el proceso de propagación (una capa oculta), es dada una información de entrada U y los pesos fijos ω_i, la salida $O(\Omega)$ puede expresarse:

$$O(\Omega) = G^{(n)}[W^{(n)}G^{(n-1)}(W^{(n-1)}G^{(n-2)}...W^{(2)}G^{(1)}(W^{(1)}U))].$$

donde $G^{(k)}(k = 1,...,n)$ es la función activa de la k-ésima capa, $W^{(k)}$ $(k = 1,...,n)$ es el peso de la k-ésima capa.
Si la información real es O^*, el vector de error de salida es:

$$e_n = O(\Omega) \text{ - } O^*.$$

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. El algoritmo Backpropagation que vamos utilizar es el mismo que el Backpropagation de redes neuronales multicapa. Los pesos en la capa de salida son actualizados de la siguiente manera:

$$W^{(n)}(k+1) = W^{(n)}(k) \text{ - } \gamma_i e_n \Lambda^{(n)}$$

donde

$$\Lambda^{(n)} \text{ Entrada de la } n_{th} \text{ capa;}$$
$$\gamma_i > 0 \text{ Ganancia adaptativa;}$$
$$W(K) \text{ Peso en el momento } k;$$
$$e_i = e_i + 1^{G'(i+1)}(x)W^{(i)}$$

los pesos en las capas ocultas son actualizados como:

$$W^{(n-1)}(k+1) = W^{(n-1)}(k) \text{ - } \gamma_{n-1} G'^{(n-1)} e_{n-1} \Lambda^{(n-1)}$$
$$.$$
$$.$$
$$.$$
$$W^{(2)}(k+1) = W^{(2)}(k) \text{ - } \gamma_2 G'^{(2)} e_2 \Lambda^{(2)}$$
$$W^{(1)}(k+1) = W^{(1)}(k) \text{ - } \gamma_1 G'^{(1)} e_1 \Lambda^{(1)}$$

donde G es la derivada de la función no lineal G.

$$G := \text{d/dx } [\mu * \text{x} / 1 + e^{-b(x-\lambda)}] = [\ \mu \text{x} b e^{-b(x-\lambda)} / (1 + e^{-b(x-\lambda)})^2\ ] + [\ \mu / 1 + e^{-b(x-\lambda)'}\ ]$$

Finalmente, la actualización de pesos mediante el algoritmo Backpropagation 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 la red con una época de ejemplos de entrenamiento. Para cada ejemplo en el conjunto, desarrollar las secuencias de ejecución descritas en los puntos 3 y 4.

3. *Cálculo 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. Calcular los campos locales inducidos y las señales de funciones de la red procediendo hacia adelante en la red, capa por capa.

   El campo inducido $v_j^{(I)}(k)$ para la neurona j en la capa l es:

   $$v_j^{(I)}(k) = \Sigma_{i=0}^{m_o} \omega_{ji}^{(l)}(k) y_i^{(l-1)}(k)$$

   Donde $y_i^{(l-1)}(k)$ es la señal de salida de la neurona i en la capa anterior l-1 en la capa iteración k y $\omega_{ji}^{(l)}(k)$ es el peso sináptico de la neurona j en la capa l que es alimentada de la neurona i en la capa l-1.

   Calculamos la señal de error como:

   $$e_j(k) = d_j(k) \text{ - } o_j(k)$$

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

4. *Cálculo hacia atrás.* Calcular los δ (gradientes locales) de la red definidos como:

   $\delta_j^{(l)}(k) = e_j^{(L)}(k)\varphi_j^{'}(v_j^{(l)}(k))$ para la neurona j en la capa de salida L
   $\delta_j^{(l)}(k) = \varphi_j^{'}(v_j^{(l)}(k))\Sigma_m \delta_m^{(l+1)}(k)\omega_{kj}^{(l+1)}(k)$ para la neurona j en la capa oculta l
   donde el apóstrofe en $\varphi_j^{'}$ denota diferenciación con respecto al argumento. Ajustar los pesos sinápticos de la red en la capa l de acuerdo a la regla delta generalizada:

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

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.

5.1.2. Deficiencias del algoritmo Backpropagation

A pesar del éxito del algoritmo para entrenar redes multicapa, este algoritmo posee una serie de deficiencias, las cuales se mencionan a continuación:

- **Dependencia de parámetros del algoritmo**. Los algoritmos de gradiente descendente hacen uso de una tasa de aprendizaje que idealmente debería ser infinitesimal. De esta manera, mediante pequeños ajustes de los pesos sinápticos el algoritmo converge hacia un mínimo. El uso de tasas de aprendizaje muy pequeñas hace que el algoritmo tenga una convergencia estable hacia un mínimo, aunque el tiempo necesario para alcanzarlo puede llegar a ser muy alto. El aumento de la tasa de aprendizaje disminuye el tiempo de convergencia, pero tiene un efecto contraproducente: el algoritmo comienza a oscilar en torno a un mínimo, disminuyendo la probabilidad de alcanzarlo. El efecto de oscilación puede reducirse mediante la adición de una tasa de momento, pero no puede eliminarse. El algoritmo Backpropagation es muy dependiente de estos parámetros, pequeñas variaciones sobre los parámetros del algoritmo pueden conducir a resultados diferentes. El principal problema es que no existe un método general que permita establecer el valor de estos parámetros.

- **Mínimos locales**. La superficie que define la función de error en base a los parámetros de la red, es compleja. Debido a la utilización del gradiente para encontrar el mínimo de dicha función de error se corre el riesgo de que el proceso de entrenamiento quede atrapado en un mínimo local. Esta situación no es deseable, si dicho mínimo esta localizado lejos del mínimo global.

- **Condiciones iniciales**. El conjunto de pesos iniciales generalmente se selecciona de manera aleatoria. Sin embargo, el algoritmo Backpropagation es muy dependiente de las condiciones iniciales seleccionadas. Pequeñas variaciones sobre las condiciones iniciales pueden ocasionar grandes diferencias en el tiempo de convergencia del algoritmo.

Estos problemas muestran la necesidad de buscar un método alternativo de aprendizaje. Este método no reemplazará los métodos existentes sino que será una alternativa a tener en cuenta por el diseñador de una red en el momento de seleccionar el algoritmo de aprendizaje.

5.2. Algoritmos evolutivos

La teoría evolutiva propuesta originalmente por Charles Darwin en combinación con el seleccionismo de August Weismann y la genética de Gregor Mendel, se conoce hoy en día como el paradigma Neo-Darwiniano. El Neo-Darwinismo establece que la historia de la mayoría de la vida en nuestro planeta puede ser explicada a través de una serie de procesos estadísticos que actúan sobre las poblaciones y especies: la reproducción, la mutación, la competencia y la selección. El término *computación evolutiva* engloba una serie de técnicas inspiradas en los principios de la teoría Neo-Darwiniana de la evolución natural.

La computación evolutiva se basa en el empleo de modelos de procesos evolutivos para el diseño e implementación de sistemas de resolución de problemas. Los distintos modelos computacionales que se han propuesto dentro de esta filosofía suelen recibir el nombre genérico de algoritmos evolutivos [25].

Un algoritmo evolutivo se basa en mantener una población de posibles soluciones del problema a resolver, llevar a cabo una serie de alteraciones sobre estas y efectuar una selección para determinar que soluciones permanecen en generaciones futuras y cuáles son eliminadas.

A continuación se hace una breve descripción del contenido de esta sección. En la *subsección 5.2.1* se presentan cuales son los principales paradigmas de la computación evolutiva, en la *subsección 5.2.2* se explica el funcionamiento de los algoritmos evolutivos, en la *subsección 5.2.3* se describen detalladamente sus componentes básicos y por último, la *subsección 5.2.4* nos muestra las ventajas de las técnicas evolutivos en el aprendizaje de conocimiento.

5.2.1. Principales paradigmas

El paradigma de la computación evolutiva consta de algoritmos estocásticos de búsqueda basados en abstracciones del proceso de la evolución de Darwin. Estos algoritmos tienen capacidad para solucionar las dificultades mencionadas anteriormente y se utilizan cada vez con mayor frecuencia para reemplazar a los métodos clásicos en la resolución de problemas reales en los que se presentan estos problemas. En esta área se han propuesto distintos modelos computacionales denominados algoritmos evolutivos [25]:

- *Algoritmos genéticos*, que modelan la evolución genética, por lo que las características de los individuos se expresan mediante genotipos.
- *Estrategias evolutivas*, que se orientan hacia la modelación de los parámetros estratégicos que controlan la variación en la evolución, es decir, la evolución de la evolución.
- *Programación evolutiva*, derivada de la simulación de comportamiento adaptativo en evolución.
- *Programación genética*, basadas en los algoritmos genéticos, pero en las que los individuos son programas (representados mediante árboles).

Todas estas instancias o tipos de algoritmos tratan de modelar la evolución y presentan las siguientes características comunes:

1. Utilizan el proceso de aprendizaje colectivo de una población de individuos. Normalmente cada individuo representa un punto dentro del espacio de búsqueda de todas las soluciones potenciales para un problema dado, es decir, codifica una solución candidata. Los individuos pueden incorporar adicionalmente otra información, como pueden ser los parámetros de la estrategia del algoritmo evolutivo. En el área de la computación evolutiva a la solución codificada se le denomina genotipo y a la solución decodificada (lo que realmente representa cada individuo en el contexto del problema) se le denomina fenotipo.

2. Los descendientes de los individuos se generan mediante procesos no deterministas que tratan de modelar los procesos de mutación y cruza. La mutación corresponde a una auto-replicación errónea de los individuos, mientras que la cruza intercambia material genético entre dos o más individuos ya existentes. Ambos operadores estocásticos, se aplican con probabilidades definidas por el usuario. Normalmente se establece una probabilidad de mutación muy inferior a la probabilidad de cruza, ya que una probabilidad de mutación muy elevada convertiría el proceso de búsqueda evolutivo en un proceso de búsqueda aleatoria. No obstante, la mutación es necesaria para incrementar la diversidad genética de individuos dentro de la población y para alcanzar valores de genes que no estén presentes en la población y que de otra forma serían inalcanzables, puesto que el operador de cruza solo intercambia genes (ya existentes) entre individuos.

3. Se asigna una medida de calidad (denominada habitualmente medida de adaptación o *fitness*) a cada individuo mediante el proceso de evaluación. El operador de selección actúa en base a esta medida y favorece, en el proceso de reproducción, a individuos mejores respecto a aquellos con peor valor de la función de adaptación.

