UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES.

CAMPUS ARAGÓN

"TÉCNICAS Y MÉTODOS PARA LA REPRESENTACION Y MANIPULACION DEL CONOCIMIENTO EN CIENCIAS COMPUTACIONALES"

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

CRISTHOPER FELIPE ZAMPAYO VÁZQUEZ

ASESOR:
ING. ANTONIA NAVARRO GONZÁLEZ

SAN JUAN DE ARAGÓN, ESTADO DE MÉXICO **2004**

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

JEFATURA DE CARRERA DE INGENIERÍA EN COMPUTACIÓN

OFICIO:ENAR/JACO/0486/04.

ASUNTO: Asignación de Jurado

LIC. ALBERTO IBARRA ROSAS
SECRETARIO ACADÉMICO
P r e s e n t e .

Por este conducto me permito presentar a usted el nombre de los profesores que sugiero integren el Sínodo del Examen Profesional del alumno **Cristhoper Felipe Zampayo Vázquez**, con el trabajo de tesis "Técnicas y Métodos para la Representación y Manipulación del Conocimiento en Ciencias Computacionales".

PRESIDENTE:	ING. ROBERTO BLANCO BAUTISTA
VOCAL:	M. EN C. JESÚS DÍAZ BARRIGA ARCEO
SECRETARIO:	ING. ANTONIA NAVARRO GONZÁLEZ
SUPLENTE :	M. EN C. MARCELO PÉREZ MEDEL
SUPLENTE:	M. EN C. CARLOS OLIVER MORALES

Quiero subrayar que el director de tesis es la Ing. Antonia Navarro González, el cual está incluido con base en lo que reza el reglamento de Exámenes Profesionales de esta Escuela.

Sin otro en particular, me es grato enviarle un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARÁ EL ESPÍRITU"
San Juan de Aragón, Edo. de México, septiembre 6 del 2004.
EL JEFE DE CARRERA

M. EN C. JESÚS DÍAZ BARRIGA ARCEO

c.c.p. Lic. Ma. Teresa Luna Sánchez.- Jefa del Departamento de Servicios Escolares.
Ing. Antonia Navarro González Asesor.
Interesado.
JDA*vjd

AGRADECIMIENTOS

Geist-Deux, Ich danke Dir daB Du mich liebtest und uns auf dem richtigen Weg hieltest.

Agradezco al pueblo de México, porque son los mexicanos quienes pagan la educación de los estudiantes de la UNAM. Pues realmente no existe educación gratuita.

A todos los académicos, administrativos y compañeros estudiantes que conforman esta institución, porque todos somos el espíritu vivo de la UNAM.

A la Ing. Antonia Navarro por asesorarme en la presente tesis.

A todos mis amigos de la ENEP Aragón y la UAM iztapalapa, que gratamente nos acompañamos en el viaje de la búsqueda del conocimiento.

DEDICATORIAS

A mis padres: Felipe y Guadalupe, por apoyarme hasta el límite de su propio abismo en la empresa de lograr mis sueños. Por amarme incondicionalme y enseñarme la esencia del núcleo familiar.

A mis hermanas: Analiliana, Carmen, Cinthya, Abigail, Elizabeth, que con su amor, celos, cuidados, virtudes y defectos, me infundieron seguir adelante.

A mis sobrinos, Gabriel y Pablo II, que con sólo verlos me hacen feliz.

A mi familia en general, porque aunque el sin sentido de la vida nos distancie, los lazos de sangre son más fuertes y siempre terminan por unirnos.

A Antonio por apoyarme en mis problemas existenciales, brindarme la mano en mi objetivo de ser filósofo. Y simplemente porque sé que cuando te necesite, ahí estarás.

A mis viejos amigos del CCH oriente, en especial Sergio, Norma Angélica y Sandra. Porque su amistad aún sigue vigente.

A quien era, pero realmente nunca fue en mi vida, aunque ahí está.

A quien primeramente fue una ente subjetivo de mi universo material, y después se transformo a un ente del mundo espiritual; mundo de las ideas del que nos habló Platón.

A quien una tarde se despidió del mundo dando su último suspiro, y con su adiós me hizo comprender mi original camino.

ÍNDICE

INTRODUCCIÓN

El desarrollo de las computadoras electrónicas a lo largo del siglo XX se caracterizó como uno de los eventos más trascendentales para la ciencia y la tecnología. Primero en la forma de prototipos de gran tamaño, luego como sofisticados equipos de cómputo en universidades e institutos de investigación, finalmente bajo la forma de pequeñas y versátiles máquinas personales que comenzaron a ponerse al servicio individual a partir de los años setenta del siglo pasado, las computadoras electrónicas se han convertido desde entonces en una herramienta muy útil para el hombre.

La llegada de la computadora abre paso a la consolidación de la «Ciencia Computacional», que es relativamente una disciplina nueva, cuyo progreso ha venido acompañado del surgimiento de una serie de áreas y campos de investigación que la conforman, y al origen de especialistas dedicados tanto al mejoramiento de los equipos de cómputo como al desarrollo de nuevas aplicaciones en el ámbito científico, industrial y en la vida común. Pocas disciplinas de la actividad humana han requerido y ocupado tantos esfuerzos y tantas mentes, no sólo para su puesta en marcha sino en su utilización diaria, como lo ha requerido la «Ciencia Computacional». Por consiguiente, cada día son más y mejores las técnicas y métodos con los cuales la computadora electrónica trabaja y elabora resultados satisfactorios, no obstante, aún falta mucho por hacer.

El objetivo de este trabajo es determinar cuáles son las técnicas y los métodos empleados en la representación y manipulación del conocimiento, en qué consiste cada uno de ellos y ver algunos de los tipos de problemas en donde se aplican. La representación y manipulación del conocimiento son temas tratados con profundidad en la Inteligencia Artificial, sin embargo, no sólo se restringen a esa área, pues actualmente se encuentran vínculos con algunos otros campos que conforman a la «Ciencia Computacional».

El capítulo primero sirve como marco histórico-referencial a nuestro tema central, pues si las técnicas y los métodos empleados en la representación y manipulación del conocimiento forman parte de los temas principales de la Inteligencia Artificial, pienso

que es conveniente esclarecer algunos conceptos y tener un esquema general de las áreas de la «Ciencia Computacional» que nos permita ubicar el tema dentro de su propia coyuntura. También, como conclusión al capítulo, se ofrece una exposición sobre algunos hechos importantes que han contribuido al progreso de la «Ciencia Computacional» a lo largo de la historia.

En el segundo capítulo comenzamos por definir aprendizaje. Después pasamos a exponer los tipos básicos de aprendizaje empleados en la Inteligencia Artificial, explicando las características y la forma de funcionar de cada uno de ellos. Este tema se incluyo en la investigación, porque se relaciona con la representación y manipulación del conocimiento. Aquí sólo hacemos una exposición breve del mismo.

En el tercer capítulo se analizan las principales teorías en torno a la representación y manipulación del conocimiento, con el objeto de poder determinar su algoritmo base y proporcionar algún ejemplo de su aplicación. Es decir, el propósito es poner un problema y su solución en donde se apliquen dichas técnicas empleando un lenguaje de programación (C++, Visual Basic, etc.) que se considere más adecuado. Pero como era especialmente importante encontrar un hilo conductor, se optó en dividir el capítulo en tres partes; la primera trata todo lo relacionado con la forma de representar el conocimiento en un computador; en la segunda parte, se ven las técnicas y métodos para que un computador pueda hacer uso eficiente del conocimiento en la solución de problemas; y en la última parte se muestran las soluciones de algunos problemas mediante el empleo de métodos de manipulación y representación del conocimiento.

El último capítulo trata sobre los enfoques de representación y manipulación del conocimiento que son actualmente el paradigma de estudio de la Inteligencia Artificial, es decir, se trata de técnicas y métodos que aunque surgieron a partir de la década de los 60's del siglo XX, son hoy día los que mayor entusiasman a los programadores e investigadores de la C. C.. Entre estas técnicas encontramos a los Sistemas Inspirados en la Naturaleza y algunos sistemas de control inteligente que emplean lógica multivaluada.

Por otra parte, para evitar que esta investigación resultase puramente teórica, en todas las técnicas o métodos de representación y manipulación aquí tratados, damos por lo menos un ejemplo dónde se muestra su utilización o indican algunas de sus aplicaciones. Además, se incluye un CD que integra algunos programas que emplean las técnicas y los métodos de representación y manipulación del conocimiento descritos en los capítulos tres y cuatro.

Por los temas tratados podría parecer que este trabajo es una introducción a la Inteligencia Artificial, pero esto no es del todo cierto, pues aunque el aprendizaje, la representación y la manipulación del conocimiento son temas importantes en la Inteligencia Artificial, hay otros como los Sistemas Expertos, la Robótica y el uso de lenguajes como PROLOG o LISP, que aquí no son tratados con su debida extensión. No obstante, lo anterior no resta mérito a este trabajo y tampoco impide que pueda ser empleado como material de apoyo por estudiantes de licenciaturas relacionadas con el computo y la informática, sirviendo como un acercamiento a la Inteligencia Artificial.

CAPÍTULO 1

BREVE HISTORIA GENERAL DE LAS CIENCIAS COMPUTACIONALES (C. C.) Y DE LA INTELIGENCIA ARTIFICIAL (I. A.).

La tecnología no nos ahorra tiempo,
pero si lo reparte de otra manera.
Helman Nahr.

1.1 DEFINICIÓN DE CIENCIAS COMPUTACIONALES (C. C.)

La «Ciencia de la Computación» (Computer Science) para los países anglosajones, o "también llamada «Informática» como resultado de la fusión de los términos INFORmación y autoMÁTICA en los países de habla hispana"[1], constituyen una disciplina relativamente nueva que combina algunos de los aspectos teóricos y prácticos de: ingeniería, electrónica, física, mecánica, lingüística, biología, teoría de la información, administración, matemáticas, lógica y comportamiento humano.

Se prefiere traducir *Computer Science* como «Ciencias de la Computación» y no ciencias de los computadores, porque así se rescata el sentido de que se trata de una disciplina que se centra en todo lo relacionado al proceso de cómputo y no sólo en el hardware del computador. Por consiguiente, se define como «Ciencia Computacional» al "conjunto de conocimientos científicos y de técnicas que hacen posible el tratamiento automático de la información por medio de computadoras"[2]; entendiéndose por «información», todo el conjunto de hechos y representaciones acerca de algún conocimiento humano en cualquier dominio. Para realizar lo anterior "se requiere de tres elementos básicos que se conocen como: circuitería [hardware], programas [software] y personal humano [informáticos, ingenieros-programadores o científico de la computación]"[3], que entran en juego para aplicar, perfeccionar o producir nuevas formas de la manera cómo se representa, se procesa, se guarda y se transmite la información en la «Ciencia Computacional».

El continuo desarrollo de la «Ciencia Computacional» ha proporcionado los fundamentos académicos para categorías específicas de aplicaciones de los computadores. De esta manera tenemos una diversidad de áreas de especialización que cubren desde la programación y la arquitectura informática hasta la inteligencia artificial, la cibernética, las redes, la simulación, la solubilidad o resolución de problemas, y la robótica, por

[1] Luis Ureña et al, *Fundamentos de informática*, Ra-ma, Madrid, 1997, p. 2.
[2] George Beekman, *Computer Currents: Navigating tomorrow's Technology*, Cummings Publishing, E.U., 1994, p. 29
[3] Laura Viana. *Memoria natural y artificial*, F.C.E, México, 1990, (La ciencia desde México, 88), p. 39

mencionar sólo algunas. Y estás cada día van depurando sus conocimientos, haciéndolos más preciso y delimitados.

1.2 CAMPOS DE INVESTIGACIÓN DE LA C. C.

Aunque cada una de las diversas áreas de investigación que subsume la C. C. van muy relacionadas entre sí, y existen varios tipos de clasificación de la misma. Algunos autores suelen subdividirla en diversas áreas de acuerdo al tipo de información a tratar, y nos hablan de áreas como la 'bioinformática' y 'física computacional', en el primer caso para referirse a la información relacionada a cualquier rama de la biología que es manipulada mediante un computador, y en el segundo, para referirse a la manipulación de información de fenómenos físicos. Sin embargo, el tipo más usual de clasificación es de acuerdo a los objetivos y aplicaciones especificas que cada área de la C. C. persigue. A continuación se ofrece una breve exposición de las áreas más significativas de la C. C de acuerdo a sus objetivos y aplicaciones.

1.2.1 Estructura de datos

La denominada "estructura de datos" se encarga de dos propósitos complementarios, "el primero es identificar y desarrollar entidades y operaciones matemáticas útiles y determinar cuáles clases de problemas se solucionan usando estas entidades y operaciones. El segundo es determinar representaciones para dichas entidades abstractas e implementar las operaciones abstractas en las representaciones concretas".[4] Como resultado de lo anterior tenemos que las "estructuras de datos" van desde la representación de elementos primitivos como números enteros, caracteres, etc., hasta sencillas listas numéricas y tablas (llamadas arreglos) que en conjunto y mediante complejas relaciones forman en el núcleo de enormes bases de datos.

---

[4] Yedidyah Langsam, *Estructuras de datos con C y C++*, 2ª ed., Prentice Hall, México, 1997, p. 23

1.2.2 Teoría de la computación o computabilidad

Es la rama de las C. C. que aplica diversos conceptos de las matemáticas teóricas a problemas computacionales. Por ejemplo, la matemática discreta y la lógica simbólica permiten la búsqueda de nuevas estrategias de manipulación de datos más eficientes mediante el computador, y el diseño de algoritmos firmes para que el computador responda rápidamente de forma confiable en la realización de sus tareas mundanas, como ordenar listas, buscar nombres, calcular coordenadas geométricas, etc. Por lo tanto, la "Teoría de la computación" viene a ser el área más formal de la C. C en donde se incluyen "aplicaciones sobre resolución de modelos complejos, demostración de teoremas, computabilidad de cálculos matemáticos, solubilidad de problemas, etc.".[5]

1.2.3 Simulación

Es una área de la C. C. encargada de repetir de forma virtual (mediante el uso de imágenes, sonidos, modelos, variables controladas, etc.), algunos de los fenómenos del mundo real, se trata de una forma particular de modelado mediante un computador. Actualmente existe un gran uso de la simulación por computadora en la mayoría de las disciplinas científicas, pues son varios los fenómenos simulados mediante un computador (desde simulación de sistemas continuos, como son: un vuelo espacial[6], la estructura molecular del ADN, reproducción celular, el comportamiento de las galaxias, etc., hasta la simulación de sistemas discretos, por ejemplo, fenómenos económicos, industriales, contables o comerciales) permitiendo a los investigadores adquirir un mejor conocimiento y una metodología nueva de abordar su investigación. También la "simulación" da cabida a lo que se conoce actualmente como "realidad virtual", que consiste en un sistema que permite a uno o más usuarios ver, moverse y reaccionar en un mundo simulado por computadora.

---

[5] Luis Ureña. Ob. Cit., p.19.

[6] En el Centro Espacial Jonson, en Houston Texas (EU), los astronautas se entrenan en un simulador de vuelo, en cuyo interior manipulan todos los sintonizadores, pantallas y controles de una aeronave. El aparato responde como lo haría la nave espacial, lo que permite que los pilotos y astronautas experimenten situaciones semejantes a las de los vuelos sin el peligro o costos que éstos implican.

1.2.4 Automatización

Consiste en el diseño, desarrollo e implementación de sistemas de control automatizados, donde la totalidad o la mayor parte del proceso es dirigido por una computadora que operan con transmisión en dirección inversa (retroacción-feedback), es decir, funciona bajo un sistema de retroalimentación de estados. La "automatización" ha contribuido notablemente en los procesos de fabricación de varios productos, y tienen un fuerte vínculo con la cibernética.

1.2.5 Cibernética

El término "cibernética" deriva del griego *kubernetes*, que denota 'timonel' o 'gobernador', fue aplicado por primera vez en 1948 por el matemático estadounidense Norbert Wiener a la teoría de los mecanismos de control. La "cibernética" es un intento de producir modelos (tanto reales como matemáticos) de control de sistemas complejos por medio de transmisión en dirección inversa y de muchos parámetros. Se apoya sobre un conjunto de teorías de control y comunicación en los sistemas mecánicos, electrónicos, biológicos e incluso sociales complejos. Actualmente está vinculada al estudio de la psicología, la inteligencia artificial, los servomecanismos, la economía, la neurofisiología, la ingeniería de sistemas y a los sistemas sociales.

1.2.6 Bioingeniería

Aunque la bioingeniería consiste en la aplicación de principios de ingeniería y de procedimientos de diseño para resolver problemas médicos. Dentro de algunas de sus especialidades (como la biomecánica, la ingeniería bioquímica y la bioelectricidad) se investiga sobre sofisticados sistemas computacionales implantados dentro del mismo sistema nervioso humano y enlazados con las partes sensitivas del cerebro. De este modo, y a través de las ondas cerebrales, el hombre podrá interactuar directamente con su "anexo cibernético" a través de sus procesos de pensamiento, mejorando su rendimiento.

expandiendo sus habilidades innatas o creando otras nuevas. Incluso se pretende que el cerebro humano tendría integradas las funciones de algunos dispositivos actuales como el celular, el e-mail o la agenda. Sin embargo, entre su más importantes logros se encuentra el poder brindar la capacidad de escuchar a personas que padecían sordera total.

1.2.7 Reconocimiento de patrones (*Pattern recognition*)

Consiste en teorías y técnicas para medir el parecido entre formas y su comparación cuantitativa, es decir, se trata de un método formal para medir el parecido o similitud entre dos formas, o fenómenos que presentan cierta regularidad.

1.2.8 Ingeniería del software

Es una área de la C. C. que se acuñó hasta el año de 1968, época en la que se estimuló el interés hacia los aspectos técnicos y administrativos utilizados en el desarrollo y mantenimiento del software. La "ingeniería del software" se encarga del desarrollo de 'sistemas operativos', 'lenguajes de programación', 'herramientas de desarrollo', sistemas de información' y 'aplicaciones', más fáciles de usar y más poderosas. También de ella depende el mantenimiento del software existente.

1.2.9 Arquitectura de computadoras

Montada sobre la frontera entre el mundo del software de las ciencias de la computación y el mundo del hardware de la ingeniería en computación, la arquitectura de los computadores se relaciona con la forma en que colaboran el hardware y el software, en la realización de tareas. Por ejemplo, se investiga sobre la forma de mejorar el trabajo con varios procesadores, cuáles son los distintos medios de almacenamiento disponibles, el diseño de microprocesadores cada vez mejores, etc.

1.2.10 Multimedia e Hipermedia

Consiste en una forma de presentar información empleando una combinación de sonidos, imágenes, animación, texto y vídeo, permitiendo además, la capacidad de interactuar con el computador. Su finalidad es en su mayoría de carácter pedagógico y de entretenimiento, motivo por el cual es la base de la enseñanza asistida por computador.

1.2.11 Inteligencia artificial (I. A.)

El campo de las I. A. fue fundado por los matemáticos: John McCarthy, Alan Turing (1912-1954), Norbert Wiener (1894-1964), discípulos de Alonzo Church, Claude Shannon y Marvin Minsky. La I. A. es un área de la C. C. relacionada con el desarrollo de simulaciones de inteligencia humana y con la construcción de algunas herramientas que exhiben algunas características del comportamiento inteligente. La I. A. es una ciencia bastante extensa, es por ello que su estudio se divide en áreas mas especializadas que más adelante se trataran con cierto detalle.

1.2.12 Redes y Telecomunicaciones

La comunicación en el momento actual es uno de los campos más sobresalientes y constituye el eje directriz de nuestro mundo globalizado, tal como lo fuera la matemática y la física en la era renacentista moderna, la teología en la era medieval, o la bioquímica en el presente siglo. En efecto, las comunicaciones son el tronco de la cultura tecnológica actual, la necesidad de las instituciones de organizar y agilizar los procesos de transmisión de información han llevado a tomar a la computadora como instrumento paradigmático en el proceso de comunicación a nivel mundial. A esta área de la C. C. se le denomina "redes y telecomunicaciones", y su objetivo es brindar la posibilidad de compartir datos y recursos entre grupos de usuarios.

1.3 DEFINICIÓN DE INTELIGENCIA ARTIFICIAL.

La Inteligencia Artificial (I. A.) es una de las disciplinas más nuevas. Formalmente se inicia en 1956, cuando se acuño el término, no obstante que ya para entonces se había trabajado en ello durante algún tiempo. Sin embargo, el estudio de la inteligencia es una de las disciplinas más antiguas. Por más de 2000 años, los filósofos se han esforzado por comprender cómo se ve, aprende, recuerda y razona, así como la manera en que estas actividades deberían realizarse. La llegada y evolución de las computadoras, permitió pasar de la especulación en torno a estas facultades mentales a su abordaje mediante una auténtica disciplina teórica y experimental denominada I.A. De tal forma que la I.A. ha resultado ser algo mucho más complejo de lo que muchos imaginaron al principio.

Actualmente no existe una única definición del concepto de I. A. con la cual coincidan todos los especialitas del tema, por el contrario, existen diversas definiciones que no son aceptadas del todo. Sin embargo, para poder conceptualizar lo que es I. A. es conveniente tener una noción sobre lo que es "artificial" y lo que es "inteligencia".

Por "artificial" entiéndase "aquel producto cuyo origen es no natural, sino que fue hecho por la mano o arte del hombre".[7] Por su parte, la "inteligencia" consiste en "la capacidad de comprender, evocar, movilizar e integrar constructivamente lo que se ha aprendido y de utilizarlo para enfrentarse a nuevas situaciones".[8] Por consiguiente, la "inteligencia" implica el conjunto de todas las funciones que tienen por objeto el conocimiento como son: sensación, asociación, memoria, imaginación, entendimiento, habilidad, destreza, razón, conciencia; funciones que se adquieren por las actividades prácticas y mentales que realizan los seres humanos relacionándose con su entorno.

Una vez aclarados por separado los conceptos de "artificial" e "inteligencia", es posible ahora indicar que el término I. A. en su sentido más amplio, se refiere a la capacidad de una máquina de realizar los mismos tipos de funciones que caracterizan al pensamiento

[7] Enciclopedia Universal Danae, III v. (México. Danae, 1982), v. I, 1982, pp. 182.
[8] Enciclopedia de la Psicopedagogia, I v. (España, Océano, 1998), v. I, 1982, pp. 855

humano. Sin embargo, mejor citemos algunas de las definiciones de I. A. que algunos investigadores han dado:

- "Ciencia de la obtención de máquinas que logren hacer cosas que requerirían inteligencia si las hiciesen los humanos" (Minsky, 1968)
- "El estudio de cómo hacer que los ordenadores hagan las cosas que por el momento las personas realizan de una forma más perfecta" (Elaine, 1983)
- "Nuevo esfuerzo excitante que logre que la computadora piense... máquinas con mentes, en el sentido completo y literal" (Haugeland, 1985)
- "Es el estudio de las facultades mentales mediante el uso de modelos computacionales" (Charniak y Mc Demott, 1985)
- "El arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia" (Kurzweil, 1990)
- "Estudio de las computaciones que hagan posible percibir, razonar y actuar a un artefacto hecho por el hombre" (Winston, 1992)
- "Rama de la ciencia computacional preocupada por la automatización de la conducta inteligente" (Luger and Stubblefield, 1993)
- Es una metodología que estudia el uso de la computadora para imitar el comportamiento inteligente propio del hombre: razonamiento, visión, aprendizaje, etc. (Ureña, 1997)
- "Es un área de investigación que tiene por objetivo el desarrollo de técnicas computacionales inspiradas en la observación de los mecanismos exitosos aplicados por la naturaleza para la solución de sus problemas, y en conceptos estudiados en otras áreas de la ciencia como la biología, la sicología, la ecología, etc." (Peña, 2003)

De acuerdo a lo anterior, podemos darnos cuenta que la I. A. implica el crear entes que simulen la inteligencia, es decir, específicamente su objetivo es desarrollar (a) sistemas que piensan como humanos, y que por ende, actúan como humanos o; (b) sistemas que piensan racionalmente, y que por ende, actúan racionalmente. Por lo cual la I. A. ha tomado dos caminos fundamentales: la investigación psicológica y fisiológica de la naturaleza del pensamiento humano, y el desarrollo tecnológico de sistemas informáticos cada vez más complejos.

1.4 RAMAS DE ESTUDIO DE ESTUDIO DE LA I. A.

La I. A. es una ciencia bastante extensa, es por ello que generalmente se suele dividir o clasificar de acuerdo con alguna de las siguientes dos formas:

1) Por tipos de tareas
2) Por campos de aplicación-investigación

Independientemente del tipo de clasificación que se tenga, es substancial considerar que las técnicas y los métodos de representación y manipulación de la información disponible son de extrema importancia, pues de ellos depende la responsabilidad de lograr eficazmente el objetivo propuesto por cada rama de la I. A..

Una clasificación de acuerdo con el tipo de tareas que la I. A. aborda es la siguiente:

A) De tareas generales:

- Percepción: Visión, Fonemas.
- Lenguaje Natural: Comprensión, generación y traducción.
- Razonamiento de sentido común.
- Control de robots.

B) De tareas formales:

- Juegos: Ajedrez, Backgammon, Damas.
- Matemáticas-Lógicas: Geometría, Cálculo Integral, demostración de teoremas.

C) De tareas expertas :

- Ingeniería: Diseño, Localización de fallas, Planeamiento.
- Análisis Científico.
- Diagnóstico Médico.
- Análisis Financiero.

Una clasificación por campos de aplicación-investigación que la I. A. Abarca, nos da una serie de ramas que van desde el procesamiento de la información, el reconocimiento de modelos, la robótica, hasta los sistemas expertos. Enseguida se presenta una breve descripción de las ramas de la I. A. por medio de este tipo de clasificación.

1.4.1 Aprendizaje automático o programación automática

Se encarga de investigar y desarrollar la manera en que un programa pueda aprender por si sólo al adquirir nuevos conocimientos basándose tanto en los hechos que tiene la *base de conocimiento*[9] (modelos), como en nuevos hechos (experiencias), para relacionarlos y obtener resultados que antes no formaban parte del sistema. Es decir, que el programa computacional tenga la capacidad para incrementar por si mismo su *base de conocimiento* y mejorar sus herramientas procedimentales para aprender de la experiencia. También otro de sus objetivos versa sobre la manera de lograr que un programa genere automáticamente otros programas que resuelvan problemas anteriormente especificados por el usuario y en apoyar al programador en cualquier parte del proceso de programación, ya sea en el diseño evaluación o depuración.

1.4.2 Sistemas expertos (S. E.)

También denominados sistemas basados en el conocimiento, y consiste "en desarrollar software que imita el comportamiento de un experto humano en la solución de un problema"[10], para hacerlo se debe almacenar conocimiento de expertos para un campo determinado y la solución a un problema se da mediante deducción lógica de conclusiones.

1.4.3 Visión por computadora

Estudia el reconocimiento de objetos a través de dispositivos que sean capaces de interpretar imágenes con ayuda de procesos exactamente definidos. La "visión por computadora" esta muy relacionada con el reconocimiento de patrones, el tratamiento de imágenes y la robótica. Su objetivo primordial consiste en proporcionar a los ordenadores una herramienta útil para la interpretación de los objetos captados y la comprensión de lo

[9] También llamado "espacio de búsqueda", y consta de una colección de información que incluye hechos (representada generalmente de forma declarativa) y un sistema (que es un conjunto de procedimientos) que permite determinar y modificar las relaciones entre los hechos.
[10] Luis Ureña, Ob. Cit, p. 17

que visualizan mediante la distinción de niveles de iluminación (que proporciona sombras), orientación (que suministra contornos) y reflexión (que provee límites de color).

1.4.4 Robótica

Su objetivo es diseñar y desarrollar máquinas que sean capaces de realizar procesos mecánicos y manuales mediante la interacción de un sistema de control y un sistema sensorial con el que cuentan, permitiéndoles así, responder a los cambios que surgen en su entorno.

1.4.5 Procesamiento del lenguaje natural (P. L. N.)

Se centra en la elaboración de programas que permiten a un ordenador o computadora "comprender" la información escrita o hablada, y generar resúmenes, realizar traducciones entre idiomas, responder a preguntas específicas o redistribuir datos a los usuarios interesados en determinados sectores de esta información. En esos programas es esencial la capacidad del sistema de generar frases gramaticalmente correctas y de establecer vínculos entre palabras e ideas. La investigación ha demostrado que mientras que la lógica de la estructura del lenguaje, su sintaxis, está relacionada con la programación, el problema del significado, o semántica, es mucho más profundo, y va en la dirección de una auténtica inteligencia artificial.

1.4.6 Búsquedas inteligentes u optimas

Un problema puede tener una gran cantidad de soluciones, lo anterior crea la necesidad de desarrollar ciertos mecanismos que ayuden a minimizar el tiempo de localización de la solución deseada o más conveniente. De manera que las búsquedas inteligentes (llamadas así porque disminuyen el tiempo de respuesta y el espacio de recorrido) consisten en técnicas básicas que recorren una base de datos en busca de todas las posibles soluciones, para encontrar una que se adapte a los hechos conocidos y que resulte ser la más óptima.

1.4.7 Representación del conocimiento

Consiste en la elaboración de métodos y técnicas cada vez más eficientes que permitan organizar los conocimientos que el sistema va a utilizar para poder dar solución a la diversa gama de problemas que se le presentan. Esta representación se hace de acuerdo con los tipos de problemas a solucionar y al criterio del programador.

1.4.8 Programas de cálculo estratégico (DARPA)

DARPA (*Defense Advanced Research Projects Agency*) se trata de un proyecto de investigación desarrollada por los Estados Unidos, cuyo objetivo es desarrollar sistemas inteligentes aplicables a aspectos bélicos y seguridad nacional, por consiguiente, algunas de las aplicaciones militares que DARPA mejora continuamente son: el ayudante de piloto o navegación, sistemas computarizados de radar, gestión de batalla y rastreo de movimiento y señales por satélite artificial.

1.4.9 Computación Suave (*Soft Computing*)

La "computación suave" combina diferentes técnicas modernas de I. A. como redes neuronales, lógica difusa, algoritmos genéticos y razonamiento probabilístico, ésta última incluyendo algoritmos evolutivos, sistemas caóticos, redes de opinión y, aunque solo parcialmente, teoría de aprendizaje. No obstante, conviene aclarar que la "computación suave" no es una mezcla con estos ingredientes, sino una disciplina en la cual cada componente contribuye con una metodología distintiva para manejar problemas en su dominio de aplicación que, de otra forma, se tornarían irresolubles. Su objetivo es aumentar el "coeficiente intelectual" de las máquinas dándoles la habilidad de imitar a la mente humana con mayor similitud. De tal forma que en lugar de confiar en las habilidades del programador, un verdadero programa de "computación suave" aprenderá de su experiencia por generalización y abstracción, emulando la mente humana tanto como pueda, especialmente su habilidad para razonar y aprender en un ambiente de

incertidumbre, imprecisión, incompletitud y verdad parcial, propios del mundo real. Como resultados de la "computación suave" tenemos actualmente los denominados "sistemas inteligentes híbridos" como son: los neuro-difusos, los difuso-genéticos, los neuro-genéticos y los neuro-difusos-genéticos.

Sin importar cualquiera que sea la tarea que desenvuelva la I. A., por sus objetivos puede considerarse como parte de la ingeniería o de la ciencia, es decir, por una parte tenemos que "el objetivo ingenieril de la I. A. es resolver problemas reales, actuando como un armamento de ideas acerca de cómo representar y utilizar el conocimiento, y de como ensamblar sistemas. Por otra lado, el objetivo científico de la I. A. es explicar varios tipos de inteligencia"[11], o sea, determinar qué ideas acerca de la representación del conocimiento, del uso que se le da a éste, y del ensamble de sistemas explican distintas clases de inteligencia. Por consiguiente, en la actualidad la Inteligencia Artificial es considerada como una de las áreas de mayor importancia en la Ciencia Computacional.

1.5 ANTECEDENTES HISTÓRICOS DE LA C.C.

Varios acontecimientos influyeron en la construcción de lo que hoy se reconoce bajo el término de "ciencia computacional", y aunque el término por sí mismo fue póstumo al surgimiento de la computadora, dar una historia de la C. C. implicaría dar una reconstrucción de todas las áreas de conocimiento que le aportan algo. Sin embargo, la anterior tarea es muy ostentosa y se sale de los objetivos propios de este trabajo, por eso a continuación se ofrecen sólo algunos de los acontecimiento más importantes que han influido en el desarrollo de la C. C.

Año	Acontecimiento
1641	El matemático francés Blaise Pascal construye la primera calculadora mecánica nombrada "pascalino".

[11] Henry Mishkoff, *Inteligencia Artificial*, Anaya multimedia, México, 1976. p. 67

Año	Acontecimiento
1822	Michael Faraday construye los dos primeros motores de electricidad.
	Charles Babbage construye el Motor de diferencias, que calcula logaritmos.
1832	Charles Babbage desarrolla el principio de la Analytic Engine, la cual es la primer computadora mundial que puede ser programada para resolver una amplia variedad de problemas lógicos y computacionales.
1843	Ada Lovelace publica sus propias notas con su traducción del ensayo de L. P. Menabrea, sobre la *Analytical Machine de Babbage*.
1847	George Boole publica sus primeras ideas de lógica simbólica. Desarrolló estas ideas en su teoría de la lógica binaria y aritmética, que es aún la base de la computación moderna.
1854	Un telégrafo eléctrico es instalado entre París y Londres.
	George Boole publica su ensayo *An Investigation on the Laws of Thought* donde expone un modelo del razonamiento lógico y simbólico seguido para el pensamiento.
1876	El teléfono de Alexander Graham Bell, recibe la patente norteamericana número 174465.
1879	Frege Gottlob publica "La conceptografía", obra en la que inventó muchas notaciones simbólicas (cuantificadores-variables), estableciendo así las bases de la lógica matemática moderna (notación para el razonamiento mecánico).
1888	Heinrich Hertz experimenta con la transmisión de lo que ahora conocemos como ondas de radio.
	Es introducida la primera cámara fotográfica comercial de rollo.
1890	Herman Hollerith procesa los resultados del censo de E. U. utilizando tarjetas perforadas.
1895	Guglielmo Marconi realiza la primera transmisión radial.
1896	Hollerith establece la *Tabulating Machine Company*, que se convertirá en la IBM
1899	Se realiza la primera grabación magnética de sonido sobre un cable y una delgada tira de metal.

Año	Acontecimiento
1900	David Hillberth introduce el "método directo" en el cálculo de variaciones y presenta un programa para las matemáticas del siglo XX que incluye una lista de los 23 problemas más apremiantes en la *International Mathematics Conference*, celebrada en París.
1904	John A. Fleming patenta la válvula de vacío (diodo), estableciendo un mejor escenario para la radiocomunicación.
1906	Lee de Forest agrega una tercera válvula al diodo de Fleming para crear una válvula de vacío tri-electrodo.
1908	El científico británico Campbell Swinton describe un método de exploración electrónica y uso del tubo de rayos catódicos para la televisión.
1911	*Tabulating Machine Co.* Y otras dos compañías se unen para formar *CTR-Calculating, Tabulating and Recording Co.*
	El científico holandés Kamerlingh Onnes de la Universidad de Leiden descubre la superconductividad.
1913	Henry Ford introduce el primer método automatizado de producción: la línea de ensamblado.
1915	El empleo de microchips es presagiado por el físico Manson Benedicks al descubrir que el cristal de germanio puede ser usado para convertir la corriente alterna a corriente directa.
1919	Los físicos norteamericanos Eccles y Jordan inventan el circuito de conmutación electrónica flip-flop.
1920	La palabra robot (derivada de la palabra checa para trabajo programado) es utilizada por primera vez por el escritor Karel Capek.
1923	Vladimir Kosma Zworkin, ofrece la primera demostración de un tubo de cámara de TV electrónica, usando un invento mecánico de transmisión.
1927	La IBM instala su primera oficina en México, en ese año proveyó a Ferrocarriles Nacionales de un procesador de datos de registro unitario.
1928	John Von Neumann presenta el teorema de mínimos-máximos, el cual será ampliamente usado en los programas de juegos.
1929	Señales de televisión a color son transmitidas exitosamente.

