INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Síntesis automática de memorias asociativas mediante programación genética

TESIS

QUE PARA OBTENER EL GRADO DE

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

Presenta
M. en C. Juan Villegas Cortez

Directores de tesis
Dr. Juan H. Sossa Azuela
Dr. Carlos Avilés Cruz

México D.F. — Noviembre 2009

820

INSTITUTO POLITÉCNICO NACIONAL

SIP-14

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 12:00 horas del día 5 del mes de Noviembre de 2009 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

"SÍNTESIS AUTOMÁTICA DE MEMORIAS ASOCIATIVAS MEDIANTE PROGRAMACIÓN GENÉTICA"

Presentada por el alumno:

VILLEGAS	**CORTEZ**	**JUAN**
Apellido paterno	Materno	nombre(s)

Con registro: A 0 6 0 1 4 5

aspirante al grado de: ***DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN***

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente

Dr. Marco Antonio Moreno Armendáriz

Secretario

Dr. Ricardo Barrón Fernández

Primer vocal
(Director de Tesis)

Dr. Juan Humberto Sossa Azuela

Segundo vocal
(Director de Tesis)

Dr. Carlos Aviles Cruz

Tercer vocal

Dr. Hector Godoy Calderón

EL PRESIDENTE DEL COLEGIO

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION

Dr. Jaime Álvarez Gallegos

INSTITUTO POLITECNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESION DE DERECHOS

En la Ciudad de México el día 25 del mes Noviembre del año 2009, el (la) que suscribe Juan Villegas Cortez alumno (a) del Programa de Doctorado en Ciencias de la Computación con número de registro A060145, adscrito al Laboratorio de Reconocimiento de Patrones, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Juan Humberto Sossa Azuela y Dr. Carlos Avilés Cruz y cede los derechos del trabajo intitulado SINTESIS AUTOMATICA DE MEMORIAS ASOCIATIVAS MEDIANTE PROGRAMACIÓN GENÉTICA, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección jvillegas@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Juan Villegas Cortez

Nombre y firma

Para las mujeres de mi casa:

Isabel *(dulce madre siempre a mi lado)*,

Aurora *(morenita de mi tierra)*,

Ñañi *(Tu sonrisa la conservo siempre en mi corazón †)*,

Josefina *(poesia ambulante)*,

Alberta *(Betty por elección, mi maestra de baile)*,

Nora Laura *(Nori y ... el Sr(a). Os@)*,

María de Lourdes *(mi'ja siempre querida)*,

Alejandra *(sonrisa feroz)*.

Para los hombres de mi casa:

Salomón *(MC)*,

Angel *(mi'jo)*,

Juan Villegas Ruiz *(El Rayito de Plata †)*.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas aquellas personas, colegas de trabajo, compañeros y amigos de laboratorios (Reconocimiento de patrones [CIC-IPN], Multimedia [UAM-Azcapotzalco], Evovisión[CICESE]), e instituciones, que con sus opiniones, trabajo y dedicación han contribuido para que este trabajo de tesis doctoral sea una realidad. En partícular quiero agradecer al Centro de Investigación en Computación (CIC), del Instituto Politécnico Nacional, por haberme proporcionado los medios, humanos y materiales, para poder realizar los estudios del programa de doctorado en ciencias de la computación.

Agradezco profundamente a mis asesores directos, Dr. Juan Humberto Sossa Azuela y Dr. Calos Avilés Cruz, y también al Dr. Gustavo Olague Caballero que fungiera como asesor indirecto; a los tres les hago presente mi admiración, agradecimiento y respeto. Gracias a su valiosa dirección, consejos, y acompañamiento se alcanzó el objetivo de esta investigación, y más allá de esto, lograron sembrar la pasión y el gusto por el quehacer científico.

Deseo expresar mi agradecimiento también a los Drs. Marco Antonio Moreno Armendáriz, Ricardo Barrón Fernández y Salvador Godoy Calderón, por sus valiosos comentarios y asesorías brindadas en la revisión de este trabajo de investigación.

Por último deseo agradecer a mis compañeros de laboratorios por su grata compañía humana y de apoyo: del Laboratorio de Reconocimiento de Patrones (CIC-IPN) a Cecilia Albortante Morato, Gabriel Omar Huerta Moreno, Jaime Alfonso Pérez León, Julio Fernando Jimenez Vielma y Laura Elena Gómez Sánchez; del Laboratorio de Evovisión (CICESE) a César Augusto Puente Montejano, Cynthia B. Perez y Leonardo Trujillo; y del Laboratorio de Multimedia (UAM-Azcapotzalco) a Agustín Sánchez y Beatriz Adriana Aguero Flores.

Índice general

Índice de figuras

Índice de cuadros

RESUMEN

Una Memoria Asociativa (MA) es un caso particular de una Red Neuronal Artificial (RNA), cuyo principal propósito es realizar una rápida asociación entre patrones, a diferencia de la asociación más compleja realizada por los modelos clásicos de RNA. Esta característica positiva ha dado lugar una nueva área de investigación consistente en la aplicación de las MA a problemas específicos del área de Reconocimiento de Patrones, tales como el reconocimiento de patrones bajo condiciones de ruido mixto o la recuperación de patrones con valores reales. Desafortunadamente, cada problema tiene sus propias dificultades y por ende reciben tratamiento diferente considerando el tipo de patrones o su posible alcance en una aplicación específica considerando la capacidad limitada del modelo clásico de MA. Por otro lado la Programación Genética (PG) ha demostrado sus grandes capacidades en la generación automatizada de soluciones, basada en una composición de conjuntos de terminales y funciones, inspirada en la teoría de la evolución biológica de las especies, para hallar soluciones apropiadas para un problema a resolver. En esta tesis se utiliza, por primera vez, la PG para la síntesis automática de MA en la solución del operador clásico de MA. Se presentan resultados experimentales en dos problemas bien conocidos en el estudio de la comunidad del Reconocimiento de Patrones. La metodología propuesta permite la generación de modelos novedosos, diferentes a los propuestos de forma tradicional por el ser humano. Esta tésis puede ser considerada hoy en día, como el primer avance en la aplicación de la PG en la síntesis automática de MA.

Descriptores: Memorias Asociativas, Redes Neuronales Artificiales, Programación Genética.

ABSTRACT

Associative memories (AM) are a particular case of artificial neural networks (ANN), whose main purpose is a faster pattern association than traditional ANN models. This positive attribute has set a new area of research consisting in the application of AMs to specific pattern recognition problems, such as pattern recognition under mixed noise, or real valued patterns recall. Unfortunately every problem has its own difficulties and receive a different treatment, considering the pattern kind or its approach into the specific application scope considering the limited capacity of the AM classic model. In the other hand, Genetic Programming (GP) have proven great success for developping automatic solutions, thus creating specialized solutions through the composition of terminals and functions using theory of evolution in order to generate a design based system for problem solving. This thesis uses GP, for the very first time, for the automatic production of AM. We present experimental results on two well-known problems that have been studied by the pattern recognition community. The proposed methodology allows us to create novel designs that are different from the traditional human-based designs. This work could be considered as the first approach that applies GP to the synthesis of automatic AM.

Keywords: Associative Memories, Artificial Neural Networks, Genetic Programming.

Simbología

$\mathbb{R}$:= Números reales.

$\mathbb{Z}$:= Números enteros.

$\mathbb{N}$:= Números naturales.

$\mathbb{B}^n = \{0, 1\}$:= Subconjunto de números enteros.

M := Matriz de memoria asociativa.

m_{ij} := componente i, j-ésimo de la memoria asociativa M

μ_{ij} := Matriz de asociación entre dos vectores, construida por medio de un operador Op^k.

$Op^k(X_i, Y_j^T)$:= Operador evolucionado de asociación entre los vectores X_i y Y_j^T.

M_k := Matriz de la Memoria Asociativa, formada a partir de $\sum \mu_{ij}$.

$\bar{X}$:= Conjunto de n vectores, acomodados en una matriz de dimensión $[n \times m]$, con cada vector tal como:

$X \in \bar{X}$, con X de dimensión $[1 \times m]$, tal que $X = \{x_1, x_2, \ldots, x_m\}$

$\widetilde{X}$:= versión con ruido del patrón X.

$\widetilde{\bar{X}}$:= conjunto de patrones X aproximado

Capítulo 1

Introducción

Una memoria asociativa (MA) es un tipo especial de red neuronal artificial (RNA) diseñada para recuperar patrones a partir de patrones de entrada, por medio de operaciones simples mediante estructuras de cálculo reducidas, en contraste con los operadores y las estructuras que usan las RNA clásicas. En una MA la información es almacenada por medio de un proceso de aprendizaje en el que un patrón de entrada X es asociado a un patrón de salida Y, relación denotada como (X^k, Y^k), con k la asociación correspondiente.

Una memoria asociativa M es representada por una matriz cuyos componentes m_{ij} pueden considerarse como las conexiones-sinapsis de una RNA simple. El operador M es generado a partir de un conjunto finito de patrones asociados, conocido *a priori*, denotado como el *conjunto fundamental* de asociaciones, el cual es representado como $\{(X^k, Y^k)|k = 1, ..., p\}$, con p el número de asociaciones.

Si $(X^k = X^k)\forall k = 1, \ldots, p$, entonces M se considera *auto-asociativa*, en caso contrario se considera *hetero-asociativa*. Sea $\hat{X}^k$ una versión distorsionada del patron X^k, es así que si alimentamos a la matriz M con $\hat{X}^k$, y la salida obtenida es Y^k se dice entonces que se tiene recuperación perfecta.

Las componentes $m_{ij} \in M$ están conformadas por operaciones simples, tales como: sumas, restas, multiplicaciones, máximos, mínimos y otras, que generalmente son compuestas a partir de éstas. Buscando preservar la simplicidad en sus estructuras, se tiene que la generación de nuevas memorias asociativas es toda una labor de arte y ciencia.

Diversos modelos de MA han sido desarrollados en los últimos cincuenta años, tan solo por mencionar algunos podemos ver [60, 14, 53, 2, 25, 47, 49, 69, 76]; sin embargo todos los modelos de MA tienen sus limitaciones, por ejemplo, su limitada capacidad de almacenamiento, dificultad de tratar con diferentes tipos de valores (binarios, bipolares, enteros,

reales), adolecen de robustez frente a patrones con ruido (aditivo, sustractivo, mixto, gausiano, etc.); y más aún, algunos modelos solo trabajan bien para un solo tipo de patrones y/o para un solo tipo de ruido para el que se realiza el diseño [82]. Cada modelo de MA ha sido diseñado cuidadosamente por personas expertas, tomando entre uno, dos o más años de tiempo de desarrollo para cada modelo.

En este tesis se presenta una propuesta nueva, inspirada en la evolución biológica, específicamente en una de las técnicas de los algoritmos bio-inspirados: la programación genética (PG) [108]. La PG se basa en principios neodarwinianos de la evolución biológica que permite la generación automática de programas de cómputo por medio de la selección natural, y es usada para hallar soluciones a problemas complejos del mundo real. La PG es considerada como una técnica nueva del *aprendizaje automatico* o *por computadora*[1]. Como tal ha sido usada para la optimización de poblaciones de programas por medio de una función de aptitud que mide la bondad de qué tan bien un programa generado resuelve una tarea específica.

La implementación de la PG comunmente es realizada sobre un conjunto de individuos o programas de cómputo descritos como estructuras de tipo árboles. Éstas son fácilmente evaluadas de forma recursiva por notación de prefijos, es así que cada nodo representa un operador función, y cada terminal (u hoja del árbol) representa un elemento a operar (valor). Hoy en día, la PG ha demostrado ampliamente su efectiva aplicación en la solución de diversos problemas reales en áreas diversas [108], particularmente podemos mencionar algunas relacionadas al área de visión por computadora (véase: [63],[70], [78], [67], [79], [80]).

Esta tesis describe por primera vez una metodología basada en PG capaz de sintetizar nuevos modelos de MA. Resultados previos pueden verse en [94]. En este trabajo se describe a detalle el perfeccionamiento de la técnica alcanzada, asi como su desarrollo por medio de experimentos tanto con patrones binarios como reales.

A continuación se proporciona una introducción a las dos áreas de técnicas que son los pilares de la investigación de esta tesis, las MA y el cómputo evolutivo; posteriormente se proporciona una breve revisión del estado del arte y la justificación de este trabajo.

1.1. Introducción a las Memorias Asociativas

En términos de patrones de entrada para RNA en la detección y reconocimiento de objetos, dos principales avances han sido usados previamente — los basados en extracción de características y los basados en pixeles [100]. En los basados en extracción de características,

[1]Del idioma inglés Machine Learning

varias de éstas, tales como el brillo, color, tamaño y perímetro, son extraídos de las subimágenes de los objetos de interés y usados como elementos de entrada. Estas características son usualmente diferentes y específicas acorde al dominio del problema. En el avance basado en pixeles, los valores de estos son usados directamente como valores de entrada.

Un tipo particular de RNA son las Memorias Asociativas (MA) [25], caracterizadas por ser de una sola capa y recuperación en un solo paso, a diferencia de las tradicionales redes neuronales artificiales clásicas que tienen que lidiar con el proceso de *back propagation* para su diseño y programación a bajo nivel [114], véase Figura 1.1. Las MA también se pueden entender como estructuras con contenido direccionable, que mapea un conjunto de patrones de entrada con otro de salida, véase Figura 1.2. Las MA en su proyección de nuevas aplicaciones a problemas complejos ha sido partiendo del enfoque de la simplicidad de operación que tienen, a la fecha no se puede afirmar que las MA puedan resolver todos los problemas en que se han aplicado las RNA. El estudio en esta tesis sobre las MA se basa en el análisis del concepto de asociación, mientras que las RNA se enfocan más como herramientas de clasificación en problemas de reconocimiento de patrones con aplicaciones vastas.

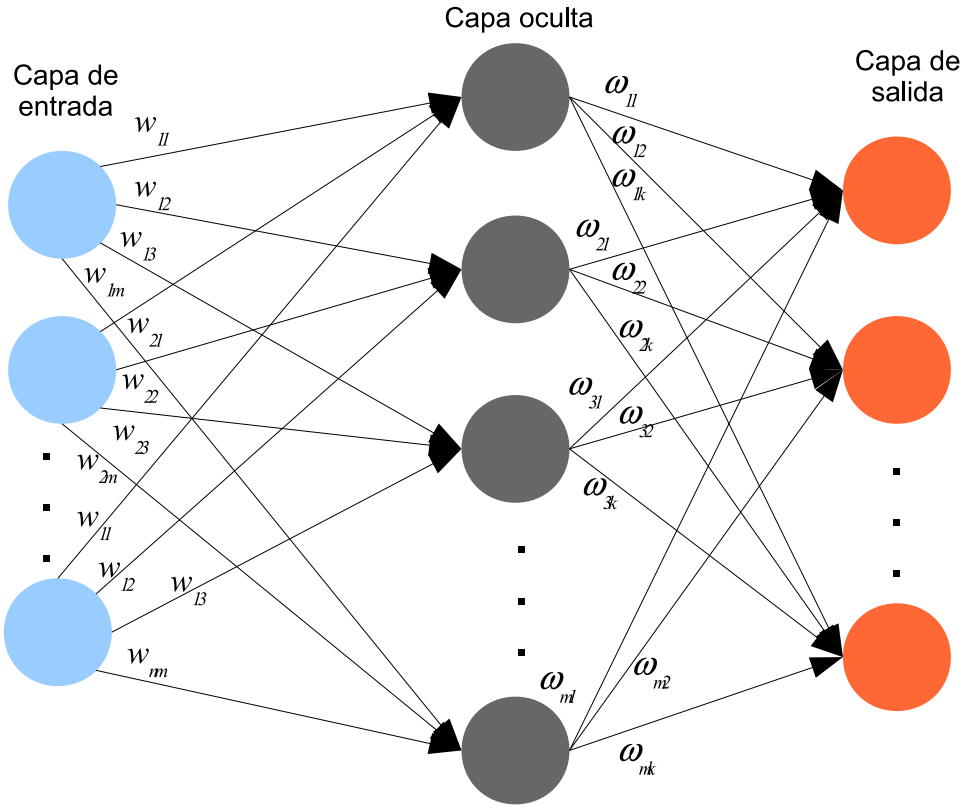


Figura 1.1: Estructura básica de una RNA.

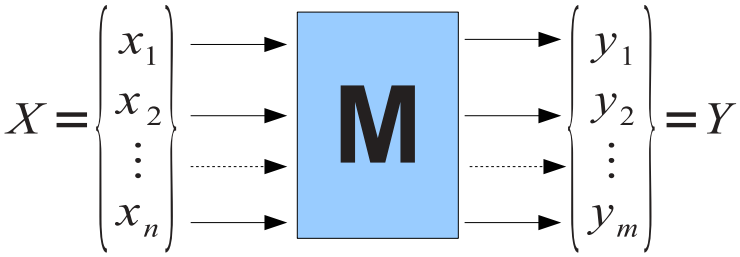


Figura 1.2: Estructura básica de una Memoria Asociativa, con X y Y patrones de entrada y salida respectivamente.

En general una memoria de contenido direccionable es un tipo de memoria que permite la recuperación de datos basados en el grado de similitud entre los patrones de entrada almacenados en la memoria, esto se refiere a una organización de la memoria por medio de la cual se accesa a ella por sus contenidos en oposición a una dirección explícita como se hace en los sistemas tradicionales de memorias de computadora; pero más aún, este tipo de memoria permite la recuperación de información basada en el conocimiento parcial de su contenido.

Supóngase que se tiene una memoria con nombres de diversas personas como se muestra en la Figura 1.3. Si la memoria es de contenido direccionable, al usar una cadena errónea, e.g., “Chrales Drarwni” como llave de entrada, esto sería suficiente para recuperar el nombre correcto “Charles Darwin”. En este sentido este tipo de memoria es robusta y tolerante a fallas, se dice que la memoria tiene capacidad de corrección de errores.

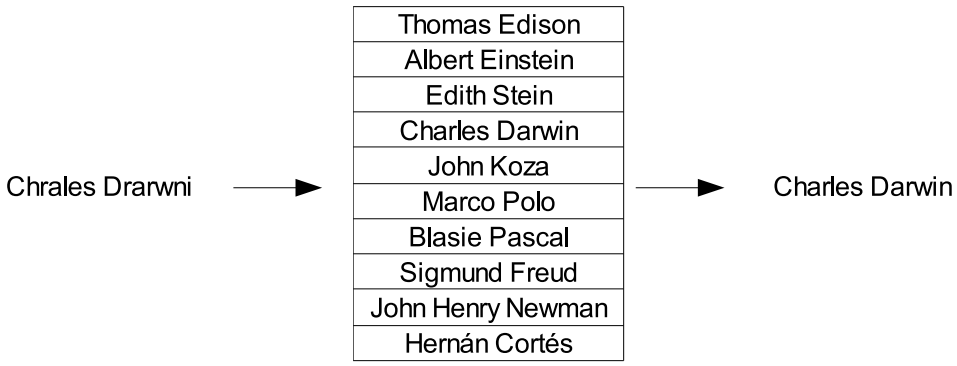


Figura 1.3: Ejemplo de una memoria de contenido direccionable.

Visto lo anterior, una MA es una estructura de contenido direccionable que mapea una representación específica de entrada a otra representación específica de salida. Es un sistema que *asocia* dos patrones (X, Y), tal que cuando uno se presenta como valor de entrada, el otro es recuperado. Los componentes de los vectores pueden ser la representación de los pixeles de imágenes cuando se hace aplicaciones de asociación de imágenes.

Una MA puede operar, en general de dos formas: en forma autoasociativa y heteroasociativa. Hay un tercer modo de operación, el bidireccional, el cual no será tratado en el contexto de esta tesis. Cuando una MA opera en forma autoasociativa permite recuperar patrones previamente almacenados a partir de versiones alteradas de estos patrones, en este caso $X^k = X^k \; \forall \; k = 1, 2, \ldots, p$. En el caso de una memoria heteroasociativa, el patrón recuperado es por lo general diferente del patrón de entrada, y posiblemente también diferencie en tipo y forma, i.e. $X^k = Y^k \; \forall \; k = 1, 2, \ldots, p$.

Uno de los modelos más simples de MA es el asociador lineal [7], pero también están los modelos de Hopfield [14] y la Memoria Asociativa Bidireccional (BAM) [21] que son populares en la literatura, cada modelo de MA trata el problema de la asociación acorde a diferentes tipos de patrones y a diferentes tipo de ruido en los patrones, tratando de hacerlas robustas, ya que una de sus particularidades es la rápida saturación de la MA, i.e., algunos modelos funcionan bien para un número reducido de patrones, o bien para patrones sencillos. En el Capítulo 2 se dará más detalle sólo de los modelos más representativos de MA y sus problemas que enfrentan respecto al tipo de patrones y el ruido en ellos.

1.2. Cómputo Evolutivo

El Cómputo Evolutivo (CP) es una área de investigación de las ciencias de la computación [115, 116]. Como el nombre lo sugiere, es un tipo especial de cómputo inspirado en los procesos naturales de la evolución. No es de asombrar el hecho de que algunos investigadores en cómputo han tomado como inspiración la evolución natural como fuente de inspiración ya que el poder de la evolución en la naturaleza es evidente en la diversidad de las especies en el mundo, cada una adaptada para sobrevivir en el medio ambiente donde se ha desenvuelto, en su propio nicho. La metáfora fundamental de cómputo evolutivo relaciona esta evolución natural poderosa para hallar soluciones para problemas dados en la heurística de ensayo-y-error [98].

A manera de introducción considérese de primera mano a la evolución natural de la forma más simple: Dado un medio ambiente con una población de individuos dispuestos para sobrevivencia y reproducción, la aptitud de los individuos —determinada por el medio ambiente— es relativa a su forma de alcanzar sus objetivos, i.e., está representa sus oportunidades de sobrevivencia y reproducción. En el contexto de la heurística de solución de problemas de ensayo-y-error (también conocida como generación-y-prueba), es que se tiene una colección de soluciones como candidatos. Su calidad (esto es, qué tan bien resuelven el problema) de-

Evolución	$\longrightarrow$	Problema a resolver
Medio ambiente	$\longrightarrow$	Problema
Individuo	$\longrightarrow$	Solución candidata
Aptitud	$\longrightarrow$	Calidad

Tabla 1.1: La metáfora básica del cómputo evolutivo y su relación con le evolución natural para la solución de un problema.

termina la oportunidad para ellos de sobrevivir, y pasar a una siguiente etapa, donde a partir de los sobrevivientes se construyan nuevas soluciones candidatas para el problema dado. Véase Tabla 1.1.

La Programación Genética (PG) es una técnica del cómputo evolutivo que permite a las computadoras resolver problemas de forma automática, es un avance relativamente reciente y de rápido desarrollo desde su concepción hace 20 años [108]. En la PG las soluciones para un problema pueden ser representadas de diferentes formas, pero usualmente se interpretan como programas de cómputo. Los principios darwinianos de selección natural y recombinación son usados para evolucionar una población de programas encaminados hacia una solución efectiva para problemas específicos. La flexibilidad y expresividad de la representación de los programas de cómputo, combinadas con las capacidades poderosas de la búsqueda evolutiva, hacen de la PG un nuevo método excitante para resolver una gran variedad de problemas [109]. Algo muy importante en este avance es que los programas evolucionados pueden ser mucho más flexibles que los altamente acotados y parametrizados modelos usados en otras técnicas, tales como las RNA y las máquinas de soporte vectorial. La PG ha sido aplicada en campos diversos y amplios como el reconocimiento de objetos, la clasificación de formas, identificación de rostros, y diagnosis médica con algunos hitos [115, 98, 51, 78, 101, 16, 36, 61, 111, 63, 64, 10, 24, 22, 23, 79, 67, 66, 75, 65, 35, 32].

1.3. Antecedentes y estado del arte

Las MA han sido usadas en la resolución de diversos problemas [46, 47, 25, 21], algunas de estas soluciones han sido desarrolladas en nuestro instituto (CIC-IPN), tales como los trabajos de Benjamín Cruz [56], Roberto Vázquez [76], del Dr. Cornelio Yáñez[41], del Dr. J. H. Sossa [53, 69] y Ricardo Barrón[60]. Más detalle de estos modelos se presentan en el Capítulo 2.

En [73] se planteó y logró con éxito la síntesis automática de RNA usando la PG, para un problema clásico y muy bien delimitado, llegando a estructuras topológicas de las RNA previstos, en similitud con [67], pero con un alto nivel de eficiencia. Este y otros trabajos han validado la posibilidad de nuestro objetivo: ***Dado el paradigma desde donde se abordan los problemas en la PG, ahora se plantea la posible síntesis automática de operadores (en memorias asociativas) usando programación genética***. Esto es, hacer la derivación de los operadores que forman las memorias asociativas M (ver Figura 1.2), apropiados para un problema dado.

Con respecto a la PG aplicada al reconocimiento de patrones, en uno de los recientes trabajos [67], se pone en práctica una novedosa técnica: *la síntesis de puntos de interés en imágenes usando programación genética*. El resultado alcanzado en este trabajo nos hace replantear los problemas clásicos de detección y reconocimiento de objetos en imágenes, en el artículo se arrojan dos nuevos detectores de puntos de interés, totalmente novedosos por la simplicidad de los mismos con respecto a los clásicos que marca la literatura del tema, y más aún cuando se hace hincapié en que los mismos fueron calculados en forma sintética, i.e., no supervisada, no sintetizados por algún ser humano; es así que por la conceptualización que se dio al planteamiento del problema desde la programación genética, en la formulación de los tres conceptos fundamentales del planteamiento de la programación genética: la función de aptitud (que refleje lo que se quiere alcanzar), el conjunto de funciones (apropiadas a usarse en el problema, en este caso funciones aritméticas simples), y el conjunto terminal (los elementos con los que trabajará el algoritmo, en este caso imágenes derivadas de cálculos de convoluciones); fue gracias a esto posible dar con soluciones nuevas a un problema conocido.

1.4. Justificación

Haciendo una revisión del detalle de construcción de los modelos de RNA clásicos [114] se ve que tras cada uno se tiene un gran trabajo de diseño, análisis de las aproximaciones con las funciones de paso para cada capa, además de lidiar con la etapa del *back propagation* al momento de hacer una programación de bajo nivel de una RNA de este tipo. Cada desarrollo de un modelo de RNA ha sido justificado para hallar la solución de una clase de problemas, y luego se adecua a un problema en particular a resolver; pero aquí el punto que las adecuaciones de su estructura y entrenamiento particulares no siempre son inmediatas.

Todo esto mueve a la necesidad de poder tener una metodología que permita generar de forma automática MA's apropiadas para la solución de diversos problemas, como se está de-

mandando en múltiples aplicaciones.

En primer lugar, se decidió por las MA como herramientas dada su simplicidad de operaciones, y porque, como se explica en el Capítulo 2, han demostrado su efectividad en recuperación de patrones [60, 49], restauración de imágenes [56, 47], reconocimiento de formas en imágenes en escala de grises y color [76], y otras aplicaciones. En segundo lugar, revisando las aplicaciones en las que ha tenido éxito la PG [57, 73, 31, 108, 65, 67, 66, 75, 43, 36, 118, 115], es que se considera factible alcanzar el objetivo con resultados que cuestionen la forma de plantear soluciones, natural por nuestra formación, como se ha demostrado en la discriminación de componentes de los patrones para la clasificación en los ambientes modelados.

1.5. Resumen

En resumen, en esta tesis la cronología de avance para alcanzar el objetivo planteado es de la siguiente manera:

- En el Capítulo 2 se habla con detalle de las MA, su formulación, diferencia con las RNA, y los principales modelos de estas; esto es una revisión del estado del arte de las MA buscando proporcionar una mejor comprensión de ellas. En el Capítulo 2 también se expone la teoría de los algoritmos bioinspirados en general y me centro en explicar más la PG, su concepción e inspiración, operadores, reglas de funcionamiento y alcances en aplicaciones reportadas.

- En el capítulo 3 se presentan las metodologías diseñadas para desarrollar nuestra propuesta.

- En el capítulo 4 se muestran los resultados obtenidos en cada una de las metodologías planteadas.

- Finalmente en el capítulo 5, se presentan las conclusiones del trabajo hasta este momento logrado, así como los productos alcanzados y los trabajos en desarrollo y futuros.

Capítulo 2

Memorias asociativas y programación genética

En este capítulo se brinda la introducción y el panorama de las dos áreas de investigación que se enlazan para el tema de investigación. Por un lado, el de las Memorias Asociativas, su naturaleza y alcance. Por otro lado, el del Cómputo Evolutivo, centrándolo en la rama de la Programación Genética.

2.1. Introducción

El cerebro humano ha sido objeto de estudio desde la antigüedad [107], de esto da constancia el documento médico más antiguo conocido del mundo, el Papiro Edwin Smith, escrito alrededor del siglo XVII a.C. (aunque se cree que se basó en textos de épocas más antiguas, de 3000 a.C.); en el papiro la palabra “cerebro” aparece ocho veces, describiendo síntomas, diagnósticos y pronósticos de dos pacientes. El cerebro, considerado como el control maestro de todo el cuerpo y el punto de inicio de todo comportamiento virtualmente hablando. Para poder controlar el comportamiento, el cerebro biológico forma modelos internos del entorno sensorial e historial. Tal entorno al cual son relacionadas todas las decisiones lo provee la memoria [119]. El concepto de memoria puede ser entendido de diferentes maneras. Generalmente, involucra un mecanismo de retención y recuperación el cual utiliza un medio de almacenamiento. Desde la perspectiva del procesamiento hay tres etapas principales en la formación y recuperación de la memoria:

- **Codificación** o registro (recepción, procesamiento y combinación de la información recibida).

- **Almacenamiento** (creación de un registro permanente de la información codificada).

- **Recuperación** (traer de nueva cuenta o volver a presentar la información almacenada en respuesta a algún estímulo o solicitud de un uso en algún proceso o actividad).

Gracias a estas operaciones, la memoria puede generar fenómenos muy complejos, desde simples ideas mentales hasta complejas secuencias de pensamiento.