5.2.2. Funcionamiento de los algoritmos evolutivos

El funcionamiento de cualquier algoritmo evolutivo se puede describir de la siguiente forma: se mantiene una población de posibles soluciones para el problema, se realizan modificaciones sobre las mismas y se seleccionan, en función de una medida de adaptación del individuo al entorno, aquellas que se mantendrán en generaciones futuras y las que serán eliminadas. La población evoluciona a través de las mejores regiones del espacio de búsqueda mediante los procesos de modificación y selección. Las modificaciones sobre la población permiten mezclar información de los padres que debe pasar a los descendientes (operador de cruza) o introducir innovación dentro de la población (operador de mutación).

Una característica importante de los algoritmos evolutivos es que realizan una búsqueda global. El hecho de que trabajen con una población de soluciones candidatas más que una solución individual, junto con el uso de operadores estocásticos, reduce la probabilidad de

caer en un óptimo local e incrementa la probabilidad de encontrar el máximo global.

Además, esta característica de búsqueda global hace a los algoritmos evolutivos especialmente adecuados para resolver problemas presentes en las distintas etapas del proceso de descubrimiento de conocimiento.

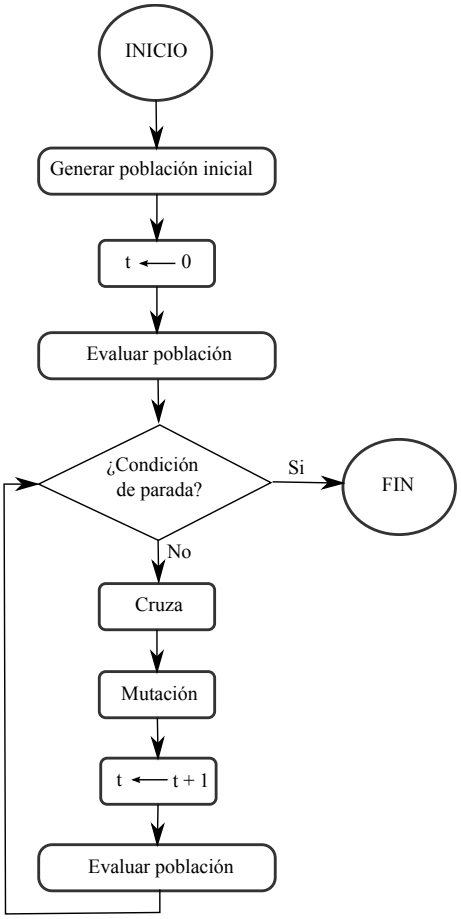

Figura 5.2: *Algoritmo de búsqueda en un enfoque evolutivo*

El funcionamiento básico de un algoritmo evolutivo es el siguiente (ver *figura 5.2*): el sistema parte de una población inicial de individuos que codifican, mediante alguna representación genética, soluciones candidatas al problema propuesto. Esta población de individuos (a los que se denomina cromosoma) evolucionan en el tiempo a través de un proceso de competición y variación controlada. Cada cromosoma de la población tiene asociada una medida de adaptación para determinar que cromosomas serán seleccionados para formar parte de la

nueva población en el proceso de competición. La nueva población se creará utilizando operadores genéticos de cruza y mutación. Este ciclo evolutivo continúa hasta que se verifique una determinada condición de parada: que se hayan realizado un determinado número máximo de evaluaciones de individuos, que la población haya evolucionado durante un número máximo de generaciones, que se haya alcanzado una solución con un determinado valor de la función de adaptación (o parte de ella), que se estacione la población por no generarse individuos nuevos durante un determinado número de generaciones, etc.

Aunque los algoritmos evolutivos no fueron diseñados específicamente para aprendizaje, sino como algoritmos de búsqueda global, ofrecen algunas ventajas en la extracción de conocimiento y específicamente para procesos de inducción de reglas:

- Tienden a tratar bien la interacción entre atributos debido a que suelen evaluar una regla como un todo mediante la función de adaptación, más que evaluar el impacto de añadir/eliminar una condición de una regla.
- Tienen la habilidad de rastrear minuciosamente el espacio de búsqueda y la capacidad de permitir funciones de adaptación arbitrarias en la búsqueda. La función de adaptación puede contener diferentes criterios así como la habilidad de penalizar el solapamiento entre reglas o conjuntos de reglas con demasiadas reglas, o una medida de calidad específica del problema.
- Además, la búsqueda genética utiliza implícitamente vuelta atrás en su búsqueda en el espacio de reglas, permitiéndole por tanto encontrar interacciones complejas que otras búsquedas sin vuelta atrás perderían.
- Una ventaja adicional sobre los algoritmos convencionales de aprendizaje de reglas es que la búsqueda se lleva a cabo entre un conjunto de reglas candidatas que compiten entre sí.

Sin embargo esto no quiere decir que los algoritmos evolutivos sean inherentemente superiores al resto de algoritmos de inducción de reglas, puesto que no hay ningún algoritmo de descubrimiento de reglas que sea mejor en todos los casos [26].

5.2.3. Componentes de un algoritmo evolutivo

Tengamos en cuenta que muchos de los componentes de un proceso evolutivo son estocásticos. Durante la selección, los individuos más aptos tienen una mayor probabilidad de ser seleccionados que los menos aptos, pero por lo general hasta los individuos débiles tienen la oportunidad de convertirse en un padre o para sobrevivir. Para la cruza de los individuos la elección de cuales piezas se van a combinar es al azar. Del mismo modo para la mutación, las piezas que se mutaron como posibles soluciones y las nuevas piezas que las reemplazaran, se eligen al azar. El esquema general de un algoritmo evolutivo es mostrado a continuación en el *algoritmo 5.1* en forma de pseudocódigo, la *figura 5.3* muestra su diagrama de flujo.

Algoritmo 5.1 Pseudocódigo del esquema general de un algoritmo evolutivo

Inicializar: *población* un número al azar de posibles candidatos.

Evaluar: cada candidato.

Repetir hasta (*CONDICIÓN DE PARO* se cumpla) **hacer**

Seleccionar padres.

Cruzar pareja de padres.

Mutar los hijos resultantes.

Evaluar los nuevos candidatos.

Seleccionar individuos para las siguientes generaciones.

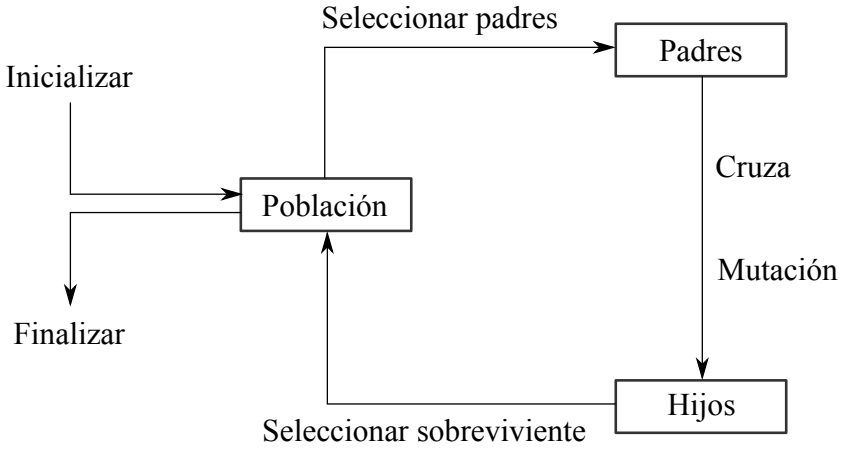

Figura 5.3: *Esquema general de un algoritmo evolutivo como un diagrama de flujo*

Los algoritmos evolutivos tienen una serie de componentes, procedimientos u operadores que deben especificarse con el fin de definir un algoritmo evolutivo particular. Los componentes más importantes son:

- Representación (definiciones de los individuos).
- Evaluaciones de función (ó función aptitud).
- Población.
- Mecanismo de selección de los padres.
- Los operadores de variación, la cruza y mutación.
- Mecanismo de selección de sobrevivientes (reemplazo).

Representación

El primer paso en la definición de un algoritmo evolutivo es vincular el "mundo real" al "mundo algoritmo evolutivo", es decir, para establecer un puente entre el contexto del problema original y el espacio para resolver problemas donde ocurre la evolución. A los objetos que forman las posibles soluciones en el contexto del problema original, se les conoce como fenotipo, mientras que su codificación, es decir, los individuos dentro del algoritmo evolutivo, son llamados genotipos.

La primera etapa de diseño se llama comúnmente representación, ya que equivale a la especificación de una asignación de los fenotipos en un conjunto de genotipos que se dice que representan estos fenotipos. Por ejemplo, dado un problema de optimización en números enteros, el conjunto de números enteros formaría el conjunto de fenotipos. Entonces se podría decidir que los representa por su código binario, por lo tanto, 18 sería visto como un fenotipo, y 10010 como el genotipo que lo representa. Es importante entender que el espacio fenotipo puede ser muy diferente del espacio genotipo, y que toda la búsqueda evolutiva se lleva a cabo en el espacio de genotipo. Una solución "un buen fenotipo" se obtiene mediante la decodificación del mejor genotipo después de la terminación.

La terminología común utiliza muchos sinónimos para nombrar los elementos de estos dos espacios. En el lado del contexto del problema original, solución candidata, fenotipo e individuo se utilizan para indicar los puntos del espacio de posibles soluciones. A este espacio se le llama el espacio fenotipo. En el lado del algoritmo evolutivo, genotipo, cromosoma y de nuevo individuo pueden ser utilizados para puntos en el espacio donde la búsqueda evolutiva en realidad ocurre. Este espacio se denomina como el espacio genotipo.

Evaluaciones de función (función aptitud)

El papel de las evaluaciones de función es representar los requerimientos para adaptarse a la forma básica de la selección y de ese modo facilitar el mejoramiento. Es decir, se trata de una función o procedimiento que asigna una medida de calidad a los genotipos. Siguiendo con el ejemplo anterior, si fuéramos a maximizar x^2 en números enteros, la aptitud del genotipo 10010 se podría definir como el cuadrado de su fenotipo correspondiente: $18^2 = 324$.