Año	Acontecimiento
1929	Es introducida la radio FM
	El departamento de Estadística en México –antecedente del Instituto Nacional de Estadística, Geografía e Información (INEGI)- utiliza sistemas de IBM con oficinas instalada hace dos años, para la clasificación contable. Gracias a lo anterior puede obtener por primera vez de manera rápida los resultados de su censo agropecuario realizado en territorio mexicano.
1930	Vannevar Bush y sus colegas en el *Massachusetts Institute of Technology* (MIT) diseñan y construyen el Analizador Diferencial, computador analógico que resuelve varias ecuaciones diferenciales.
	Konrad Zuse, en Alemania, comienza la construcción de mejores máquinas calculadoras.
1931	Reynold B. Johnson, un profesor de preparatoria en Michigan, concibe una forma de registrar resultados de exámenes de opción múltiple por conductividad de las marcas de lápiz en las hojas de respuesta. Posteriormente IBM compraría la tecnología (1931).
	El estadounidense Vannevar Bush construye la primera máquina de calcular con componentes electrónicos.
	Kurt Gödel publica su teorema el cuál lleva su nombre "Teorema de Gödel", y es reconocido como el más importante de todas las matemáticas, pues demuestra que el conjunto de enunciados verdaderos de la aritmética no es recursivamente enumerable (o no es finitamente axiomatizable).
1935	IBM introduce la máquina calculadora de tarjetas perforadas 601 y una máquina de escribir eléctrica.
1936	Konrad Zuse determina que programas compuestos por combinaciones de bits pueden ser almacenados y llena una solicitud de patente en Alemania para la ejecución automática de cálculos incluyendo una "memoria de combinación".
1936	Turing mientras era todavía un estudiante, publicó un ensayo titulado *On Computable Numbers* (Sobre números calculables), con el que contribuyó a la lógica matemática al introducir el concepto teórico de un dispositivo de cálculo que hoy se conoce como la máquina de Turing.

Año	Acontecimiento
1937	La tesis Church-Turing, desarrollada independientemente por Alonzo Church y Alan Turing, establece que todos los problemas a resolver por un ser humano son reducibles a un sistema de algoritmos, o todavía más simple, que la inteligencia de una máquina y la inteligencia humana, son equivalentes. Claude Elwood Shanon, en su tesis plantea por primera vez el poder aplicar a la electrónica los conceptos de la teoria de Boole, además definió la unidad de información conocida como bit.
1940	Jonh V. Atanasoff y Clifford Berry, construyen una computadora electrónica, conocida como ABC. Es la primera computadora electrónica, pero no es reprogramable, sino que funciona únicamente con el programa con el cual se diseño. El alemán Konrad Zuse finaliza el computador Z2. Utiliza relevadores telefónicos en lugar de circuitos lógicos mecánicos.
1943	Colossus, la primera computadora totalmente electrónica se desarrolla en Gran Bretaña para descifrar los códigos militares alemanes, utilizaba 2000 tubos de vacío y ocupaba una habitación entera.
1944	Se pone en funcionamiento la MARK I. El Dr. Howard Aiken en la Universidad de Harvard (E.U.), la presenta como la primera máquina procesadora de información. Funcionaba eléctricamente, las instrucciones e información se introducen en ella por medio de tarjetas perforadas, sus componentes trabajan bajo en principios electromecánicos.
1945	Konrad Zuse desarrolla el primer lenguaje de alto nivel: Plankalkul.
1946	Se presenta la ENIAC como la primera computadora electrónica, construida por J. P. Eckert y J. W. Mauchly en la Universidad de Pensilvania, (E.U.) y se le llamó ENIAC (*Electronic Numerical Integrator And Computer*), ó Integrador numérico y calculador electrónico.
1947	Se presenta la EDVAC (Eletronic Discrete-Variable Automatic Computer, computadora automática electrónica de variable discreta) desarrollada por Dr. John W. Mauchly, John Presper Eckert Jr. y John Von Neumann. Primera computadora en utilizar el almacenamiento de información (datos e instrucciones) usando un código especial llamado notación binaria, por lo cual, era la primera computadora que podría ser utilizada para varias aplicaciones cargando y ejecutando el programa apropiado.

Año	Acontecimiento
1947	Invención del transistor por William Bradfor Schockley, Walter Hauser Brittain y John Ardeen.
1948	Se realizan una serie de conferencias sobre Cibernética en un congreso celebrado en París. El matemático estadounidense Norber Wiener instaura el término de Cibernética al publicar una serie de obras de gran importancia como son Cybernetic (1948), The Human Use of Human Beings (1950), Nonlinear Problems of Random Theory (1958) y God and Golem, Inc. (1964).
1950	Se presenta la ACE PILOT. Turing tuvo listos desde 1946 todos los planos de lo que posteriormente sería conocido como ACE Pilot (Automatic Calculating Engine) que fue presentado públicamente en 1950. La ACE Pilot estuvo considerada por mucho tiempo como la computadora más avanzada del mundo, pudiendo realizar operaciones tales como suma y multiplicación en cuestión de microsegundos.
1951	UNIVAC I (1951) Desarrollada por Mauchly y Eckert para la Remington Rand Corporation. Primera computadora comercial utilizada en las oficinas del censo de los Estados Unidos. En 1952 fue utilizada para predecir la victoria de Dwight D. Eisenhower en las elecciones presidenciales de los Estados Unidos
1952	En 1952 Grace Murray Hoper una oficial de la Marina de EE.UU., publicó su primer ensayo sobre autoprogramadores (*Compilers)* y desarrolló el primer compilador, acto que le valió ser nombraba directora e ingeniero de sistemas de la División *Univac de la Sperry Rand Corp.*
1954	Aparece FORTRAN (FORmula TRANslator) lenguaje de programación de alto nivel desarrollado por John Backus y un grupo de programadores de IBM para aplicaciones científicas.
1955	Surgen las primeras teoría sobre el procesamiento del lenguaje Información Processing Language I (IPL-I)
1955	Surgen los General Problem Solver (GPS), los llamados Sistemas Cognitivos o primer lenguaje de IA (IPL-II). En esta época Newell y Simon no se preocupaban cómo obtener la respuesta correcta en sistemas de computo, sino por qué los sistemas llegaban a dar las respuestas que daban.
1956	Se inicia la investigación más profunda sobre I. A., al darse el seminario de Dartmouth College sobre I. A..

Año	Acontecimiento
1957	Noam Chomsky escribe *Syntatic Structures*.
1958	Se instala la primer computadora en el país, se trata de la IBM-650 ubicada en el Centro de Cálculo Electrónico (CCE) en la Facultad de Ciencias de la Universidad Nacional Autónomas de México. Los encargados del proyecto fueron los doctores Carlos Graef y Alberto Barajas.
	Se fabrica el primer circuito integrado (chip) desarrollado por Jack Kilby y Robert Noyce.
	Aparece LISP, lenguaje de programación para I. A.
1959	Aparece COBOL (Common Business Oriented Language), un lenguaje compilador diseñado para aplicaciones de negocios. Desarrollado por el gobierno federal de los Estados Unidos y fabricantes de computadoras bajo el liderazgo de Grace Hopper. Es el más utilizado por los procesos administrativos.
1960	APL (A Programming Language) fue desarrollado por Kenneth Inverson a mediados de la década de 1960 para resolver problemas matemáticos. Este lenguaje se caracteriza por su brevedad y por su capacidad de generación de matrices y se utiliza en el desarrollo de modelos matemáticos.
1962	Una compañía norteamericana coloca en el mercado los primeros robots industriales.
	El primer departamento de Ciencias de la computación, establecido en la Universidad Purdue, ofrece un doctorado.
	D. Murphy y Richard Greenblatt desarrolan el editor de texto TECO para la computadora PDP1 del MIT.
	Frank Rosenblatt publica Principles of Neuro-dynamics, en donde define el perceptrón, un elemento procesador simple para redes neuronales.
1963	Aparece BASIC (Beginners All-purpose Symbolic Instruction Code), el lenguaje de programación interactivo más popular en la década de los 70. Es un lenguaje de propósito general. Desarrollado por John Kemeny y Thomas Kurtz en "Dartmouth College". Existen numerosas versiones, algunas son compiladores y otras son intérpretes.
1963	Se introduce el código ASCII (American standard code for information interchange)

Año	Acontecimiento
1964	Es desarrollado por IBM la primer aplicación para generar informes comerciales o de negocios llamada RPG (Report Program Generator).
1965	Se consolida la Ciencia Cognitiva. Y se inicia el proyecto de sistema experto DENDRAL en la Universidad de Stanford, su función es la de un interprete de espectrogramas de masas..
1968	Douglas Englebart presentó en una conferencia el sistema ratón-icono.
	Se utiliza el término "Ingeniería del Software" en una reunión que se convocó en Garmisch, Alemania Oriental. A partir de entonces se estimula el interés hacia los aspectos técnicos y administrativos utilizados en el desarrollo y mantenimiento del software.
1969	Se anuncia el Intel 4004, procesador no comercial
1970	El dispositivo floppy es introducido para almacenar datos en las computadoras.
	Surge PASCAL, desarrollado por el científico suizo Niklaus Wirth en 1970 y diseñado para enseñar técnicas de programación estructurada. Es más fácil de utilizar en comparación al lenguaje ensamblador o máquina.
	Se produce el Intel 8008, primer procesador de 8 bits
	Surge Lenguaje C, desarrollado a principios de la década de los 70 en Bell Laboratories por Brian Kernigham y Dennis Ritchie. Ellos necesitaban desarrollar un lenguaje que se pudiera integrar con UNIX, permitiendo a los usuarios hacer modificaciones y mejorías fácilmente.
1971	La primer calculadora de bolsillo es presentada: puede sumar, restar, multiplicar y dividir.
1973	Aparece el primer video juego y simulador deportivo llamado Pong.
	Alain Colmerauer presenta un diseño de PROLOG que se popularizará.
	Se desarrollan por parte de Vinton Cerf el protocolo de internet (IP) y el protocolo de control de transmisión (TCP) como parte de un proyecto de la Agencia de Programas Avanzados de Investigación (ARPA, por sus siglas en inglés) del departamento Estadounidense de Defensa, dando como resultado la funcionalidad optima de la Red ARPANET, antecedente directo de lo que años más adelante se convertiría en Internet.

Año	Acontecimiento
1974	Edward Shortliffe presenta como tesis doctoral MYCIN (Stanford), sistema experto para el diagnóstico médico.
	Es desarrollado el primer robot industrial controlado por computadora.
	Aparece el primer sistema operativo llamada CP/M, que luego dos años más tarde será remplazado por MS-DOS.
1975	Benoit Mandelbrot escribe *Les formes de les objects fractals, hasard et dimenson*, su primer ensayo sobre geometría fractal. Las formas fractales servirán para modelar fenómenos caóticos en la naturaleza y generar imágenes realista de objetos de la naturaleza por computadora.
1976	Se crea en la UNAM en Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) a cargo del Dr. Tomás Garza Hernández, cuyo antecedente es el Centro de Calculo Electrónico. El IIMAS tiene un fuerte impacto en la investigación desarrollada en la UNAM.
1977	Es construida por Steven Jobs y Stephen Wozniak la computadora Apple I. Sin embargo, ese mismo año la que sale al mercado como la primera computadora personal de forma ensamblada es la Apple II.
1979	Se crea ADA, un lenguaje de computadora desarrollado para las fuerzas armadas.
	Se popularizan los videojuegos en el mercado, y comienza la industrialización de los mismo, convirtiéndose en una de las áreas más rentables.
1980	Surge robots con 5 o 6 grados de libertad empleados en la industria.
	Aparece SMALLTALK desarrollado por el Centro de Investigación Xerox, con este lenguaje surge el enfoque de Programación Orientada a Objetos.
	C evoluciona dando lugar al Lenguaje C++, desarrollado por *Bell Laboratories*. C++ introduce la programación orientada al objeto en C con lo cual se convierte en un lenguaje extremadamente poderoso y eficiente.
1981	Microsoft (fundada en 1977) presentó el sistema operativo DOS.
	IBM, da a conocer su computadora personal (PC) con un procesador Intel 8088, el sistema operativo MS-DOS, 16Kb de memoria principal, un teclado y un puerto para conectar la reproductora.

Año	Acontecimiento
1982	Los equipos de reproducción de disco-compacto son colocados en el mercado por vez primera.
	Sale al mercado el Lotus 1-2-3
1983	Se da a conocer la existencia de la DARPA, un sistema de I. A. utilizable en la defensa nacional.
	Apple introduce LISA, la primera computadora comercial con un sistema operativo puramente gráfico. Pero no obtiene mucho éxito en el mercado.
1984	Apple presentó el sistema operativo Macintosh, con una gran aceptación en el mercado por ofrecer una interfaz gráfica del usuario con sus iconos, menús de bajada, ventanas y el popular dispositivo de señalamiento mouse.
	Durante el periodo de 1983-86, en el Departamento de Ingeniería Eléctrica del CINVESTAV-IPN, el Dr. Alfonso Guzmán Arenas fundó y dirigió la Sección de Computación. Dicha sección se convirtió posteriormente en el Departamento de Computación, y en 1984 fue la primera en ofrecer un programa de doctorado en Informática en México. Ahí se desarrollaron investigaciones sobre computación paralela, inteligencia artificial, sistemas expertos, tecnología de software, y programación visual.
	Son lanzados discos ópticos para el almacenamiento de datos de computadora.
	IBM saca un chip de banco de datos tipo RAM de un megabyte.
1985	Japón se consolida como líder mundial en perfeccionar la producción y aplicación de la robótica.
	Microsoft, una empresa fundada en 1975 por William H. Gates III y Paul Allen, lanza Windows, un sistema operativo que ampliaba las presentaciones de MS-DOS al incorporar una interfaz gráfica de usuario.
1986	El procesamiento de imágenes y reconocimiento de patrones mediante computadora forma parte ya de los métodos de investigación de las ciencias.
1987	Japón desarrolla el sistema de identificación de huellas digitales.
1988	El mercado de los sistemas expertos se amplia, tan solo representa 800 millones de dólares para los E. U.

Año	Acontecimiento
1988	Se reduce el costo del PC y aumenta el costo del software.
	La tendencia de analógico a digital se generaliza en los procesos industriales.
En la década de los 90's	Las naciones desarrolladas confían fuertemente en las armas inteligentes. Se da una gran evolución en técnicas de reconocimiento de patrones.
	Son introducidos al mercado los sistemas caseros de videojuegos de más de 16 bits, área donde se aplican sofisticadas técnicas de programación y graficación por computadora.
	Un estudiante finlandés llamado Linux Torvalds comienza el desarrollo del sistema operativo de licencia libre llamado Linux. Para 1997 la versión Red Hat de Linux consigue el premio al mejor sistema operativo.
	En agosto de 1995 sale al mercado Windows 95, lo que significó un nuevo vuelco en la línea de los SO de Microsoft. Se trata de un entorno multitarea con interfaz simplificada y con otras funciones mejoradas.
	La UNAM crea en el año de 1995 la "Licenciatura en ciencias computacionales" para atender la necesidad creciente de gente especializada en el ámbito de la computación con una orientación hacia aplicaciones científicas. Desde entonces la licenciatura se imparte en las instalaciones de la Facultad de Ciencias en Ciudad Universitaria.
	Cada vez las computadoras tienen mayor capacidad de almacenamiento y más velocidad de procesamiento. Al mismo tiempo, se crea una amplia gama de software permitiendo un sin fin de aplicaciones. Es un hecho que la computadora pasa a ser una herramienta para una multiplicidad de tareas empresariales, educativas y en el hogar. Y desempeña un papel importante en el proceso de globalización.
	En 1997 sale el sistema operativo de Macintosh MAC OC8, al mismo tiempo la empresa se consolida como líder en edición y producción de medios digitales.
	En 1998 Microsoft saca al mercado el Windows 98, desde entonces se consolida como una empresa líder en los sistemas operativos.
	Se establecen plenamente Redes de Comunicación de distancia mundial, con envío de imágenes, grandes cantidades de datos, audio y video. World Wide Web. Surgen tecnologías de comunicación inalámbrica: Cómputo Móvil.

Año	Acontecimiento
Principios del siglo XXI	El empleo de la computadora electrónica se expande tanto en actividades industriales, científicas, educativas, como en la vida cotidiana. Es tan reconocida la utilidad de la computadora, que la mayoría de los programas educativos a nivel profesional integran por lo menos una materia relacionada con el computo. Los "Sistemas de Tiempo Real" a menudo son utilizados como dispositivos de control en aplicaciones dedicadas: como control industrial, de experimentos científicos, procesamiento de imágenes médicas, etc... Intel y Hewlett-Packard han definido conjuntamente una nueva tecnología de arquitectura llamada EPIC, denominada así por la habilidad del software de extraer el máximo paralelismo (potencial para trabajar en paralelo) del código original y explícitamente describirlo al hardware. Por consiguiente, Intel y HP se han basado en esta tecnología EPIC para definir la arquitectura del set de instrucciones (ISA) que será incorporada en la arquitectura final del microprocesador de 64-bits de Intel. Esta nueva tecnología trae consigo un modus operandi innovador, ya que haciendo uso de su tecnología EPIC, y combinando paralelismo explícito con conceptos y técnicas avanzadas de arquitectura de computadoras llamadas especulación y predicación superará todas las limitaciones de las arquitecturas tradicionales. Intel anunció el nuevo nombre para su primer microprocesador IA-64 de nombre clave Merced, Itanium. Se afirma que tendrá un rendimiento para redes suficiente como para sacarle una ventaja a los RISC de un 20-30% en este rubro. Intel espera que el nuevo procesador opere a una frecuencia de reloj alrededor de los 800 MHz y que entregue entre 45-50 SPECint95 y 70-100 SPECfp95 (base).

De lo anterior, podemos observar que la computadora obtuvo una significativa evolución a lo largo de todo el siglo XX, hasta convertirse en una herramienta trascendental para el desarrollo científico, tecnológico y cultural del hombre. Es indudable que en las próximas décadas tendrán lugar más y mejores equipos de hardware con mayor velocidad, más eficiencia y capacidad de almacenamiento, menor tamaño y un bajo costo. Nuestras computadoras actuales nos parecerán tan obsoletas como hoy nos lo parecen las calculadoras mecánicas de antaño. Por su parte, el software cada vez ofrece una multiplicidad de aplicaciones con mejor precisión y seguridad. Al mismo tiempo, día a día se ve una creciente necesidad de gente más especializada en el ámbito de la computación.

CAPÍTULO 2

TIPOS DE APRENDIZAJE EN CIENCIAS COMPUTACIONALES

No hay nada repartido más equitativamente en el mundo que la razón: todos están convencidos de tener suficiente.

René Descartes.

2.1 DEFINICIÓN DE APRENDIZAJE

El aprendizaje puede definirse como "el proceso mediante el cual un ente adquiere conocimiento y la manera en que lo aplica para resolver problemas en su entorno".[12] Básicamente el aprendizaje humano suele dividirse en dos tipos que son: el repetitivo y el cognoscitivo[13]. El primero de ellos, también denominado aprendizaje de tipo E-R (estímulo-respuesta), es el más básico y rudimentario; consiste en el almacenamiento de información adquirida por memorización de procedimientos y de información vía la experiencia, por ejemplo: el abecedario, las tablas de multiplicar, el manejar un automóvil, y demás procesos que involucran una serie repetitiva de pasos. Por su parte, el aprendizaje cognoscitivo es el más difícil de aplicar en una computadora, debido a que requiere de analizar, organizar, correlacionar e inferir elementos específicos del conocimiento que permiten, entre otras cosas, poder definir una clase a través de los atributos que las caracterizan, determinar propiedades y generalizar hechos.

Actualmente existen computadoras que toman datos e información (lo cual es ciertamente una forma de aprendizaje) que ordenan, buscan, clasifican, simulan, comparan, reconocen y hacen cálculos lógicos y aritméticos, para producir nueva información. Así, tenemos de alguna manera que cualquier programa escrito equivale a un tipo de enseñanza de nuestra parte y a un aprendizaje por parte de la máquina. Sin embargo, los computadores siguen siendo mejores para manejar datos que conocimiento. Por el contrario, el cerebro humano, que no es muy bueno para almacenar y recordar hechos en comparación a un computador, es excelente para manipular la información que incorpora las «relaciones» entre los hechos, es decir, el conocimiento.

A pesar de no saber con exactitud cómo nuestro cerebro almacena y manipula conocimientos, los investigadores de la C. C. apoyándose de la psicología, la neurología, la lógica, los lenguajes formales, las matemáticas discretas, y otras disciplinas, trabajan en

---

[12] Tim Hartnell, *Inteligencia artificial: conceptos y programas*, Anaya Multimedia, México, 1993, p. 232

[13] No hay que confundir los métodos por los que se adquiere el conocimiento con los tipos de aprendizaje que hay, pues los primeros son mecanismos que permiten que el aprendizaje se dé.

proporcionar a los computadores, formas de adquirir y almacenar conocimiento de sentido común sobre el mundo real. Y en crear técnicas para representar el conocimiento. o sea. pretenden que las computadoras sean capaces de aprender por sí mismas de una forma eficiente como lo hace cualquier persona con buena salud. El esquema 2.1 muestra una clasificación de los tipos de aprendizaje empleados en la C.C.

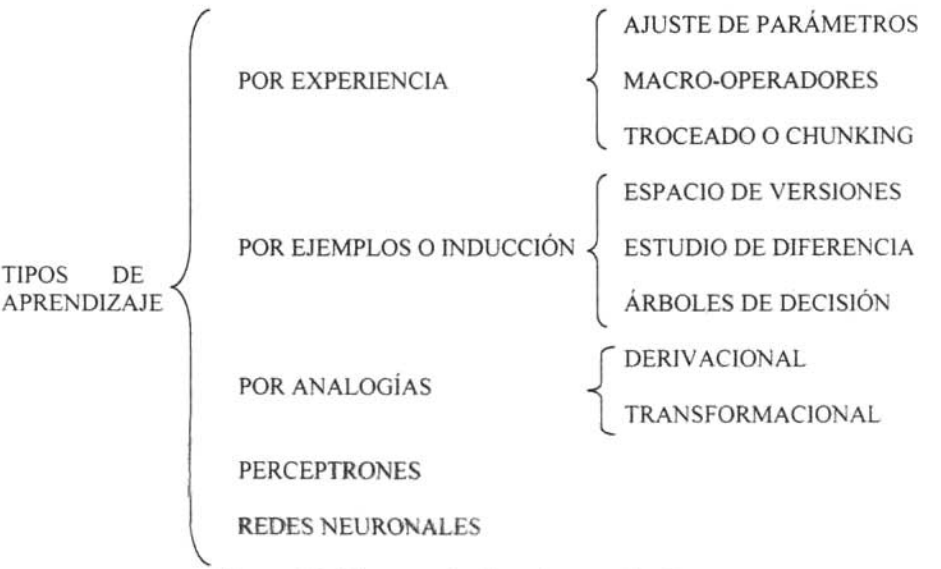


Figura 2.1. Esquema de tipos de aprendizaje

Enseguida se presenta en qué consiste cada tipo de aprendizaje.

2.2 APRENDIZAJE POR EXPERIENCIA

Consiste en que la computadora, hasta cierto punto, pueda aprender más por experiencia que por haber sido completamente alimentada por el programador. Para lo cual es necesario dar una serie de reglas y descripciones que conformarán un conjunto de información llamada antecedente, que servirá para optar y realizar una determinada acción a la cual se le denomina consecuente. Así, para que una computadora adquiera conocimientos a través de la experiencia es necesario implementar un programa que

simplemente sea capaz de agregar información a su memoria organizada con anticipación, en patrones que el programador ha encontrado útiles.

En este tipo de aprendizaje, es necesario que en la determinación del antecedente, el programador especifique la definición o descripción correcta bajo el contexto que se esté manejando, es decir, se tienen que evitar todo tipo de ambigüedades de conocimiento y dar una ubicación clara del tema. Precisamente, teniendo en cuenta la importancia de una buena ubicación en cada situación para poder distinguir si se trata de objetos similares, es indispensable que los procedimientos cuenten con la capacidad suficiente para aprender tanto de los ejemplos como de los resultados obtenidos. Además, el procedimiento debe ser capaz de descartar datos que no formen parte del mismo contexto y modificar su antecedente mediante el ajuste de parámetros, o ampliar su antecedente mediante el agregado de información, o de un conjunto de posibles instrucciones llamadas macro-operadores seleccionar la mejor de todas, con base en la forma en que el programador considere más conveniente.

2.2.1 Aprendizaje por medio del ajuste de parámetros

Consiste en usar una función de evaluación:

$$F = (C_1)(p_1) + (C_2)(p_2) + \ldots + (C_N)(p_N)$$

Que combina la información recibida de diferentes fuentes (p_1, p_2,...,p_N), para formar un único resumen estadístico, es decir, el objetivo es tener una función cuyo único valor refleje la solución más adecuada al problema que se intenta resolver. La dificultad principal es saber asignar a priori los coeficientes (C_1,C_2,...,C_N) y los pesos o ponderaciones (p_1,p_2,...,p_N) adecuados a cada característica esencial. Una forma de hacerlo es comenzar con alguna estimación de los valores correctos, y entonces, permitir que el programa los modifique en base a su propia experiencia.

Hay dos cuestiones importantes al diseñar un programa de este tipo, el primero es determinar cuándo debe aumentar o disminuir el valor un coeficiente, y en segundo, cuánto debe cambiar el valor. Por lo que aquellas características que parezcan adecuadas deberán aumentar su peso, y aquellas que no lo hagan, lo disminuirán. Es decir, los coeficientes de los términos que ayuden a predecir el final deben aumentar, mientras que deben disminuir los coeficientes que sean poco predictivos. Para que lo anterior resulte fácil, se necesita que el programa tenga retroalimentación. Por ejemplo, un programa de clasificación de patrones utiliza una función de evaluación para clasificar una entrada y obtener la respuesta adecuada, entonces los términos que hayan contribuido a alcanzar ese objetivo deben aumentar su peso, con lo cual el programa se retroalimenta de su propia información generada al elevar cada peso.

La ejecución del programa[14] que se muestra en la figura 2.2 es un ejemplo de aprendizaje por ajuste de parámetros.

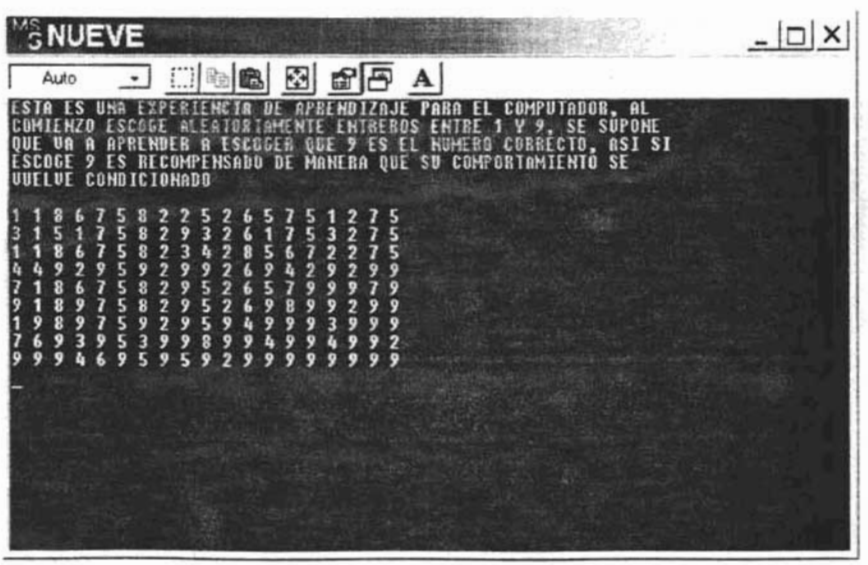


Figura 2.2, Programa con Ajuste de parámetro

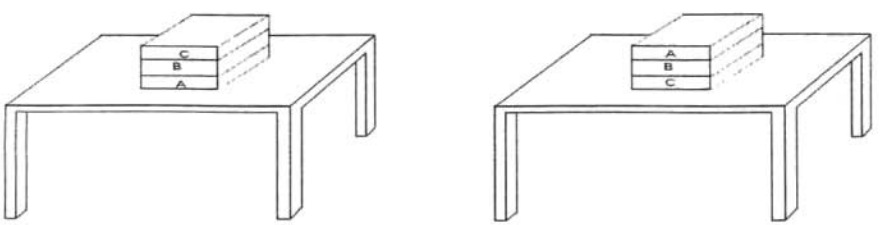
[14] El código fuente del programa se incluye en el anexo de este trabajo.

El programa consiste en que inicialmente el computador mediante una función realice diez listas de 20 números seleccionados aleatoriamente del 1 al 9, pero por medio de un ajuste de parámetro de la función principal, el programa genera más el número que nosotros queramos durante cada salida producida, en nuestro caso hemos seleccionado el 9. Así, conforme la computadora genere el número deseado, ira modificándose para al final obtener puros 9, como si el programa aprendiera tras cada iteración que esa es la respuesta correcta y que los demás números deben ser descartados.

Las mayores limitaciones de este proceso de aprendizaje son debidas a que no se usa ningún conocimiento acerca de la estructura del problema y las relaciones lógicas entre componentes del mismo, por ello, se suele emplear en combinación con otros tipos de aprendizaje para diseñar sistemas de control automatizado de mayor eficiencia.

2.2.2. Aprendizaje con macro-operadores

Se denomina macro-operador a una secuencia de acciones que se pueden tratar en conjunto. Su uso trata de evitar volver a realizar cálculos que ocupen mucho tiempo de respuesta y en minimizar el número de instrucciones disponibles. Para ilustrar, supóngase que se tiene un brazo mecánico con el que se intenta resolver problemas en el dominio del mundo de los bloques. El problema se plantea dando las condiciones iniciales y el objetivo, en nuestro caso supongamos que tenemos una situación como la que se muestra en la figura siguiente:



a) Iniciales b) Objetivo

Figura 2.3, Condiciones

Vemos que en las condiciones iniciales (inciso a de la Figura 2.3) tenemos tres bloques apilados sobre una mesa. A cada bloque se les asignó una letra de forma aleatoria para distinguirlos entre sí. El objetivo es que el bloque de abajo (o sea A) quede hasta la cima de la pila, y que el bloque de arriba (o sea C) quede como la base (inciso b de la figura 2.2). Para proceder, las condiciones iniciales se describen mediante una precondición, y el objetivo mediante una postcondición, separadas ambas por una flecha (=>) que denota implicación. De acuerdo con lo anterior, tendríamos lo siguiente:

SOBRE(C,B) ∧		SOBRE(A,B) ∧
SOBRE(B,A) ∧	=>	SOBRE (B,C) ∧
SOBRE(A,MESA)		SOBRE(C,MESA)

Inicialmente los operadores disponibles son:

ES A
ES B
ES C
APILAR (A,B)
APILAR (B,C)
DESAPILAR (C,B)
DESAPILAR (B,A)
BAJAR (A)
BAJAR (B)
BAJAR (C)
AGARRAR (A)
AGARRAR (B)
AGARRAR (C)

La secuencia de operadores más corta para solucionar este problema es la siguiente:

ES (C), DESAPILAR (C,B), BAJAR (C), ES (B), DESAPILAR (B,A),
APILAR (B,C), AGARRAR(A), APILAR (A,B)

Después de cada resolución, el componente de aprendizaje toma el plan computado y lo almacena en un macro-operador con las precondiciones y postcondiciones correspondientes. Para que en el futuro, este nuevo operador pueda utilizarse para

solucionar un subproblema con las mismas pre y postcondiciones. Además, si se generalizan los macro-operadores antes de almacenarlos, el programa será capaz de hacer uso de él en situaciones no idénticas sino similares. Por ejemplo, la generalización del macro-operador de solución del problema del brazo mecánico es:

ES (x3), DESAPILAR (x3,x2), BAJAR (x3), ES (x2), DESAPILAR (x2,x1), APILAR (x2,x3), AGARRAR(x1), APILAR (x1,x2)

El cual tiene la siguiente precondición:

SOBRE(x3,x2) $\wedge$ SOBRE(x2,x1) $\wedge$ SOBRE(x1,MESA)

Y como postcondición tiene:

SOBRE(x1,x2) $\wedge$ SOBRE(x2,x3) $\wedge$ SOBRE(x3,MESA).

2.2.3. Aprendizaje mediante troceado o *chunking*

Esta técnica denominada troceado o *chunking* es un proceso similar al de los macro-operadores. Se basa en un conjunto de hipótesis específicas cognoscitivas y motivadas similares a la estructura de resolución de problemas empleada por los seres humanos.

La idea consiste en la distinción entre dos tipos de memoria:

- Memoria a corto plazo u operacional.
- Memoria a largo plazo.

Con la memoria operacional podemos recordar un conjunto pequeño de datos, pero estos pronto se ven olvidados al tratar de recordar otro dato, por ello se suele también llamarle como memoria a corto plazo. Por su parte, la memoria a largo plazo almacena información

de manera constante, que puede ser recordada en cualquier momento sin tener que olvidarse al recordar otros datos. Así, en promedio el ser humano puede recordar un número de teléfono de siete u ocho dígitos (58786728) durante unos segundos sin dificultad y con sólo haberlo visto algunas veces. Pero se le complica el recordar uno de cuarenta dígitos: 6893742696452686337876329076413696234785 4.

Para solventar la anterior limitación, se emplea la técnica denominada troceado (*chunking*), que nos permite recordar un numeró de hasta 40 dígitos si los agrupamos en trozos (chunks) de 5. Así, el número 6893742696452686337876329076413696234785 4 es más fácil de ingresar a la memoria a largo plazo si se emplea la memoria operacional para verlo como un conjunto de partes y memorizar cada una de ellas:

68937 42696 452886 33787 63290 76413 69623 47854

Bajo los anteriores supuestos, para efectuar este tipo de aprendizaje en el computador se suele implementar un sistema que cuenta de las siguientes partes:

- Memoria a largo plazo: contiene reglas u operadores a aplicar.
- Memoria a corto plazo: almacena hechos deducidos de la aplicación de reglas.
- Actividades resolutivas: incluyen razonamiento de qué estados se deben explorar y qué reglas se deben aplicar. En lugar de emplear una estrategia fija, la elección de una regla depende de la utilidad que está presente frente una situación dada.

Cuando el sistema detecta una secuencia de reglas útiles para resolver un subproblema dado, estos resultados intermedios son almacenados o troceados del conjunto de resultados obtenidos, para referencia futura. Como la resolución de problemas es uniforme, el troceado puede usarse para aprender conocimiento de control de búsqueda general, además de secuencia de operadores. Es decir, si el sistema al probar con varias reglas diferentes se percata de que sólo una conduce por un camino adecuado en el espacio de

búsqueda, entonces el programa debe poder delimitar las reglas que le ayuden a elegir a su vez los operadores de un modo más acertado en el futuro, o facilitar la solución mediante la división del conjunto de conocimiento disponible.

A diferencia de los macro-operadores, "el troceado casi siempre se aplica en dirección a cualquier estado objetivo en lugar de planearse para alcanzar un único estado objetivo o postcondición especial"[15]. No obstante, esto hace que su principal problema sea que las reglas ocupen demasiado espacio, y por lo tanto, el tiempo de análisis de aplicación de reglas es grande. Una alternativa de solución es mantener una medida de utilidad asociada a cada regla, la cual toma en cuenta la frecuencia de aplicación de la misma, así como el tiempo y memoria consumidos. Si una regla dada tiene una utilidad negativa, se la ignora, por lo tanto se mejoran los tiempos de búsqueda y acceso.

Al igual que con el aprendizaje con macro-operadores, el troceado (chunking) también suele emplearse en combinación con otros tipos de aprendizaje para diseñar sobretodo sistemas de control automatizado, sistemas expertos y robots.

2.3 APRENDIZAJE A PARTIR DE EJEMPLOS O DE INDUCCIÓN

Este tipo de aprendizaje hace empleo de la inducción, que se define como "el procedimiento que de lo particular lleva a lo general"[16], para realizar la clasificación, la cual consiste "en el proceso de asignarle a una cierta entrada concreta el nombre de una clase a la que pertenece."[17] No obstante, fundamentalmente utiliza una diversidad de búsquedas llamadas heurísticos, gracias a los cuales los procedimientos conocen las especificaciones de la clase (soluciones-métodos) o especie que se pretende sean aprendidas (conceptos-objetos).

---

[15] Tim Hartnell, *Inteligencia artificial: conceptos y programas*, Anaya Multimedia, México, 1993, p. 156
[16] Alan, Chalmers. ¿Qué es esa cosa llamada ciencia?, Siglo XXI. México, 1999, p. 28
[17] Irvin, Copi. Lógica simbólica, CECSA, México, 1989, pp. 344-350

En este tipo de aprendizaje, primeramente se deben definir las clases que emplea la clasificación, las cuales pueden describirse usando alguno de los siguientes modelos:

- Modelo estadístico: cada clase está definida por una función F que incluye las características $C_1,...,C_N$ relevantes del dominio específico ponderadas por un peso $p_1,...,p_2$

$$F = (C_1)(p_1) + (C_2)(p_2) + ... + (C_N)(p_N)$$

- Modelo estructural: cada clase está definida por una estructura compuesta por las características relevantes del dominio específico. Por lo cual, si la tarea es definir clases de animales, el cuerpo puede verse como una estructura con un conjunto de propiedades como: pigmentación, cantidad de patas, longitud, manchas, etc.