Llegado a este punto uno se sitúa frente al concepto de *Inteligencia Artificial* (IA), el cual concierne al comportamiento inteligente en las máquinas. La IA es la rama de las ciencias de la computación que estudia este comportamiento. El comportamiento inteligente involucra a la percepción, el razonamiento, el aprendizaje, la comunicación y la actuación en ambientes complejos. Uno de los objetivos a alcanzar en el largo plazo para la IA es el desarrollo de máquinas que puedan hacer estas tareas tan bien como lo hacen los humanos, o posiblemente aún mejor. Otro de sus objetivos es entender este tipo de comportamiento ya sea que se presente en máquinas, humanos u otros animales. Es así que la IA tiene objetivos científicos e ingenieriles. Esta tesis se centra en la parte científica de tratar de entender el fenómeno de la asociación en la memoria del cerebro humano. Una mayor discusión del concepto de IA y sus posibles alcances puede verse en [113, 119, 107].

Desde una aproximación de modelación matemática, las operaciones de procesamiento de información dentro del cerebro pueden ser expresadas en términos de funciones de filtros adaptativos. La idea general es que el cerebro está organizado en un número de unidades funcionales. Una de estas unidades funcionales, que es una parte de las tantas redes neuronales en el cerebro, es un sistema bien definido de entradas y salidas conocido como filtro para reconocimiento o recuperación asociativo (véase Figura 1.2 del Capítulo 1).

Se cree que la información en la memoria humana es almacenada en forma de interconexiones complejas entre varias neuronas. En las redes neuronales artificiales (RNA) que juegan el papel de memorias asociativas, un dato de entrada es almacenado conjuntamente en forma de una matriz de pesos, la cual es usada para generar la salida asociada con la entrada correspondiente.

La operación básica de dicho filtro es asociar un conjunto de valores de señales de entrada, denotados por $X = \{x_1, x_2, \ldots, x_n\}$, con otro conjunto de señales de salida denotados por $Y = \{y_1, y_2, \ldots, y_m\}$.

2.2. Memorias Asociativas

Una memoria asociativa M es un dispositivo que permite asociar patrones de entrada con patrones de salida. Esto es, al presentarle a una memoria asociativa M un patrón de entrada X, ésta responde con el correspondiente patrón (vector) de salida Y. Podría decirse que una memoria asociativa es una Red Neuronal Artificial (RNA) de una sola capa.

Una Memoria Asociativa (MA) es un tipo especial de Red Neuronal Artificial (RNA) que permite la recuperación de un patrón de salida dado un patrón de entrada, aún habiendo experimentado una alteración a través de un tipo de ruido (aditivo, sustractivo o mixto) en el patrón. En las últimas décadas se han desarrollado ya diversos modelos de MA como los descritos en [1, 2, 25, 14, 76, 60, 53, 47, 49, 69]. Algúnos modelos trabajan bien sólo con un tipo de ruido (aditivo o sustractivo), pero no funcionan con ambos tipos de ruido, a excepción de los modelos propuestos en [47, 76] que son robustos al ruido mixto, y en especial el trabajo de [76] que desarrolla la aplicación de MA a patrones con valores reales en imágenes en color.

La asociación entre un patrón de entrada X y uno de salida Y se denota como (X^k, Y^k), donde k es el índice de la asociación correspondiente. La memoria asociativa M es representada por una matriz cuyos componentes m_{ij} se puede considerar como las conexiones "sinapsis" de una RNA. El operador M es generado mediante un conjunto finito de asociaciones conocidas a priori. Este conjunto conocido de asociaciones es denotado en la literatura como el conjunto fundamental de asociaciones y se representa como: $\{(X^k, Y^k)|k = 1, ..., p\}$, con *p* el número de asociaciones.

El proceso de construir dicha matriz (M) es llamado *aprendizaje* o *entrenamiento*, mientras que el obtener un patrón de salida cuando un patrón de entrada es presentado a la memoria es llamado *recuperación*.

Si $X^k = Y^k \ \forall k = 1, ..., p$ entonces M es autoasociativa $X \overset{M}{\rightarrow} X$, de otra forma es heteroasociativa $X \overset{M}{\rightarrow} Y$. Se define como $\widetilde{X}$ una versión con ruido del patrón X . Si una MA M es alimentada (propagada) con un patrón distorsionado de $\widetilde{X}^k$ y como salida se obtiene exactamente Y^k , se dice que la recuperación es correcta.

Al hablar de las entradas m_{ij} de la MA M, éstas están conformadas por operaciones muy simples como suma, multiplicación, máximos, mínimos y sub asociaciones simples, buscando establecer una asociación local entre los pares asociados, mientras que la matriz final M conforma la asociación global.

2.3. Memorias Asociativas vs. Redes Neuronales Artificiales

Las RNA usadas en técnicas de inteligencia artificial han sido conceptualizadas como un modelo aproximado del procesamiento neuronal del cerebro, aunque la relación entre este modelo y la arquitectura del cerebro biológico sigue en debate dadas las últimas investigaciones sobre el tema. El asunto tratado en las investigaciones actuales relacionadas con el tema reside en la cuestión del grado de complejidad asi como las propiedades de los elementos neuronales individuales (las neuronas). Estas debieran tender a reproducir algún comportamiento de inteligencia animal.

Históricamente, las computadoras evolucionaron a partir del modelo propuesto por von Neumann[1]. Este modelo se basa en procesamiento y ejecución secuencial de instrucciones explícitas. Por otro lado, los orígenes de las RNA se basan en esfuerzos para modelar el procesamiento de la información en sistemas biológicos, el cual se funda en un amplio procesamiento en paralelo así el como considerar instrucciones implícitas basadas en reconocimiento de patrones adquiridos a través de los sentidos desde un ambiente externo. Los sentidos vienen siendo esa conexión con el exterior. Es así que los modelos de RNA han evolucionado al considerar estos aspectos, volviéndose cada vez más complejos tratando de simular procesos en oposición a realizar tareas secuenciales de procesamiento y ejecución.

Una RNA puede ser conceptualizada en su forma más simple con características tales como: la RNA posee una estructura de propagación en una dirección, aproxima una función que delimita clases (clasificador), el aprendizaje es acorde a las conexiones (pesos, funciones de paso, interconexiones); ésta tiene su representación básica en tres capas (entrada, intermedia y de salida). Véase la Figura 2.1. Contrario a esta idea de que las RNA que se pueden complicar tanto más a partir de este modelo tratando de modelar la complejidad de los sistemas biológicos acorde con las últimas investigaciones, las MA siguen la idea simple de la asociación (aprendizaje) y recuperación, sus dos etapas, almacenando o recordando la relación entre los patrones a asociar. En su etapa de aprendizaje el proceso de asociación sigue un conjunto de operaciones, y en el proceso de recuperación usa otro conjunto de operaciones en función del primero usado para asociar. Véase Figura 2.2. Más detalles se exponen en la sección siguiente.

[1]Diseño básico de la computadora moderna o clásica. El concepto fue totalmente articulado por tres científicos principalmente involucrador en la construcción de la computadora ENIAC durante la Segunda Guerra Mundial. “von Neumann machine.” Encyclopedia Britannica. 2009. Encyclopedia Britannica Online. < http://www.britannica.com/EBchecked/topic/1252440/von-Neumann-machine >

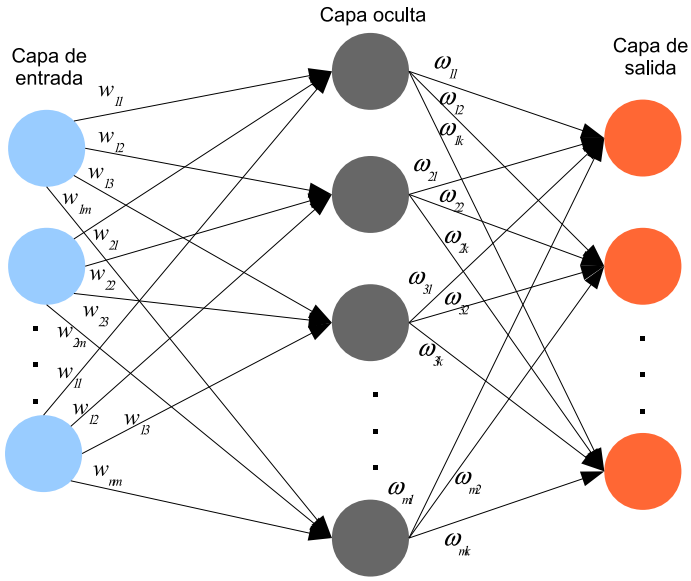


Figura 2.1: Estructura básica de una RNA.

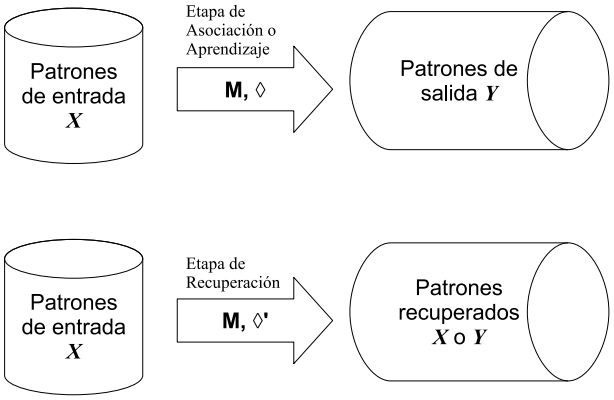


Figura 2.2: Las dos etapas de una MA con su conjunto de operaciones para la asociación ◊, y el conjunto de operaciones para la recuperación ◊ ′.

2.4. Modelos representativos de Memorias Asociativas

En esta sección se describen los modelos más representativos de MA, se explican los operadores que las definen, con el fin de entender su base de funcionamiento.

2.4.1. La red Lernmatrix de Steinbuck (1961)

La red Lernmatrix desarrollada por Steinbuch [2] es el primer modelo de MA conocido, es una memoria hetero-asociativa, y aprendía mediante la correlación hebbiana. En esta red

las neuronas se conectaban en forma matricial, asignándose un peso a cada punto de conexión. La Lernmatrix puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $X \in A^n, A = \{0,1\}$, y produce como salida la clase $Y \in A^m$. El método de asociación por correlación hebbiana es simple, a saber: para representar la clase $k \in \{1,2,\ldots,m\}$, se asignan a las componentes del vector de salida Y los valores: $y_k = 1$, y $y_j = 0$ para $j = 1,2,ldots,k-1,k+1,\ldots,m$. En la tabla siguiente se muestra la fase de aprendizaje para la Lernmatrix de Steinbuck, con la pareja de patrones fundamentales $(X,Y) \in A^n \times A^m$.

	x_1	x_2	$\ldots$	x_j	$\ldots$	$\ldots$	x_n
y_1	m_{11}	m_{12}	$\ldots$	m_{1j}	$\ldots$	$\ldots$	m_{1n}
y_2	m_{21}	m_{22}	$\ldots$	m_{2j}	$\ldots$	$\ldots$	m_{2n}
$\vdots$	$\vdots$	$\vdots$		$\vdots$			$\vdots$
y_i	m_{i1}	m_{i2}	$\ldots$	m_{ij}	$\ldots$	$\ldots$	m_{in}
$\vdots$	$\vdots$	$\vdots$		$\vdots$			$\vdots$
y_m	m_{m1}	m_{m2}	$\ldots$	m_{mj}	$\ldots$	$\ldots$	m_{mn}

Cada uno de los componentes m_{ij} de M, la Lernmatrix de Steinbuch, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} = m_{ij} + \Delta m_{ij}$, donde:

$$\Delta m_{ij} = \begin{cases} +\varepsilon & \text{si } y_i = 1 = x_j \\ -\varepsilon & \text{si } y_i = 1\ ,\ x_j = 0 \\ 0 & \text{en otro caso} \end{cases} \tag{2.1}$$

siendo ε una constante positiva escogida previamente. La fase de recuperación consiste en encontrar la clase a la que pertenece un vector de entrada $X^\omega \in A^n$ dado. Encontrar la clase significa obtener las coordenadas del vector $Y^\omega \in A^m$ que le corresponde al patrón X^ω. La $i-$ésima coordenada y_i del vector Y^ω se obtiene de la siguiente expresión, donde $\bigcup$ es el operador máximo:

$$y_i = \begin{cases} 1 & \text{si } \sum_{j=1}^{n} m_{ij}x_j = \bigcup_{h=1}^{m}[\sum_{j=1}^{n} m_{hj}x_j] \\ 0 & \text{en otro caso} \end{cases} \tag{2.2}$$

La forma planteada por Steinbuch para las fases de aprendizaje y recuperación para su memoria son poco útiles para la descripción de su funcionamiento, pero en [52] se muestra un refinamiento de este modelo para su aplicación como clasificador.

2.4.2. El asociador lineal (1972)

Este modelo fue propuesto por T. Kohonen [7]. Consiste en codificar la relación entre la información de entrada y de salida como una correlación. Posteriormente agrupa las correlaciones principales en una matriz global (asociativa). Sean $\{(x^k, y^k)|k = 1, ..., p\}$ asociaciones en $\mathbb{R}^n$ tales que $(x^i)x^j = 1$ sólo si $i = j$, y 0 de otra manera. La matriz Mdel sistema dinámico se construye como:

$$M = \sum_{i=1}^{p} y^i(x^i) \tag{2.3}$$

Al presentar un patrón de referencia x^k como estado inicial de M, se obtiene el patrón exacto correspondiente y^k tal como:

$$[M]x^k = [\sum_{i=1}^{p} y^i(x^i)]x^k = y^k\|x^k\|^2 = y^k \tag{2.4}$$

Para obtener x^k dado y^k es necesario trasponer M

$$M = \sum_{i=1}^{p} x^i(y^i) \tag{2.5}$$

Para que esta memoria presente recuperación correcta se requiere que los patrones de entrada sean ortonormales entre ellos, pero esto es difícil de encontrar en los problemas reales.

2.4.3. Modelo de Hopfield (1982)

Modelo propuesto por John Hopfield a principios de los 80's [14]. La operación de su modelo es diferente del *asociador lineal* o de Kohonen. Para el caso de la memoria de Hopfield se calcula la salida de forma recursiva hasta que el sistema se vuelve estable. En la Figura 2.3 se tiene un modelo de Hopfield con seis unidades, donde cada nodo se conecta a todos los restantes en la red.

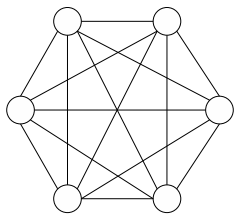


Figura 2.3: Estructura del modelo de Hopfield.

A diferencia del modelo del asociador lineal consistente de dos capas de unidades de procesamiento, una de entrada y una de salida, el modelo de Hopfield consiste de una sola capa de elementos a procesar, donde cada unidad aparece conectada a las demás, menos a sí misma. La matriz de pesos de las conexiones W de este tipo de redes es cuadrada y simétrica, i.e., $w_{ij} = w_{ji},\ \forall i, j = 1, 2, \ldots, m$. Cada unidad tiene una salida externa I_i. Esta salida adicional modifica el cálculo de la entrada por unidad como:

$$entrada_j = \sum_{i=1}^{m} x_i w_{ij} + I_j \tag{2.6}$$

para $j = 1, 2, \ldots, m$.

A diferencia del asociador lineal, en el modelo de Hopfield las unidades actúan tanto de entrada como de salida, pero al igual que en el asociador lineal, una pareja de valores es almacenado al calcular la matriz de pesos como:

$$\boldsymbol{W}_k = \boldsymbol{X}_k^T \boldsymbol{Y}_k \tag{2.7}$$

con $\boldsymbol{Y}_k = \boldsymbol{X}_k$

$$W = \alpha \sum_{i=1}^{p} W_k \tag{2.8}$$

para almacenar p pares de patrones diferentes asociados en la memoria. Dado que el modelo de Hopfield es una memoria autoasociativa son los patrones, más que los patrones en parejas, los que se almacenan en la memoria.

Una vez hecha la codificación, la recuperación se hace al presentar un patrón de entrada ***X*** y calculando su salida por propagación de la red en sus unidades para obtener el patrón ***X'***, luego este mismo patrón de salida, ***X'***, se presenta nuevamente a las unidades y se vuelven a calcular todas las unidades, la salida ***X"*** se vuelve a retroalimentar a la red. Así sucesivamente se repite el proceso hasta que a la salida un patrón se estabiliza, uno para el cual tras de ser retroalimentado una y otra vez no se presenten cambios a la salida de la red.

Si el patrón de entrada ***X*** es un patrón incompleto o una versión distorsionada del mismo, el patrón de salida en el cual la red se estabiliza es por lo general el patrón más parecido a ***X*** pero sin las distorsiones. Esta característica de *recuperación perfecta* también se le llama *restauración de patrones* y es muy útil en aplicaciones de procesamiento de imágenes.

Durante la decodificación hay varios esquemas para el cálculo de las unidades de salida.

Los esquemas son *sincronizado* (o paralelo), *asíncrono* (o secuencial), o una combinación de los dos (*híbrido*).

Hopfield demostró que el número máximo de patrones que pueden ser almacenados en su modelo de m nodos, antes del error en los patrones recuperados, es orden de $0.15m$. La capacidad de la memoria de Hopfield puede incrementarse con modificaciones [58]. Contrario a pensar en la utilidad de este modelo por baja capacidad, este modelo ha sido aplicado con éxito en diversos campos. Como por ejemplo el tema de su capacidad se sigue discutiendo a manera de hacer más flexible el modelo con modificaciones para aplicaciones afines.

Esta MA ha sido muy extendida en sus aplicaciones en la actulidad, debido principalmente a la relativa facilidad en su implementación en circuitos integrados VLSI.

2.4.4. Memorias asociativas morfológicas de Ritter (1998)

Al modelo introducido por G. X. Ritter y P. Sussner [25], se le conoce como *memoria morfológica*. Este modelo utiliza los operadores *max* y *min* que están relacionados con las operaciones morfológicas de dilatación y erosión sobre conjuntos.

Sean p asociaciones $\{(x^k, y^k)|k = 1, ..., p\}$, $x^k \in B^n,\ y^\eta \in B^m$ con $B = \{0, 1\}$. Sean $C = [c_{ij}]_{qxr}$ y $D = [d_{ij}]_{rxs}$ dos matrices. Sean las siguientes dos operaciones entre matrices:

$$E = C\nabla D = \max_{k=1}^{r}(c_{ik} + d_{kj}) \tag{2.9}$$

$$E = C\Delta D = \min_{k=1}^{r}(c_{ik} + d_{kj}) \tag{2.10}$$

Dos tipos de memorias asociativas se pueden producir a través de estas operaciones (2.9) y (2.10), la memorias *max* y la *min*. La memoria tipo *max* se construye de la siguiente manera:

Paso 1

Construir p matrices usando la ecuación (2.9) como: $y^k\Delta(-x^k)$, con $(-x^k) = (-x_1^k, -x_2^k, ..., -x_s^k)$. Véase que la correlación entre el patrón de entrada y de salida está dada por una resta.

Paso 2

Aplicar el operador *max* a las p matrices para obtener la matriz M:

$$M = \max_{k=1}^{p}[y^k\Delta(-x^k)] = [m_{ij}]_{qxs} \tag{2.11}$$

con

$$m_{ij} = \overset{p}{\underset{k=1}{max}}(y_i^k - x_j^k)$$

Para la recuperación se multiplica el patrón de entrada x^k por M de la siguiente forma:

$$y = M\Delta x^{\xi} \tag{2.12}$$

con

$$y_i = \overset{s}{\underset{j=1}{min}}(m_{ij} + x_j^{\xi})$$

El operador de correlación en este caso es una suma.

2.4.5. Memorias asociativas $\alpha\beta$ (2003)

Su operación se basa en dos operadores binarios α y β [41]. Se tienen dos tipos de memorias: M y W, que al igual que en el caso morfológico se construyen al usar los operadores **max** ($\vee$) y **min** ($\wedge$) respectivamente. Estas memorias también operan en el caso autoasociativo y heteroasociativo. El operador α es una versión desplazada de la correlación morfológica $\alpha(x, y = x - y + 1)$ y se usa durante la fase de entrenamiento, mientras que el operador β es el operador de correlación que se usa durante la recuperación. Los operadores α y β se definen de la siguiente forma:

$$\alpha : A \times A \rightarrow B \tag{2.13}$$

$$\beta : B \times A \rightarrow A \tag{2.14}$$

donde $A = \{0, 1\}$ y $B = \{0, 1, 2\}$. Sus operaciones se muestran en la Tabla 2.1. También se definen las operaciones matriciales $\vee_\alpha, \wedge_\alpha, \vee_\beta, \wedge_\beta$ cuyo detalle se muestra también en [41].

Con las operaciones matriciales se construyen las dos memorias, hablando específicamente del caso autoasociativo, M y W. Las memorias M son usadas para la recuperación en el caso de ruido aditivo, y las W para el caso de ruido sustractivo.

Debido a las propiedades algebraicas de los operadores α y β, las memorias $\alpha\beta$ presentan en muchos casos propiedades similares a las de su contra parte, las memorias morfológicas. Una de las principales ventajas de estas memorias sobre las morfológicas es que en términos

x	y	$\alpha(x,y)$	x	y	$\beta(x,y)$
0	0	1	0	0	1
0	1	0	0	1	0
1	0	2	1	0	0
1	1	1	1	1	1
			2	0	1
			2	1	1

Tabla 2.1: Valores para los operadores α y β.

binarios requieren menos operaciones para la fase de recuperación, una de sus desventajas es que sólo son útiles para el caso binario.

2.4.6. El método de independencia morfológica y representaciones mínimas de Ritter y Urcid (2006)

Este método propone la creación de *"kernels"* o *núcleos* para los patrones a asociar, esto con el fin de hacer frente al problema de patrones con ruido mixto. Por lo visto en los modelos anteriores, unas de las memorias funcionan bien para recuperar patrones alterados con ruido aditivo, otras para patrones con ruido sustractivo, pero para el caso de ruido mixto este es uno de los planteamientos. Un detalle a tomar en cuenta es que no es fácil la tarea de crear o seleccionar un conjunto óptimo de kernels, estos son acorde al tipo de patrones y no se tiene una metodología para el caso general. Para ser útiles estos kernels requieren representar versiones fuertemente erosionadas del conjunto de entrenamiento (CE), pero otro problema es que el erosionado de patrones no siempre conducirá al conjunto de núcleos deseado. Para comprender mejor esta técnica véase la secuencia de la Figura 2.4, tomada de [62].

Combinando el método para hallar núcleos [25], las nociones de independencia morfológica, independencia fuerte y representación mínima, en [6] se proporciona una herramienta para afrontar el ruido mezclado usando memorias morfológicas. En [62] muestran un avance al hecho de adecuar el método a diferentes tipos de patrones usando la técnica de *algoritmos evolutivos*, esta propuesta es novedosa y efectiva para la construcción de núcleos.

2.4.7. La memoria mediana (2004)

Modelo propuesto por H. Sossa y R. Barrón en [53], estas memorias son efectivas para la recuperación en patrones con ruido mixto, y otra ventaja es que no se limitan al caso

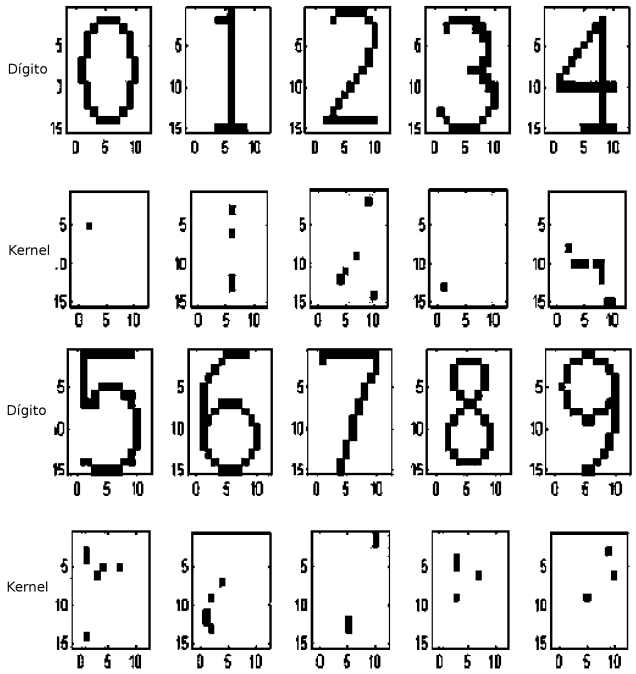


Figura 2.4: Ejemplo de imágenes “kernel” como representaciones reducidas de dígitos.

binario. Por razones de espacio aquí se expone sólo el caso heteroasociativo, para el caso autoasociativo basta hacer $x = y$.

Sea MAH la memoria mediana heteroasociativa. Sean $x \in \mathbb{Z}^n$ y $y \in \mathbb{Z}^m$ dos vectores. Para operar las MAH se requieren dos operadores, uno para entrenar a la memoria: $\diamond_A$, y otro para la recuperación: $\diamond_B$.

Se requieren dos pasos para construir la MAH:

Paso 1 :

Para cada $\xi = 1, 2, ..., p$, y para cada pareja (x^ξ, y^ξ) construir la matriz $[y^\xi \diamond_A (x^\xi)^t]_{m\times n}$ como:

$$y \diamond_A x^t = \begin{pmatrix} A(y_1, x_1) & A(y_1, x_2) & \dots & A(y_1, x_n) \\ A(y_2, x_1) & A(y_2, x_2) & \dots & A(y_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ A(y_m, x_1) & A(y_m, x_2) & \dots & A(y_m, x_n) \end{pmatrix}_{m\times n} \tag{2.15}$$

Paso 2 :

Aplicar el operador mediana a las matrices obtenidas en el paso 1 para tener la matriz M de la siguiente forma:

$$M = \underset{\xi=1}{\overset{p}{med}}[y^{\xi} \diamond_A (x^{\xi})^t] \tag{2.16}$$

La *ij*-ésima componente de M esta dada por :

$$m_{ij} = \underset{\xi=1}{\overset{p}{med}} A(y_i^{\xi}, x_i^{\xi}) \tag{2.17}$$

Para la fase de recuperación tememos dos casos:

Caso 1 :

Recuperación de un patrón del conjunto fundamental. Un patrón x^w , con $w \in \{1, 2, ..., p\}$ se presenta a la memoria Macorde con la siguiente operación:

$$M \diamond_B x^w \tag{2.18}$$

El resultado es un vector columna de dimensión *n*, con la *i*-ésima componente dada por:

$$(M \diamond_B x^w)_i = \underset{j=1}{\overset{n}{med}} B(m_{ij}, x_j^w) \tag{2.19}$$

Caso 2 :

Recuperación de un patrón alterado. Un patrón $\tilde{x}$ (versión alterada del patrón x) se presenta a la memoria MAH a través de la operación:

$$M \diamond_B \tilde{x} \tag{2.20}$$

De nuevo el resultado es un vector columna de dimensión *n*, con la i-ésima componente dada por:

$$(M \diamond B\tilde{x})_i = \underset{j=1}{\overset{n}{med}} B(m_{ij}, \tilde{x}_j) \tag{2.21}$$

Más detalles acerca de las condiciones para una recuperación perfecta están dadas en [53].

2.4.8. Modelo asociativo dinámico (2007)

La MA propuesta en [76], con este modelo de MA es posible asociar un único patrón de entrada con múltiples patrones de salida, que aplicado a imágenes es posible asociar una imagen de entrada con múltiples facetas de la misma imagen, estas vistas como imágenes vector. Este modelo se inspira en aspectos visuales de asociación en el ser humano, una entidad hacia variantes de la misma. El modelo propuesto consta de dos partes, una primera donde se realiza la asociación de patrones codificados con patrones de-codificados, su conjunto fundamental simplificado, considerando al área de separación de ambos conjuntos de patrones como una medida $\boldsymbol{d}$, así mismo esta misma medida determina el nivel de ruido soportado por el modelo, construyendo la memoria W; en la segunda parte, calcula un conjunto de kernels de la MA donde codifica su propuesta de modelo de sinapsis entre las partes de la memoria que se pueden modificar y las que no, estas últimas conforman la MA. Posteriormente la recuperación se hace a partir de los kernels propuestos.

Este enfoque hace robusta a la MA propuesta a traslaciones, rotaciones y deformaciones en las imágenes de los objetos, al ser estos casos asociaciones hacia el objeto de interés a aprender en la MA. En adición a esto el autor plantea la dinámica de obtener el resultado inverso, en lugar de recuperar la imagen asociada usando cualquier imagen de la colección aprendida, también hace recuperación de todas las imágenes que pertenecen en la colección usando solo una de ellas, i.e. busca todas las comunes a una identidad. El conjunto de memorias finales que se forma y su detalle esta en [76, 82, 83, 81, 91, 93, 92].

2.4.9. Memorias asociativas geométricas (2009)

Otro modelo muy interesante es el propuesto desde el enfoque del álgebra geométrica, las MA *geométricas* [55, 77, 85, 96], realizan una etapa de preprocesamiento de los patrones, mapeandolos hacia el espacio del álgebra geométrica, una conceptualización de las entidades distintas a como se hace con el álgebra euclidiana a la que se está acostumbrado. Este modelo logra la representación de las clases de los patrones en esferas, y a partir de cómo se definen las operaciones en espacio geométrico es que se tienen operaciones y representaciones simples y poderosas. Las MA geométricas han logrado excelentes resultados ante el ruido mixto y es uno de los modelos que nos muestra desde otra perspectiva el fenómeno clave de la "asociación" como fundamento para el aprendizaje. A continuación se resume su novedosa operación. El álgebra geométrica conforme es un marco de trabajo libre de coordenadas de $(3, 2)$ dimensiones, provee una representación conforme para objetos tridimensionales. Las esferas,

círculos y demás objetos goemétricos son objetos algebráicos con un significado geométrico. En esta álgebra, los puntos, esferas, planos, etc. se represenatan como *multi-vectores*. Un multi-vector es el producto exterior de varios vectores.