Las evaluaciones de función son comúnmente llamadas funciones de aptitud. A menudo, el problema original a resolver por un algoritmo evolutivo es un problema de optimización. En este caso, el nombre de función objetivo es utilizado en el contexto del problema original y la evaluación de función (aptitud) puede ser idéntica o una simple transformación de la función objetivo dada.

Población

El papel de la población es llevar a cabo la representación de las posibles soluciones. Una población es un conjunto múltiple de genotipos. Los individuos son objetos estáticos que no cambian o se adaptan: es la población la que lo hace. Dada una representación, la definición de una población puede ser tan simple como especificar el número de individuos que hay en ella, es decir, establecer el tamaño de la población. A diferencia de los operadores de variación que actúan sobre los individuos de uno o dos padres, los operadores de selección (selección de los padres y la selección sobreviviente) trabajan a nivel de población.

Por ejemplo, el mejor individuo de la población es elegido para la semilla de la próxima generación, o el peor individuo de la población es elegido para ser reemplazado por uno

nuevo. En casi todas las aplicaciones de algoritmos evolutivos el tamaño de la población es constante y no cambia durante la búsqueda evolutiva. La diversidad de una población es una medida del número de las diferentes soluciones presentes.

Mecanismo de selección de padres

El papel de la selección de padre o selección de apareamiento, es distinguir a los mejores individuos basados en su calidad, en particular, para permitir que los mejores individuos seleccionados se vuelven a seleccionar para someterlos a variaciones con el fin de crear descendencia. Junto con el mecanismo de selección de sobreviviente, la selección de los padres es responsable de impulsar mejoras en la calidad. La selección de los padres suele ser probabilística. Por lo tanto, las personas de alta calidad reciben una mayor probabilidad de convertirse en padres que aquellos con baja calidad. Sin embargo, las personas de baja calidad a menudo reciben una pequeña pero significativa oportunidad: de lo contrario toda la búsqueda podría llegar a ser demasiada codiciosa y atascarse en un óptimo local.

Operadores de variación

El papel de los operadores de variación es crear a nuevos individuos a partir de los viejos. En el espacio correspondiente del fenotipo esto equivale a generar nuevas soluciones candidatas. Desde la perspectiva de búsqueda: generar y probar, los operadores de variación realizan el paso de generar.

Mecanismo de selección de sobrevivientes (reemplazo)

El papel de la selección sobreviviente o selección ambiental consiste en distinguir entre las personas basadas en su calidad. En eso es similar a la selección de los padres, pero se usa en una etapa diferente del ciclo evolutivo. El mecanismo de selección sobreviviente se llama después de haber creado la descendencia de los padres seleccionados. El tamaño de la población es casi siempre constante, así una decisión que tiene que hacerse en los individuos se permitirá en la próxima generación. Esta decisión se basa generalmente en sus valores de aptitud, favoreciendo aquellas de mayor calidad.

5.2.4. Ventajas de las técnicas evolutivas

Es importante destacar las diversas ventajas que presenta el uso de técnicas evolutivas:

- Simplicidad conceptual.
- Amplia aplicabilidad.
- Superiores a las técnicas tradicionales en muchos problemas del mundo real.
- Tienen el potencial para incorporar conocimiento sobre el dominio y para hibridizarse con otras técnicas de búsqueda/optimización.
- Son robustas a los cambios dinámicos.
- Generalmente pueden auto-adaptar sus parámetros.
- Capaces de resolver problemas para los cuales no se conoce solución alguna.

5.3. Aprendizaje utilizando algoritmos evolutivos

Los algoritmos evolutivos no fueron diseñados específicamente para la obtención del aprendizaje de conocimiento, sino como algoritmos de búsqueda global, sin embargo, pueden realizar la extracción de conocimiento y específicamente para procesos de inducción de reglas. Aunque esto no los convierte en los mejores algoritmos de inducción de reglas, puesto que no existe ningún algoritmo que sea mejor en todos los casos.

A continuación se hace una breve descripción del contenido de esta sección. En la *subsección 5.3.1* se explica con un poco de teoría el proceso que seguido para el aprendizaje con el algoritmo evolutivo AWEA, la *subsección 5.3.2* describe cada uno de los componentes seleccionados para su elaboración y finalmente en la *subsección 5.3.3* se da una explicación general del funcionamiento del algoritmo evolutivo AWEA generando también el algoritmo básico de aprendizaje del modelo AFPN.

5.3.1. Aprendizaje de conocimiento utilizando el algoritmo evolutivo AWEA

La idea principal es que todos los pesos puedan ser actualizados mediante el algoritmo evolutivo AWEA, si los factores de certeza de todos los lugares están dados. La información se transmite de un lugar a otro lugar de la capa siguiente, desde la entrada a la salida. La información que sale de un lugar a otro de la capa siguiente se representa por un peso w_{ji}, de forma que la información total que llega a cada lugar j de la nueva capa es $\Sigma_i \omega_{ji} x_i$, siendo x_i el valor de salida de los lugares i de la capa anterior.

Este valor de entrada, para los lugares de las capas ocultas y la de salida, se transforma mediante una función de transferencia f produciendo la salida del lugar j, $x_j = f(\Sigma_i \omega_{ji} x_i)$. Las funciones de transferencia más usadas son la sigmoide $y = 1 \ / \ 1{+}e^{-x}$ y la identidad o lineal $y = x$.

Para el aprendizaje de una red se usan vectores de entrenamiento (cromosomas), consistentes cada uno en un vector de entrada y un vector de salida. Para un determinado vector de pesos w y cada vector de entrenamiento, la red lee el vector de entrada y compara la salida producida (cromosoma actual) por la red, con el vector de salida (cromosoma ideal).

A continuación se muestra la notación formal; sean n y K los números de lugares respectivamente de las capas de entrada y salida, sea S el conjunto de vectores de entrenamiento, $\forall(x, y \epsilon S)$ x indica el vector de entrada y y el de salida; sea $O = g(x, w)$ la salida producida en la red con el vector de pesos w cuando lee el vector de entrada x. Entonces el problema se puede formular como:

$$\text{minimizar } E(w)$$

siendo

$$E(w) = (1 \ / \ |S|) \ \Sigma_{(x,y)\epsilon S} \Sigma_{k=1}^{k} (O_k - y_k)^2$$

Las red con las que se va a trabajar va a estar compuesta por una capa oculta. Se denota a n el número de lugares de la capa de entrada (que lógicamente coincide con el número de variables de entrada) y m el número de lugares de la capa oculta. En nuestro caso se usa la función sigmoide como función de transferencia para la capa oculta. Un aspecto importante en este trabajo, los métodos de aprendizaje propuestos solo se aplican a los pesos asociados con la capa oculta. Cuando se genera un nuevo vector de pesos en la capa oculta, automáticamente se deben calcular los de la capa de salida y evaluar el valor del error E.

Hemos desarrollado un algoritmo evolutivo para el aprendizaje mostrado a continuación:

Algoritmo 5.2 Algoritmo Evolutivo con Adaptación de Pesos

Entrada: El cromosoma ideal (vector de peso W), la población (conjunto de cromosomas a evaluar).

Salida: Los mejores cromosomas (vectores que minimizan el error) y el mejor (los menores errores de cada llamada de función).

Iniciar población.

Calcular aptitud.

Obtener el mejor de la población.

Para $i = 0$, $i <$ número de iteraciones **hacer**

Selección de torneo determinista.

Cruzar intermedia de elementos.

Mutación uniforme de la población.

Calcular aptitud de la población.

Sustituir padres por hijos.

Elitismo.

5.3.2. Componentes del algoritmo evolutivo AWEA

A continuación se describen los componentes de nuestro algoritmo propuesto.

Selección mediante torneo determinista

La idea básica del método es seleccionar con base en comparaciones directas de los individuos. El algoritmo de la versión determinista es el siguiente:

- Barajar los individuos de la población.
- Escoger un número p de individuos (normalmente 2).
- Compararlos con base en su aptitud.
- El ganador del *torneo* es el individuo más apto.
- Se debe revolver el orden de la población un total de p veces para seleccionar N padres (donde N es el tamaño de la población).

Veamos un ejemplo de su funcionamiento:

Orden	Aptitud	Revolver	Ganadores
(1)	254	(2)	
(2)	47	(6)	(6)
(3)	457	(1)	
(4)	194	(3)	(3)
(5)	85	(5)	
(6)	310	(4)	(4)

Revolver	Ganadores
(4)	
(1)	(1)
(6)	
(5)	(6)
(2)	
(3)	(3)

Padres
(6) y (1),
(3) y (6),
(4) y (3).

Otra forma:

r = rnd(0,1) *aleatorio real*

		Ganador
Selecciona	*t1 = r = (3)*	*(3)*
	t2 = r = (6)	
Selecciona	*t1 = r = (1)*	*(1)*
	t2 = r = (4)	

Al realizar el análisis de la selección mediante torneo determinista podemos notar que: la técnica es eficiente y fácil de implementar, no requiere escalamiento de la función de aptitud (usa comparaciones directas), puede introducir una presión de selección muy alta porque a los individuos menos aptos no se les da oportunidad de sobrevivir, puede regularse la presión de selección variando el tamaño del torneo, etc.

Cruza intermedia

Una vez que se seleccionan los individuos, éstos se recombinan para producir la descendencia que se insertará en la siguiente generación. La idea principal de la cruza se basa en que, si se toman dos individuos (padres), se obtiene un hijo que comparte genes de ambos, existe la posibilidad de que los genes heredados sean precisamente una mejora de los padres.

En este caso, si suponemos que tenemos dos padres:

P1 = $(v_1, ..., v_m)$ y P2 = $(w_1, ..., w_m)$, los cuales se cruzan en la posición k, entonces los hijos producidos son:

$$\text{H1} = (v_1, ..., v_k, w_{k+1} * a + v_{k+1} * (1 - a), ..., w_m * a + v_m * (1 - a))$$
$$\text{H2} = (w_1, ..., w_k, v_{k+1} * a + w_{k+1} * (1 - a), ..., v_m * a + w_m * (1 - a))$$

donde: $a \in [0,1]$

Ejemplo de cruza intermedia, suponiendo $k = 3$, $a = 0.3$:

P1 = (*1.6, -2.1, 3.5, 0.4, 5.6, 7.8*) y P2 = (*0.2, 4.5, -2.3, 8.6, -1.4, 0.5*)

los hijos serían:

H1 = (*1.6, -2.1, 3.5, 8.6 * 0.3 + 0.4 * (1 - 0.3), -1.4 * 0.3 + 5.6 * (1 - 0.3), 0.5 * 0.3 + 7.8 * (1 - 0.3)*)

H2 = (*0.2, 4.5, -2.3, 0.4 * 0.3 + 8.6 * (1 - 0.3), 5.6 * 0.3 - 1.4 * (1 - 0.3), 7.8 * 0.3 + 0.5 * (1 - 0.3)*)

Mutación uniforme

Una mutación de un individuo hace que algunos de sus genes tengan valores aleatorios. La mutación se utiliza a menudo junto con la cruza. Al principio se seleccionan dos individuos de la población por la cruza. Si la cruza tiene éxito, entonces uno de los descendientes, o ambos, se mutan con una cierta probabilidad.