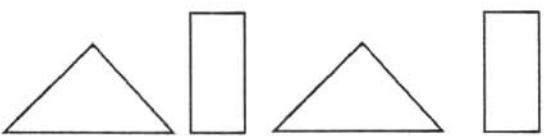
Se sabe que si las clases se definen con un modelo estadístico, el aprendizaje de conceptos se puede hacer utilizando ajuste de parámetros (apartado 2.2.1). Por el contrario, si las clases son definidas de un modo estructural, el aprendizaje de conceptos se puede hacer mediante: espacio de versiones, árboles de decisión o por estudio de diferencias. Estas técnicas tienen en común que comienzan el proceso de aprendizaje a partir de una serie de ejemplos de entrenamiento de los que se conoce su clasificación. Y dado que la tarea de construir manualmente las clases resulta un proceso difícil, lo más conveniente es que el programa de clasificación incluya las definiciones de sus propias clases.

2.3.1 Espacios de versiones (*version spaces*)

Consiste en un seguimiento de ejemplos acertados y no acertados, en la solución de un problema bajo ciertas especificaciones, sin verse afectado por el orden en que se presentan los ejemplos. Además, en lugar de describir un único concepto, mejor se mantiene un conjunto de descripciones posibles, hasta arribar a la definición del mismo mediante el empleo combinado de un heurístico de enlace necesario HEN (que determina lo que se

busca porque es necesario que exista), y otro de enlace prohibido HEP (que determina lo que no se busca). Estas dos heurísticas son consideradas las más relevantes en el aprendizaje por espacio de versiones, puesto que en ellas existen las descripciones esenciales de una clase, además, la información que de ellas se obtiene sirve para encaminar a la elaboración de deducciones ó conclusiones más acertadas.

Lo primero es entrenar al procedimiento con un elemento acertado de la clase por aprender mediante la elaboración de una descripción preliminar, esta descripción debe tender a crecer como resultado de toda la información que se proporciona de la especie, llegando así a obtener lo que se conoce como «modelo de progresión». Es importante señalar que los ejemplos no acertados son de suma importancia, ya que de ellos se derivan los detalles, especificaciones y características esenciales de la clase; en este sentido se dice que el procedimiento aprende o se entrena mejor con ejemplos no acertados que con los acertados. Para ilustrar la manera de como procedería un programa con este tipo de aprendizaje, consideremos dos pares de figuras, dos triángulos y dos rectángulos como se muestran a continuación.



Ahora supongamos que estamos entrenando a un procedimiento sobre la forma de cómo acomodar las cuatro figuras de tal modo que se forme un cuadrilátero con ellos y que los rectángulos no se toquen entre sí. El procedimiento tiene que comenzar su aprendizaje con una especificación acertada (figura 2.4, I), para después continuar con la serie de especificaciones de prueba hasta conseguir una que sea similar a (I) de la figura 2.4. Cuando la encuentra, entonces se dice que se ha obtenido una especificación acertada. La figura 2.4 ilustra sobre algunas de las especificaciones que se generan, donde (I) es la única acertada, y (II, III, IV, V, VI) son todas erróneas.

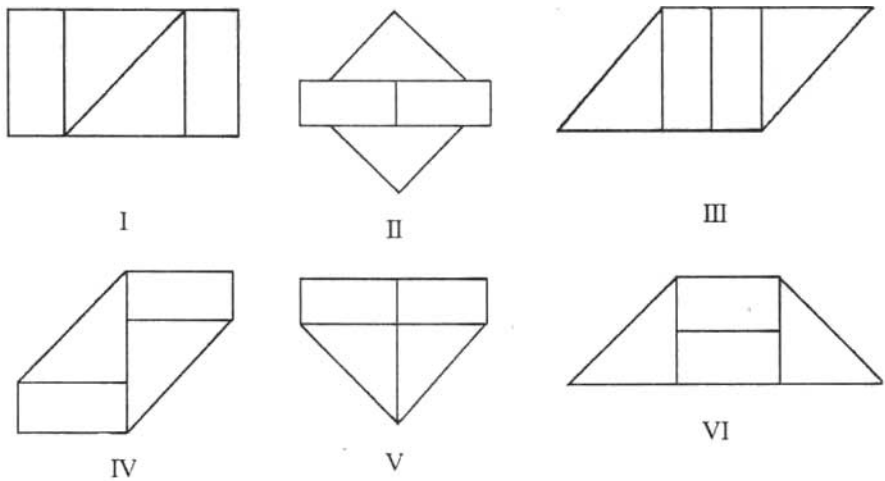


Figura 2.4, Especificación de espacios de versiones en la obtención de un cuadrilátero sin que los rectángulos se toquen.

El procedimiento utiliza los ejemplos no acertados para observar porqué no cumplen con lo que se requiere, aunque entre los rectángulos y triángulos forman de algún modo un cuadrilátero y algunas otras figuras sólidas, no es exactamente eso lo que se pide que aprenda, porque no cumple con las especificaciones; sin embargo, estos ejemplos son de ayuda para el procedimiento, porque gracias a ellos obtiene información del acomodo correcto de las figuras. De forma que en 2.4 los enlaces necesarios podrían ser que los triángulos se encuentren unidos uno a otro por su perímetro más grande, y que los rectángulos se encuentren tanto separados entre ellos, como unidos a alguno de los triángulos. Cuando se considera ésta heurística, ningún ejemplo que no tenga los enlaces necesarios no será reconocido como cuadrilátero sólido.

A medida que se procesan los ejemplos de entrenamiento se va refinando la noción de dónde se encuentra el objeto destino. Existe un subconjunto del espacio de conceptos llamado espacio de versiones constituida por todas las descripciones por las que se va pasando a medida que se procesan los ejemplos.

2.3.2 Por estudio de diferencias

Este tipo de aprendizaje hace empleo únicamente de HEP (*heurístico de enlace prohibido*), debido a que es utilizado cuando un ejemplo de los no acertados cuenta con un enlace que el modelo progresivo no tiene, es así como el enlace del ejemplo no acertado se transforma en un enlace prohibido, y por lo tanto, no debe ser encontrado. Como ejemplo de esta prohibición de enlace, observemos el punto (II) de la figura 2.4. Vemos que el acomodo de los triángulos cumple con los enlaces necesarios (unidos por su perímetro más amplio) no obstante al estar los rectángulos unidos entre sí, podemos considerar que se trata de un enlace prohibido, puesto que la unión de los rectángulos hace que no se cumpla con las condiciones que se piden.

La implantación del aprendizaje por estudio de diferencias tiene tres pasos:

1- Comenzar analizando una instancia conocida del concepto y construir su descripción estructural.
2- Estudiar descripciones de otras instancias conocidas del concepto para generalizar la definición hecha en el paso anterior, de esta forma, todas las instancias están incluidas en la nueva descripción.
3- Examinar descripciones de semejantes u objetos que no son instancias del concepto, para restringir la definición hecha en el paso 2. De esta manera, todos los semejantes están excluidos en la nueva descripción.

El programa lo único que debe hacer es ir descartando todas las instancias que sean diferentes al paradigma de descripción estructural, etiquetándolas como ejemplos negativos por no cumplir con los requisitos señalados. Sin embargo, un problema que puede surgir es un error en el etiquetado de un ejemplo causando un retraso en la solución. Es importante destacar que de acuerdo al problema a resolver, el programador es quién debe estipular el paradigma y las condiciones básicas de las instancias de los conceptos que intervienen. Un programa eficiente dependerá de la selección de un buen paradigma.

2.3.3 Árboles de decisión (*decision trees*)

También se le conoce como aprendizaje por árboles de identificación. Consiste en la representación de datos en forma de listas que señalan características de la clase que se esta ramificando, y en los que se opta seguir por una rama considerando las propiedades de los objetos.

Para clasificar una entrada particular, se empieza de la parte superior del árbol y con base en la solución de pregunta se va trazando un camino hasta llegar a una hoja en donde se guarda la clasificación. Por ejemplo, la figura 2.5 muestra la especificación con un árbol de decisión del concepto "coche-económico-mexicano-tres-puertas". Lo que se requiere es encontrar un objeto que forme parte de los elementos que integran al concepto (que sea subsumido dentro de dicho concepto), es decir, un objeto que tenga las propiedades siguientes: ser coche, tener bajo precio, ser de manufactura mexicana, y que tenga tres puertas.

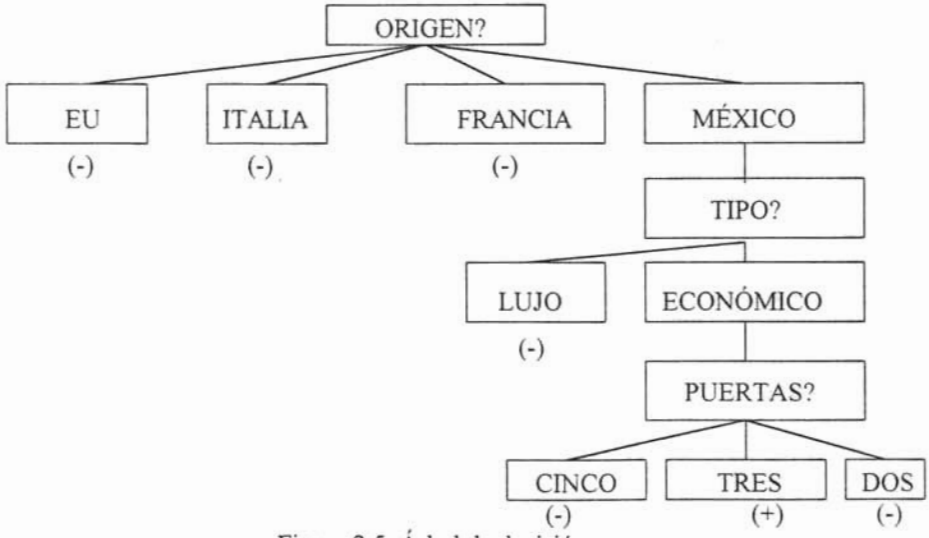


Figura 2.5, Árbol de decisión

De acuerdo con la figura 2.5, el programa tendría que construir automáticamente un árbol de decisión dadas diferentes instancias positivas y negativas del concepto destino. Así, se trata de un algoritmo iterativo, que comienza eligiendo un subconjunto aleatorio de ejemplos de entrenamiento llamado ventana. El algoritmo construye un árbol que clasifica todos los ejemplos de la ventana, y prueba con ejemplos de entrenamiento fuera de la ventana. Si todos los ejemplos son correctamente clasificados, el algoritmo finaliza. En caso contrario, se añaden a la ventana un número de ejemplos y el proceso se repite.

Como los árboles se construyen creando nodos a partir de atributos que proporcionan más información que otros, se tiene que poner hincapié en la determinación correcta de propiedades, por ejemplo, para el caso del concepto "coche-económico-mexicano-tres-puertas" probar con el atributo «color» es menos útil que el atributo «número de puertas», pues el primero no ayuda a la hora de clasificar correctamente. Así, el que un atributo o propiedad sirva para clasificar más que otros, depende de las características particulares del problema (o concepto a considerar). Cuando se llega a atributos que dividen perfectamente las instancias de entrenamiento en subconjuntos cuyos miembros participan con una etiqueta común (ya sea positivo o negativo) la ramificación ha terminado y los nodos hoja están etiquetados.

2.4 APRENDIZAJE POR ANALOGÍAS

La analogía es "una relación de semejanza que permite establecer visiones de conjunto entre realidades distintas, admitiendo una correspondencia entre conceptos aparentemente diferentes"[18]. Es una herramienta de inferencia utilizada naturalmente en nuestro lenguaje y razonamiento mediante la cual se resuelven problemas haciendo analogías con cosas que ya se conocen. Sin embargo, se trata de un proceso complejo. Por ejemplo, si tomamos la frase: el mes pasado la bolsa era como una montaña rusa. Por analogía podemos entender que la bolsa sufrió grandes fluctuaciones. Para entender la frase debemos escoger una

[18] Enciclopedia Universal Danae, III v. (México, Danae, 1982), v. I, 1982, p. 114.

propiedad clave de la montaña rusa, por ejemplo, que baja y sube muy rápido. Y darse cuenta de que el viaje físico es una analogía para la fluctuación numérica de la bolsa.
La analogía implica un proceso que no resulta fácil, pues el espacio de búsqueda es muy amplio y no debe caerse en conclusiones erróneas como: la bolsa es como una montaña rusa porque está hecha de metal.

Existen dos tipos de aprendizaje por analogía que son: la derivacional y la transformacional, en seguida se da una breve descripción de cada una de ellas.

2.4.1 Analogía derivacional

A menudo las peculiaridades implicadas en la resolución de un problema antiguo son relevantes para resolver uno nuevo. Al razonamiento por analogía que toma en cuenta el historial detallado de los episodios de resolución anteriores se le nombra analogía derivacional (figura 2.6.) Y consiste en el empleo de una derivación nueva que integra los aciertos realizados por derivaciones viejas en la solución de problemas.

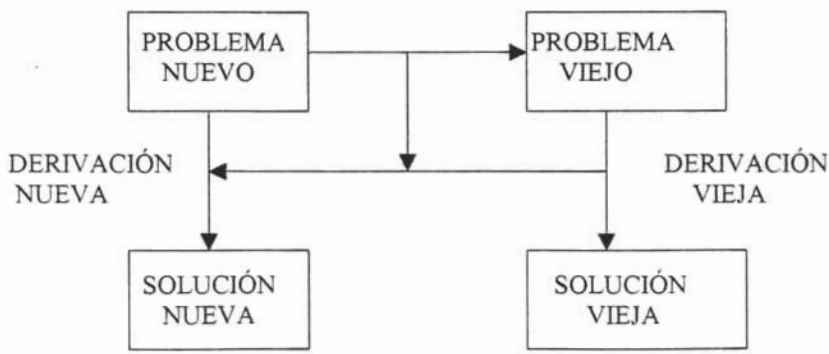


Figura 2.6, Esquema de Analogía derivacional

Por ejemplo, supóngase que se ha codificado en C++ un programa capaz de realizar clasificación de animales, ahora lo que se pide es realizar un programa que pueda traducir programas hechos en C++ a otro lenguaje de programación como PASCAL, o sea, se pide

que el computador sea capaz de traducir programas de un lenguaje de programación a otro.

Procediendo de acuerdo a una analogía derivacional, una traducción línea por línea del programa en C++ a PASCAL no resulta adecuada, sino que para obtener el programa clasificador de animales en PASCAL se debería reutilizar el análisis, es decir, las principales decisiones estructurales y de control que se tomaron al construir el programa en C++.

Una manera de modelar este comportamiento es tener implementada una función que repita la derivación previa en la solución de problemas, y que sea apta de modificarla cuando sea necesario.

2.4.2 Analogía transformacional

Este tipo de analogía consiste en transformar una solución de un problema previo en la solución del nuevo problema a resolver. La figura 2.7 nos ilustra sobre lo que se pretende hacer.

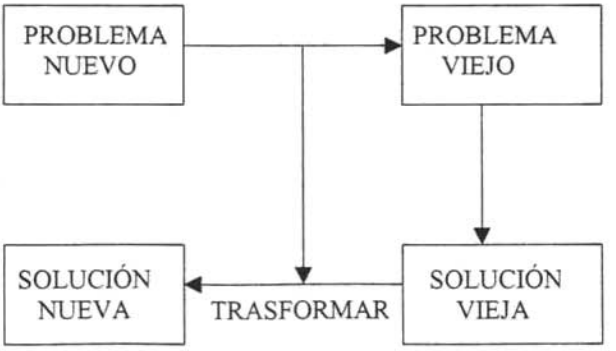
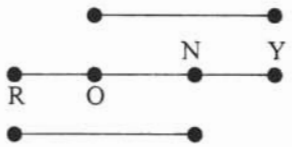


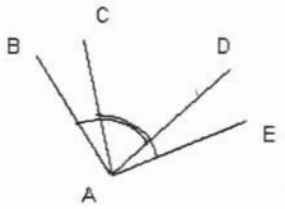
Figura 2.7, Esquema de Analogía transformacional

Por ejemplo, un programa de resolución de problemas de geometría plana que emplea demostraciones acerca de puntos y segmentos lineales, al aplicar analogía transformacional podría probar un teorema sobre ángulos.[19] Así, tendríamos lo siguiente:

Problema viejo:



Problema nuevo:



Solución al problema viejo:

A partir de:

$$RO = NY$$

Realiza la demostración de:

$$RN = OY$$

Solución:

$$RO = NY$$
$$ON = NO$$
$$RO + ON = ON + NY$$
$$RN = OY$$

Solución al problema nuevo:
Sustituyendo la noción de punto por la de línea, y la de segmento por la de ángulo:

R por AE,
O por AD,
N por AC,
Y por AB

Tenemos como solución:

$$EAD = CAB$$
$$DAC = CAD$$
$$EAD + DAC = DAC + CAB$$
$$EAC = DAB$$

2.5 PERCEPTRONES

Los perceptrones son considerados como la red neuronal más simple y más especializada. Consta fundamentalmente de:

[19] Anderson y Kline (1979) fueron los que diseñaron dicho programa. Para detalles consultar: E.G. Dougherty and Ch. R. Giardina, *Mathematical Methods for Artificial Intelligence and Autonomous System*. Prentice-Hall, E.U., 1988.

- Una neurona
- Una o más entradas binarias cuyo valor binario es 1 ó 0.
- Una serie de cajas lógicas que pueden interponerse entre las entradas y el perceptrón. Cada caja lógica actúa de acuerdo a una tabla de verdad que produce una salida (1 ó 0) para cada combinación posible de las entradas.
- Una salida igual a (1 ó 0) producida por la función umbral de la suma ponderada de las salidas de las cajas lógicas.

Y gráficamente se representa de la siguiente forma:

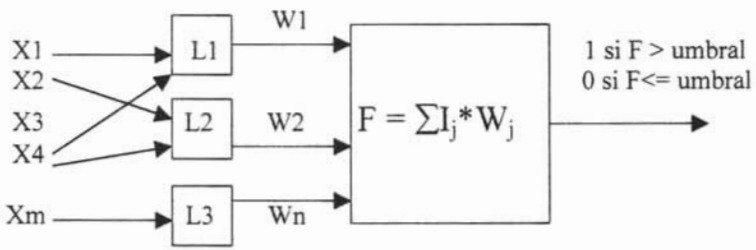


Figura 2.8, Perceptrón

Donde:

X1,X2, X3,X4,...,Xm	Son las diferentes entradas del perceptrón.
L1, L2,...,Ln	Son las cajas lógicas, aparecen entre las entradas y los pesos del perceptrón, su salida también es binaria y su función es una o más entradas y produce sólo una salida.
W1,W2,..., Wn	Representan a los pesos.
ΣL_i*W_i	Es la suma ponderada, se encarga de sumar los pesos ya relacionados con la salida de las cajas lógicas.
F	Es la salida del perceptrón, y provee como resultado un 0 ó 1 dependiendo de si la suma ponderada es mayor que el umbral. Por ejemplo, considerando el umbral Z tenemos: $F = 1$ si $\Sigma L_i*W_i > Z$, o $F = 0$ si $\Sigma L_i*W_i < Z$

Si hay Xm entradas, existen 2^m combinaciones posibles de las mismas. Por lo tanto, si este número es muy grande no puede ser atendido por una sola caja lógica y se requiere del empleo de varias.

Los perceptrones pueden clasificarse en:

- Perceptrón limitado por el orden de cajas lógicas n: donde cada caja lógica atiende n o menos entradas que pueden ser las mismas.
- Perceptrón limitado por el diámetro d: donde las entradas se distribuyen rectangularmente, cada caja lógica atiende una serie de entradas que están dentro de un círculo de diámetro d, es decir, no hay cajas lógicas que reciban la misma entrada.
- Perceptrón directo: es el más simple ya que cada caja lógica tiene una entrada por lo que su salida es la misma (equivale a perceptrón sin cajas lógicas).

El perceptrón directo es el más fácil de entrenar para que su salida sea la deseada. El algoritmo de entrenamiento de los perceptrones es el siguiente:

1- Inicializar el vector de pesos en ceros: (0,0,...,0)
2- Probar con todas las muestras hasta que el perceptrón produzca el resultado correcto, así por cada muestra:
   Si produce un 0 cuando debe producir 1: w = w + 1
   Si produce un 1 cuando debe producir 0: w = w - 1
   Si no se equivoca con la muestra no hacer nada.

Es decir, si el valor no deseado es un 0 cuando se esperaba un 1 entonces el peso de las cajas lógicas que en ese momento producen el 1 se incrementan (w = w + 1), si por el contrario, se logra un 1 cuando se requiere un 0, el peso de las cajas que producen el 1 se decrementan (w = w - 1) lo cual ayuda a hacer menos probable un resultado indeseable. Cuando el valor obtenido corresponde al esperado, no se hace nada puesto que no influyen

en el resultado final. Así, la idea consiste en ajustar los pesos cada vez que el perceptrón produce una respuesta errónea con una muestra, de manera que el error sea menos probable. Este algoritmo converge siempre y cuando la solución exista para el conjunto de muestras de entrenamiento. Por ejemplo, para entrenar un perceptrón de tipo directo (en el cual las entradas X_1,X_2 coinciden con las salidas de las cajas lógicas L_1,L_2, o sea, en el que son iguales) con el fin de que realice la operación lógica OR, tenemos que utilizar una entrada de más que siempre esté en 1 para modelar el umbral igual a (-w3) como se muestra en la figura 2.9

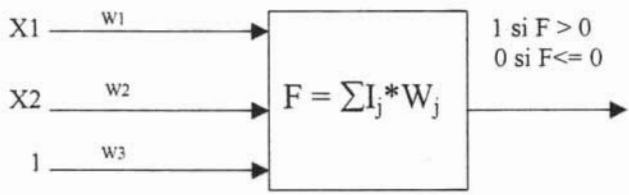


Figura 2.9, Perceptrón que realiza operación lógica OR

Y en el cual se tienen las muestras de entradas que a continuación se dan:

	Paso 1	Paso 2	Paso 3	
Muestras	$X_1 = L_1$	$X_2 = L_2$	$X_3 = L_3$	Salida Deseada
1	0	0	1	0
2	0	0	1	1
3	1	1	1	1
4	1	1	1	1

Primeramente se procede a inicializar el peso del perceptrón W = (0,0,0), después se cicla a través de las muestras hasta que la salida obtenida con cada una sea igual a la salida deseada. Así, el procedimiento es el siguiente:

En el primer paso:

- Muestra 1 tenemos (0,0,1), donde X_1 es igual a la salida deseada.
- Muestra 2, tenemos (0,0,1), donde X_1 es diferente a la salida deseada, por lo que se procede a ajustar el vector , sumando el vector peso a la muestra tomada: vector peso w = (0,0,0) + (0,0,1) = (0,0,1)
- Muestra 3 tenemos (1,1,1) donde X_1 es igual a la salida deseada.
- Muestra 4 tenemos (1,1,1) donde X_1 es igual a la salida deseada.

En el paso dos:

- Muestra 1 tenemos (0,0,1) donde X_2 es igual a la salida deseada.
- Muestra 2 tenemos (0,0,1) donde $X_2 = 0$ cuando se esperaba $X_2 = 1$, o sea X_2 es diferente a la salida deseada, por lo que se le suma al vector peso actual W = (0,0,1) el vector de la muestra (0,0,1) quedando un nuevo vector peso W = (0,1,0)
- Muestra 3 tenemos (1,1,1) donde X_2 es igual a la salida deseada.
- Muestra 4 tenemos (1,1,1) donde X_2 es igual a la salida deseada.

Y en el tercer paso:

- Muestra 1 tenemos (0,0,1) donde $X_3 = 1$, o sea diferente a la salida deseada, como se tiene un 1 en lugar de 0 lo que se procede es restar este vector muestra (0,0,1) al vector peso W (0,1,0), quedando el vector peso final (0,0,1) el cual ya no se modifica.
- Muestra 2 tenemos (0,0,1) donde X_3 es igual a la salida deseada.
- Muestra 3 tenemos (1,1,1) donde $X_3 = 1$, igual a la salida deseada.
- Muestra 4 tenemos (1,1,1) donde $X_3 = 1$, igual a la salida deseada.

Por lo tanto, el vector peso se fue ajustando conforme se desarrolla el procedimiento, y converge encontrando el vector peso W = (0,0,1).

2.6 REDES NEURONALES (RN)

Las redes neuronales son sistemas de cómputo distribuidos y paralelos inspirados en la estructura del cerebro humano. Para poder entender el aprendizaje por medio de las redes neuronales, es necesario considerar el conocimiento que se tiene sobre el funcionamiento del cerebro humano.

El cerebro humano consta de miles de millones de neuronas, cada una conectada a miles de otras neuronas en una estructura distribuida, con paralelismo masivo. Este tipo de estructura otorga al cerebro una gran ventaja en la mayoría de las capacidades perceptivas, motrices y creativas. Así mismo, una neurona es una célula de gran longitud formada por un área central engrosada que contiene el núcleo, una prolongación larga llamada axón (que transporta la salida de la neurona hasta las conexiones de otras neuronas), y unas prolongaciones arborescentes más cortas llamadas dendritas (que son protuberancias que facilitan la conexión con los axones de otras neuronas) como muestra en la figura 2.10

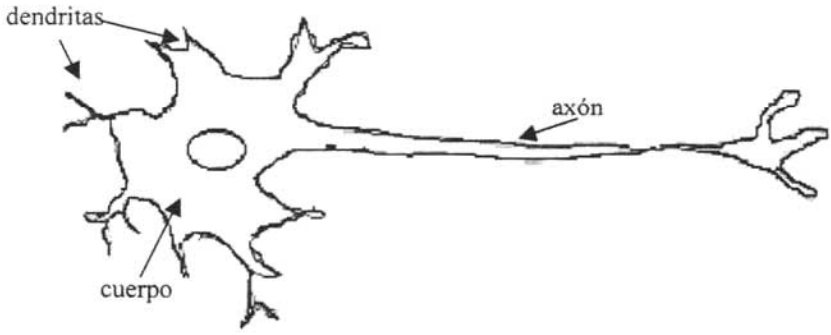


Figura 2.10, Neurona

Una neurona se encuentra inhabilitada hasta que la influencia colectiva de todas sus entradas (sinapsis) rebasen su nivel de umbral al recibir de otras neuronas algún tipo de excitación; está excitación hace dispararse a la neurona, es decir, produce una salida de potencia completa, que se manifiesta como un pulso estrecho que se desplaza del cuerpo por el axón, hasta las ramas de éste. Considerando las anteriores características se puede realizar el modelo de una neurona como se muestra en la figura 2.11.

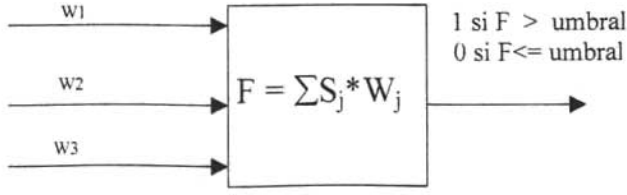


Figura 2.11, Esquema de una Neurona

Al modelo de la neurona también se le suele llamar perceptrón. En el podemos apreciar que a cada enlace se le asigna un peso determinado (w_1, w_2, w_j) los cuales modelan las propiedades de las sinapsis; la sumatoria de la función de activación modela la capacidad de combinar la influencia de todas las dendritas; el resultado de la sumatoria es comparado con el nivel de umbral, si es mayor se dice que la neurona produce una salida '1' o simplemente que la neurona ha sido excitada, por el contrario, si se produce como resultado 0, la neurona no hace nada porque queda inhabilitada.

En lugar de una sola neurona (perceptrón), una red neuronal usa una red de unos millares de neuronas. Así, una red neuronal es una colección de perceptrones conectados entre sí, cuya representación se da en la figura 2.12

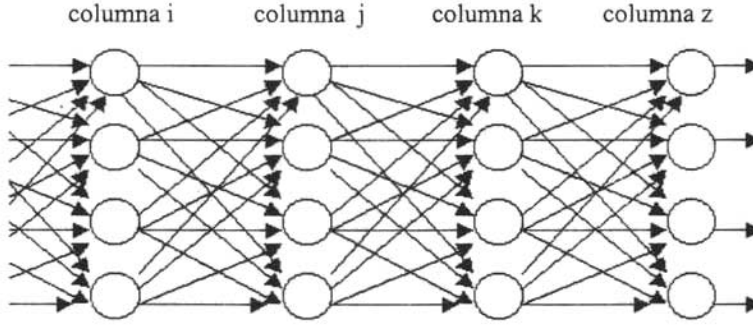


Figura 2.12, Red Neuronal

Las propiedades de la red en su conjunto dependen en gran medida de los elementos que las integran; así como de la conectividad y dinámica que exista entre ellos. El proceso de aprendizaje en las Redes Neuronales no consiste en programarlas de la manera habitual, sino en entrenarlas. Por lo cual, en vez de usar una estrategia basada en reglas, una red neuronal aprende patrones por ensayo y error, como lo hace el cerebro. Así, cuando se repiten los patrones con frecuencia, las redes neuronales desarrollan hábitos. Sin embargo, este tipo de aprendizaje puede presentar problemas en algunas aplicaciones, ya que no hay reglas claramente definidas. Cuando una red neuronal toma una decisión, no hay forma de averiguar por qué lo hizo.

Las redes neuronales almacenan la información de manera distinta que los computadores tradicionales. Los conceptos se representan como patrones de actividad entre varias neuronas, de modo que son menos susceptibles a averías de la máquina. Como la información se distribuye por la red, una red neuronal (como el cerebro humano) puede seguir funcionando aunque se destruyan algunas de sus neuronas.

Sobre el algoritmo de entrenamiento de la RN, la tarea más difícil es la asignación de los pesos (w_1, w_2, w_j) y los umbrales para que la red se comporte de la manera deseada. En la práctica la RN se entrenan con una serie de muestras de entrenamiento con la que realizan modificaciones de los pesos y umbrales de forma iterativa. Así, en cada paso se realizan pequeños ajustes a los valores de pesos y umbrales para acercarlos a los valores que proporcionarán el comportamiento deseado de la red. El entrenamiento finaliza cuando la salida obtenida sea la deseada.

Para su óptimo funcionamiento, el algoritmo de entrenamiento usa una medida de calidad sobre toda la red actual, es decir, se necesita determinar cómo se comporta la red con cada uno de los ejemplos de entrenamiento. La medida de calidad estará dada en función de los pesos y umbrales, y por lo tanto, se necesita saber qué tan buenos son los pesos y cómo se pueden mejorar; para ello se considera una función en el campo vectorial de los pesos y umbrales la cual consiste en que: se sume el cuadrado del error de cada salida, al tenerlas todas se suman las entradas, anteponiendo el signo negativo para obtener un desempeño general con un máximo en cero, expresado matemáticamente es:

$$P = - \Sigma \ (\Sigma \ (d_{SZ} - o_{SZ})^2)$$

Donde:

P, es el desempeño a obtener.

S, es un índice que fluctúa entre todas las muestras de entrada.

Z, es un índice que fluctúa para todos los nodos de salida.

d_{SZ}, es la salida deseada para la muestra de entrada s en el nodo z-ésimo

o_{SZ}, es la salida real para la muestra de entrada s en el nodo z-ésimo.[20]

Precisamente el objetivo del algoritmo de entrenamiento es obtener en cada paso, los cambios de los valores de pesos y umbrales que proporcionen la máxima mejora de la medida de calidad. Esto es lo que se denomina ascenso del gradiente (el gradiente de una función indica la dirección en la que se encuentra la máxima variante de la función). No obstante, antes de seguir con el algoritmo de entrenamiento, se debe considerar que resulta conveniente tratar a pesos y umbrales de manera similar (por esto se unifica dicho tratamiento). Así, en lugar de considerar una función de activación como:

1 si F > umbral

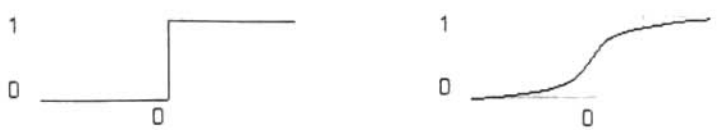
0 si F ≤ umbral

Se considera una nueva F' como:

1 si F'= F – umbral > 0

0 si F'= F – umbral ≤ 0

De esta manera, el umbral es tratado como un peso de una entrada extra que siempre está en –1. Dado que la función de activación contiene un salto (figura 2.13.a) para tratarla matemáticamente en el algoritmo a explicar, se transforma en una función suave de derivada continua (figura 2.13.b).



(a) (b)

Figura 2.13, Umbral de una Red Neuronal

[20] Patrick H. Winston, *Inteligencia artificial*, Addison-Wesley, México, 1995, p. 486.

La derivada de la función umbral F con respecto a su argumento se puede expresar como:

$$\frac{\delta F(\sigma)}{\delta \sigma} = O(1-O) \tag{1}$$

Recordemos el objetivo: se tienen una serie de pesos que se desean mejorar y se tiene un conjunto de muestras de entrada junto con la salida deseada de cada una. La idea de *ascenso del gradiente* consiste en ascender la calidad o mejora de la función P más rápidamente, mediante la alteración de todos los pesos en proporción a la derivada parcial correspondiente. El cambio a cada peso en particular, se realiza en la medida que este conduzca a reducir el error observado en la salida. En otras palabras, en cada paso, la variación de cada peso $W_{i\to j}$ (correspondiente a la conexión de un nodo de la columna i con otro de la columna j) será proporcional a la derivada parcial de la medida de calidad P con respecto al peso:

$$\Delta W_{i\to j} \alpha \frac{\delta P}{\delta W_{i\to j}} \tag{2}$$

La medida de calidad P está dada como una sumatoria sobre todas las muestras de entrada. A continuación se enfoca la atención a una muestra de entrada en particular para reducir los subíndices, sabiendo que luego se realizará la suma de los ajustes derivados de cada muestra. Ahora, para realizar la derivada parcial de P con respecto a $W_{i\to j}$ de manera eficiente, se expresa usando la regla de la cadena, mediante la variable intermedia O_j (la salida del nodo de la columna j):

$$\frac{\delta P}{\delta W_{i\to j}} = \frac{\delta P}{\delta O_j} \frac{\delta O_j}{\delta W_{i\to j}} \tag{3}$$

Dado que la salida de un nodo de la columna j, O_j, es calculada en base a las entradas (salidas de los nodos de la columna anterior i), mediante la función umbral, se expresa como:

$$O_j = F(\sum_i O_i W_{i\to j}) = F(\sigma_i) \tag{4}$$

Luego se expresa la derivada parcial de la salida O_j con respecto a $W_{i\to j}$ usando la regla de la cadena, mediante la variable intermedia delta:

$$\frac{\delta O_j}{\delta W_{i\to j}} = \frac{\delta F(\sigma_i)}{\delta O_i}\frac{\delta \sigma_i}{\delta W_{i\to j}} = O_i O_j (1-O_j) \tag{5}$$

Como el efecto de O_j sobre P se efectúa a través de las salidas de los nodos de la capa siguiente, las O_K, se puede aplicar la regla de la cadena para calcular la derivada parcial de P con respecto a O_j:

$$\frac{\delta P}{\delta O_j} = \sum_K \frac{\delta P}{\delta O_k}\frac{\delta O_k}{\delta O_j} \tag{6}$$

Cada salida O_K se determina mediante la suma de todas las entradas al nodo k, pasando el resultado a través de la función umbral F:

$$O_K = F(\sum_j O_j W_{j\to K}) = F(\sigma_K) \tag{7}$$

Se puede expresar entonces la derivada parcial de O_K con respecto a O_j usando la regla de la cadena mediante la variable intermedia delta como:

$$\frac{\delta O_K}{\delta O_j} = \frac{\delta F(\sigma_{Ki})}{\delta O_K}\frac{\delta \sigma_{Ki}}{\delta O_j} = W_{j\to K} O_K (1-O_K) \tag{8}$$

Finalmente, la derivada de P con respecto a O_j se expresa como:

$$\frac{\delta P}{\delta O_j} = \sum_K \frac{\delta P W_{j\to k} O_K (1-O_K)}{\delta O_K} \tag{9.1}$$

Se debe determinar la derivada parcial de P con respecto a la salida de la última capa O_Z:

$$\frac{\delta P}{\delta O_Z} = \frac{\delta P-(d_Z - O_Z)}{\delta O_Z} = 2(d_Z - O_Z) \tag{9.2}$$

Si se sustituyen estos cálculos en la ecuación 3 de la derivada parcial de P con respecto a $W_{i\to j}$ y se multiplica por un factor de rapidez r, se obtiene:

$$\frac{\delta P}{\delta W_{i\to j}} = r\frac{\delta P}{\delta O_j} O_i O_j (1-O_j) \qquad (10)$$

En primer lugar, de la ecuación 10 se concluye que la derivada parcial de la medida de calidad con respecto a un peso depende de la derivada parcial de la medida de calidad con respecto a la salida. En segundo lugar, de la ecuación 9.1 se concluye que la derivada parcial de la medida de calidad con respecto a una salida depende de la derivada parcial de la medida de calidad con respecto a las salidas de la capa siguiente. Por lo tanto, la derivada parcial de la medida de calidad con respecto a variables de una capa dada, se da en términos de los cálculos que ya se necesitaron dos capas arriba.