En particular, un punto euclidiano $p \in \mathbb{R}^n$ se extiende a un espacio conforme $n+2$ dimensional como:

$$P = p + \frac{1}{2}(p)^2 e_\infty + e_0 \tag{2.22}$$

donde p es una combinación linear de los vectores base euclidianos. e_0 y e_∞ representan el origen euclidiano y el punto al infinito, respectivamente, tal que $e_0^2 = e_\infty = 0$ y $e_0 \cdot e_\infty = -1$.

Una medida de distancia entre un punto conforme P y una esfera S se puede definir con la ayuda del producto interior, como sigue:

$$2(P \cdot S) = (\gamma)^2 - (s-p)^2) \ . \tag{2.23}$$

Basado en (2.23), si $P \cdot S > 0$ entonces p está dentro de la esfera, si $P \cdot S < 0$ entonces p está fuera de la esfera, y, finalmente, si $P \cdot S = 0$ entonces p está sobre la superficie de la esfera. Por lo tanto, es posible clasificar un patrón como perteneciente a una vecindad esférica, por medio del producto interior y de esta forma saber si ese patrón está dentro o fuera de la vecindad.

La fase de aprendizaje de una memoria asociativa consiste en alamcenar las asociaciones entre los patrones de entrada y sus correspondientes patrones de salida. En el caso de las Memorias Asociativas Geoemétricas (MAG), la fase de aprendizaje consiste en formar las vecindades esféricas por cada clase. Una MAG **M** es, por tanto, una matriz de tamaño $m \times (n+2)$, (donde m es el número de clases y n es la dimensión del espacio). El *k*-ésimo renglón de la memoria es, precisamente, la *k*-ésima esfera de la clase *k*. Un ejemplo de una MAG puede verse en (2.24). Donde C^k y γ^k son el centro y el radio de la *k*-ésima esfera, respectivamente.

$$\mathbf{M} = \begin{bmatrix} S^1 \\ S^2 \\ \vdots \\ S^m \end{bmatrix} \tag{2.24}$$

La fase de clasificación se realiza usando el producto interior entre un patrón de entrada $x \in \mathbb{R}^n$ (en notación fonforme X) y la MAG **M**. El resultado es un vector u. Este vector contiene todos los productos interiores esntre el patrón de entrada y las esferas de clase. Este

vector se define como:

$$u_k = \mathbf{M}_k \cdot X = S^k \cdot X \tag{2.25}$$

Cuando X se encuentra dentro de una esfera, la ecuación (2.25) regresa un valor positivo (o cero) y, cuando se encuentra fuera de la esfera regresa un valor negativo. Nótese que la fase de clasificación es idnependeinte de la fase de entrenamiento.

En algunos casos, cuando el patrón está afectado por ruido, X podría estar dentro de dos o más esferas o podría estar fuera de todas las esferas. Para decidir a cual efera pertenece dicho patrón se debe utilizar el siguiente mapeo:

$$v_k = \begin{cases} -\infty & \text{if } u_k < 0 \\ u_k - (r^k)^2 & \text{en otro caso} \end{cases} \tag{2.26}$$

Se debe hacer este procedimiento para $k = 1, \ldots, m$. El identificador de clase se obtiene de la siguiente manera:

$$j = \arg \max_k [v_k \mid k = 1, \ldots, m] \ . \tag{2.27}$$

como se puede observar, cuando X está fuera de la *k*-ésima esfera, la expresión (2.26) regresa $-\infty$. Cuando X está dentro de la *k*-ésima esfera, esa misma expresión regresa la distancia (con signo negativo) entre X y el centro de la esfera. Con esto, X será clasificado por una esfera de clase cubriendo su representación conforme; y, con ayuda de la expresión (2.27), X será clasificado por la esfera con centro más cercano a él.

La fase de clasificación es independiente de la fase de entrenamiento. Las MAG funcionan perfectamente cuando las clases son esféricamente separables.

2.4.10. Resumen de modelos de memorias asociativas

A partir del primer modelo de MA planteado como tal por Steinbuch [2], que se basa en el aprendizaje hebbiano[2], a la fecha se tiene una variedad de MA que parten de este mismo aprendizaje, entre ellas el modelo de Hopfield [14]. Grossberg también lo utilizó en 1968 [4] en la red llamada “Additive Grossberg”, y en 1973 [8] en la red “Shunting Grossberg”, ambos casos son redes de una capa con conexiones laterales y autorecurrentes.

Otras redes que utilizan el aprendizaje hebbiano son: la MA bidireccional o BAM (Bi-directional associative memory), desarrollada por Kosko en 1988 [21]; la TAM (Temporal

[2]véase el Apendice A para detalle de este concepto.

associative memory) desarrollada por Amari en 1972 [6], con la misma topología de la BAM pero ideada para aprender la naturaleza temporal de las informaciones que se le muestran (para recordar valores anteriores en el tiempo); la red de dos capas de Anderson, introducida en 1968 [3], LAM (Linear associative memory), y refinada por Kohonen [7], la red OLAM (Optimal linear associative memory) desarrollada por Wee en 1968 [5], y las variantes por Kohonen y Ruohonen en 1973 [9], con dos variantes, en una y dos capas, usando una versión optimizada del método de correlación hebbiano utilizado en la red LAM, llamado *optimal least mean square correlation (OLMSC)*.

Hablando de los modelos que utilizan variaciones del aprendizaje hebbiano tenemos el modelo de Sejnowski en 1977 [11], donde utilizó la correlación de la covarianza de los valores de activación de las neuronas. Sutton y Barto en 1981 [13] utilizaron la correlación del valor medio de una neurona con la varianza de otra. Klopf en 1986 [18] propuso una correlación entre las variaciones de los valores de activación en dos instantes de tiempo sucesivos, que denominó *drive-reinforcement* y que utilizó en redes del mismo nombre con topología feedforward de dos capas.

Otra versión de este aprendizaje es el llamado *hebbiano diferencial*, que utiliza la correlación de las derivadas en el tiempo de las funciones de activación de las neuronas. Este tipo de aprendizaje es utilizado en la red feedforward/feedback de dos capas ABAM (Adaptive bidirectional associative memory) introducida por Kosko en 1987 [19], y también Kosko en el mismo año introdujo otro modelo con la misma topología pero utilizando otra versión del aprendizaje llamado *aprendizaje hebbiano difuso (Fuzzy Hebbian learning)*; llamando así a esta red como FAM (Fuzzy associative memory), y se basaba en la representación de las informaciones que debía *aprender* la red en forma de conjuntos difusos. Por otro lado se tiene los modelos basados en la combinación de la correlación hebbiana con algún otro método como las *Boltzmann Machine* y *Cauchy Machine*; también está el modelo de tres capas llamada CPN (Counterpropagation), desarrollada por Hecht-Nielsen en 1987 [17], donde es combinado el aprendizaje hebbiano con el aprendizaje competitivo introducido por Kohonen denominado *LVQ (Learning vector quantization)*.

Dadas las posibilidades de combinación exploradas en los diversos modelos de MA, se tiene el concepto de *aprendizaje competitivo* y *cooperativo*. En las redes con aprendizaje competitivo y cooperativo suele decirse que las neuronas compiten (y/o cooperan) unas con otras con el fin de llevar a cabo una tarea dada. El objetivo de este aprendizajee es categorizar (separar en clases) los datos que se introducen a la red; pero la implementación de esta característica de aprendizaje es llevada a cado en redes neuroles de más de una capa, con lo que se deja el concepto de la sencillez de una MA y se habla ya de redes multicapa o el concepto

generalizado de RNA [120].

Podemos mencionar las variantes como el método basado en lógica difusa de Sussner, las redes neuronales morfológicas de Ritter y Urcid, basadas en operadores de multiplicación en lugar de sumas y otros más como los últimos trabajos desarrollados por [53], pero aquí la observación es apreciar como los modelos se inspiran por un lado en aspectos bio-inspirados, pero por el otro lado persiguen la modelación de un tipo de patrón específico para su clasificación y/o recuperación.

Uno de los modelos de reciente creación [69] propone una versión modificada de las memorias $\alpha\beta$ de Yáñez con el objetivo de extenderlas al caso de números reales, parte de los mismos operadores y los modifica haciéndolos más generales de tal forma que los operadores originales terminan siendo un caso particular de las nuevas memorias extendidas.

En general, los sistemas de MA, como parte de la inspiración biológica de las RNA, adaptan su comportamiento a fin de conformar un conjunto de reglas de aprendizaje y de asociación, tratando de mimetizar aspectos del aprendizaje en los sistemas biológicos, pero esta adaptación, como se puede ver tras revisar en resumen algúnos de sus modelos, es el trabajo de diseño manual supervisado que hay de por medio. Otra forma de lograr esto consiste en hacer que los modelos de sistemas biológicos se adapten a nuevas circunstancias de forma automática por medio de la *evolución* de los mismos: por medio de generaciones de descendentes reproducidos con el fin de ir mejorando el desempeño de sus antecesores. Este proceso similar a la evolución produce programas de gran utilidad aplicados en diferentes áreas. Aquí recae la propuesta de esta tesis: en aplicarla para hallar nuevas MA aplicadas a diversas circunstancias. En la sección siguiente se habla más de este aspecto.

La Tabla 2.2 muestra un resumen de los tipos de MA considerados como los representativos de los descritos previamente.

2.5. Introducción al Cómputo Evolutivo

El llevar a las computadoras la famosa teoría de Darwin (publicada en *El origen de las especies basada en la selección natural*) consiste en tratar de imitar, con programas de cómputo, la capacidad de una población de organismos vivos para adaptarse a su medio ambiente por medio de mecanismos de selección y reproducción. En los últimos cuarenta años varios métodos estocásticos de optimización se han basado en este principio. *Darwinismo Artificial* o *algoritmos evolutivos* es el nombre común para estas técnicas, quizás la más comúnmente escuchadas sean *algoritmos genéticos, estrategias evolutivas* o *programación genética*.

Modelo	**Año**	**Tipo de patrones**	**Regla**	**Asociación (A)/(H)**	**Autores**
Lernmatrix	1961	Binarios	Hebbiano	H	Steinbuch
Additive Grossberg	1968	Reales	Hebbiano competitivo	A	Grossberg
LAM	1968	Reales	Hebbiano	H	Anderson, Kohonen
OLAM	1968	Reales	Hebbiano OLMSC	A, H	Wee
Asociador lineal	1972	Binarios	Hebbiano	H	Kohonen
TAM	1972	Binarios	Hebbiano	H	Amari
Shunting Grossberg	1973	Binarios	Reales	A	Grossberg
OLAM con OLMSC	1973	Reales	Hebbiano	A	Kohonen, Rouhonen
Hopfield discreta	1982	Binarios	Hebbiano	A	Hopfield
TPM	1982	Reales	Competitivo	H	Kohonen
Hopfield continua	1984	Reales	Hebbiano	A	Hopfield
BM (Boltzmann machine)	1984	Binarios	Estocástico + Hebbiano ó corrección de error	H	Hinton, Ackley, Sejnowski
Drive-reinforcement	1986	Reales	Hebbiano	H	Klopf
CM (Cauchy machine)	1986	Binarios, reales	Estocástico	H	Szu
ABAM	1987	Reales	Hebbiano diferencial	H	Kosko
FAM	1987	Reales	Hebbiano difuso	H	Kosko
CPN	1987	Reales	Corrección error + competitivo	H	Hecht-Nielsen
BAM	1988	Binarios	Hebbiano	H	Kosko.
Morfológicas	1998	Binarios	Hebbiano	A	G. X. Ritter, P. Sussner
Alfa-Beta	2003	Binarios	Hebbiano	A, H	J. L. Díaz de León, C. Yañez
Mediana	2004	Enteros	Hebbiano	A, H	H. Sossa, R. Barrón
Representación mínima	2006	Binarios	Hebbiano	A	Ritter, Urcid
Alfa-Beta extendidas	2007	Enteros	Hebbiano	A, H	R. Barrón, H. Sossa
Dinámicas	2007	Binarios, enteros	Hebbiano	A, H	R. A. Vázquez, H. Sossa
Delta	2009	Binarios, reales	Hebbiano diferencial	A, H	M. Aldape-Pérez, C. Yáñez
Geométricas	2009	Binarios, enteros	Transformación geométrica	A	B. Cruz, H. Sossa

Tabla 2.2: Modelos de memorias asociativas más relevantes

Los componentes comunes en estas técnicas son las *poblaciones*, que representan puntos muestra de un espacio de búsqueda, y que evolucionan bajo la acción de operadores estocásticos. La evolución usualmente se organiza en *generaciones* y esto asemeja de una forma simple la genética natural. El motor de esta evolución está dada por:

1. *Una selección*, asociada a una medida de la calidad de un individuo respecto al problema a resolver,

2. *Operadores genéticos*, usualmente *mutación* y *cruza* o *recombinación*, que producen una nueva generación de individuos.

La eficiencia de un algoritmo evolutivo depende fuertemente de los parámetros establecidos: las poblaciones sucesivas (generaciones) tienen que converger hacia lo que se desea, esto es, hacia el óptimo global de una función de desempeño. Una gran parte de la búsqueda teórica en algoritmos evolutivos está dedicada al delicado problema de la convergencia, así también el tratar de focalizar qué tipo de problemas es fácil o difícil para un algoritmo evolutivo. La respuesta en teoría es que converge, pero otras cuestiones prácticas que cuestionan, como la rapidez de convergencia, permanecen abiertas. Se puede ver que el interés en las técnicas evolutivas están suficientemente bien fundadas teóricamente, y de esto que justificadamente en los últimos cuarenta años se han podido aplicar con éxito en diferentes desarrollos experimentales.

Hay que agregar que las técnicas evolutivas son de optimización estocástica de orden cero, esto es, que no son necesarias las propiedades de continuidad y deribabilidad, y la única información requerida es el valor de la función a optimizar en los puntos muestra (algunas veces hasta una aproximación puede ser usada); es así que estos métodos son particularmente adaptables para funciones irregulares, raras, complejas o con condiciones malas. Sin embargo su tiempo de cómputo puede ser muy grande.

Las técnicas evolutivas se suelen recomendar cuando los métodos clásicos y sencillos fallan (para espacios de búsqueda muy grandes, variables mixtas, cuando se tienen varios óptimos locales, o cuando las funciones son muy irregulares). En otros problemas tales como los dinámicos o interactivos también pueden retomarse desde los algoritmos evolutivos, y finalmente estos métodos también pueden combinarse (hacerlos híbridos) con los métodos clásicos de optimización (e.g. gradiente descendente, búsqueda tabú)[110].

Más allá de lo atractivo en simplicidad que pueda parecer un proceso evolutivo, la construcción y eficiencia de un algoritmo evolutivo es una tarea difícil, dado que un proceso

estocástico evolutivo es muy sensible a los parámetros y estructura del algoritmo. La elaboración de un algoritmo evolutivo eficiente se basa en un buen conocimiento del problema a resolver, así como del buen entendimiento de los mecanismos de evolución. El decidirnos a atacar la solución de un problema sin esto ultimo sería como estar tratando con un problema de "caja negra" y esto no se recomienda.

Se ha tenido mucho éxito en la implementación de estas técnicas en la industria [101], en diversos y variados campos como lo es el dominio del análisis de imágenes [43, 63, 61, 67, 66, 75, 70, 72, 74, 71, 78, 79, 89, 90, 87, 88] y la visión robótica [80].

2.6. Conceptos básicos de la evolución artificial

Los algoritmos evolutivos han tomado prestado (y considerablemente simplificado) algúnos principios de la genética natural. Es así que se habla de *individuos* que representan soluciones o puntos de un espacio de búsqueda, llamado *ambiente*. En este ambiente, hallar un máximo de una *función de aptitud* o *función de evaluación* es lo que está en juego, su búsqueda.

Los individuos se representan usualmente como códigos (reales, binarios, de tamaño variable o fijo, simple o complejo), son *cromosomas* o *genomas*, esto es, *genotipos*. Las soluciones correspondientes (i.e., los vectores del espacio de búsqueda) son *fenotipos*. Un algoritmo evolutivo involucra su población de tal manera que vuelve a sus individuos más y más *adaptados* al ambiente. En otras palabras, la función de aptitud es *maximizada*.

En la siguiente sección se describe lo más básico de un algoritmo evolutivo "canónico". Las aplicaciones en la vida real son por supuesto mucho más complejas, con el principal problema de adaptar, o más aún de crear, operadores que correspondan al problema que se tiene en mano.

2.7. El ciclo de la evolución

El primer elemento es un bucle de generaciones de poblaciones de individuos, donde cada individuo corresponde a una solución potencial del problema a tratar de resolver. Véase la Figura 2.5.

La inicialización usualmente es aleatoria (algunas veces otras estrategias también son usadas, particularmente en espacios de búsqueda complejos o multidimensionales). Las soluciones iniciales (obtenidas, e.g., usando una técnica clásica de optimización) también puede integrarse en la población inicial. Si el contenido de la población inicial en teoría no tiene

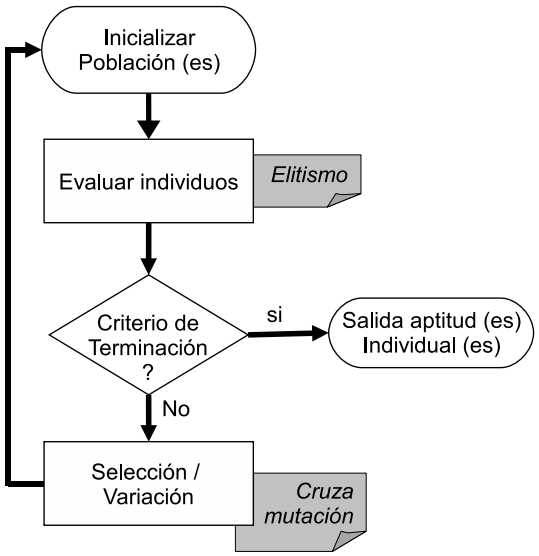


Figura 2.5: Organigrama de un algoritmo evolutivo simple.

importancia (la distribución límite para tales procesos estocásticos es siempre la misma), se ha visto experimentalmente que la inicialización tiene un gran influencia sobre la variación de los resultados y la velocidad de convergencia. Es muy eficiente algunas veces la inserción de información "a priori" acerca del problema en la etapa de inicialización.

En la selección se decide cuales individuos de la actual población se reproducirán. Esto se basa en la noción de "calidad" de un individuo, y esto se encaja en la *función de aptitud*.

El principal parámetro de selección es la *presión selectiva*, usualmente definida como el cociente de la probabilidad de seleccionar al mejor individuo sobre la probabilidad de seleccionar un individuo promedio. La presión selectiva tiene una fuerte influencia en la *diversidad genética* de la población, y consecuentemente sobre la eficiencia del algoritmo completo. Por ejemplo, una presión selectiva excesiva puede llegar a producir una rápida concentración de la población en la vecindad de sus mejores individuos, con el riesgo de una convergencia prematura hacia un óptimo local.

La selección más simple es la *selección proporcional*, implementada con un tiro aleatorio sesgado, donde la probabilidad de selección de un individuo es directamente proporcional a su valor de aptitud:

$$P(i) = \frac{aptitud(i)}{\sum_{k=1}^{TamPob} aptitud(k)} \tag{2.28}$$

Este esquema no permite controlar la presión selectiva. Otros esquemas de selección, y

más eficientes, son:

1. *Escalado*, que linealmente escala la función de aptitud en cada generación tratando de obtener un máximo en la aptitud que es *C* veces la aptitud promedio de la población actual. La *C* mide la presión selectiva, usualmente fija entre 1.2 y 2 [101].

2. *Ranking*, que le asigna a cada individuo una probabilidad que es proporcional a una lista ordenada de aptitud.

3. *Torneo*, que selecciona aleatoriamente *T* individuos de la población (independientemente de sus valores de aptitud) y elige el mejor. La presión selectiva se relaciona con el tamaño *T* de la ruleta.

La **reproducción** genera una nueva descendencia. En el esquema canónico "à la Goldberg" [101], 2 padres producen 2 hijos, un número de padres igual al número de individuos en la nueva población es seleccionado. Por supuesto también se pueden programar otros esquemas (2 padres producen 1 hijo, *n* padres producen *p* hijos, etc.).

Los dos principales *operadores de variación* son la cruza, o recombinación, que combina genes de los padres, y la mutación, que delicadamente perturba al genoma. Estos operadores son aplicados aleatoriamente, basándose en dos parámetros: probabilidad de cruza p_c y probabilidad de mutación p_m.

Intuitivamente, la selección y cruza tienden a concentrar la población cerca de individuos "buenos" (explotación de la información). Por el lado contrario, la mutación limita la atracción de los mejores individuos con el fin de dejar a la población explorar otras áreas del espacio de búsqueda.

La **evaluación** calcula (o estima) la calidad de los nuevos individuos. Este operador es el único que usa la función a ser optimizada. Ninguna hipótesis se hace sobre esta función, excepto para el hecho de que debe ser usada para definir una probabilidad o al menos una calificación para cada solución.

El **reemplazo** controla la composición de la generación $n + 1$. El elitismo se recomienda para las tareas de optimización con el fin de lograr contener a los mejores individuos de una población (generación) a la siguiente. Las estrategias suelen transmitir directamente un porcentaje dado de los mejores individuos en la siguiente población (e.g., la brecha generacional).

En el caso de implementaciones en paralelo, algunas veces es más práctico usar otros esquemas en lugar del basado en generaciones: el esquema del *estado constante* agrega

directamente cada nuevo individuo en la población en turno vía un operador de reemplazo (selección inversa) que reemplaza los individuos malos de la población actual por nuevos.

El **criterio de paro** del proceso evolutivo en el momento adecuado es crucial en términos prácticos, pero si no se tiene la mínima información disponible, o ninguna, acerca del valor óptimo buscado es muy difícil conocer cuando detener el proceso. Una estrategia usual es detener la evolución después de un número fijo de generaciones, o cuando ocurra un estancamiento. También es posible probar la dispersión de la población. Un buen control que se puede utilizar como criterio de paro que obviamente influenciará la eficiencia del algoritmo, y es un importante parámetro a considerar en la evolución (tamaño de la población, probabilidades de cruza y mutación, presión selectiva, porcentaje de reemplazo, etc.).

Un algoritmo evolutivo (AE) es parcialmente un algoritmo de búsqueda ciega, cuyo componente ciego o aleatorio tiene que ser afinado o delimitado de forma inteligente, como una función de qué es lo que se conoce “a priori” acerca del problema a ser resuelto, ya que demasiada aleatoriedad es tiempo consumido, y puede en una pequeñísima proporción hacer que el proceso se estanque en un óptimo local.

2.8. Representaciones y operadores

Los operadores genéticos dependen directamente de la elección de la representación, la cual, por ejemplo, hace la diferencia entre algoritmos genéticos, estrategias de evolución, programación genética, y evolución gramática. En la siguiente sección se muestran las representaciones más usadas, los operadores y los esquemas de selección y reemplazo. En la literatura se pueden hallar otros esquemas para espacios de búsquedas no estándar o espacios de grafos.

2.8.1. Representación discreta

Los algoritmos genéticos se basan en el uso de representaciones binarias de soluciones, extendidas más tarde a representaciones discretas.

Cada individuo de la población se representa por una cadena de longitud fija, con los caracteres (genes) elegidos de un alfabeto finito. Esta representación es obviamente apropiada para problemas de combinatoria discreta, y los problemas de tipo continuo pueden modelarse con esta representación gracias al muestreo del espacio de búsqueda. En este caso la precisión del muestreo (relativo a la longitud del cromosoma) es un parámetro importante del método.

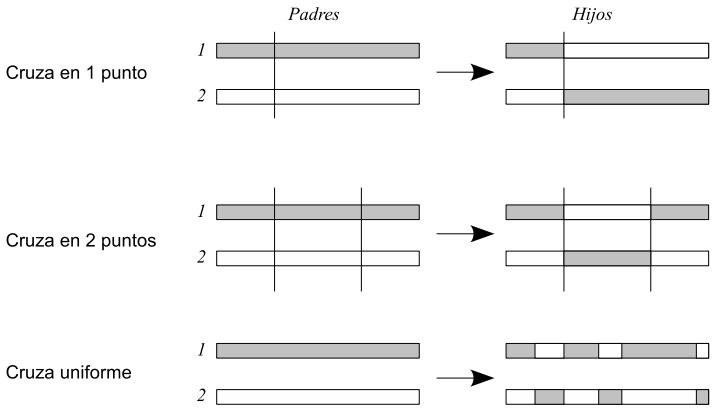


Figura 2.6: *Ejemplos de cruzas binarias.*

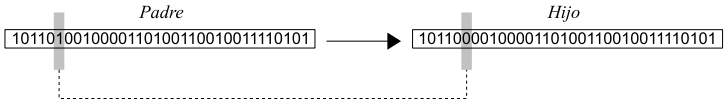


Figura 2.7: Ejemplo de mutación binaria.

Los operadores de cruza más clásicos usados en tareas de optimización se describen en la Figura 3.1. La *cruza en un punto* elige aleatoriamente un posición del cromosoma y entonces intercambia partes de cadena alrededor del punto. La *cruza a dos puntos* también intercambia porciones de los cromosomas, pero selecciona dos puntos para el intercambio. Finalmente, la *cruza uniforme* es una generalización multipunto del previo: cada gen de una nueva generación es aleatoriamente elegido de entre los genes de los padres en la posición dada. Existen otros operadores de cruza especializados, como en el problema del agente de ventas viajero o en problemas de distribución de horarios, que considera la estructura específica del gen codificado.

La mutación binaria clásica intercambia cada bit del cromosoma con una probabilidad p_m (véase la Figura 2.7). La probabilidad de mutación p_m es usualmente muy baja y es constante a lo largo del proceso evolutivo, pero en algúnos esquemas existentes, esta probabilidad de mutación decrece a lo largo de las generaciones.

2.8.2. Representación continua

La representación continua o representación real se relaciona históricamente con estrategias evolutivas. Este alcance realiza una búsqueda en $\mathbb{R}^n$ o en una parte de él. Los operadores genéticos asociados son extensiones al espacio continuo de los operadores discretos, o bien

directamente operadores continuos.

La *cruza discreta* es una mezcla de genes reales de un cromosoma, sin cambio en su contenido. Los operadores binarios de cruza previos (un punto, dos puntos, uniforme) pueden ser adaptados de forma directa.

El beneficio de una representación continua es mejor aprovechada con operadores especializados, es decir, una *cruza continua* que mezcle íntimamente los componentes de los vectores padres para producir nuevos individuos. La cruza *barycentrica*, también llamada *aritmética*, produce una descendencia x' de una pareja (x, y) de $\mathbb{R}^n$ gracias al tiro uniformemente aleatorio de una constante α en [0,1] (o $[-\epsilon, 1+\epsilon]$ para la cruza $BLX-\epsilon$) tal que:

$$\forall i \in 1, \ldots, n, \;\; x'_i = \alpha x_i + (1-\alpha) y_i \tag{2.29}$$

La constante α puede ser escogida para todas las coordenadas de *x'*, o de forma independiente para cada coordenada.

La generalización para la cruza de más de 2 padres, o de toda la población (cruza "global") se muestra con más detalle en [117]

Varios operadores de mutación se han propuesto para la representación real. La más clásica es la *mutación Gaussiana*, que agrega ruido Gaussiano a los componentes del individuo. Se requiere de un parámetro adicional, σ, la desviación estándar del ruido, tal que se aplica como:

$$\forall i \in 1, \ldots, n, \; x'_i = x_i + N(0, \sigma) \tag{2.30}$$

La afinación de σ es relativamente compleja (demasiado pequeña alentará la evolución, demasiado larga afectará negativamente la convergencia del algoritmo). Se han probado varias estrategias para hacer que σ tenga variaciones a lo largo del proceso evolutivo: σ como una función del tiempo o del valor de la aptitud, como una función de la dirección de búsqueda, o auto adaptables (i.e., considerando a σ un parámetro adicional, que vaya evolucionando con el algoritmo). Otros estudios se han enfocado en el uso de ruido no Gaussiano.

2.8.3. Representación de árboles y Programación Genética

La Programación Genética (PG) corresponde con una representación de estructuras de longitud variable como árboles. La PG fue diseñada a manejarse con programación en LISP [108], con el propósito de crear programas capaces de resolver problemas para los cuales no

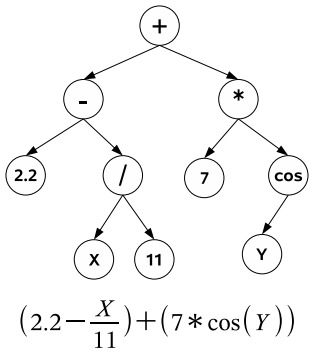


Figura 2.8: Ejemplo de una función representada como una estructura de árbol.

hubiesen sido explicitamente programados. La riqueza y versatilidad de la representación de tamaño variable en forma de árbol (véase la Figura 3.3 en el Capítulo 3), es parte del origen del éxito de la PG. Muchos problemas de optimización o control pueden ser formulados como un problema de inducción de programas. Recientemente en el dominio de la visión por computadora la PG ha demostrado alcanzar resultados humanamente competitivos [67].

Un algoritmo de PG explora un espacio de búsqueda de programas recursivos hechos de elementos de un conjunto de funciones, un conjunto de variables (datos, constantes)[3]. Los individuos de la población son programas que cuando son ejecutados producen una solución para el problema dado.

Las cruzas son un intercambio de subárboles. Las mutaciones son más complejas, y diversas mutaciones se tienen que usar, produciendo diferentes tipos de perturbaciones sobre la estructura del genoma: supresión/adición de un nodo, modificación del contenido de un nodo, mutación de las constantes (valores continuos), y la mutación de variables discretas (véase la Figura 3.5).

Las aplicaciones de la PG son numerosas, por ejemplo, en control óptimo, en planeación de trayectorias y movimientos de robots, o en regresión simbólica (la búsqueda de una fórmula matemática que aproxime un conjunto finito de puntos), por mencionar algunas [108].

