SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE LA PAZ

SECRETARÍA DE
EDUCACIÓN PÚBLICA

INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
MAESTRÍA EN SISTEMAS COMPUTACIONALES

SIMULACIÓN DE UN ROBOT AUTÓNOMO UTILIZANDO REDES NEURONALES COOPERATIVAS

T E S I S

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
OSCAR LUIS RUIZ BERNAL

DIRECTOR DE TESIS:
DR. SAÚL MARTÍNEZ DÍAZ

MIEMBROS DEL JURADO:

PRESIDENTE: DR. SAÚL MARTÍNEZ DÍAZ, CICESE
SECRETARIO: MSC. JAVIER ALBERTO CARMONA TROYO, ITLP
VOCAL: MSC. ILIANA CASTRO LIERA, ITLP

LA PAZ, BAJA CALIFORNIA SUR, MÉXICO, JUNIO 2012

SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE LA PAZ

SECRETARÍA DE EDUCACIÓN PÚBLICA

La Paz, B.C.S., a 15/Junio/2012

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

DEPI/132/2012

Asunto: Autorización de impresión.

C. OSCAR LUIS RUÍZ BERNAL,
ALUMNO DE LA MAESTRÍA EN SISTEMAS COMPUTACIONALES
P R E S E N T E.

Con base en el dictamen de aprobación emitido por la Comisión Revisora de la Tesis denominada: "SIMULACIÓN DE UN ROBOT AUTÓNOMO UTILIZANDO REDES NEURONALES COOPERATIVAS", mediante la opción de tesis (Proyectos de Investigación), entregado por usted para su análisis, le informamos que se AUTORIZA la impresión.

A t e n t a m e n t e,
"Ciencia es Verdad, Técnica es Libertad"

M.C. MANUEL E. CASILLAS BROOK,
JEFE DE LA DIV. DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.

INSTITUTO TECNOLOGICO DE LA PAZ
DIVISION DE ESTUDIOS DE POSGRADO
E INVESTIGACION

C.c.p. Coordinación de la Maestria.
C.c.p. Depto. de Servicios Escolares.

Blvd. Forjadores de B.C.S. No. 4720 C.P. 23080 Apdo. Postal 43-B
Tels. (612) 12-107-05, 12-104-24, 12-104-26, Fax:(612) 12-112-95
email:direccion@itlp.edu.mx www.itlp.edu.mx

SUBSECRETARÍA DE EDUCACIÓN SUPERIOR
DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE LA PAZ

SECRETARÍA DE EDUCACIÓN PÚBLICA

La Paz, B.C.S., 15/Junio/2012

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Asunto: Dictamen de la Comisión Revisora

C. M.C. MANUEL E. CASILLAS BROOK
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
P R E S E N T E.

Por este conducto, enviamos a usted el dictamen de la Comisión Revisora de tesis para la obtención del Grado de Maestro, con los siguientes datos generales:

No. de control / Nombre del Alumno (a): M09310020 / OSCAR LUIS RUÍZ BERNAL
Maestría en: SISTEMAS COMPUTACIONALES
Título de la tesis: SIMULACIÓN DE UN ROBOT AUTÓNOMO UTILIZANDO REDES NEURONALES COOPERATIVAS
DICTAMEN: Se autoriza el trabajo de investigación, en virtud de que realizó las correcciones correspondientes conforme a las observaciones planteadas por esta Comisión Revisora

A t e n t a m e n t e
La Comisión Revisora

M.S.C. JAVIER ALBERTO CARMONA TROYO

M.S.C. ILIANA CASTRO LIERA

DR. SAÚL MARTÍNEZ DÍAZ

c.c.p. Coordinación de la Maestría.
c.c.p. Interesado.
c.c.p. Departamento de Servicios Escolares.

Blvd. Forjadores de B.C.S. No. 4720 C.P. 23080 Apdo. Postal 43-B
Tels. (612) 12-107-05, 12-104-24, 12-104-26, Fax:(612) 12-112-95
email:direccion@itlp.edu.mx
www.itlp.edu.mx

A mis Niñas

A Dios, un gran administrador

de nuestro tiempo y espacio

Agradecimientos

Le agradezco a cada una de las personas que lograron cambiar algo en mí, para ser quien soy ahora.

Al Dr. Saúl Martínez Díaz, su orientación fue pieza importante en la construcción y término de este trabajo.

A mis maestros, que abrieron este nuevo camino para mí; una guía académica llena de consejos y experiencias. Lic. María del Carmen Rodríguez Aguilar, M.S.C. Javier Alberto Carmona Troyo, Dr. Marco Antonio Castro Liera, M.A.T.I. Luis Armando Cárdenas Florido, M.C. Jesús Antonio Castro, M.C. Bernabé Ortiz y Hebert, Dr. Mario Cortés Larrinaga. Por su apoyo al M.C. Antonio Hernández y al M.C. Manuel E. Casillas Brook, Jefe de la división de estudios de posgrado y administración. Muchas Gracias.

Al personal del Instituto Tecnológico de La Paz que me ha tomado como parte de esta gran familia, en especial a la Lic. Margarita Alicia Martínez Gutiérrez y a la Lic. Karla Gallardo Ruiz.

Resumen

Los sistemas autónomos han incrementado su presencia en muchas áreas de la actividad humana. Los entornos industriales donde se requieren mecanismos capaces de tomar decisiones “Inteligentes”, son los espacios que han dado auge a estos sistemas. Aplicaciones tales como actividades de rescate, actividades en ambientes tóxicos, misiones espaciales e incluso en el hogar los utilizan cada vez más.

Este trabajo es un análisis comparativo entre la eficiencia de una red neuronal monolítica y una estructura de redes neuronales cooperativas. Se espera que la red cooperativa presente mejores resultados al distribuir la tarea principal entre cada una de las subredes. Para el estudio se desarrolla un agente autónomo en un ambiente simulado el cual tiene la tarea de recoger una pelota en un área de búsqueda y depositarla en un contenedor. Para el entrenamiento se implementan algoritmos genéticos donde la función de eficiencia divide el objetivo principal en tareas más pequeñas como “no chocar”, ”detectar el objeto de frente”, ”detectar el objeto por un lado”.

Abstract

Autonomous systems have increased their presence in many areas of human activity. Industrial environments where mechanisms capable of making "Intelligent" decisions are needed are the spaces that have been booming these systems. Such as, rescue activities, activities in toxic environments, space missions and even in housing use them more and more.

This work is a comparative analysis of the efficiency of monolithic and cooperative neural network structures. It is expected that a cooperative network distribute will be able to more efficient to distribute the main task between each of the subnets. An autonomous agent was simulating for the study with the task of collecting a ball in a search area and places it in a container. Genetic algorithms are use for training, where the efficiency function divides the main goal into smaller tasks such as "avoid crashing", "detect the object in front," "detect the object on one side."

Índice

Capítulo 1. Introducción 1
1.1. Objetivo 3
1.2. Alcance 3
1.3. Hipótesis 3
1.4. Conceptos Generales 4
1.5. Inteligencia artificial y robótica 6
1.6. Descripción del trabajo 12
1.6.1. Introducción 12
1.6.2. Funciones del agente y utilidad 14
1.6.3. Unidades de Medida a utilizar en la comparación y análisis 18
1.7. Estado del arte 20
1.7.1. Un Enfoque Híbrido en el Desarrollo de Robótica Basada en el Comportamiento 20
1.7.2. Control neural distribuido para robots autónomos complejos 21
1.8. Cinemática del robot móvil 22

Capítulo 2. Redes Neuronales Artificiales 25
2.1. La Neurona Biológica 25
2.2. La Neurona Artificial 27
2.2.1. Modelo General de la Neurona Artificial 27
2.2.2. Arquitectura de las Redes Neuronales 29

Capítulo 3. Algoritmos Genéticos 38
3.1. Selección 42
3.2. Cruce 45
3.3. Mutación 47
3.4. Función de Aptitud 50

Capítulo 4. Neuro-Evolución y Modularidad 51
4.1. Modularidad 51
4.1.1. Introducción 51
4.1.2. Definiciones 51
4.1.3. Evidencias Biológicas 52
4.2. Neuro-Evolución 53

Capítulo 5. Diseño del Modelo 55
5.1. Entorno de Simulación ODE 55
5.1.1. ODE 56
5.1.2. Motor de Física 58
5.1.3. Objetos 59
5.1.4. Colisiones 60
5.2. Arquitectura del modelo 63
5.2.1. Modelo físico 63
5.2.2. Modelo Neuronal 67
5.2.3. Algoritmo Genético 72

Capítulo 6. Progreso del modelo y evolución 76
6.1. Análisis de resultados 80
6.2. Conclusiones 88

Anexo A Análisis de complejidad 90
Anexo B Comportamientos Típicos 91
Anexo C Éxitos por posición inicial de la pelota. 92
Anexo D Tiempo de búsqueda por posición de la pelota. 93
Anexo E Código Fuente 95
Operadores Genéticos 95
Cálculos de Red Neuronal 98
Función de Eficiencia 99

Bibliografía 100

Índice de Figuras

Figura 1.1 Modelo simulado 4
Figura 1.2 Tipos de propulsión 23
Figura 1.3 Propulsión del modelo 24
Figura 1.4 Grados de libertad. Propulsión de carro y diferencial 24

Figura 2.1 Neurona biológica 26
Figura 2.2 Funciones de activación 28
Figura 2.3 Elementos de una Neurona Artificial 28
Figura 2.4 Redes Neuronales Dos capas (izq.), Con capa oculta (Der.) 30
Figura 2.5 Regiones de decisión en el plano 34
Figura 2.6 Función lógica NAND 35
Figura 2.7 Función lógica XOR 35
Figura 2.8 Red con retroalimentación global (izq.) y Red con retroalimentación global y local (der.) 36

Figura 3.1 Diagrama de flujo de un algoritmo genético 42
Figura 3.2 Gráfica de tabla 3.1 en selección por ruleta 43
Figura 3.3 Asignación de probabilidad en selección por ruleta 44
Figura 3.4 Selección por torneo 44
Figura 3.5 Cromosomas binarios de dos individuos 45
Figura 3.6 Selección de un punto aleatorio para el cruce 45
Figura 3.7 Intercambio de cromosomas 46
Figura 3.8 Operador de cruce utilizando valores reales 47
Figura 3.9 Función de Weierstrass, presenta múltiples óptimos locales 48
Figura 3.10 Mutación con valores binarios 49

Figura 5.1 Tipos de uniones en ODE, izq. a der., Articulación, Bisagra, Pistón 60
Figura 5.2 Chasis del Modelo 64
Figura 5.3 Posición de las ruedas en el chasis 64
Figura 5.4 Posición de los sensores del vehículo 65
Figura 5.5 Área delimitada y contenedor 66
Figura 5.6 Estructura de Red Neuronal Monolítica 67
Figura 5.7 Modelo individual de Red Neuronal Cooperativa 68

Figura 6.1 Posiciones de las pelotas para el entrenamiento.....78
Figura 6.2 Las 25 posiciones fijas de la pelota.....80
Figura 6.3 Porcentaje de pruebas según porcentaje de efectividad en posiciones aleatorias.....82
Figura 6.4 Porcentaje de pruebas según porcentaje de efectividad en posiciones fijas.....84
Figura 6.5 Magnitud de los éxitos por posición de la pelota, Red Única.....85
Figura 6.6 Magnitud de los éxitos por posición de la pelota, Red Cooperativa.....85
Figura 6.7 Rutas típicas del vehículo.....86

Figura B.1 Trayectorias típicas de la Red Única.....91
Figura B.2 Trayectorias típicas de la Red Cooperativa.....91
Figura C.1 Magnitud de los éxitos por la posición de la pelota, Red Única.....92
Figura C.2 Magnitud de los éxitos por la posición de la pelota, Red Cooperativa...92
Figura D.1 Tiempo promedio de búsqueda de la pelota por posición inicial de la pelota, Red Única.....93
Figura D.2 Tiempo promedio de búsqueda del contenedor por posición de la pelota, Red Única.....93
Figura D.3 Tiempo promedio de búsqueda de la pelota por posición inicial de la pelota, Red Cooperativa.....94
Figura D.4 Tiempo promedio de búsqueda del contenedor por posición de la pelota, Red Cooperativa.....94

Índice de Tablas

Tabla 3.1 Tabla acumulativa en selección por ruleta.....43
Tabla 6.1 Mayor grado de eficiencia en posiciones aleatorias.....81
Tabla 6.2 Porcentaje de pruebas según porcentaje de efectividad en posiciones aleatorias.....81
Tabla 6.3 Mayor grado de eficiencia en posiciones fijas.....83
Tabla 6.4 Porcentaje de pruebas según porcentaje de efectividad en posiciones fijas.....83

Capítulo 1 Introducción

La mente humana ha desarrollado mecanismos para adquirir con cierta facilidad un nuevo conocimiento. Esta habilidad nos permite asimilar nueva información creando representaciones abstractas de los conceptos y los objetos, así como el desarrollo de patrones entre objetos de características similares. Además de crear cadenas de reacciones o estímulos que nos permiten realizar alguna tarea, a partir de "aprender" la serie de pasos a seguir para el logro de un determinado objetivo.

Sin embargo el poder realizar esta actividad de aprendizaje, con esta aparente sencillez, se debe a la compleja estructura de nuestro cerebro y demás conexiones nerviosas del cuerpo, esto aunado a la experiencia que se viene acumulando desde nuestro nacimiento y la cual permite que el próximo conocimiento sea más fácil de asimilar.

En la actualidad se conoce parte del funcionamiento del cerebro, en principio gracias al científico Español Santiago Ramón y Cajal quien en 1888, demostró que la estructura del sistema nervioso está compuesta por una red de células individuales interconectadas entre sí, las neuronas. Además de establecer que dentro de la neurona la información fluye desde las dendritas, pasa por el soma que es el cuerpo de la célula y llega hasta el axón para conectarse a otra neurona, es decir, que el flujo de la información es siempre en un mismo sentido. Esto es, un canal de comunicación para enviar instrucciones entre el cerebro y las articulaciones.

Desde hace mucho tiempo el hombre ha intentado imitarse a sí mismo, esfuerzo que lo ha llevado por dos vertientes, la primera imitando la apariencia del hombre, área que desarrolla la robótica, y la segunda imitando su forma de comportarse, aspecto que genera muchas líneas de investigación en la inteligencia artificial.

Hoy en día se trata de reproducir el complejo funcionamiento del cerebro utilizando redes neuronales artificiales. Esto es, utilizar el concepto de neurona como unidad y definirla como una función con entradas y salidas, establecer comunicación entre cada función pasando parámetros entre ellas y formar en su conjunto una red que logre un objetivo específico.

Claro que ésto es mucho más complejo ya que por definición cada neurona es idéntica a cualquier otra de la red. El “Truco” por decirlo así, consiste en que la relación entre una neurona y otra es diferente, el resultado de la función es uno pero al pasar la información a otras neuronas pasa por un filtro que llamamos “Peso”, el cual afecta el valor que recibe la siguiente neurona. En fin, este flujo de información crea un comportamiento único para cada combinación de “Pesos”.

Un aspecto aún en estudio es la estructura que debe tener esta red, esto es, si es posible tener una sola red que decida de forma muy eficiente el actuar de todo el sistema o si es mejor crear subsistemas de redes independientes. Éste es el centro de nuestro estudio, dada una situación en particular realizar un estudio de eficiencia entre dos modelos de redes neuronales artificiales, uno de manera monolítica o global y otro formado por subsistemas cooperativos.

Este concepto se basa en la idea de que un comportamiento complejo puede definirse a partir de varios comportamientos simples. Tal como sucede en la acción de encontrar una pelota dentro de una habitación se considera qué hacer al chocar con una pared, al tener a la vista o tocar la pelota, y durante la búsqueda cuando aún no se ubica la pelota, avanzar a la izquierda, a la derecha, al frente o atrás.