El anterior procedimiento, en el cual se emplean fórmulas, se le conoce comúnmente como *algoritmo de retropropagación*, la idea general es "hacer un cambio grande a un peso en particular, si el peso conduce a una reducción grande de los errores observados en los nodos de salida"[21]. Resumiendo, la forma de proceder consistiría en:

1. Tomar parámetro de rapidez r.
2. Inicializar los pesos
3. Hasta que la medida de calidad P sea satisfactoria, por cada muestra de entrada:
    a. Calcular la salida resultante
    b. Para los nodos de la capa de salida, calcular ecuación 9.1
    c. Para los demás nodos ocultos, calcular ecuación 9.2
    d. Calcular los cambios en los pesos con la ecuación 10
    e. Acumular los cambios de los pesos
4. Cambiar los pesos con los cambios acumulados para todas las muestras
5. Los cambios en los pesos son proporcionales a los errores de salida, y las salidas tenderán a 1 y 0, en consecuencia la medida de calidad normalmente se

[21] *Ibidem*, p. 486

considera satisfactoria cuando, todas las salidas cuyo valor deseado es "1" muestran valores > 0.9, pero cuando se desea "0" muestran valores < 0.1

Por otra parte, el entrenamiento de una red con muchas salidas puede hacerse:

- En etapas: cuando primero se entrena para una salida, luego se agrega otra y se entrena comenzando con los pesos ya entrenados para la primera, a continuación se entrena agregando una tercera salida, etc
- Simultáneamente: cuando la red se entrena para todas las salidas a la vez.

Para ilustrar el aprendizaje por red neuronal, supóngase tener una sencilla red neuronal que, a partir de dos entradas activadas entre los elementos A1,A2,A3, B1,B2,B3 (los tres primeros de la clase A y los tres últimos de la clase B), reconoce si éstos son de igual o distinta clase (figura 2.14).

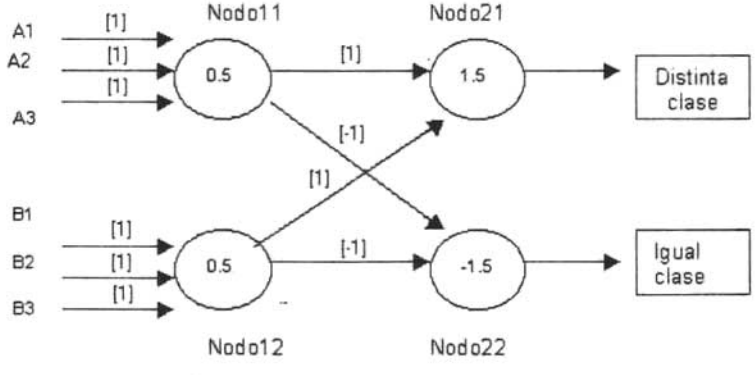


Figura 2.14, Red Neuronal de cuatro perceptrones

Por lo tanto, tenemos una red neuronal que consta de una capa de nodos ocultos (Nodo11, Nodo12) porque se sabe su entrada pero no su salida, y una capa de nodos de salida (Nodo21, Nodo22) que emiten las conclusiones. Se muestran los pesos asignados a las entradas dentro de corchetes cuadrados, y el valor de los umbrales dentro de cada nodo.

Si por ejemplo, sólo se activan las entradas A1 y A3, esto quiere decir que las entradas tendrían los valores siguientes: A1=1, A2=0, A3=1, B1=0, B2=0, B3=0. Ahora, la manera en que la Red Neuronal procede sería:

- El nodo11 procesa 1*A1+1*A2+1*A3 = 1*1+1*0+1*1 = 2, y como 2 >0.5 su salida S11 = 1
- El nodo12 procesa 1*B1+1*B2+1*B3 = 1*0+1*0+1*0 = 0, y como $0 \leq 0.5$ su salida S12 = 0
- El nodo21 procesa 1*S11+1*S12 = 1*1+1*0 = 1, y como $1 \leq 1.5$ su salida S21 = 0.
- El nodo22 procesa (-1)*S11+(-1)*S12 = (-1)*1+(-1)*0 = -1, y como $-1>(-1.5)$ su salida S22 = 1.

La interpretación de las salidas nos indica que las entradas A1 y A3 son de igual clase. Para probar con otras entrada se procede de forma semejante.

Por otra parte, las redes neuronales se vuelven erráticas y poco confiables si tienen muchos pesos, situación denominada *sobreentrenamiento.* Como solución a tal problema se emplea la siguiente heurística:

NúmeroDePesos < NúmeroDeMuestrasDeEntrenamiento

La cual limita el número de pesos disponibles en la red neuronal.

Desde sus inicios hubo muchos debates en la comunidad de la inteligencia artificial sobre el futuro de las redes neuronales. Algunos pensaban que estas redes sólo tendrían una función limitada en la inteligencia artificial; otros esperaban que eclipsarían plenamente a la estrategia tradicional basada en reglas. Pese a lo anterior, actualmente las redes neuronales ya se están utilizando en diversas aplicaciones, desde la visión artificial hasta sistemas expertos debido a que son muy útiles para reconocer patrones ocultos en

cantidades enormes de números, como en la investigación científica, el análisis de préstamos y el análisis de la bolsa de valores.

Actualmente algunas de las máquinas de módem más sofisticadas emplean redes neuronales para distinguir señales a partir de ruidos en las líneas telefónicas. Y corporaciones como Intel y otras compañías de hardware producen chips de redes neuronales que contienen miles de neuronas. A su vez, varias compañías de software han elaborado programas que simulan redes neuronales en computadores personales y otras máquinas que no operan en paralelo. Sin embargo, en la actualidad no hay hardware ni software de redes neuronales que se aproxime a la complejidad o la capacidad del cerebro humano. No obstante, la mayoría de los investigadores consideran que las redes neuronales son, en el mejor de los casos, los primeros pasos hacia las máquinas que emularán con mayor precisión el funcionamiento del cerebro humano.

CAPÍTULO 3

MÉTODOS DE MANIPULACIÓN Y REPRESENTACIÓN DEL CONOCIMIENTO

Lo que puede decirse puede decirse claramente;
donde no se puede hablar hay que callar.
Ludwig Wittgenstein.

3.1 REPRESENTACIÓN DEL CONOCIMIENTO

Representar el conocimiento en una computadora, consiste en encontrar una correspondencia entre el "cuerpo del conocimiento" y un "sistema simbólico" que lo denote o le haga referencia, y que además, permita solucionar problemas con base en dicha correspondencia. Es decir, se trata de formalizar y estructurar el "conocimiento" para que una computadora pueda trabajar con él, por tal motivo dicho proceso también recibe el nombre de formalismo.

Hasta la fecha no existe un formalismo ideal para poder representar el conocimiento. No obstante, se han logrado distinguir tres tipos de representación que son:

1) Representación procedimental: consiste en autómatas finitos y programas que expresan explícitamente las interrelaciones entre fragmentos de conocimiento pero que son dificilmente modificables.

2) Representación declarativa como son: lógica proposicional, cálculo de predicados, reglas de producción y redes semánticas. Que representan fragmentos de conocimiento independientes unos de otros siendo por ello fácilmente modificables.

3) La representación mixta que emplea objetos estructurados como: esquemas, marcos, grafismo, objetos, etc., que realizan representación procedimental y declarativa.

En esta sección del capítulo prestaremos atención sólo a los tipos de representación declarativa, ya que ellos tienen considerables ventajas como son: legibilidad, flexibilidad y facilidad de modificación, en comparación a los de representación procedimental, y además, sirven de base a la representación mixta.

3.1.1 Lógica proposicional o de enunciados.

La lógica de proposiciones es la más simple, y se llama así porque en ella las proposiciones o enunciados forman la única categoría semántica básica. El objetivo de esta lógica es construir un modelo formal del pensamiento deductivo, y aunque no rescata totalmente el pensamiento deductivo humano, tiene las características esenciales de modelos más sofisticados y es fácil de manejar.

El razonamiento deductivo se presenta en forma de argumentos, entendiéndose por «argumento»: "una lista de proposiciones relacionadas de tal manera que la última, llamada conclusión del argumento se sigue de las anteriores, llamadas premisas del argumento".[22]

Las proposiciones se dividen en simples (atómicas) y combinadas (moleculares). Lo esencial de una proposición es que expresa algo que puede ser verdadero o falso, es decir, facilita los elementos para demostrar los valores de verdad de cada afirmación que se haga. Así, "una proposición es toda afirmación de la que se puede decir sin ambigüedad y de manera excluyente que es cierta o falsa. Cuando es cierta se le atribuye el valor lógico 1 ó V (verdad) y si es falsa 0 ó F (falsa)."[23] Por ejemplo:

a) "México es la capital de China" es una proposición falsa
b) "La numeración es infinita" es una proposición verdadera
c) "Hoy es un bonito día" no es una proposición, sino un enunciado declarativo.

Las proposiciones (a) y (b) son simples ya que ninguna de ellas tiene conectivos lógicos. Cuando se combinan proposiciones entre sí mediante el empleo de conectivos lógicos obtenemos otra proposición la cual es compuesta. Los conectivos lógicos principales son:

[22] Julio Solis Daun y Yolanda Torres Falcón, *Lógica matemática*, UAM, México, 1995, p.1
[23] Juan Ferrando y Valentín Gregori, *Matemática discreta*, Reverte, México, 1994, p.1

Nombre	Conectivo lógico	Símbolo
Negación	No	~
Conjunción	Y	$\wedge$
Disyunción	O	V
Condicional	Si... entonces	$\rightarrow$
Bicondicional	Si y solo si	$\leftrightarrow$

La negación significa un cambio de valor de verdad. Si una proposición es verdadera, su negación es falsa y viceversa. Una conjunción es falsa si por lo menos uno de los conjuntos es falso. La disyunción es verdadera cuando al menos uno de sus disyuntos es verdadero, y es falso en el caso en que los dos disyuntos sean falsos. Un condicional es falso sólo en el caso de que el consecuente sea falso. Y por su parte, el bicondicional es verdadero si los dos son verdaderos o falsos.

La característica común de los conectivos lógicos es que son «veritativo-funcionales», es decir, mediante ellos y una proposición simple (en el caso de la negación), o a partir de dos o más proposiciones simples (en el caso de los restantes conectivos lógicos), se pueden formar proposiciones compuestas cuyo valor de verdad es exclusivamente una función del valor de verdad de los componentes. Para entender lo anterior consideremos las letras «p» y «q», que ya no son ellas mismas proposiciones, ni tampoco son nombres de oraciones particulares que expresan proposiciones; sino letras denominadas *metavariables* proposicionales que sirven para hacer aserciones sobre todas las proposiciones. Así, para «p» y «q» hay cuatro posibles valores de verdad:

p	q
V	V
V	F
F	V
F	F

p	q
1	1
1	0
0	1
0	0

Extendiendo la tabla anterior hacia su derecha con el fin de anotar el valor de verdad que los compuestos veritativos funcionales toman para cada una de las combinaciones posibles de valores de verdad de sus proposiciones componentes. Tenemos la siguiente tabla de verdad:

p	q	~p	p V q	p Λ q	p → q	p ↔ q
1	1	0	1	1	1	1
1	0	0	1	0	0	0
0	1	1	1	0	1	0
0	0	1	0	0	1	1

Es importante señalar que existen algunas notaciones alternativas que son de uso frecuente para algunos de los conectivos lógicos, estos son los que se presentan a continuación:

—p, ¬p,	para	~p
p&q, p·q	para	pΛq
p ⊃ q	para	p→q
p~q, p ≡ q	para	p↔q

Ahora consideremos las siguientes expresiones:

(p) París esta en Cuba

(q) Japón esta en el caribe

Al poder asignar un valor de verdad a las anteriores expresiones, tenemos entonces que se trata de dos proposiciones que podemos unir con cualquiera de los conectivos lógicos con el fin de obtener proposiciones compuestas, y además, de acuerdo a su estructura podemos representarlas mediante un simbolismo que nos permita trabajar sobre sus posibles valores de verdad de cada una. Haciendo esto tendríamos lo siguiente:

I) París esta en cuba y Japón esta en el caribe ≡ p Λ q

II) París esta en cuba o Japón esta en el caribe ≡ p V q

III) Si París esta en cuba, entonces Japón esta en el caribe ≡ p → q

IV) París esta en cuba si y sólo si Japón esta en el caribe ≡ p ↔ q

V) No es cierto que París esta en Cuba ≡ ~p

Cada una de las anteriores proposiciones compuestas expresa una formula bien formada (fbf) molecular cuyo valor se puede calcular a partir de las proposiciones que aparecen en ella (en este caso: "p" y "q"), y de las tablas de verdad de los conectivos lógicos. Por ejemplo, "p" y "q" son falsas por que París no esta en Cuba, ni mucho menos Japón esta en el Caribe, ahora, en I tenemos que p Λ q ≡ F Λ F que de acuerdo a la tabla de verdad de la conjunción nos da como resultado de la proposición molecular el valor F.

Para saber si un argumento dado es correcto o no, lo que se verifica es si la verdad de la conclusión se sigue de la verdad de las premisas, por tanto, debemos tener una manera precisa de saber cuándo una fórmula bien formada es verdadera. De esta manera, la lógica proposicional es un formalismo lógico que proporciona una forma de derivar nuevo conocimiento a partir de otros. Así, se puede concluir que una nueva sentencia es verdadera probando que se deriva de una sentencia ya conocida[24], sin embargo, tiene ciertos inconvenientes, por ejemplo, una generalización se expresa únicamente mediante una lista exhaustiva de sus elementos, por lo cual si se quiere expresar el hecho de que todos los hombres son mortales, se tendría que hacer mediante una lista exhaustiva de proposiciones de todos los hombres existentes y la afirmación de su mortalidad.

La lógica proposicional, se nombra como lógica de orden 0 porque no contiene variables, posee una extensión, la lógica de orden 1 o lógica de predicados, la cual tiene en cuenta variables y cuantificadores que permiten realizan generalizaciones de forma más sencilla, y de la que hablaremos enseguida.

[24] Se somete a una prueba, la cual consiste en una deducción para contestar preguntas y elaborar soluciones a problemas. Al anterior proceso también se le llama «inferencia lógica».

3.1.2 Lógica de predicados.

En la lógica de predicados, también conocida como cálculo de predicados, las proposiciones simples se descomponen en partes más simples, que forman así una segunda categoría semántica: la categoría de los nombres. Los nombres aparecen en las proposiciones unidos a predicados (de ahí el nombre de lógica de predicados), que expresan propiedades y relaciones, funcionando como verbos. Además, la lógica de predicados permite hacer generalizaciones mediante el empleo de dos cuantificadores, que son:

Nombre	Símbolo	Significado
Cuantificador universal	$\forall$	Todos
Cuantificador existencial	$\exists$	Existe o Algunos

Bajo la regla de que los individuos serán representados por letras minúsculas, las propiedades de los individuos y las relaciones entre ellos serán representados por letras mayúsculas, y los conceptos "todos" y "algunos o existe" serán representados respectivamente por el símbolo de sus cuantificadores, podemos expresar proposiciones como:

1. Todos los hombres son mortales
2. Hay un hombre llamado Sócrates
3. Todo individuo tiene un padre

De la manera siguiente:

1. $\forall x\ (H^1x \rightarrow M^1x)$
2. $\exists x\ (H^1x\ \Lambda\ N^2(x,s))$
3. $\forall z\ (\exists x\ (P^2zx))$

Donde la letra minúscula "x" y "z" representa a individuos arbitrarios, y la minúscula "s" representa una constante individual que es Sócrates. Las letras mayúsculas "H", "M", "P" y "N" representan predicados, tales que:

H^1 = ser hombre
M^1 = ser mortal
N^2 = x se llama s
P^2 = z es hijo de x

Los índices indican el tipo de predicado: el 1 corresponde a un predicado monádico, el 2 a uno diádico, y 3 o más a predicados poliádicos. No obstante, para el empleo en computadora, no se suelen abreviar las propiedades de los individuos y las relaciones entre ellos, y se descartan los índices, por eso la forma de representarse sería la siguiente:

1. $\forall$x hombre (x) es mortal (x)
2. $\exists$x hombre (x) y nombre (x, Sócrates)
3. $\forall$z $\exists$x Es_hijo (z,x)

El cálculo de predicados proporciona un medio natural de representar el conocimiento de forma declarativa, y nos permite demostrar cuando un argumento es correcto o válido, o incluso derivar nuevas proposiciones mediante la aplicación de un conjunto de reglas de inferencia.

Uno de los métodos más difundidos es la «Demostración por resolución". Para efectuarlo se establece una "base de conocimiento" que consistirá en un conjunto de fbf y de reglas semánticas que las relacionan en el dominio de la aplicación. La deducción de nuevos conocimientos se realiza a partir de los antiguos mediante procedimiento de prueba automática de teoremas. Por ejemplo, considerando las siguientes oraciones:

- A Frege[25] le gusta todo tipo de comida.
- Las manzanas son comida
- El pollo es comida
- Todo lo que alguien come y no lo mata es comida
- Russell come nueces y esta vivo
- Quine come todo lo que come Russell

El primer paso es escribir las oraciones con lógica de predicados obteniendo:

1. $\forall x$ comida(x) $\rightarrow$ gusta(Frege,x)
2. comida(manzanas)
3. comida(pollo)
4. $\forall x \forall y$ come(y,x) $\wedge$ ~muerto(y) $\rightarrow$ comida(x)
5. come (Russell, nueces) $\wedge$ ~muerto(Russell)
6. $\forall x$ come(Russell,x) $\rightarrow$ come(Quine,x)

Ahora el objetivo es obtener cláusulas (disposiciones o reglas básicas) ya que ellas permiten trabajar mejor en una computadora la deducción de conocimientos, por ello es conveniente eliminar las implicaciones mediante la afirmación del consecuente y la negación del antecedente, es decir, mediante la equivalencia "p $\rightarrow$ q $\equiv$ ~pVq" tenemos:

1. $\forall x$ ~comida(x) V gusta(Frege,x)
2. comida(manzanas)
3. comida(pollo)
4. $\forall x \forall y$ ~[come(y,x) $\wedge$~muerto(y)] V comida(x)
5. come (Russell, nueces) $\wedge$ ~muerto(Russell)
6. $\forall x$ ~come(Russell,x) V come(Quine,x)

[25] Gottlob Frege (1848-1925) es considerado el padre de la lógica de predicados.

Aplicando en 4 D´Morgan "~(pΛq)≡~pV~q, ~(pVq)≡~pΛ~q" y doble negación "~~p≡p"; y para facilitar su manejo separamos las cláusulas en 5 (esto es permitido porque tenemos conectivo lógico "Λ") nos queda:

1. ∀x ~comida(x) V gusta(Frege,x)
2. comida(manzanas)
3. comida(pollo)
4. ∀x∀y ~come(y,x) V muerto(Russell) V comida(x)
5. come (Russell, nueces)
6. ~muerto(Russell)
7. ∀x ~come(Russell,x) V come(Quine,x)

Ahora se mueven los cuantificadores a la izquierda y se eliminan para obtener la forma clausular pura, con lo cual tendríamos:

1. ~comida(x) V gusta(Frege,x)
2. comida(manzanas)
3. comida(pollo)
4. ~come(y,x) V muerto(Russell) V comida(x)
5. come (Russell, nueces)
6. ~muerto(Russell)
7. ~come(Russell,x) V come(Quine,x)

Las anteriores expresiones forma la "base de conocimiento". Ahora para probar una nueva proposición como: ¿A Frege la gustan las nueces?, se procede a demostrar que su negación no es verdadera porque genera una contradicción, es decir, para demostrar cierta proposición P se emplea «Reducción al absurdo» procediendo de la siguiente manera:

I. Se supone lo contrario, es decir, No P

II. De esta suposición se llega a una contradicción (expresión de la forma A y ~A). O sea, No P me lleva a una situación donde no se cumple con el principio de no contradicción; lo cual es imposible (absurdo).

III. El absurdo surge de suponer No P. Por lo tanto, eso no puede ser, y entonces tenemos que P.

Siguiendo con nuestro ejemplo tendríamos el procedimiento siguiente:

Sea P = A Frege le gustan las nueces, que se representa en lógica de predicados como:

gusta (Frege, nueces) cuya negación es: ~P = ~gusta(Frege, nueces)

Primer paso:

~gusta(Frege, nueces)
~comida(nueces) V gusta(Frege, nueces) por cláusula 1

~comida(nueces)

Segundo paso:

~comida(nueces)
~come(Frege, nueces) V muerto(Russell) V comida(nueces) por cláusula 4

~come(Frege, nueces) V muerto(Russell)

Tercer paso:

~come(Frege, nueces) V muerto (Russell)
~muerto(Russell) por cláusula 6

~come(Frege, nueces) ≡ ~come(Frege/Russell, nueces) por emparejamiento

Cuarto paso:

~come(Frege/Russell, nueces)
come(Frege/Russell, nueces) por cláusula 5

NIL o Nulo

Por lo tanto, el teorema negado "~gusta(Frege, nueces)" es falso porque genera una contradicción. Así se demuestra que a Frege le gustan las nueces porque su negación no lo es.

Muchos problemas de naturaleza deductiva se expresan muy bien en la lógica de predicados, motivo por el cual el lenguaje PROLOG haga uso de este tipo de representación. Sin embargo, la mayor parte de los problemas de la vida real que tienen que resolver los seres humanos son de naturaleza inductiva, por tal razón se han desarrollado otros tipos de formalismo con el fin de tener en cuenta aspectos inherentes al razonamiento humano como los que veremos a continuación.

3.1.3 Reglas de producción y sistemas de reacción.

Consiste de descripciones de acciones dependientes de ciertas condiciones. Su principio básico es que cada descripción denominada regla, es un trozo independiente del conocimiento (llamado gránulo o módulo; del inglés chunk) que contiene todas las condiciones para su aplicación, es decir, se trata del componente más pequeño del que consta el sistema en su totalidad.

Las reglas de producción operan sobre la base de datos (o base de conocimientos). Es decir, la aplicación de una regla produce cambios en la base de datos. Cuentan con un mecanismo de inferencia, el cual es un sistema de control que escoge qué reglas deben ser aplicables y suspende el proceso cuando la base de datos global satisface una condición de terminación.

La arquitectura de un sistema de producción se representa de la manera siguiente:

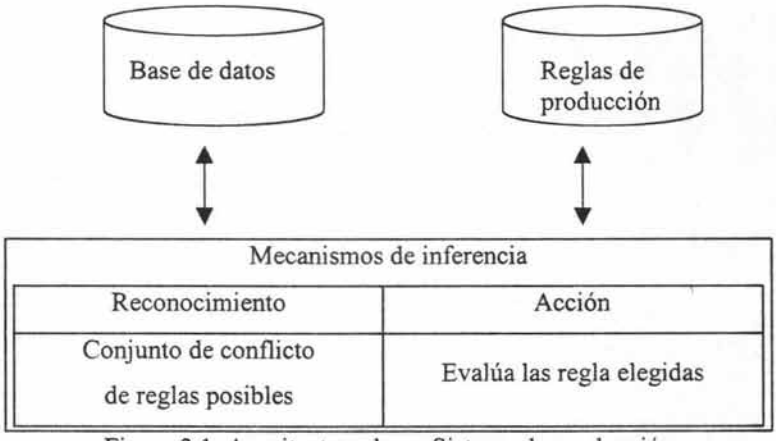


Figura 3.1, Arquitectura de un Sistema de producción

La estructura básica de las reglas es "Si... entonces" (if... then) en donde la parte posterior a la palabra "si" se le conoce como antecedente o premisa, y a la parte posterior a "entonces" se le llama consecuente o conclusión. De esta forma se consiguen expresiones como:

SI el profesor llega ENTONCES imparte clases
Si la lámpara tiene pilas ENTONCES se enciende

Con la estructura anterior se pueden formar términos completos, es decir, sentencias compuestas por uno o más conectivos "y" u "o" para obtener expresiones como las siguientes:

SI es un animal y es racional ENTONCES es un hombre. [26]
Si es mamífero y maúlla o tiene pelo y ladra ENTONCES es un animal doméstico.

Observemos que con éste tipo de enunciados podemos tener una mejor base de conocimientos, motivo por el cual este método de representación es muy utilizado en

[26] De acuerdo con la definición aristotélica de hombre: hombres es animal racional.

sistemas expertos (por ejemplo: MYCIN, DENDRAL[27], etc.) ya que brinda determinadas ventajas derivadas de su estructura como son: sencillez, facilidad de agrupación y manejabilidad de reglas, es decir, que las reglas pueden fácilmente modificarse, ya sea mediante el cambio, la ampliación o supresión, sin producir alteraciones considerables en la base del conocimiento.

En casos particulares de los sistemas basados en regla, el «consecuente» o «conclusión» en vez de indicar una afirmación, lo que hace es denotar acciones a realizar. A esta forma de sistema se le conoce como «sistemas de reacción» debido a que obedecen al «antecedente» ejecutando la acción, y su aplicación esta muy difundida en procesos industriales.

Las desventajas que se le atribuyen a este tipo de representación del conocimiento son tres. La primera es la pérdida de coherencia lógica cuando se tiene un gran número de reglas; en segundo tenemos la utilización de «metareglas» (una regla acerca de otra regla) que guían bajo qué condiciones se tienen que considerar unas reglas en lugar de otras debido a la dificultad que hay para fijar relaciones; y por último, la velocidad en el proceso de inferencia que se ve disminuido cuando existe un gran número de reglas.

3.1.4 Redes semánticas

Son un método de representación del conocimiento sobre las relaciones de los conceptos (objetos o hechos), mediante grafos que constan de nodos y arcos. Los nodos de una red semántica representan los conceptos y los arcos describen las relaciones entre los conceptos. La primera red semántica se presentó en 1968, y se han utilizado para construir algunos S. E., por ejemplo, PROSPECTOR[28] que trabaja en el dominio de la geología mineral. La figura 3.2 muestra un ejemplo de una pequeña red semántica.

---

[27] MYCIN es un S. E. para el diagnostico médico y la selección de terapias. DENDRAL es un S. E. que propone estructuras plausibles para compuestos bastante complejos en el área de química.

[28] Nilsson Nils, *Principios de inteligencia artificial*, Díaz de Santos, Madrid, 1987, pp. 374, 377

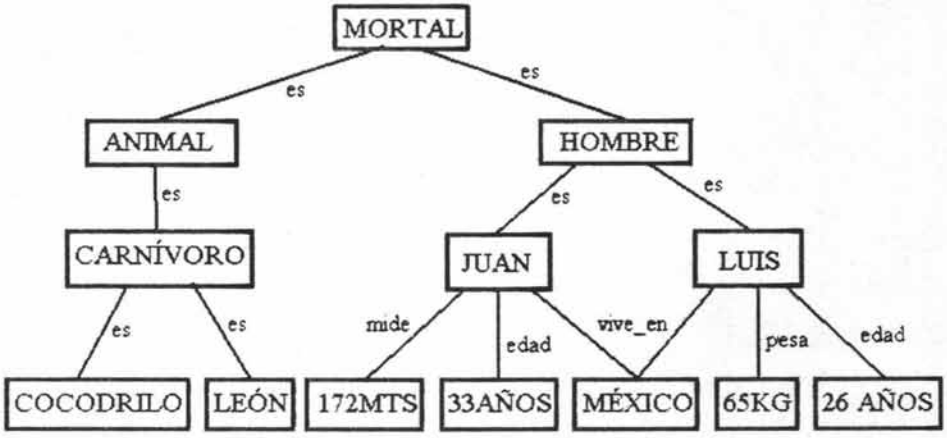


FIGURA 3.2, Red Semántica

Podemos constatar que una red semántica "ofrece una buena visión general sobre las relaciones y dependencias de un área del conocimiento (dominios) y es muy apropiada para la estructuración del conocimiento"[29]. Una desventaja consiste en que no da información sobre el procesamiento de la red de tal forma que "las reglas de inferencia deben estar expresadas de forma explicita"[30] y los enunciados de las relaciones mencionadas en los arcos deben ser formulados fuera de la red.

Como una red semántica se puede expresar con cualquier frase, por consiguiente, es fácil responder a preguntas elaboradas sobre el tema que en ellas se representa. En nuestro ejemplo, es sencillo responder a preguntas como: ¿León es un animal carnívoro? ¿Felipe vive en México? ¿Antonio tiene 33 años?, etc. Sin embargo, como las reglas que posibilitan deducciones adicionales se formulan fuera de la red, siguiendo con nuestro ejemplo se tiene:

- $es(x,y) \wedge es(y,z) \rightarrow es(x,z)$

[29] Nebendahl Dieter, *Sistemas expertos*, Siemens-Marcombo, Barcelona, 1988, p. 59
[30] *Ibidem*, p. 59

La cual es una regla que dice que cuando "x" es "y", y además "y" es "z", entonces "x" también es "z". O sea: cuando León es un Carnívoro, y Carnívoro es un animal, entonces León es un Animal.

La red semántica de la figura 3.2 representa el conocimiento de una forma gráfica, no obstante, no se puede representar el conocimiento de está forma dentro del computador, por ello las redes semánticas tienen un caso peculiar llamado *ternas* que consta de: objeto-atributo-valor. En la *terna* se eliminan los enlaces con el fin de ser sencillos para programar en lenguajes utilizados en I. A. Bajo el anterior concepto una parte de la red de la figura 3.2 se representaría de la forma siguiente:

Segmento de red semántica:

LEÓN es CARNÍVORO es un ANIMAL es MORTAL

Se expresa por las ternas:

LEÓN-carnívoro-ANIMAL
LEÓN-animal-MORTAL

La mayor desventaja de este tipo de representación es que cuando se tiene una gran cantidad de conocimiento, es decir, si hay presencia de dominios múltiples, se ocupan enormes sistemas gráficos para representarlos, lo cual trae consigo dificultades a la hora de establecer las relaciones que en las redes se pretenden definir.

3.1.5 Frames (Marcos), Plantillas u objetos estructurados

Los Frames fueron introducidos por Minsky en 1957, como un tipo de representación cuya tentativa es representar el conocimiento sin recurrir a algún tipo de inferencias. Sin

embargo, se puede considerar como una extensión de las redes semánticas con la ventaja de que incorpora conocimientos procedimentales e introduce un rigor conceptual.

Un Frame o Plantilla consiste en la división de objetos o situaciones, en sus componentes integrantes, motivo por el cual también se le denomina «objetos estructurados». Estos componentes son atributos o propiedades del objeto o hecho que se introducen en ranuras (slots). Cada ranura puede a su vez estar subdividida en otras llamadas *facets* que le permiten tener una estructura más precisa.

Cuando en el Frame-Plantilla se describen tipos o cosas individuales, éstas reciben el nombre de ejemplares u objetos, cuando en él se describe un conjunto de cosas completas, se les conoce como clases. Asimismo, el Frame-Plantilla antes de ser utilizado no es más que un armazón preestructurado de datos vacío, que va llenándose en sus ranuras con contenido a lo largo del procesamiento. Para ilustrar supongamos que tenemos la siguiente clase y:

SILLA

Clase	Valor	Mueble
Numero de patas	Defecto	4
Color	Posibilidad	Blanco, azul, marrón
Material	Posibilidad	Madera, metal, plástico
Edad	Restricción	0< y <100

Y el siguiente objeto:

SILLA DE MI ABUELA

Clase	Silla
Numero de patas	3
Color	Blanco
Material	Madera
Edad	50

En donde:

- SILLA es una clase, y SILLA DE MI ABUELA es un objeto.
- Valor designa valor de un atributo.

- Defecto indica el valor en ausencia del atributo si no se especifica otra cosa.
- Posibilidad son casos particulares de restricción que se indican mediante una lista exhaustiva de los valores posibles.
- Restricción es una lista de predicados que devolverán CIERTO cuando se aplican al valor al que se trata de afectar el atributo apropiado.

Para asignar ejemplares a sus clases correspondientes o de las que son miembros, se emplea la forma sintetizada de «*es un miembro de la clase*» que consiste en el descriptor "ES UN". Por su parte, el descriptor "A.K.O." por ser siglas de la frase en inglés *a kind of* es la forma sintetizada de «*es un tipo de*», que se emplea para unir clases entre sí. De tal forma que haciendo uso de los descriptores en nuestro ejemplo de SILLA y SILLA DE MI ABUELA se obtiene un esquema como el mostrado en la figura 3.3.

Es importante observar que en la figura existen jerarquías de clases. En nuestro ejemplo, la jerarquía que hay es muy simple, en ella el objeto SILLA DE LA ABUELA pertenece a la clase SILLA; la SILLA a su vez es una subclase de MUEBLES o MUEBLES es una superclase inmediata de SILLA; y Muebles es una subclase de UNIVERSO. La jerarquización es importante, porque de ella se deriva la herencia, esto es, las clases superiores heredan sus propiedades a las clases inferiores u objetos.

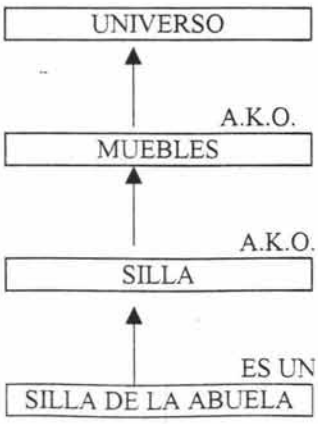


Figura 3.3, Herencia de clases

Para crear plantillas-frames ya sea de clase o ejemplares, y para poder manejarlas más eficientemente, se emplean procedimientos denominados demonios (demons), los más usuales son:

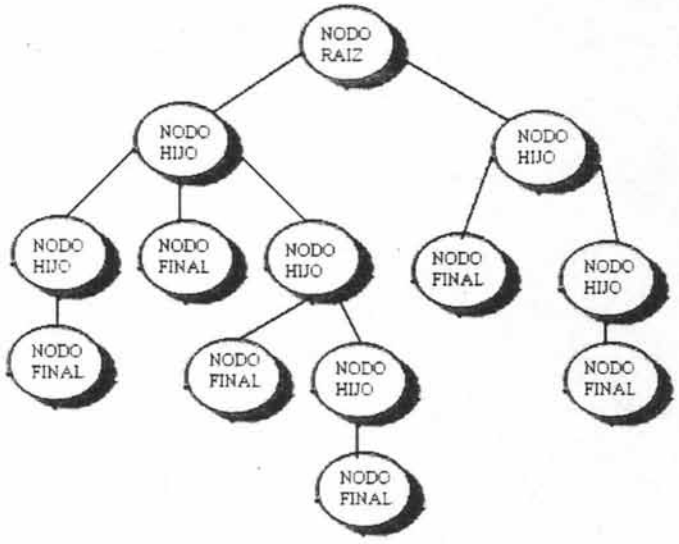
- si-fuese-necesario (if-needed)
- si-es-agregado (if-added)
- si-es-borrado (if-removed)
- si-se lee (if-read)

El primero de ellos se encarga de conseguir el valor para un *slot* vacío, el segundo tiene la función de almacenamiento de un valor, el tercero sirve para borrar el contenido de los *slots*, y el último sólo permite leer.

3.2 MANIPULACIÓN DEL CONOCIMIENTO O ESTRUCTURAS DE CONTROL

El conocimiento se manipula con el fin de resolver problemas. Los métodos (también llamados: mecanismos, estrategias o estructuras de control) utilizados en la manipulación del conocimiento consisten en búsquedas. En donde cada búsqueda trata de hallar la solución óptima, al recorrer todo un universo de posibles soluciones conocido como «espacio de búsqueda». La rapidez con que se encuentre la solución a un problema depende del mecanismo utilizado.

Para visualizar la forma como trabajan estos métodos, se han desarrollado esquemas gráficos representativos de los espacios de búsqueda llamados árboles o grafos. La figura 3.4 muestra de forma general un árbol-grafo donde pueden apreciarse los elementos más característicos como son: un conjunto de círculos conectados por líneas; a cada círculo se denomina Nodo, y a cada línea Rama. Un nodo raíz, algunos nodos hijos y varios nodos terminales. Por regla general, cada nodo debe estar conectado por lo menos a otro mediante una rama.



Figurar 3.4, Árbol o grafo de búsqueda

El nodo raíz se toma como «Estado inicial», es decir, es el punto de partida de una búsqueda; los demás nodos corresponden cada uno a un estado o situación posible, y tomados en conjunto se denominan «espacio de búsqueda». Los nodos del espacio de búsqueda se suelen acomodar de forma descendente, cuando se llega a un nodo donde ya no se puede avanzar a otro (o sea, que no tiene descendientes), entonces, se ha llegado a un «nodo final». No hay que confundir al «nodo final» con el «nodo objetivo», este último es la solución de un determinado problema.

Se puede construir el grafo completo empleando las reglas que definen los movimientos permitidos en el espacio de solución para demostrar cómo es que funciona, sin embargo en la práctica casi nunca sucede así. En vez de construir el grafo explícitamente, y después realizar la búsqueda, la mayoría de veces se representa el grafo implícito en las reglas y se generan en forma explícita sólo aquellas partes que se decide explorar. Para realizar lo anterior, se suele emplear la descripción formal del problema, siguiendo los subsecuentes pasos:

1° Definir espacios de estados[31] que contengan todas las configuraciones de objetos relevantes.