2.9. Haciendo más que optimización

La evolución de una población en un espacio de búsqueda acorde con los principios previamente expuestos permite no solo localizar el óptimo global de una función compleja [111],

[3]Un problema actual en la PG es el llamado *bloat*, esto es la saturación del espacio de memoria dado el desproporcionado crecimiento del tamaño de los árboles a lo largo de la evolución. Una buena forma de evitar este efecto es limitar el tamaño del genoma.

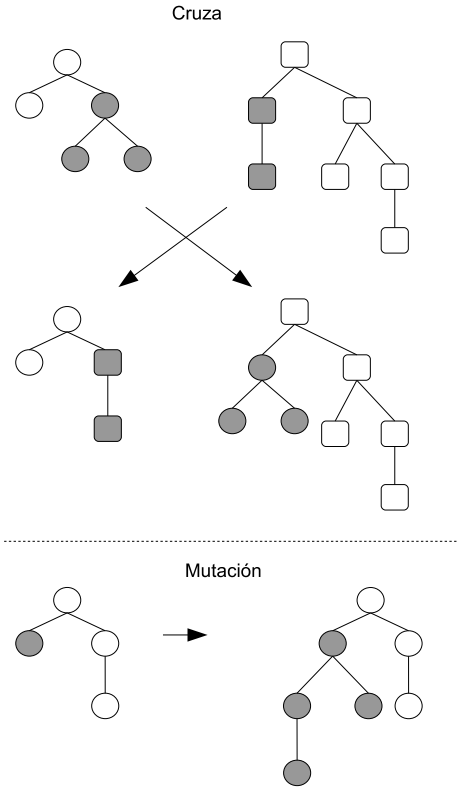


Figura 2.9: Ejemplos de los operadores de cruza y mutación en representación de árboles.

sino también obtener más información de la función y su espacio de búsqueda.

Por ejemplo, si la función a optimizar es multimodal, unas sencillas modificaciones en el proceso evolutivo permiten hacer que la población converja en subpoblaciones localizadas en "nichos" correspondientes a cada óptimo. Estos métodos controlan la diversidad de la población, o implementan un mecanismo de compartición de orígenes entre individuos vecinos [101, 16] para favorecer la emergencia de distintas especies. La definición de una distancia intra-cromosoma se vuelve necesaria.

También se puede considerar un problema de aprendizaje colectivo como la búsqueda de la solución desde el conjunto completo de individuos de una población evolucionada, y no solo por solo el mejor individuo, esto es, que la solución sea construida por todos los de la generación *n*-ésima de criterio de paro del proceso evolutivo. Las técnicas más famosas en este tipo son los sistemas clasificadores [23], la propuesta/acercamiento Parisian [33, 64], coevolución cooperativa [35], y las técnicas basadas en colonias sociales de insectos, como en los algoritmos de colonias de hormigas (ACO) [97, 29].

Por ejemplo, el modelo Parisian ha producido aplicaciones en recuperación de texto [43,

44], arte y diseño [16, 61], o más aún en aplicaciones en tiempo real (visión estéreo usando el "algoritmo mosca"[40]), es cual es representativo de los algoritmos que tienen reputación de tener un gran consumo de tiempo en CPU.

Más aún, en algunas aplicaciones, la identificación precisa de cantidades a ser optimizadas es algunas veces difícil, especialmente en casos donde existen diversos criterios de juicio, posiblemente contradictorios (e.g., maximizar la resistencia de una parte mecánica, mientras se minimiza su peso y costo). Estas optimizaciones son aún más complejas de manejar si no existe forma de estimar la importancia relativa de cada criterio. Luego entonces se considera la optimización multicriterio, sin dar prioridad algúna a los criterios varios. La solución para un problema multicriterio es así entonces un conjunto, el *frente de Pareto*[4], de óptimos comprometidos. La idea de usar técnicas evolutivas para hallar el frente de Pareto de un problema multicriterio es muy natural, y esta basado en una pequeña modificación del esquema de evolución clásico. Más precisamente, la selección de operadores se adapta parar presionar a la población hacia el frente de Pareto, mientras se conserva la diversidad para proveer un buen muestreo del frente. Una vez más, el control en la diversidad es el punto clave. Un estudio comparativo de métodos evolutivos para optimización multicriterio puede verse con más detalle en [36].

Finalmente si lo que se desea optimizar no es medible con una función matemática o un procedimiento por cómputo (e.g., la simple noción de "estar satisfechos" con un resultado), es necesaria la intervención humana en el procedimiento evolutivo, esto es, considerar los *algoritmos evolutivos interactivos*. Los primeros estudios en este campo [24, 22, 118], estuvieron orientados hacia diseños artísticos (e.g., imágenes numéricas o síntesis de estructuras en 3D). Actualmente mucho del trabajo concierne en varios dominios de aplicaciones donde las cantidades a optimizar son relativas con una valuación subjetiva (visual o auditiva); por ejemplo las características a trabajar en aplicaciones de adaptación de aparatos de audición [30], en el control de un brazo robótico para lograr movimientos cercanos a lo humano [28], o en el diseño de páginas HTML-web [34]. Una mejor revisión de este tema de aplicaciones puede verse en [26, 115, 34]. Un enfoque de competencia y cooperación derivado de la teoría de juegos aplicada en los algoritmos evolutivos se da con la *co-evolución*, que se explica con más detalle en la sección siguiente.

[4]Véase el Apéndice A para más detalle acerca del frente de Pareto.

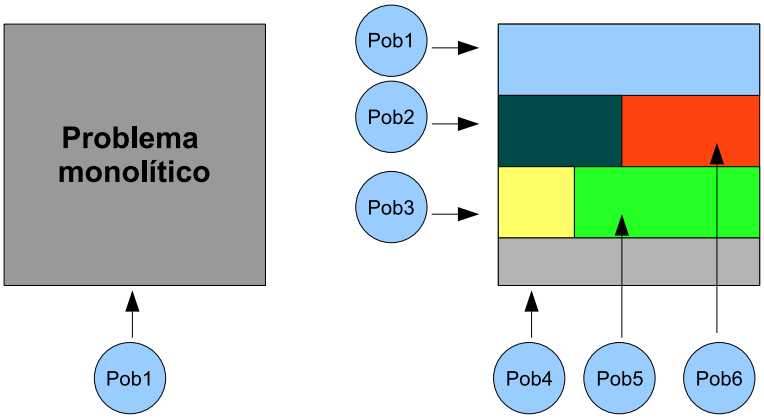


Figura 2.10: El paradigma coevolutivo: Problema monolítico (una sola población, una sola parte) vs. problema en varias partes (varias poblaciones, una para cada parte).

2.10. Coevolución

La hipótesis básica de aplicación de un algoritmo evolutivo (AE) a un problema es ver a los individuos generados como posibles soluciones del problema, pero dado el éxito de la aplicación de los AE a cada vez más problemas, estos se han vuelto más complicados o complejos acorde a los nuevos campos que se van abarcando, de esta forma la noción implícita de *modularidad* se ha ido introduciendo en la medida en que se acoplan las soluciones, i.e., se han separado en partes a los problemas en un principio enunciados como monolíticos, y a este problema se le ajusta una población de posibles soluciones a evolucionar; el enfoque se da en separar entonces poblaciones para cada parte del problema que se reconozca como separable y armar una solución a partir de las partes evolucionadas; de esta forma las posibilidades de hallar soluciones óptimas se incrementa, así también la diversidad de datos que éstas arrojan. Este paradigma de romper el problema en partes, conocido también como el *paradigma coevolutivo*, se muestra en la Figura 2.10. Problemas de ejemplos de gran complejidad los tenemos en el dominio de la robótica, donde las tareas a llevar a cabo pueden parecer simples a primera vista, pero éstas se descomponen en sub-tareas o módulos que se tienen que interconectar para poder lograr realizarce. El cómo descomponer de manera “natural” una tarea en varias, analizar cómo se interconectan, en qué medida puede ser independiente cada parte, su sub-comportamiento, es una tarea difícil y depende fuertemente del entorno del campo de aplicación [35], todo esto lleva a conceptualizar una coadaptación de las partes, y esto es un requisito crítico en el proceso evolutivo natural, mismo que inspira su implantación en los modelos tradicionales de AE.

Existen dos principales razones por las cuales los AE tradicionales no son totalmente

adecuados para solucionar este tipo de problemas. Primero, la población de individuos a evolucionar por estos AE tradicionales tienen una tendencia fuerte a converger como respuesta de un gran número de ensayos o pruebas en una área o región del espacio de búsqueda a partir de una medida de aptitud promedio; esta propiedad de convergencia fuerte evita la preservación a largo plazo de la diversidad de los subcomponentes dado que cualquiera de los individuos más fuertes o aptos será eliminado. Segundo, los individuos evolucionados a partir de los AE tradicionales representa tipicamente soluciones completas y son evaluados de forma aislada. Considerando que la interacción entre poblaciones no es modelada se tiene que no hay presión para que una coadaptación ocurra con el modelo tradicional de AE. Más aún complicado es el que se debe tener una idea pre-concebida de cuántas subpoblaciones se deben tener, qué partes modelar, y cuales no, la dificultad que surge de aquí es de qué forma se pueda dar la separación de éstas partes de forma no supervisada, de forma razonable para la naturaleza del problema a considerar. Un análisis detallado de los avances en el desarrollo de la co-evolución en los AE se tiene en [35], y de cómo su bioinspiración proviene también de la teoria de juegos se tiene en [38].

Hablando de historia, Nils Aall Barricelli fue uno de los pioneros del cómputo evolutivo al conceptualizar lo que ahora llamamos *vida artificial* en un estudio donde realizo la simulación de una población de números en un arreglo matricial [1], en el arreglo cada número se movia de acuerdo a una regla específica para cada uno, a éstas las llamó *reglas de migración*. Barricelli hizo diversas observaciones acerca de los patrones que emergieron a partir de las reglas simples, a estos los calificó de “organismos”. Éstos organismos se definieron de forma *independiente* para el caso en que pudieran reproducirse sin requerir de otro organismo proveniente de un patrón diferente, el cual definió como “otra especie”, de forma análoga un organismo que no era independiente se definió como *dependiente*, ya que requiere de otros organimos para su reproducción, por lo que se le describió también como un *parásito*. Barricelli notó nuevos patrones a partir de la recombinación de patrones con las operaciones que definió en su experimento. Posteriormente experimentó una versión de dos dimensiones de su experimento obteniendo resultados similares al trabajo del *Juego de la vida de Conway* publicado en [15], y es entonces que llega a la simulación de un juego co-evolucionado.

En la Figura 2.11 se muestra la forma convencional de operación de los algoritmos coevolutivos. Se puede ver que la diferencia aparece en la evaluación de los individuos, el cálculo de su *aptitud*, y en ello es donde recae la forma de operar de una u otra forma en que se aplica esta teoría. En general el paradigma de la coevolución trata de flexibilizar los AE para hallar mejores soluciones a los probelmas, por medio de preservar la diversidad en los individuos obteniendo con ello una rápida convergencia [50]. Por otro lado se tiene que el éxito

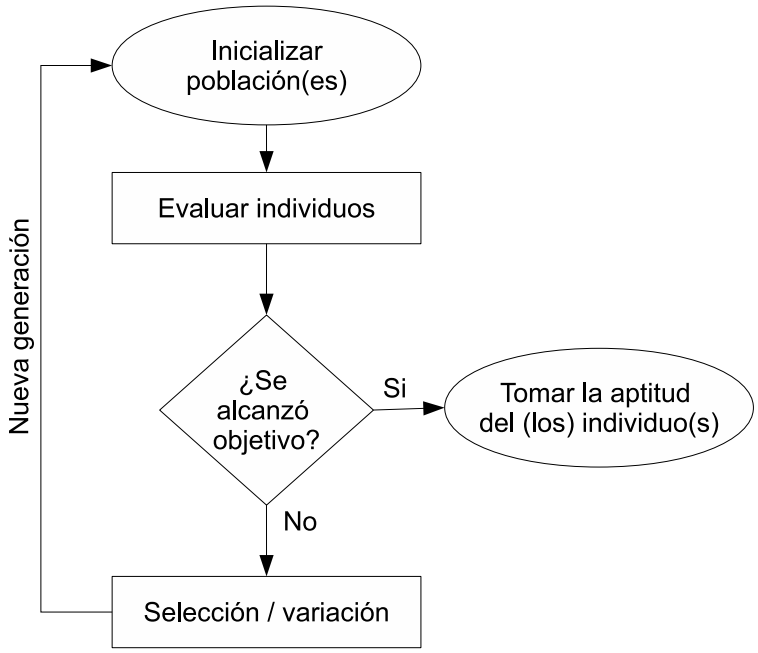


Figura 2.11: Diagrama de flujo de la coevolución convencional.

de la aplicación de la coevolución depende de la apropiada descomposición del problema en su espacio de búsqueda, el cual pudiera conocerse *a priori* y/o éste puede cambiar de forma dinámica con el tiempo, con lo que se tiene la introducción de mecanismos diversos de elitismo y diversidad cuyos diseños son particulares para cada problema.

Los algoritmos coevolutivos se pueden clasificar en dos tipos:

- Coevolución competitiva.
  El modelo de coevolución competitiva es inspirada por las interacciones depredador-presa o anfitrión-parásito, donde la presa (o el anfitrión) implementa las soliciones potenciales para la optimización del problema, mientras el depredador (o el parásito) implementa casos particulares de aptitud, con esta idea base se tienen dos sub poblaciones, donde la aptitud de una impacta en el desempeño de la otra, a fín de que una especie sobreviva a la otra.

- Coevolución cooperativa.
  El modelo de coevolución cooperativa es inspirado por la relación ecológica de simbiosis para el caso en que varias especies conviven juntas en una relación beneficiosa mutuamente. La idea básica de la coevolución cooperativa es *divide y vencerás*, i.e., dividir un sistema grande en módulos, evolucionando los módulos por separado y

entonces combinarlos conjuntamente para tener al sistema completo, de tal forma de proporcionar soluciones por partes a sistemas o problemas complejos.

En [50] se muestra un mayor detalle de estos dos tipos de coevolución, así como de sistemas de coevolución híbridos, donde se aplica para algunos problemas de optimización multiobjetivo.

Capítulo 3

Metodología y desarrollo

En este capítulo se describe la metodología utilizada para logar sintetizar de manera automática nuevos modelos de memorias asociativas (MA), mediante la técnica de programación genética (PG).

3.1. Introducción

Considerando a una MA como una estructura de contenido direccionable útil para mapear conjuntos de patrones de entrada hacia conjuntos de patrones de salida, y tras analizar los diversos modelos de MA, se puede ver, por un lado, que el conjunto de operaciones utilizado parte de operadores elementales. Por otro lado, se vislumbra la construcción de nuevos operadores que permitan relaciones entre los patrones de entrada y de salida a partir de estos operadores simples. Los distintos desarrollos de modelos asociativos han evolucionado hacia la modelación de cómo modelar una regla de asociación.

El desarrollo de una MA, como vimos en el capítulo 2, conlleva el uso de técnicas acorde al tipo de patrones que se quiera asociar, topándose siempre con el problema de la recuperación en presencia de ruido. En particular el reto, al menos durante los ultimos años, ha consistido en el desarrollo y puesta en operación de MA robustas ante el ruido mixto o mezclado. En especial cabe resaltar los trabajos realizados en [62] y [73]. En [62] se describe el uso de algoritmos genéticos para el desarrollo de núcleos a aplicarse con MA morfológicas. En [73] los autores presentan un desarrollo basado en PG aplicado al caso de RNAs. Estas técnicas siguen la estrategia general de los algoritmos bioinspirados (AB): una población inicial consistente de diferentes tipos de genotipos cada uno de los cuales codificando diferentes parámetros (típicamente el peso de la sinapsis y/o la arquitectura de la MA o de la RNA y/o

las reglas de aprendizaje), creada aleatoriamente. Esta población es evaluada para determinar qué tan bueno es cada individuo como solución; consecutivamente este conjunto se evoluciona repetidamente por medio de los operadores genéticos (replica, cruza, mutación, etc.) hasta que se cumple un criterio de terminación, que puede ser un nivel de suficiencia en la aproximación que se desea (minimización del error), o hasta alcanzar un número determinado de generaciones.

A continuación presentamos un primer ejercicio de programación genética para lograr una MA.

3.2. Desarrollo de una memoria msociativa con programación genética

Como regla general se puede considerar a la tarea de generación de MA por medio de algoritmos evolutivos como el simplemente seguir la regla de generación de RNA:

- Primero, evolucionar los pesos de la RNA a partir de una topología predeterminada [51]. En el caso de una MA, en esta tesis se considera una topología sencilla mediante el uso de operadores aritméticos simples (teniendo especial cuidado con el operador de multiplicación con vectores); este es nuestro punto de partida.

- Segundo, evolucionar de las arquitecturas, esto contempla la generación de estructuras topológicas, i.e. para establecer la conectividad para cada sinapsis, en nuestro caso, entre los componentes del vector de entrada hacia sus correspondientes componentes del vector asociado de salida, en una correspondencia uno-a-uno. La clave está en tener siempre en mente que la MA guarda todos los aspectos de la asociación, teniendo como entrada ambos conjuntos de patrones (origen y destino), durante la fase de asociación, y como salida solo el patrón salida de su correspondiente patrón entrada para la fase de recuperación.

3.3. Primer modelo

Como primer intento para obtener una MA mediante GP, será considerado como el tratar de hallar una aproximación en la asociación entre dos conjuntos de vectores, para los casos: autoasociativo y heteroasociativo. Para esto se inicia con patrones al azar, tales como vectores origen (V_x) y destino (V_y):

$$Vx = \begin{bmatrix} 1 & 0 & 1 \\ 4 & 2 & 3 \\ 6 & 5 & 5 \end{bmatrix} \text{ , } Vy = \begin{bmatrix} 4 & 2 & 3 \\ 1 & 1 & 1 \\ 2 & 6 & 0 \end{bmatrix}$$

Cada vector en operación se considera como un renglón de la matriz, e.g. $x_1 = [1 \quad 0 \quad 1]$ asociado con $y_1 = [4 \quad 2 \quad 3]$, o $x_3 = [6 \quad 5 \quad 5]$ con $y_3 = [2 \quad 6 \quad 0]$.

Usando este conjunto como base, se trato de evolucionar en forma intuitiva *el asociador lineal*, pero tratando a los patrones de forma completa, esto es al asociar todo el vector de entrada con todo un vector de salida. Es así que se toma como el *conjunto terminal* T a los vectores mismos, porque serán las entradas que tomarán los individuos X_i formados como imagen vector, permitiendo arrojar individuos de la misma forma, una imagen vector.

$$T = X_i \tag{3.1}$$

3.3.1. Función de aptitud

Como primera función de aptitud (*fitness*), se consideró la propuesta de [65, 31], que define un cociente para medir la coincidencia de pixeles de una imagen con la otra, f como se muestra en las ecuaciones (3.2) y (3.3), que nos da una medida de *similitud* $(0 \leqslant f \leqslant 1)$ como el coeficiente de correlación normalizado entre la imagen origen (*f*) y la imagen destino (*g*), para nuestro caso lo consideramos entre el vector-patrón obtenido ($\widehat{Y}$) y el que deseamos aproximar (Y).

$$f = \frac{Y \cdot \widehat{Y}}{\sqrt{Y \cdot Y}\sqrt{\widehat{Y} \cdot \widehat{Y}}} \tag{3.2}$$

con Y y $\widehat{Y}$ patrones de dimensión *1×N*, y el producto $Y \cdot \widehat{Y}$ definido como se muestra

$$Y \cdot \widehat{Y} = \frac{1}{N} \sum_{j=1}^{N} Y(1,j) \cdot \widehat{Y}(1,j) \tag{3.3}$$

El valor optimo $f = 1$ se tiene cuando todos los pixeles son comunes en ambas imágenes (o matrices), el peor valor $f = 0$ se tiene para cuando no se tiene ni un solo valor de pixel en común.

En este primer modelo las partes del modelo son:

- Op_i, el operador evolucionado, que define a una m_i, generado como individuo en forma de árbol, siendo éste el genotipo codificado.

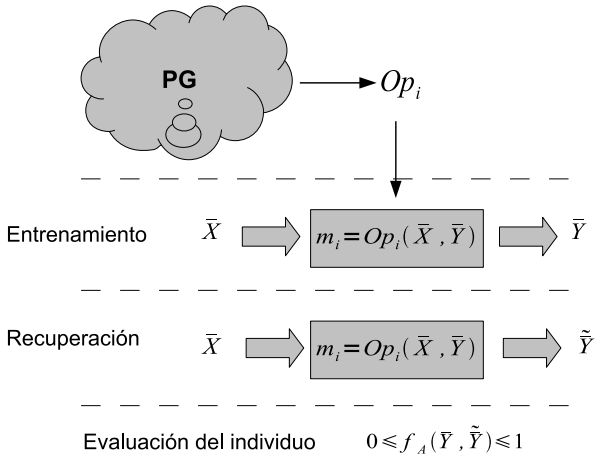


Figura 3.1: Metodología del primer modelo.

- $\bar{X}$ y $\bar{Y}$, nodos que ahora definen nuestro *conjunto Terminal*: $T = \{X_i, Y_i\}$.
- $F = \{+, -, min, max\}$, el conjunto de funciones..

3.3.2. Conjunto de funciones (F)

Véase que las operaciones que se pueden definir en el conjunto de funciones están contempladas para tomar vectores como entradas, ya que el individuo resultante será de la misma forma que una de las entradas, i.e. un vector. Luego entonces se propone el siguiente primer conjunto de funciones:

$$F = \{+, -, min, max\} \tag{3.4}$$

3.3.3. Parámetros de la Programación Genética

Los experimentos se realizaron usando Matlab© en su versión para Linux con el Toolbox GPLab versión 3.0 [48], en computadoras tipo Workstation con procesadores tipo x86/Intel/AMD de 32 y 64 bits, y 4 Gb de memoria RAM. Dado que los parametros de PG usados en nuestros experimentos realizados son similares a los usados por Koza en [108, 65], con un 0.9 la tasa de cruza y 0.1 — 0.2 la tasa de mutación. La mutación e inicialización de la población están consideradas de acuerdo al método de inicialización *ramped-half-and-half.*

3.3.4. Análisis del primer modelo

La metodología utilizada para este experimento es la siguiente (véase la Figura 3.1). De acuerdo con la función de aptitud utilizada, ésta fue programada de tal forma que cada individuo fuera evaluado permitiendo generar m_i. Para esto se tomaron en cuenta las consideraciones descritas anteriormente (conjunto de funciones, terminales, funcion de aptitud, y tasas de operaciones de mutación y cruza), aplicando la *asociación* del conjunto de valores de aprendizaje, considerando para este experimento sólo el caso heteroasociativo. Para cada individuo se realiza una etapa de entrenamiento, asociando el conjunto de aprendizaje $\bar{X}$ con el conjunto $\bar{Y}$. En segundo paso se lleva a caso el proceso de *recuperación*, ingresando el conjunto de aprendizaje $\bar{X}$ y guardando los patrones recuperados en la matriz $\widehat{\bar{Y}}$, para finalmente realizar una evaluación del individuo a partir de la función de *Aptitud* entre $\bar{Y}$ y $\widehat{\bar{Y}}$. En el código se usó la opción de que mientras el fitness se encontrara más cercano a valor cero fuera mejor (a diferencia de otros enfoques en que se buscan valores de aptitud cercanos a uno), esto para considerar la idea del error en la aproximación.

En el Capítulo 4 se muestran los resultados obtenidos con este primer modelo, sus limitantes y alcances.

3.4. Segundo modelo

Del primero modelo se puede ver que es muy limitado para ajustar valores en el espacio de búsqueda, en analogía se puede ver como tratar de ajustar con una sola recta un conjunto de valores que no se ajusta como tal[1]. Resulta evidente el hacer una mejor aproximación que explore otras posibilidades con mayor flexibilidad en los operadores. La clave ahora está en dar más holgura al manejo hacia dentro de la asociación. Al analizar el proceso global de funcionamiento de una MA, las dos etapas evidentes son la asociación (o entrenamiento), y la recuperación, pero veamos que se tiene en la primera etapa dos fases, la primera es una asociación local, armando un descriptor m_i del par asociado (x_i, y_i), luego se ensambla un descriptor que generaliza todas las asociaciones locales, que finalmente conforma la MA misma M (como lo hace el asociador lineal). Para la fase de recuperación, de forma sencilla se usa el operador de la multiplicación, según el modelo clásico [7].

En este segundo modelo se programa una nueva evolución de operadores, pero buscando evolucionar en la dirección del operador local, Op_i, como se muestra en la Figura 3.2. En este caso las delimitaciones de la PG para este modelo quedan así:

[1]Mostrado en el capítulo 4

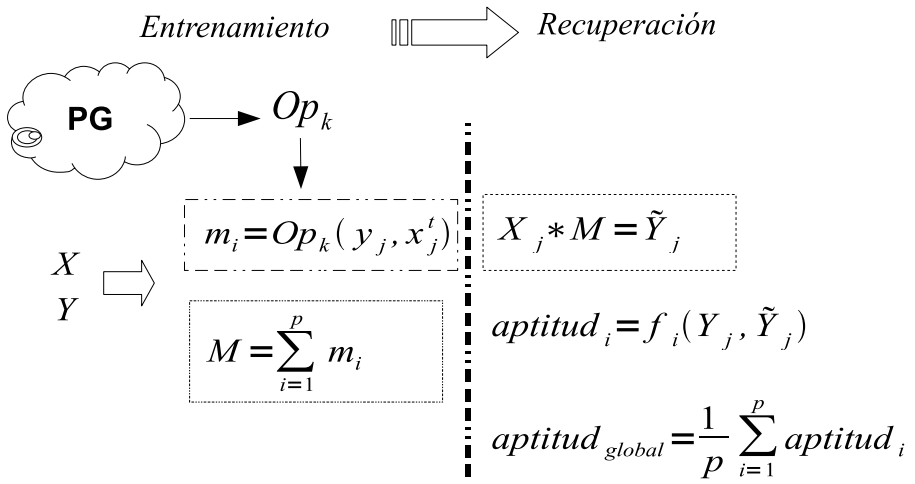


Figura 3.2: Metodología del segundo modelo

- Op_i, el operador evolucionado, que define a una M_i, generado como individuo en forma de árbol, siendo éste el genotipo codificado.

- x_i y y_i, nodos que ahora definen nuestro *conjunto Terminal* T. Estos nodos de entrada ahora son las componentes de cada vector en turno de la asociación, i.e. $x_i \in X$ y $y_i \in Y$. Es así que se define $T = \{x_i, y_i\}$, con lo que se tiene codificada la correlación local, recuperando de forma acumulativa el efecto de repetidas operaciones al llevar los valores binarios hacia la notación bipolar.

- $F = \{+, -, min, max, times\}$, el conjunto de funciones, los operadores considerados previamente, pero ahora agregado el operador de multiplicación, con el detalle de que ahora se aplica sobre valores escalares.

La parte de recuperación se ha fijado en el producto directo, entre un patrón presentado y la matriz candidata formada M_i, a partir del operador evolucionado Op_i. El cálculo de la *aptitud* se ha dejado igual que para el primer modelo, porque esa función así definida nos permite calificar qué tan buena es la recuperación del patrón, primero, haciendo un cálculo de *aptitud local*, o i*-ésima*, comparando qué tanto aproxima patrón a patrón del conjunto de aprendizaje, luego calculando una *aptitud promedio*, o *global*, que es la aptitud asignada al operador Op_i.

En el capítulo 4 se muestran los resultados, alcances y limitantes, de este modelo. Es de notar que por la flexibilidad que presenta esta metodología se puede tener un mejor ajuste en la asociación de los patrones.

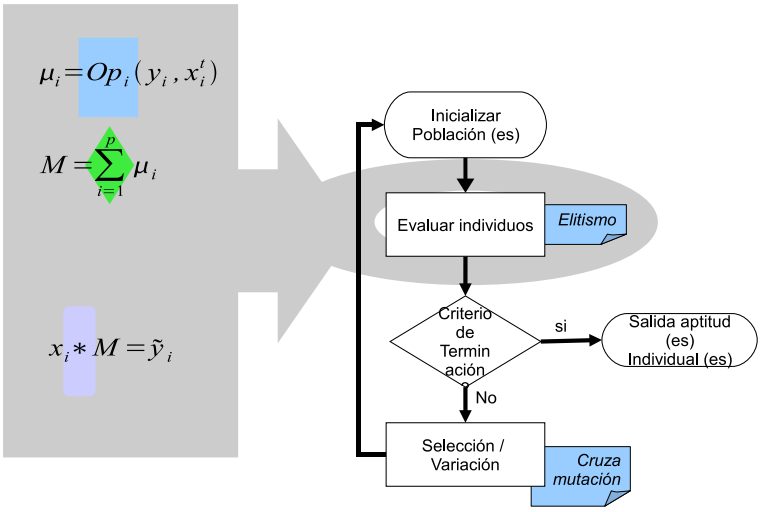


Figura 3.3: Aspectos considerados del tercer modelo.

Ejemplo de aplicación del tercer modelo

3.5. Tercer modelo: El diseño de MA a traves de coevolución

Ahora tenemos un tercer modelo con un nuevo planteamiento de nuestro modelo de asociación considerando los dos procesos básicos de toda MA, el de asociación y el de recuperación, cada uno en evolución cooperando para una solución común, una coevolución cooperativa, considerando tres aspectos fundamentales: la evaluación de los individuos, la selección o variación en la población, y las aptitudes en cada uno de los procesos evolucionados. Véase la Figura 3.2 vs. El paradigma de la coevolución convencional mostrado en la Figura 3.3.

En la Figura 3.2 se muestra que para la etapa de asociación se tienen dos partes, primero una asociación *local*, entre los patrones, y luego una asociación *global* entre todas las asociaciones locales; de forma similar para la etapa de recuperación, acorde con algunos modelos de MA como la mediana o las $\alpha\beta$, ésta se hace en dos partes también, primero una se tiene una primera recuperación, que se podría llamar *global*. A ese resultado se le aplica otra transformación para tener la salida conforme a la escala de valores de los vectores, que bien podría llamársele una recuperación *local*.