1.1 Objetivo

Realizar un estudio comparativo entre una estructura de red neuronal conformada por redes interdependientes o cooperativas, y una red neuronal con estructura monolítica.

1.2 Alcance

Generar un robot móvil autónomo en un ambiente simulado. Este robot debe contar con un mecanismo de movimiento con 4 ruedas, las dos traseras motoras y las delanteras sin motor, con sensores para medir la distancia y un mecanismo (pinza) para tomar un objeto.

La función del robot consiste en encontrar un objeto1 (pelota), tomar el objeto, buscar un contenedor (objeto2) y depositar el objeto1 (pelota) dentro de él.

1.3 Hipótesis

Un controlador para un robot autónomo utilizando redes neuronales artificiales, es más eficiente con una estructura de redes neuronales distribuidas o independientes, que utilizando una estructura de redes monolítica.

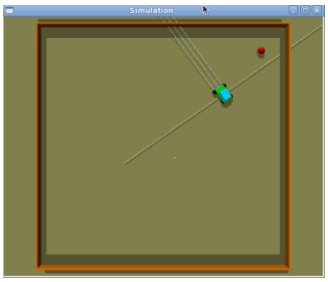


Figura 1.1 Modelo simulado

1.4 Conceptos Generales

El enfoque de esta tesis es el análisis del comportamiento de un robot simple usando redes neuronales artificiales, y realizar un comparativo del funcionamiento dadas dos estructuras de redes neuronales artificiales. Una de ellas monolítica, donde una sola red neuronal determina el comportamiento del robot. La otra estructura comprende la interrelación de varias redes neuronales, estas llamadas redes neuronales cooperativas.

El comportamiento del robot se puede definir a través de un “control” o “controlador”. Para generar este control neuronal se realiza un proceso de entrenamiento que consiste en determinar la diferencia o el grado de error entre el comportamiento que se está analizando del robot y el comportamiento deseado. Esto nos permite realizar algunos ajustes a los parámetros, hasta que se consigue un comportamiento aceptable.

Un controlador también es llamado “agente autónomo inteligente”. La denominación “agente”, en un sentido general es cualquier cosa capaz de percibir su medio ambiente con la ayuda de sensores y actuar en consecuencia utilizando actuadores. Tal como lo

hace una persona que utiliza sus ojos, nariz y oídos para percibir y piernas, brazos y manos para actuar.

El término percepción se utiliza para indicar que el agente puede recibir entradas o información de los sensores en cualquier instante. La secuencia de percepciones forma un historial completo que determina la acción a tomar por el agente. Al tomar cada decisión a partir del historial de percepciones se define una "función agente" que especifica la acción a realizar dada una percepción. [Stuart-Norvig 2008].

Esta "función agente" como una reacción sin un objetivo no genera una acción racional, o que se pueda aceptar como correcta, ésta depende de cuatro factores: [Stuart-Norvig 2008]

- Una medida del rendimiento que define el criterio del éxito.
- El conocimiento medio acumulado por el agente.
- Las acciones que el agente puede llevar a cabo.
- La secuencia o historial de percepciones del agente hasta ese momento.

Esto es, una respuesta "inteligente" por parte del agente depende de las reglas que indican lo bueno y lo malo, y de cómo influye todo su conocimiento previo.

Así pues un "Agente Autónomo Inteligente" se puede definir como un sistema computacional artificial independiente dotado de capacidades sensoriales y actuadoras propias, de tal forma que no requiere de supervisión por parte del usuario. Además de tener la capacidad de aprender de su propia experiencia. [Bellas 2003]

1.5 Inteligencia artificial y robótica

Uno de los usos principales de la inteligencia artificial es la creación de estas entidades autónomas artificiales con comportamientos inteligentes en un ambiente real, que permiten realizar actividades complejas o peligrosas y actividades tediosas o repetitivas.

Esta actividad o generación de entidades autónomas se lleva a cabo en combinación con la robótica, ya que ésta determina las bases físicas de la entidad en un ambiente real, mientras que la inteligencia artificial provee los mecanismos de control para lograr el objetivo final; es decir, el software de control o controlador; diseñado para generar determinado comportamiento del robot.

La implementación del controlador puede realizarse en base a diferentes teorías:

Subsumption architectures [1]

Es una forma de descomponer un comportamiento inteligente complicado en varios módulos de comportamiento "simple", que a su vez se organizan en capas. [Toal 1996] Cada capa implementa un objetivo particular del agente, y las capas superiores son cada vez más abstractas o generales. Por ejemplo, la capa más baja de un robot podría ser "evitar un objeto", en la capa superior estaría el concepto "ir de paseo", que a su vez se encuentra debajo de la capa "explorar el mundo". Cada una de estas capas del mismo nivel accede a todos los datos de los sensores y genera acciones para los actuadores. El inconveniente más importante es que las tareas por separado se pueden suprimir o anular entre ellas. De esta manera, las capas más bajas pueden funcionar como mecanismos de

[1] Subsumption architectures es un concepto acuñado por Rodney A. Brooks en 1986 del que no se encuentra una traducción literal.

respuesta rápida (como el reflejo de cerrar los ojos al ver algo que se acerca a la altura de la cara), mientras que las capas más altas se especializan en alcanzar el objetivo general [Toal 1996].

Teoría de Esquemas

Según esta teoría, los esquemas representan el conocimiento acerca de los conceptos: los objetos y las relaciones que tienen con otros objetos, situaciones, acontecimientos, secuencias de eventos, acciones y secuencias de acciones. Un ejemplo sencillo es pensar en el esquema para un perro. Dentro de este esquema el conocimiento general sobre los perros sería (corteza, cuatro patas, dientes, pelo, cola) y, la información acerca de perros específicos, tales como collies (pelo largo, grande) o springer spaniels (Inglés, cola cortada). También se puede pensar en los perros dentro de un contexto mayor de los animales y otros seres vivos; es decir, los perros respiran, necesitan alimento, y se reproducen. Su conocimiento de los perros también puede incluir el hecho de que son mamíferos y por lo tanto son de sangre caliente y tener sus crías en lugar de poner huevos. Dependiendo de su experiencia personal, el conocimiento de un perro como mascota (doméstica y leal) o como un animal peligroso (propenso a morder o atacar) puede ser una parte del esquema. Y así ocurre con el desarrollo de un esquema, cada nueva experiencia incluye más información.

Lógica Difusa

La lógica difusa es una lógica alternativa a la lógica clásica que introduce cierto grado de vaguedad a las cosas que evalúa. En el mundo real existe mucho conocimiento que es

ambiguo o impreciso por naturaleza. Este es un método que pretende imitar el razonamiento que utiliza el ser humano cuando se encuentra con este tipo de información.

Se utiliza cuando algunas variables de un sistema son desconocidas o no se pueden medir de manera confiable y cuando el ajuste de una variable puede ocasionar el desajuste de otras. Para ello utiliza umbrales o rangos en lugar de valores definidos y una propiedad o nivel de pertenencia que actúa en conjuntos difusos. [Zadeh 1994]

Sistemas Bayesianos

En estos modelos se obtiene la probabilidad de un evento, en función de sus probabilidades condicionales respecto de otros. Se basan en el Teorema de Bayes que calcula la probabilidad condicional de un evento aleatorio A dado B, en términos de la distribución de probabilidad condicional de B dado A. Y la distribución de probabilidad marginal de A. [Duda y otros 2001]

En la práctica, el teorema de Bayes se utiliza para conocer la probabilidad *a posteriori* de cierta variable de interés dado un conjunto de hallazgos. Las definiciones formales son las siguientes:

- Hallazgo. Es la determinación del valor de una variable, H = h, a partir de un dato (una observación, una medida, etc.).
- Evidencia. Es el conjunto de todos los hallazgos disponibles en un determinado momento o situación: $e = \{H_1 = h_1, \dots, H_r = h_r\}$

- Probabilidad *a priori*. Es la probabilidad de una variable o subconjunto de variables cuando no hay ningún hallazgo. La probabilidad *a priori* de $\bar{X}$ coincide, por tanto, con la probabilidad marginal $P(\bar{x})$.
- Probabilidad *a* posteriori. Es la probabilidad de una variable o subconjunto de variables dada la evidencia e. La representaremos mediante P^*:

$$P^*(\bar{x}) \equiv P^*(\bar{x}|e)$$

Sistemas Dinámicos

Los sistemas dinámicos son sistemas cuyos parámetros internos (variables de estado) siguen una serie de reglas temporales. Se llaman sistemas porque están descritos por un conjunto de ecuaciones (sistema), y dinámicos porque sus parámetros varían con respecto a una variable que generalmente es el tiempo.

Por ejemplo, se sabe que ciertas especies animales se multiplican, en condiciones ideales (comida en abundancia, sin luchas internas, etc.), de manera tal que el crecimiento de la población es proporcional a la cantidad de miembros de la especie. Es decir que si $P(t)$ denota la población en el tiempo t, y $P'(t)$ es la velocidad con que varía la población, entonces existe una constante $a > 0$, tal que $P'(t) = a\ P(t)$ para todo $t > 0$, donde 0 denota el tiempo a partir del cual comenzamos a hacer nuestra observación, y la constante a depende de la población en cuestión y del medio ambiente.

Metaheurísticas

Los algoritmos metaheurísticos son métodos de aproximación de propósito general para optimización y búsqueda. Son procedimientos iterativos que guían un comportamiento

heurístico o de descubrimiento para explotar o explorar determinado espacio de búsqueda imitando ciertos procesos naturales.

Estos algoritmos se utilizan en problemas llamado “difíciles” que debido a su naturaleza NP (Polinomial No determinístico, ver anexo A) no es posible encontrarles una solución factible en un tiempo “aceptable” con los recursos actuales. Se aplican a tareas como, encontrar el camino más corto entre varios puntos, el costo mínimo para el reparto de mercancías, la mejor ruta para un paquete en una red de datos, etc.

Se pueden clasificar de acuerdo a su similitud como basados en:

- La física: Recocido Simulado.
- La Biología: Redes Neuronales.
- La Evolución Natural: Algoritmos Evolutivos.

Las redes neuronales artificiales se utilizan en problemas donde se requiere una alta capacidad de generalización; esto es, a condiciones similares respuestas similares. A diferencia de lo que ocurre en un sistema de cómputo *estándar* como la arquitectura *Von Neumann* donde se utiliza el procesador central que se encarga de interpretar de manera secuencial y de manera estricta cada una de las instrucciones almacenadas, este procesamiento no permite generalizaciones implícitas, cualquier proceso que identifica similitudes entre cierta información debe programarse y considerar las semejanzas de cada caso para incluirse en el código.

Otra de las principales diferencias de la arquitectura de *Von Neumann* con las redes neuronales, es que el procesamiento se realiza en paralelo; es decir, existen muchas

unidades de procesamiento trabajando al mismo tiempo. Una diferencia importante es que la información o conocimiento que guardan las redes neuronales se encuentra distribuida entre los pesos de las conexiones. Esto presenta ventajas sobre el modelo de *Von Neumann* que aglutina la memoria en un solo lugar, una de ellas es la resistencia a fallas ya que la respuesta completa depende de la operación simple de cada uno de sus elementos y es posible que el sistema pueda seguir operando cuando una pequeña parte se dañe. Un uso importante en el reconocimiento de imágenes es la posibilidad de reconocer objetos a pesar de que sólo se tenga una parte de él o con una imagen distorsionada.

1.6 Descripción del trabajo

1.6.1 Introducción

Este trabajo se desarrolla en un entorno de simulación para implementar el agente robótico autónomo; como se mencionó anteriormente, la autonomía consiste en designar ciertos valores iniciales al agente para que realice su tarea, una vez hecho ésto el agente se desenvuelve solamente con los valores recibidos por los sensores y los mismos actuadores.

Nuestro agente móvil cuenta con un mecanismo de movimiento de cuatro ruedas, de manera inicial se considera que el par trasero es quien realice la tracción. Se tienen dos posiciones para el mecanismo de dirección, la primera es que el par de ruedas delanteras indiquen la dirección al rotar de manera paralela a la orden del agente. La segunda opción es que el par de ruedas traseras sea independiente y de acuerdo a la velocidad de cada una se generen los cambios de dirección, en este último caso el par de ruedas delanteras deben girar libremente en todos sentidos.

La recepción de datos del ambiente se hace a partir de sensores de distancia distribuidos al frente y a los lados del robot, éstos tienen una distancia máxima de detección para que exista la posibilidad de que el robot no detecte los objetos (pelota y contenedor), que estén fuera del rango del sensor, es decir el sensor no abarca toda la superficie de prueba.

Dentro de este ambiente es necesario definir un área plana donde se desplazará el robot, la superficie está rodeada por paredes que limitan el paso del mismo. Y se determina un tiempo límite para valorar si se logró el objetivo.

El robot debe localizar la posición de la pelota, ubicarse de frente a ésta y tomarla. Al tomar la pelota debe buscar la posición del contenedor y depositar la pelota dentro de él.

La posición de la pelota es aleatoria, es decir, para determinar si el robot es capaz de lograr su objetivo es necesario correr varias simulaciones ubicando la pelota en diferentes posiciones cada vez y estimar una media de eficiencia. Lo mismo sucede con el contenedor, una vez que el robot toma la pelota el robot debe encontrar el contenedor, a pesar de que éste se encuentra en la misma ubicación, la posición inicial y orientación del robot es diferente ya que comienza la búsqueda a partir de su posición y orientación al momento de tomar la pelota.

El robot cuenta con un mecanismo que le permite simular el tomar la pelota, este mecanismo es automático, es decir, se activa al ubicarse frente a la pelota y su función no depende de la red neuronal.

Para obtener la modularidad del comportamiento u objetivo global, éste se divide en varias funciones específicas que consideran a sensores y actuadores, cada función se asigna como comportamiento particular a una red neuronal específica. Estos comportamientos pueden ser:

Avanzar, Retroceder, Tomar Objeto, Colocar Objeto, Colocarse enfrente de objeto, Identificar Objeto, Voltear al reconocer el objeto, Seguir de frente al reconocer el objeto, etc.

Metodología

- Definir el ambiente de simulación adecuado para implementar el robot y su comportamiento, este ambiente debe ser versátil para soportar la programación de la red neuronal y el algoritmo de entrenamiento.
- Realizar el diseño del agente en el ambiente de simulación así como la estructura de la red neuronal monolítica que definirá el comportamiento.
- Desarrollar el algoritmo de entrenamiento, se considera más apropiado el método de entrenamiento a través de algoritmos genéticos debido a que la definición de la función de aptitud puede no ser derivable, lo cual es necesario para utilizar el método del descenso del gradiente.
- Definir y probar los diferentes parámetros de entrenamiento y la función que servirá para calificar la eficiencia del agente.
- Una vez que la estructura monolítica presenta un avance considerable en la consecución del objetivo es tiempo de realizar el diseño de la red cooperativa, es decir, utilizando los mismos parámetros realizar las pruebas comparativas.

1.6.2 Funciones del agente y utilidad

La función global a realizar por el agente autónomo es encontrar una pelota y ubicar un contenedor donde ponerla. La división más clara es separar las dos búsquedas, estos dos objetivos particulares presentan similitudes, mientras que la búsqueda de la pelota inicia en un punto fijo y termina en una posición aleatoria dentro de la superficie de prueba, la búsqueda del contenedor inicia en una posición aleatoria a partir de haber encontrado la

pelota y termina en un punto fijo. Por esto se toman estos dos subobjetivos como si fueran uno, es decir, al solucionar uno de ellos la solución del otro es similar y se engloban en un subobjetivo de búsqueda.