2° Identificar uno o más estados que describan las situaciones en las que comience el proceso de resolución del problema (estado inicial).

3° Especificar uno o más estados que pueden ser soluciones aceptadas al problema (estados objetivos).

4° Especificar un conjunto de reglas que describan las acciones disponibles (operadores) que permitan pasar de un estado a otro.

La representación como «espacio de estados» permite definir formalmente el problema, mediante la necesidad de convertir una situación dada en una situación deseada por medio de un conjunto de operaciones permitidas; al mismo tiempo que define el proceso de resolución de un problema como una combinación de técnicas de búsqueda. Por ejemplo, el llamado "problema de las jarras de agua" consiste en que se tienen dos jarras de agua, una de 4 litros y otra de 3 litros, sin escala de medición.

El objetivo es tener 2 litros de agua en la jarra de 4 litros únicamente mediante las siguientes operaciones: llenar las jarras, tirar agua de las jarras, pasar agua de una jarra a otra. La descripción formal del problema es el siguiente:

(X,Y)	Se define como el espacio de estados donde: X representa los litros en la jarra de 4L con $0 \leq X \leq 4$ Y representa los litros de la jarra de 3L con $0 \leq Y \leq 3$
(0,0)	Se define como el estado inicial
(2,0)	Se define como el estado objetivo[32]

[31] Un estado es la representación de un problema en un instante dado. Para definir el espacio de estados no es necesario hacer una enumeración exhaustiva de todos los estado válidos, sino que es posible definirlo de manera más general.

Las reglas que se pueden aplicar, junto con sus respectivas condiciones son:

1. Llenar la jarra de 4L: Si $((X,Y) \wedge X<4) \rightarrow (4,Y)$
2. Llenar la jarra de 3L: Si $((X,Y) \wedge Y<3) \rightarrow (X,3)$
3. Vaciar la jarra de 4L: Si $((X,Y) \wedge X>0) \rightarrow (0, Y)$
4. Vaciar la jarra de 3L: Si $((X,Y) \wedge Y>0) \rightarrow (X, 0)$
5. Pasar agua de la jarra de 4L a la jarra de 3L hasta llenarla:
   Si $[(X,Y) \wedge X>0 \wedge (X+Y)\geq 3] \rightarrow (X-(3-Y),3)$; o sea, podemos tener (1,3)
6. Pasar agua de la jarra de 3L a la jarra de 4L hasta llenarla:
   Si $[(X,Y) \wedge Y>0 \wedge (X+Y)\geq 4] \rightarrow (4, Y-(4-X))$; o sea, podemos tener (4,2)
7. Pasar toda el agua de la jarra de 4L a la jarra de 3L:
   Si $[(X,Y) \wedge X>0 \wedge (X+Y)<3] \rightarrow (0,X+Y)$;
8. Pasar toda el agua de la jarra de 3L a la jarra de 4L:
   Si $[(X,Y) \wedge Y>0 \wedge (X+Y)<4] \rightarrow (X+Y,0)$

El programa deberia encontrar un pasaje de estados mediante la aplicación de las reglas para ir del estado (0,0) al estado (2,0)n Puede existir más de un pasaje de estados hacia la solución, por ejemplo:

(0,0) => (0,3) => (3,0) => (3,3) => (4,2) => (0,2) => (2,0)

es una posible solución en la cual, a partir del estado inicial, se aplicaron las reglas 2, 8, 2, 6, 3 y 8, hasta conseguir el estado objetivo. Es decir:

ESTADOS:	(0,0)	=>	(0,3)	=>	(3,0)	=>	(3,3)	=>	(4,2)	=>	(0,2) => (2,0)
REGLAS:		R2		R8		R2		R6		R3	R8

Otra ruta de estados hacia la solución es la siguiente:

(0,0) => (4,0) => (1,3) => (1,0) => (0,1) => (4,1) => (2,3) => (2,0)

[32] El estado final podria ser (2,N) en caso de que no importen los litros de la segunda jarra.

En la cual se aplicaron las reglas 1, 5, 4, 7, 1, 5 y 4

Es importante señalar que las condiciones que se establecen en las reglas a veces no son altamente necesarias, pero restringen la aplicación de la regla a estados más adecuados. Esto incrementa la eficiencia del programa que utiliza las reglas. En el "problema de las jarras de agua", la regla uno que consiste en:

Llenar la jarra de 4L: Si $((X,Y) \wedge X<4) \rightarrow (4,Y)$;

que contiene la condición (X<4), especificando que la jarra no se encuentra llena. Si ésta condición no se incluye, se puede aplicar la regla aún cuando la jarra está llena. Dado que en este caso el estado del problema no cambia, la aplicación de la regla se considera inútil. Asimismo, las reglas no sólo deben describir el problema sino también algún tipo de conocimiento sobre su solución. Si en el ejemplo anterior se considera la siguiente nueva regla:

Vaciar 'un poco' la jarra de 4l: Si $((X,Y) \wedge X>0) \rightarrow (X-Q,Y)$

Es decir, tirar agua sin cuantificar, intuitivamente se concluye que la aplicación de esta regla nunca nos acercará a la solución del problema.

Por otra parte, como son diversos los tipos de búsqueda aplicables a un problema, a fin de elegir el más apropiado se deben analizar las características de cada problema en particular de acuerdo con los siguientes lineamientos:

1. La "recuperación de pasos" en el problema, que se suele clasificar en uno de los siguientes:

   a) Los ignorables: en el cual pueden ignorarse algunos de los pasos dados, y que por lo general son los problemas referentes a la demostración de teoremas.
   b) Los recuperables: en el cual se puede deshacer pasos, es decir, retroceder a estados que ya antes se habían obtenido, por ejemplo, el juego de 8 puzzle utiliza la recuperación de pasos para su correcto funcionamiento en un computador.

c) Los no recuperables: que es donde no se pueden deshacer pasos, es decir, cada estado es único. Por ejemplo, el juego de ajedrez, ya que cada tirada significa un estado único a través de una partida.

2. La "predicción" de estados en el problema, que suele ser de dos tipos:

   a) De consecuencia cierta: donde el resultado de una acción se puede predecir perfectamente, por ejemplo, el 8 puzzle, el juego del gato, etc.
   b) De consecuencia incierta: donde la búsqueda se genera por una secuencia de operadores que tienen una buena probabilidad de conducir a una solución. Por ejemplo, el dominó, la baraja, etc.

3. Por los tipos de solución de problemas, que suelen ser dos:

   a) Absoluta: comprobar si algo es cierto dentro de un entorno nos lleva a una solución de este tipo, es decir, se tiene seguridad total sobre la mejor solución del problema, y lo único que se tiene que hacer es encontrarla.
   b) Relativa: donde no puede tenerse con seguridad total la solución a un problema, pero sin embargo se obtiene una respuesta satisfactoria.

4. Por el papel del conocimiento en el problema, que puede servir para:

   a) Acotar (reducir) la búsqueda ó,
   b) Reconocer una solución.

5. La interacción más conveniente en el problema.

   a) Los solitarios: donde únicamente se da una descripción del problema o Estados iniciales, y el sistema da una respuesta sin ningún tipo de comunicación, ni necesidad de explicaciones del proceso de razonamiento. Por ejemplo, el juego de ajedrez, el 8 puzzle, etc., en donde un usuario se enfrenta contra un computador.
   b) Los conversacionales: en donde hay una comunicación intermedia entre sistema y usuario a fin de proporcionar una ayuda al sistema. Por ejemplos los S. E.

Como "la mayoría de los problemas de I. A. pueden plantearse como problemas de recorrido en espacios de estados (espacio de búsqueda), donde cada estado corresponde a una situación posible".[33] Tenemos entonces que cada problema es un árbol "virtual" de todas las posibilidades, de buen o mal éxito. Lo importante reside en encontrar estrategias de búsqueda eficientes. Por ejemplo, un programa para el juego del gato (o tres en línea) implica un problema con las siguientes características:

- De pasos no recuperables, pues tras cada tirada ya no es posible retroceder.
- Con predicción de consecuencia cierta, porque se sabe cuales son las tiradas disponibles conforme avanza el juego.
- Con solución absoluta porque se sabe cuáles son en todos los casos las jugadas más convenientes, las que lleven al triunfo o al empate.
- Donde el papel del conocimiento es reconocer soluciones.
- La interacción no necesariamente es solitaria, ya que también se puede especificar el programa para que el usuario se enfrente ante la computadora.

En donde una de las soluciones más convenientes reside en una búsqueda Min-Max, cuyo árbol correspondientes es:

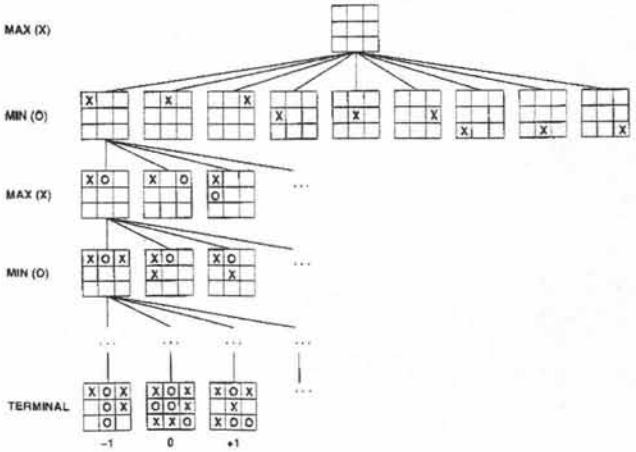


Fig. 3.5, Árbol de juego del gato (o juego de tres en línea)

[33] *Ibidem*, p. 97.

Los problemas más difíciles en la Inteligencia artificial son los no recuperables de consecuencia incierta y solución relativa, que además emplean los dos tipos de papel del conocimiento, por ejemplo, el control de un brazo robot que tiene que clasificar objetos con cierta irregularidad de acuerdo a su forma y color, o realizar soldadura por puntos en un proceso industrial (figura 3.6).

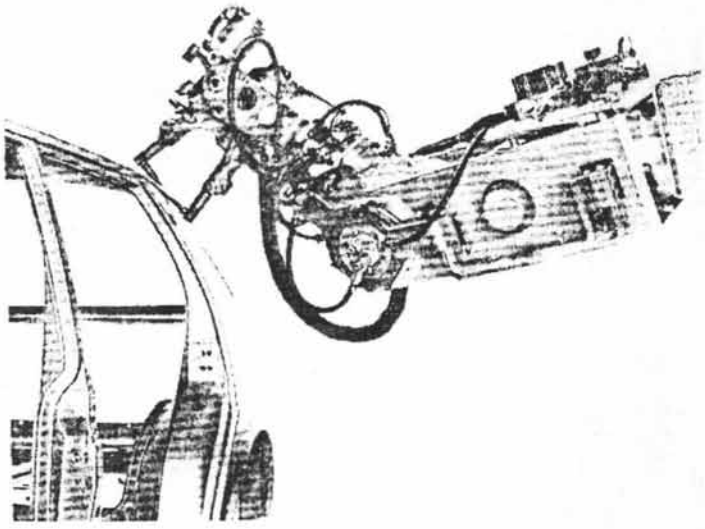


Figura 3.6, Brazo robot que realiza soldadura de puntos

Mediante la combinación de reglas y una «estrategia de control» es como se ejecuta la búsqueda a través del espacio de estados, cuyo fin es encontrar un camino desde el estado inicial hasta el estado final. Por lo que "una «estrategia de control» especifica el orden en el que se deben aplicar las reglas, así como también la forma de resolver conflictos cuando es posible aplicar más de una regla"[34]. El primer requisito que debe cumplir una buena estrategia de control es que cause algún cambio. El segundo es que la estrategia sea sistemática (que no sea al azar). Este último requisito se corresponde con una necesidad de cambio global (en el curso de varios pasos) así como de varios cambios (en el curso de un paso sencillo). Por el contrario, las estrategias de control que no causan cambios de estado nunca alcanzan la solución, y las que no son sistemáticas pueden utilizar secuencias de

[34] *Ibidem* , p. 125.

operaciones no apropiadas varias veces hasta alcanzar la solución, lo que implicaría a veces un tiempo considerable en la solución de un problema.

Los mecanismos o estrategias de control se suelen clasificar usualmente como se muestra a continuación en el esquema 3.7.

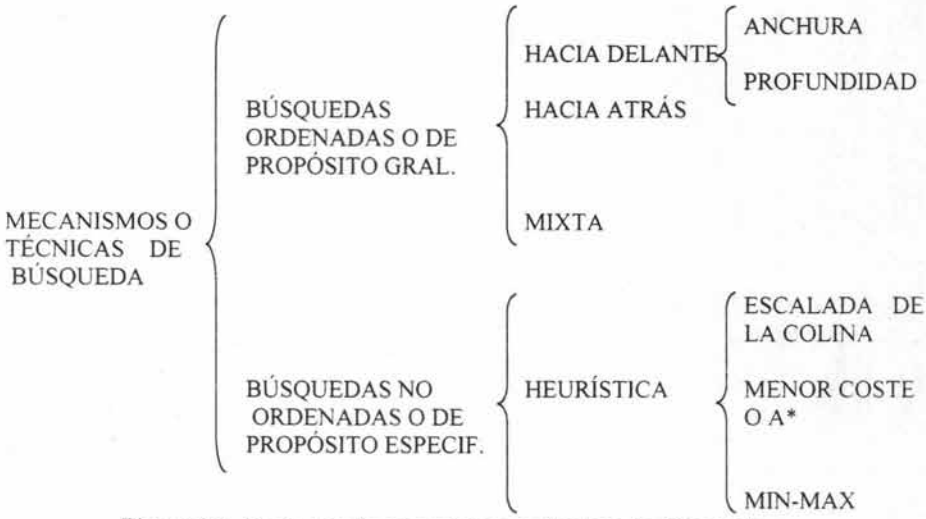


Figura 3.7, Esquema de mecanismos o técnicas de búsqueda

Enseguida se presenta en qué consiste cada tipo de búsqueda. La mayoría se basan en considerar un árbol de estados cuya raíz es el estado inicial, y en cada nivel se hallan los estados sucesores correspondientes.

3.2.1 Búsquedas ordenadas.

Se denominan así todas aquellas búsquedas donde se conoce la manera en que se va a recorrer el espacio de estados (o espacio de búsqueda). Como característica general se sabe que los nodos se encadenan a través de reglas que establecen que el nodo consecuente de uno sea el antecedente de otro, y así sucesivamente, hasta encontrar la solución deseada o en su defecto se termine de recorrer el espacio de búsqueda. Son

mecanismos bastante eficaces cuando el espacio de búsqueda no es muy amplio, ya que garantiza encontrar la solución, no obstante son muy lentas cuando el espacio de búsqueda es inmenso. Los tipos de búsqueda ordenada más comunes son los que a continuación se presentan.

3.2.1.1 Encadenamiento hacia delante

Se llama de esta forma cuando se tiene una búsqueda cuyo punto de partida es el «estado inicial», y que concluye al llegar al «estado objetivo». También se le denomina como "razonamiento guiado por los datos" debido a que a partir del conjunto de datos iniciales se van enlazando los conocimientos hasta llegar a la solución (estado objetivo). Su funcionamiento se basa en la llamada «implicación lógica" que señala lo siguiente:

Si X entonces Y

Que se suele interpretar como: Si 'X' es cierto, entonces 'Y' también lo es; o si sucede 'X', entonces también sucede 'Y'; o 'Y' sucede si 'X' sucedió. Para ilustrar supongamos las siguientes reglas:

Regla 1: Q si R y S	$(R \wedge S) \rightarrow Q$
Regla 2: R si X o T	$(X \vee T) \rightarrow R$
Regla 3: S si Y o Z	$(Y \vee Z) \rightarrow S$

Como datos de entrada tenemos: X, Y; y como objetivo supongamos que queremos a: Q. Ahora considerando las entradas y aplicando la regla 2 se obtiene:

X, Y, R

Aplicando la regla 3 se obtiene S, porque si Y o Z entonces S, con lo cual se tiene:

X, Y, R y S

Aplicando la regla 1 se obtiene Q porque si R y S entonces Q, que es nuestro estado objetivo.

3.2.1.2 Primero en anchura o extensión (Breadth first search)

Esta búsqueda esta orientada a recorrer el árbol o grafo correspondiente por niveles. Donde "un nivel es la posición descendiente donde se encuentra cada nodo, es decir, el nodo raíz se encuentra en el nivel 0, sus descendientes directos en el nivel 1 y los descendientes directos de estos, en el siguiente nivel y así sucesivamente".[35] Una búsqueda en anchura comienza en el nodo raíz y continua hacia abajo, examinando todos los nodos de un nivel antes de pasar al sucesor, y sólo se detiene cuando se encuentra ante un estado que se haya definido como estado objetivo. Como se ilustra en la siguiente figura.

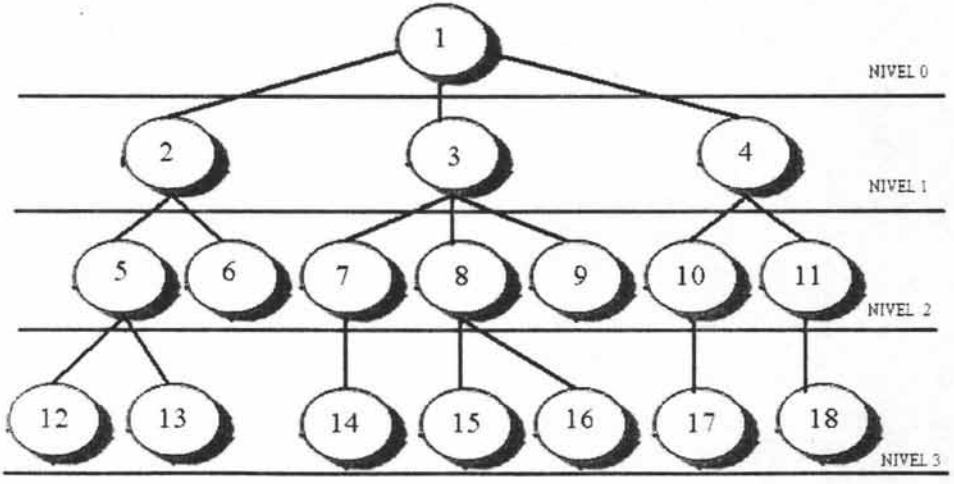


Figura 3.8, Búsqueda en anchura

Para conseguir una búsqueda de este tipo, a nivel de programa se suele utilizar una estructura tipo cola (FIFO) en la que se van introduciendo los nodos que son generados. A continuación se detalla un pseudo-código[36] de una búsqueda en anchura:

[35] Stuar Russell, *Inteligencia artificial: un enfoque moderno*, PPH, Argentina, 1998, p. 317
[36] Trazado escrito de la estructura y curso de un programa sobre un tipo de aplicación generalizada.

```
Lista_nodos = [estado_inicial];
Mientras Not Vacia(lista_nodos)

        estado_actual = lista_nodos.primero;
        Si EstadoFinal(estado_actual) entonces
                Terminar;
        Sino
                lista_reglas = ReglasAplicables (estado_actual);
                Mientras NOT Vacia(lista_reglas)
                        estado_sucesor = AplicarRegla (lista_reglas);
                        lista_nodos = lista_nodos + [estado_sucesor];
                Fin Mientras;
        Fin Sino;
Fin Mientras;
```

El anterior pseudo-código debería modificarse para detectar el caso en que se vuelva a alcanzar un estado que ya ha sido visitado con anterioridad. En este caso, debería realizarse una "poda" de la rama del árbol, ya que en caso contrario se volvería a generar un subárbol ya generado. Otra forma para esclarecer cómo es que funciona dicha búsqueda es mediante su algoritmo[37] correspondiente, el cual es el siguiente:

1. Crear una lista de nodos llamada ABIERTA asignarle el nodo raíz , que representa el estado inicial del problema planteado.
2. Hasta que ABIERTA este vacía o se encuentre un Edo_objetivo, realizar las siguientes acciones:
   2.1 Extraer el primer nodo de ABIERTA y llamarlo *m*.
   2.2 Expandir *m* (generar todos sus sucesores). Para cada operador aplicable y cada forma de aplicación.
      (1) Aplicar el operador a *m*, obtener un nuevo estado y crear un puntero que permita saber que su predecesor es $m=m+1$.
      (2) Si el nuevo estado generado es Edo_Objetivo, salir del proceso iterativo iniciado en 2.2 y devolver dicho estado.

[37] Por «algoritmo» se entiende una lista de instrucciones donde se especifica una sucesión de operaciones necesarias para resolver cualquier problema de un tipo dado.

(3) Incluir el nuevo estado al final de ABIERTA (una vez empleado este proceso para todos los sucesores de m –cuando no se haya encontrado antes un Edo_Objetivo- se continua el proceso iterativo en el paso 2).

Hay que aclarar que si el algoritmo termina con una meta, el camino de la solución puede obtenerse recorriendo los punteros creados desde el nodo meta al nodo raíz. En caso contrario, el proceso terminará sin haber encontrado la solución.

3.2.1.3 Primero en profundidad o paridad (Depth first search)

La *búsqueda en profundidad* se centra en expandir un único camino desde la raiz, en el caso de llegar a un callejón sin salida se retrocede hasta el nodo mas cercano desde donde se pueda tomar una rama alternativa para poder seguir avanzando. Es decir, se comienza en el nodo raíz y van recorriendo los distintos niveles del árbol hasta alcanzar el nodo más profundo de la rama más a la izquierda. Si el nodo final de dicha rama no es un nodo objetivo, se realiza un retroceso (backtracking) hasta un nivel en que se pueda continuar la búsqueda. La siguiente figura ilustra una búsqueda en profundidad.

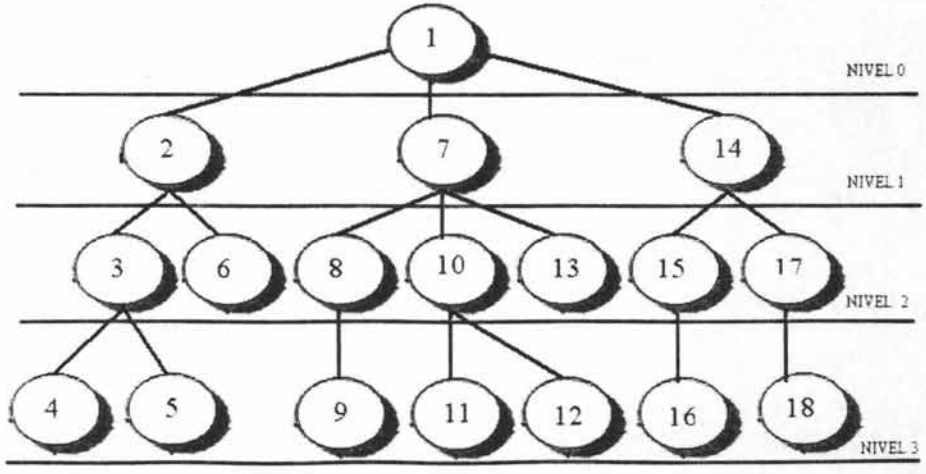


Figura 3.9, Búsqueda en profundidad

Para llevar a cabo este tipo de búsqueda debe utilizarse una estructura tipo pila (LIFO) que vaya almacenando los nodos generados, suele establecerse por otra parte ,el llamado *limite de exploración*, que marca la máxima longitud que puede alcanzar cualquier camino desde la raiz durante el proceso de búsqueda. A continuación se detalla un pseudo-código de este tipo de búsqueda:

```
Función Buscar (estado_actual) devuelve Boolean
Comienzo

        Si EstadoFinal(estado_actual) entonces
                Devolver TRUE;
        Sino
                exito = FALSE;
                lista_reglas = ReglasAplicables (estado_actual);
                Mientras NOT exito AND NOT Vacia(lista_reglas)
                        estado_sucesor = AplicarRegla (lista_reglas);
                        exito = Buscar (estado_sucesor);
                Fin Mientras;
                Devolver exito;
        Fin Sino;
Fin Buscar
```

La ventaja de este método radica en que no es necesario tener almacenado todo el espacio de estados, sino sólo el camino que se está explorando. De esta forma puede encontrarse la solución sin tener que explorar gran parte del espacio de estados. No obstante, como desventajas de este algoritmo tenemos que se puede seguir una ruta infructuosa durante muchos pasos, y además la primera solución que encuentra puede distar mucho de ser la solución de mínima cantidad de pasos. Su correspondiente algoritmo consiste en:

1. Crear un alista de nodos llamada ABIERTA y asignarle l nodo raiz, que representa el estado inicial del problema planteado.
2. Hasta que se encuentre una Edo_Objetivo o se devuelva fallo realizar las siguientes acciones:

   2.1 Si ABIERTA esta vacía, terminar con fallo; en caso contrario continuar.

   2.2 Extraer el primer nodo de ABIERTA y denominarlo *m* .

2.3 Si la profundidad de *m* es igual a *lp* (limite de profundidad) regresar a 2; en caso contrario continuar.

2.4 Expandir *m* creando punteros hacia este nodo desde todos sus sucesores, de modo que pueda saberse cual es su predecesor.

Introducir dichos sucesores al principio de ABIERTA siguiendo un orden arbitrario. (la falta de orden refleja el carácter no informado de este procedimiento.)

2.4.1 Si algún sucesor de *m* es meta , abandonar el proceso iterativo señalado en 2 , devolviendo el camino de la solución , que se obtiene recorriendo los punteros de sus antepasados.

2.4.2 Si algún sucesor de *m* se encuentra en un callejón sin salida eliminarlo de ABIERTA y continuar el proceso iterativo en el paso 2.

3.2.1.4 Encadenamiento hacia atrás.

Una ligera variante de la búsqueda en profundidad es la *búsqueda de retroceso*. En esta última, en vez de generar todos los sucesores al expandir un nodo, se genera un único sucesor en cada paso, pero partiendo del estado objetivo hasta llegar al nodo raíz. Este método sirve más que nada como comprobación ya que verifica la «no falsedad» de un hecho o una solución, claro que necesariamente debe conocerse la solución e introducirse como dato de entrada, y los demás datos pueden introducirse cuando el programa lo requiera. Su fundamento se basa en el *Modus Tollens* que nos dice:

$$((R \rightarrow S) \wedge \sim R) \rightarrow \sim S$$

Que nos dice lo siguiente: Sí sucede R entonces sucede S, y como no sucedió R, por lo cual tampoco sucedió S. Para ilustrar supongamos las siguientes reglas:

Regla 1: Q si R y S	$(R \wedge S) \rightarrow Q$
Regla 2: R si X o T	$(X \vee T) \rightarrow R$
Regla 3: S si Y o Z	$(Y \vee Z) \rightarrow S$

Teniendo: X, Z; demostrar que es cierto Q. Ahora considerando lo que se tiene y aplicando la regla 1 se obtiene:

Q es cierta si R y S

Aplicando la regla 2 se obtiene:

R es cierta si existe X o T, y como existe X, R es cierta.

Aplicando la regla 3:

S es cierta si existe Y o Z, y como existe Z, S es cierta.

3.2.1.5 Encadenamiento mixto

El encadenamiento hacia delante es un proceso que parte del nodo raíz y se desplaza hacia el nodo objetivo mediante la aplicación de reglas a las condiciones iniciales. Por su parte, el encadenamiento hacia atrás parte del estado objetivo (solución del problema) y trabaja hacia el nodo raíz a través de las condiciones requisito. Cuando en un problema se ocuparon ambos tipos de encadenamiento (el primero de ellos para encontrar la solución, y el segundo para verificarla o comprobarla) se dice entonces que se tiene un encadenamiento mixto. También sirve para acortar el tiempo real de búsqueda cuando se usan al mismo instante, pues la búsqueda se da por terminada cuando al utilizar ambos encadenamientos simultáneamente se encuentran en un nodo común, de tal forma que cada encadenamiento realiza una parte de la búsqueda.

3.2.2 Búsquedas no ordenadas

Hay problemas que por sus propias características llegan a tener un gran conjunto de nodos o estados (espacio de búsqueda), dando como resultado la dificultad para recorrer o explorar sus estados en su totalidad. Es decir, se trata de problemas cuyo árbol a medida

que se va extendiendo en sus descendientes, a su vez estos también van aumentando en los suyos, y así sucesivamente, o sea, un árbol que crece exponencialmente[38]. Ante esta complicación se han visto como solución las llamadas búsquedas no ordenadas, que tienen por objetivo reducir el espacio de búsqueda a través de reglas que nos permiten minimizar el número de caminos de un árbol al escoger sólo aquellos en donde sea más probable que se encuentre la solución; para realizar esto se hace uso no sólo de los datos con los que se cuenta, sino también de la experiencia para indicar el camino a seguir en la solución del problema. Las más difundidas son las que enseguida presentamos.

3.2.2.1 Generación y prueba.

Es un método muy sencillo, básicamente se trata de una búsqueda primero en profundidad con backtracking. Consiste en lo siguiente:

1. Genera una posible solución (un estado particular, una trayectoria desde el estado inicial).
2. Si el estado generado es la solución, finalizar, en caso contrario, regresa a 1.

3.2.2.2 Escalada o ascenso por la colina (Hill-climbing).

Es una variante de «Generación y prueba» con la diferencia de que en él existe retroalimentación a partir del proceso de prueba que se usa para ayudar a decidirse por cual dirección debe moverse en el espacio de búsqueda. Su nombre se deriva de "la analogía de un alpinista perdido en la oscuridad, en mitad de una montaña; entonces, incluso en la oscuridad, el alpinista sabe que cualquier paso hacia arriba le sitúa en la dirección correcta".[39] Sin embargo este método heurístico presenta ciertos inconvenientes, pues puede no encontrar una solución cuando el programa se topa con un máximo pendiente (un estado que es mejor que todos sus vecinos, pero que no es mejor que otros

[38] A esta expansión exponencial del campo de búsqueda se le ha nombrado "explosión combinatoria".
[39] Herbert Schildt, *Utilización de C en la inteligencia artificial*, McGraw-Hill, México, p. 35

estados de otros lugares) o con una meseta (un área del espacio de búsqueda, en la que un conjunto de estados vecinos poseen el mismo valor).

Existen algunas formas de evitar los inconvenientes que presenta la heurística de «ascenso por la colina», pero no dan garantías. Uno de ellos es volver hacia atrás, a un nodo anterior, e intentar seguir un camino diferente. A fin de implementar esta estrategia se debe mantener una lista de los estados visitados. Otro sería realizar un gran salto en alguna dirección a fin de intentar buscar en una nueva parte del espacio de búsqueda. Su algoritmo correspondiente es el siguiente:

1. Generar la primer propuesta de solución y colocarlo en lista de soluciones.
2. Mientras no se encuentre la solución y haya estados que expandir
    2.1 Extraer un estado de la lista
        2.2 Aplicar las reglas necesarias para expandir el estado
    2.3 Para cada nuevo estado generado:
        - Si no es solución
            Ver si es el estado más cercano a la solución, en cuyo caso guardarlo, sino olvidarlo
        - Insertar en lista el mejor estado

Mostrar solución o indicar que no se encontró

3.2.2.3 Búsqueda del primero mejor o del menor coste (Algoritmo A*)

Combina las ventajas de los métodos primero en profundidad y primero en amplitud en un sólo método. En cada paso de un nodo a otro se selecciona el más promisorio de todos los nodos generados, es decir, se sigue un sólo camino y lo cambia cuando una ruta parece más prometedora. Si se llega a un estado objetivo se termina el proceso, en caso contrario, todos los nuevos nodos generados se colocan en el conjunto de estados generados. Para que el algoritmo sepa cuando cambiar un camino por otro mejor, se requiere de una

función de estimación sobre el costo necesario para llegar a una solución. Usualmente se detona como una función compuesta de la suma de dos componentes:

$$F' = g + h' \; ; \; g \ni \text{Valor Real}, \; h \ni \text{Estimación}$$

Donde la función «g» es una medida del costo para ir desde el Edo_Inicial al Nodo_Actual. Es decir, la función «g» no implica el mejor nodo, sino la mejor ruta, cuyo valor no es una estimación. Su valor es exactamente la suma de los costos de aplicación de las reglas que se han ido eligiendo.

Por su parte, la función «h» es una estimación del costo adicional necesario para alcanzar un nodo objetivo a partir del nodo actual, por lo que con «h'» se esta empleando el conocimiento acerca del dominio del problema. En conclusión, la función «F'» específicamente se compone de:

$$F' = \underset{\substack{\text{costo} \\ \text{Edo\_Inicial - Edo\_Actual}}}{g} + \underset{\substack{\text{estimación} \\ \text{Edo\_Actual – Edo\_Objetivo}}}{h'}$$

Si únicamente es relevante alcanzar el objetivo de la forma que sea, se puede definir la $g = 0$, de forma que se elige el nodo que parece más cercano al objetivo. Si lo que se desea es encontrar un camino con el menor número de pasos, se hace que el costo aplicado para pasar de un nodo a otro sea constante.

Por su parte, si «h'» hace una estimación perfecta de «h», el algoritmo converge inmediatamente hacia el objetivo. Si se acerca cuanto más a esta aproximación, mejor es la función «h'». Si por otro lado «h'» es siempre cero (no hay estimaciones) la función g será la que controle la búsqueda. Pero hay que tener cuidado, ya que si $g = 0$, entonces la búsqueda es aleatoria. Su pseudo-código correspondiente es el siguiente:

```
Estado_Inicial.g := 0;
Estado_Inicial.h := H(Estado_Inicial);
Estado_Inicial.f := Estado_Inicial.g + Estado_Inicial.h;

NodosNoExplorados := Estado_Inicial;
NodosExplorados := Ø;

Bool EncuentraEstadoFinal := FALSE;

Mientras NOT EncuentraEstadoFinal AND NodosNoExplorados != Ø

        EstadoActual := SeleccionarMejorNodo(NodosNoExplorados);
        NodosExplorados := NodosExplorados + EstadoActual;

        Si EsEstadoFinal (EstadoActual) entonces

                EncuentraEstadoFinal := TRUE;
        Sino
                Sucesores := CalcularSucesores (EstadoActual);
                Para cada EstadoSucesor de Sucesores
                        EstadoSucesor.padre := EstadoActual;
                        EstadoSucesor.g: EstadoActual.g +
                        Coste(EstadoActual,EstadoSucesor);

                        Si Pertenece(EstadoSucesor, NodosNoExplorados) entonces
                                EstadoViejo:=ExtraerNodo(EstadoSucesor,
                                NodosNoExplorados);
                                EstadoActual.sucesores := EstadoActual.sucesores +
                                EstadoViejo;
                                Si EstadoViejo.g > EstadoSucesor.g entonces
                                        EstadoViejo.padre := EstadoActual;
                                        EstadoViejo.g := EstadoActual.g;
                                        EstadoViejo.f := EstadoViejo.g + EstadoViejo.h;
                                Fin si;
                        Sino si Pertenece(EstadoSucesor, NodosExplorados) entonces
                                EstadoViejo:=ExtraerNodo(EstadoSucesor,
                                NodosNoExplorados);
                                EstadoActual.sucesores := EstadoViejo;
                                Si EstadoViejo.g > EstadoSucesor.g entonces
                                        EstadoViejo.padre := EstadoActual;
                                        EstadoViejo.g := EstadoActual.g;
                                        EstadoViejo.f := EstadoViejo.g + EstadoViejo.h;
                                        ActualizarSucesores(EstadoViejo);
                                Fin si;
                        Sino
```

```
                    NodosNoExplorados := NodosNoExplorados +
                    EstadoSucesor;
                    EstadoActual.sucesores :=EstadoActual.sucesores +
                    EstadoSucesor;
                    EstadoSucesor.f := EstadoSucesor.g + EstadoSucesor.h;
                fin sino;
            Fin para;
        Fin sino;
Fin mientras;
```

Esta heurística es recomendable para juegos simples unipersonales (por ej. 8-puzzle, Batumi, etc.), o para problemas donde sólo se necesite dar las condiciones iniciales y reglas para que la computadora realice el cómputo por sí misma. Su gran ventaja reside en que siempre que exista solución se acaba por encontrarla.

3.2.2.4 Método Min-Max

Esta heurística es recomendable para problemas que presentan decisiones con incertidumbre involucrando dos o más oponentes *inteligentes,* donde cada oponente aspira a optimizar su propia decisión pero a costa de los otros. Es decir, se puede aplicar a campañas publicitarias para productos competitivos y a tácticas destinadas entre contendientes. Sin embargo, sus aplicaciones más difundidas son sobre los juegos, en donde el usuario es el contendiente de la computadora, por ejemplo, el juego del gato o tres en linea.