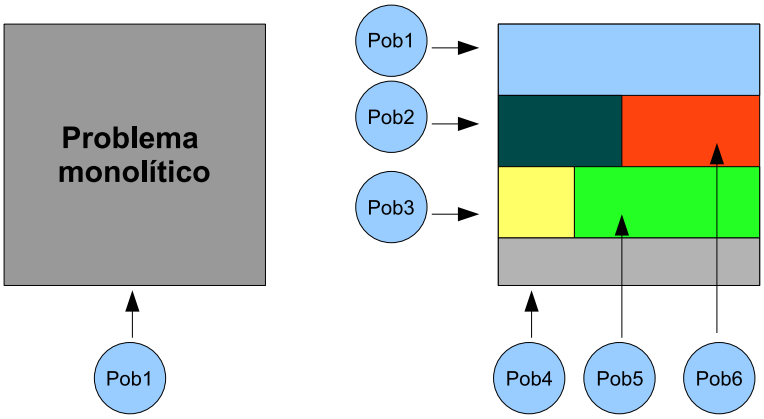


Figura 3.4: El paradigma coevolutivo: Problema monolítico (una sola población) vs. problema en varias partes (varias poblaciones, una por parte).

3.5.1. El diseño de MA con un modelo coevolutivo

Siguiendo nuestras ideas planteadas en el párrafo anterior para lograr un mejor diseño automático de MA ahora vemos al problema con la perspectiva de "divide y vencerás". Hasta este momento se ha visto al problema de forma monolítica, pero viendo al problema en varias partes ahora se plantea hallar una solución para cada una de ellas, que la solución de una parte involucre a la otra, este es el aspecto *coevlutivo*. En la Figura 3.4 se muestra el paradigma de la coevolución, mismo que se implementa en este tercer modelo, y es el aspecto central.

En este planteamiento se consideran dos poblaciones, una correspondiente a la etapa de asociación, y la otra a la etapa de recuperación. En los dos modelos previos se ha considerado sólo una población a evolucionar, la correspondiente a la asociación (habiendo ya evolucionado operadores-reglas de asociación), y se ha fijado un operador para la recuperación, y es esta segunda etapa la que ahora es mejorada.

El paradigma coevolutivo inspirado en aspectos biológicos plantea dos aspectos en el proceso de evolución por partes en su interacción, el aspecto de competencia (supervivencia de una sola especie en juego depredador-presa), y el de cooperación (cada una de las especies aporta una parte de la solución) [86]. Con esto en mente ahora se tienen dos espacios de búsqueda, uno para cada etapa, y se tiene el modelo en tres pasos:

- Primero, se definen manualmente los operadores para la etapa de asociación.
- Segundo, se define el espacio de recuperación considerando la matriz de asociación formada con cada una de las reglas-operadores de asocición evolucionados en el primer paso.

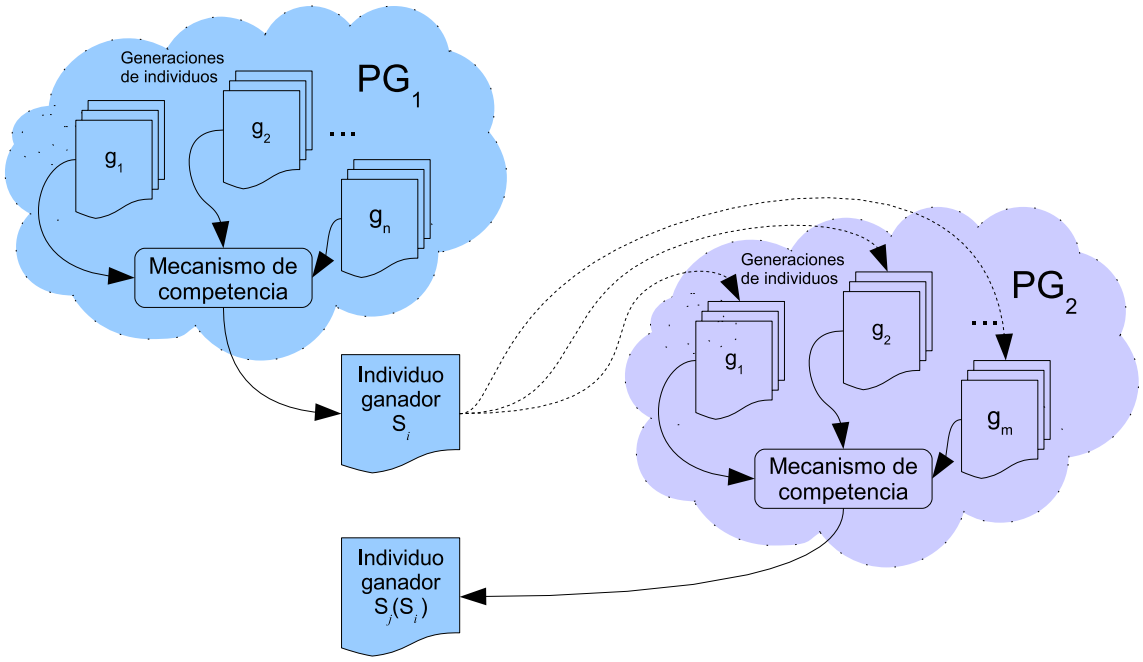


Figura 3.5: Diagrama del modelo co-evolutivo propuesto para el desarrollo de MA mediante PG.

- Tercero, se implementa el aspecto co-evolutivo que nos resuelve el problema usando ambos espacios, conectándolos a partir de evolucionar reglas-operadores para la etapa de recuperación.

En la Figura 3.5 se muestra el diagrama de aplicación de esta idea en tres pasos. En la figura se muestra que las soluciones, las nuevas MA, provienen de ambos espacios de búsqueda, y de la interacción de ambos procesos evolutivos, i.e., la influencia del primer proceso evolutivo en el segundo proceso. La función de aptitud del proceso global pretende ser representativa.

Los componentes de éste modelo se definen a continuación (véase la Figura 3.6):

- Op_k^a. El operador evolucionado para la asociación de patrones. Este corresponde al genotipo codificado. La matriz de asociación local μ_i se construye aplicando éste operador sobre las componenes de cada par de vectores en la asociación local $\{Y_j, X_j^T\}$.

- M_k. La matriz de asociación general, se construye sumando todas las matrices de asociaciones locales μ_i, proporcionando la acumulación del conocimiento de asociación local (por inspiración del principio del perceptrón).

- T_a. El *Conjunto Terminal* para la etapa de asociación, $T_a = \{x_j, y_j\}$. Estos nodos son entradas de sus respectivos vectores-patrones $\{X\}$ y $\{Y\}$; esto es para tener codificada la asociación local como una correlación de entrada.

- F_a. El *conjunto de funciones* para la etapa de asociación, como ya se definió para el segudo modelo: $F_a = \{+, -, min, max, times\}$.

- Op_p^r. El operador evolucionado para la recuperación de los patrones. Éste involucra al vector de entrada X_j, así como a la matriz de asociación M generada por el operador evolucionado del primer proceso.

- T_r. El *conjunto terminal* para la etapa de recuperación. $T_r = \{v, Ren_1, Ren_2, \ldots, Ren_m, M_k\}$, con $v \in X$ como el vector de entrada, y Ren_i como el vector-renglón i-ésimo que pertenece a M_k.

- F_r. El *conjunto de funciones* para la etapa de recuperación. $F_r = \{+, -, min, max, mytimesm\}$; donde $mytimesm$ se define como el operador de multiplicación entre las componentes de los vectores como: $mytimesm(X, Y) = [x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n]$, y cuando se considera en la operación la matriz de asociación en turno, M_k, se define con el orden de operación como: $mytimesm(X, M_k) = X * M_k$; es así que se satisface la condición de dimensiones entre los vectores y la matriz para la operación.

- $\hat{Y_j}$. Finalmente, el patrón aproximado resultante de Y_j, de la aplicación de la regla-operador evolucionado Op_p^r en la etapa de recuperación.

El proceso evolutivo demanda la asignación de un valor de *aptitud* para los individuos de cada etapa. La aptitud que asignamos para cada etapa es a partir de la misma ecuación del coeficiente normalizado de comparación entre los vectores-patrones que en los dos modelos anterior (ecuaciones (3.2) y (3.3)), para esta primera etapa de experimentación con este tercer modelo.

La función de aptitud se aplica para evaluar cada individuo generado mu_i de la etapa de asociación entre el conjunto de patrones origen X y el conjunto de patrones destino Y, ambos definen el conjunto fundamental. Una evaluación de aptitud se realiza en cada parte del proceso de la siguiente forma:

- Primero, aplicamos la asociación entre X y Y.

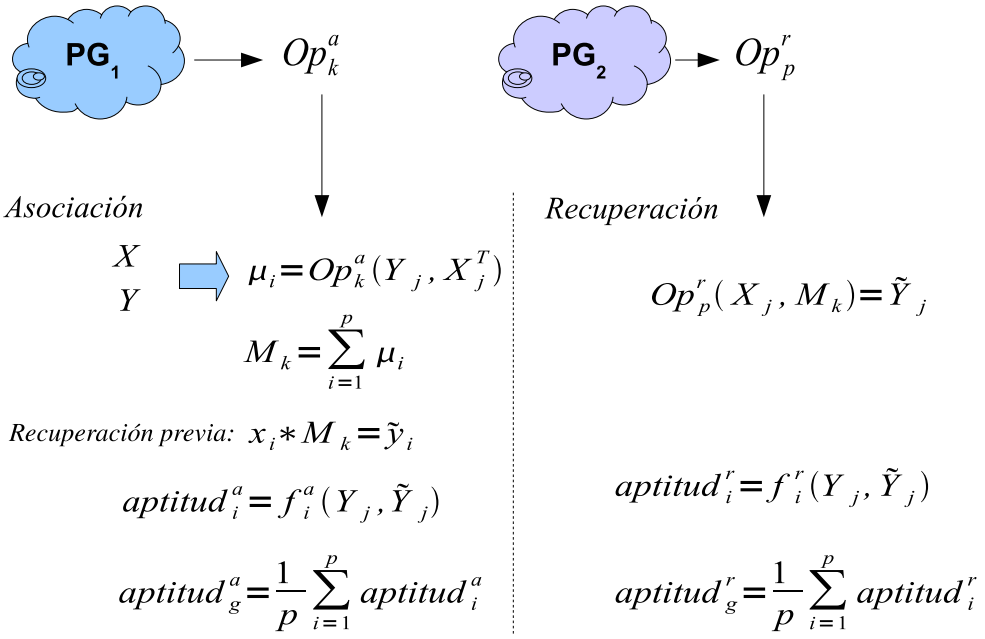


Figura 3.6: Modelo co-evolutivo propuesto para el desarrollo de MA mediante PG.

- Segundo, se realiza una recuperación simple usando el operador de multiplicación entre la matriz de asociación general generada en esta primera etapa M_k, y el vector de entrada X, como se hizo en el segundo modelo, con la idea de tener un estimador de aptitud local para la asociación. Los patrones recuperados en esta etapa son $\widehat{\widehat{Y}}$.

- Tercero, se calcula la aptitud entre los patrones Y y $\widehat{Y}$ para todas las asociaciones locales $aptitud_i^a$, para luego calcular la aptitud asociada a la asociación por medio del operador Op_a^r, y la calculamos como una aptitud promedio global de la asociación $aptitud_g^a$.

Este proceso lo repetimos para la etapa de recuperación:

- Primero, se calcula la recuperación del patrón asociado a cada vector de entrada X, usando el operador ganador del proceso evolutivo de recuperación Op_p^r.

- Segundo, se calcula la aptitud para este operador-regla usando la misma función (Ec. (3.2)), la aptitud de recuperación local $aptitud_i^r$, para cada par de asociación, asociado a una matriz de asociación M_k generada con cada operador ganador del proceso de asociación Op_k^r.

- Tercero, se realiza el cálculo de la aptitud global promedio asociado al operador de recuperación evolucionado, $aptitud_g^r$

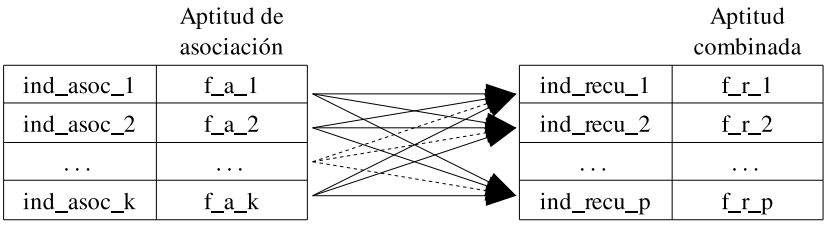


Figura 3.7: Modelo co-evolutivo propuesto de evaluación de la aptitud co-evolutiva para el desarrollo de MA mediante PG.

Finalmente se arma con el cálculo de la aptitud global para la etapa de recuperación y cada matriz de asociación generada con cada operador evolucionado en la etapa de asociación, una tabla para conocer con que regla-operador de asociación se obtiene en competencia el mayor valor de aptitud con cada operador de recuperación. Esto se muestra en la Figura 3.7.

Con este modelo coevolutivo se obtienen resultados excelentes, tanto para el caso de patrones binarios como con valores reales. Muestra tener la flexibilidad deseada para lograr ajustar reglas de asociación por duplas (una para cada etapa). En el capítulo 4 se muestran los resultados alcanzados con experimentos prácticos como con casos de bases de datos con valores de aplicaciones reales.

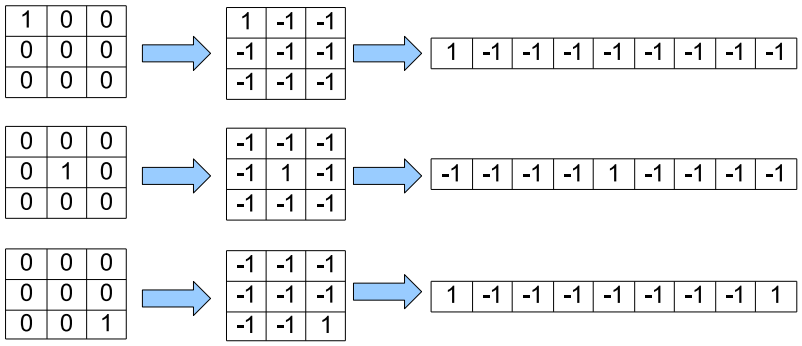
3.5.2. Ejemplo de aplicación del tercer modelo

A continuación mostramos el funcionamiento de nuestro tercer modelo, con un ejemplo con patrones binarios. Para el caso binaro se tiene la particularidad de convertir los vectores a forma bipolar. Este ejemplo es para el caso autoasociativo.

Sean los patrones de asociación los vectores que se extraen de las matrices tal como se muestra en la Figura 3.8, con el conjunto de vectores a autoasociar X.

Procedemos a aplicar nuestro modelo coevolutivo con 20 generaciones, para la etapa de asociación de 30 individuos cada una, y para la etapa de recuperación también fijamos 20 generaciones de 30 individuos cada una, en un proceso por lotes para la generación de 3 duplas, i.e., 3 individuos-reglas para la asociación y estos 3 individuos colaborarán en la generación de cada una de los 3 individuos-reglas de recuperación. La Tabla 3.1 muestra las duplas en su aptitud conjunta, y vamos a tomar la dupla del ... renglón, éstos individuos se muestran en la Figura 3.9.

Paso 1 Aplicamos la regla de asociación del individuo ganador, asociando cada patrón consigo mismo, y de esa asociación obtenemos la matriz de asociación local: $\mu_i = Op^a(X, X^T)$, para $i = 1, 2, 3$ en este caso.

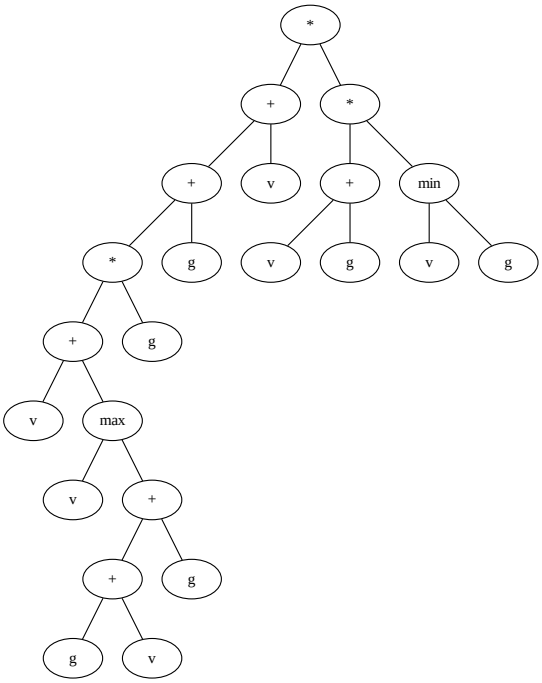


$$X = \begin{bmatrix} 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

Figura 3.8: Patrones de ejemplo para el caso binario, los convertimos a notación bipolar.

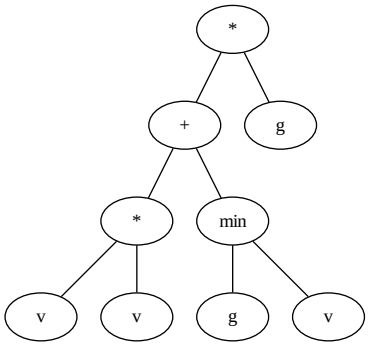
Regla de asociación	Regla de recuperación	Aptitud global
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_1^r	1
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_2^r	1
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_3^r	1

Tabla 3.1: Ejemplo de duplas ganadoras. Las reglas de asociación en competencia (primera columna) con su correspondiente regla de recuperación ganadora (segunda columna), y la aptitud global del proceso con cada par de reglas en cooperación. El operador ganador indexado de la primer columna Op_1^a corresponde al individuo de la Figura 3.9a, y los números 2 y 3 corresponden con los individuos de las Figuras 3.9b y 3.9c; el operador de recuperación de la segunda columna Op_1^r corresponde al individuo mostrado en la Figura 3.10a, y así sucesivamente para los índices 2 y 3.



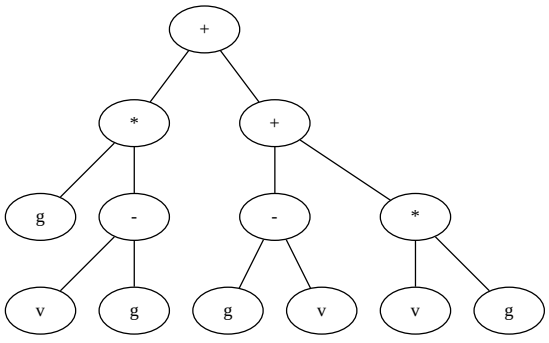
times(plus(plus(plus(times(plus(v,max(v,plus(plus(g,v),g))),g),g),v),
times(plus(v,g),min(v,g)))

(a)



times(plus(times(v,v),min(g,v)),g)

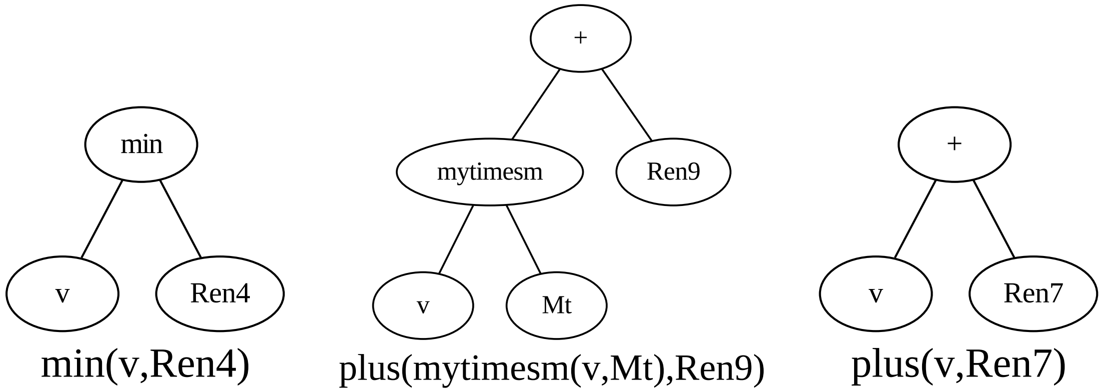
(b)



plus(times(g,minus(v,g)),plus(minus(g,v),times(v,g)))

(c)

Figura 3.9: Ejemplo de individuos coevolucionados. Reglas de asociación.



(a) (b) (c)

Figura 3.10: Ejemplo de individuos coevolucionados. Reglas de asociación.

$$m_1 = \begin{bmatrix} 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} m_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$m_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \end{bmatrix}$$

Paso 2 Creamos la matriz de asociación global $M = \sum m_i$, ésta es nuestra memoria asociativa para este conjunto de patrones.

$$M = \begin{bmatrix} 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \\ 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \end{bmatrix} \tag{3.5}$$

Paso 3 Probamos la recuperación de los patrones, procediendo a operar con la regla de recuperación ganadora, $Op_1^r = min(v, Ren4)$, como se muestra a continuación.

$$min(\begin{bmatrix} 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \end{bmatrix}) =$$

$$\begin{bmatrix} 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$min(\begin{bmatrix} -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \end{bmatrix}) =$$

$$\begin{bmatrix} -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$min(\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}, \begin{bmatrix} 12 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 12 \end{bmatrix}) =$$

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \end{bmatrix}$$

Como vemos, se tiene recuperación perfecta.

Capítulo 4

Resultados experimentales

En este capítulo se presentan los resultados de la metodología propuesta en el Capítulo 3, con sus particularidades y análisis de los resultados obtenidos.

4.1. Resultados y análisis del primer modelo

Este modelo puede ser considerado como un modelo preliminar de una MA. En su modelación se puso en práctica la PG como un primer ejercicio, analizando el comportamiento evolutivo de los tres conjuntos que delimitan el proceso: el conjunto de terminales (T), el conjunto de funciones (F), y la función de aptitud (f) apropiada para los procesos de selección. Como se explicó en el Capítulo 3, en este modelo se consideran como nodos de nuestros árboles a todo el vector, con todas sus componentes.

Tras varias corridas en lotes de 20 ejecuciones de nuestro programa con las condiciones descritas en la sección anterior, se llegó al individuo mostrado en la Figura 4.1, equivalente a $(-, X_i, X_i) \rightarrow X_i - X_i$, con un valor de aptitud mínimo (que es lo que buscamos para este modelo), de 0.2778. Este resultado concuerda con la idea básica planteada acerca de una MA, que consiste en establecer una relación de asociación entre dos o más patrones, en este caso el individuo generado nos devuelve la más básica de ellas: la diferencia.

Considerando el conjunto de funciones usado y los parámetros de la PG vemos que ésta es una solución con la que se agotan las posibilidades de este planteamiento, la efectividad del 72 % es significativa considerando todas las posibilidades explotadas por la PG en este proceso donde unicamente se está considerando al vector como entidad de entrada, con todas sus entradas, por lo que todas las operaciones del conjunto F afectan de forma binaria al vector, se suman o se restan, o se minimiza o se maximiza, pero teniendo siempre como resultado

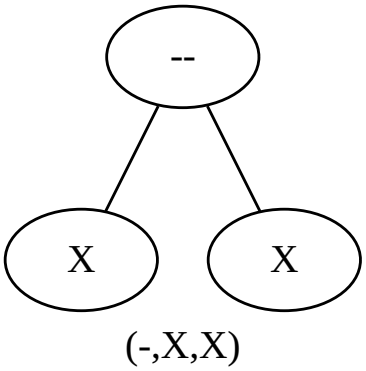


Figura 4.1: Individuo evolucionado a partir de PG, con $aptitud = 0{,}2778$.

un vector de la misma dimensión, que es una limitante de esta preliminar aproximación, se tienen vectores de la misma dimensión en la asociación. Por todo lo anterior, no se tiene la suficiente holgura para tener una mejor aproximación.

4.2. Resultados y análisis del segundo modelo

Para este segundo modelo, se programaron nuevos lotes de corridas con el valor de aptitud apegandonos al estándar de $(0 \leq f \leq 1)$, siendo $f = 1$ cuando hay una recuperación perfecta, y $f = 0$ cuando es nula la recuperación.

Los parámetros de inicialización de la PG se dejaron iguales que en el primer modelo, y ahora los lotes fueron de 50 generaciones con 70 individuos cada una, y se programaron en tres diferentes modalidades, con la restricción de ahora sólo tratar vectores binarios. Las tres modalidades en que se probo este modelo son:

- Evolucionar individuos a partir de un conjunto de patrones fijo, no precisamente ortogonal, de forma heteroasociativo.
- Evolucionar individuos a partir de conjuntos de patrones ortogonales, de forma autoasociativa.
- Evolucionar individuos a partir de conjuntos de patrones ortogonales, de forma heteroasociativa.

La primera modalidad fue el evolucionar una MA para un conjunto de patrones generado

de forma aleatoria, no precisamente ortogonales[1], como los que se muestran a continuación:

$$X = \begin{bmatrix} -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}, \; Y = \begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

con X y Y los conjuntos de patrones origen y destino, asociados de la misma forma ya descrita, cada línea de cada matriz se considera un vector, $x_1 = [-1 \; -1 \; -1 \; 1]$, que es asociado con $y_1 = [1 \; -1 \; 1 \; 1]$, y sucesivamente x_2 con y_2, etc. Ahora buscamos una MA que nos muestre la forma de asociación para estos conjuntos de patrones.

Tras 20 lotes de corridas se obtuvieron varios individuos, de los cuales en forma representativa, todos con *aptitud* $= 1$, se escogieron los mostrados en la Figura 4.2, y se obtuvo también como operador evolucionado el modelo clásico (Figura 4.2c).

La segunda modalidad en que aplicamos la evolución de operadores fue para conjuntos de patrones ortogonales, en forma autoasociativa. En la Figura 4.3 se muestran ejemplificados tres candidatos con $aptitud = 1$. De la tercera modalidad se muestran tres individuos seleccionados en la Figura 4.5.

Posteriormente de todos los individuos resultantes con $aptitud = 1$, para cada modalidad, se probaron para analizar su robustez ante patrones desconocidos para los casos de patrones generados de forma aleatoria y ortogonales, tanto para el caso auto-asociativo como heteroasociativo. Los vectores generados se hicieron partiendo de 2 vectores de 2 entradas, relacionados con otros 2 vectores de 2 entradas, y asi sucesivamente hasta tener 100 vectores de 100 entradas. Los resultados de estas pruebas se muestran en las Figuras [4.6, 4.7, y 4.8].

4.3. Resultados y análisis del tercer modelo

En esta sección se muestran los experimentos hechos con patrones binarios y reales con el tercer modelo desarrollado.

4.3.1. Experimentos con patrones binarios

Se aplicó la metodología del modelo co-evolutivo propuesto, en una primera etapa de prueba para vectores con valores binarios sencillos de matrices representando digitos del 0 al 9, como se muestran en la Figura 4.9. Cada matriz es de dimensión 7×5, y se convierte a

[1] Definición: dos vectores son ortogonales en un espacio vectorial V si su producto interno $<x, y>$ es cero. Esto se denota como $x \perp y$.

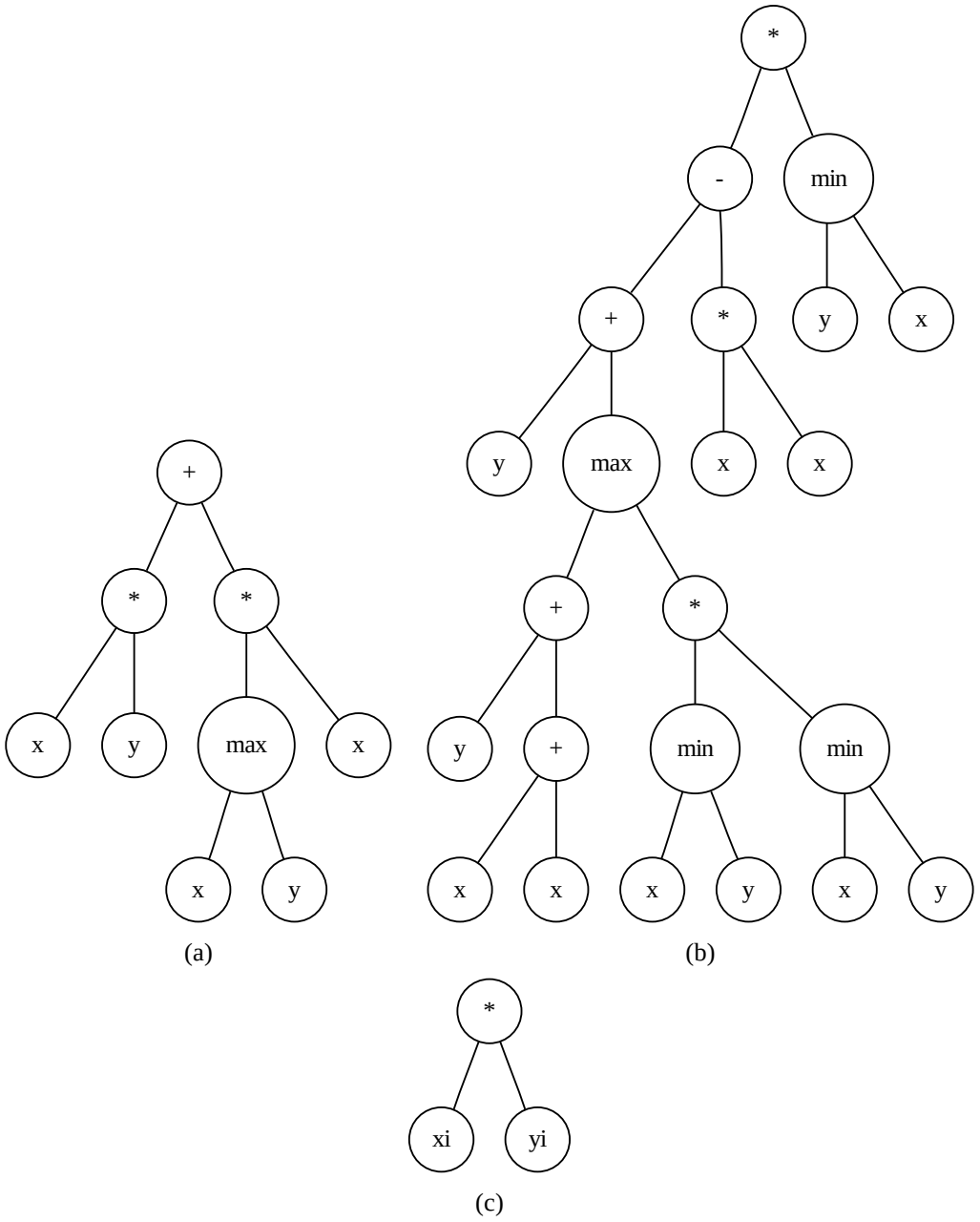


Figura 4.2: Individuos sintetizados usando PG, para un conjunto de patrones aleatorios, no precisamente ortogonales.