Dado:

$$\mathrm{P} = (V_1,...,V_k,...,V_m)$$

el individuo mutado será:

$$P' = (V_1,...,V_k',...,V_m)$$

donde:

$$V_k' = rnd(LB,\ UB)$$

se usa una distribución uniforme y $[LB,\ UB]$ definen los rangos mínimos y máximos de la variable V_k'.

Ejemplo:

$$\mathrm{P} = (5.3,\ -1.3,\ 7.8,\ 9.1)$$
$$V_k = 5.3,\ LB = 0.0,\ UB = 10.5$$
$$V_k' = rnd(0.0,\ 10.5) = 4.3$$

Calcular aptitud

La evaluación se realiza calculando la función objetivo para cada uno de los cromosomas que forman la población actual. De esta forma se determina la adaptación de cada individuo de la población actual. Indica si los individuos de la población representan una buena solución al problema. Por lo tanto para cada tipo de problema a resolver deberá derivarse un nuevo método, así como la codificación de los individuos.

Sustituir padres por hijos

Los padres son reemplazados por la nueva descendencia. Después de realizar una cruza surgen nuevos individuos, que se limitan a ocupar el lugar de sus padres en la población. En esta técnica se seleccionan dos individuos para obtener dos hijos, siendo incorporados a la población mediante la sustitución de uno o de dos individuos.

Elitismo

En ciertas ocasiones puede suceder que tras el cruce y la mutación, perdamos el cromosoma con mejor adaptación. Este método de selección copia el mejor cromosoma o alguno de los mejores en la nueva población. El elitismo puede mejorar el funcionamiento de los algoritmos evolutivos al evitar que se pierda la mejor solución. Una variación del elitismo es que el cromosoma solo se copie a la siguiente generación en caso de que tras una mutación no se haya generado un cromosoma mejor.

5.3.3. Algoritmo de Aprendizaje del modelo AFPN

El objetivo del algoritmo es hacer que los pesos se adapten, esto es, debemos asegurarnos de que un conjunto de pesos elegido al azar (el cromosoma inicial) sea lo más aproximado posible a nuestro conjunto de peso ideal (el cromosoma ideal).

En este algoritmo, el cromosoma ideal es un arreglo de tamaño igual al número de pesos que serán adaptados, este cromosoma es el ideal puesto que nuestra función depende de el, lo que significa que la evaluación de sus valores en la función objetivo, da como resultado el error mínimo.

Al principio del algoritmo, se crea una población de tamaño aleatorio y se evalúa sobre la función de error, la cual va a ser nuestra medida para decidir qué cromosoma es mejor. Después, el siguiente paso es encontrar el mejor cromosoma dentro de la población, esta tarea consiste en la búsqueda del cromosoma que contenga el error más bajo, y por lo tanto, que sus valores estén más cerca de los valores del cromosoma ideal.

A través del proceso evolutivo (cruza, mutación y selección) nuevos cromosomas se generan, para evitar perder el mejor elemento que tenemos, de acuerdo con la función de error, seleccionamos el cromosoma con el valor de error más alto y luego se reemplaza con el mejor cromosoma de la generación anterior, esto último tiene como objetivo la mejora de los valores de error de la población. Este procedimiento se ejecuta hasta que el algoritmo llega a un cierto número de generaciones.

Finalmente, basados en la forma en que la AFPN combinada con el AWEA aprende, podemos describir el siguiente algoritmo de aprendizaje de los sistemas basados en conocimiento:

Algoritmo de Aprendizaje del modelo AFPN

1. Seleccionar un conjunto de valores de peso inicial.
2. Para cada conjunto de datos de entrada, encontrar todas las rutas activas y marcarlas $\Omega_1, \Omega_2, ..., \Omega_k$.
3. Después de cada ruta activa, de acuerdo al algoritmo de razonamiento, calcular la salida correspondiente.
4. De acuerdo con el algoritmo evolutivo, obtener los mejores pesos (cada iteración los pesos se actualizan conforme al peso ideal) para ajustar los pesos en las rutas.

Capítulo 6

Experimentos

En este capítulo se presenta tres ejemplos típicos del aprendizaje de conocimiento, con el fin de mostrar los resultados obtenidos por nuestro algoritmo evolutivo y compararlos con el algoritmo clásico.

El contenido de este capítulo está distribuido de la siguiente manera. En la *sección 6.1* se realiza un experimento sencillo, que puede ser fácilmente comprobado, dando valores a las entradas $\alpha(P_1)$ y $\alpha(P_2)$ aleatoriamente entre 0 y 1, en la sección *sección 6.2* nos encontramos con el ejemplo 2, en el cual se explica como se realiza el aprendizaje, tanto con el algoritmo Backpropagation como con el algoritmo evolutivo y finalmente, en la *sección 6.3* tenemos el ejemplo 3 que además de mostrar los resultados del aprendizaje como en los ejemplos 1 y 2, también hace una comparación entre ambos algoritmos.

6.1. Diseño experimental

Los experimentos fueron diseñados para establecer una comparación directa entre ambos métodos. Todos los experimentos fueron diseñados en una laptop ASUS N53SM con las siguientes características:

- Procesador. Intel® Core^{TM} i7 2670QM
- Chipset: Intel® HM65 Express Chipset
- Memoria: DDR3 1333 MHz SDRAM, 4GB

- Almacenamiento: 2.5" SATA 750GB 7200RPM con 750 G SSD
- Networking: 802.11 b/g/n con 10/100/1000 Base T

Para el algoritmo Backpropagation se utilizaron 400 iteraciones en el ejemplo 1 y 1500 en los ejemplos 2 y 3, aunque sólo se utilizaron las primeras 200 iteraciones para observar el comportamiento del error producido comparado con el error producido por el algoritmo AWEA en el ejemplo 3, todos los $\alpha(P_i)$ fueron dados aleatoriamente en el rango [0,1], los demás valores como λ_i, μ_i, ω_i y W_i están dados dentro de la descripción de cada ejemplo.

Para el algoritmo evolutivo AWEA se utilizaron 200 iteraciones en todos los ejemplos, el tamaño del cromosoma en cada ejemplo depende del número de pesos que se quieren adaptar (para el ejemplo 3, el tamaño del cromosoma es 6), el tamaño de la población dado es de 4 (tamaño de población es aleatorio), el porcentaje de cruza es de 0.8, el porcentaje de mutación es de 0.2; al igual que en el algoritmo Backpropagation; todos los $\alpha(P_i)$ fueron dados aleatoriamente en el rango [0,1], los demás valores como λ_i, μ_i, ω_i y W_i están dados dentro de la descripción de cada ejemplo.

Recordemos que se debe establecer una probabilidad de mutación muy inferior a la probabilidad de cruza, ya que una probabilidad de mutación muy elevada convertiría el proceso de búsqueda evolutivo en un proceso de búsqueda aleatoria. No obstante, la mutación es necesaria para incrementar la diversidad genética de individuos dentro de la población y que de otra forma serían inalcanzables, puesto que el operador de cruza solo intercambia genes (ya existentes) entre individuos.

Las siguientes simulaciones sirven para mostrar el razonamiento difuso y el algoritmo de aprendizaje de pesos.

6.2. Ejemplo 1

P_1, P_2, P_3 son proposiciones relacionadas de un sistema experto Γ_1. Entre ellas existe la siguiente regla de producción difusa con pesos:

R_1: *Si* P_1 AND P_2 *Entonces* P_3 (ω_1, ω_2, λ_1, μ_1).

Primero, basados en el principio de traslación, mapeamos Γ_1 en una AFPN (ver *figura 6.1*):

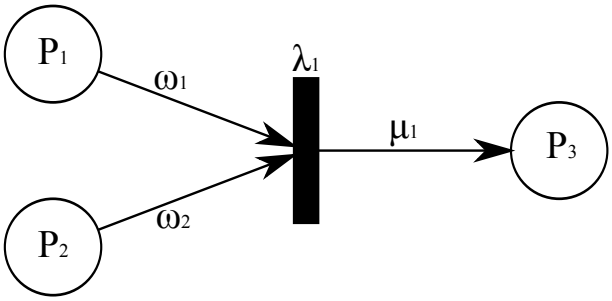

Figura 6.1: Ejemplo de AFPN para el sistema experto Γ_1

$$AFPN = \{P, T, D, I, O, \alpha, \beta, Th, W\},$$

donde

$$P = \{p_1, p_2, p_3\}$$

$$T = \{t_1\}$$

$$D = \{P_1, P_2, P_3\}$$

$$Th = \{\lambda_1\}$$

$$W_I = \{\omega_1, \omega_2\}$$

$$W_O = \{\mu_1\}$$

Tenemos dos proposiciones de entrada (P_1, P_2) y una proposición de salida (P_3). Los datos están dados como:

$$\mu_1 = 0.7,$$

$$\lambda_1 = 0.4,$$

$$\omega_1 = 0.8 \text{ y } \omega_2 = 0.1.$$

Utilizamos la función sigmoide:

$$F_i(x) := \mu_i \ / \ 1+e^{-b_i(x-\lambda_i)},\ i = 1$$

para aproximar el umbral λ_1 la pendiente b_i es seleccionada como 20.

$$F_1(x) := 0.7 \ / \ 1{+}e^{-20(x-0.4)}$$

Usando el algoritmo de razonamiento difuso, un conjunto de datos de salida pueden ser calculados de acuerdo a los datos de entrada. El uso de una función sigmoide para aproximar un umbral significa que el cero exacto es imposible de obtener, pero si el coeficiente de la pendiente es suficientemente pequeño, la función sigmoide aproxima al umbral con buena exactitud.