El nombre del algoritmo deriva de considerar que, dada una función estática que devuelve valores en relación a uno de los jugadores, éste procura maximizar su valor mientras que su oponente procura minimizarlo. Por lo anterior, al jugador que pretende maximizar su valor se le denota como 'Max', y a su oponente como 'Min'. En un árbol de juego donde los valores de la función estática están en relación al jugador que pretende maximizar, se maximiza y minimiza alternadamente de un nivel a otro. La figura 3.10 muestra un ejemplo de los mejores movimientos por parte del jugador 'Max' quien juega con 'X', según la búsqueda Mini-Max:

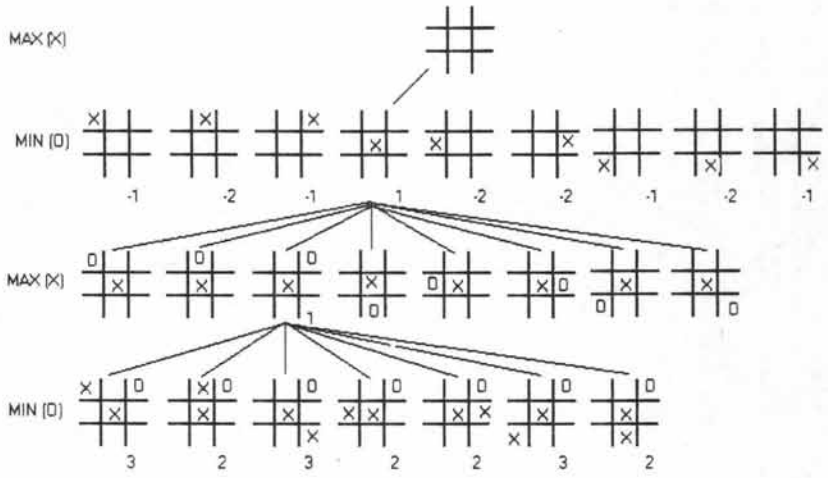


Fig. 3.10, Árbol para juego del gato por algoritmo Min-Max

Como podemos ver, la idea consiste en comenzar en la posición actual y usar el generador de movimientos plausibles para generar las posibles posiciones sucesivas hasta un cierto límite de niveles. A continuación se aplica la función de evaluación estática a las posiciones obtenidas y se elige la mejor posición para el jugador correspondiente, llevando los valores un nivel hacia atrás para continuar la evaluación en todos los niveles anteriores. La función de evaluación estática devuelve valores positivos para indicar buenas situaciones y valores negativos para indicar buenas situaciones para el oponente. Por lo tanto, la meta es maximizar el valor de la función estática de la siguiente posición de tablero por parte del jugador 'Max', y minimizar el valor por parte del oponente 'Min', de tal forma que tras la primer mejor jugada de Max, la mejor respuesta de Min es:

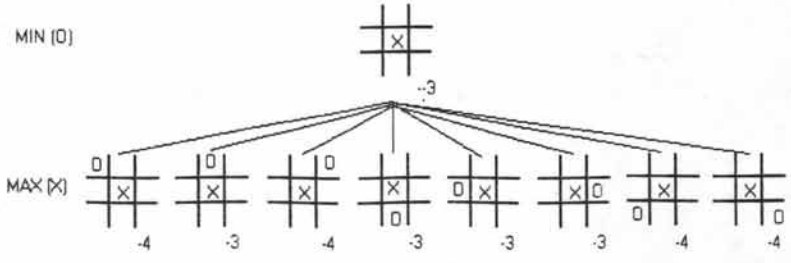


Fig. 3.11, Mejor respuesta de Min

El algoritmo Mini-Max es el siguiente:

```
MINIMAX( posición, profundidad, jugador)
Max = JUGADOR-UNO
Min = JUGADOR-DOS
Comienzo
        Si SUFICIENTE (posición, profundidad) entonces
                resultado.VALOR := ESTATICA (posición, jugador);
                resultado.CAMINO := Nulo;
                return resultado;
        Sino
                Sucesores := GENMOV (posición, jugador);
                Si EstaVacia (Sucesores) entonces
                        resultado.VALOR := ESTATICA (posición, jugador);
                        resultado.CAMINO := Nulo;
                        return resultado;
                Sino
                        resultadoMejor.VALOR := MININT;
                        por cada sucesor de Sucesores
                                resultadoSucesor := MINIMAX (sucesor, profundidad+1,
                                CONTRARIO (jugador));
                                Si resultadoMejor.VALOR < - resultadoSucesor.VALOR
                                entonces
                                        resultadoMejor.VALOR := -
                                        resultadoSucesor.VALOR;
                                        resultadoMejor.CAMINO := sucesor +
                                        resultadoSucesor.CAMINO;
                                fin si;
                        fin por;
                        return resultadoMejor;
                fin sino;
        fin sino;
fin MINIMAX;
```

Donde la primera llamada a la función recursiva será:

MINIMAX (posiciónactual, 0, JUGADOR-UNO)

Además el algoritmo MINIMAX(posición, profundidad, jugador) usa dos funciones:

1) GENMOV(posición, jugador): función que devuelve la lista de movimientos posibles para el jugador a partir de la posición.

2) ESTÁTICA(posición, jugador): función que devuelve un número que representa la conveniencia de la posición desde el punto de vista de jugador, es decir, devuelve valores en relación al jugador actual en vez del jugador maximizante. Por esta razón, en lugar de realizar maximización y minimización alternativas, siempre se maximiza el valor que devuelve esta función.

Por lo tanto, la función MÍNIMAS devuelve una estructura con:

- VALOR: que representa el valor propagado de la función estática.
- CAMINO: que es camino para llegar a la solución desde la posición actual.

Así, el procedimiento recursivo de MÍNIMAS termina por alguna de las siguientes condiciones:

- Gana algún jugador
- Se han explorado N capas, siendo N el límite establecido.
- Se ha agotado el tiempo de exploración.
- Se ha llegado a una situación estática donde no hay grandes cambios de un nivel a otro.

3.3 EJEMPLOS DE PROBLEMAS RESUELTOS MEDIANTE LA MANIPULACIÓN Y REPRESENTACIÓN DEL CONOCIMIENTO.

Existe un gran número de problemas que pueden ser resueltos mediante técnicas de manipulación y representación del conocimiento. El propósito de éste apartado es sólo poner algunos ejemplos en donde la solución se da mediante el empleo de alguna de las

técnicas descritas anteriormente. Únicamente se indican las características propias de cada problema y la forma en que se llega a la solución aplicando una estructura de control. No obstante, un CD que acompaña este trabajo integra los programas de los problemas aquí tratados.

3.3.1 El cruce del río (resuelto mediante el método primero en anchura)

Hay un hombre que es dueño de una gallina, un perro y un costal de maíz, y necesita cruzar un profundo río llevando consigo todas sus pertenencias de un extremo del río a otro. El problema es que sólo dispone de una balsa muy pequeña en la que sólo cabe él (que tiene que ir remando) y una de sus pertenencias, así que tiene que llevarlas a la orilla una por una (figura 3.12). Pero si en algún momento deja al perro con la gallina, éste se la come. Y si deja sola a la gallina con el costal de maíz, la gallina rompe el costal regando el maíz y comiéndose una parte. El objetivo es formalizar el problema y resolverlo mediante una estructura de control.

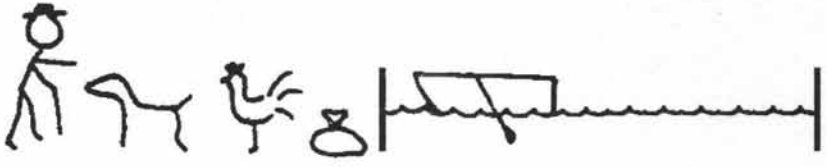


Figura 3.12, Esquema del problema del cruce del río

Primero se formaliza el problema para que podamos trabajar más fácil. Por ello se plante un arreglo con cuatro letras que representan a los elementos (individuos) que entran en juego:

Río(H, P, G, M)

Donde correspondientemente:

H = Hombre
G = Gallina
P = Perro
M = costal de maíz

Los posibles estados son:

Río (	H	P	G	M)	
Río (	0,	0,	0,	0)	Estado Inicial
Río (	1,	1,	1,	1)	Estado Objetivo
Río (	0,	1,	1,	0)	Estado prohibido
Rió (	0,	0,	1,	1)	Estado prohibido
Río (	0,	1,	1,	1)	Estado prohibido
Río (	1,	0,	0,	0)	Estado prohibido
Río (	1,	1,	0,	0)	Estado prohibido
Río (	1,	0,	0,	1)	Estado prohibido
Río (	0	1,	0,	1)	Estado no riesgo
Río (	1,	0,	1,	0)	Estado no riesgo

Las reglas aplicables que permiten pasar de un estado a otro son:

1) Pasar al Hombre:

   $(H, P,G,M) \longrightarrow (H+1, P, G, M); P \neq G, G \neq M$

2) Pasar al Hombre y al Perro:

   $(H, P,G,M) \longrightarrow (H+1, P+1, G, M); H=P, G \neq M$

3) Pasar al Hombre y a la Gallina:

   $(H, P,G,M) \longrightarrow (H+1, P, G+1, M); H = G$

4) Pasar al Hombre y al Maíz:

   $(H, P,G,M) \longrightarrow (H+1, P, G, M+1); H = M, G \neq P$

5) Regresar al Hombre:

   $(H=1, P,G,M) \longrightarrow (H-1, P, G, M); P \neq G, G \neq M$

6) Regresar al Hombre y al Perro:

   $(H=1, P=1,G,M) \longrightarrow (H-1, P-1, G, M); P = H, G \neq M$

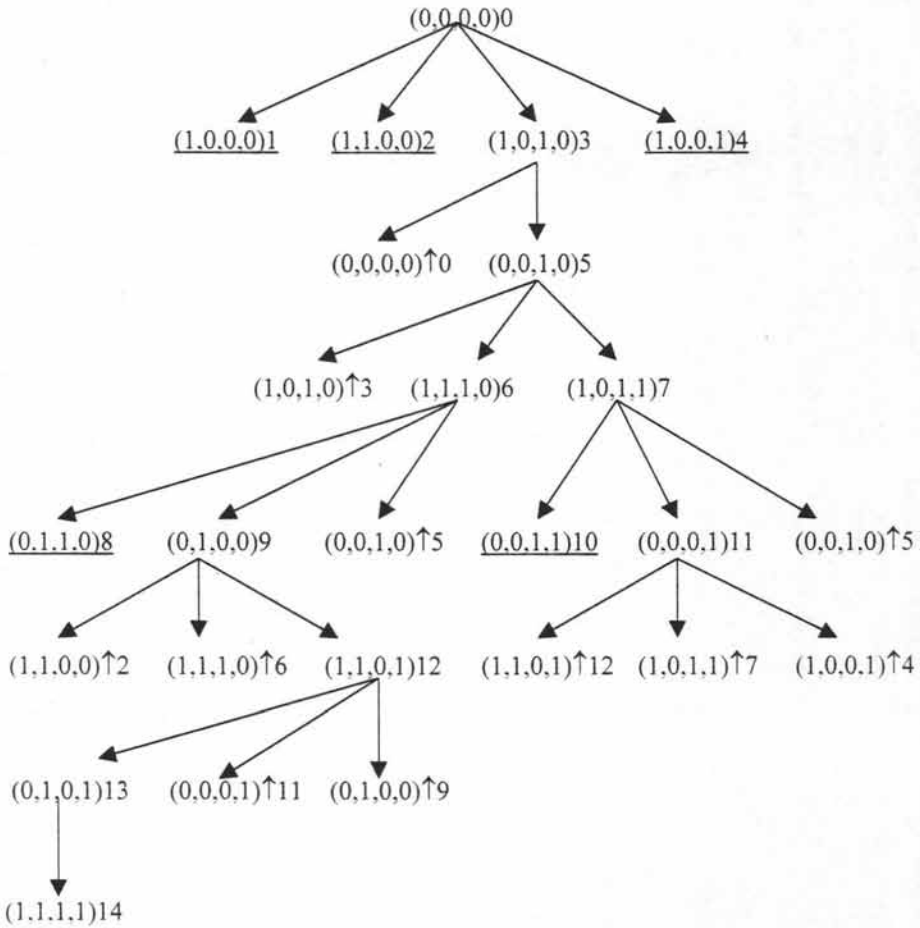
7) Regresar al Hombre y a la Gallina:

   $(H=1, P,G=1,M) \longrightarrow (H-1, P, G-1, M); H = G$

8) Regresar al Hombre y al Maíz:

   $(H=1, P,G,M=1) \longrightarrow (H-1, P, G, M-1); H \neq M, G \neq P$

Mediante la búsqueda en anchura se tendría la solución siguiente:



Como es una búsqueda en anchura se generan todos los nodos-estados de cada nivel y se van marcando los nodos prohibidos (se subrayan). A la derecha de cada nodo hay un número para indicar el orden en que fue generado, de tal forma que cada vez que se da un nodo que ya se tenia con anterioridad, se remite a él mediante una flecha hacia arriba y un número que indica de qué nodo se trataba. La solución más corta es a través del recorrido siguiente:

$$0 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 12 \rightarrow 13 \rightarrow 14$$

Que corresponde a los estados siguientes:

NIVEL	No.NODO	VALORES (H,P,G,M)	DESCRIPCIÓN
0	0	(0,0,0,0)	Es el punto inicial
1	3	(1,0,1,0)	El Hombre pasa a la Gallina
2	5	(0,0,1,0)	El Hombre regresa
3	6	(1,1,1,0)	El Hombre pasa con el Perro
4	9	(0,1,0,0)	El Hombre regresa con la Gallina
5	12	(1,1,0,1)	El Hombre pasa con el Maíz
6	13	(0,1,0,1)	El Hombre regresa
7	14	(1,1,1,1)	El Hombre pasa con la Gallina

3.3.2 Laberinto (resuelto mediante el método primero en profundidad)

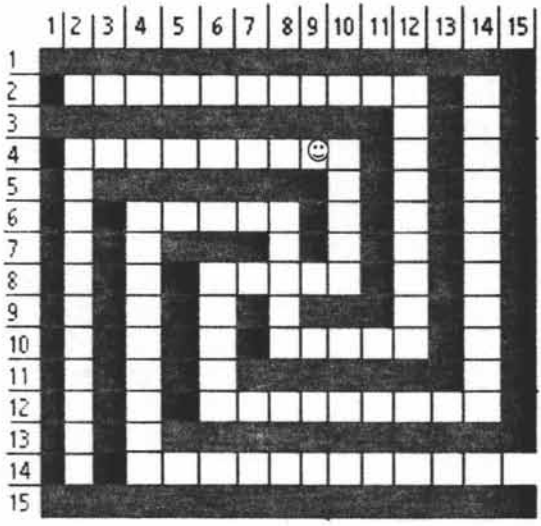
La mitología griega nos relata la historia del héroe Teseo que entró a un laberinto para hallar y matar al monstruo minotauro. Le ayudó Ariadna dándole un ovillo de hilo. Teseo fue desenredándolo conforme se adentraba en el laberinto, mientras Ariadna sujetaba uno de los extremos del hilo, para que al enredarlo Teseo encontrase fácilmente la salida.

Supongamos que hay un hombre que se halla perdido en el punto señalado (donde esta la carita) al interior del siguiente laberinto:

Figura 3.13, Laberinto

El objetivo es formalizar el problema y resolverlo para que el hombre sea capaz de encontrar la salida del laberinto.

Primeramente se establece un sistema de coordenas cartesianas con el fin de poder tener un control sobre el desplazamiento a través del laberinto, así tenemos una matriz de 15X15. Se señala el estado inicial y el estado objetivo, junto con las reglas aplicables, con lo que cual tenemos lo siguiente:



Desplazamiento [eje X][eje Y]

Abreviando sería:

D [X][Y]

El estado Inicial es:

D[9][4]

Y el estado objetivo es:

D[15][14]

Tal que:

X es creciente

Y es creciente

La reglas aplicables partiendo de la posición inicial (D[X][Y]) son:

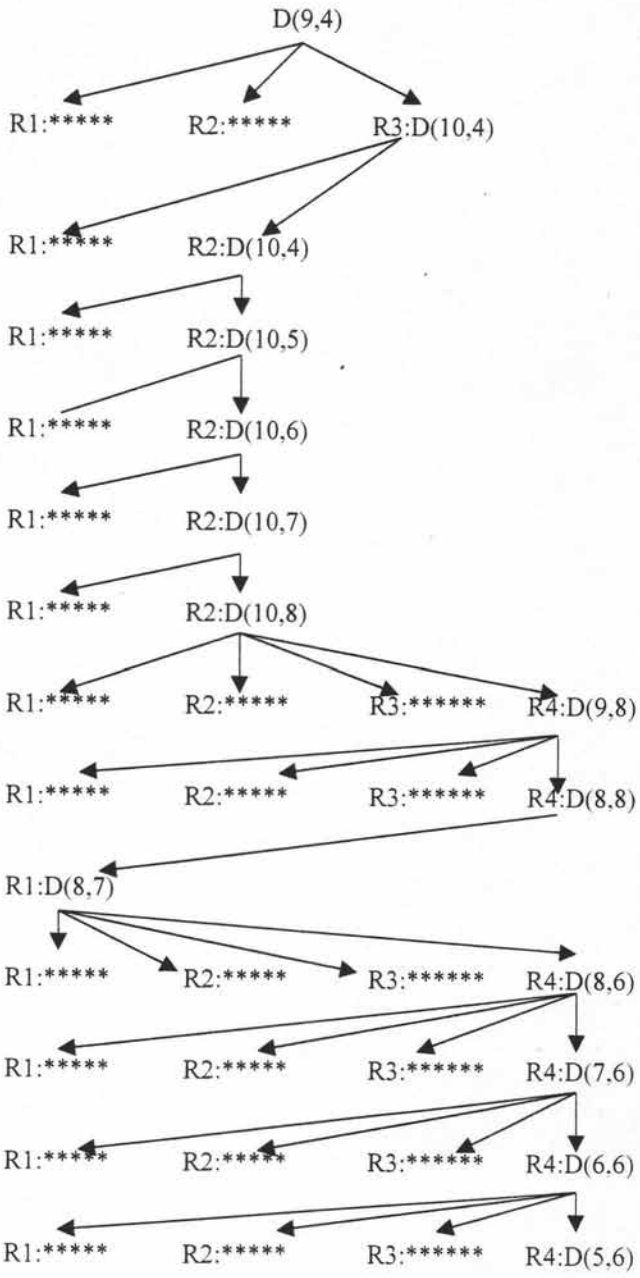
Regla 1: Hacia arriba: D[X][Y]|Y>1, Y-1= Movimiento→ D(X, Y-1)

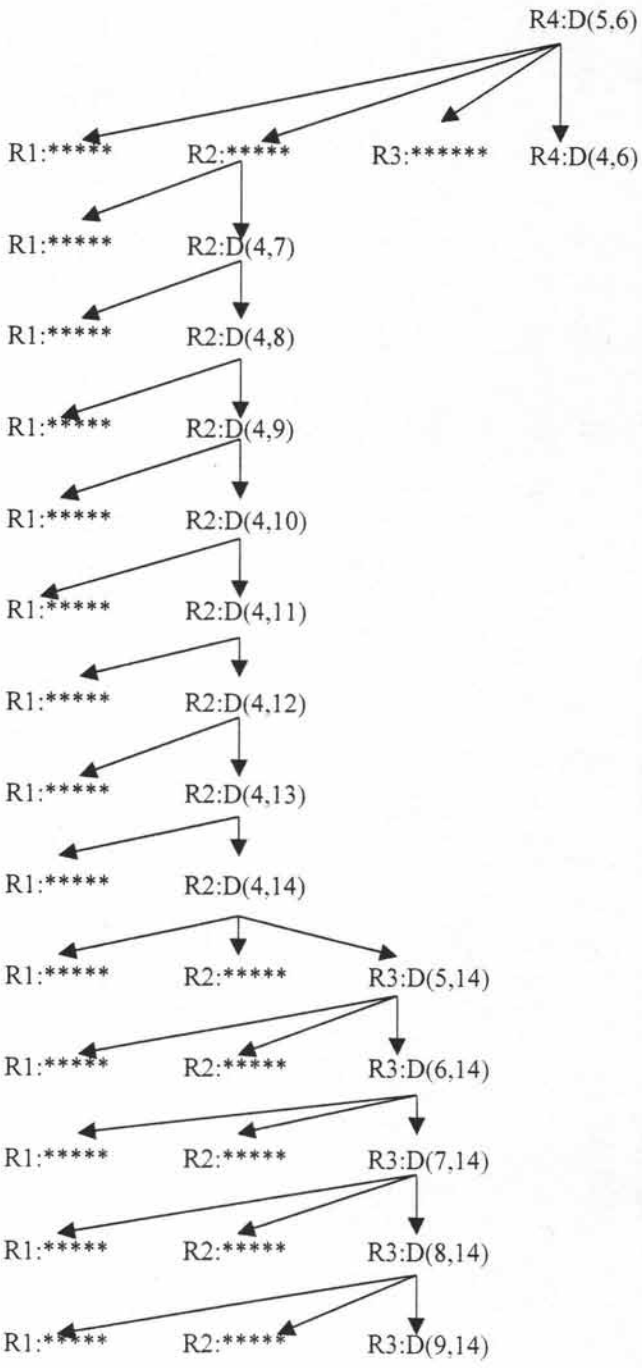
Regla 2: Hacia abajo: D[X][Y]|Y<16, Y+1= Movimiento→ D(X, Y+1)

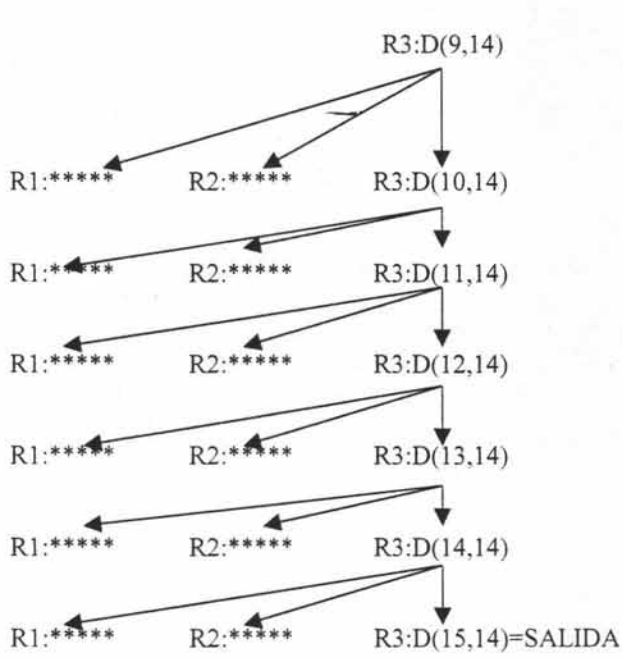
Regla 3: Derecha: D[X][Y]|X<16, X+1= Movimiento→ D(X+1, Y)

Regla 4: Izquierda: D[X][Y]|X>1, X-1= Movimiento→ D(X-1, Y)

Aplicando la búsqueda primero en profundidad tenemos:


D(9,4)
R1:*****
R2:*****
R3:D(10,4)
R1:*****
R2:D(10,4)
R1:*****
R2:D(10,5)
R1:*****
R2:D(10,6)
R1:*****
R2:D(10,7)
R1:*****
R2:D(10,8)
R1:*****
R2:*****
R3:******
R4:D(9,8)
R1:*****
R2:*****
R3:******
R4:D(8,8)
R1:D(8,7)
R1:*****
R2:*****
R3:******
R4:D(8,6)
R1:*****
R2:*****
R3:******
R4:D(7,6)
R1:*****
R2:*****
R3:******
R4:D(6,6)
R1:*****
R2:*****
R3:******
R4:D(5,6)


R4:D(5,6)
R1:*****
R2:*****
R3:******
R4:D(4,6)
R1:*****
R2:D(4,7)
R1:*****
R2:D(4,8)
R1:*****
R2:D(4,9)
R1:*****
R2:D(4,10)
R1:*****
R2:D(4,11)
R1:*****
R2:D(4,12)
R1:*****
R2:D(4,13)
R1:*****
R2:D(4,14)
R1:*****
R2:*****
R3:D(5,14)
R1:*****
R2:*****
R3:D(6,14)
R1:*****
R2:*****
R3:D(7,14)
R1:*****
R2:*****
R3:D(8,14)
R1:*****
R2:*****
R3:D(9,14)



Como es una búsqueda en profundidad, se aplica la primer regla hasta generar toda una rama, pero en caso de que la regla no sea aplicable (hecho que se señala con asteriscos), entonces se pasa a ver si la siguiente regla lo es, de tal forma que no se generan todos los nodos de cada nivel, si no sólo los necesarios que permitan seguir por una rama.

3.3.3 El teorema de los cuatro colores (resuelto por el método generación y prueba)

Una de las grandes conjeturas de más difícil demostración de las matemáticas, es la del famoso teorema de los cuatro colores. ¿Cuántos colores son necesarios para colorear un mapa arbitrario de manera que nunca dos regiones colindantes sean del mismo color?. Se conjeturaba que son necesarios solamente cuatro colores.

Desde mediados del siglo XIX han sido varios los matemáticos que intentaron infructuosamente demostrar este teorema hasta que, gracias a la ayuda de la informática, se consiguió una prueba a mediados de los años 70.

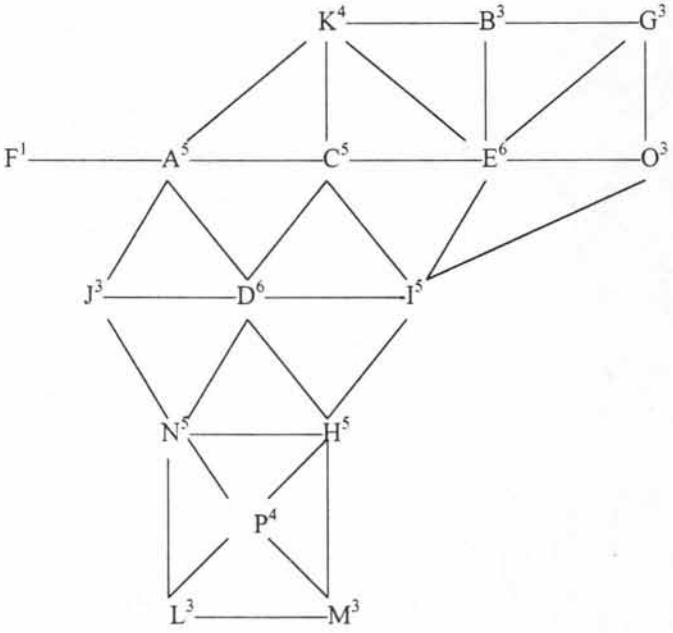
De acuerdo con el «Teorema de los cuatro colores», tiene que ser posible colorear el mapa de las Delegaciones del Distrito Federal utilizando solamente cuatro colores, y de tal manera que las comunidades colindantes sean de diferente color. El objetivo es formalizar este problema y desarrollar el código necesario para realizar la búsqueda de la solución al problema planteado.

1. Álvaro Obregón = A
2. Azcapotzalco = B
3. Benito Juárez = C
4. Coyoacan = D
5. Cuahutemoc = E
6. Cuajimalpa = F
7. Gustavo A. Madero = G
8. Iztapalapa = H
9. Iztacalco = I
10. Magdalena Contreras = J
11. Miguel Hidalgo = K
12. Milpa Alta = L
13. Tlahuac = M
14. Tlalpan = N
15. Venustiano Carranza = O
16. Xochimilco = P

Para facilitar el manejo de los nombres de cada delegación les asignamos una letra del alfabeto, al mismo tiempo que también a cada color le asignamos un número.

0 = SIN COLOR 1 = ROJO 2 = VERDE
3 = AZUL 4 = ROSA

El siguiente grafico muestra como colindan entre sí las delegaciones, ya que si comparten la misma frontera no pueden tener el mismo color. El subíndice de cada letra mayúscula indica el número de fronteras compartidas de dicha delegación.



Determinado las fronteras entre delegaciones tenemos que:

AF[40]	BK	CK	DI	EK	GO	HI	IO	JN	LN	MP	NP
AK	BE	CE	DH	EG		HM			LP		
AC	BG	CI	DN	EO		HN			LM		
AD		CD	DJ	EI		HP					
AJ											

Para ilustrar la forma de interpretar la tabla anterior tomemos el caso de (AF). que nos dice que 'A' y 'F' son colindantes, 'A' representa a la delegación 'Álvaro Obregón', por su parte, 'F' representa a 'Cuajimalpa'. Por lo tanto, ambas no pueden tomar el mismo valor, es decir, no pueden tener el mismo color en el Mapa de Delegaciones del D. F.

[40] La relación AF es igual a FA, por tal motivo sólo basta con poner una de ellas. Lo anterior se aplica a todas las demás relaciones de unión entre fronteras.

El estado inicial es:

TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

El estado objetivo es:

TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P) = (2,1,1,3,4,3,2,1,2,4,3,3,2,2,3,4)

Y las reglas aplicables para realizar cambios de un estado a otro son:

Pintar A = Álvaro Obregón es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(0<A<5,B,E,G,H,I,L,M,N,O,P);A≠C,D,F,J,K

Pintar B = Azcapotzalco es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(A,0<B<5,C,D,F,H,I,J,L,M,N,O,P);B≠K,E,G

Pintar C = Benito Juárez es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(B,0<C<5,F,G,H,J,L,M,N,O,P);C≠A,K,E,I,D

Pintar D = Coyoacan es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(0<D<5,B,E,F,G,K,L,M,O,P);D≠A,C,I,H,N,J

Pintar E = Cuauhtemoc es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(0<E<5,A,D,F,H,J,L,M,N,P);E≠B,C,O,I,G,K

Pintar F = Cuajimalpa es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(0<F<5,B,C,D,E,G,H,I,J,K,L,M,N,O,P);F≠A

.
.
.

Pintar N = Tlalpan es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(0<N<5,A,B,C,E,F,G,I,K,M,O,);A≠H,D,P,J,L

Pintar O = V. Carranza es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(0<O<5,A,B,C,D,F,H,J,K,L,M,N,P);0≠G,E,I

Pintar P = Xochimilco es:
TCC(A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P)→(A,B,C,D,E,F,G,I,J,K,O,0<P<5);P≠H,L,M,N

Realizando una búsqueda tenemos:

NODO	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0
3	0	1	0	0	4	0	2	0	0	0	0	0	0	0	0	0
4	0	1	0	0	4	0	2	0	0	0	3	0	0	0	0	0
5	0	1	0	0	4	0	2	0	0	0	3	0	0	0	3	0
6	0	1	0	0	4	0	2	0	2	0	3	0	0	0	3	0
7	0	1	1	0	4	0	2	0	2	0	3	0	0	0	3	0
8	0	1	1	0	4	3	2	0	2	0	3	0	0	0	3	0
9	2	1	1	0	4	3	2	0	2	0	3	0	0	0	3	0
10	2	1	1	0	4	3	2	0	2	4	3	0	0	0	3	0
11	2	1	1	3	4	3	2	0	2	4	3	0	0	0	3	0
12	2	1	1	3	4	3	2	1	2	4	3	0	0	0	3	0
13	2	1	1	3	4	3	2	1	2	4	3	0	2	0	3	0
14	2	1	1	3	4	3	2	1	2	4	3	0	2	0	3	4
15	2	1	1	3	4	3	2	1	2	4	3	0	2	2	3	4
16	2	1	1	3	4	3	2	1	2	4	3	3	2	2	3	4

En nuestro ejemplo tenemos la generación de una trayectoria satisfactoria, pues del Estado Inicial se llega al Estado Objetivo. Cuando la posible solución generada no es correcta, entonces se tiene que regresar al primer nodo generado (no al 0, sino al 1) y desde ahí generar otra posible solución, si otra vez no se llega al Estado Objetivo se debe regresa al Nodo N+1 (es decir el Nodo 2) para volver intentar otra solución, y así sucesivamente.

En nuestro ejemplo el primero en pintar fue B (Azcapozalco) de Rojo (igual 2), después se siguió con G (Gustavo A. Madero) por ser una delegación colindante, y así sucesivamente hasta terminar un intento. La búsqueda termina al encontrar un estado satisfactorio, el cual consiste en:

Rojo: B, C, H

Verde: A, G, I, M, N

Azul: D, F, K, L, O

Rosa: E, J, P

3.3.4 El juego de los once cubos (min-max)

Se tienen once cubos sobre una mesa. El primer jugador toma 1, 2 ó 3 cubos. A continuación, el segundo jugador toma 1, 2 ó 3 de los cubos restantes. Después es nuevamente el turno del primer jugador, y se siguen alternando hasta que no queden cubos. El jugador que tome el último cubo es el perdedor.

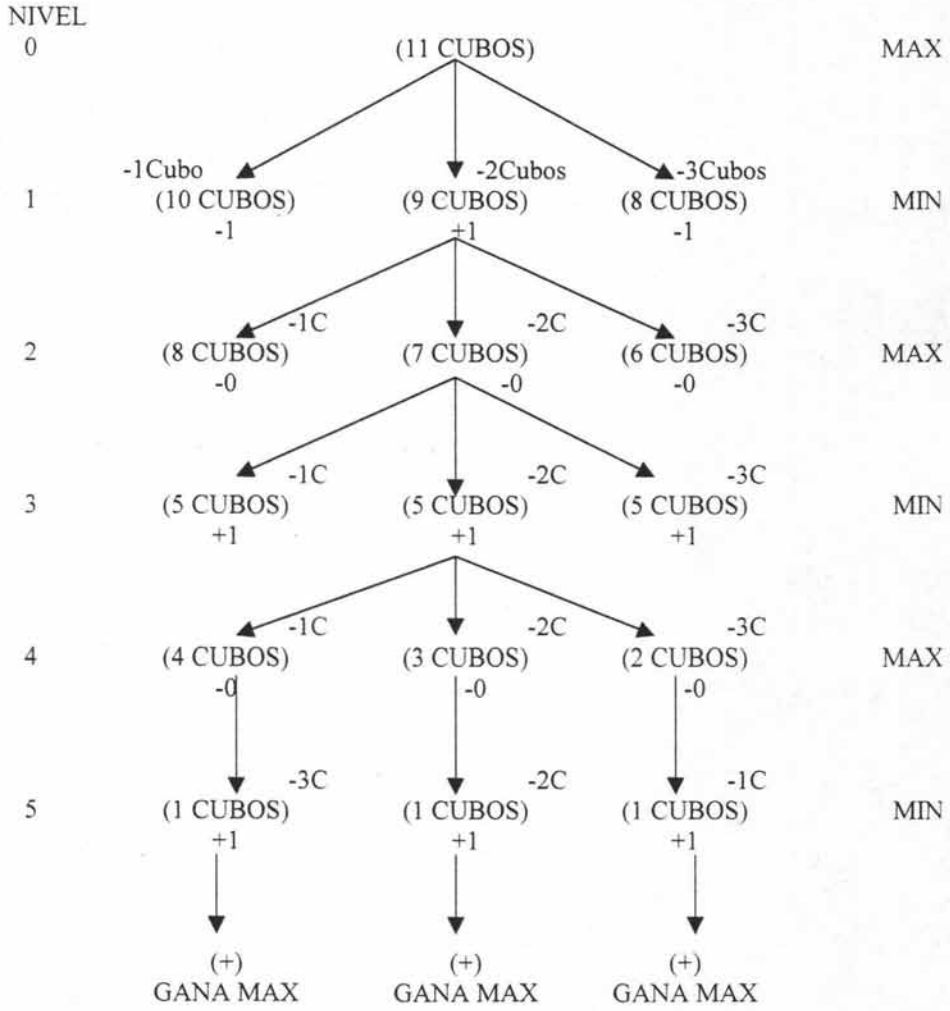
El objetivo es desarrollar una estrategia mediante el algoritmo min-max para el computador gane en la mayoría de casos, ya sea que tenga el papel del jugador A o de B. Y su oponente sea un usuario. Las características propias del juego son:

i. En el juego participan dos jugadores que toman turno alternativamente para hacer una jugada.
ii. El juego termina sólo con uno de los dos resultados posibles: a) el jugador que realiza la primera jugada llamado MAX es el ganador, o bien, b) gana el otro jugador llamado MIN.
iii. Cada jugada consiste en una elección que hace cada jugador de un conjunto de posibles jugadas (estados).
iv. En cualquier punto del juego ambos jugadores tienen toda la información de las jugadas que ya se han hecho y las que pueden hacerse.
v. Existe un límite superior para el número de jugadas en una partida.