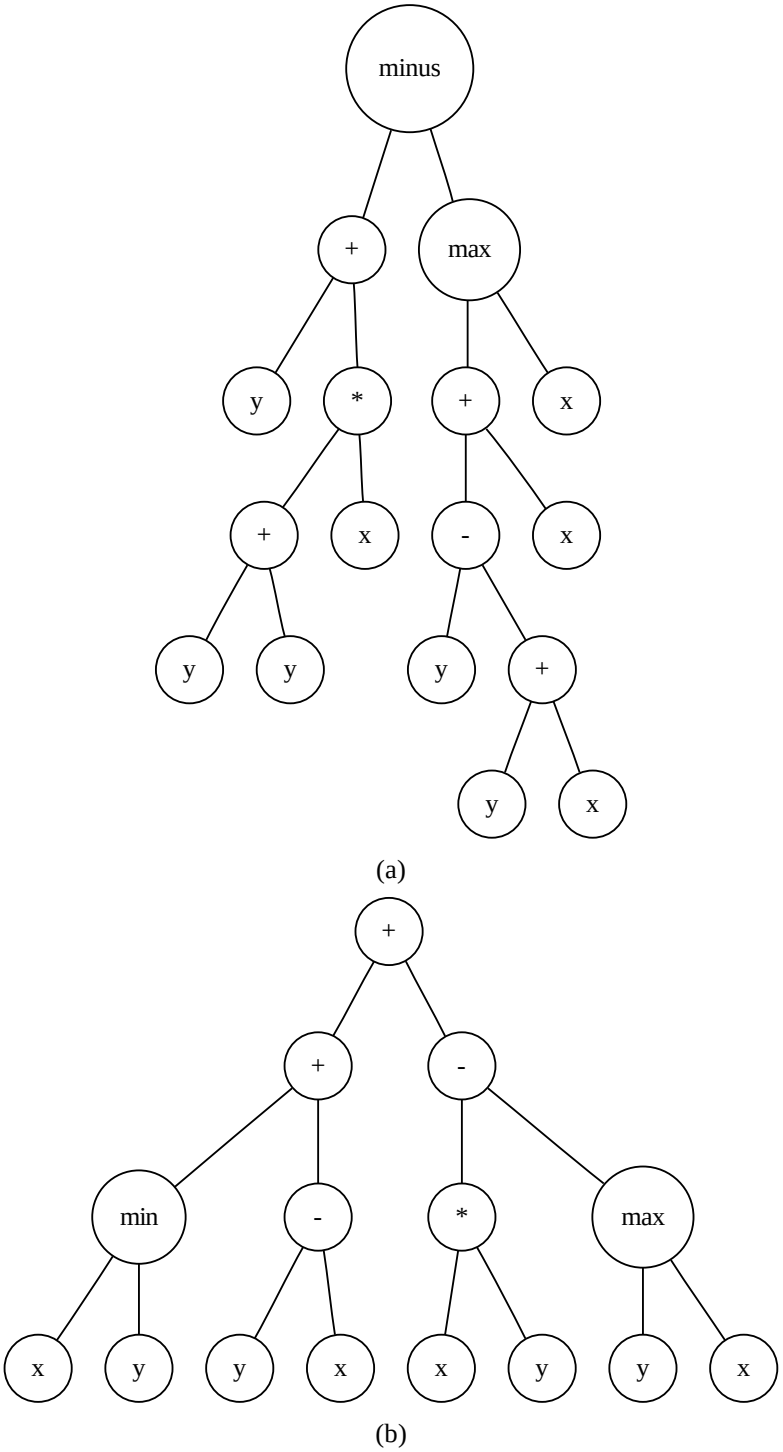


Figura 4.3: Dos primeros individuos evolucionados a partir de patrones generados de forma ortogonal, bajo relación auto-asociativa.

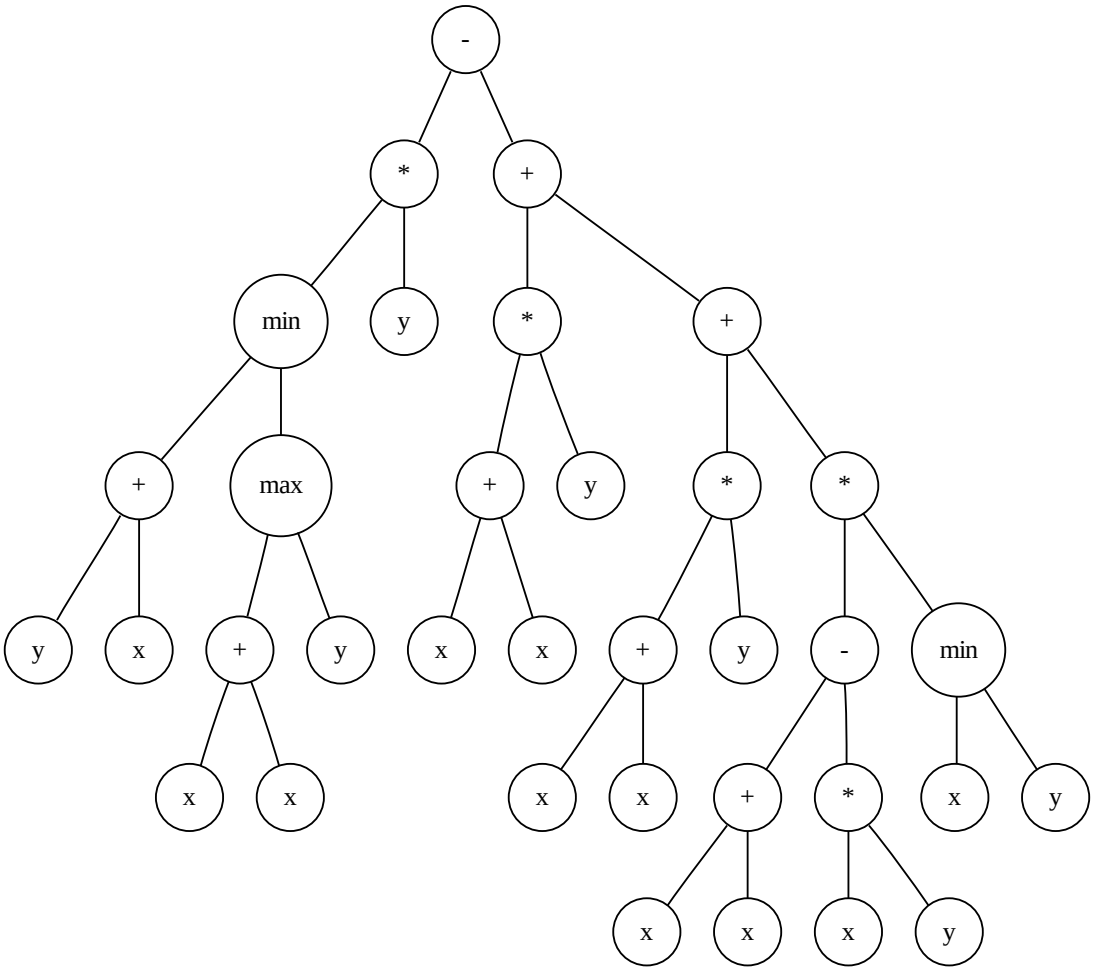


Figura 4.4: Tercer individuo evolucionado a partir de patrones generados de forma ortogonal, bajo relación auto-asociativa.

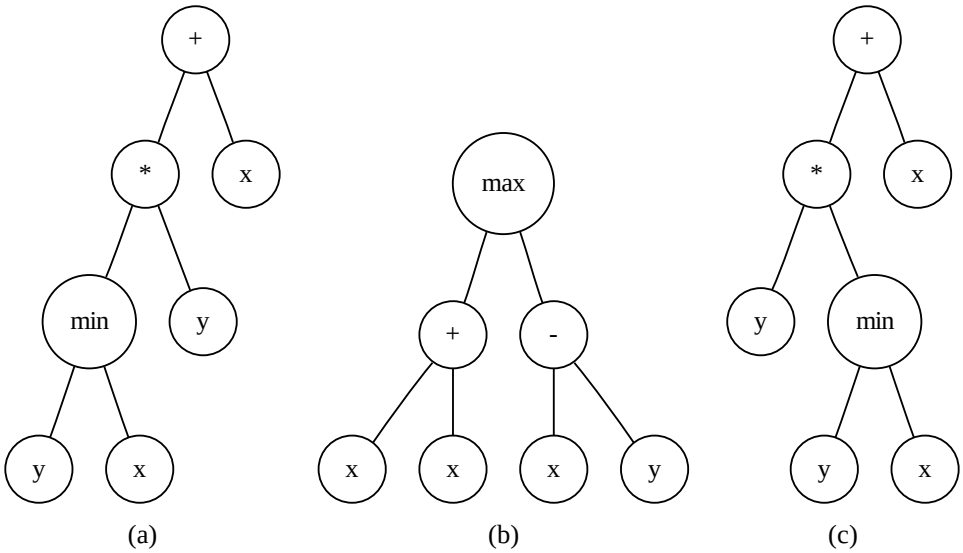


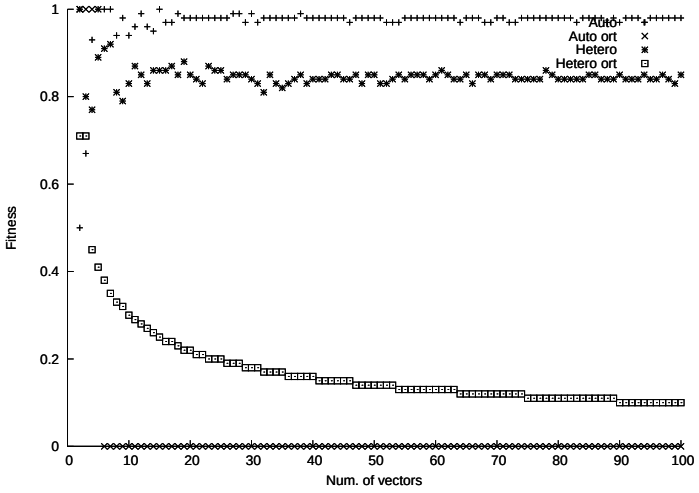
Figura 4.5: Individuos evolucionados a partir de patrones generados de forma ortogonal, bajo relación hetero-asociativa.

matriz-vector quedando de dimensión 1×35, y como se trabajaron los valores en los modelos primero y segundo ahora también se convierten a notación bipolar para minimizar el impacto de operaciones con el valor cero en la multiplicación. Para este conjunto sencillo de patrones se experimentó para el caso auto-asociativo ($X = Y$).

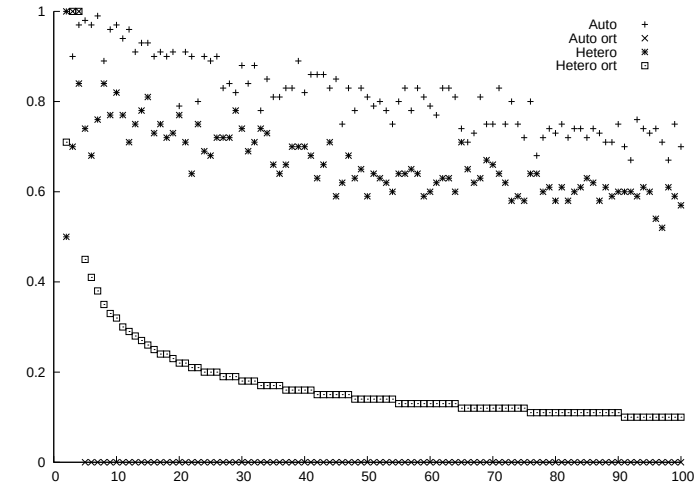
La aplicación de nuestra metodología generó un conjunto de reglas-operadores para la etapa de asociación, y otro conjunto de reglas-operadores para la etapa de recuperación; como se describió en el detalle del modelo, ambos conjuntos de reglas entran en competencia quedando duplas ganadoras. Para este conjunto de patrones una dupla ganadora se muestra en la Figura 4.10. La dupla ganadora mostrada conforma la MA. Está es solo un ejemplo de una vasta producción de reglas que se tiene en el proceso por lotes llevado a cabo, y cada dupla generada, con diferentes grados de complejidad, es una MA que se ajusta perfectamente a este conjunto de patrones.

La solución mostrada en la Figura 4.10 muestra una característica importante para este conjunto de patrones en particular, véase que en la regla de recuperación 4.10b el rengón 7 ($ren7 \in M_k$), es el más significativo, i.e. donde recae el almacenamaniento de la memoria asociativa creada con el operador de asociación 4.10a en un proceso auto-asociativo.

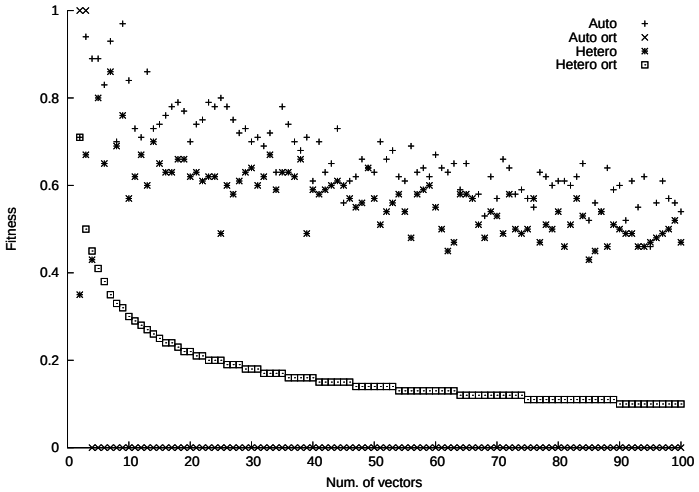
Se experimentó con la MA mostrada en la Figura 4.10 ante el ruido mixto (sal y pimienta) para analizar su robustez. El ruido se aplicó en escala de 0.1 % a 1.0 %, como se muestra en la Figura 4.11. Con agrado se obtuvo una recuperación del 100 %, i.e., cero error, es así que nuestra MA generada mediante PG tiene un mejor comportamiento en comparación con las



(a)

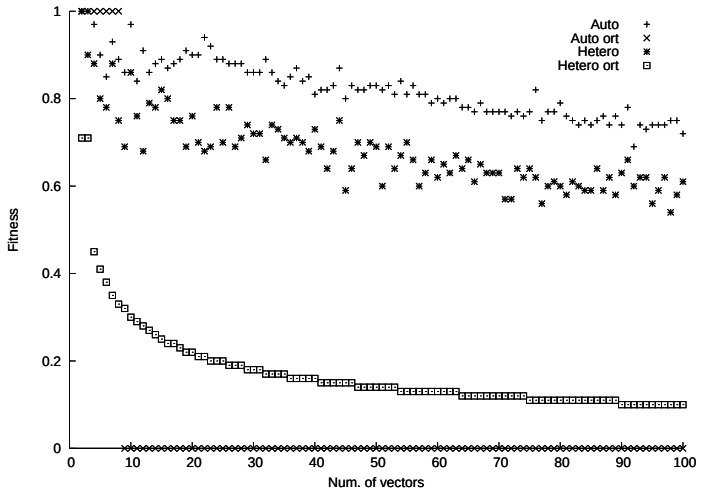


(b)

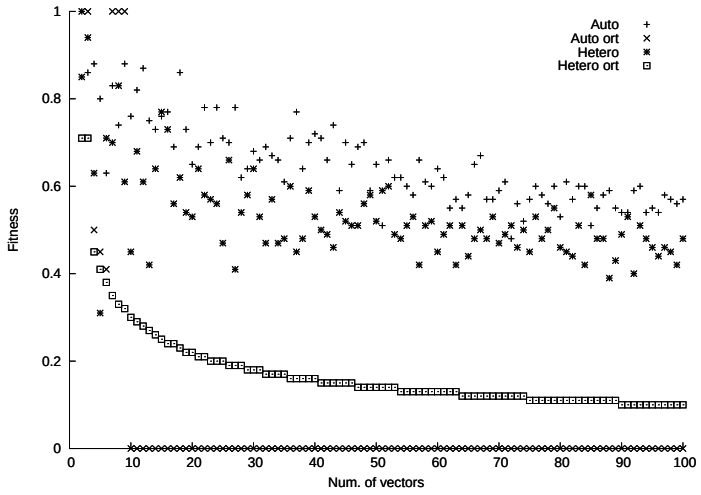


(c)

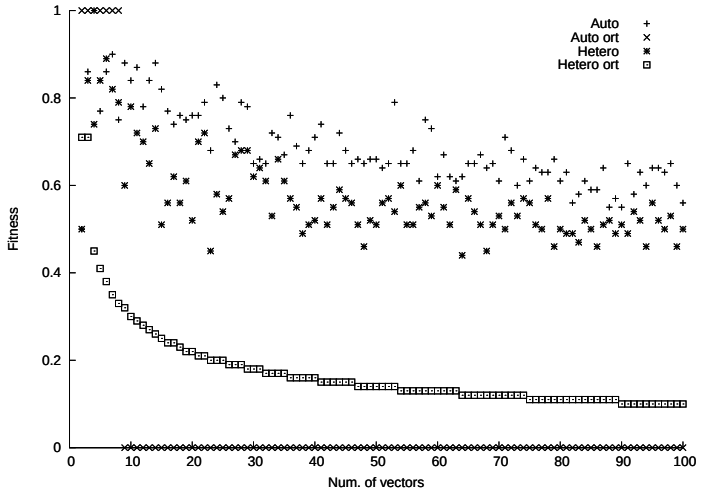
Figura 4.6: Prueba de desempeño de individuos evolucionados a partir de patrones aleatorios, no precisamente ortogonales, en relación hetero-asociativa.



(a)

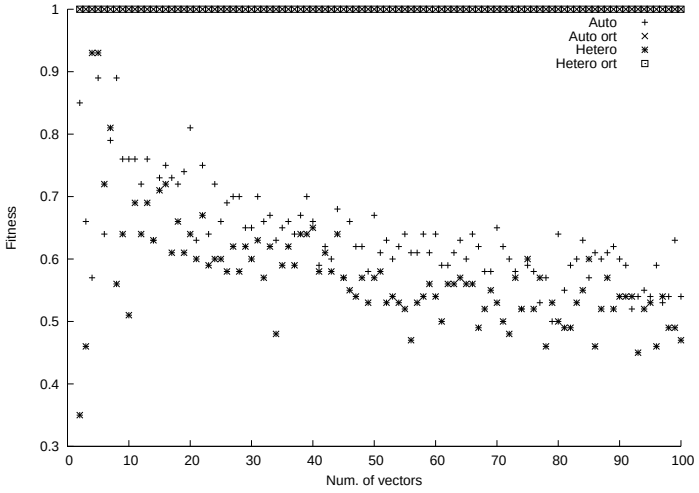


(b)

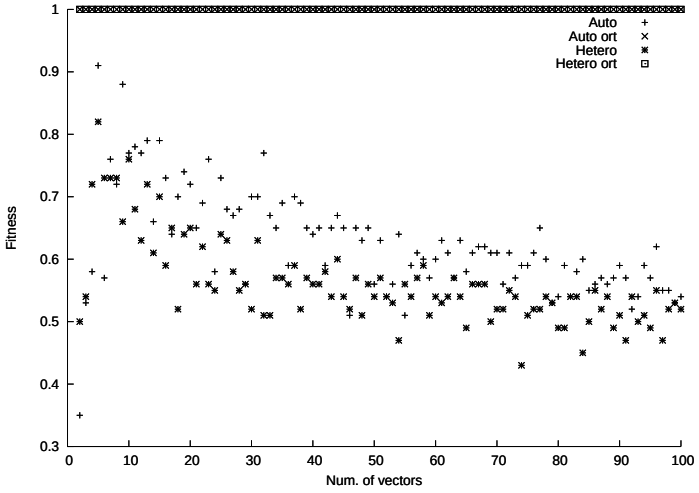


(c)

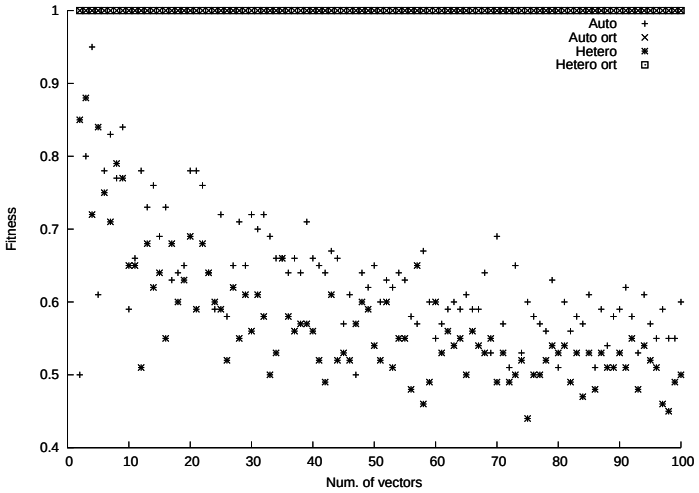
Figura 4.7: Prueba de desempeño de individuos evolucionados a partir de patrones ortogonales, en relación auto-asociativa.



(a)



(b)



(c)

Figura 4.8: Prueba de desempeño de individuos evolucionados a partir de patrones ortogonales, en relación hetero-asociativa.

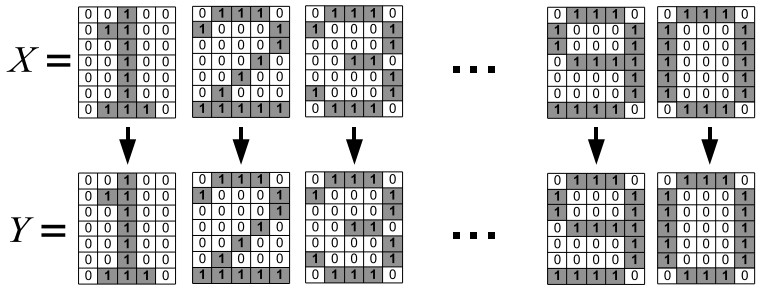


Figura 4.9: Matrices representando digitos para el primer experimento de generación de MA con patrones binarios.

memoras morfológicas [25] y las $\alpha\beta$ [41], que son robustas sólo a ruido aditivo o sustractivo.

4.3.2. Experimentos con patrones en valores reales

Considerando estos excelentes resultados arrojados con la metodología mejorada se decidió experimentar con patrones más complejos, en este caso con patrones en valores reales. Se optó por tomar una base de datos de un problema ampliamente conocido, el problema de la clasificación de la planta del Iris [68], con las caracterísitcas siguientes:

- 150 instancias.
- 4 atributos: longitud de sépalo, ancho de sépalo, longitud de pétalo, ancho de pétalo (en cm.).
- 3 clases: iris setosa, iris versicolor, iris virginica.

Las MA obtenidas para este problema, con sus patrones en valores reales, se muestran en las Figuras 4.12 y 4.13 las reglas de asociación, y en las Figuras 4.3.2 y 4.15 las reglas de recuperación. Todos los experimentos fueron bajo autoasociación.

La metodología la aplicamos en un proceso por lotes para obtener tres candidatos-reglas para ser usados como reglas de asociación, y posteriormente en la etapa de recuperación, cada una de las reglas de asociación entra en competencia con cada posible candidato-regla de recuperación. De esta forma es como se co-evolucionan de forma cooperativa las reglas para finalmente obtener las duplas ganadoras al final del proceso. Cada individuo tiene su etiqueta acorde al operador evolucionado de cada etapa del proceso. En la Tabla 4.1 se muestran de cada competencia en la etapa de asociación, cual de los tres operadores gana en conjunto con su respectiva regla-operador de recuperación, y se muestra el fitness global del proceso en la tercer columna de la tabla.

Regla de asociación	Regla de recuperación	Aptitud global
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_1^r	1
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_2^r	1
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_3^r	1

Tabla 4.1: Duplas ganadoras. Las reglas de asociación en competencia (primera columna) con su correspondiente regla de recuperación ganadora (segunda columna), y la aptitud global del proceso con cada par de reglas en cooperación. El operador ganador indexado de la primer columna Op_1^a corresponde al individuo de la Figura 4.12, y los números 2 y 3 corresponden con los individuos de las Figuras 4.13a y 4.13b; el operador de recuperación de la segunda columna Op_1^r corresponde al individuo mostrado en la Figura 4.3.2, y así sucesivamente.

Analizando las Figuras 4.13 y 4.15, junto a la Tabla 4.1, se puede observar que el proceso co-evolutivo proporciona soluciones diversas, desde reglas complejas (Figura 4.13b), hasta reglas sencillas para la recuperación (Figura 4.15b). Según los datos de la Tabla 4.1, las reglas de asociación de las Figuras 4.13a y 4.13b, no son lo suficientemente buenas, i.e. su forma de asociar los patrones no es la adecuada para lograr una recuperación perfecta con la regla de recuperación en turno, que para este experimento el primer individuo (regla de asociación Op_1^a) es el que gana en todos los procesos conjuntos.

Otro aspecto importante que se revela a través de esta metodología son las principales conexiones o sinapsis que entran en el juego de la competencia. Paa este caso de patrones de la planta Iris, las reglas de recuperación ganadoras muestran que los renglones 1, 2 y 4, pertenecientes a la matriz de asociación (M_k) del operador (Op_i^a) correspondiente a cada caso, son los datos más significativos para lograr una recuperación perfecta. Estos renglones corresponden a las características de la planta: (1) longitud del sépalo, (2) ancho del sépalo y (4) ancho del pétalo. Es así que estas nuevas MA muestran que sólo son relevantes tres características de las cuatro.

Finalmente se probaron estas tres MA ante ruido aleatorio. Dada la naturaleza del tipo de patrones con las que fueron generadas, se probó aplicando pequeñas perturbaciones de ruido aleatorio a los valores, en proporciones desde 0 hasta 0.009 %. Los resultados se muestran en la Figura 4.16. Se puede observar que la tasa de recuperación decrece lentamente conforme el ruido aumenta, y este comportamiento es perceptiblemente el mismo para las tres MA.

Adicionalmente se probó la metodología con la base de datos de clasificación de vinos [68], con características multivariadas y un mayor número de atributos:

- 178 instancias.

- 13 atributos: (1) acidez fija, (2) acidez volatil, (3) acidez cítrica, (4) residuos de azúcar, (5) cloruros, (6) dioxido de sulfuro libre, (7) dioxido de sulfuro total, (8) densidad, (9) pH, (10) sulfatos, (11) alcohol, (12) puntaje de calidad (entre 0 y 10), (13) proline.

- 3 clases: 59 instancias para clase 1, 71 instancias para clase 2, 48 instancias para clase 3.

Esta base de datos es interesante porque proporciona caracterísitcas de variantes de vinos dentro de cada clase (tinto y blanco), son proporcionadas en orden de cata por expertos enólogos, pero no están balanceadas bajo algún criterio (e.g. hay más datos de vinos regulares que excelentes o con puntaje de calidad bajo).

Las MA obtenidas para este problema, con sus patrones en valores reales, se muestran en las Figuras 4.18 y 4.19. Todos los experimentos fueron bajo autoasociación.

Regla de recuperación ganadora en competencia:

Figura (4.19) :

min(plus(v,v),mytimes(v,Mk))

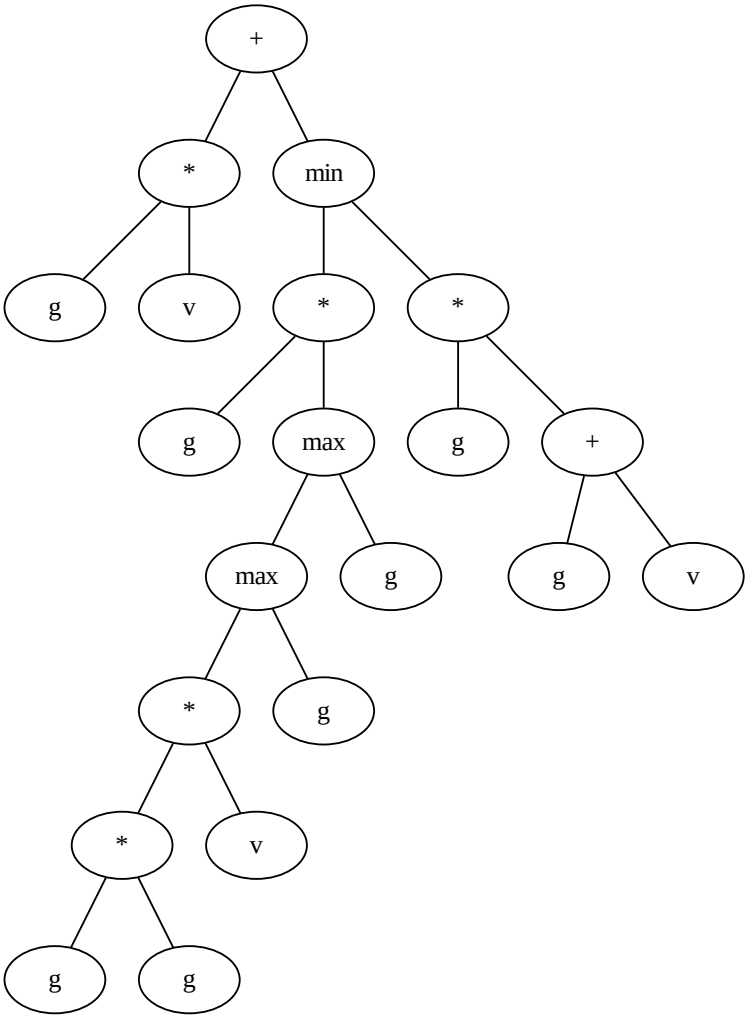
En la Tabla 4.2 se muestra la competencia de los operadores en co-evolución para el caso de la base de datos del vino.