Si la proposición $\alpha(P_1) < \lambda_1$ o la proposición $\alpha(P_2) < \lambda_1$, la transición t_1 no puede disparar, así de acuerdo a R_1 el factor de certeza de salida de $\alpha(P_3)$ es 0. Si las entradas $\alpha(P_1)$ y $\alpha(P_2)$ están dadas aleatoriamente desde 1 a 0, podemos obtener la salida real $\alpha(P_3)$ de acuerdo al sistema experto Γ_1. Dada una condición inicial para ω_1 y ω_2, ponemos las mismas entradas a la red.

$$\omega_1 = 0.8 \text{ y } \omega_2 = 0.1$$
$$x = \alpha(P_1)\omega_1 + \alpha(P_2)\omega_2$$
$$x = \alpha(P_1)^*0.8 + \alpha(P_2)^*0.1$$
$$\alpha(P_3)(k) = \mu_1 \ / \ 1{+}e^{-b(x-\lambda_1)}$$
$$\alpha(P_3)(k) = 0.7 \ / \ 1{+}e^{-20(x-0.4)}$$

Proponiendo que los pesos ideales sean:

$$W_1 = 0.6 \text{ y } W_2 = 0.3$$
$$y = \alpha(P_1)W_1 + \alpha(P_2)W_2$$
$$y = \alpha(P_1)^*0.6 + \alpha(P_2)^*0.3$$
$$\alpha'(P_3)(k) = \mu_1 \ / \ 1{+}e^{-b(y-\lambda_1)}$$
$$\alpha'(P_3)(k) = 0.7 \ / \ 1{+}e^{-20(y-0.4)}$$

El error entre la salida de la red ($\alpha'(P_3)$) y la del sistema experto ($\Gamma_1\alpha(P_3)$) puede ser usado para modificar los pesos, podemos utilizar la siguiente ley de aprendizaje:

$$W(k+1) = W(k) + G\delta e(k) \wedge (P(k)),\ \delta > 0$$
$$e(k) := \alpha(P_3)(k) \text{ - } \alpha'(P_3)(k)$$

aquí δ es la tasa de aprendizaje, una pequeña δ puede asegurar la estabilidad del proceso de aprendizaje. Seleccionamos $\delta = 0.07$, $W(k) = [\omega_1(k), \omega_2(k)]$, $\wedge(P(k)) = [\alpha(P_1(k)), \alpha(P_2(k))]$ y $G(x) := \mu_1 * x \ / \ 1{+}e^{-b(x-\lambda_1)}$.

$$\omega_1(k+1) = \omega_1(k) + \mu_1 * b * e^{-b(y-\lambda_1)} * \delta * e(k) * \alpha(P_1(k)) \ / \ (1{+}e^{-b(y-\lambda_1)})^2$$
$$\omega_2(k+1) = \omega_2(k) + \mu_1 * b * e^{-b(y-\lambda_1)} * \delta * e(k) * \alpha(P_2(k)) \ / \ (1{+}e^{-b(y-\lambda_1)})^2$$

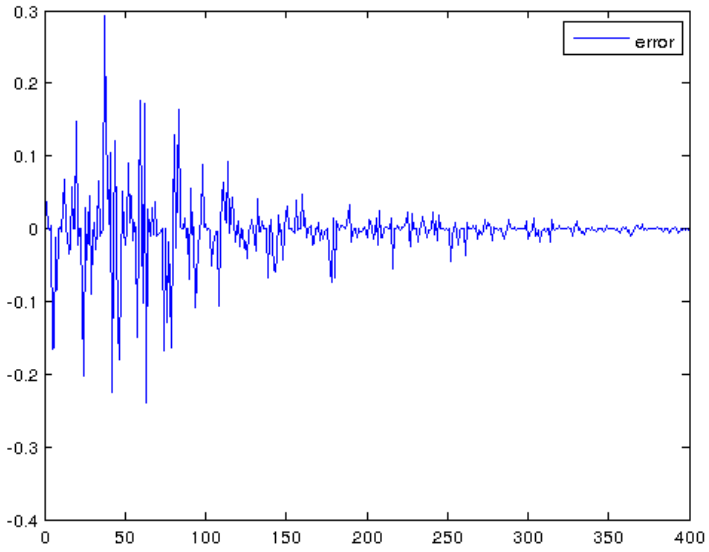

Figura 6.2: Comportamiento del error del sistema experto Γ_1

En la *figura 6.2* se observa como el error va disminuyendo cada que los pesos se van ajustando, hasta que al final el error es el mínimo.

Después del proceso de entrenamiento, los pesos convergen a valores reales, esto se observa en la gráfica obtenida de la simulación donde ω_1 y ω_2 oscilan hasta estabilizarse en $k > 400$, (ver *figura 6.3*).

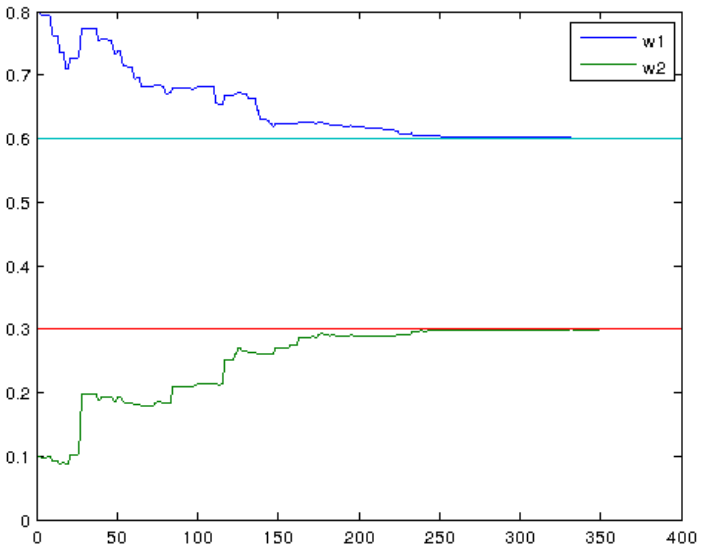

Figura 6.3: *Ejemplo del sistema experto φ_1 mapeado en una AFPN*

6.3. Ejemplo 2

P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11} y P_{12} son proposiciones relacionadas de un sistema experto Γ_2. Existen las siguientes reglas de producción difusas con pesos:

$$R_1\text{: } Si\ P_1 \text{ AND } P_2 \text{ AND } P_3\ Entonces\ P_8\ (\omega_1, \omega_2, \omega_3, \lambda_1, \mu_1)$$
$$R_2\text{: } Si\ P_4 \text{ AND } P_5\ Entonces\ P_9\ (\omega_4, \omega_5, \lambda_2, \mu_2)$$
$$R_3\text{: } Si\ P_6 \text{ AND } P_8\ Entonces\ P_{10}\ (\omega_6, \omega_8, \lambda_5, \mu_5)$$
$$R_4\text{: } Si\ P_7 \text{ OR } P_9\ Entonces\ P_{11}\ (\lambda_3, \lambda_4, \mu_3, \mu_{42})$$
$$R_5\text{: } Si\ P_9 \text{ OR } P_{10}\ Entonces\ P_{12}\ (\lambda_4, \lambda_6, \mu_{41}, \mu_6)$$

Basado en el principio de traslación, mapeamos nuestro sistema experto Γ_2 en una AFPN como se muestra a en la *figura 6.4*:

$$AFPN_1 = \{P, T, D, I, O, \alpha, \beta, Th, W\}$$

donde

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}\},$$
$$T = \{t_1, t_2, t_3, t_4, t_5, t_6\},$$
$$D = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}\},$$
$$Th = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\},$$
$$W_I = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_8\},$$
$$W_O = \{\mu_1, \mu_2, \mu_3, \mu_{41}, \mu_{42}, \mu_5, \mu_6\}.$$

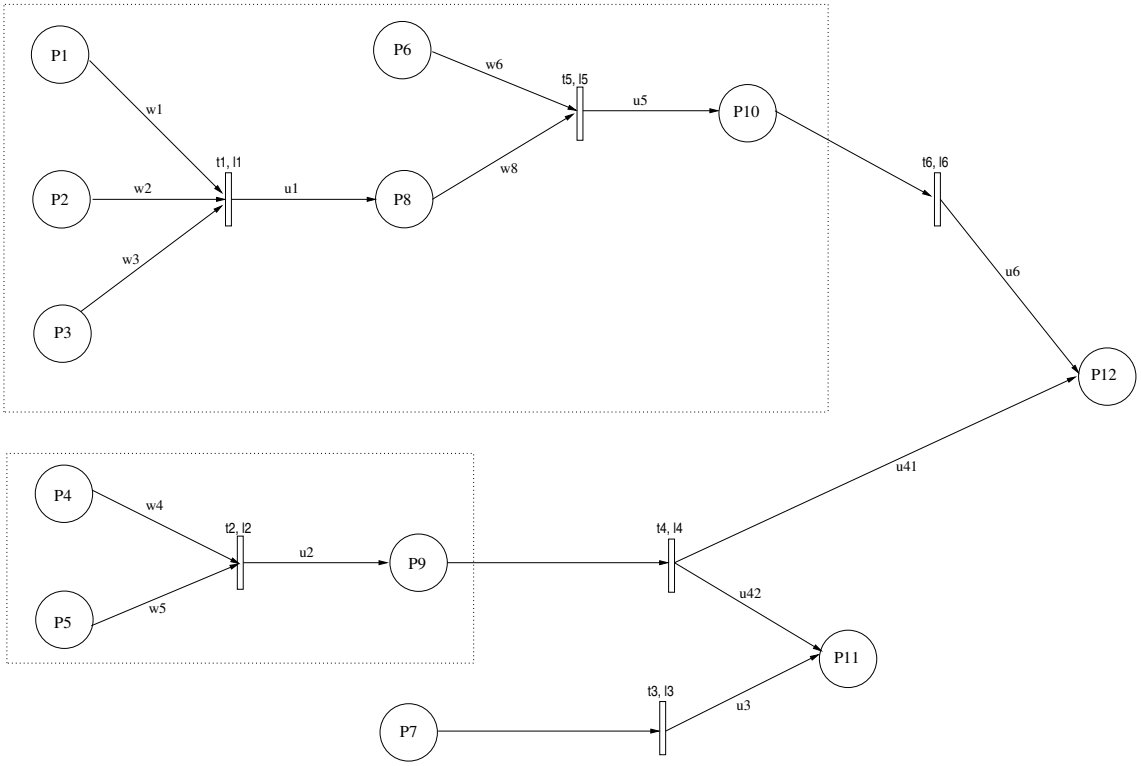

Figura 6.4: *Ejemplo del sistema experto* Γ_2 *mapeado en una AFPN*

Así el modelo AFPN para este sistema experto puede ser representado como en la *figura 6.4*, las dos cajas de guiones discontinuos son las partes del aprendizaje.