En este caso en particular el objetivo en realidad es la planeación de una ruta. Una de las características de nuestro robot es que no percibe el objetivo en todo momento por lo que no le es posible determinar una ruta desde el principio, el algoritmo de navegación es la clave para que cualquier robot móvil se pueda desenvolver en un entorno.

Existen dos grandes grupos de algoritmos de navegación [Avila 2008], los algoritmos basados en información completa o mapas del entorno donde van a moverse, y los algoritmos basados en información incompleta o reactiva. Estos últimos constituyen una de las técnicas más interesantes en robótica móvil, pues en la mayoría de los casos los robots no dispondrán de un mapa del entorno sobre el que se van a mover.

El hecho de que los algoritmos reactivos no posean información previa sobre el entorno implica que el robot deberá de tener un sistema de sensores suficiente y confiable, que permita al algoritmo de control “conocer el entorno”, a medida que se mueve el robot. El agente requiere tener información confiable y completa para poder generar acciones que cumplan un objetivo.

Otra de las características importantes de este tipo de técnicas es que se debe tener definida una política clara de actuación y decisión, es decir, qué hacer cuando cierto sensor esté activo, qué hacer si los sensores no devuelven información sobre el entorno, cómo incorporar la información histórica, etc.

La planeación de rutas utilizando algoritmos reactivos se puede clasificar en cuatro grupos de acuerdo a sus estrategias de control [Avila 2008]:

- Método de campos potenciales
- Teoría de grafos
- Inteligencia artificial convencional, como primero en profundidad y primero en anchura
- Técnicas inteligentes, como algoritmos genéticos y colonia de hormigas.

Sin embargo estas tres últimas estrategias requieren de la formulación de grafos o árboles de búsqueda para aplicar los algoritmos.

El método de campos potenciales se basa en la analogía de la fuerza magnética entre los elementos o las partículas. Se establecen las siguientes premisas.

- El robot se ve como una partícula con carga eléctrica.
- El espacio libre se considera un campo potencial.
- Los obstáculos se consideran con carga eléctrica del mismo signo del robot (se repelen).
- La meta tiene una carga eléctrica de un signo opuesto al robot (se atraen).

Con esta estrategia se pueden generar trayectorias en tiempo real a partir del campo de fuerza, las trayectorias generadas son suaves pero sólo crea una ruta como solución, que puede no ser la más corta. A pesar de que es necesario recrear los datos como un diagrama en nuestro caso se retoma el concepto de atracción y repulsión como parte del

entrenamiento, para que el robot se sienta atraído por la pelota y el contenedor en cada caso y que trate de esquivar las paredes.

Utilidad

La utilidad de un estudio de investigación recae en dos vertientes: primero, el sentido conceptual al reforzar una teoría, un proceso de investigación o un proceso para la solución de un problema en este ámbito. El segundo aspecto es la utilidad práctica de dicho conocimiento, ésto es, que sea aplicable en un entorno real como un proyecto de ingeniería, un mecanismo o solución para realizar una actividad de manera diferente y mejor utilizando la tecnología. Este último es el sentido de nuestro estudio.

Existen infinidad de mecanismos en ingeniería que utilizan sistemas de rastreo y detección de objetos, y dado el crecimiento de la capacidad tecnológica y por consiguiente la reducción de los costos de la misma, es factible crear soluciones a problemas que hace diez años era imposible concebir. Se plantean un par de casos para ejemplificar este hecho:

Dentro de una huerta los frutos de un árbol caen en periodos de tiempo aleatorio y una condición para la calidad del producto es que no permanezca demasiado en el piso ya que tiende a deteriorarse, por lo que se establecen lapsos de tiempo fijos para recolectar los frutos. Al automatizar el proceso se delimita un área bajo el árbol para que un vehículo automático busque y recolecte los frutos cada periodo de tiempo.

Al interior de una fábrica circulan de manera constante vehículos y personas, como medida de seguridad es necesario que la superficie del piso de los pasillos se encuentre

libre de objetos que puedan ocasionar un accidente. Para ello se diseña un sistema que utiliza los tiempos muertos de la planta un vehículo autónomo busque los objetos extraños y los retire.

En estos ejemplos simples, se puede ver cómo un módulo de búsqueda y recolección de objetos es parte esencial del sistema general que controla al agente autónomo. Esta es la intención práctica del presente estudio, mejorar el conocimiento en este tipo de aplicaciones al generar un controlador que utilice redes neuronales artificiales, que a diferencia de los modelos heurísticos presenta un mejor soporte a fallas de los sistemas que integran el agente como son los sensores o una diferencia de potencia en la tracción.

1.6.3 Unidad de Medidas a utilizar en la comparación y análisis

Una vez definidas las dos estructuras de red neuronal, es necesario identificar cuál de ellas presenta un mejor desempeño, para ello se realiza el análisis en dos etapas, la etapa de entrenamiento de cada una de las redes neuronales y la etapa de prueba donde se va a calificar a los mejores individuos de cada estructura.

Durante la etapa de entrenamiento se analiza el tiempo y la complejidad de los parámetros de cada red con la idea de igualar estos factores. Al unificar estos criterios se pretende que ambas estructuras de red neuronal posean características equivalentes en las que se pueda evaluar la diferencia en sus comportamientos durante la siguiente etapa.

La etapa de prueba es la de mayor importancia para la evaluación, dado que es quien determina si se llega o no al resultado deseado en mayor o menor proporción. Las

unidades a medir son el porcentaje de éxitos, la velocidad para llegar a estos éxitos y la capacidad de tolerancia a fallos.

- El número de éxitos entre una y otra estructura determinan en gran medida la eficiencia del control en óptimas condiciones
- La velocidad o el número de pasos promedio que se requieren para encontrar la solución es un factor importante para el usuario final del controlador.
- Finalmente es primordial considerar la posibilidad de fallas en los componentes del agente considerando que su aplicación es en la vida real, es por ello que se realiza la evaluación simulando fallas en uno de los sensores del agente.

1.7 Estado del Arte

1.7.1 Un Enfoque Híbrido en el Desarrollo de Robótica Basada en el Comportamiento

En este artículo se presenta el desarrollo de un método que utiliza la robótica evolutiva en conjunto con el de selección de acción, el segundo concepto hace referencia a que el robot debe seleccionar una acción entre una serie de acciones posibles. [Montes y otros 2010]. De manera que en una arena se implementa una tarea de recolección para el robot Khepera que debe recoger cilindros que simulan comida. Existen dos motivaciones que afectan a los módulos de conducta para realizar alguna acción, se denominan 'miedo' y 'hambre'. En este artículo se propone una etapa inicial de evolución donde se diseñan estos dos módulos conductuales para que puedan ser elegidos utilizando una selección de conducta, en esta etapa se divide un poco más el comportamiento en conceptos como "buscar la pared", "buscar la esquina" y "buscar el cilindro". Posteriormente se emplea evolución para optimizar la red de selección de acción entre las dos motivaciones. Finalmente, se compara el ajuste de selección obtenido mediante evolución artificial y mediante un diseñador humano, favoreciendo el uso de un enfoque híbrido en el desarrollo de robótica basada en el comportamiento.

Una de las similitudes principales con el presente trabajo es la utilización de comportamientos que son divididos en subobjetivos para realizar el entrenamiento evolutivo.

1.7.2 Control neural distribuido para robots autónomos complejos

El trabajo trata sobre la creación de redes neuronales artificiales para el control de robots complejos. Una técnica utilizada para ésto es la robótica evolutiva. Pero hasta la fecha, estos métodos no consiguen generar redes capaces de controlar robots complejos con muchos sensores y actuadores. En estos casos, no es posible generar tales redes debido al gran espacio de búsqueda al que se enfrenta el método de entrenamiento. Además, existe el problema de arranque (bootstrap), que impide la convergencia de las redes en las primeras generaciones hacia soluciones cada vez mejores. [Tellez 2010]

Este problema sucede cuando la función de eficiencia que utilizan los operadores genéticos, produce resultados muy bajos o nulos en todos los individuos de la población, lo que hace que las primeras etapas del proceso evolutivo avancen muy lentamente hacia la solución o en el peor de los casos no existan mejoras entre una generación y otra.

Dado que no es posible eliminar o reducir ninguna de las restricciones del robot a evolucionar (estas son, el cuerpo del robot, los sensores y actuadores utilizados, el comportamiento requerido del robot y el entorno en el que deberá trabajar el robot), su propuesta consiste en añadir conocimiento externo durante el proceso de evolución, siempre que ninguna de las restricciones se limite.

Desarrolla una arquitectura llamada DAIR (Distributed Architecture with Internal Representation) que permite la introducción de conocimiento externo en diferentes etapas del proceso de evolución, de manera altamente selectiva, por lo que es posible afectar sólo a algunas partes del control neuronal. Menciona que la propuesta permite

evitar el problema del bootstrap obteniendo un controlador altamente distribuido. En el trabajo se muestra cómo aplicar este control a un robot tan complejo como Aibo (con al menos 30 sensores y actuadores), en tareas como caminar, levantarse o aprender a tocar el suelo, todo ello en simulación y en el robot real.

Uno de los conceptos importantes que utiliza este trabajo de tesis, es la generación de redes modulares para realizar los controladores y la forma en que el entrenamiento genera una coordinación entre las redes para lograr el objetivo final.

1.8 Cinemática de un robot móvil

La cinemática se centra en el estudio del movimiento del robot en función de su geometría. Normalmente se consideran las siguientes limitaciones para la construcción del modelo cinemático:

- El robot se mueve sobre una superficie plana.
- No existen elementos flexibles en la estructura del robot (incluidas las ruedas).
- Las ruedas poseen uno o ningún eje de direccionamiento, de manera que éste último siempre es perpendicular al suelo.
- No se considera ningún tipo de fricciones en elementos móviles del vehículo, o contra el suelo.

El comportamiento cinemático se establece en el principio de que las ruedas en contacto con el suelo se comportan como una articulación plana de tres grados de libertad. Diferentes tipos de ruedas (tracción y dirección) tienen diferentes propiedades cinemáticas, figura 1.2. Los cuatro tipos de propulsión más utilizados son:

- Diferencial.- Se conforma de dos ruedas tractoras en un eje común (diferencial) donde cada una de estas ruedas es independiente y la variación de velocidades determina la dirección del vehículo. El resto de las ruedas son omnidireccionales.
- Triciclo.- Se utiliza una sola rueda delantera tractora que además puede girar para modificar la dirección. Las ruedas traseras son pasivas.
- Síncrono.- Se utiliza en robots de tres o más ruedas que se encuentran acopladas mecánicamente de forma que todas giran en la misma dirección a la misma velocidad.
- Carro (Dirección Ackerman).- Utilizada en los automóviles, cuenta con dos ruedas traseras con tracción sincronizada y dos ruedas delanteras con giro sincronizado para indicar la dirección.

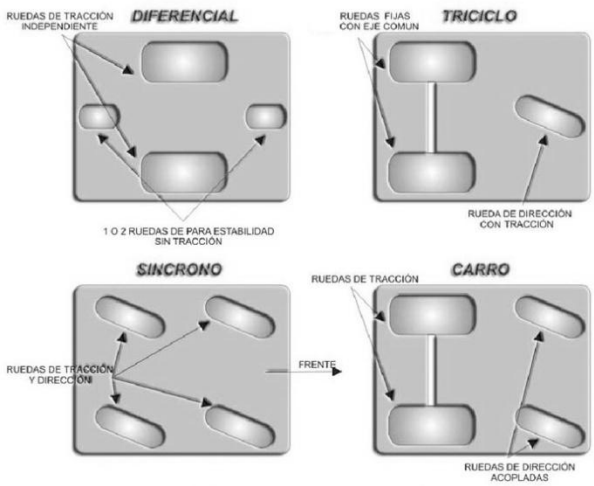


Figura 1.2 Tipos de propulsión

El vehículo robot que se diseña cuenta con propulsión diferencial, las ruedas traseras sobre el mismo eje son las encargadas de proporcionar la tracción del vehículo además de la dirección, también cuenta con dos ruedas delanteras pasivas omnidireccionales que permiten la estabilidad del agente. Figura 1.3

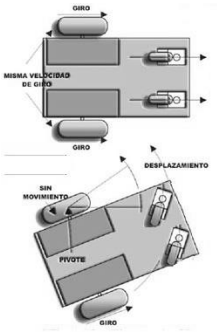


Figura 1.3 Propulsión del modelo

Se escoge una tracción diferencial ya que presenta un mayor grado de libertad al realizar giros que utilizando una propulsión de carro (dirección Ackerman) y reduce el número de elementos actuadores en el controlador. Figura 1.4

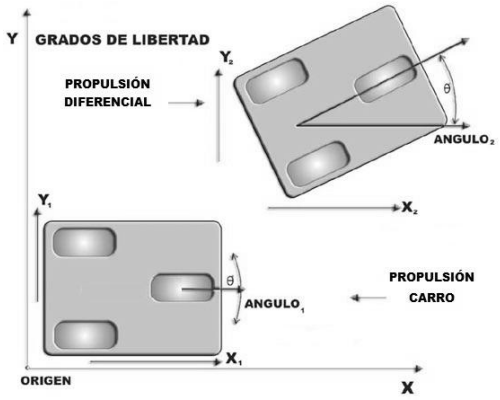


Figura 1.4 Grados de libertad. Propulsión de carro y diferencial

Capítulo 2. Redes Neuronales Artificiales

2.1 La Neurona Biológica

Las estructuras de los sistemas biológicos pueden proporcionar bases para el diseño de modelos de sistemas inteligentes. Estos procesan información de una forma no convencional, no requieren modelos de referencia, y se desempeñan de manera exitosa en presencia de incertidumbre; aprenden a realizar nuevas tareas y se adaptan con facilidad en ambientes cambiantes.

"*Se dice que un sistema tiene la capacidad de aprender si adquiere y procesa información acerca de su desempeño y del ambiente que lo rodea, para mejorar dicho desempeño*". [Anderson 2007]

En 1888 el científico Español Santiago Ramón y Cajal demostró que la estructura del sistema nervioso está conformada por una red de células individuales interconectadas entre sí, las neuronas, figura 2.1. Además de establecer que dentro de la neurona la información fluye desde las dendritas, pasa por el soma que es el cuerpo de la célula y llega hasta el axón para conectarse a otra neurona, es decir, que el flujo de la información es siempre en un mismo sentido. Este descubrimiento fue fundamental para el desarrollo de la neurociencia. [Ramón y Cajal 1906]

Desde un punto de vista funcional cada neurona actúa como un procesador con pequeño poder de procesamiento, donde las entradas al sistema son las dendritas y el axón las salidas. Sin embargo a diferencia de un procesador normal, las neuronas establecen relaciones con las neuronas contiguas traspasando información y trabajando en paralelo.

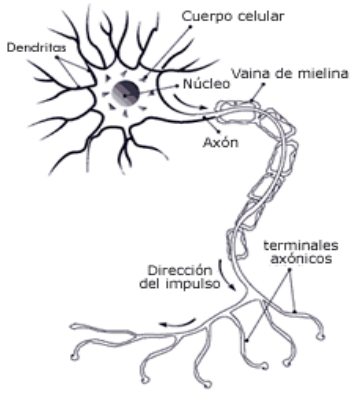


Figura 2.1 Neurona biológica

La unión o comunicación entre dos neuronas se llama *sinapsis.* Este es un proceso químico o eléctrico que va en un solo sentido, siempre en el mismo. Esta comunicación o falta de comunicación entre las neuronas contiguas es la que forma la estructura de la red. Existen neuronas presinápticas, las cuales envían los estímulos y neuronas postsinápticas que reciben estos estímulos.

La *sinapsis* puede variar en magnitud y duración, cada neurona se comunica con un número finito de neuronas postsinápticas y de igual manera puede recibir estímulos de un número limitado de neuronas presinápticas. La comunicación siempre es en un solo sentido, genera un flujo dentro de la red neuronal para generar un estímulo al final de ésta.