Analizando los resultados para esta base de datos, vemos que no hay una fuerte relevancia por parte de cada característica de los patrones. i.e. en la recuperación se involucra al vector de entrada y a toda la matriz de asociación formada por el operador de asociación Op_1^a.

La complejidad del operador de asociación (véase la Figura(4.18a)) es compensada con la simplicidad del operador para la recuperación del patrón, pero más aún, el conjunto de operaciones permanece simple y de rápido cálculo.

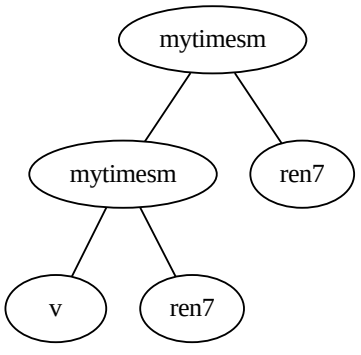
Regla de asociación	Regla de recuperación	Aptitud global
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_1^r	< 1
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_1^a$	Op_2^r	1
$Competencia(Op_1^a, Op_2^a, Op_3^a) := Op_2^a$	Op_1^r	< 1

Tabla 4.2: Duplas ganadoras. Las reglas de asociación en competencia (primera columna) con su correspondiente regla de recuperación ganadora (segunda columna), y la aptitud global del proceso con cada par de reglas en cooperación. El operador ganador indexado de la primer columna Op_1^a corresponde al individuo de la Figura 4.3.2, y los números 2 y 3 corresponden con los individuos de las Figuras 4.18a y 4.18b; el operador de recuperación de la segunda columna Op_1^r corresponde al individuo mostrado en la Figura 4.19, que para este caso fue la única regla de asociación que ganó.



g*v + min (g * max (max (g^2 *v, g), g), g*(g+v))

(a)



(v mytimesm ren7) mytimesm ren7

(b)

Figura 4.10: MA evolucionada para el caso auto-asociativo. La regla de asociación ganadora es 4.10a, y 4.10b es la regla para la recuperación de los patrones.

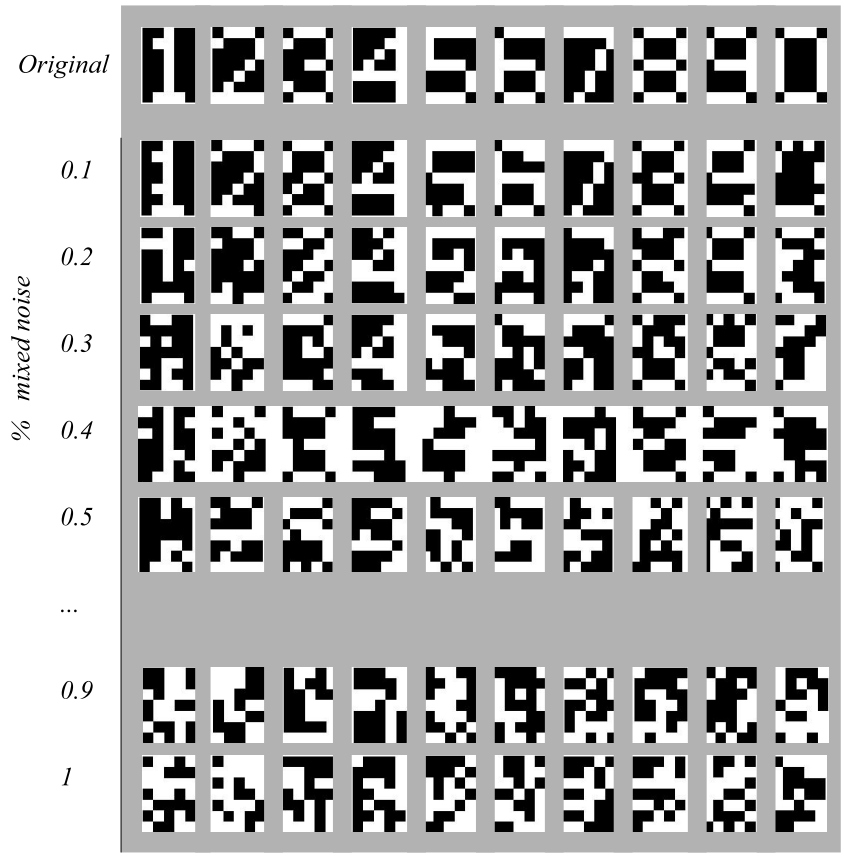


Figura 4.11: Matrices representando digitos. La primer línea sin ruido, y progresivamente en aumento el porcentaje de ruido mixto.

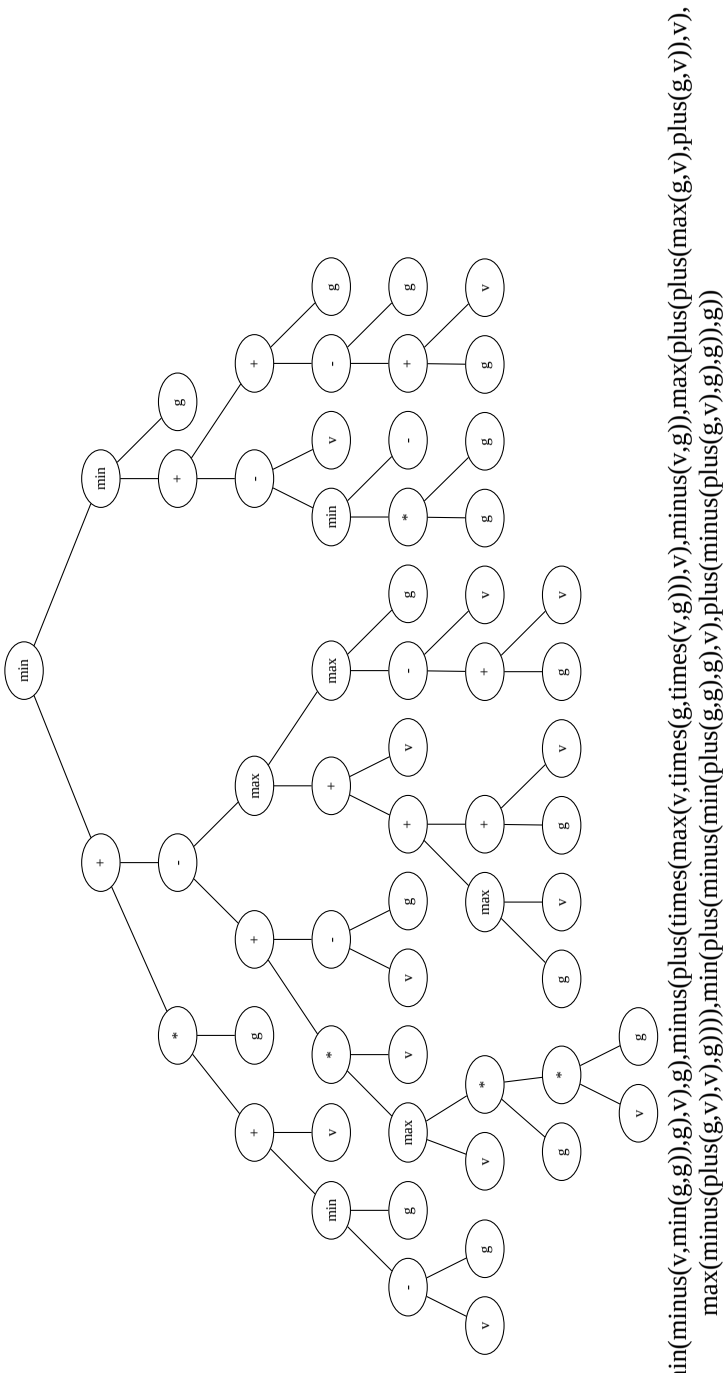
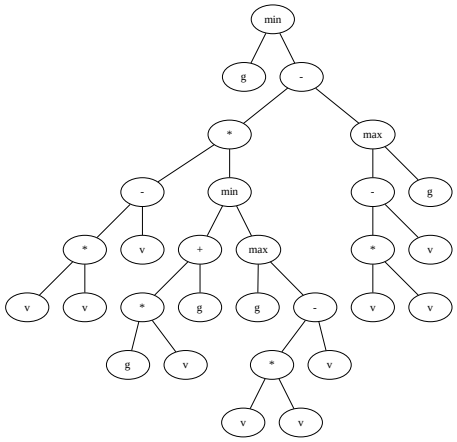
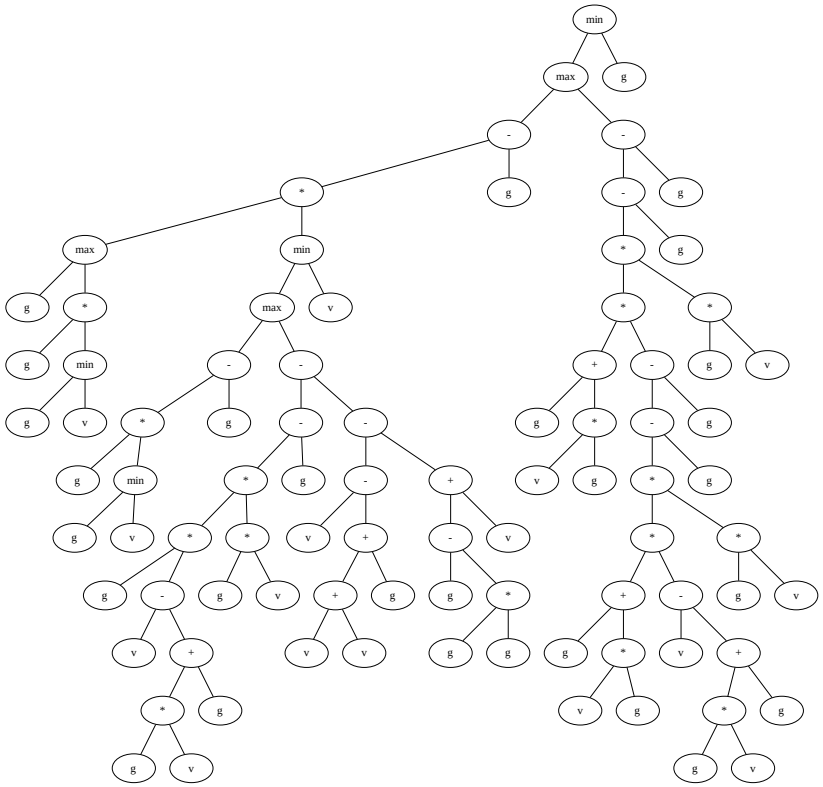


Figura 4.12: Regla de MA evolucionada para la base de datos de la planta Iris.



min[g, (v*v -v)*min{max(v*v -v, g), g*v +g } - max(v*v -v, g)]

(a)



min(max(-(*(max(g,*(g,min(g,v))),min(max(-(*(g,min(g,v)),g),-(-(*(*(g,-(v,+(*(g,v),g))),
(g,v)),g),-(-(v,+(+(v,v),g)),+(-(g,(g,g)),v)))),v)),g),-(-(*(*(+(g,*(v,g)),-(-(*(*(+(g,*(v,g)), -(v,+(*(g,v),g))),*(g,v)),g),g)),*(g,v)),g),g)),g)

(b)

Figura 4.13: Reglas de MA evolucionadas para la base de datos de la planta del Iris. Estas son dos reglas diferentes para la asociación de patrones, con v y g como las componentes de los patrones de entrada y salida respectivamente.

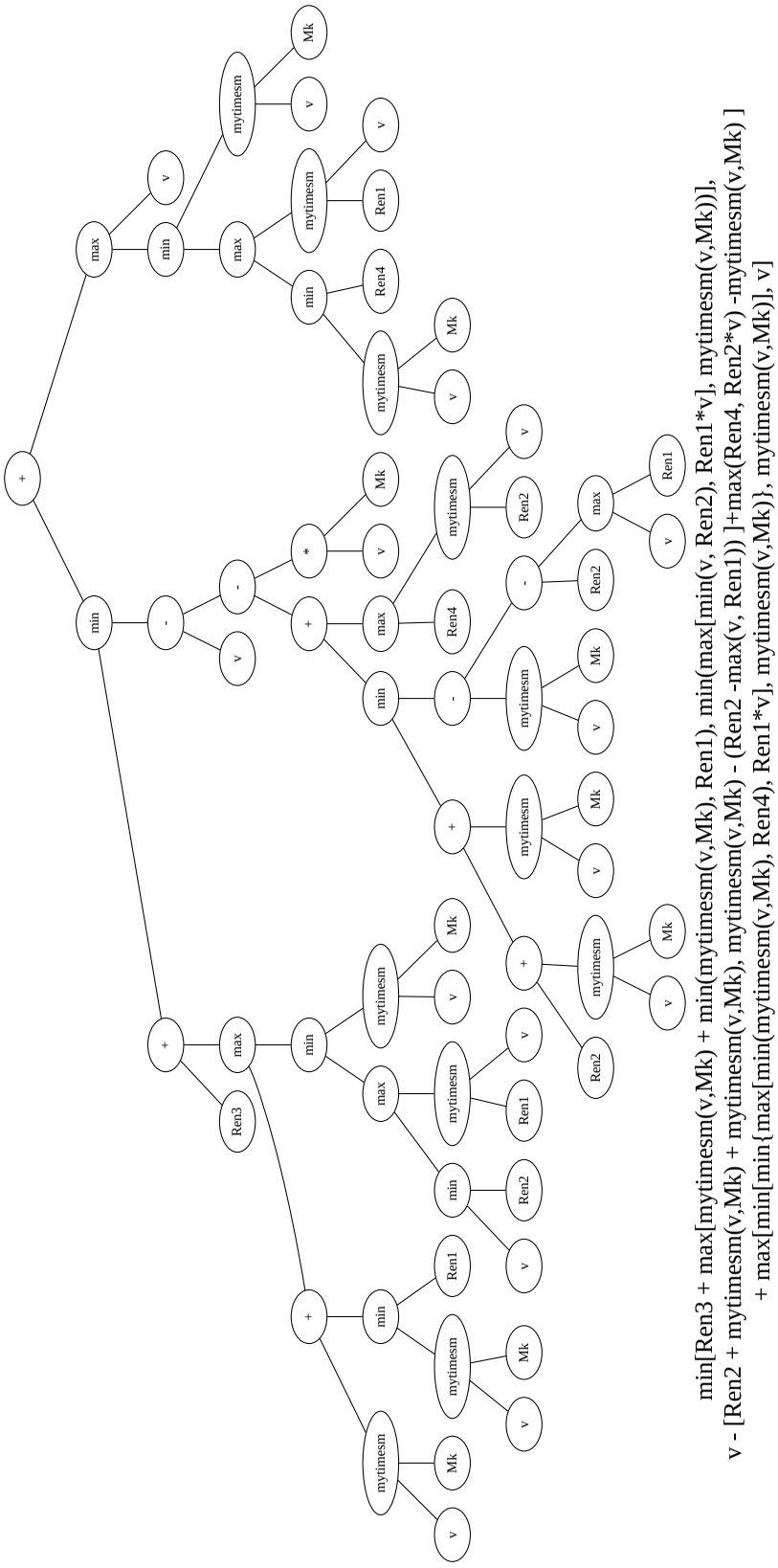
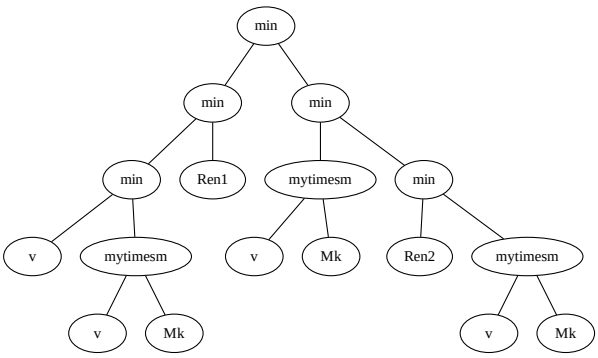
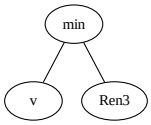


Figura 4.14: Regla para la recuperación, caso de planta Iris, con v como el vector de entrada, y M_k la matriz de asociación.



min[min(min{v, mytimesm(v,Mk)},ren1), min(mytimesm(v,Mk), min{Ren2, mytimesm(v,Mk)})]

(a)



min(v, Ren3)

(b)

Figura 4.15: MA evolucionadas para la base de datos de la planta del Iris. Estas son tres reglas diferentes para la recuperación, con v como el vector de entrada, y M_k la matriz de asociación.

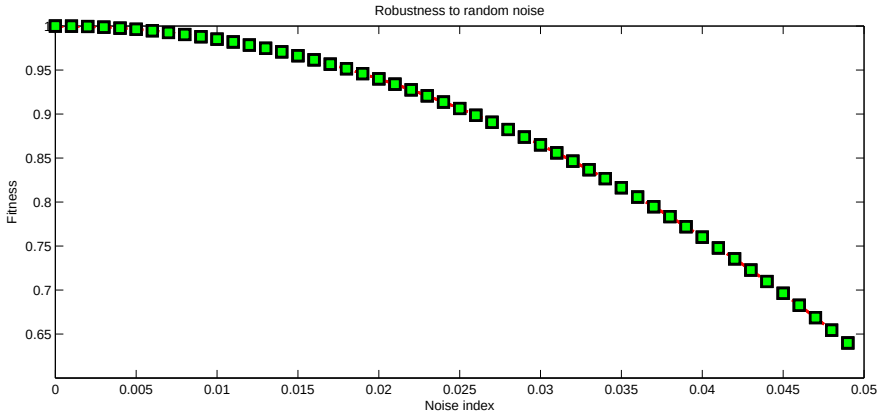


Figura 4.16: Prueba de robustes de las tres nuevas MA generadas por medio de PG. Los patrones de la base de datos se alteraron con ruido aleatorio.

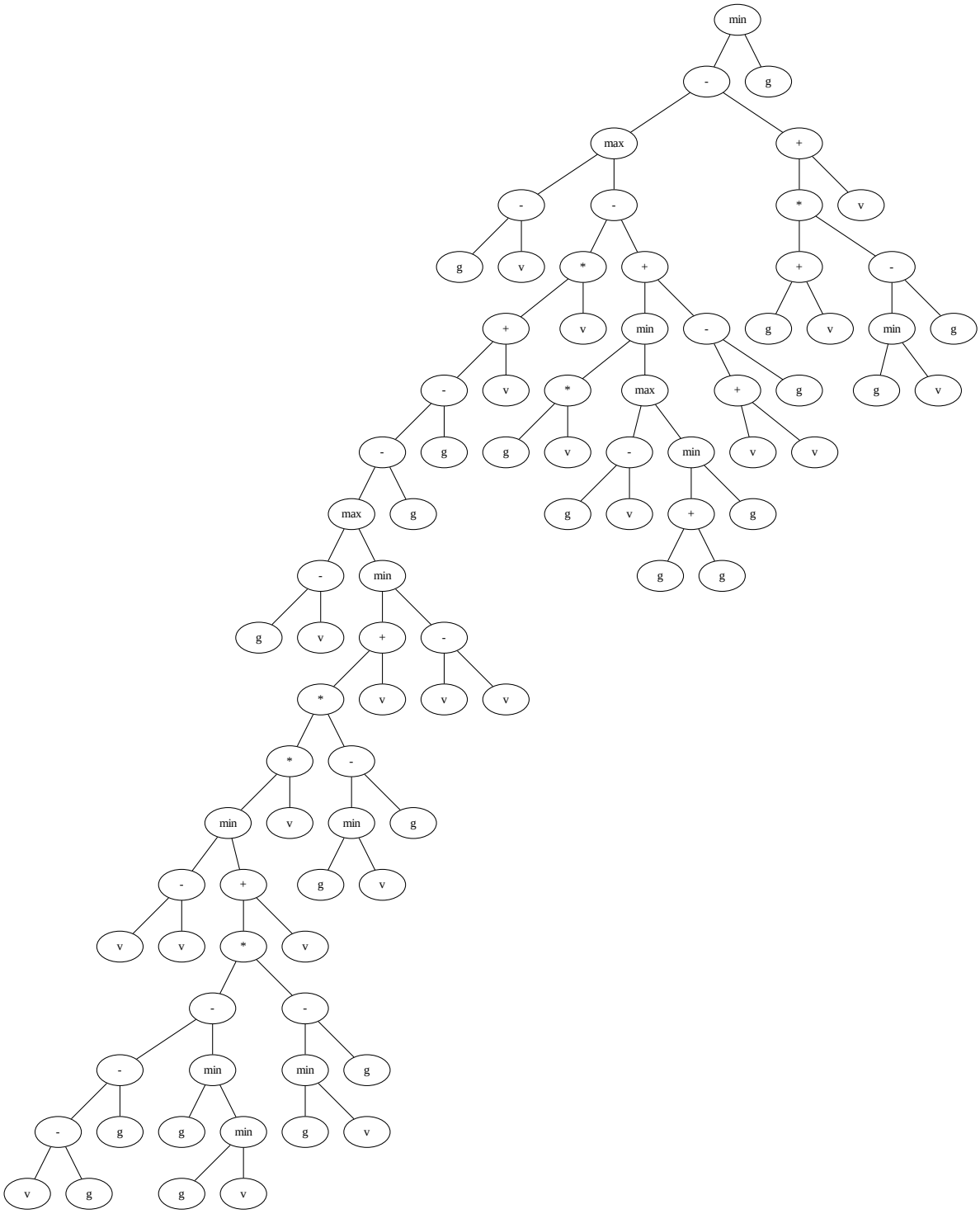
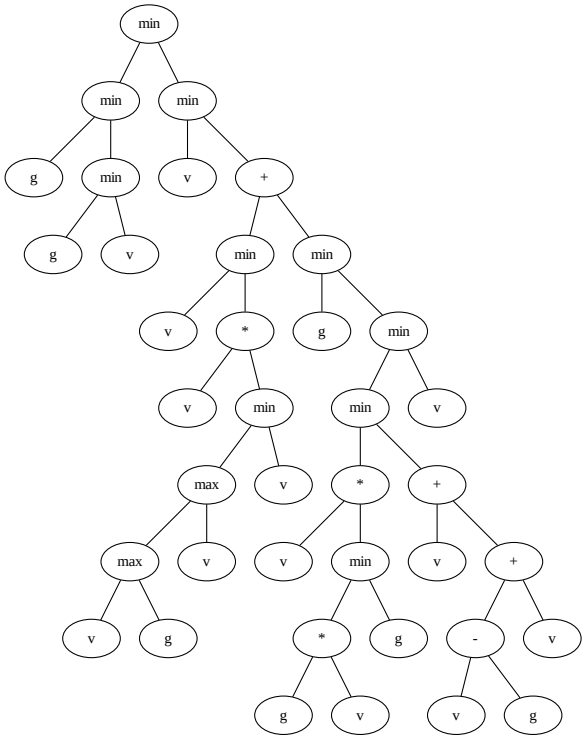
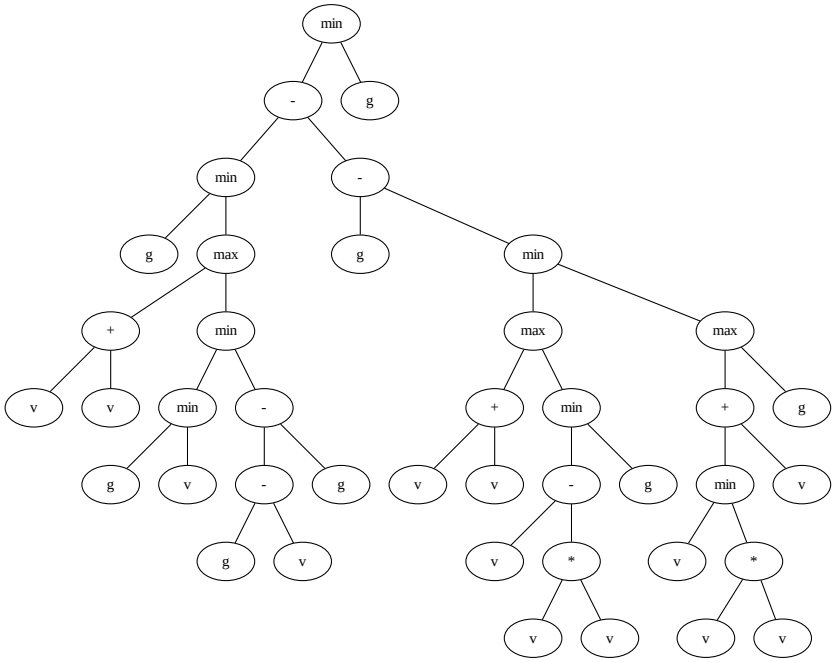


Figura 4.17: Regla-individuo evolucionado como operador de asociación de patrones de la base de datos de vinos.



min(min(g,min(g,v)),min(v,plus(min(v,times(v,min(max(max(v,g),v),v))),
min(g,min(min(times(v,min(times(g,v),g)),plus(v,plus(minus(v,g),v))),v)))))

(a)



min(minus(min(g,max(plus(v,v), min(min(g,v),minus(minus(g,v),g)))),minus(g,min(max(plus(v,v),
min(minus(v,times(v,v)),g)),max(plus(min(v,times(v,v)),v),g)))),g)

(b)

Figura 4.18: Reglas-individuos evolucionados como operadores de asociación de patrones de la base de datos de vinos.

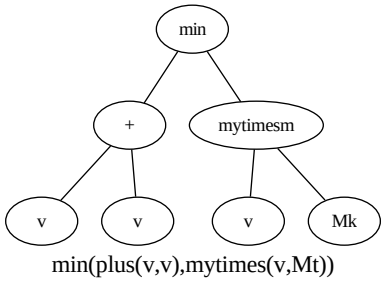


Figura 4.19: Regla-individuo ganador evolucionado como operador de recuperación de patrones de la base de datos de vinos.

Capítulo 5

Conclusiones y perspectivas

En este capítulo se presentan, por una parte, las conclusiones a las que se llegó a lo lardo del desarrollo de la tesis. Por otro lado, se enlistan los productos obtenidos hasta el momento. Finalmente, se enuncian las perspectivas de trabajo y procesos a desarrollar en un futuro próximo como resultado inmediato de esta investigación.

5.1. Conclusiones

En esta tesis el objetivo planteado se ha podido llevar a cabo. Se puede concluir con los resultados mostrados hasta este momento que sí es posible la generación automática (síntesis) de memorias asociativas (MA) mediante la programación genética (PG).

Se partió de la idea de analizar en la literatura el estado del arte de las MA, y dos de los motivos más influyentes para esta investigación han sido, por un lado, la sencillez de operación de las MA con respecto los modelos más complejos de las redes neuronales artificiales (RNA) y, por otro lado, el evidente éxito mostrado de la implementación de la PG en otros campos.

Las nuevas MA logradas por medio de la metodología aquí presentada nos permiten tener un mejor conocimiento de los patrones que se estudian en la asociación, gracias a la naturaleza de las MA como ente que nos muestra un mapa de distribución matricial-numérico de cómo es que se asocian dos conceptos-patrones, esto como idea básica del proceso del aprendizaje en los seres vivos, que aprendemos por asociaciones.

Con esta metodología de generación automática se puede tener en muy corto tiempo un análisis, prácticamente inmediato, de cuales características de los patrones son los más relevantes, que en una primera impresión sólo se puede decir “para el proceso de la asociación”,

pero en aplicaciones reales o de estudio de la naturaleza de los entes que representan las bases de datos usadas, se puede ver cuales características son relevantes y cuales no, y poder así prescindir de éstas últimas, las no relevantes, para el cálculo en aplicaciones que involucren esas bases de datos estudiadas.

Las solucuiones proporcionadas por la metodología presentada son resultado de una exploración exhaustiva en los espacios de búsqueda, que gracias al poder de cómputo actual, es posible el poder generar nuevas MA y de ellas escoger las menos complejas. Se puede generar tantos modelos como sea necesario en muy corto tiempo, en cuestión de horas, en lugar de un mínimo de años que han llevado desarrollos manuales de varios de los modelos revisados (véase el Capítulo 2).

5.2. Productos

Como rusultado de esta investigación, hasta el momento, se ha obtenido:

- Una publicación en las memorias de un congreso, donde se describe el primer modelo obtenido mediante la primera metodología derivada de esta tesis [84].
- La publicación de un artículo arbitrado en [94] donde se muestra un resumen de los resultados alcanzados con el segundo modelo, presentado en sesión de poster en el congreso CORE09, el cual fue aceptado para su publicación en el número especial *Research in Computing Science*.
- Un artículo arbitrado con los resultados del tercer modelo, sometido a la *Revista Mexicana de Física*, publicación indexada, mismo que está en proceso de revisión.
- Un artículo sometido al congreso internacional *EvoStar 2010*, donde se muestran los resultados de MA hetero-asociativas, con patrones en valores reales y con otra medida de aptitud. De ser aceptado el trabajo será publicado en la serie LNCS de Ed. Springer.

5.3. Perspectivas

Actualmente se esta experimentando el refinamiento de la metodología con la implementación de nuevas medidas de aptitud, buscando reducir el tiempo de convergencia, sin sacrificar la sencillez de los operadores de las MA. También se está analizando el comportamiento

del modelo de asociación hetero-asociativa con experimentos en valores reales enfocados a tareas de clasificación.