Para la caja superior de guiones discontinuos suponemos que los pesos ideales son:

$$W_1 = 0.53,\ W_2 = 0.17,\ W_3 = 0.9,\ W_6 = 0.68,\ W_8 = 0.37$$

un conjunto de datos sobre la parte de aprendizaje de la AFPN

$$\lambda 1 = 0.5,\ \lambda_5 = 0.4,\ \mu_1 = 0.9,\ \mu_5 = 0.8$$

dado un conjunto de valores iniciales de los pesos

$$\omega_1 = 0.8,\ \omega_2 = 0.2,\ \omega_3 = 0.5,\ \omega_6 = 0.1,\ \omega_8 = 0.7$$

y la tasa de aprendizaje $\delta = 0.5$. Los resultados del aprendizaje con el algoritmo Backpropagation se muestran en la *figura 6.5* y la *figura 6.7*. Y los resultados del aprendizaje con enfoque evolutivo se muestran en la *figura 6.6* y la *figura 6.8*.

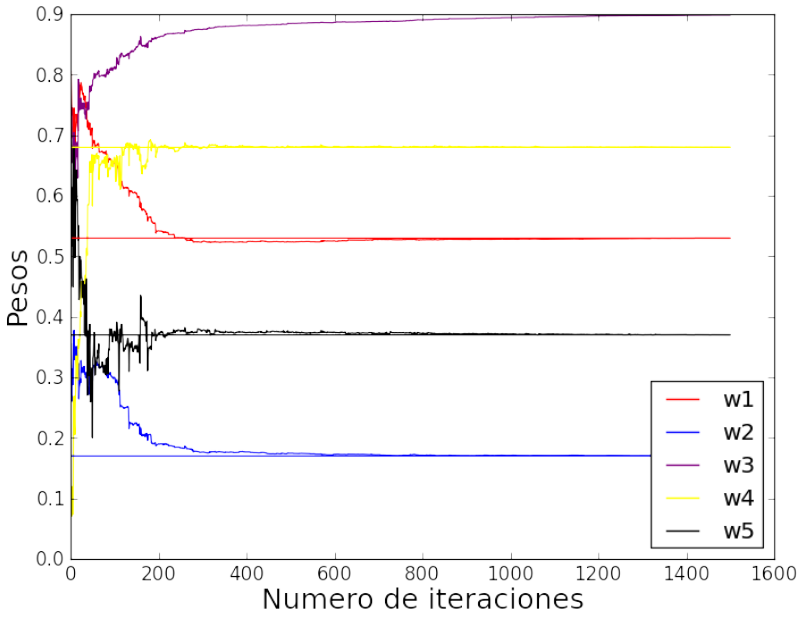

Figura 6.5: *Resultados del aprendizaje con el algoritmo Backpropagation para la caja superior*

En la *figura 6.5* se puede observar como los valores iniciales se están acercando hacia los valores ideales. Esto se realiza mediante la actualización de los pesos basados en el peso anterior utilizando una función de adaptación que en este caso es la función sigmoide.

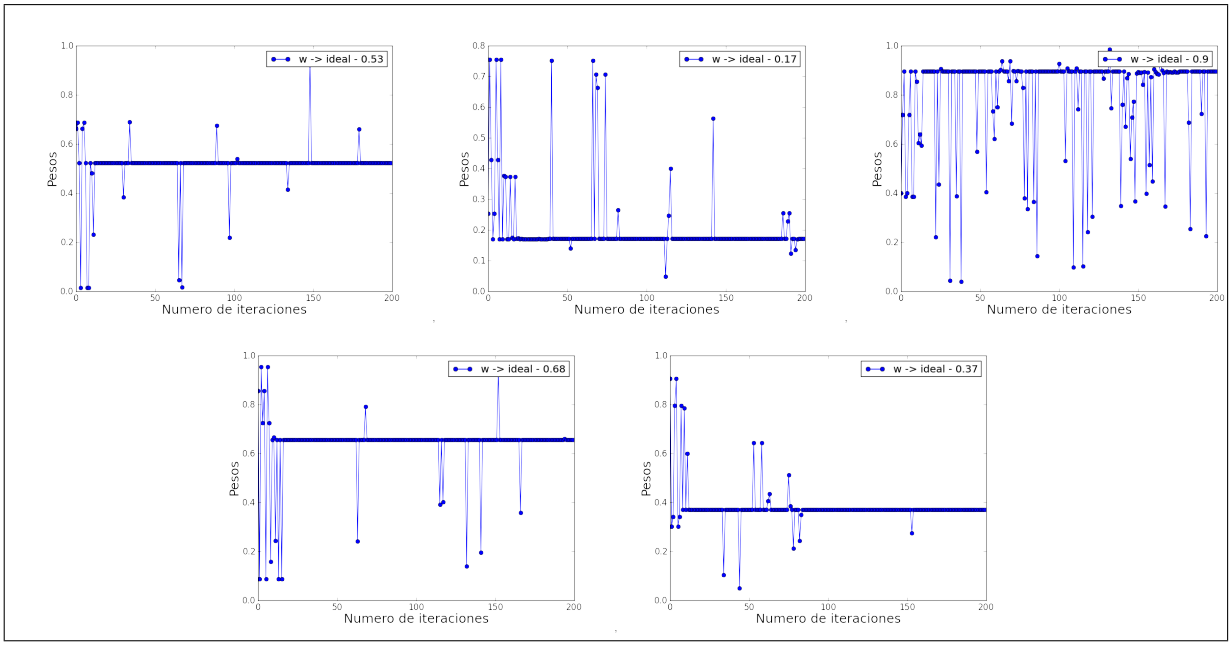

Figura 6.6: *Resultados del aprendizaje con AWEA para la caja superior*

En la *figura 6.6* se muestran los valores que tomo cada uno de los pesos del cromosoma inicial, los valores oscilan entre los pesos del cromosoma ideal puesto que en esos valores es donde la función está más cerca de converger, es decir, donde el error es el mínimo. Por ejemplo, para nuestra W_1 vemos que el peso que con más frecuencia aparece es su peso ideal 0.53, o que para nuestra W_4 también nos damos cuenta que el peso que más aparece es su peso ideal 0.68, esto es, porque el AWEA va a ir encontrando los pesos que minimicen el error hasta que simplemente nuestros pesos iniciales se hayan convertido en los pesos ideales.

Para la caja inferior de guiones discontinuos suponemos que los pesos ideales son:

$$W_4 = 0.63, W_2 = 0.37$$

dado un conjunto de valores iniciales de los pesos

$$\omega_4 = 0.8, \omega_2 = 0.2$$

el conjunto de datos para la parte de aprendizaje de la AFPN y la tasa de aprendizaje conservan los mismos valores.

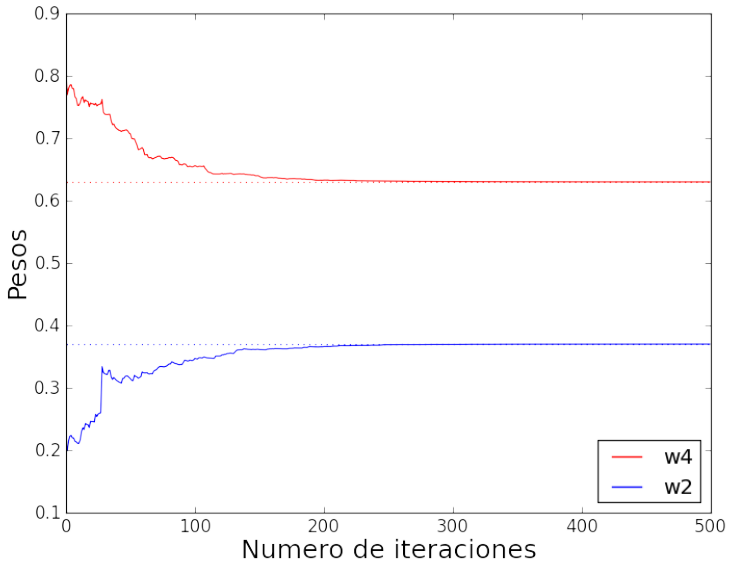

Figura 6.7: *Resultados del aprendizaje con el algoritmo Backpropagation para la caja inferior*

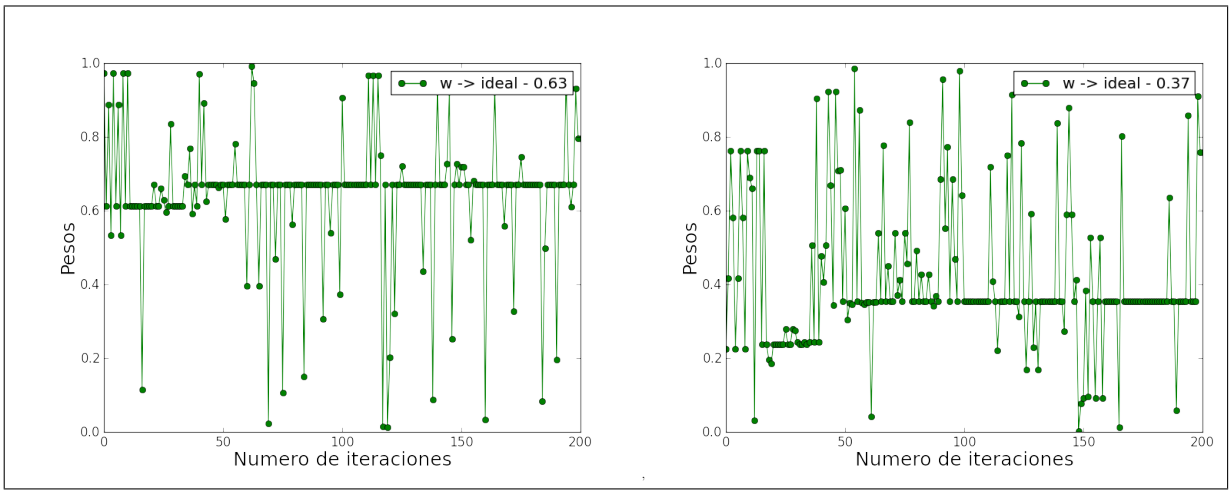

Figura 6.8: *Resultados del aprendizaje con AWEA para la caja inferior*

En la *figura 6.7* el aprendizaje es a través del algoritmo Backpropagation y se puede apreciar claramente donde inician los pesos y como se van actualizando hasta llegar a su valor ideal. Mientras que en la *figura 6.8* apreciamos el comportamiento de ambos pesos y como cada uno aparece con más frecuencia en el valor de su peso ideal.