El aprendizaje de una red neuronal se debe a la variación que sufre la intensidad y duración de las sinapsis, esto es, durante la vida de la neurona ésta recibe unos estímulos sinápticos y genera otros. Las neuronas moderan su *sinapsis* en intensidad y duración para que ésta sea la que presente mejores resultados al sistema en general, de acuerdo a lo que han “aprendido”.

2.2 La Neurona Artificial

Esta línea de conocimiento se basa en la construcción de redes neuronales artificiales, esto es, un conjunto de neuronas y la forma en que se relacionan. Por ello, el primer elemento que es necesario definir es el modelo general de una neurona artificial.

"Se denomina procesador elemental o neurona a un dispositivo simple de cálculo que, a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta de salida". [Anderson 2007]

2.2.1 Modelo general de la neurona artificial

Los elementos principales de una neurona artificial se pueden ver en la figura 2.3, estos son:

- El Conjunto de entradas o vector de entradas $x_j(t)$: Representa todos los valores que recibe la neurona. De acuerdo al problema que se esté representando con el modelo neuronal puede ser un sólo valor o varios, de la misma forma puede recibir datos binarios o reales. Esta es la información que se recibe de otras neuronas o del exterior del sistema neuronal.
- Los pesos sinápticos w_{ij}: Indican la intensidad o fuerza de interacción entre las neuronas presinápticas "i" y postsinápticas "j", se representa como un factor que multiplica (para incrementar o disminuir la intensidad), a cada valor del conjunto de entrada cuando este viene de otra neurona de la red neuronal.

- La regla de propagación $\sigma\left(w_{ij}, x_i(t)\right)$: Proporciona el potencial postsináptico de la neurona, lo más común es utilizar la suma ponderada del producto escalar del conjunto de entradas y los pesos sinápticos $h_i(t) = \sum w_{ij} x_i$.
- La función de Activación $a_i(t) = f_i(a_i(t-1), h_i(t))$: Genera el estado actual de activación de la neurona en función de su estado anterior y su potencial sináptico actual. Las funciones más utilizadas son, de señal o escalón, lineal a tramos y sigmoidea. Figura 2.2

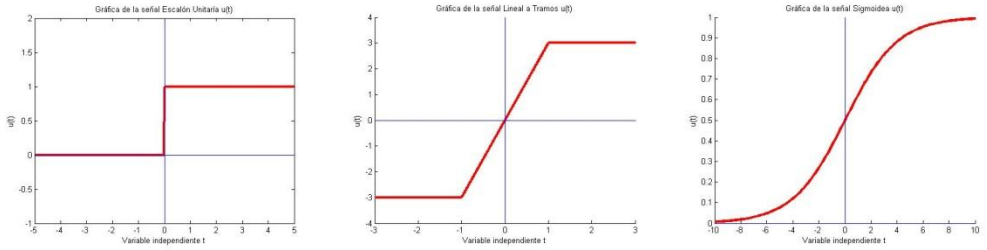


Figura 2. 2 Funciones de activación

- La función de Salida: Proporciona la salida de la neurona, normalmente es una función de identidad o se utiliza una función umbral para activar el valor de salida.

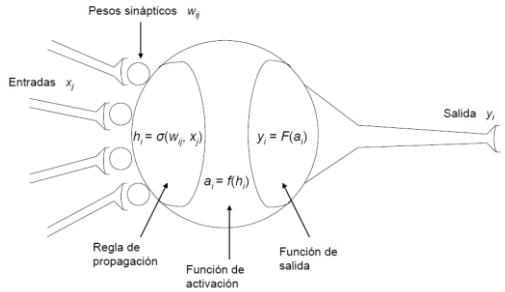


Figura 2.3 Elementos de una Neurona Artificial

Además de los elementos ya mencionados en algunas ocasiones se agrega al conjunto de pesos un parámetro adicional θ_i denominado umbral por lo que finalmente queda como:

$h_i(t) = \sum w_{ij} x_i - \theta_i$

Neuronas todo-nada

Este tipo de neuronas utilizan una función de escalón que implica la activación o no de la salida al sobrepasar un nivel mínimo o umbral. Este es el grado de libertad que agrega la utilización de θ_i. Por ejemplo se tiene la función de activación

$$h_i(t) = \begin{cases} 0, si \sum w_{ij}x_i \geq \theta_i \\ 1, si \sum w_{ij}x_i < \theta_i \end{cases}$$

Donde la neurona se activa cuando el potencial sináptico supera θ_i, de lo contrario permanece inactiva.

Neurona continua sigmoidea

Si el modelo neuronal puede recibir entradas tanto discretas como continuas y utiliza salidas continuas, puede recurrir a la función sigmoidea, continua y diferenciable en determinado parámetro.

$$y = \frac{1}{1+ e^{-x}} \ , \ y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = tgh(x)$$

Algunas reglas de aprendizaje como la retropropagación, requieren que la función sea diferenciable es por ello que utilizan estas funciones.

2.2.2 Arquitectura de las redes neuronales

La conexión entre las neuronas se realiza a través de las sinapsis, así que la configuración de éstas determina la estructura y el comportamiento de la red. La información en cada conexión viaja en un solo sentido como en una neurona biológica y la estructura general se divide en capas, existen tres tipos de capas, de entrada, de salida y ocultas. Figura 2.4

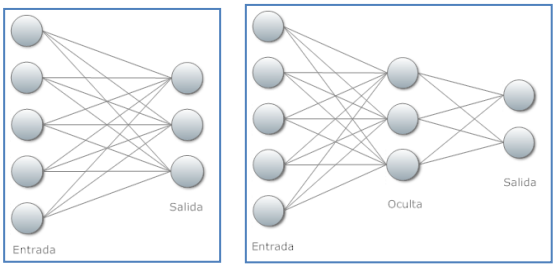


Figura 2.4 Redes Neuronales Dos Capas (Izq.), Con Capa oculta (Der)

Dependiendo del flujo de la información entre las capas la estructura de la red puede ser de dos tipos: La unidireccional o feedforward, donde la información va en un solo sentido, y las recurrentes o realimentadas, en la que el flujo puede ser en cualquier sentido ya sea entre las capas o dentro de la misma capa.

Las neuronas de la capa de entrada reciben información que proviene de afuera de la red. La capa de salida es, en sí, la respuesta que genera la red neuronal. La capa oculta no tiene ninguna comunicación con el exterior solamente con neuronas de la misma red, esta capa se utiliza para agregar grados de libertad a la red ya que de acuerdo a James A. Anderson, *"descubren maneras efectivas de representar la información de entrada".* [Anderson 2007]

Proceso de Aprendizaje

La operación de una red neuronal tiene dos fases muy características: La fase de ejecución o recuerdo y la fase de entrenamiento. La fase de entrenamiento es muy importante ya que una de las características principales de las redes neuronales es su capacidad de ser entrenadas o de aprender a partir de ciertos patrones.

El aprendizaje se explica como el ajuste de los parámetros que afectan el flujo de información de la red. El tipo de aprendizaje es determinado por la forma en que se van

ajustando estos parámetros, el más usual es el cambio en la intensidad de la sinapsis que consiste en ajustar los pesos a partir de una regla de aprendizaje, donde es necesario construir una función de error y a partir de ésta realizar la optimización de los pesos sinápticos.

Si se denomina a $w_{ij}(t)$ al peso que afecta la sinapsis entre las neuronas i y j, de la iteración t, el algoritmo de aprendizaje proporciona un valor $\Delta w_{ij}(t)$ en función de las salidas generadas y la corrección de la función de error, de manera que el nuevo peso en el instante $(t+1)$ se actualiza de la siguiente forma:

$$\Delta w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

Este proceso normalmente se repite una y otra vez hasta que la red neuronal alcanza un mínimo de error predeterminado.

Otros modelos de aprendizaje en lugar de modificar la intensidad de la sinapsis modifican la estructura de la red neuronal, esto es, durante el entrenamiento algunos modelos crean y destruyen sinapsis entre las neuronas creando un nuevo flujo de la información, incluso existen modelos que permiten la creación o destrucción de neuronas.

Los dos tipos de aprendizaje básicos son el supervisado y el no supervisado, la diferencia sustancial radica en la información que le presentamos a la red para modificar sus parámetros, de manera sencilla esta diferencia se puede expresar en el siguiente ejemplo: Deseamos que un niño aprenda a guardar sus juguetes, una opción es indicarle al niño el lugar de cada juguete y corregirle cada vez que cometa un error hasta que lo realice lo "suficientemente bien". La otra opción es darle los juguetes para que los guarde y

permitirle que él determine dónde poner cada juguete. La primera opción se utiliza cuando se conoce bien el resultado final; la segunda deja libre a la red neuronal para que relacione y clasifique, cuando se desean descubrir nuevos patrones o clasificaciones.

- Aprendizaje supervisado.- *Se presenta a la red un conjunto de patrones, junto con la salida deseada u objetivo, e iterativamente ésta ajusta sus pesos hasta que la salida tiende a ser la deseada, utilizando para ello información detallada del error que comete a cada paso.* [Del Brío y Sanz 2000]
- Aprendizaje no supervisado.- *Se presentan a la red multitud de patrones sin adjuntar la respuesta que deseamos. La red por medio de la regla de aprendizaje estima p(x), a partir de lo cual pueden reconocerse regularidades en el conjunto de entradas, extraer rasgos, o agrupar patrones según su similitud.* [Del Brío y Sanz 2000].

Los algoritmos para resolver el problema del aprendizaje se diferencian en cómo determinan $\Delta w_{ij}(t)$, entre los más importantes se encuentran:

- Corrección de error.- En estos algoritmos se ajustan los pesos en proporción a la diferencia que existe entre la salida actual de la red y la salida deseada, con la intención de minimizar el error actual de la red. Aquí se incluyen métodos como el de mínimos cuadrados y retropropagación.
- Boltzman.- Este modelo se aplica en redes recurrentes de operación binaria. Resalta la utilización de un parámetro adicional denominado energía o temperatura que se va reduciendo mientras avanza, esto le permite describir una trayectoria errática que abarca un área o espacio de estados con la utilidad de poder salvar con ello algunos mínimos locales.

- Regla de Hebb (Delta).- Fue el primer postulado de cómo aprenden las neuronas (Hebb, 1994): *Cuando el axón de una neurona A está lo suficientemente cerca de la neurona B, y repetida y persistentemente ayuda a su disparo, se produce un proceso de crecimiento o cambio de metabolismos en una o ambas neuronas, de tal manera que la eficiencia de A, como una de las neuronas que disparan B, se incrementa.*
- Competitivo.- En estas redes la neuronas compiten entre sí para ser la única ganadora, una sola neurona se activa en cada iteración. Se utiliza cuando se desea descubrir características importantes, patrones o agrupamientos.

El Perceptrón

El perceptrón es la primera red neuronal que tuvo un éxito significativo, fue propuesta por Frank Rosenblatt a finales de los años cincuenta. La estructura del perceptrón simple consiste en dos capas, una sensorial o de entrada que se conecta a otra capa llamada de salida. Las neuronas de entrada no realizan ningún cómputo, únicamente envían la información a las capas de salida. La función de activación de las neuronas de la capa de salida es de tipo escalón.

Una neurona tipo perceptrón solamente permite dividir dos clases linealmente separables (patrones que se encuentran en lados opuestos de un hiperplano, también conocido como dicotomía). Figura 2.5

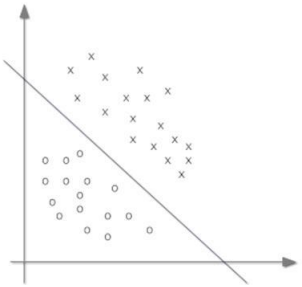


Figura 2.5 Regiones de decisión en el plano

Dada una neurona tipo perceptrón de dos entradas x_1 y x_2, con una salida y de tipo binario, su operación se define como: $y = H(w_1x_1 + w_2x_2 - \theta)$

$$y = \begin{cases} 1, si\ w_1x_1 + w_2x_2 \geq \theta \\ 0, si\ w_1x_1 + w_2x_2 < \theta \end{cases}$$

Si se considera x_1 y x_2 situadas sobre los ejes de abscisas y ordenadas en el plano, la condición $w_1x_1 + w_2x_2 - \theta = 0 \Rightarrow x_2 = -\frac{w_1}{w_2}x_1 + \frac{\theta}{w_2}$, representa una recta que divide el plano en dos regiones, para las que la neurona proporciona una salida 0 o 1 respectivamente. Entonces, una neurona tipo perceptrón representa un discriminador lineal, al implementar una condición lineal que separa dos regiones en el espacio, que representan dos diferentes clases de patrones.

Ejemplificando la función NAND, representada en el plano (Figura 2.6), en este caso pueden encontrarse unos parámetros $w_1, w_2\ y\ \theta$ que determinen una recta que separa perfectamente las regiones correspondientes a los valores lógicos 0 y 1. Por ello se dice que la función NAND es linealmente separable.

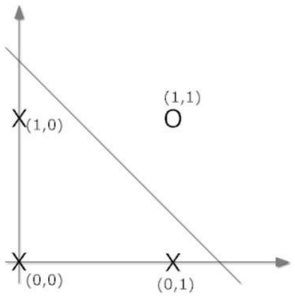


Figura 2.6 Función lógica NAND

Sin embargo, si se toma la función lógica XOR y se representa en el plano (Figura 2.7), vemos claramente que en este caso no es posible encontrar una única función lineal que separe las regiones correspondientes a los valores 0 y 1, XOR no es linealmente separable y por tanto no es posible representarse con un perceptrón simple.

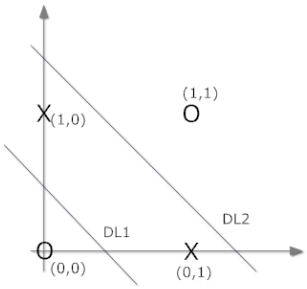


Figura 2.7 Función lógica XOR

Una solución a este problema debe tomar en cuenta que una neurona tipo perceptrón genera sólo una solución lineal, al ver la figura 2.7 se puede considerar el utilizar dos neuronas perceptrón, una desarrolla la solución DL1 y otra la DL2. También supongamos una capa adicional, compuesta por una neurona perceptrón encargada de componer o acomodar las regiones en las que queda dividido el plano por las dos neuronas DL1 y DL2. Si esta neurona se activa sólo cuando la neurona correspondiente a DL1 está activada y DL2 está desactivada, se tendrá una red de tres capas (una de ellas oculta), que soluciona

la función XOR, al incluir capas adicionales tendremos la definición de un perceptrón multicapa.

A partir del concepto anterior se pueden definir tres tipos de regiones de decisión que pueden formarse utilizando estructuras simples o multicapa.

- Mediante una estructura de dos capas sin capa oculta la región de decisión es un hiperplano que separa en dos el espacio de las variables.
- Utilizando una estructura de tres capas, una de ellas oculta, se pueden separar regiones convexas, sean cerradas o abiertas.
- Finalmente con una estructura de cuatro capas, dos de ellas ocultas, se tiene la capacidad de separar regiones arbitrarias, donde el límite es el número de nodos utilizados.

Redes Neuronales Realimentadas o Recurrentes

Las redes neuronales recurrentes son aquellas que tienen uno o más lazos de retroalimentación. Esta retroalimentación puede ser local, es decir, la neurona se retroalimenta a sí misma, o global, cuando la neurona realimenta neuronas de la misma capa o capas anteriores, figura 2.8.