Un producto que esta contemplado es la creación de una aplicación (software) para la automatización de la generación de las MA, el cual en su interfase gráfica permita la operación sencilla de carga de la base de datos a usar, el tipo de asociación requerida, la cantidad mínima de candidatos a generar para la competencia del proceso coevolutivo (por los experimentos hasta este momento se puede establecer un mínimo de tres individuos por etapa para tener un buen nivel de cooperación).

Bibliografía

[1] N. Aall Barricelli. Esempi numerici di processi di evoluzione. *Methodos*, pages 45–68, 1954.

[2] K Steinbuch. Die lernmatrix. *Biological Cybernetics*, 1(1):36–45, Jan. 1961.

[3] J. A. Anderson. A memory storage model utilizing spatial correlation functions. *Biological Cybernetics*, 5(3):113–119, Sep. 1968.

[4] S. Grossberg. Some nonlinear networks capable of learning a spatial pattern of arbitrary complexity. 59:368–372, 1968.

[5] W. G. Wee. Generalized inverse approach to adaptive multiclass pattern classification. *IEEE Trans. Comput.*, 17(12):1157–1164, 1968.

[6] S. I. Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Trans. Comput.*, 21(11):1197–1206, 1972.

[7] T. Kohonen. Correlation matrix memories. *IEEE Transactions on Computers*, C-21:353–359, 1972.

[8] S. Grossberg. Contour enhancement, short-term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, (53):213–257, 1973.

[9] T. Kohonen and M. Ruohonen. Representation of associated data by matrix operators. *IEEE Trans. Comput.*, 22(7):701–702, 1973.

[10] Ingo Rechenberg. *Evolutionsstrategie. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Berlin, Germany, 1973.

[11] T. J. Sejnowski. Storing covariance with nonlinearly interacting neurons. *Journal of Mathematical Biology*, 4(4):303–321, 1977.

[12] Richard Forsyth. BEAGLE A Darwinian approach to pattern recognition. *Kybernetes*, 10:159–166, 1981.

[13] R. Sutton and A. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88:135–171, 1981.

[14] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[15] A.K. Dewdney. Computer recreations: The game of life acquires some successors in three dimensions. *Sci. Amer.*, 256(2):16–24, Feb. 1987.

[16] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 41–49, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[17] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26(23):4979–4983, 1987.

[18] A H Klopf. A drive-reinforcement model of single neuron function: An alternative to the hebbian neuronal model. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 265–270, Woodbury, NY, USA, 1987. American Institute of Physics Inc.

[19] B. Kosko. Adaptive bidirectional associative memories. *Applied Optics*, 26(23):4947–4960, 1987.

[20] Teuvo Kohonen. An introduction to neural computing. *Neural Networks*, 1(1):3–16, 1988.

[21] B. Kosko. Bidirectional associative memories. *IEEE Trans. Syst. Man Cybern.*, 18(1):49–60, 1988.

[22] Karl Sims. Artificial evolution for computer graphics. In Francisco J. Varela and Paul Bourgine, editors, *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 319–328, New York, NY, USA, 11-13 1991. ACM.

[23] L. Bull and T. C. Fogarty. Co-evolving communicating classifier systems for tracking. In *Proceedings of ANNGA93 an International Conference on Neural Networks and Genetic Algorithms*, pages 522–527. Springer-Verlag, 1993.

[24] J. Angeline. Evolving fractal movies. In J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 503– 511. MIT Press and Stanford University, CA, USA, 1996.

[25] G. X. Ritter, P. Sussner, and J. L. Díaz de León S. Morphological associative memories. *IEEE Transactions on Neural Networks*, 9(2):281–293, 1998.

[26] H. Takagi. Interactive evolutionary computation: System optimisation based on human subjective evaluation. *IEEE Int. Conf. on Intelligent Engineering Systems*, pages 1–6, 1998.

[27] Y. Deng, S.Kenney, M.S.Moore, and B. S.Manjunath. Peer group filtering and perceptual color image quantization. In *IEEE International Symposium on Circuits and Systems VLSI (ISCAS'99)*, volume 4, pages 21–24, Jun 1999.

[28] Y. Deng, B. S.Manjunath, and H.Shin. Color image segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR'99*, volume 2, pages 446–51, Jun 1999.

[29] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artif. Life*, 5(2):137–172, 1999.

[30] H. Ohsaki M. Takagi. Iec-based hearing aid fitting. In *IEEE International Conference on Systems, Man, and Cybernetics.*, volume 3, pages 657–662, Tokyo, Japan, Oct. 1999.

[31] K. Yamamoto, H. Yamada, and I. Yoda. Automatic acquisition of hierarquical mathematical morphology procedures by genetic algorithms. *Image and Vision Computing*, 17 (10):749–760, 1999.

[32] Xin Yao. Evolving artificial neural networks. In *Proceedings of the IEEE*, pages 1423–1447, 1999.

[33] P. Collet, E. Lutton, M. Schoenauer, P. Collet, E. Lutton, F. Raynal, and M. Schoenauer. Polar ifs + parisian genetic programming = efficient ifs inverse problem solving. *Genet. Programm. Evolvable Mach. J*, 1:361, 2000.

[34] Nicolas Monmarché, G.Ñocent, Gilles Venturini, and P. Santini. On generating html style sheets with an interactive genetic algorithm based on gene frequencies. In *AE '99: Selected Papers from the 4th European Conference on Artificial Evolution*, pages 99–110, London, UK, 2000. Springer-Verlag.

[35] Potter, A. Mitchell, and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation.*, 8(1):1–29, 2000.

[36] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[37] Y. Deng, , and B. S.Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '01)*, 23(8):800–810, Aug 2001.

[38] S. G. Ficici and J. B. Pollack. Game theory and the simple coevolutionary algorithm: Some preliminary results on fitness sharing. *GECCO 2001 Workshop on Coevolution: Turning Adaptative Algorithms upon Themselves*, 2001.

[39] A. Broder. A taxonomy of web search. *IBM Research, ACM SIGIR Forum*, 36(2), 2002.

[40] J. Louchet, M. Guyon, M.J. Lesot, and A. Boumaza. Dynamic fies: a new pattern. recognition tool applied to stereo sequence processing. *Pattern Recognition Letters*, 23(1):335–345, 2002.

[41] J. L. Diaz de León and C. Yánez. Memorias asociativas basadas en relaciones de orden y operaciones binarias. *Computación y Sistemas*, 6(4):300–311, 2003.

[42] M. Kokare, Chatterji B.N., and Biswas P.K. Comparison of similarity metrics for texture image retrieval. In *Proceedings of 2003 Asia-Pacific Region Conference on Convergent Technologies*, volume 2, pages 571–575. IEEE, 2003.

[43] Yann Landrin-Schweitzer, Pierre Collet, and Evelyne Lutton. Interactive gp for data retrieval in medical databases. *Genetic Programming, LNCS. Springer Berlin / Heidelberg*, 2610/2003, Jan. 2003.

[44] Yann Landrin-Schweitzer, Pierre Collet, Evelyne Lutton, and Thierry Prost. Introducing lateral thinking in search engines with interactive evolutionary algorithms. In *SAC*

'03: Proceedings of the 2003 ACM symposium on Applied computing, pages 214–219, New York, NY, USA, 2003. ACM.

[45] R. Rangel-Kuoppa and C. Avilés-Cruz. Buscador de imágenes por internet. *CNyCIIC2003*, II:111–118, Oct. 2003.

[46] Aistis Raudys. High speed associative memories for feature extraction and visualisation. *Pattern Recogn. Lett.*, 24:1317–1329, 2003.

[47] G. X. Ritter, G. Urcid, et al. Reconstruction of patterns from noisy inputs using morphological associative memories. *International Journal of Mathematical Imaging and Vision*, 19(2):95–111, 2003.

[48] S. Silva and J. Almeida. Gplab - a genetic programming toolbox for matlab. *In Gregersen L (ed), Proceedings of the Nordic MATLAB Conference*, pages 273–278, 2003.

[49] P. Sussner. Generalizing operations of binary auto-associative morphological memories using fuzzy set theory. *Journal of mathematical imaging and vision*, 19 (2):81–93, 2003.

[50] K.C. Tan, Y.J. Yang, and T.H. Lee. A distributed cooperative coevolutionary algorithm for multiobjective optimization. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 4, pages 2513–2520 Vol.4, Dec. 2003.

[51] A. Ajith. Meta learning evolutionary artificial neural networks. *Neurocomputing*, 56:1 – 38, 2004.

[52] F. A. Sánchez Garfias, J. L. Díaz de León S, and C. Yá nez Márquez. Lernmatrix de steinbuch: Avances teóricos. *Computación y Sistemas*, 7:175 – 189, 03 2004.

[53] H. Sossa. New associative memories to recall real-valued patterns. *LNCS 3287. Springer Verlag.*, pages 195–202, 2004.

[54] K. Barnard and K. Yanai. Probabilistic web image gathering. *Multimedia Information Retrieval*, pages 57–64, 2005.

[55] E. Cantu-Paz and C. Kamath. An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, 35(5):915–927, Oct. 2005.

[56] B. Cruz. Restauración de palabras impresas usando memorias asociativas. Master's thesis, CIC-IPN, 2005.

[57] B. Sumengen and B. S. Manjunath. Multi-scale edge detection and image segmentation. In *European Signal Processing Conference (EUSIPCO)*, Sep 2005.

[58] E. R. Sykes and A. Mirkovic. A fully parallel and scalable implementation of a hopfield neural network on the sharc-net supercomputer. In *HPCS '05: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications*, pages 103–109, Washington, DC, USA, 2005. IEEE Computer Society.

[59] K. Yanai. Image collector ii: A system to gather a large number of images from the web. *IEICE Transactions*, 88-D(10):2432–2436, 2005.

[60] R. Barrón. *Memorias asociativas y redes neuronales morfológicas para la recuperación de patrones*. PhD thesis, CIC-IPN, 2006.

[61] E. Dunn, G. Olague, and E. Lutton. Parisian camera placement for vision metrology. *Pattern Recogn. Lett.*, 27(11):1209–1219, 2006.

[62] A. R. Silva Lavalle. Un método de algoritmos genéticos para optimización de memorias asociativas morfológicas. Master's thesis, Univesidad de Puerto Rico, 2006.

[63] G. Olague and C. Puente. Honeybees as an intelligent based approach for 3d reconstruction. *International Conference on Pattern Recognition. August 20-24, 2006. Hong Kong, China.*, Aug. 2006.

[64] G. Olague and C. Puente. Parisian evolution with honeybees for three-dimensional reconstruction. *Genetic and Evolutionary Computation Conference*, July 2006.

[65] M. Quintana, R. Poli, and E. Claridge. Morphological algorithm design for binary images using genetic programming. *Genetic Programming and Evolvable Machines*, 7, Issue 1:81–102, 2006.

[66] L. Trujillo and G. Olague. Synthesis of interest point detectors through genetic programming. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 887–894, New York, NY, USA, 2006. ACM.

[67] L. Trujillo and G. Olague. Using evolution to learn how to perform interest point detection. In *ICPR (1)*, pages 211–214, 2006.

[68] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[69] R. Barrón and H. Sossa. Extended $\alpha\beta$ associative memories. *Revista Mexicana de Física*, 1(53):10–20, 2007.

[70] B. Hernández, G. Olague, R. Hammoud, L. Trujillo, and E. Romero. Visual learning of texture descriptors for facial expression recognition in thermal imagery. *Comput. Vis. Image Underst.*, 106:258–269, 2007.

[71] G. Olague, E. Romero, L. Trujillo, and B. Bhanu. Multiclass object recognition based on texture linear genetic programming. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 291–300, Berlin, Heidelberg, 2007. Springer-Verlag.

[72] C. B. Pérez and G. Olague. Unsupervised evolutionary segmentation algorithm based on texture analysis. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 407–414, Berlin, Heidelberg, 2007. Springer-Verlag.

[73] J. Rabuñal, D. Rivero, and Alejandro Pazos. Automatic desing of anns by means of gp for data mining tasks: Iris flower classification problem. *ICANNGA 07, Springer-Verlag LNCS*, I(4431):276–285, 2007.

[74] L. Trujillo and G. Olague. Scale invariance for evolved interest operators. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 423–430, Berlin, Heidelberg, 2007. Springer-Verlag.

[75] L. Trujillo, G. Olague, P. Legrand, and E. Lutton. Regularity based descriptor computed from local image oscillations. *Opt. Express*, 15(10):6140–6145, 2007.

[76] R. A. Vazquez and H. Sossa. A new model of associative memories network. *Third International Workshop on Artificial Networks and Intelligent Information Processing (ANNIP 2007). Angers, France*, pages 9–12, 2007.

[77] B. Cruz, R. Barrón, and H. Sossa. Pattern classification based on conformal geometric algebra and optimization techniques. In *MICAI '08: Proceedings of the 7th Mexican International Conference on Artificial Intelligence*, pages 273–283, Berlin, Heidelberg, 2008. Springer-Verlag.

[78] C. B. Perez and G. Olague. Learning invariant region descriptor operators with genetic programming and the f-measure. *International Conference on Pattern Recognition. ICPR.*, 2008.

[79] L. Trujillo and G. Olague. Automated design of image operators that detect interest points. *Evolutionary Computation. MIT Press.*, 16(4):483–507., 2008.

[80] L. Trujillo, G. Olague, F. Fernández, and E. Lutton. Behavior-based speciation for evolutionary robotics. *Genetic and Evolutionary Computation Conference. July 12-16, 2008. Atlanta, GA, USA.*, 2008.

[81] R. A. Vázquez and H. Sossa. A bidirectional hetero-associative memory for true-color patterns. *Neural Process. Lett.*, 28(3):131–153, 2008.

[82] R. A. Vázquez and H. Sossa. Hetero-associative memories for voice signal and image processing. In *CIARP '08: Proceedings of the 13th Iberoamerican congress on Pattern Recognition*, pages 659–666, Berlin, Heidelberg, 2008. Springer-Verlag.

[83] R. A. Vázquez and H. Sossa. Voice translator based on associative memories. In *ISNN '08: Proceedings of the 5th international symposium on Neural Networks*, pages 341–350, Berlin, Heidelberg, 2008. Springer-Verlag.

[84] J. Villegas-Cortez. Síntesis automática de memorias asociativas mediante programación genética. Primer encuentro de estudiantes de doctorado en Ciencias de la Computación en México - 50 Años del Cómputo en México, Sep. 2008. Cinvestav - IPN, Unidad Zacatenco. México.

[85] B. Cruz, H. Sossa, and R. Barrón. Geometric associative processing applied to pattern classification. In *ISNN 2009: Proceedings of the 6th International Symposium on Neural Networks*, pages 977–985, Berlin, Heidelberg, 2009. Springer-Verlag.

[86] Chi-Keong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103–127, February 2009.

[87] C. B. Pérez and G. Olague. Evolutionary learning of local descriptor operators for object recognition. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1051–1058, New York, NY, USA, 2009. ACM.

[88] C. B. Pérez and G. Olague. Evolving local descriptor operators through genetic programming. In *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 414–419, Berlin, Heidelberg, 2009. Springer-Verlag.

[89] C. Puente, G. Olague, S. V. Smith, S. Bullock, M. A. Gonzalez, and A. Hinojosa. Genetic programming methodology that synthesize vegetation indices for the estimation of soil cover. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1593–1600, New York, NY, USA, 2009. ACM.

[90] C. Puente, G. Olague, S. V. Smith, S. H. Bullock, M. A. González-Botello, and A. Hinojosa-Corona. A novel gp approach to synthesize vegetation indices for soil erosion assessment. In *EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing*, pages 375–384, Berlin, Heidelberg, 2009. Springer-Verlag.

[91] R. A. Vázquez and H. Sossa. A computational approach for modeling the role of the focus visual attention in an object categorization task. *BMC Neuroscience*, 10:P310, 2009.

[92] R. A. Vázquez and H. Sossa. Morphological hetero-associative memories applied to restore true-color patterns. In *ISNN 2009: Proceedings of the 6th International Symposium on Neural Networks*, pages 520–529, Berlin, Heidelberg, 2009. Springer-Verlag.

[93] R. A. Vázquez, H. Sossa, and B. A. Garro. The role of the infant vision system in 3d object recognition. *LNCS. Advances in Neuro-Information Processing*, pages 800–807, 2009.

[94] J. Villegas-Cortez, H. Sossa, C. Avilés-Cruz, and G. Olague. Automatic synthesis of associative memories by genetic programming, a first approach. *Research in Computing Science. Advances in Computer Science and Engineering*, 42:91–102, 2009.

[95] S. Cagnoni, E. Lutton, and G. Olague. *Genetic and Evolutionary Computation for Image Processing and Analysis*, volume 8 of *EURASIP Book Series on Signal Processing and Communications Series*. Hindawi Publishing Corporation, 2008.

[96] B. Cruz, R. Barrón, and H. Sossa. *Geometric algebra computing for Computer Science and Engineering.*, chapter Geometric Associative Memories and their Applications in Pattern Classification. Springer Verlag, 2009.

[97] M. Dorigo and G. Di Caro. *The ant colony optimization meta-heuristic*, chapter 2, pages 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[98] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[99] L. Fausett. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[100] Keinosuke Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[101] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[102] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1992.

[103] Woods Eddins Gonzalez. *Digital Image Processing using Matlab*. Prentice Hall, Boston, MA, USA, 2004.

[104] D. Hebb. *The organization of behavior*. John Wiley & Sons, Inc., 1949.

[105] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[106] H.Sossa. *Rasgos descriptores para el reconocimiento de objetos*. Ciencia de la Computación. Instituto Politecnico Nacional, 2006.

[107] ER Kandel, Schwartz JH, and Jessell TM. *Principles of Neural Science*. New York: McGraw-Hill, 4th ed. edition, 2000. ISBN 0-8385-7701-6.

[108] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[109] W. B. Langdon, N. F. McPhee, and R. Poli. *A Field Guide to Genetic Programming*. Lulu.com, under Creative Commons, 2008. Creative Commons.

[110] David G Luenberger. *Programación lineal y no lineal*. Addison-Wesley Iberoamericana, 1989.

[111] E. Lutton, G. Olague, and S. Cagnoni. *Genetic Evolutionary Computation for Image Processing Analysis*, volume 8 of *EURASIP Book Series on Signal Processing and Communications*. Hindawi Publishing Corporation, Feb. 2008.

[112] Vishvjit S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[113] Nils J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, Inc. San Francisco, California, 5 edition, 1998.

[114] R. Rojas and J. Feldman. *Neural Networks: A Systematic Introduction*. Springer, Mar. 1996.

[115] F. Rothlauf, Jürgen Branke, S. Cagnoni, E. Costa, C. Cotta, R. Drechsler, E. Lutton, P. Machado, J. H. Moore, J. Romero, G. D. Smith, G. Squillero, and H. Takagi, editors. *Applications of Evolutionary Computing, EvoWorkshops 2006: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC, Budapest, Hungary, April 10-12, 2006, Proceedings*, volume LNCS 3907. Springer, 2006.

[116] Leszek Rutkowski. *Computational Intelligence: Methods and Techniques*. Springer Publishing Company, Incorporated, 2008.

[117] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 1995.

[118] S. Todd and W. Latham. *Evolutionary Art and Computers*. Academic Press, 1992.

[119] E. Tulving and F. Craik. *The Oxford Handbook of Memory*. Oxford University Press, 2000.

[120] J. Martínez Victor and Hilera José R. *Redes neuronales artificiales. Fundamentos, modelos y aplicaciones.* Ra-ma / Paradigma. Addison-Wesley Iberoamericana, 1995.

Apéndice A

Glosario de términos

Aprendizaje

El aprendizaje es el proceso a través del cual se adquieren nuevas habilidades, destrezas, conocimientos, conductas o valores como resultado del estudio, la experiencia, la instrucción y la observación. Este proceso puede ser analizado desde distintas perspectivas, por lo que existen distintas teorías del aprendizaje. El aprendizaje es una de las funciones mentales más importantes en humanos, animales y sistemas artificiales. El aprendizaje como establecimiento de nuevas *relaciones* temporales entre un ser y su medio ambiental han sido objeto de diversos estudio empíricos, realizados tanto en animales como en el hombre. Midiendo los progresos conseguidos en cierto tiempo se obtienen las curvas de aprendizaje, que muestran la importancia de la repetición de algunas predisposiciones fisiológicas, de «los ensayos y errores», de los períodos de reposo tras los cuales se aceleran los progresos, etc. Muestran también la última relación del aprendizaje con los reflejos condicionados.

Adquisición de conceptos

Las teorías del aprendizaje tratan de explicar como se constituyen los significados y como se aprenden los nuevos conceptos.

Un concepto puede ser definido buscando el sentido y la referencia, ya sea desde arriba, en función de la intensión del concepto, del lugar que el objeto ocupa en la red conceptual que el individuo posee; o desde abajo, haciendo alusión a sus atributos. Los conceptos nos sirven para limitar el aprendizaje, reduciéndo la complejidad del entorno; nos sirven para identificar objetos, para ordenar y clasificar la realidad, nos premiten predecir lo que va a ocurrir.

Hasta hace poco, los psicólogos suponían, siguiendo a Mill y a otros filósofos empiris-

tas, que las personas adquirimos conceptos mediante un proceso de abstracción (teoría inductivista) que suprime los detalles idiosincráticos que difieren de un ejemplo a otro, y que deja sólo lo que se mantiene común a todos ellos. Este concepto, llamado prototipo, está bien definido y bien delimitado y tiene sus referentes en cada uno de sus atributos En consecuencia, la mayoría de los experimentos han utilizado una técnica en la cual los sujetos tienen que descubrir el elemento común que subyace a un concepto.

Los conceptos cotidianos, en cambio, no consisten en la conjunción o disyunción de características, sino más bien en relaciones entre ellas. Otro aspecto de los conceptos de la vida diaria es que sus ejemplos puede que no tengan un elemento común. Wittgenstein en sus investigaciones filosóficas: sostuvo que los conceptos dependen, no de los elementos comunes, sino de redes de similitudes que son como las semejanzas entre los miembros de una familia.

Los conceptos cotidianos no son entidades aisladas e independientes, están relacionados unos con otros. Sus límites están establecidos, en parte, por la taxonomía en que aparecen. Las relaciones más claras son las jerarquías generadas mediante la inclusión de un concepto dentro de otro.

Existen dos vías formadoras de conceptos: *mediante el desarrollo de la asociación* (empirista) y *mediante la reconstrucción* (corriente europea).

Para la corriente asociacionista no hay nada en el intelecto que no haya pasado por los sentidos. Todos los estímulos son neutros. Los organismos son todos equivalentes. El aprendizaje se realiza a través del proceso recompensa-castigo (teoría del conductismo: se apoya en la psicología fisiológica de Pavlov). Es antimentalista. El recorte del objeto está dado por la conducta, por lo observable. El sujeto es pasivo y responde a las complejidades del medio.

Para las corrientes europeas, que están basadas en la acción y que tienen uno de sus apoyos en la teoría psicogenética de Piaget, el sujeto es activo. Los conceptos no se aprenden sino que se reconstruyen y se van internalizando. Lo importante es lo contextual, no lo social.

Aprendizaje hebbiano

En [104] Donald O. Hebb, un psicólogo de la universidad McGill, Montreal, Canada, sugirió que el aprendizaje involucra específicamente el reforzamiento de ciertos patrones de la actividad neuronal, por medio de incrementar la probabilidad (los pesos)

de como estan conectadas las neuronas en asociaciones, este tipo de aprendizaje se le llama hebbiano, el cual se basa en el siguiente postulado formulado por D. O. Hebb:

> Cuando un axón de una celda A está suficientemente cerca como para conseguir excitar una celda B y repetida o persistentemente toma parte en su activación, algún proceso de crecimiento o cambio metabólico tiene lugar en una o ambas celdas, de tal forma que la eficiencia de A, cuando la celda a activar es B, aumenta.

Por celda, Hebb entiende un conjunto de neuronas fuertemente conectadas a través de una estructura compleja. La eficiencia podría identificarse con la intensidad o magnitud de la conexión; i.e., con el peso. Se puede decir, por tanto, que el aprendizaje hebbiano consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la correlación (multiplicación en el caso de valores binarios $+1$ y -1), de los valores de activación (salidas) de las dos neuronas conectadas:

$$\Delta w_{ij} = x_i . y_j \tag{A.1}$$

Esta expresión responde a la idea de Hedd, puesto que si las dos unidades son activas (positivas), se produce un reforzamiento de la conexión. Por el contrario, cuando una es activa y la otra pasiva (negativa), se produce un debilitamiento de la conexión. Se trata de una regla de aprendizaje no supervisado, pues la modificación de los pesos se realiza en función de los estados (salidas de las neuronas) obtenidos tras la presentación de cierto estímulo (información de entrada a la red), sin tener en cuenta si se deseaba obtener o no esos estados de activación.

Cómputo evolutivo

El llevar a las computadoras la famosa teoría de Darwin (publicada en *El origen de las especies basada en la selección natural*) consiste en tratar de imitar con programas de cómputo la capacidad de una población de organismos vivos para adaptarse a su medio ambiente por medio de mecanismos de selección y reproducción. En los últimos cuarenta años varios métodos estocásticos de optimización se han basado en este principio. *Darwinismo Artificial* o *algoritmos evolutivos* es el nombre común para estas técnicas, quizás la más comúnmente escuchadas sean *algoritmos genéticos, estrategias evolutivas* o *programación genética*.

Frente de pareto

Este concepto es relativo a la *optimización multiobjetivo* (u *optimización multicriterio*). se define como el problema de encontrar un vector de variables de decisión que satisfacen unas restricciones y optimizan una función vectorial cuyos elementos representan a la función objetivo. Estas funciones forman una descripción matemática de criterios de evaluación que generalmente están en conflicto unos con otros. Por tanto, el término .ºptimizar"implica encontrar una solución que daría los valores de todas las funciones objetivo aceptables para el diseñador.

Es raro que exista un solo punto que satisfaga todas las funciones objetivo; normalmente se busca un equilibrio al tratar con problemas de optimización multiobjetivo; por lo tanto, el concepto de óptimo es diferente. El concepto más normal de óptimo fue propuesto originalmente por Francis Ysidro Edgeworth (1881), y más adelante por Vilfredo Pareto (1886), y se suele denominar óptimo Pareto u óptimo tipo Pareto; un vector es óptimo pareto si no existe ninguno cuyos valores sean mejores para cada uno de los componentes (estrictamente, si son menores o iguales para todos, y al menos menor estricto para uno de ellos). Lo más habitual es que no exista un solo óptimo Pareto, sino un conjunto de ellos; a este conjunto de solucione se le suele denominar *conjunto no dominado*, y a su gráfico se le suele llamar *frente Pareto*.

Optimalidad de Pareto

Decimos que un punto $\overrightarrow{x}^* \in \Omega$ es un óptimo de Pareto si para toda $\overrightarrow{x} \in \Omega$, e I $I = \{1, 2, \ldots, k\}$ se tiene:

$$\forall i \in I(f_i(\overrightarrow{x}) = f_i(\overrightarrow{x}^*)) \tag{A.2}$$

o existe al menos una $i \in I$ tal que

$$f_i(\overrightarrow{x}) > f_i(\overrightarrow{x}^*) \tag{A.3}$$

Dominancia de Pareto

Un vector $\overrightarrow{u} = (u_1, u_2, \ldots, u_k)$ domina a otro vector $\overrightarrow{v} = (v_1, v_2, \ldots, v_k)$ (denotado mediante $\overrightarrow{u} \leq \overrightarrow{v}$), si y sólo si u es parcialmente menor a v, i.e., $\forall i \in \{1, 2, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, 2, \ldots, k\} : u_i < v_i$.

Conjunto de Óptimos de Pareto

Para un problema multiobjetivo dado $\overrightarrow{f}(x)$, el conjunto de óptimos de Pareto (P^*) se define como:

$$\mathsf{P}^* := \{x \in \Omega | \neg \exists\, x' \in \Omega,\ \overrightarrow{f}(x') \leq \overrightarrow{f}(x)\} \tag{A.4}$$

Frente de Pareto

Para un problema multiobjetivo dado $\overrightarrow{f}(x)$ y un conjunto de óptimos de Pareto (P^*), el frente de Pareto (PF^*) se define como:

$$(\mathsf{PF}^*) := \{\overrightarrow{u} = \overrightarrow{f} = (f_1(x), \ldots, f_k(x)) | x \in \mathsf{P}^*\} \tag{A.5}$$

Memoria asociativa (MA)

La MA es un dispositivo que permite asociar patrones de entrada con patrones de salida. Esto es, al presentarle a una memoria asociativa **M** un patrón de entrada X, ésta responde con el correspondiente patrón (vector) de salida Y. Podría decirse que una memoria asociativa es una Red Neuronal Artificial (RNA) de una sola capa.

Red neuronal artificial (RNA)

Las RNA son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Es un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida. En inteligenia artificial es frecuente referirse a ellas como “redes de neuronas” o “redes neuronales”.

Existen varias formas de definir a las RNA, en el anterior párrafo se ha intentado dar una definición generalizada, pero hay que resaltar dos definiciones por parte de autores relevantes:

“... un sistema de computación hecho por un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas”. [17]

“RNA son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico”.[20]

Programación genética (PG)

La programación genética (PG) es un método automático para la creación de programas

de cómputo como solución en alto nivel a problemas.La programación genética inicia a partir de la formulación del problema de alto nivel como “¿qué se necesita hacer?”, y automáticamente se procede a la creación de programas para resolver el problema[1].

Con otras palabras podemos decir que la PG es una metodología basada en los algoritmos evolutivos e inspirada en la evolución biológica para construir programas de cómputo que realicen una tarea definida por el usuario. Es una técnica de aprendizaje automático utilizada para optimizar una población de programas de acuerdo a una función de aptitud que evalúa la capacidad de cada programa para resolver la tarea en cuestión.

[1]Definición corta aportada por J. Koza en la página oficial de PG: http://www.genetic-programming.org/