Las reglas aplicables son:

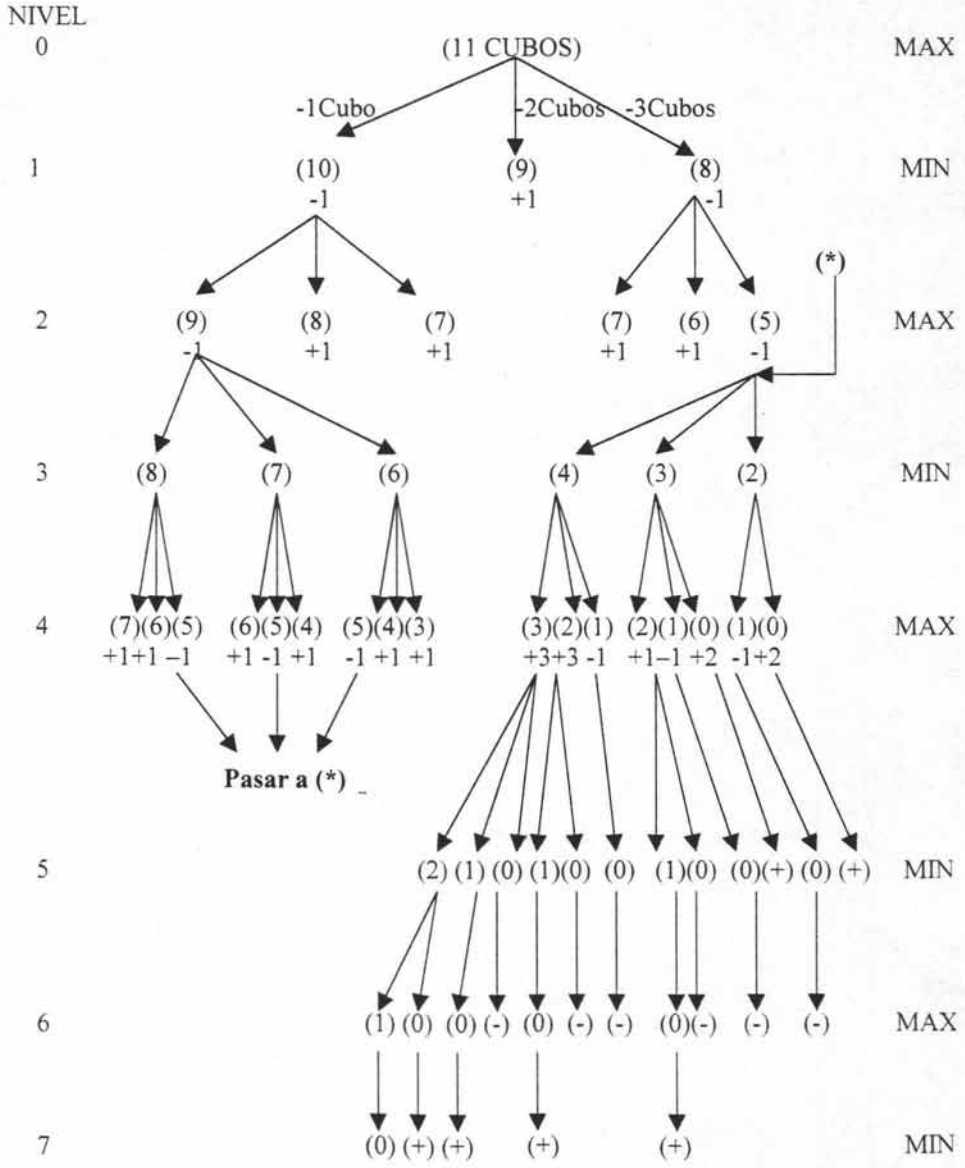
Regla 1: MAX toma X; tal que 0<X<4 si CantidadDeCubos <= 11
Regla 2: MIN toma X; tal que 0<X<4 si CantidadDeCubos <= 11
Regla 3: MAX pretende aumentar su valor inicial tras cada jugada.
Regla 4: MIN pretende minimizar su valor inicial tras cada jugada.
Regla 5: MAX siempre comienza el juego.

Cuando MAX es la computadora y MIN el usuario, tenemos:

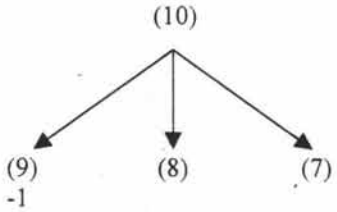


Si MAX selecciona en su primera tirada la opción que maximiza su valor (la de levantar dos cubos), cuando es el turno de MIN ya no hay una función óptima que disminuya su valor, por lo tanto, todas sus opciones valen (-0). A partir de entonces, si MAX sigue una buena estrategia es seguro que gane el juego.

Ahora, cuando MAX es el usuario y MIN la computadora, y se da el caso de que MAX no tomó la mejor opción en su primera tirada, entonces MIN tiene gran posibilidad de ganar el juego. La estrategia más óptima para tal situación es:



Por motivos de espacio, entre paréntesis se indica el número de cubos que van quedando tras cada posible jugada. También únicamente se desarrolla la ruta óptima (la que minimiza) para MIN. Las fechas hacia abajo indican el nuevo estado al que se llega al retirar 1, 2 o 3 cubos respectivamente, de izquierda a derecha así:



La flecha a la izquierda significa que de 10 cubos, al retirar 1 quedan nueve; la de en medio señala que quedan 8 al retirar dos; y la de la derecha índica que de 10 cubos al retirar tres nos quedan 7. El número ubicado debajo de cada estado, es la estimación de optimización para cada jugada, el (-1) significa que se trata de la mejor opción para MIN y la peor para su contrincante. Cuando al final de un camino nos encontramos con (+), significa que el jugador MAX ha ganado el juego y MIN lo ha perdido. Pero si tenemos un (-). entonces MIN ha ganado y MAX perdió.

Algunas posibles partidas que puede tenerse son:

JUEGOS	MAX	MIN	MAX	MIN	MAX	MIN	MAX	GANA
1	11-1 = 10	10-1=9	9-1=8	8-3=5	5-3=2	2-1=1	1-1=0	MIN
2	11-1 = 10	10-2=8	8-1=7	7-3=4	4-1=3	3-2=1	1-1=0	MIN
3	11-1 = 10	10-3=7	7-1=6	6-1=5	5-3=2	2-1=0		MAX
4	11-2 = 9	9-2=7	7-2=5	5-1=4	4-3=1	1-1=0		MAX
5	11-2 = 9	9-1=8	8-2=6	6-3=3	3-1=2	2-1=1	1-1=0	MIN
6	11-2 = 9	9-3=6	6-3=3	3-2=1	1-1=0			MIN
7	11-3 = 8	8-1= 7	7-3=4	4-1=3	3-2=1	1-1=0		MAX
8	11-3 = 8	8-2 =6	6-2=4	4-2=2	2-2=0			MIN
9	11-3 = 8	8-3 =5	5-2=3	3-1=2	2-1=1	1-1=9		MAX

El jugador que tiene el último tiro (el que levanta de 1 a 3 cubos dejando 0) es quien pierde.

3.3.5 El mejor camino entre Capitales de los Estados de México

El objetivo es encontrar la ruta más corta entre una ciudad capital y otra de los Estados de México. Para ello se deben considerar las distancias de las carreteras disponibles, y la distancia en línea recta entre capitales, para determinar una función de estimación de costos por cada decisión que se tome.

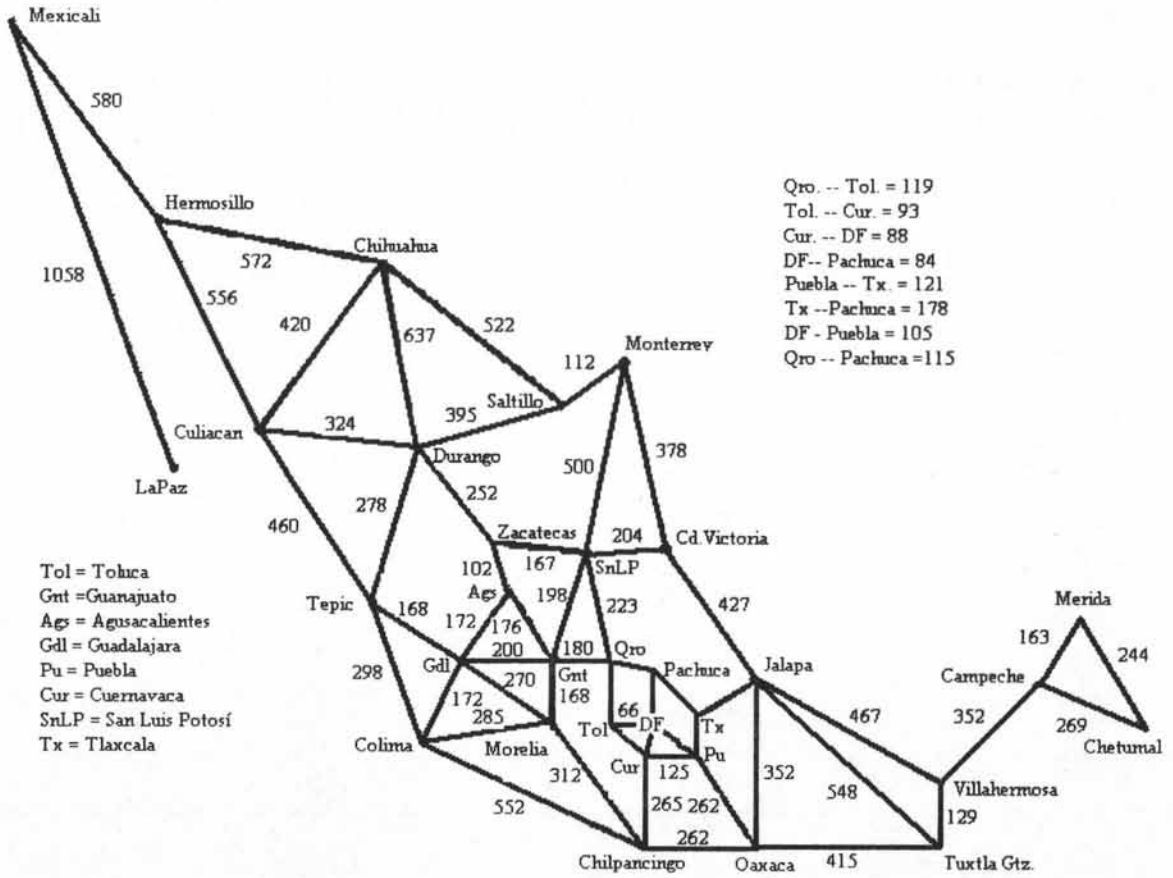
Para ilustrar, supóngase que el punto de salida es:

Durango - Durango

Y el punto objetivo es:

Morelia - Michoacán

	CAPITAL	ESTADO
1	Aguascalientes	Aguascalientes
2	Campeche	Campeche
3	Ciudad Victoria	Tamaulipas
4	Colima	Colima
5	Cuernavaca	Morelos
6	Culiacán	Sinaloa
7	Chetumal	Quinta Roo
8	Chihuahua	Chihuahua
9	Chilpancingo	Guerrero
10	D. Federal	Ciudad de México
11	Durango	Durango
12	Guadalajara	Jalisco
13	Guanajuato	Guanajuato
14	Hermosillo	Sonora
15	Jalapa	Veracruz
16	La paz	Baja California Sur
17	Mérida	Yucatán
18	Mexicali	Baja California Nt.
19	Monterrey	Nuevo León
20	Morelia	Michoacán
21	Oaxaca	Oaxaca
22	Pachuca	Hidalgo
23	Puebla	Puebla
24	Querétaro	Querétaro
25	Saltillo	Coahuila
26	S. Luis Potosí	San Luis Potosí
27	Tepic	Nayarit
28	Tlaxcala	Tlaxcala
29	Toluca	Estado de México
30	Tuxtla Gutiérrez	Chiapas
31	Villahermosa	Tabasco
32	Zacatecas	Zacatecas

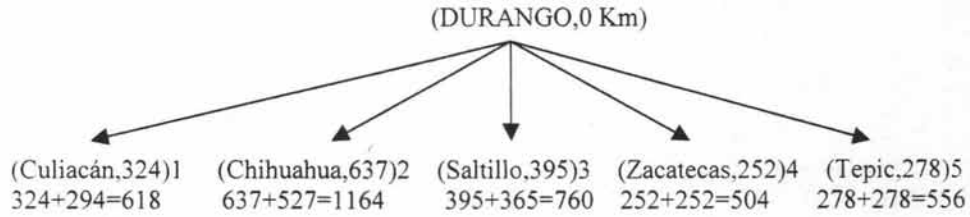


Caminos disponibles entre capitales de un estado a otro con distancia en Km.

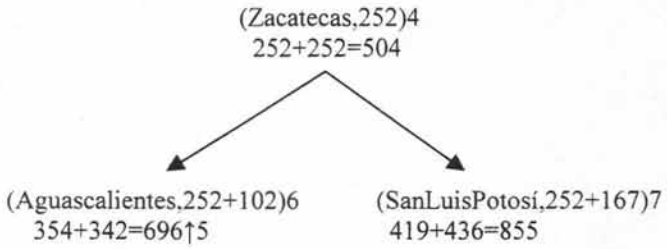
Distancias en línea Recta de Durango a las demás capitales de los Estados de México:

Capital	Distancia	Capital	Distancia
Aguascalientes	342	Mérida	1558
Campeche	1447	Mexicali	1506
Cd. Victoria	511	Monterrey	460
Colima	505	Morelia	596
Cuernavaca	798	Oaxaca	1138
Culiacán	294	Pachuca	758
Chetumal	1809	Puebla	864
Chihuahua	527	Querétaro	611
Chilpancingo	896	Saltillo	416
DF	764	Sn. Luis Potosí	436
Durango	***	Tepic	278
Guadalajara	415	Tlaxcala	825
Guanajuato	767	Toluca	720
Hermosillo	915	Tuxtla Gutiérrez	1390
Jalapa	1258	Villahermosa	1410
La Paz	612	Zacatecas	252

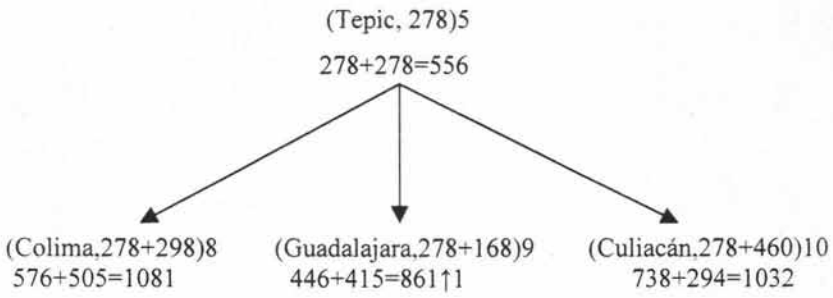
Aplicando el algoritmo de "menor coste" o A*, se tendría que:



Durango comienza con un valor de 0, pues es el punto de partida, desde ahí se puede ir a cinco lugares de acuerdo con el mapa de carreteras disponibles. Para Culiacán hay 324 km, para Chihuahua 637 km, para Saltillo 395 km, para Zacatecas 252 km y para Tepic 278 km. A cada nodo generado se le coloca a la derecha un número para indicar cuando fue generado, y en la parte de abajo se pone la función de evaluación heurística con sus respectivos valores (distancia recorrida + distancia en línea recta de Durango a esa Capital). Se opta por desarrollar los nodos hijos (descendientes) de la función heurística de menor valor, en este caso corresponde a Zacatecas, por lo que tenemos:

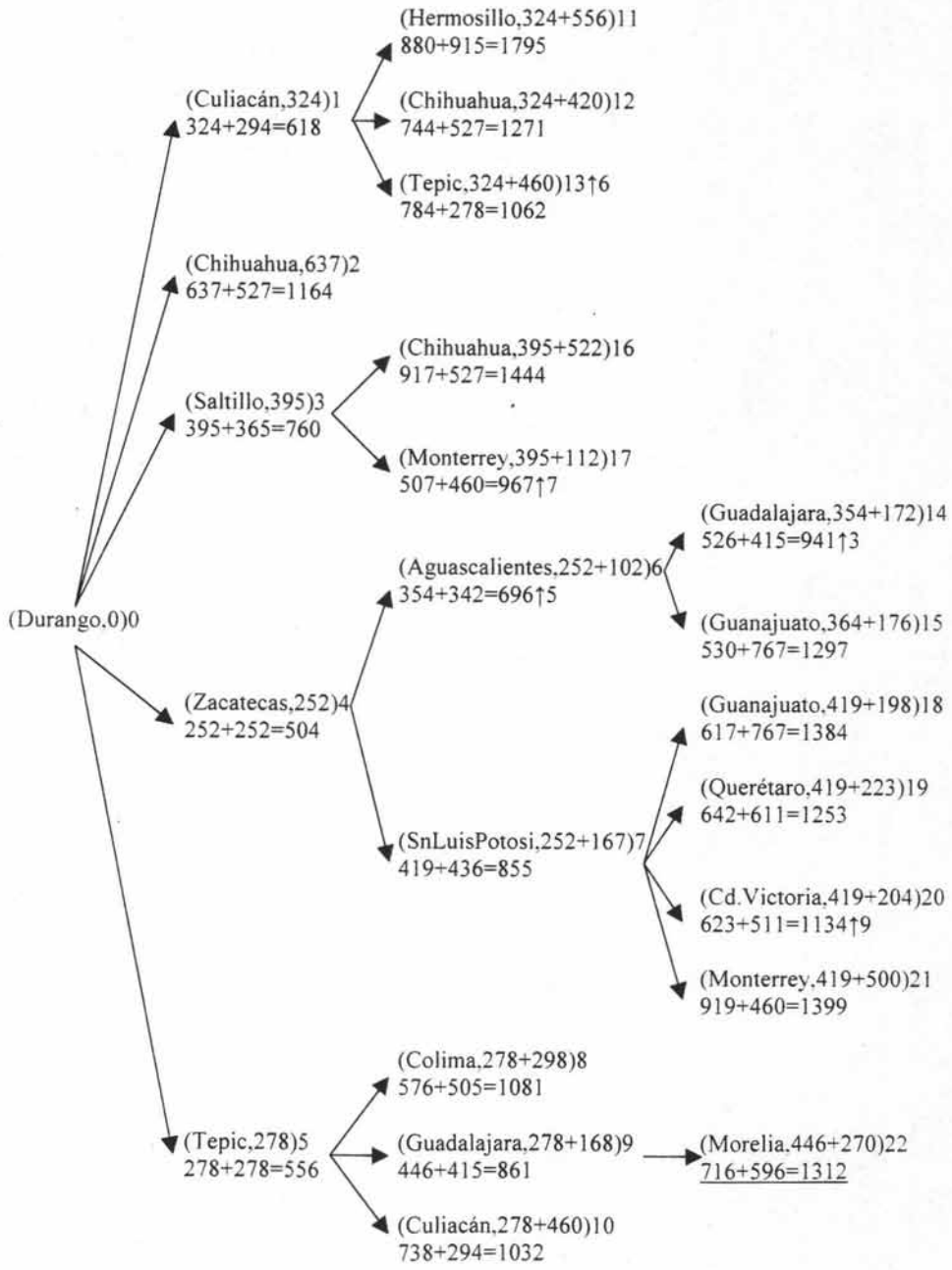


Se verifica el valor de la función heurística de los nodos hijos de Zacatecas, se toma el de menor valor y si éste es mayor que la de otros nodos generados con anterioridad, entonces se pasa a dicho nodo (se indica con una flecha hacia arriba a que nodo debe saltarse) y se desarrollan sus descendientes. En nuestro ejemplo, Aguascalientes tiene una función heurística menor que San Luis Potosí, pero su valor de 696 es mayor que el de Tepic con sólo 556, por eso se pasa a desarrollar los descendientes de Tepic quedando:



La función heurística de menor valor es la de Guadalajara, pero su valor es superior a la de otros nodos generados con anterioridad. Saltando al nodo de menor costo llegamos al número 1 que corresponde a Culiacán. Al generar sus descendientes tenemos tres: Hermosillo, Chihuahua y Tepic, con una función heurística de 1795, 1271 y 1062, respectivamente.

En seguida se presenta el recorrido completo de Durango a Morelia, por razones de espacio los descendientes se generan hacia la derecha.



Por lo tanto, el camino más corto es: Durango—Tepic—Guadalajara—Morelia.

CAPÍTULO 4

TECNICAS DE REPRESENTACIÓN Y MANIPULACIÓN DEL CONOCIMIENTO QUE ACTUALMENTE SON DE MAYOR INVESTIGACIÓN EN LA I. A.

La ciencia es sólo un ideal; la de hoy corrige
la de ayer y la de mañana corregirá la de hoy.
Ortega y Gasset.

4.1 SISTEMAS INSPIRADOS EN PROCESOS DE LA NATURALEZA

La naturaleza ha sido siempre fuente de inspiración para el hombre. Comprender los complejos mecanismos por los cuales la naturaleza atraviesa ha sido una de las tareas más ambiciosas del hombre. Y aunque faltan muchas cosas por descubrir, el conocimiento logrado hasta la fecha ha permitido la creación de algoritmos que forman parte del área de los Sistemas Inspirados en la Naturaleza. Es decir, se trata de métodos que simulan procesos naturales y que se aplican a soluciones de problemas reales de diversa índole. Estos métodos fueron desarrollados desde hace años, sin embargo, son actualmente los de mayor estudio para los investigadores de la Inteligencia Artificial, aquí sólo trataremos algunos de los más importantes.

Dos de los paradigmas principales de la computación evolutiva se desarrollaron en ambos lados del océano Atlántico a mediados de la década de los 60's: las Estrategias Evolutivas (EEs) en Alemania, y los Algoritmos Genéticos (AGs) en Estados Unidos.

Las EEs y los AGs son métodos generalizado de búsqueda, diseño y optimización que simulan el proceso natural de evolución. La forma de operar de estos algoritmos no es dependiente del dominio del problema, sino que el objetivo es aplicar la "Teoría de la Evolución Natural" que el inglés Charles Darwin presentó el 1 de julio de 1858 ante la *Sociedad Linnean* de Londres. El paradigma darwinista de la evolución sostiene que "el mecanismo evolutivo de las especies e individuos está sustentado en cuatro procesos (también llamados operadores genéticos) principales: reproducción, mutación, competencia y selección".[41] Los anteriores operadores genéticos frecuentemente se resumen en la frase: sobrevivencia del más apto y fuerte de los individuos de una especie, es decir, evolución.

[41] Fraser, A. *Simulation of genetic system*, Journal of Theoretical Biology, E.U.,1962, p. 329,

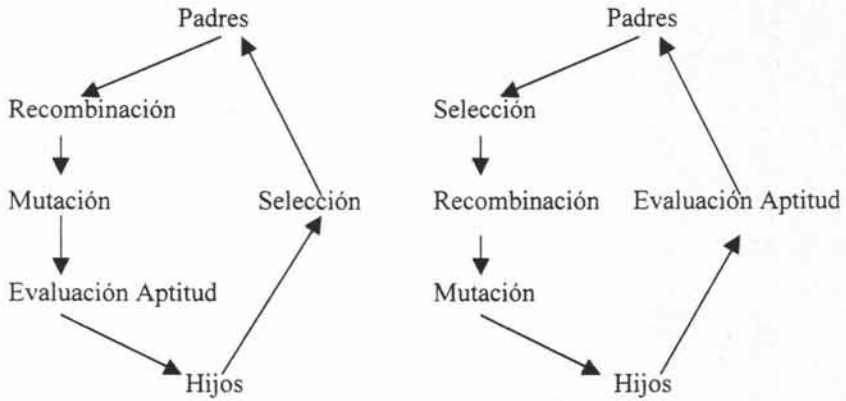
Su característica principal de los sistemas evolutivos es "una población (#####) que representa un conjunto de soluciones potenciales. El tamaño de la población puede variar a lo largo de varias generaciones, pero usualmente permanece sin cambios."[42] Los componentes que integran la población son denominados organismos o individuos. "La estructura de los individuos es determinada a priori y es la misma para toda la población. Sin embargo, la estructura precisa de los individuos es dependiente del dominio del problema"[43], lo cual implica que para cada problema tiene que idearse una representación adecuada.

Cada problema debe integrar una medida comparativa de las soluciones que compiten. Es decir, debe existir un mecanismo derivado del dominio del problema que nos permita asignar un valor escalar (o equivalente) a cada individuo de la población que sea representativo de su calidad como solución. Este valor se denomina aptitud. Un individuo de mayor aptitud representa una mejor solución a un problema, que en las condiciones específicas de éste pueden representar un solución correcta o inclusive óptima. Un individuo con menor aptitud representa por lo tanto una peor solución.

Mientras que los AGs emplean para evolucionar una representación de genotipos, las EEs lo hacen a través de fenotipos. Es por esta diferencia, que los individuos en las EEs son organismos constituidos directamente a partir de las variables que se busca optimizar. Por consiguiente, los operadores genéticos no solo se usan en forma diferente, sino que la importancia de cada uno de ellos es distinta a la de los AGs. Así, en los AGs, la búsqueda progresa por medio de la recombinación del material genético de los individuos de mayor aptitud, y en las EEs la búsqueda progresa por medio de la mutación de los individuos más aptos. Otra diferencia es que en los AGs la selección es aleatoria, mientras que en las EEs es determinística. El esquema 4.1 muestra el orden de los operadores genéticos para los AGs y las EEs.

---

[42] Arturo Hernández, *Estrategias evolutivas*, Soluciones avanzadas, 1998, p. 40.
[43] *Ibidem*, p. 40



a) Estrategia Evolutiva b) Algoritmos Genético

Fig.4.1, Esquema de Operadores Genéticos

Los AGs gozan en la actualidad de enorme popularidad, probablemente debido a su generalidad y sencillez conceptual, que les proporciona un enorme facilidad de uso y aplicabilidad a una gran variedad de problemas. Por su parte, las EEs son una metodología específicamente diseñada para optimización de funciones reales, lo cual tal vez disminuye su área de aplicación pero no así su efectividad. A continuación se describen las características básicas de las EEs y AGs.

4.1.1 Estrategia Evolutiva (EEs).

La versión más simple de la EEs es la de dos miembros, y se denota mediante la expresión: EE(1+1). Se denomina así porque únicamente trabaja con dos individuos: un padre que mediante mutación produce un solo hijo. Ambos se someten a un proceso de selección donde el más apto (el de mayor aptitud) pasa (sobrevive) para la siguiente generación y el menos apto se desecha.

Las mejores mutaciones son favorecidas mediante una regla que determinísticamente adapta la desviación estándar, esta regla se llama: éxito de 1/5. Y establece que "la

relación de las mutaciones exitosas contra el total de las mutaciones debe ser de 1/5".[44] Entonces, si la relación es menor se reduce la desviación estándar, y viceversa. Es decir, si la mutación es exitosa, la búsqueda debe continuar con cambios aleatorios grandes; en caso contrario, los cambios deben ser más pequeños. Por ejemplo, en la búsqueda del óptimo de una función cualquiera f(x,y) con variables independientes cuyos dos individuos 'x1' e 'y1' tienen la estructura siguiente:

$$(x1, y1, \Delta x1, \Delta y1)$$

Tanto $\Delta x1$ y $\Delta y1$ son calculados mediante una variable de control correspondiente a cada uno de ellos, estas variables de control establecen un rango de valores a priori. Establezcamos dichos valores como '$\sigma 1$' y '$\sigma 2$' cuyo valores están comprendidos entre 0 y 1. Una mutación consiste en generar un número aleatorio con determinada desviación, así la mutación de '$\Delta x1$' es calculada con desviación-estándar-x1, y por su parte la mutación '$\Delta y1$' es calculada con desviación- estándar-y1, entonces el nuevo hijo es simplemente:

$$(x1 + \Delta x1, y1 + \Delta y1)$$

Supongamos que el punto de partida es A = <1.2, 1.5> donde x1=1.2 e y2=1.5, y que los valores de la desviación estándar comprendidos entre 0 y 1 son <-0.4, 0.9> respectivamente, entonces el valor resultante es B = <0.8, 2.4>. Ahora al comparar B con A, notamos que el hijo B tiene menor aptitud que el padre A, entonces es desechado y se realiza una nueva mutación sobre A (de manera análoga a la anterior), y si el hijo producido es mejor por ser más apto que A se convierte en el padre de la siguiente generación.

Cuando la EE(1+1) se usó en problemas prácticos de optimización surgieron dos problemas importantes: presentó una velocidad lenta de convergencia hacia el óptimo y

[44] *Ibidem*, p. 44

una notable incapacidad de salir de los mínimos locales. Esto se debe a que al entrar a un mínimo local no se detectará mejoría alguna durante un buen número de iteraciones y en consecuencia la «regla del 1/5» disminuirá la desviación estándar, dificultando aún más abandonar el mínimo local, pero esto no quiere decir que el óptimo global (respuesta satisfactoria) no sea encontrado, sino simplemente que necesita un considerable tiempo de respuesta. Pues para salir de un mínimo local es necesario esperar a que aleatoriamente se produzca una secuencia de mutaciones, tales que, la suma de todos estos pequeños incrementos coloquen al último hijo de esta secuencia en la orilla de la ondulación correspondiente, a fin de sacarlo del mínimo local en que se encuentra atrapado.

4.1.2 Algoritmo Genético (AGs).

En el ámbito de la computación, un AG "es un algoritmo matemático que transforma un conjunto de «objetos matemáticos» individuales con respecto al tiempo, usando operaciones modeladas de acuerdo al principio de selección natural de Darwin, tras haberse presentado de forma natural una serie de operaciones genéticas de las que destaca la recombinación de géneros".[45] Estos «objetos matemáticos» suelen ser representados genotípicamente, es decir, como cadenas de caracteres de longitud fija que se ajusta al modelo de las cadenas de cromosomas y se les asocia con una cierta función matemática que refleja su actitud. Así, cada posición de la cadena es un aleolo (valor de un gene).

Uno de los objetivos principales al utilizar un AG es hacer que la máquina tenga la capacidad de solucionar un problema sin tener que decirle específicamente cómo debe proceder para encontrar la respuesta. No obstante, para optar por el uso de un AG, es conveniente tomar en consideración las siguientes especificaciones:

1. El espacio de búsqueda necesariamente debe estar delimitado dentro de un cierto rango.

[45] Leon, Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, E.U. 1991, p. 158

2. Se debe poder definir una función de aptitud que sirva para indicar qué tan buena es una respuesta.
3. Las soluciones, y en general el problema a tratar, debe codificarse de forma que resulte fácil la implementación por computadora.

Las operaciones que efectúa un AG son la selección, reproducción, mutación y cruza. Para ello primeramente se crean diversos nodos-estados[46] aleatoriamente, que se encuentran en el dominio de un problema (generándose un espacio de búsqueda diverso y convergente llamado población). A continuación se plantea un caso particular y se someten a prueba los diversos nodos-estados creados, cuando estos dan la solución se clasifican según el grado de exactitud de la solución presentada. Los nodos-estados con mayor exactitud (más prometedores) se seleccionan para hacer una cruza con ellos y así lograr la reproducción de otros nodos-estados más evolucionados (en sentido de que se van acercando a soluciones satisfactorias, incluso llegar a dar una solución definitiva). Los descendientes pasarán a formar parte de la población de la siguiente generación, que estará constituida por los individuos más aptos y más adaptables a los cambios que se producen en su entorno.

Una vez realizada la selección y reproducción, los nodos-estados que no fueron seleccionados se eliminan junto con los padres, quedando sólo la población de los hijos. El pseudo-código correspondiente a un AG es:

```
Generar una población inicial: G(0)
Evaluar G(0)
T = 0;
Repetir:
        T = T+1;
        Reproducir  G(T) usando:
                Cruza ó
                Mutación
        Evaluar G(T);
Hasta encontrar una solución.
```

[46] También pueden ser «Programas» en vez de «Nodos-Estados», si el dominio donde se trabaja así lo requiere.

La reproducción se hace generalmente con uno o dos puntos de cruza, sobre dos individuos seleccionados denominados padres. Por regla general se tiene que en toda cruza necesariamente se dan dos hijos en donde uno de ellos hereda los primeros bits de uno de los padres (P0) y los últimos del otro (P1), y el otro los primeros del padre P(1) y los últimos del padre (P0), tomando como frontera de conversión los puntos de cruza. Por ejemplo:

Punto de cruza: 3° lugar
contando desde la derecha

Padres:

P(0)	P(1)
10101101	11011011

Hijos:

11011101
10101011

Un punto único de cruza

Puntos de cruza: 3° y 7° lugar
Contando desde la derecha

Padres:

P(0)	P(1)
10101101	11011011

Hijos:

10011101
11101011

Dos puntos de cruza

Para mutar un estado nodo, también se toma uno ó máximo dos puntos de anclaje de un único nodo-estado y se cambia su valor. Ejemplo:

Punto de anclaje: 4° lugar
contando desde la derecha

Nodo

111011011

Mutante:

111010011

Un punto de anclaje

Puntos de anclaje: 2° y 5° lugar
contando desde la derecha

Nodo:

101100111

Mutante:

101110101

Dos puntos de anclaje

Entre más a la izquierda se encuentren los puntos de cruza o anclaje, mayor es su porcentaje de afectación (es más significativo). Para ilustrar supóngase el siguiente nodo:

(1001011101)

Cuando su anclaje es en el 1er lugar su afectación va de 0-10%; si es el en 5º lugar afecta un 50-75%; y si va en el último lugar su porcentaje correspondiente es de 100%, recordando que los lugares se cuenta de derecha a izquierda.

Para ilustrar el comportamiento de un AG en un problema práctico, supongamos que se desea encontrar el máximo de la función $F(x) = X^2$ sobre los enteros (1,2,...,31). Es decir, se quiere encontrar el valor de 'x' que hace que la función F(x) alcance su valor máximo, pero restringiendo a la variable 'x' a tomar valores enteros comprendidos entre 0 y 31. Obviamente el máximo se tiene para x = 31, donde F vale 961. Evidentemente para lograr dicho óptimo, bastaría actuar por búsqueda exhaustiva, dada la baja cardinalidad del espacio de búsqueda. Sin embargo, nuestro ejemplo es puramente ilustrativo.

De acuerdo al pseu-código del AG descrito con anterioridad, vemos que el primer paso a efectuar consiste en determinar el tamaño de la población inicial, para a continuación obtener dicha población al azar y computar la función de evaluación de cada uno de sus individuos. Pero para hacer esto, previamente se debió de tener una manera de representar y codificar las posibles soluciones (posible valores de x); la forma de hacerlo que se seleccionó es mediante codificación binaria. Con esta codificación un posible valor de x es:

$$(0,1,0,1,1) = 11$$

Recordando que en el sistema de numeración binaria existe un equivalente decimal entero para cada número. Por lo cual tenemos que: (0,1,0,1,1) = 11, y se obtuvo multiplicando la última componente (un 1) por 1, la penúltima (un 1) por 2, la anterior (un 0) por 4, la segunda (un 1) por 8 y la primera(un 0) por 16, y al final hacer la sumatoria del total de cada producto:

$$(0*16) + (1*8) + (0*4) + (1*2) + (1*1) = 0+8+0+2+1 = 11$$

Se tomó una representación de números binarios con una extensión de 5 localidades, porque son los necesarios para poder representar el máximo valor de 'x', que es de 31, así:

(1,1,1,1,1) equivale a X = 31

A cada posible valor de la variable X en representación binaria se le llama individuo. Una colección de individuos constituye lo que se denomina población y el número de individuos que la componen es el tamaño de la población.

Ahora se genera de manera aleatoria una población con un determinado número de individuos integrantes, en este caso digamos que se compone de 6. Esta primera población recibe el nombre de «Generación cero», y es la población inicial.

(0,1,1,0,0)
(1,0,0,1,0)
(0,1,1,1,1)
(1,1,0,0,0)
(1,1,0,1,0)
(0,0,0,0,1)

Generación cero, G(0)

Nuestro siguiente paso es hacer competir a los individuos entre sí. Este proceso se conoce como selección. La tabla siguiente resume el proceso.

Tabla G(0) – SELECCIÓN				
No. Asignado Al individuo	**Población Inicial Genotipos**	**Valor del genotipo**	**Valor de la función de adaptación**	**Probabilidad de selección**
1	(0,1,1,0,0)	12	144	6
2	(1,0,0,1,0)	18	324	3
3	(0,1,1,1,1)	15	225	2
4	(1,1,0,0,0)	24	576	5
5	(1,1,0,1,0)	26	676	4
6	(0,0,0,0,1)	1	1	1

Cada fila en la tabla de selección está asociada a un individuo de la población inicial. El significado de cada columna de la tabla es el siguiente: en la primera columna tenemos el número que le asignamos al individuo, en la columna siguiente al individuo en codificación binaria, la tercera columna muestra el valor de 'X', la cuarta columna corresponde al valor de F(x), y la última columna muestra al individuo rival.

La rivalidad es una manera de realizar el proceso de selección mediante un torneo entre dos. A cada individuo de la población se le asigna una pareja y entre ellos se establece un torneo: el mejor genera dos copias y el peor se desecha. La columna cinco indica la pareja asignada a cada individuo, lo cual se ha realizado aletoriamente. Existen muchas variantes de este proceso de selección, aunque este método nos vale para ilustrar el ejemplo.

Efectuado los torneos tenemos que:

- En la competencia entre el individuo 1 y 6 de la población inicial, el primero de ellos ha recibido dos copias, mientras que el segundo cae en el olvido.
- En la competencia entre el individuo 2 y 3 de la población inicial, el primero de ellos ha recibido dos copias, mientras que el segundo cae en el olvido.
- En la competencia entre el individuo 4 y 5 de la población inicial, el segundo de ellos ha recibido dos copias, mientras que el primero cae en el olvido.