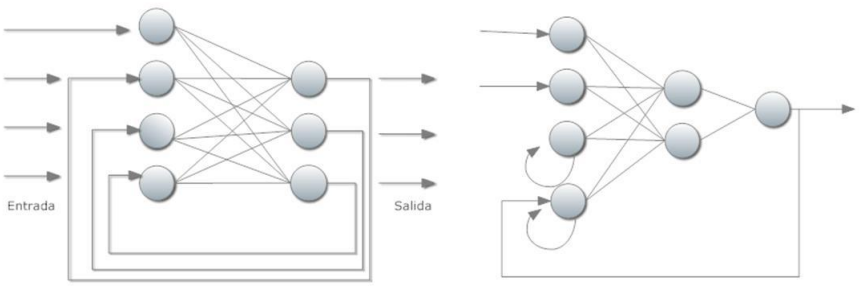


Figura 2.8 Red con retroalimentación global (izq) y Red con retroalimentación global y local (der)

Una de las características más importantes que justifican el uso de las redes retroalimentadas es su capacidad de poder representar un espacio de estados. Esto los hace convenientes para múltiples aplicaciones como son los sistemas de control.

Capítulo 3. Algoritmos Genéticos

El concepto de algoritmos genéticos se basa en las reglas de evolución que desarrolló Charles Darwin sobre el comportamiento que presenta una población dentro de su ecosistema. Lo que se conoce de manera común como “Selección Natural”. El desarrollo de esta técnica se debe en gran parte a Jon H. Holland que en 1975 implementa los algoritmos genéticos imitando el proceso de selección natural y utilizando los estudios de Mendel sobre la transferencia de genes. [Holland 2001].

“La lucha por la existencia de un ser orgánico contra otros seres orgánicos”, puede ser una definición muy simple del concepto de selección natural.

Un concepto importante para entender esta teoría es que “Entre los seres orgánicos existe variabilidad individual”, esto es, cada ser orgánico presenta alguna diferencia con cualquier otro organismo aún de su misma especie, sin importar la causa que la haya generado. Como ejemplo podemos tomar dos cachorros que a pesar de provenir de la misma madre y que a simple vista puedan parecer iguales, uno de ellos puede tener mayor vigor o ser más curioso que el otro.

Dado que los organismos se encuentran en constante lucha, estas variaciones, por ligeras que sean, pueden de alguna manera ser benéficas para el individuo en su relación con otros individuos y sus condiciones físicas de vida.

De ser así, los individuos con esta variación en sus características sobreviven y heredan este cambio a su descendencia, que tiene más posibilidades de sobrevivir, dado que, de los individuos que nacen en una población generalmente sobreviven los mejor adaptados.

Por lo que el principio sería; “Si la variación es útil, se conserva”.

“Metafóricamente puede decirse que la selección natural está buscando cada día y cada hora por todo el mundo las más ligeras variaciones; rechazando las que son malas; conservando y sumando todas las que son buenas; trabajando silenciosa e insensiblemente, cuandoquiera y dondequiera que se ofrece la oportunidad, por el perfeccionamiento de cada ser orgánico en relación con sus condiciones orgánicas e inorgánicas de vida.” [Darwin 1859]

Además de la sobrevivencia otro aspecto de la selección natural es la selección sexual. La podemos definir como; “La lucha entre individuos de un sexo en relación a la posesión de otro a fin de procrear”. [Darwin 1859]

En este caso vencer no depende de estar mejor adaptado para sobrevivir, sino de tener mejores armas para combatir con individuos de su misma especie o para convencer al sexo opuesto.

Un ciervo sin cuernos, un gallo sin espolones, tienen pocas posibilidades de dejar numerosa descendencia. En muchas especies de aves, los machos atraen a la hembra cantando, desplegando su pecho y exhibiendo de la mejor manera su esplendoroso plumaje, además de ejecutar movimientos a modo de baile ante las hembras, que son finalmente quienes escogen al compañero más atractivo.

Como ya se mencionó la teoría de la Selección Natural sirve como base para el concepto de algoritmo genético. Se toman los conceptos principales para su aplicación a problemas matemáticos, esto es:

Dado un ecosistema

- Los individuos que mejor se adaptan a las condiciones que les presenta el ecosistema tienen más posibilidades de reproducirse.
- Los menos adaptados también se reproducen aunque tienen menos posibilidades de hacerlo.
- Dados 2 padres, los genes o características de los hijos se obtienen como una combinación de los genes o características de los padres.
- Periódicamente la naturaleza envía una catástrofe o mutación que cambia las características de algunos individuos.

En general un algoritmo genético está compuesto por 5 elementos, resumidos por Zbigniew Michalewicz en 1999, [Michalewicz 1999].

- Un individuo como una representación genética de la solución del problema
- Un camino para crear una población inicial de soluciones
- Una función de evaluación de la calidad de las soluciones en términos de su eficiencia.
- Operadores genéticos tal que intercambien componentes genéticos en los hijos durante la reproducción
- Valores para los parámetros del algoritmo genético

El desarrollo de un algoritmo genético representa la conceptualización de varios elementos, cada algoritmo genético mantiene una población de individuos, cada uno de estos individuos representa una posible solución al problema planteado.

Cada individuo es evaluado en su comportamiento para detectar y seleccionar las mejores soluciones hasta el momento. Algunos de estos individuos se transforman de manera estocástica utilizando operadores genéticos, con el fin de generar nuevos individuos, específicamente son dos tipos de transformaciones, mutación y cruce. Estos serán los individuos que conformarán la siguiente generación. Figura 3.1

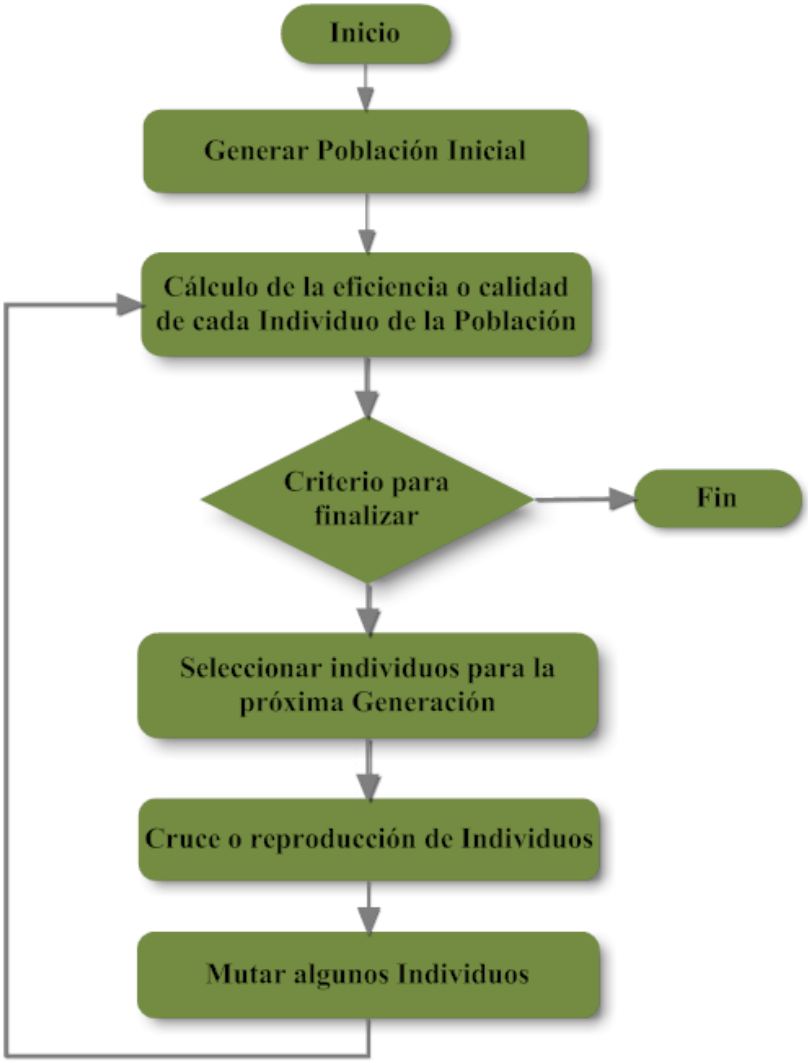


Figura 3.1 Diagrama de Flujo de un Algoritmo Genético

3.1 Selección

Los individuos que pasan a la siguiente generación, como en la selección natural, son los mejor adaptados. De acuerdo al resultado de la función de evaluación los individuos tienen mayor o menor probabilidad de continuar, incluso los individuos que tienen una mala calificación pueden pasar a la siguiente generación pero con mucha menor probabilidad que quienes obtuvieron las mejores calificaciones.

Para seleccionar a los individuos que van a pasar a la siguiente generación se tiene diferentes algoritmos, los más utilizados son Ruleta y Torneo:

La ruleta consiste en asignar la probabilidad según sea la calificación del individuo, dando mayores porcentajes a quienes obtuvieron mejores calificaciones.

Es un algoritmo donde el resultado más alto de la función de evaluación representa una mejor solución, se realiza una suma acumulada de los valores de cada individuo y se asigna a los individuos la proporción que corresponde de su resultado sobre el total de la población. En analogía con el juego de la ruleta cada individuo es una rebanada de la circunferencia, entre mejor calificado más grande es la rebanada. Tabla 3.1 y figura 3.2

Número de Individuo	Función de Evaluación	Acumulado	%
1	158	158	11.1
2	245	403	17.2
3	456	859	32.0
4	79	938	5.6
5	485	1423	34.1

Tabla 3.1 Tabla acumulativa en selección por ruleta

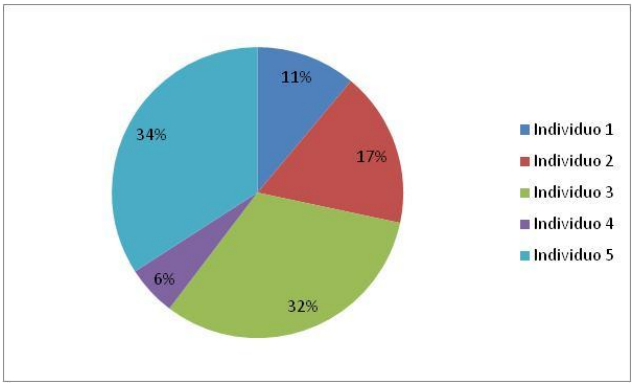


Figura 3.2 Gráfica de tabla 3.1 en selección por ruleta

Una vez definida la ruleta se tira la pelota, es decir, se genera un número aleatorio entre 0 y el total acumulado (ó entre 0 y 1); el individuo al que le toque el número es seleccionado. Figura 3.3

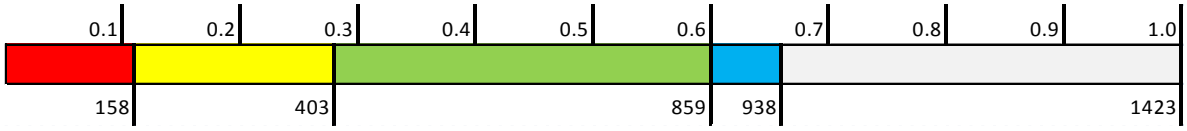
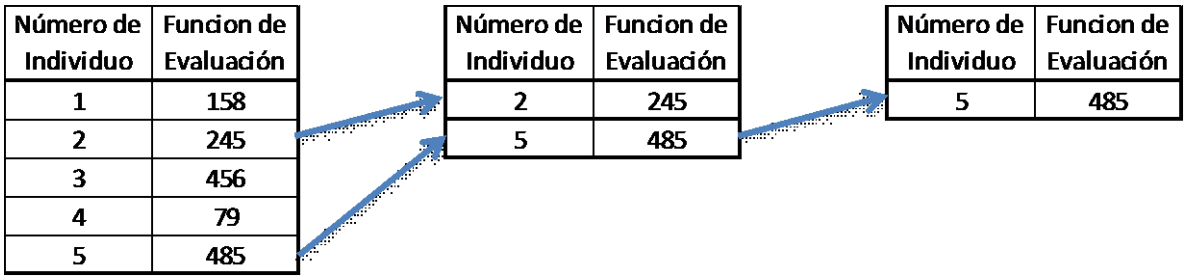


Figura 3.3 Asignación de probabilidad en selección por ruleta

El Torneo como su nombre lo indica es la competencia entre dos o más individuos por pasar a la siguiente generación, figura 3.4. El tamaño del torneo determina el número de individuos que participarán, estos se seleccionan aleatoriamente de toda la población. Identificados los individuos se selecciona el que tenga una mejor evaluación.



Número de Individuo	Funcion de Evaluación
1	158
2	245
3	456
4	79
5	485

Número de Individuo	Funcion de Evaluación
2	245
5	485

Número de Individuo	Funcion de Evaluación
5	485

Figura 3.4 Selección por torneo

Los dos algoritmos coinciden en la posibilidad de que los individuos más adaptados no pasen a la siguiente generación. Una diferencia importante entre ambos radica en que, durante el torneo, el individuo con la calificación más baja es eliminado, mientras que en la ruleta éste tiene solo la menor probabilidad de ser seleccionado.

3.2 Cruce

El cruce consiste en crear nuevos individuos combinando las partes de dos individuos progenitores, figuras 3.5, 3.6 y 3.7. La selección de individuos para este proceso se realiza de manera aleatoria, al igual que la forma en que se realiza la combinación.

Existen dos tipos de valores que pueden contener los “cromosomas” de cada individuo, reales o binarios, y para cada uno es diferente el algoritmo a utilizar.

Durante un cruce de valores binarios se hace la selección de un punto de cruce, a partir de este punto se hace un intercambio de valores.

Si tenemos dos individuos

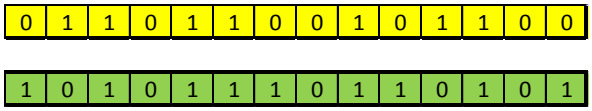


Figura 3.5 Cromosomas binarios de dos individuos

Se selecciona un punto al azar

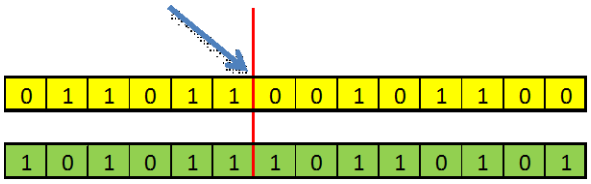


Figura 3.6 Selección de un punto aleatorio para el cruce

A partir de este punto se intercambia el material genético, generando dos nuevos individuos que sustituyen a los originales dentro de la población.

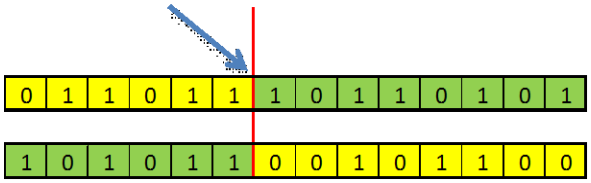


Figura 3.7 Intercambio de cromosomas

Para realizar el cruce de valores reales se utiliza un factor aleatorio **λ** que realiza la función del punto de cruce. En este proceso cada cromosoma en lugar de ser un número binario es un número real, es por ello que vamos a realizar el cruce por cromosoma y no por el individuo completo como al utilizar valores binarios. [Jamshidi 2002]

El proceso es generar un número aleatorio **λ** entre 0 y 1 (por ejemplo 0.23) y aplicar la siguiente ecuación a los cromosomas C1 y C2.

$$C1 = \lambda\, C1 + (1-\lambda)C2$$

$$C2 = \lambda\, C2 + (1-\lambda)C1$$

Esta ecuación nos permite generar dos nuevos individuos conservando ciertas características de los individuos originales, en nuestro caso, que los valores se mantengan en la tendencia que marcan los dos individuos, por ejemplo:

$C1 = 6{,}756 \quad C2 = 11{,}189 \quad \lambda = 0.23$

$C1 = (0.23)(6{,}756) + (1-0.23)(11{,}189) = 10{,}169.41$

$C2 = (0.23)(11{,}189) + (1-0.23)(6{,}756) = 775.59$

Como se puede ver los nuevos valores quedan dentro del intervalo que representan los valores originales, figura 3.8, lo que nos da la seguridad de que los nuevos individuos pertenecen a una solución semejante a las anteriores.