6.4. Ejemplo 3

P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10} y P_{11} son proposiciones relacionadas para un sistema experto Γ_3. Entre estas proposiciones existen las siguientes reglas de producción difusas con pesos:

$$R_1\text{: } Si\ P_1 \text{ AND } P_2 \text{ AND } P_3\ Entonces\ P_5\ (\omega_1, \omega_2, \omega_3, \lambda_1, \mu_1)$$
$$R_2\text{: } Si\ P_4 \text{ AND } P_5 \text{ AND } P_6\ Entonces\ P_9\ (\omega_4, \omega_5, \omega_6, \lambda_2, \mu_2)$$
$$R_3\text{: } Si\ P_7 \text{ AND } P_8\ Entonces\ P_{10}\ (\omega_7, \omega_8, \lambda_3, \mu_3)$$
$$R_4\text{: } Si\ P_9 \text{ OR } P_{10}\ Entonces\ P_{11}\ (\lambda_4, \lambda_5, \mu_4, \mu_5)$$

$$AFPN_2 = \{P, T, D, I, O, \alpha, \beta, Th, W\}$$

donde

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}\}$$
$$T = \{t_1, t_2, t_3, t_4, t_5\}$$
$$D = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}\}$$
$$Th = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$$
$$W_I = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6, \omega_7, \omega_8\}$$
$$W_O = \{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5\}$$

En primer lugar, al igual que en los ejemplos anteriores, se mapea nuestro sistema experto Γ_3 en una AFPN como se muestra en la figura siguiente (ver *figura 6.9*).

En este ejemplo, el aprendizaje se puede lograr en dos partes diferentes de acuerdo con el grafo de la *figura 6.9*, es importante mencionar que en este ejemplo sólo vamos a trabajar con la caja superior de guiones discontinuos puesto que para la caja inferior el resultado es igual que en el ejemplo 2.

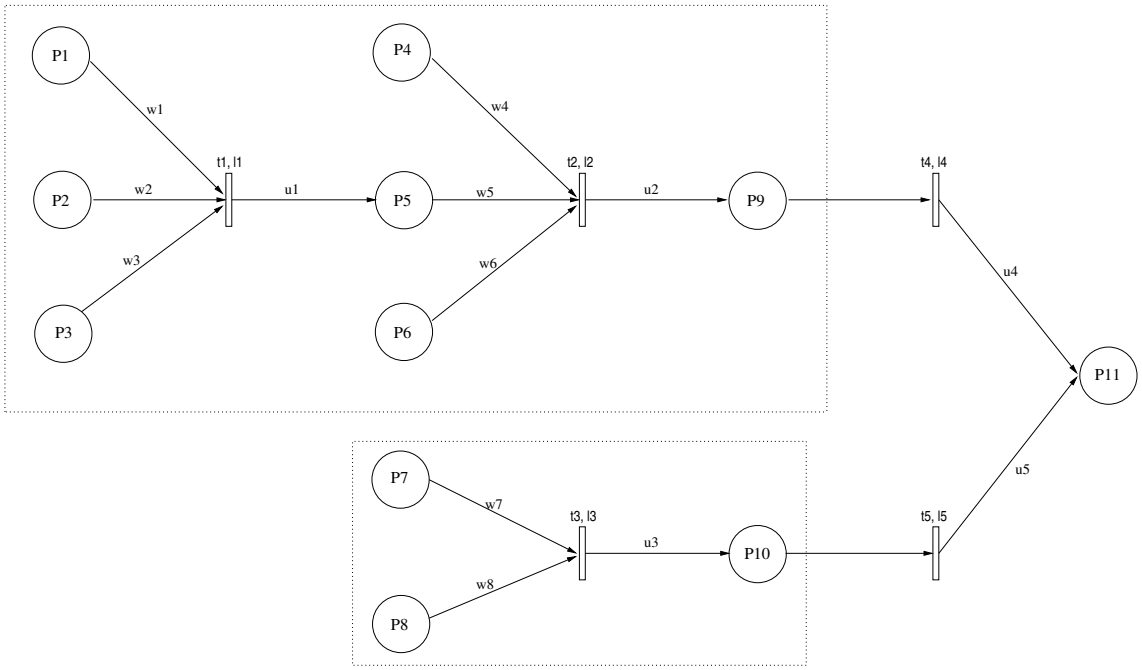

Figura 6.9: *Ejemplo del sistema experto* Γ_3 *mapeado en una AFPN*

Suponemos que los pesos ideales son:

$$W_1 = 0.71,\ W_2 = 0.13,\ W_3 = 0.84,\ W_4 = 0.62,\ W_5 = 0.33,\ W_6 = 0.21$$

un conjunto de datos sobre la parte de aprendizaje de la AFPN

$$\lambda 1 = 0.5,\ \lambda_2 = 0.4,\ \mu_1 = 0.9,\ \mu_2 = 0.8$$

dado un conjunto de valores iniciales de los pesos

$$\omega_1 = 0.1,\ \omega_2 = 0.35,\ \omega_3 = 0.65,\ \omega_4 = 0.9,\ \omega_5 = 0.7,\ \omega_6 = 0.42$$

y la tasa de aprendizaje $\delta = 0.5$. Los resultados del aprendizaje con el algoritmo Backpropagation se muestran en la *figura 6.10* y los resultados del aprendizaje con enfoque evolutivo se muestran en la *figura 6.11.*

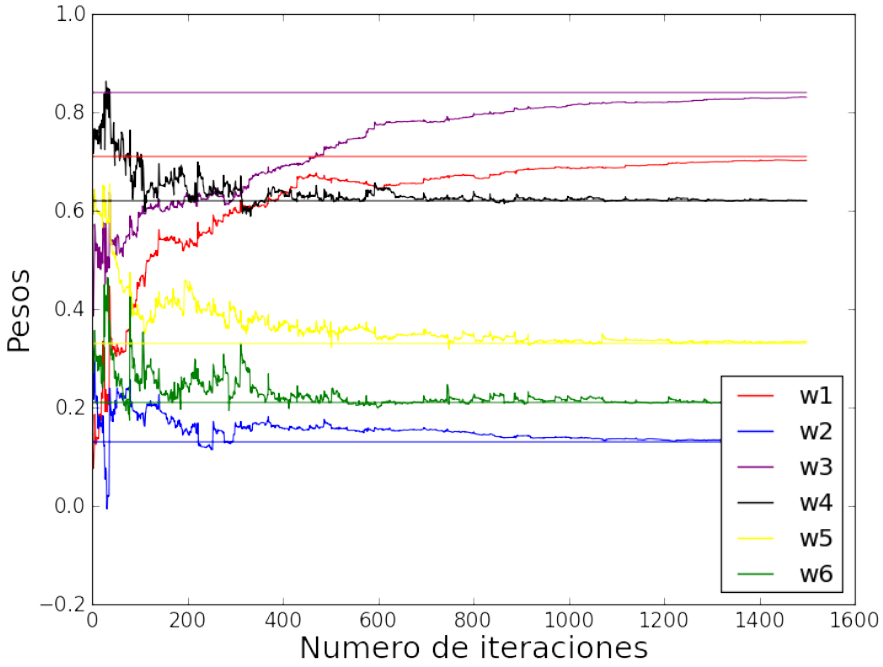

Figura 6.10: *Resultados del aprendizaje con el algoritmo de Backpropagation*

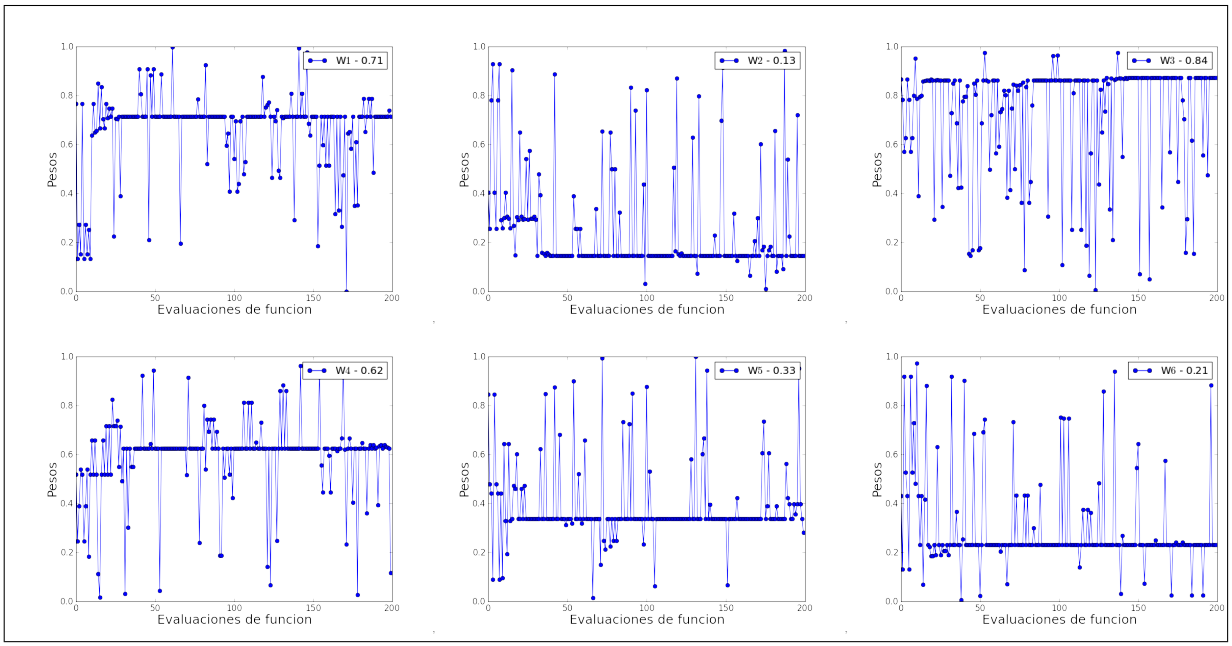

Figura 6.11: *Resultados del aprendizaje con AWEA*

Los ejemplos demuestran que el algoritmo de razonamiento difuso puede trabajar de igual manera con el algoritmo Backpropagation que con nuestro algoritmo evolutivo. La *figura 6.12* muestra una comparación entre el enfoque evolutivo contra el algoritmo Backpropagation.