Así, la población que tenemos es la mostrada a continuación:

No. Asignado al individuo	**Población tras la Selección**
1	(0,1,1,0,0)
2	(0,1,1,0,0)
3	(1,0,0,1,0)
4	(1,0,0,1,0)
5	(1,1,0,1,0)
6	(1,1,0,1,0)

Tras realizar la selección, se realiza el cruce. Una manera de hacerlo es mediante el cruce 1X: se forman parejas entre los individuos aleatoriamente de forma similar a la selección, como se muestra a continuación:

Tabla – CRUCE			
No. de individuo	**Población**	**Su pareja seleccionada**	**El punto de cruce**
1	(0,1,1,0,0)	5	1
2	(0,1,1,0,0)	3	3
3	(1,0,0,1,0)	2	3
4	(1,0,0,1,0)	6	1
5	(1,1,0,1,0)	1	1
6	(1,1,0,1,0)	4	1

Dados dos individuos pareja se establece un punto de cruce aleatorio, que no es más que un número aleatorio entre 1 y 4 (la longitud del individuo menos 1). Por ejemplo, en la pareja 2-3 el punto de cruce es 3, lo que significa que un hijo de la pareja conserva los tres primeros bits del padre y hereda los dos últimos de la madre, mientras que el otro hijo de la pareja conserva los tres primeros bits de la madre y hereda los dos últimos del padre. La población resultante se muestra en la columna dos de la tabla de población tras el cruce.

Tabla G(1) – POBLACIÓN TRAS EL CRUCE			
No. Asignado al individuo	**Población Inicial +1 Genotipos**	**Valor del genotipo**	**Valor de la función de adaptación**
1	(0,1,0,1,0)	10	100
2	(1,1,1,0,0)	28	784
3	(0,1,1,1,0)	14	196
4	(1,0,0,0,0)	16	256
5	(1,1,0,1,0)	26	676
6	(1,0,0,1,0)	18	324

En la columna tres se tiene el valor de 'x'; en la siguiente el valor de 'F(x)'. Ahora el valor máximo de 'F(x)' es 784 (para el individuo 2), mientras que antes de la selección y el cruce era de 676 del individuo 5. Por lo tanto, esto indica que los individuos después de la selección y el cruce son mejores que antes de estas transformaciones.

El siguiente paso es volver a realizar la selección y el cruce tomando como población inicial G*1). Esta manera de proceder se repite tantas veces como número de iteraciones se fijen, tras cada una de ellas se puede ir obteniendo valores óptimos, por ello es recomendable ir guardando la mejor solución de todas las iteraciones anteriores y al final quedarte con la mejor solución de las exploradas

A pesar de que los AG son eficientes, las funciones de algunos problemas complejos requieren de un tiempo de procesamiento considerable. En tales casos los AG emplean mucho tiempo en alcanzar una solución satisfactoria. Por tal motivo se han desarrollado AG que se ejecutan en computadoras con arquitectura paralelas, permitiendo reducir el tiempo de respuesta. Así, llegamos a tener AG-Paralelos con una valor bajo, en vez de uno alto. A tal valor se le denomina como «granularidad», y la ecuación que nos permite determinar su nivel es:

$$Granularidad = \frac{TiempoDeprocesoIndividual}{TiempoDeprocesoPoblaciones}$$

Por lo tanto, la «granularidad» se define como la razón entre el tiempo dedicado por el algoritmo al procesamiento de los nodos-estados (individuos) y el tiempo dedicado a la comunicación entre las subpoblaciones. Si esta relación es alta se dice que el procesamiento es de grado burdo, de lo contrario es de grado fino.

Mientras que en un AG básico las generaciones se van dando una a una (es un proceso serial), en el AG-Paralelo se van dando generaciones simultáneamente. Por lo cual, este último emplea el intercambio entre los elementos de cada generación (población). Así, en un AG-Paralelo tenemos gran cantidad de emigrantes, es decir, un estado-nodo que se produjo como miembro de una generación aleatoria "G(0)" se copia tal como está a otro generación aleatoria "G(1)", y es tratado por "G(1)" de la misma forma que los individuos locales.

El intercambio de individuos entre poblaciones se conoce como migración. Y su frecuencia se determina introduciendo un valor denominado «intervalo de emigración» que se define como el número de generaciones que ocurren entre emigraciones. Por lo cual, si se tienen seis generaciones:

$$G(0), G(1),G(2), G(3),G(4), G(5)..... G(n)$$

Y de la generación G(4) pasan a la generación G(0) un número de individuos, se dice que se tiene un intervalo de emigración = 3, ya que son tres generaciones las que hay entre G(0) y G(4). Por su parte, el número de individuos que emigran cada vez nos da la «tasa de emigraciones», que se define como la proporción de una población que se va o llega en un momento dado. Pero los nodos-estados (individuos) no pueden migrar a cualquier de la generaciones producidas al mismo tiempo, sino que solamente pueden pasar a aquellas generaciones con las cuales su propia generación (población) esta comunicada. Esto es, las generaciones aleatorias están comunicadas entre sí mediante una topología que especifica su estructura de enlaces, entre las más comunes tenemos: anillo, malla, estrella y la totalmente conectada.[47] Por lo tanto, los emigrantes se apegan a las topologías descrita. Para ilustrar supongamos una malla de grado 2:

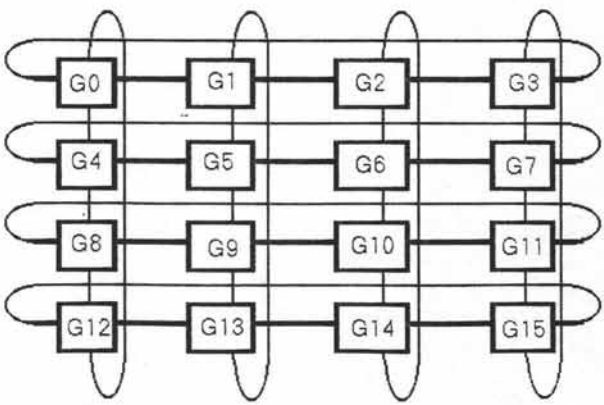


Figura 4.2, Topología Malla de grado 2

[47] Nótese la semejanza con las topologías de redes de computadoras.

Es de grado 2 porque son sólo dos las generaciones como máximo que hay entre la emigración de un nodo-estado (individuo) de una generación a otra. Un nodo sólo puede emigrar a una generación con la que esté conectado directamente, por lo que un nodo de G(10) únicamente podrá pasar a G(8), G(9), G(11), G(2), G(6) o G(14). Sin embargo, no sólo la conexión es lo que no permite emigrar a un nodo-estado a otra generación, sino también hay un criterio de selección de emigrantes que dice quienes pueden viajar. Es muy simple, el criterio es que emigran aquellos nodos-estados de mayor aptitud (los que se acercan más a una solución satisfactoria).

Las aplicaciones más comunes de un AG han sido soluciones a problemas de optimización, por ejemplo, un "Sistema Adaptable Clasificador", que es un modelo de aprendizaje automático que emplean un AG para evolucionar conjuntos de reglas coadaptadas. En un sistema adaptable clasificador existen mecanismos automáticos para ajustar los pesos de las reglas en base a su propia capacidad para obtener recompensa de un medio ambiente; de esta forma, los sistemas adaptables clasificadores intentan emular la forma en que aprenden pequeños animales. Por ser capaces de realizar aprendizaje basados únicamente en una señal de recompensa, los sistemas adaptables clasificadores tiene una gran aplicación potencial.

4.2 ENFRIAMIENTO SIMULADO

El enfriamiento simulado es una variante de la escalada, existen diferentes versiones de este proceso; en el más difundido se van seleccionando los movimientos de acuerdo con una distribución de probabilidad definida sobre el conjunto de posibles movimientos. La distribución de probabilidad tiende a favorecer los movimientos hacia nodos situados a menor altura. Es decir, utiliza una técnica de optimización no determinista, en la que el proceso comienza con una distribución de probabilidades que apenas favorece los movimientos hacia los nodos más bajos (movimientos descendentes).

Al principio del proceso estaremos moviéndonos de forma aleatoria por el conjunto total de los nodos. Sin embargo, a medida que el proceso va avanzando, empezaremos a descender hacia los niveles inferiores. Si el camino recorrido no es muy profundo, probablemente tampoco será muy ancho, con lo que pronto ejecutaremos un movimiento que nos hará salir de él. En cambio, es menos probable que nos salgamos de un camino más ancho, con lo que al final del proceso estaremos situados en el punto-nivel más bajo. De esta manera, la idea consiste en realizar una exploración lo suficientemente amplia al principio, ya que la solución final es relativamente insensible con el estado inicial. Así se disminuye la posibilidad de caer en un máximo local o una meseta[48].

Visto como un proceso computacional, el enfriamiento simulado se basa en el proceso físico de la aleación en la que ciertas sustancias físicas, como los metales se funden (es decir, incrementar sus niveles de energía) para luego sufrir un proceso gradual de enfriamiento hacia un estado sólido. El objetivo de este proceso es alcanzar un estado final de mínima energía (las sustancias físicas normalmente evolucionan hacía configuraciones de más baja energía de forma que el descenso ocurre de forma natural). Sin embargo, existe cierta probabilidad de que se produzcan transacciones hacia estados con energías más altas, esta probabilidad viene dada por:

$$P = e^{-\Delta E / KT}$$

Donde: ΔE es el cambio positivo en el nivel de energía, T es la temperatura, y K es la constante Vol.

El enfriamiento simulado incluye un criterio para que ocasionalmente la estrategia de optimización acepte configuraciones de alto cambio (alto costo) entre estados del sistema. Su funcionamiento se basa en una función de costo (PC_k) y una solución inicial (X_0), la estrategia iteractiva de búsqueda trata de mejorar la solución actual perturbando (X_0)

[48] Recordando que un «Máximo local» es un Nodo mejor que todos los nodos de su mismo nivel, pero no es mejor que los Nodos de otros niveles del mismo espacio de búsqueda. Y una «Meseta» es un área donde un conjunto de estados vecinos posee el mismo valor.

aleatoriamente por medio de L(k) iteraciones. Es decir, la simulación del proceso de enfriamiento se utiliza para describir un proceso de generación de sucesión de soluciones de un problema de optimización combinatoria, en donde se vayan obteniendo, conforme el proceso avanza, mejores soluciones al mismo. Para este propósito, se observa una analogía entre el sistema físico y un problema de optimización combinatoria en donde cada solución del problema puede verse como un estado del sólido, y el valor de la función objetivo como el nivel de energía del sólido. En resumen, se puede pensar en las siguientes equivalencias.

Las soluciones de un problema de optimización combinatoria	→ son equivalentes a →	los estados de un sistema físico
El costo de una solución	→ es equivalente a →	la energía de un estado.

Necesariamente se introduce un parámetro que juega el papel equivalente de la temperatura (T), llamada «parámetro de control». El algoritmo de enfriamiento simulado tras cada iteración, va evaluando en valores decrecientes del parámetro de control, soluciones más optimas. Por ejemplo, sea (S, *f*) una instancia de un problema de optimización combinatoria, y teniendo denotadas por '*i*' y '*j*' dos soluciones con costo *f(i)* y *f(j)*, respectivamente. Entonces el criterio de aceptación determina si '*j*' se acepta de '*i*' a partir de aplicar la siguiente probabilidad de aceptación:

$$\text{PC}\{\text{aceptar}\} = \begin{cases} 1 & sif(j) \le f(i) \\ \exp\left(\dfrac{f(i)-f(j)}{c}\right) & sif(j) > f(i) \end{cases}$$

Inicialmente casi todos los movimientos se aceptan, explorándose aleatoriamente sobre todo el espacio de soluciones. Progresivamente (PC) va disminuyendo y la probabilidad de aceptar nuevas soluciones será menor. Al final del algoritmo, sólo los movimientos que mejoran la función se aceptarán.

En resumen, se parte de una solución inicial 'X_0 = inicial' y de una cierta temperatura o valor del parámetro de control 'PC_k', y en cada etapa 'n' se realizan 'L(k)' iteraciones. En cada iteración se genera una solución 'X_{n+1}' cercana a la solución actual 'X_o= inicial', y que se acepta como nueva solución actual siempre si es mejor, o con una probabilidad de '$e^{-D/T(n)}$' si es peor que la actual. Tras la última iteración, se disminuye la temperatura PC_k. El algoritmo de enfriamiento simulado en su forma más simple es:

```
Comienza
        INICIALIZA (X0 = inicial, PC0, L0)
        k: = 0
        i: = X0
                Repite
                para L : = 1 a Lk haz
                        Comienza
                        GENERA ( J de Si )
                        Si   f(j) ≤ f(i)
                                entonces  i := j
                        sino
                             si exp[(f(i)-f(j))/PCk)]> NúmeroAleatorio en (0,1)
                                entonces  i := j
                fin para
                k := k+1
                CalculaLongitud (Lk)
                CalculaControl (PCk)
        hasta criterio de paro
fin
```

Sea PC_k el valor del parámetro de control y L_k el número de transiciones generadas en la k-ésima iteración. El algoritmo comienza llamando a un procedimiento de inicialización donde se definen la solución inicial, la temperatura inicial y el número inicial de generaciones necesarias para alcanzar el equilibrio térmico para la temperatura inicial. La parte modular del algoritmo consta de dos ciclos. El externo «Repite ... hasta», y el interno «para ... fin para». El ciclo interno mantiene fijo el parámetro de control hasta que se generan L_k soluciones y se acepta o se rechaza la solución generada conforme los criterios de aceptación ya discutidos. El ciclo externo disminuye el valor de la temperatura mediante el procedimiento CalculaControl y calcula el número de soluciones a generar

para alcanzar el equilibrio térmico mediante el procedimiento CalculaLongitud. Este ciclo finaliza cuando la condición de paro se cumple.

Los parámetros a establecer en el algoritmo son: temperatura inicial, longitud de la temperatura, grado de enfriamiento y temperatura final. La elección de estos parámetros junto con otras mejoras que se pueden introducir conduce a la adaptación del algoritmo de enfriamiento simulado a las características del problema a resolver. Sin embargo, el enfriamiento simulado ha probado ser una herramienta poderosa para resolución de problemas no lineales y/o combinatorios grandes.

4.3 LÓGICA DIFUSA (MULTIVALUADA O POLIVALENTE)

Actualmente existen lógicas alternativas a la lógica de enunciado y de predicados que ofrecen algunas ventajas considerables en la solución y tratamiento de ciertos problemas, como son los sistemas lógicos multivaluados, que permiten ciertos formalismos (representación) del conocimiento imposibles para otras lógicas. Por tal motivo se incluye una breve exposición de la llamada lógica borrosa, un sistema de lógica multivaluada que permite ciertas aplicaciones muy útiles en la Inteligencia Artificial, y en el control de sistemas complejos.

La lógica difusa, también llamada lógica borrosa, es una generalización de la lógica moderna[49] de dos valores. Mediante lógica difusa es posible manipular matemáticamente conceptos definidos con incertidumbre o vagamente. Es decir, la lógica borrosa incorpora a la lógica moderna el concepto de que: "todo es cuestión de grado", y es precisamente este concepto en el que se funda.

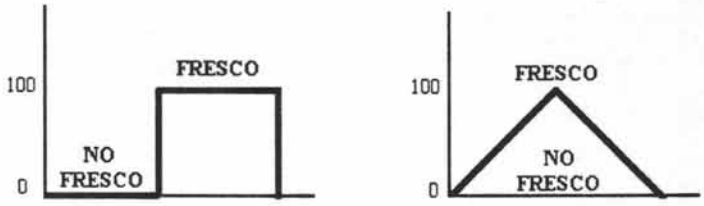
El estudio de la lógica borrosa comenzó en los años veinte del siglo pasado con los trabajos realizados por el polaco Jan Lukasiewicz y Emil Post. El primero propuso una

[49] Por "lógica moderna" nos referimos a la lógica de enunciado, a la de predicados, y en cierta forma a la lógica binaria.

lógica de tres valores; el segundo, una de un número finito cualquiera, *n valores*. Más tarde, Lukasiewicz y Alfred Tarski desarrollaron lógicas polivalentes con un número infinito de valores, tales lógicas integran una serie de enunciados que pueden tener valores de verdad fraccionarios, o más de dos valores de los usados en la lógica binaria. Sin embargo, fue hasta casi 30 años más tarde cuando Lofti A. Zadeh, entonces director del departamento de ingeniería eléctrica de la Universidad de California en Berkeley, publicó "Fuzzy Sats" (conjuntos borrosos). Artículo que proporcionó a la disciplina su nombre, y en el cual Zadeh desarrollo una álgebra completa para conjuntos borrosos aplicando la lógica de Lukasiewicz a todo objeto de un conjunto. No obstante, los conjuntos borrosos no tuvieron aplicación práctica hasta mediados de los años setenta, cundo Ebrahuim H. Mämdani, del Queen Mary College de Londres, diseña un controlador borroso para un motor de vapor.

La diferencia entre la lógica bivalente y la lógica borrosa se halla en lo que Aristóteles llamó «ley del tercero excluso» que dice: "una cosa es o no es, no cabe un término medio. Así, entre dos proposiciones contradictorias no hay termino medio: A es B o A no es B."[50] Extendiendo la «ley del tercero excluso» a la teoría de conjuntos habitual, tenemos que un objeto cualquiera pertenece a un conjunto o bien no pertenece a él. No hay término medio: el número 1 pertenece por completo al conjunto de los números impares y no pertenece en absoluto al conjunto de los números pares. Así, ningún objeto puede pertenecer al mismo tiempo a un conjunto y a su complemento, o no pertenecer a ninguno de los dos. Pero la lógica borrosa viola en alguna medida la ley del tercero excluido. Por lo que los conjuntos se definen mediante funciones de pertenencia, o membresía, que representan la forma en que los humanos manejan incertidumbre. A un conjunto borroso sólo se pertenece en parte. Los bordes de los conjuntos normales son abruptos y sus condiciones de pertenencia se presentan gráficamente mediante un escalón como (a) de la figura 4.3. Por su parte, la frontera de un conjunto borroso se va difuminando: la condición de pertenencia es una curva o un monte, que se representa gráficamente como se muestra en (b) de la figura 4.3.

[50] Misael, Mateos. Lógica para inexpertos, Edere, México, 2001, p. 33



(a) Bivaluada (b) Borroso

Figura 4.3, Graficas de pertenencia a conjuntos

En un conjunto bivaluado, el aire es fresco o no lo es, sólo se puede permanecer a uno de los dos estados. En cambio, los conjuntos borrosos permiten contradicciones parciales, pues el aire puede ser fresco en un 20%, y no fresco en un 80%. No obstante, es importante señalar que no es lo mismo un grado borroso que un porcentaje de probabilidad, ya que las probabilidades señalan en qué medida cabe esperar que suceda o no algo concreto. En cambio, la borrosidad mide el grado de *factum* (algo que esta sucediendo ya). La única restricción impuesta a la lógica borrosa es que los grados de pertenencia de un objeto o conjunto complementario deben sumar 1. De tal manera que si el aire fresco es un 20%, también deberá parecer no fresco en un 80%, sorteando con ello la contradicción bivaluada que implicaría la lógica formal bivaluada: que algo es 100% fresco y 0% no fresco, o 0% fresco y 100% no fresco. La «ley del tercero excluso» vale tan sólo como casos especiales en la lógica borrosa: es cuando un objeto pertenece al 100% a un grupo.

Las reglas de un sistema borroso definen un conjunto de zonas que se solapan y relacionan un dominio completo de entradas con un dominio completo de salidas. En este sentido, el sistema borroso es una aproximación de alguna función o ecuación matemática de causa y efecto, una norma que determinase, por ejemplo, cómo debe ajustar un microprocesador la potencia de un acondicionador de aire al interior de un avión o la velocidad de un motor en respuesta a un dato recién medido.

Un sistema borroso razona (en sentido de que realiza inferencias lógicas), basándose en las zonas generadas por sus reglas. Cuando las zonas se solapan, dos o más reglas pueden

transformar cualquier número de entrada en un resultado. Cuando los datos hacen que las reglas actúen, las superficies solapadas se activan en paralelo, pero sólo hasta cierto punto.

Para ilustrar, imagínese un controlador borroso para aire acondicionado que cuente con cinco reglas y, por tanto, con cinco zonas que emparejen temperatura y velocidad de motores. Los conjuntos de temperaturas que cubren todas las entradas borrosas posibles son:

- Fría
- Fresca
- Agradable
- Cálida
- Tórrida

Los conjuntos de velocidad de motor que describen todas las salidas borrosas son:

- Muy lenta
- Lenta
- Media
- Rápida
- Muy rápida

Una temperatura de 20 grados podría ser:

- 20% fresca y 80% no fresca
- 70% agradable y 30% no agradable
- 0% fría
- 0% cálida
- 0% tórrida

Que gráficamente, y de acuerdo a la manera de representar la pertenencia a los conjuntos borrosos tenemos:

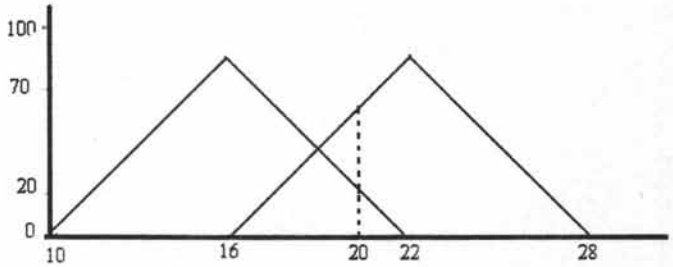


Fig. 4.4, Temperatura del aire en grados centígrados

Donde se activarían las reglas: «si es fría» y «si es agradable», determinar en base a lo anterior las velocidades de motor lenta y media. Es decir, la velocidad final del motor sería un resultado proporcional de ambas reglas. Como la temperatura era un 20% fresca, la altura de la curva que describe la velocidad del motor disminuirá a un 20% de su magnitud original; como además, era un 70% agradable, el alzado de la curva media se reducirá a un 70% del inicial. Por lo tanto, la curva definitiva para el conjunto borroso de salidas se obtendrá sumando estas dos curvas reducidas.

El paso final es un proceso de "desemborramiento" que transforma la curva de salida borrosa en un único valor numérico concreto utilizable por los controladores. La técnica más común consiste en calcular el centro de masas o centroide del área que encierra la curva. En nuestro ejemplo, el centroide de la curva de salida borrosa corresponde a una velocidad de motor de 47 revoluciones por minuto. De esta forma, partiendo de una entrada cuantitativa de temperatura, el controlador electrónico se sirve de conjuntos borrosos de temperatura y velocidad de motor para inferir la salida de velocidad apropiada y precisa. La gráfica que detalla el sistema completo de airea acondicionado mediante un controlador borroso es la que se muestra en la figura 4.5, en ella podemos apreciar las entradas y salidas del sistema, así como la curva de control.

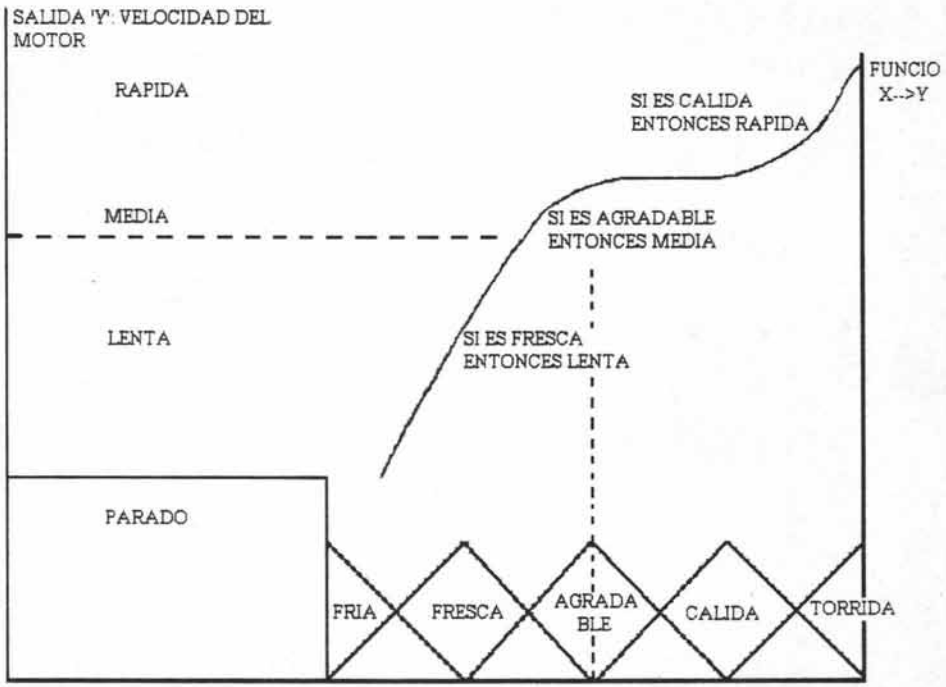


Figura 4.5, Sistema borroso de un acondicionador de aire

Las aplicaciones de la lógica borrosa van más allá de los sistemas de control difuso (un controlador difuso representa mediante reglas difusas relaciones entre variables continuas). Actualmente se le utiliza en algunos sistemas expertos y en otras aplicaciones de inteligencia artificial, en la que las variables pueden tener varios niveles de verdad o falsedad representados por rangos de valores entre el 1 (verdadero) y el 0 (falso). Por ejemplo, en las redes neuronales se le emplea con el fin de proporcionar a dichos sistemas la capacidad de aprender mediante la experiencia y el manejo de hechos con entradas semiprecisas. Como muestra tenemos al detector de explosivos ocultos en el equipaje, utilizado por algunos de los aeropuertos de los Estados Unidos. Los explosivos plásticos contienen grandes cantidades de nitrógeno, de modo que las neurocomputadoras (computadoras equipadas con redes neuronales) se equiparon con sensores que detectan la cantidad y la ubicación de nitrógeno dentro de una bolsa. De acuerdo a la teoría del entrenamiento de las redes neuronales, se puede asegurar que con el método de ensayo y

error es como el detector incrementó su índice de acierto hasta volverse un sistema confiable en su tarea.

Hasta este momento se ha descrito de forma general en que consiste la lógica borrosa y se han señalado algunas de sus aplicaciones, dejando de lado un tratamiento formal de la misma. Ahora pasaremos a señalar algunas reglas sintácticas y semánticas para el uso del calculo de predicados difuso.

Mientras que en un sistema de lógica de predicados tendríamos una notación del aire es fresco de la siguiente manera:

F = {x | es fresco}
F(aire), tal que:
Si 'F(aire)' es verdadero, entonces '~F(aire)' es falso.

Así, nunca podría ser que 'F(aire)' y '~F(aire)' fueran verdaderos o falsos al mismo tiempo, en cambio en la lógica borrosa tenemos que:

Sea E referencial (numerable o no) $x \in E$, tal que un subconjunto borroso F de E es un conjunto de parejas $\{(x,\chi_F(x)\} \forall x \in E$, donde $\chi_F(x)$ es el grado de pertenencia de x en F. Si 'F(aire)' es verdadero, también '~F(aire)' es verdadero, ya que $\chi_F(aire)=0.47$ y $\chi_{\sim F}(aire)=0.63$ al mismo tiempo.

Es decir, el aire es fresco más el no fresco nos dan una proporción de 1, y ambas son permisibles en un mismo tiempo. Así, de igual manera a como se procede en el cálculo de predicados clásico, para el calculo de predicados difuso se supone dada una signatura (universo del problema): una colección de constantes, una colección de relaciones y una colección de funciones. Cada función 'F' y cada relación 'R' tiene asociada una aridad, es decir, un número de argumentos. Los términos se forman considerando a las constantes y

a las variables como términos, y a la composición de símbolos de funciones con términos, también como términos. Las fórmulas atómicas se obtienen como composición de símbolos de relaciones con términos. Las fórmulas se obtienen considerando como tales a las fórmulas atómicas, a la composición de fórmulas con conectivos lógicos y a las cuantificaciones universales y existenciales de fórmulas. Una interpretación consiste de un universo *M*, de una correspondencia de cada constante *c* en la signatura con un elemento mc en *M*, y de una correspondencia de símbolos de funciones con funciones en M. Si f es un símbolo de función de aridad *n*, entonces mf es una función con dominio M^n y contradominio M , es decir, mf: $M^n \rightarrow M$. De esta manera, a cada término t que no involucre variables, le corresponderá un elemento m_t en M.

Entre los problemas suscitados por las lógicas multivaluadas, hay dos que merece la pena señalar brevemente. El primero es la dificultad de la interpretación de sus valores de verdad. El segundo concierne a la consideración sobre la insuficiencia para expresar los valores de verdad "graduados" que se manifiestan, ya en el lenguaje ordinario, en expresiones tales como "no completamente", "casi", "más o menos", etc.

CONCLUSIONES

Considerando la investigación histórica, sabemos que desde sus comienzos el ser humano ha buscado (y casi siempre con éxito) la manera de superar los obstáculos impuestos por sus propias limitaciones, desde la invención de la escritura como una forma de romper la barrera que le impedía interactuar con sus pares; pasando por etapas en las que su ingenio lo llevara a construir máquinas que simplificaran y resolvieran las tareas administrativas, estadísticas y contables, disminuyendo los esfuerzos del trabajo humano y acelerando el tiempo de cada proceso; hasta lograr la invención de la computadora como una de las herramientas más importantes en la época actual.

Las computadoras son el reflejo de la inteligencia humana, representan la materialización de todos aquellos aspectos del pensamiento que son automáticos, mecánicos y determinísticos. Como vimos en el capítulo primero, su invención y desarrollo ha implicado el esfuerzo y conocimiento de muchas disciplinas. Pero los resultados han sido satisfactorios, a tal grado que las computadoras han invadido ya todos y cada uno de los campos de la actividad humana: ciencia, tecnología, arte, educación, recreación, administración, comunicación, defensa, y de acuerdo a la tendencia actual, nuestra civilización y las venideras dependerán cada vez más de éstas.

No obstante, las computadoras no hacen por si solas este tránsito de manipular diferentes áreas del quehacer humano, sino que aún requieren de personas con cierta preparación que las acople, programe y adapte, para poder realizar dichas aplicaciones. Estas personas son los profesionistas dedicados a la computación, quienes mediante métodos de representación formal como: lógica de enunciado, lógica de predicados, lógica multivaluada, redes semánticas, etc, adaptan el conocimiento a una forma adecuada para poder trabajo en conjunto con los sistemas computarizados.

Una vez adaptado el conocimiento a una forma con la cual el computador pueda trabajar, el siguiente paso es determinar qué técnicas de manipulación son las conveniente para que la computadora resuelva problemas de diversa índole, de una manera satisfactoria. Como

vimos a lo largo de esta investigación, existen algunas técnicas que son básicas. Entre estas tenemos a las búsquedas ordenadas y heurísticas, cuyo objetivo es resolver y optimizar el tiempo de respuesta de una computadora, al ser empleada en la solución de problemas. Todo profesionista dedicado al área de la ciencia computacional debe tener un manejo adecuado de ellas. Sin embargo, hay otras técnicas y métodos vanguardistas. El optar por un método en vez de otro, es cuestión muy subjetiva, pero muy influenciada por las características propias del problema a resolver.

Por otra parte, con este trabajo quedo satisfecho porque cumplí con lo que me había propuesto al inicio del mismo. El de poder determinar cuáles eran y cómo funcionan las técnicas y los métodos empleados en la manipulación y representación del conocimientos. Como vimos, estos son muy diversos, y cada uno tiene diferencias importantes a considerar al momento de ser seleccionado para resolver un problema o diseñar una aplicación.

Aunque la mayoría de los programas educativos a nivel profesional del área de computo integran a la Inteligencia Artificial como una materia indispensable de formación, actualmente también es un área de especialización, pues un buen dominio de ella requiere de una capacidad de abstracción muy elevada. Espero que este trabajo sea útil para aquellos interesados en el tema.

ANEXO

Algunas partes de la presente investigación, sobre todo los capítulos finales, remiten a ciertos programas incluidos en un CD. A continuación tenemos una lista de dichos programas:

- Ajuste de parámetros (sec. 2.2.1)
- Cruce del río (sec. 3.3.1)
- Laberinto (sec. 3.3.2)
- Teorema de los cuatro colores (sec. 3.3.3)
- Juego de los once cubos (sec.3.3.4)
- El mejor camino entre capitales de los estados de México (sec. 3.3.5)
- Máximo valor de una función mediante una estrategia evolutiva y enfriamiento simulado (sec. 4.1 y 4.2)

Las características y demás pormenores de cada programa, se incluyen conjuntamente en el CD. Para cualquier consulta técnica o aclaración, estoy a sus ordenes en: ingyfil@yahoo.com.mx

BIBLIOGRAFÍA

Alchourrón, Carlos. *Lógica,* Madrid, Trota, 1995, (Enciclopedia Iberoamericana de filosofía, 7). [BC15/L3.6/c.3/UAM-I]

Beekman, George. *Computer Currents: Navigating tomorrow's Technology*, E.U., Cummings Publishing, 1994.

Cohen, Sandro. *Redacción sin dolor*, 3ª. ed., México, Planeta, 1998 (Manuales Planeta, S/N).

Copi, Irving. *Lógica simbólica*, México, CECSA, 1989.

Chalmers, Alan. *¿Qué es esa cosa llamada ciencia?*, 23ª ed., México, Siglo XXI, 1999.

Davis, Leon. *Handbook of Genetic Algorithms*, E.U., Van Nostrand Reinhold, 1991.

Dieter, NebendahL, *Sistemas Expertos*, 6ª ed., Barcelona, Marcombo1990.

Dussauchoy, Alain. *Sistemas Expertos, Métodos y Herramientas*, Madrid, Paraninfo 1988.

Dougherty E., y Ch. Giardina. *Mathematical Methods for Artificial Intelligence and Autonomous System*, E.U., Prentice-Hall, 1988.

Enciclopedia Universal Danae, III v. (México, Danae, 1982), v. I, 1982.

Enciclopedia de la Psicopedagogía, I v. (España, Océano, 1998), v. I, 1982.

Ferrater Mora, José. *Lógica matemática*, 2ª ed., México, FCE, 1994

Fraser, A. *Simulation of genetic system*, E.U., Journal of Theoretical Biology,1962.

Guzmán, Adolfo. *Técnicas Modernas de Programación*. México, Centro Nacional de Cálculo del IPN y Software Pro International, 1994.

Hernández, Arturo. *Estrategias evolutivas*, Soluciones Avanzadas No. 62, 1998.

Jeffrey, Richard. *Lógica formal: sus alcances y sus limites*, España, Universidad de Navarra, 1986.

Kurzweil, Raymond. *La era de las máquinas inteligentes*, México, CONACYT y SIRIUS, 1994

Langsam, Yedidyah., y M. Augenstein. *Estructura de datos con C y C++*, 2ª ed., México, Prentice Hall, 1997.

Mateos, Misael. *Lógica para inexpertos*, México, Edere, 2001

Miskkoff , Henry. *Inteligencia artificial*, Anaya Multimedia, 1876.

Murria, Marco., y E. Chicurel. *Aplicaciones de computación a la ingeniería*, México, Limusa, 1983.

Nilsson, Nils. Principios de inteligencia artificial, tr. del inglés por Julio Fernández Biarge, Madrid, Ediciones Díaz De Santos, 1987. [Q335/N5.1518/UAM-I]

Nueva enciclopedia temática, 14 v. (México, Editorial Cumbre, 1987), v. VII, 1987.

Penrose, Roger. *La mente nueva del emperador; en torno a la cibernética, la mente y las leyes de la física*, tr. del inglés de José Javier García Sánz, México, CONACYT – FCE, 1996. [Q335/P4.48/C.2/UAM-I]

Rico, John K. *Ciencias de la computación: problemas, algoritmos, lenguajes, información y computadoras*, México, Interamericana, 1973

Rich, Elaine. *Inteligencia artificial*, Barcelona, Gustavo Gili, 1988.

Russell, Stuart., y Perter Norving. *Inteligencia artificial; un enfoque moderno*, 5ª ed., Argentina, Prentice Hill, 1998.

Schildt, Herbert. *Utilización de C en la inteligencia artificial*, México, McGraw-Hill, México, 1992

Solis, Julio., y Y. Torres. *Lógica matemática*, México, UAM-Unidad Iztapalapa, 1995.

Tim, Hartnell. Inteligencia artificial: conceptos y programas, México, Anaya Multimedia, 1993.

Ureña, Luis Antonio, A. Sánchez, et al. *Fundamentos de informática*, Madrid, Ra-ma. 1997. [QA76/F9.42/c.5/UAM-I]

Viana, Laura. *Memoria natural y artificial*, México, F.C.E. y S. E. P., 1990 (La ciencia desde méxico, 88) [Q335/V5.3/c.7/UAM-I]

Winston, Patrick Henry. *Inteligencia artificial*, México, Addison-Wesley Iberoamericana, 1995.