Figura 3.8 Operador de cruce utilizando valores reales

Se realiza el intercambio de valores para cada uno de los valores reales o “cromosomas”.

Para ambos casos se establece una probabilidad de cruce donde se especifica el porcentaje de la población que va a ser modificada en cada generación, esto quiere decir que algunos individuos pasan a la siguiente generación sin “cruzarse”, o ser modificados.

3.3 Mutación

La mutación produce cambios aleatorios en las características o “cromosomas” del individuo, que permiten preservar la diversidad genética de la población, esto no garantiza que los cambios generen una mejor solución que la original, el objetivo de esta mutación es evitar que en algún momento la solución a la que nos estamos acercando a través del cruce sea una solución local. La función de Weierstrass es un ejemplo de un problema con múltiples mínimos locales. Figura 3.9

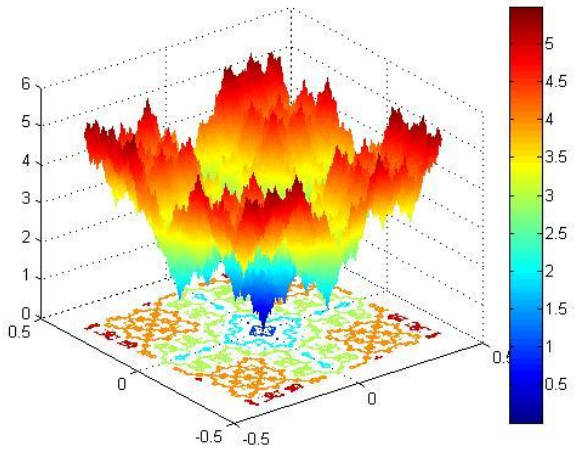


Figura 3.9 Función de Weierstrass, presenta múltiples óptimos locales

En una función con un espacio de soluciones tridimensional podemos encontrar mínimos locales donde el algoritmo se estanque al suponer que no existe una mejor solución, ya que en los alrededores (en el ámbito local), no la hay.

Mutar a un individuo es hacerlo dar un "salto" aleatorio dentro de nuestro espacio de soluciones, con esto se hace un rastreo para identificar soluciones que puedan ser mejores fuera del entorno local hacia donde convergen los individuos, de no ser así, se obtienen resultados con menor calidad y en las próximas generaciones serán reemplazados.

El procedimiento de mutación utilizado modifica uno de los genes o cromosomas del individuo, para ello se determina aleatoriamente:

- El gen a mutar
- La dirección del cambio; si el nuevo valor será mayor o menor al original.
- Se elije un factor (entre 0 y 1) que indica la proporción a acumular.

El proceso consiste en seleccionar a un individuo de la población total (cualquiera de los individuos puede mutar), La mutación se aplica después del cruce y normalmente a un porcentaje muy bajo de la población.

Cuando se utilizan genes binarios, se selecciona el gen a modificar y se intercambia el valor que contiene. Figura 3.10

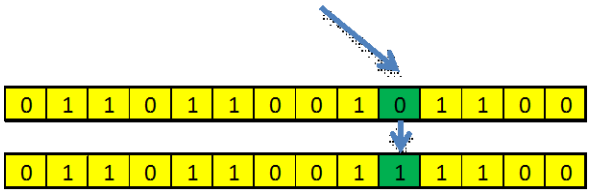


Figura 3.10 Mutación con valores binarios

Cuando los genes son reales, se requiere de dos factores más que indiquen la dirección (α) y proporción del cambio (Δ).

$$Para\ \alpha < 0.5 \quad C1 = C1 + (C1 * \Delta)$$

$$Para\ \alpha \geq 0.5 \quad C1 = C1 - (C1 * \Delta)$$

$$\Delta = \{0, 1\} \quad \alpha = \{0, 1\}$$

También es común que el porcentaje de mutación se reduzca conforme pasa cada generación esto reduce el salto por mutación con cada generación hasta que se elimina.

$$Para\ \alpha < 0.5 \quad C1 = C1 + \beta(C1 * \Delta)$$

$$Para\ \alpha \geq 0.5 \quad C1 = C1 - \beta(C1 * \Delta)$$

$$\Delta = \{0, 1\} \quad \alpha = \{0, 1\} \quad \beta = No.\,de\ Generación$$

3.4 Función de Aptitud

La función de aptitud se puede definir como “La forma de calificar al individuo, que tan bueno es el trabajo que realiza”.

Holland explica la función de aptitud en términos de la capacidad del individuo de influir en el futuro de la población. “La capacidad de un fenotipo se basa en el número de crías que sobreviven y se reproducen”. [Holland 2001].

El resultado de la función de aptitud determina la posibilidad de un individuo de ser seleccionado y poder reproducirse, éste es un factor muy importante dentro del algorítmo. Esto quiere decir que una mala definición de la función de aptitud puede llevar la búsqueda a puntos muy lejanos al óptimo.

Capítulo 4. Neuro-Evolución

4.1 Modularidad

4.1.1. Introducción

En general se considera a un sistema de cómputo de estructura modular, si se puede dividir en dos o más subsistemas donde cada subsistema evalúa las entradas, sean las mismas o no, sin comunicarse con otros subsistemas. La salida general del sistema se realiza a partir de la integración o combinación de las salidas de cada subsistema para generar una salida común que resuelva el objetivo global. Desde otro enfoque "*La modularidad puede ser vista como la existencia de un subconjunto de variables en un sistema, que pueden ser optimizadas de manera independiente del resto de las variables del sistema".* [De Jong 2004].

En realidad la modularidad es una rama del principio "divide y vencerás" mediante el cual, una tarea de cómputo puede dividirse en tareas simples, tareas que pueden realizarse por sistemas locales o especializados. La solución final a la tarea se logra combinando los resultados de los subsistemas especializados que realizaron su tarea de manera independiente y más cercanos al óptimo.

4.1.2 Definiciones

Una red neuronal se dice que es modular si el trabajo computacional de la red puede ser descompuesto en dos o más módulos (subsistemas) que operan con distintas entradas sin comunicarse entre ellos. Las salidas de los módulos se realizan mediante una unidad de integración que no permite que la información regrese a los módulos. Específicamente la

unidad de integración decide cómo se combinan los módulos para formar la salida final del sistema y cuáles módulos deben aprender los patrones de entrenamiento. [Jordan y Jacobs 1991].

De manera general un sistema modular presenta las siguientes características:

- Los módulos responden a dominios específicos y se han realizado arquitecturas computacionales especiales para identificar y responder a ciertos dominios de la tarea general.
- Cada módulo normalmente es independiente de otros módulos en su funcionamiento y no influye o es influido por otros módulos.
- Cada módulo generalmente tienen una arquitectura más sencilla en comparación con el sistema en su conjunto. Así, un módulo puede responder más rápido que un sistema monolítico mucho más complejo.
- Las respuestas de los distintos módulos son simples y tienen que ser combinados por parte de algunos mecanismos de integración a fin de generar la respuesta general del sistema.

Sin embargo, el realizar la descomposición de una tarea no quiere decir que las sub-tareas o subsistemas identificados, sean totalmente independientes unos de otros. De hecho, un sistema puede ser modular y todavía contener interdependencias entre sus módulos. Por lo que una tarea que pueda ser fraccionada, se puede descomponer en otras sub-tareas, pero la solución óptima de algunas depende de la solución óptima de otras sub-tareas.

4.1.3 Evidencias Biológicas

De a cuerdo a Michael A. Arbib [Arbib 1995], no hay evidencia física de que el cerebro este masivamente distribuido en paralelo y de que sea un sistema modular auto-organizado, lo que sugiere que el cerebro está compuesto por módulos muy especializados que

funcionan más o menos de manera independiente uno de otro, y se organizan en una estructura jerárquica con diferentes niveles y tipos de modularidad, incluyendo modularidad en paralelo y en serie. Estas evidencias incluyen conectividad entre las neuronas dispersas en el cerebro, donde grupos de redes forman "clusters", o módulos funcionales, que por ejemplo se encuentran en el sistema de visión, donde estos módulos especializados realizan funciones de contraste, movimiento y detección de color.

4.2 Neuroevolución

La neuroevolución utiliza el concepto de "selección natural" para el desarrollo de los parámetros que requiere una red neuronal para lograr su objetivo. De acuerdo con Richard Dawkins. "*La selección natural es un relojero ciego; ciego porque no ve el más allá, no planifica las consecuencias, no tiene una finalidad en mente. Aun así, los resultados vivos de la selección natural nos impresionan de forma irresistible, por su apariencia de haber sido diseñados por un maestro relojero, nos impresionan con la ilusión del diseño y la planificación".* [Dawkins 1986]

Esto es, la principal dificultad de una red neuronal está en determinar el valor de los pesos sinápticos correctos para concretar un objetivo, esta tarea en ocasiones es muy compleja si deseamos utilizar fuerza bruta (probar todas la posibles combinaciones). Así que el uso de los algoritmos genéticos es una opción que nos permite obtener una solución "aceptable" sin tener que "planear" o "diseñar" la cadena de pesos correctos.

El método consiste en tomar cada uno de los pesos sinápticos de toda la red neuronal y formar con ella una cadena de cromosomas. Se generan varios "individuos", con

diferentes cadenas de cromosomas llenados de manera aleatoria y se realiza la evaluación de la red neuronal con esta lista de pesos. Los operadores genéticos transforman las cadenas a partir del resultado de la evaluación para conservar los mejores y evitar los óptimos locales, Se repite el proceso de evaluación sobre las cadenas resultantes el número de veces (generaciones) de acuerdo a un límite de error deseado o un número fijo de ocasiones.

Utilizar operadores genéticos no garantiza obtener la mejor solución al problema ya que no se comprueban la totalidad de las posibilidades, por lo que se considera sólo como una “buena aproximación” a la solución.

Capítulo 5. Diseño del Modelo

5.1 Entorno de simulación ODE

El objetivo del presente trabajo va encaminado a evaluar la funcionalidad de diferentes estructuras de redes neuronales, específicamente dos tipos de estructuras una monolítica y una segunda conformada por varias redes neuronales que trabajen en conjunto o "colaboren" entre ellas para el logro del objetivo. Por ello, la construcción del modelo de robot real donde estas redes se apliquen no está dentro del ámbito de la tesis.

En la actualidad la construcción de un robot se desarrolla normalmente utilizando modelos simulados, a partir de la funcionalidad de éstos se realiza la fabricación del modelo real lo que ahorra mucho tiempo en la detección y corrección de fallas.

El software de simulación que se utiliza para realizar las pruebas requiere de modelos de simulación física. El objetivo de la simulación física es imitar las condiciones del mundo real, para ello se han desarrollado lo que de manera común se llaman "motores físicos", estos son componentes de software o utilerías que permiten introducir las leyes de la física en el comportamiento de los objetos creados en ese entorno, como por ejemplo la gravedad y la inercia.

La programación de una simulación física es sólo una aproximación a la física del mundo real dado que existen demasiados elementos a considerar en la realidad para poder recrearla completamente. Un ejemplo de las limitaciones de una simulación es la necesidad de utilizar valores discretos que permitan a la computadora realizar los cálculos, mientras que en la realidad es normal que los valores sean continuos.

Los dos componentes principales que forman un motor de física son: Un motor físico, usado para aplicar las leyes de la mecánica de Newton, y un detector de colisiones, que se utiliza para determinar cuando dos o más objetos físicos se cruzan e implementar la respuesta correspondiente.

5.1.1 ODE

Se decide utilizar la API, Open Dynamics Engine (ODE), es un software libre y de alta calidad para simular la dinámica de cuerpos rígidos y articulados. De acuerdo a las referencias se le considera rápida, flexible y robusta. Implementa además la detección de colisiones. El desarrollo de ODE lo inicia Russell Smith desde 2001. Es un conjunto de librerías y estructuras de datos escritos en lenguaje C++ que realizan los cálculos de las leyes de Newton sobre los cuerpos simulados.

El simulador representa a los modelos como cuerpos rígidos que no sufren deformaciones ni pueden ser penetrados. En la realidad este tipo de objetos no existe, sin embargo limitar o ignorar estas características en los modelos simplifica de manera significativa el trabajo de cómputo.

La simulación física consiste en una serie de fotografías consecutivas, en cada marco o fotografía se mantiene el estado actual de posición, velocidad, rotación, inercia, etc., de cada objeto del sistema, estos marcos tomados en serie forman el comportamiento dinámico del sistema en el tiempo. El lapso entre dos marcos se llama fase de transición, la tarea del motor de simulación física es calcular el estado de cada objeto en el siguiente marco. El lapso de tiempo (o paso), entre cada marco es regulado por el programador, es

decir, al utilizar valores discretos de tiempo con diferentes pasos se obtiene diferente resultado o precisión, si el paso es pequeño la simulación es más precisa, sin embargo entre más pequeño se requiere realizar un mayor número de cálculos para simular el mismo periodo de tiempo.

El proceso se ejecuta utilizando un ciclo de simulación

- Establecer la nueva posición de los objetos
- Detectar las colisiones
- Responder a las colisiones.

Mundo (World)

El mundo contiene todos los objetos que se crearon en la simulación ODE, los objetos se mueven dentro del mundo que se utiliza como el entorno físico. Este mundo utiliza varios parámetros aplicables a todos los objetos como la gravedad o la penetración de los objetos. Un ejemplo de la inicialización del mundo es el siguiente.

```
/*Se declara la variable */
dWorldID world;
// Se crea el mundo e inicializa la gravedad
world = dWorldCreate();
dWorldSetGravity(world, 0.0,0.0, -9.8);
```

Cuerpos y Geometría

Una definición de cuerpo en ODE es un contenedor que no ocupa un espacio, al que se le puede agregar masa y definir velocidad, forma y posición. En cambio la geometría es una estructura rígida que cubre al cuerpo, se define por separado del cuerpo del objeto pero ocupa el mismo espacio ya que sirve de base al sistema de detección de colisiones. Esta

estructura se utiliza para identificar los puntos de contacto durante una colisión y decidir las condiciones posteriores al contacto, además de servir para rotar o mover los objetos cuando su definición así lo permite. La estructura es un objeto independiente del cuerpo al que no se le puede asignar masa, color o textura.

Espacio

Un espacio es parecido a un mundo (wold) pero solo puede contener la geometría de los objetos y otros espacios. Esto permite que se pueda crear una jerarquía de espacios donde cada espacio calcula las colisiones internamente, lo que incrementa la eficiencia en los cálculos. Al declarar un espacio independiente para cada sistema al que se desea detectar las colisiones por separado se evita que se verifique si existe una colisión entre cada una de las geometrías de un espacio contra todas las geometrías del otro espacio.

En nuestro trabajo el uso de estos espacios se aplica en la definición del robot, se declaran todas las geometrías que forman al robot (Cajas y Cilindros), dentro de un mismo espacio que evita se calculen las colisiones entre éstas, y el cálculo se realiza sobre el espacio en conjunto con las otras geometrías del espacio global.

5.1.2. Motor de Física

El motor de física se divide en cuatro submódulos

- Control de tiempo.- Es la unidad que coordina el tiempo y sincroniza los pasos de cada unidad del sistema.
- Detector de límites.- Realiza los cálculos del sistema sobre la posibilidad de colisiones y empalmes entre los objetos. Toma los puntos de impacto generados

por el detector de colisiones y llama a los módulos de colisiones y movimiento para determinar la nueva trayectoria del objeto.

- Módulo de movimiento.- Realiza el cálculo del movimiento de cada objeto del sistema
- Módulo de colisiones.- Realiza el cálculo del nuevo impulso de cada objeto que ha colisionado en el sistema.