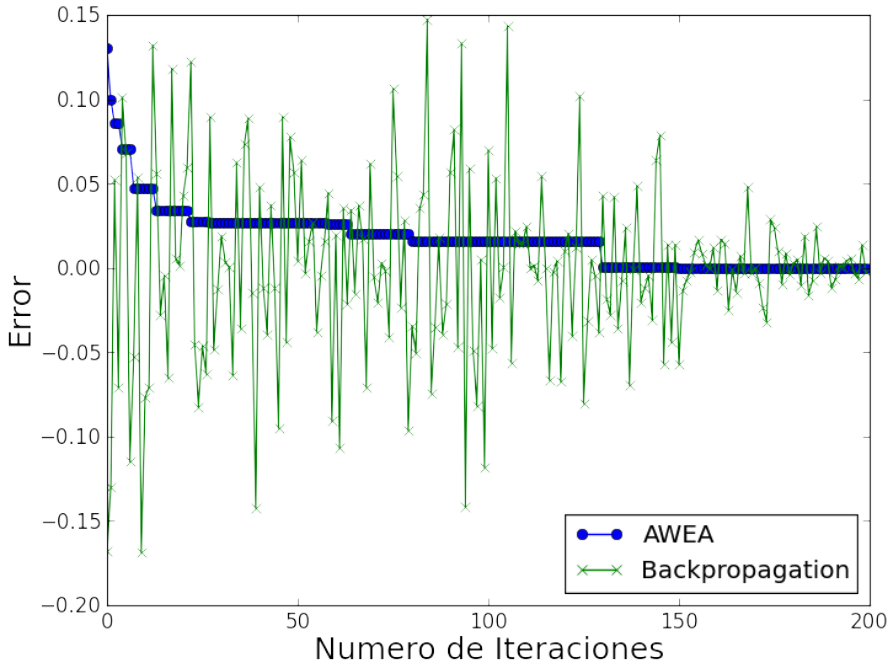

Figura 6.12: *Comparación de los algoritmos*

Capítulo 7

Conclusión

Del *ejemplo 2* de los experimentos, concluimos que el método propuesto con el algoritmo evolutivo puede resolver problemas de la misma forma en que lo hace el algoritmo Backpropagation, con las mismas características, pero de una forma más eficaz, sencilla y evitando los problemas que suele tener el algoritmo Backpropagation.

Del *ejemplo 3* de los experimentos, podemos concluir que los problemas a resolver pueden incrementar por mucho su grado de complejidad, debido a que todavía es muy rápido calcular el error en los problemas conocidos.

El algoritmo Backpropagation se caracteriza por su alta dependencia de los parámetros seleccionados. La forma en que se ajustan los pesos se determina por las tasas de aprendizaje y el tiempo. El uso de tasas bajas, en general, lleva a aprendizaje lento, con un alto riesgo de quedar atrapado en un mínimo local. Por otra parte, el uso de las tasas de aprendizaje grandes acelera los tiempos de entrenamiento, pero se corre el riesgo de que el algoritmo oscile alrededor de la solución sin ser capaz de estabilizar la misma.

Los resultados obtenidos en esta tesis muestran que el método propuesto es una alternativa interesante para el aprendizaje. Los objetivos propuestos para el nuevo método se cumplen de manera satisfactoria. El método propuesto es capaz de adaptarse al modelo AFPN y lograr el aprendizaje con las mismas características que el algoritmo Backpropagation. Al mismo tiempo, evita los problemas generales que este algoritmo tiende a tener, mientras que tiene una convergencia más rápida que el algoritmo Backpropagation.

7.1. Trabajo a futuro

Del *ejemplo 3* de los experimentos, sabemos que se pueden resolver problemas más complejos, es decir, con mayor número de reglas, por consecuencia más estados en los que se puede obtener el aprendizaje. Por consecuencia nos llevaría a verificar hasta que punto nuestro algoritmo sigue siendo estable y así decidir que mejoras se le podrían realizar.

Disponemos de extender nuestro trabajo a través de más enfoques evolutivos con el fin de encontrar el mejor para el aprendizaje. También estamos en condiciones de modificar nuestro algoritmo AWEA para lograr mejores resultados al probar diferentes tipos de mutación, cruza y selección.

Bibliografía

[1] Jie Yuan, Chang Liu, Bo Jiang, Yugang Shan, and Wenli Shang. A Knowledge Representation Method for Modeling Rule-Based Systems. *Intelligent Control and Automation (WCICA), 2010 8th World Congress, Jinan, China*, pages 1585 – 1589, July 2010.

[2] Dong-Her Shih, Hsiu-Sen Chiang, and Binshan Lin. A Generalized Associative Petri Net for Reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1241 – 1251, September 2007.

[3] Meimei Gao, MengChu Zhou, Xiaoguang Huang, and Zhiming Wu. Fuzzy Reasoning Petri Nets. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 33:314 – 324, May 2003.

[4] Victor R. L. Shen. Knowledge Representation Using High-Level Fuzzy Petri Nets. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 36:1220 – 1227, November 2006.

[5] Xiaoou Li and Wen Yu. Fuzzy Knowledge Learning via Adaptive Fuzzy Petri Net with Triangular Function Model. *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress, Dalian, China*, 1:4249 – 4253, June 2006.

[6] Liangbing Feng, Masanao Obayashi, Takashi Kuremoto, and Kunikazu Kobayashi. *Construction and Application of Learning Petri Net.* Physical Sciences, Engineering and Technology. Intech Open Science - Open Minds, August 2012.

[7] Victor R. L. Shen, Yue-Shan Chang, and Tony Tong-Ying Juang. Supervised and

Unsupervised Learning by Using Petri Nets. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40:363 – 375, March 2010.

[8] Maikel León, Gonzalo Nápoles, Ciro Rodriguez, María M. García, Rafael Bello, and Koen Vanhoof. *A Fuzzy Cognitive Maps Modeling, Learning and Simulation Framework for Studying Complex System*, volume 6687 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011.

[9] Alejandro Pena Ayala. *Sistemas basados en Conocimiento: Una Base para su Concepción y Desarrollo*. Instituto Politécnico Nacional, México, D.F., Centro de Investigación en Computación del IPN, January 2006.

[10] Rosario Aldana and Roberto Vázquez. Sistema Experto: Diagnóstico y Tratamiento Genérico de Asma Bronquial. Tesis de maestría, Universidad Veracruzana, 2002.

[11] Alo Kenneth, Alo Richard, Korvin Andre, and Kreinovich Vladik. Spinal Cord Simulation for Chronic Pain Management: Towards an Expert System. *Pain and Health Management Center, USA. In 4th. World Congress on Expert Systems: Application of Advanced Information Technologies*, pages 156 – 164, 1998.

[12] Lorena Chavarría Báez. *Verificación y Análisis de Base de Reglas Activas*. Tesis de doctorado, Centro de Investigación y de Estudios Avanzados del IPN, Departamento de Computación, December 2008.

[13] Luis Valencia Cabrera and Manuel García-Quismondo. Introducción a la Inteligencia Artificial. Technical report, Universidad de Sevilla, Escuela Técnica Superior de Ingeniería Informática, 2011.

[14] Matilde Inés Césari. *Estrategias de Análisis y Exploración de Datos como Soporte a la Adquisición de Conocimiento*. PhD thesis, Instituto Tecnológico de Buenos Aires, Desarrollo de Sistemas Expertos, 2006.

[15] Tadao Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541 – 580, 1989.

[16] Amilcar Meneses Viveros, Sergio Chapa Vergara, and Hugo García Monroy. PetrA: Herramienta para la Modelación y Simulación de Redes de Petri. Tesis de maestría, Centro de Investigación y de Estudios Avanzados del IPN, Departamento de Computación, January 2002.

[17] Joselito Medina Marín. *Desarrollo de reglas ECA en Bases de Datos, un enfoque de Red de Petri*. Tesis de doctorado, Centro de Investigación y de Estudios Avanzados del IPN, Departamento de Computación, October 2005.

[18] Wil c.d. Aalst. *The Applications of Petri Nets to Workflow Management*. PhD thesis, Eindhoven University of Technology, Department of Mathematics and Computing Science, 1997.

[19] Br. Acosta Eleiny. Razonamiento Aproximado Basado en Lógica Difusa: Fundamentos y Ejemplos de Aplicación. Tesis de licenciatura, Universidad Centroccidental Lisandro Alvarado, Barquisimeto, Venezuela, 2012.

[20] Carlos Alejandro Villasenor Lozano. Modelado Difuso Neuronal con Algoritmo de Aprendizaje Estable. Tesis de maestría, CINVESTAV-IPN, Departamento de Control Automático, April 2003.

[21] Jair Cervantes Canales. Representación y Aprendizaje de Conocimiento con Redes de Petri Difusas. Tesis de maestría, CINVESTAV-IPN, Departamento de Control Automático, March 2005.

[22] Sergio Antonio Pérez Moo. Modelado y Control de Sistemas Híbridos con Redes de Petri Difusas y Redes Neuronales. Tesis de maestría, CINVESTAV-IPN, Departamento de Control Automático, June 2002.

[23] Xiaoou Li, Wen Yu, and Felipe Lara Rosano. Dynamic Knowledge Inference and Learning under Adaptive Fuzzy Petri Net Framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 30(4):442 – 450, 2000.

[24] Aleksander I. Backpropagation in non-feedforward networks. *eural Computing Architectures:The Design of Brain-Like Machines*, pages 74 – 91, 2003.

[25] Coello Coello Carlos Artemio and Santana Quintero Luis Vicente. Una Introducción a la Computación Evolutiva y algunas de sus Aplicaciones en Economía y Finanzas. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, pages 3 – 26, December 2006.

[26] Pedro Gonzáles García. *Aprendizaje Evolutivo de Reglas Difusas para Descripción de Subgrupos*. PhD thesis, Universidad de Granada, Departamento de Ciencias en Computación E Inteligencia Artificial, 2007.