5.1.3. Objetos

Para la construcción de nuestro modelo se utilizaron dos tipos de objetos, los cuerpos rígidos y las uniones. Los cuerpos rígidos se utilizan para crear cajas, esferas y cilindros que dan forma al robot móvil, de los que se necesita controlar y calcular varias propiedades en cada paso de la simulación.

Propiedades que se modifican con el tiempo:

- Vector de posición (x, y, z), el punto de referencia del cuerpo, corresponde comúnmente al centro de gravedad.
- Velocidad lineal del punto de referencia, un vector (vx, vy, vz).
- Orientación del cuerpo representado por un cuaternion (qs, qx, qy, qz) o una matriz de rotación 3X3.
- Velocidad angular (wx, wy, wz) que indica cómo cambia de orientación a través del tiempo.

Propiedades que permanecen constantes:

- Masa del cuerpo.

- La posición del centro de gravedad con respecto al punto de referencia.
- La inercia, es una matriz 3X3 que indica como la masa se distribuye a través del centro de gravedad.

Una característica muy útil es la posibilidad de agrupar los cuerpos de manera que sean un solo objeto y que las características físicas se aplique al conjunto, lo que permite crear cuerpos de diversas formas, es similar al concepto de espacio.

El otro tipo de objeto fundamental en el modelo, son las uniones, figura 5.1, éstas unen el chasis del robot con las ruedas, en este objeto se utiliza como parámetro el torque y la fricción para calcular el movimiento que realiza el robot y en su definición se determina el tipo de movimiento que puede realizar. Existen tres tipos básicos de uniones, de articulación de rótulo, de bisagra y de pistón.

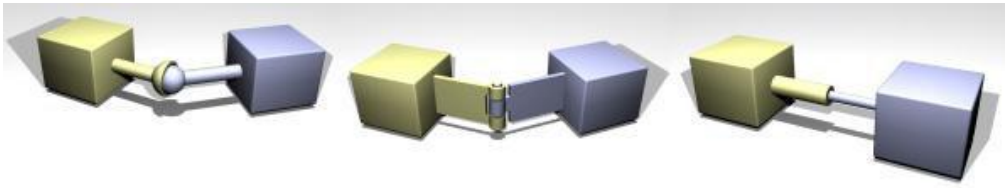


Figura 5.1 Tipos de uniones en ODE, de izq. a der., Articulación, Bisagra, Pistón

5.1.4. Colisiones

La detección de colisiones es una de las funciones más costosas del ciclo de simulación ya que es necesario verificar las posiciones de cada uno de los objetos en la nueva posición y determinar si se cruzan entre ellos. La función realiza los siguientes pasos:

1. Antes de cada nuevo paso en la simulación, el usuario llama a la función para detectar las colisiones y determinar que objetos hacen contacto con otros. La función regresa una lista de todos los puntos de contacto. Cada punto especifica una posición en el espacio, un vector normal a la superficie y la profundidad de la penetración.
2. Cada punto de contacto crea un registro con información extra del punto de contacto como la fricción presente en el punto de contacto, la dureza, etc.
3. Los puntos de contacto son puestos en un grupo el cual puede agregar o eliminar puntos de contacto fácilmente. La velocidad de la simulación baja si el número de contactos se eleva, es por esto que el programador puede establecer el límite de puntos de contacto a generarse.
4. Entonces la simulación da el siguiente paso
5. Todos los puntos de contacto son eliminados del sistema.

Antes de dar el siguiente paso es el momento en que el programador puede realizar algún cambio resultado de un choque, de no ser así las librerías establecerán la nueva posición a partir de las condiciones del choque.

El modelo puede determinar un gran número de puntos a partir de los cuales existe una colisión, por defecto se limita a 10, utilizar más puntos nos da una mayor precisión en los puntos de contacto y más carga de procesamiento, sin embargo un número muy pequeño puede representar que no se detecte la colisión y el objeto sea traspasado.

Un modelo típico de simulación sigue los siguientes pasos:

1. Crea la simulación del mundo o “world” sobre el que van a interactuar nuestros objetos.
2. Crea los cuerpos dentro del mundo.
3. Establece el “estado inicial” de todos los cuerpos, esto es; posición, velocidad, rotación, etc.
4. Crear las “uniones” o bisagras que se utilizarán en el mundo.
5. Colocar las uniones sobre los cuerpos.
6. Establecer los parámetros de todas las uniones (grados de rotación, friccion, etc.).
7. Crear una colisión con el mundo y geometría de colisión en los objetos que sea necesario.
8. Crear los grupos de uniones para mantener los puntos de contacto.
9. Se crea el ciclo de toda la simulación.
    a. Aplicar las fuerzas a los cuerpos que lo requieran
    b. Ajustas los parámetros de las uniones que lo requieran
    c. Llamar al detector de colisiones.
    d. Crear un punto de contacto para cada colisión y ponerlo en el grupo de puntos de contacto.
    e. Realizar el siguiente paso de la simulación.
    f. Remover todos los puntos del grupo de contactos
10. Destruir la simulación del mundo.

5.2. Arquitectura del Modelo

5.2.1 Modelo físico

Para comprobar la hipótesis se especifica un objetivo particular donde la solución permita determinar a partir del análisis de las variables, si el comportamiento de algunas de las estructuras de redes neuronales artificiales presenta una ventaja o un mejor comportamiento que la otra. La tarea a realizar por el agente autónomo es encontrar un objeto, tomarlo y depositarlo en un punto determinado del área.

La estructura del agente autónomo debe permitir cierto nivel de movilidad por lo que su arquitectura es parecida a un vehículo de motor. Se forma por tres bloques (ortoedros) de diferentes dimensiones suspendidos en cuatro ruedas de forma cilíndrica.

La consistencia de estos objetos es de “cuerpos rígidos”, por su definición, no sufren deformaciones por el movimiento o el contacto con otros cuerpos. Esta característica permite al simulador realizar mucho más rápido las operaciones necesarias para determinar las colisiones y los desplazamientos de los objetos al evitar los cálculos para la presentación gráfica de los objetos deformados.

ODE permite definir los tres bloques que definen el “chasis” del vehículo como un solo cuerpo lo que permite un desplazamiento en conjunto y también elimina los cálculos por colisiones entre ellos. Se toma uno de los bloques como base para posicionar en él las llantas, el bloque sobre éste permite darle estabilidad por la variación en peso que significa la velocidad del vehículo y el tercer bloque al frente determina la posición de los sensores frontales. Figura 5.2

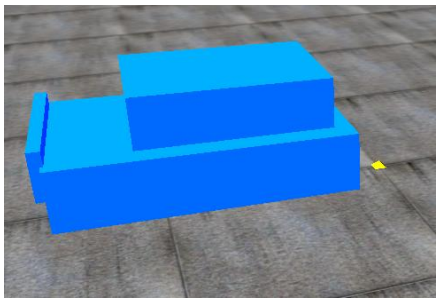


Figura 5.2 Chasis del Modelo

Las llantas se acoplan al “chasis” a través de uniones, en ODE existen tres tipos de uniones, de articulación de rótulo, de bisagra y de pistón. La unión de articulación permite al objeto moverse en cualquier sentido (omnidireccional), si no se le aplica fricción alguna, en cambio al aplicarse fricción puede limitarse algún grado de libertad en el movimiento.

Con este mecanismo se unen las llantas al “chasis” del vehículo, figura 5.3, a las ruedas delanteras no se les define fricción alguna por lo que se mueven libremente mientras que a las ruedas traseras se les restringe el movimiento lateral, es decir, permanecen siempre paralelos al chasis. Esto es, el vehículo utiliza una propulsión diferencial, las ruedas traseras sobre el mismo eje son las encargadas de proporcionar la tracción del vehículo y la dirección a partir de la diferencia de velocidades entre las ruedas.

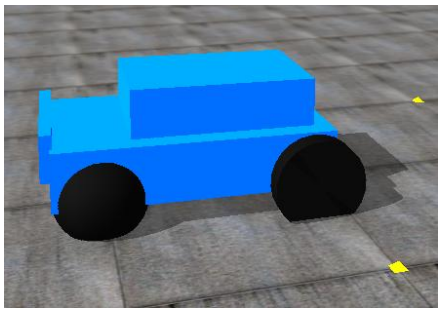


Figura 5.3 Posición de las ruedas en el chasis

Esto se consigue al establecer en la definición de la unión una fricción muy alta que impide el movimiento lateral mientras que la velocidad y dirección del vehículo son consecuencia de la salida de las redes neuronales, la magnitud de cada salida se aplica al torque de cada rueda.

```
dJointSetHinge2Param (joint[2],dParamVel2,-velocidad);    //giro vertical

                    // Parámetro para mantener las ruedas siempre alineadas
dJointSetHinge2Param (joint[i],dParamSuspensionERP,0.4);
dJointSetHinge2Param (joint[i],dParamSuspensionCFM,0.8);
dJointSetHinge2Param (joint[i],dParamLoStop,0);
dJointSetHinge2Param (joint[i],dParamHiStop,0);
```

Finalmente se define un cuerpo rígido más. El sensor, este se define como un bloque rectangular que no posee cuerpo en el espacio de simulación, lo que significa que no tiene masa y por tanto tampoco peso adicional para el cuerpo del vehículo, figura 5.4. A pesar de esto, si cuenta con una geometría que permite detectar las colisiones con los objetos y establecer la distancia entre el punto de contacto y el chasis, este punto de posible colisión es detectado por el simulador pero al no tener un cuerpo la consecuencia del choque no se genera.

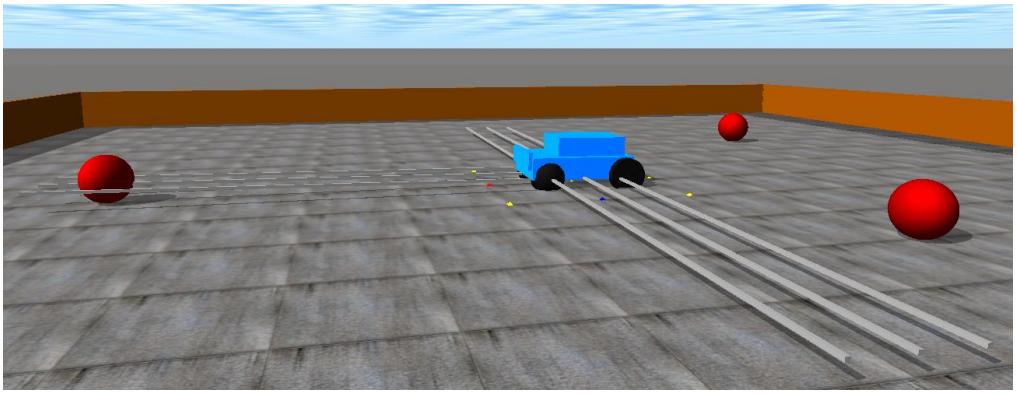


Figura 5.4 Posición de los sensores del vehículo

La superficie sobre la que va a moverse el agente es cuadrada, rodeada por paredes que impiden que el vehículo pase del límite y una altura superior a la del vehículo para que puedan ser detectadas por los sensores. El contenedor o depósito donde el agente móvil coloca el objeto se ubica en una esquina de la superficie, está limitado por dos paredes que son las que detectan los sensores.

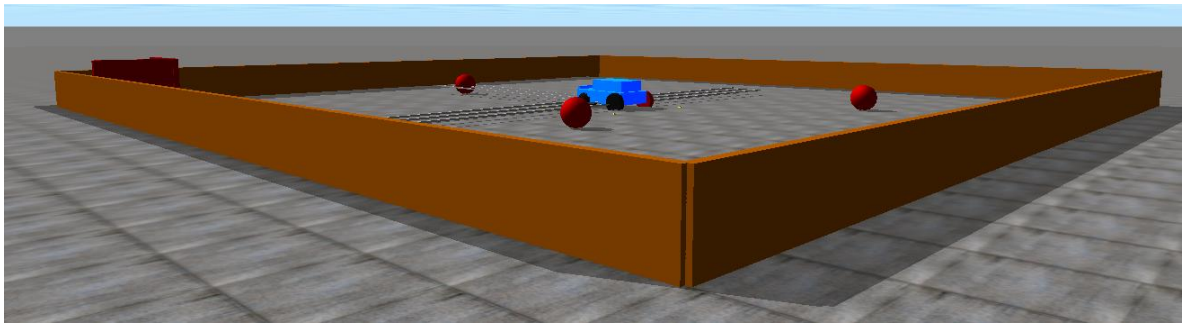


Figura 5.5 Área delimitada y contenedor

El último componente que se define es la pelota, figura 5.5, una esfera rígida de la misma altura que el vehículo para que éste pueda detectarlo a través de los sensores, se ubica en diferentes posiciones durante el entrenamiento y en posiciones aleatorias durante las pruebas finales.

5.2.2 Modelo Neuronal

En cuanto al modelo neuronal se utilizan dos estructuras diferentes, la estructura de red neuronal monolítica, figura 5.6 y la estructura de redes cooperativas conformada por tres redes con la misma estructura, figura 5.7

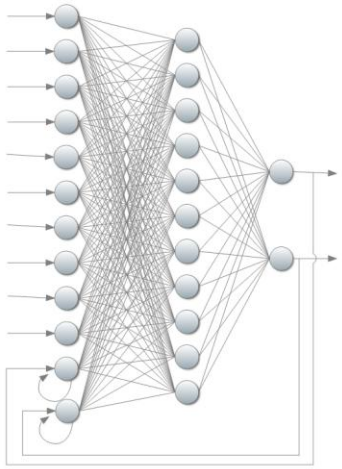


Figura 5.6 Estructura de Red Neuronal Monolítica

En la red neuronal de estructura monolítica se definen tres capas de neuronas una de ellas oculta, la capa de entrada cuenta con doce neuronas, nueve reciben información de los sensores de distancia, una recibe cero o uno al haber capturado o no la pelota, y las dos siguientes reciben retroalimentación de la capa de salida. La capa oculta cuenta con 11 neuronas sin retroalimentación de la capa de salida o de sí misma. Finalmente se tienen dos neuronas de salida que definen el movimiento de cada rueda del vehículo y cada neurona retroalimenta a una neurona de entrada diferente.

El modelo de redes cooperativas utiliza tres redes neuronales con la misma estructura, figura 5.7, la primera se configura para evitar chocar, la segunda para detectar de manera lateral los objetos, y la tercera para detectar de manera frontal los objetos.

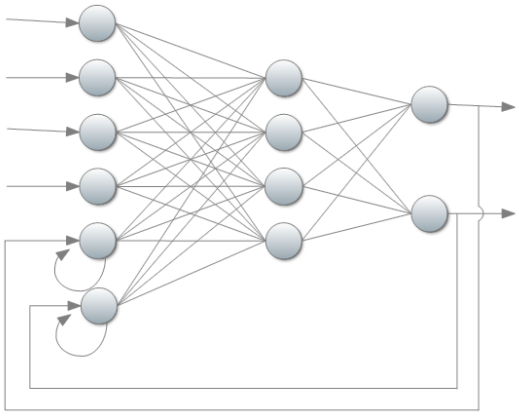


Figura 5.7 Modelo individual de Red Neuronal Cooperativa

En cada red cooperativa se definen tres capas de neuronas una de ellas oculta, la capa de entrada cuenta con seis neuronas, cada red recibe diferente información de entrada. La capa oculta cuenta con cuatro neuronas sin retroalimentación de la capa de salida o de sí misma. Y finalmente se tienen dos neuronas de salida que definen el movimiento de cada rueda del vehículo y cada neurona retroalimenta a una neurona de entrada diferente.

El valor final de salida que define la velocidad y la dirección del vehículo es la sumatoria de las salidas de las tres redes neuronales, esto es, la neurona de salida 1 de cada red neuronal se suma para alimentar a la rueda izquierda, la neurona de salida 2 de cada red neuronal se suma para alimentar a la rueda derecha.

La diferencia en las entradas de cada red de acuerdo a la función que se define para ella es la siguiente:

Primera Red (Para evitar chocar)

- Las tres primeras neuronas de entrada reciben información de los sensores del vehículo, uno frontal y dos laterales.
- La cuarta y quinta neuronas reciben retroalimentación de cada una de las llantas.

- La sexta neurona recibe uno al haber capturado la pelota o cero cuando no.

Segunda Red (Detectar de forma lateral)

- Las cuatro primeras neuronas de entrada reciben información de los sensores laterales del vehículo, dos de cada lado.
- La quinta y sexta neuronas reciben retroalimentación de cada una de las llantas.

Tercera Red (Detectar de manera frontal)

- Las tres primeras neuronas de entrada reciben información de los sensores frontales del vehículo.
- La cuarta y quinta neuronas reciben retroalimentación de cada una de las llantas.
- La sexta neurona recibe uno al haber capturado la pelota y cero cuando no.

Nuestro modelo utiliza una sola capa oculta ya que muchos autores como Turhan, mencionan: *"La regla de oro es: '**más capas ocultas pueden manejar casos más complicados'**, sin embargo, las últimas pruebas indican que una capa oculta generalmente es suficiente para muchos de los problemas que enfrentamos hoy",* [Turhan_1995]. Y las pruebas realizadas en el caso de análisis con esta estructura, dan resultados satisfactorios.

El número de neuronas en la capa oculta al igual que el número de capas es una definición importante en la estructura general ya que cantidades excesivas o escasas producen efectos llamados Underfitting y Overfitting.

"Underfitting se produce cuando hay muy pocas neuronas en las capas ocultas para detectar adecuadamente las señales en un complicado conjunto de datos". [Heaton 2008]

"Overfitting se produce cuando la red neuronal tiene una gran capacidad de procesamiento de la información tal que una limitada cantidad de información contenida en los entrenamientos no es suficiente para capacitar a todas las neuronas en las capas ocultas". [Heaton 2008]

Desafortunadamente no existe una fórmula para decidir el número de neuronas en esta capa, sin embargo, algunos autores mencionan "reglas de oro" a considerar como guía. A pesar de que unas reglas se contraponen a otras.

Heaton mencionan como "reglas de oro" para el número de neuronas ocultas las siguientes, [Heaton 2008].

- Debe estar entre el tamaño de la capa de entrada y el tamaño de la capa de salida.
- Debe ser 2/3 del tamaño de la suma de las capas de entrada y salida.
- Debe ser inferior a dos veces el tamaño de la capa de entrada.

Amalendu sugiere otra "regla de oro" que dice, "*el mínimo de neuronas requeridas en la capa oculta para evitar el sobre-entrenamiento y para obtener una buena exactitud en los datos de prueba es* $\sqrt{N_i \times N_0}$ *donde* N_i *es el número de neuronas de la capa de entrada y* N_0 *es el número de neuronas de la capa de salida",* [Amalendu y otros 2004].

Estas reglas son solo "propuestas" de cada autor, ya que finalmente será necesario realizar algunas pruebas para determinar cuál es la estructura más adecuada al problema. En nuestro caso se decide el número de neuronas siguiendo la primera "regla" de Heaton "Debe estar entre el tamaño de la capa de entrada y el tamaño de la capa de salida"; y a

partir de este número se realizan los ajustes de acuerdo a los resultados que se fueron obteniendo.

Se utiliza el concepto de retroalimentación de la red neuronal para que nuestro agente autónomo sea capaz de “recordar” el estado anterior y sea éste un parámetro adicional en el comportamiento del vehículo.

Una explicación general de este concepto es: cuando la salida de una neurona realimenta a una neurona de una capa anterior, se dice que está en función de las entradas de la capa anterior en un tiempo t, y su propia salida que existía en un momento anterior, es decir, en un tiempo $(t - \Delta t)$, donde (Δt) es el tiempo para un ciclo de cálculo. Por lo tanto, estas redes presentan características similares a una memoria a corto plazo, debido a que la salida de la red depende de las aportaciones actuales y anteriores. [Tsoukalas y otros 1997].

Se utiliza retroalimentación a sí misma para que cada neurona establezca un nivel de “estados”, donde una parte de la entrada sea un historial de los comportamientos anteriores. Con la finalidad de incrementar la influencia de los comportamientos anteriores, la entrada de estas neuronas se compone de:

$$Entrada = (Neurona\ salida\ * \ 0.2) + \ (Neurona\ Entrada * \ 0.8)$$

El resto de las neuronas se definen con la sumatoria de todas las entradas y en cada neurona se utiliza como función de activación una función sigmoidea, la cual nos permite utilizar entradas continuas y discretas con un rango de salida de [0, 1]. El parámetro de umbral no se utiliza.

$$y = \frac{1}{1 + e^{-x}}$$

5.2.3 Algoritmo Genético

Los algoritmos genéticos se utilizaron principalmente porque: trabajan sobre varias soluciones de forma simultánea, lo cual facilita su implementación en paralelo, en lugar de hacerlo de manera secuencial como las técnicas tradicionales. Al utilizarse en problemas de optimización son menos susceptibles de quedar atrapados en óptimos locales, comparados con los métodos de búsqueda local. Además el método de retropropagación que es el más utilizado para la fase de entrenamiento no se aplica a modelos recurrentes.

Como ya se mencionó la simulación se realiza con la librería ODE, por lo que todo el prototipo se implementa en C++ con Visual Estudio 2010, bajo la plataforma de Windows.

Los parámetros que se utilizaron

- Una población de 100 individuos
- Un total de 50 generaciones por entrenamiento
- Se utiliza el torneo de tres individuos como método de selección
- Probabilidad de cruce de un 70%
- Durante el cruce se hace una selección aleatoria del número de cromosomas a intercambiar.
- Probabilidad de mutación 10% que disminuye con cada generación.
- Se utiliza el elitismo al conservar el mejor de cada generación.

El uso de 100 individuos se debe a que presenta mejores resultados por la diversidad de los valores iniciales que una población menor, al inicio de la prueba era de 50 individuos. Lo mismo sucede al determinar 50 generaciones ya que de la generación 40 en adelante el avance de la eficiencia se incrementa muy poco.

El operador de selección por torneo presenta la característica de preservar sólo a los mejores individuos a diferencia del método de la ruleta donde existe la posibilidad de que el peor calificado de los individuos pase a la siguiente generación. El número de individuos es de tres ya que da mayor seguridad sobre la supervivencia de los mejor calificados. Esta decisión va acorde al uso del "Elitismo" que permite que el mejor de los individuos se conserve tal cual para la siguiente generación.

El operador de cruce tiene una probabilidad de 70% para que se dé una mayor diversidad en la siguiente generación conservando un 30% de los individuos que hasta el momento "han logrado sobrevivir" para conservar la dirección que sigue el proceso de optimización. En este operador se hace el ajuste para que sólo un número aleatorio de cromosomas se intercambien ya que el resultado de la cruza total genera nuevos individuos con resultados más pobres que los primeros.

Un 10% de mutación permite explorar mayores espacios de solución fuera de la actual "ruta" que sigue la generación de individuos, ya que muchas de las pruebas quedan estancadas en óptimos locales. Este porcentaje disminuye para que cada vez sean más los individuos sin cambios que siguen la "ruta" de la generación.

En el caso particular de las redes independientes se toman como un sólo individuo por lo que evolucionan en su conjunto, esto es, se suma el valor de eficiencia de las tres redes para estimar su valor al momento del torneo y se seleccionan las tres para conformar el siguiente individuo. Al momento del cruce se intercambian cromosomas entre las redes del mismo tipo.

La función de aptitud le da puntos por diferentes conceptos

1. El máximo valor se obtiene al depositar la pelota en el contenedor (10,000).
2. Al encontrar la pelota (1,000).
3. Al detectar la pelota o el contenedor por los sensores frontales se agrega 0.1 si la diferencia de velocidad en ambas ruedas es menor a 0.05 (en un rango de -3 a 3), en caso contrario se resta 0.1.
4. Al detectar la pelota o el contenedor por los sensores laterales se agrega 0.1 si la diferencia de velocidades de las ruedas es menor a 0.1 a favor del sentido donde fue detectado el objeto, en caso contrario se resta 0.1.
5. Cuando hay un choque con la pared se le resta 0.1. Cuando ocurre un choque de manera automática el vehículo retrocede durante 10 tiempos.

Se obtienen valores muy grandes al encontrar la pelota y posteriormente el contenedor ya que la meta es encontrar primero la pelota y luego el contenedor. El objetivo es que las recompensas por detectar los objetos sean menos significativas que el hecho de encontrarlos. El valor asignado al contenedor no se da si no se ha encontrado la pelota, esto es, si la pelota no se ha encontrado y se encuentra primero el contenedor no genera

valor y la búsqueda continúa. Sólo cuando se encuentra la pelota y luego el contenedor se detiene la prueba y se suma la recompensa por el éxito.

Al detectar la pelota de frente se toma en cuenta la dirección del vehículo, que está determinada por la velocidad de cada rueda. Dado que es muy poca la probabilidad de que la diferencia de velocidades sea cero, se toma un margen de 0.05 para considerar que el vehículo va de frente y asignar la recompensa.

Dicho margen también se aplica al detectar la pelota por un costado y asignar la recompensa, sólo si el vehículo da vuelta en el sentido que se detecta el objeto.

Capítulo 6. Progreso del modelo y evolución

El proceso para la definición del modelo final se desarrolla en seis etapas:

1) Elaborar un modelo preliminar de red única para la búsqueda de la pelota en una posición fija

El desarrollo del modelo consiste en probar una arquitectura simple de cada uno de los módulos a utilizar: el modelo simulado, la red neuronal artificial única, y el algoritmo genético. Se elabora un modelo con 5 sensores de distancia que tiene como objetivo encontrar únicamente la pelota en un punto fijo del área de búsqueda, se utiliza una población de 50 individuos y 30 generaciones. El objetivo es confirmar que nuestro algoritmo de entrenamiento permite una evolución que encuentre el camino más corto al objetivo. Este modelo converge con cierta facilidad en 10 generaciones.

2) Ajuste del modelo para ubicar la pelota en posiciones aleatorias

Se desarrolla un comportamiento para encontrar la pelota en una posición aleatoria, se prueban tres modelos:

- Poner cinco pelotas al mismo tiempo en posiciones aleatorias.
- Probar cinco veces a cada individuo con una sola pelota en una posición aleatoria diferente cada vez.
- Probar cinco veces a cada individuo con una sola pelota en posiciones fijas distribuidas de manera uniforme por la superficie.

El último modelo es el que da mejores resultados ya que de forma aleatoria la eficiencia es relativa a la posición de la pelota, si ésta se encuentra cerca del vehículo o de las paredes.

3) Ajuste del modelo para encontrar el contenedor una vez que se encontró la pelota

En esta etapa se realizan ajustes al modelo:

- Se pone un mecanismo automático para simular que el vehículo recoge la pelota.
- Se define un mecanismo para que el vehículo retroceda 10 tiempos al chocar con la pared. Esto evita que el vehículo se quede atorado en el punto de choque.
- Se establece un porcentaje de 10 % en la mutación ya que muchos de los entrenamientos caen en óptimos locales al encontrar la pelota de manera eficiente y no al contenedor.

4) Generar y probar el modelo con la estructura cooperativa

Se genera la estructura de red cooperativa y se ajustan los parámetros necesarios para que encuentre el contenedor, como son:

- Se aplican los operadores genéticos para que evolucione cada red por separado.
- Que la evolución sea de las redes en conjunto, que los individuos que cooperan para una solución se tomen como uno solo al momento de aplicar cada operador genético. Esta opción genera mejores resultados.

5) Determinar los parámetros del entrenamiento definitivo

Se define el entrenamiento para cada individuo, son 5 pruebas de 4,000 pasos, cada prueba con una posición diferente de la pelota, siendo las mismas 5 posiciones para todos los individuos, figura 6.1, el contenedor siempre en la misma posición. En cada paso se calcula la velocidad a partir de la salida de la red neuronal y se suman los puntos por eficiencia, utilizando la función de eficiencia que se mencionó con anterioridad.

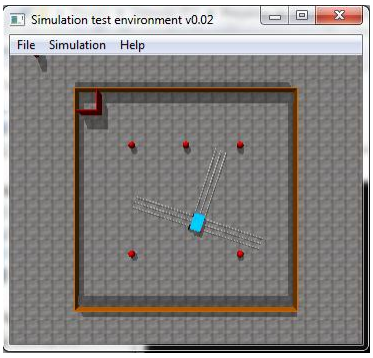


Figura 6.1 Posiciones de las pelotas en el entrenamiento

6) La prueba del controlador utilizando los mejores individuos de cada entrenamiento.

Se evalúa al mejor individuo de la generación 50 de cada uno de los entrenamientos realizados. Se definen dos pruebas finales. En la primera cada individuo realiza 21 pruebas con la pelota en posiciones aleatorias. En la segunda cada individuo realiza 25 pruebas en las mismas 25 posiciones predeterminadas.

El tiempo promedio aproximado de procesamiento de la red neuronal para cada individuo es de 1462 micro segundos para la red única y de 1440 micro segundos para la red cooperativa, lo que representa aproximadamente el 0.39% y 0.38% respectivamente del procesamiento total de la simulación, así que el utilizar una u otra estructura de red no es un factor en el tiempo de entrenamiento y tampoco para el tiempo de prueba.

El tiempo aproximado del entrenamiento es de 2 segundos para cada sesión de un individuo, una vez que se suprimen la textura, las sombras y la presentación gráfica en la simulación lo que redujo en un 80% el tiempo necesario.

Como ya se mencionó la parte más costosa de la simulación son los cálculos para generar las siguientes posiciones, en este caso representa un poco más del 58% del tiempo, el otro factor que consume tiempo del procesador es la detección de colisiones, un poco más del 20% del tiempo representa determinar las colisiones entre todos los objetos y otro 20% el manejo de colisiones para simular los sensores de distancia.

El equipo utilizado para el cálculo de los tiempos es una Laptop Gateway con procesador Intel Core I3-370M, utilizando como plataforma Windows 7. Los tiempos son aproximados ya que existen muchos procesos que corren al mismo tiempo que las rutinas de entrenamiento como son el antivirus y procesos del mismo sistema operativo, así que se toma la media aritmética de 30 mediciones de tiempos.

6.1. Análisis de resultados

Para el análisis de resultados se realizan 96 entrenamientos, 48 para cada estructura, cada entrenamiento consta de una 100 individuos generados aleatoriamente que evolucionan durante 50 generaciones. De cada entrenamiento se toma al mejor de los individuos y se le hace una evaluación final de eficiencia que consta de 40,000 pasos y donde sólo se recompensa al encontrar la pelota y posteriormente el contenedor.

Esta evaluación final para ambas estructuras consta de dos módulos de prueba de eficiencia, uno utilizando posiciones aleatorias y otro utilizando un conjunto de posiciones predeterminadas. En el primer módulo de prueba cada individuo realiza 21 pruebas con una pelota en posición aleatoria. En el segundo módulo de prueba cada individuo realiza 25 pruebas con una pelota en posición predeterminada, la Figura 6.2 muestra las 25 posiciones de la pelota para la evaluación.

Figura 6.2 Las 25 posiciones fijas de la pelota.

A partir de que los entrenamientos del módulo de pruebas en posiciones aleatorias dan resultados satisfactorios, se configura el módulo de pruebas en posiciones fijas con el objetivo de comprobar los resultados y obtener información adicional sobre las áreas

donde el vehículo tiene dificultades para cumplir su misión. Estas posiciones se fijan de manera homogénea por toda la superficie para que cada individuo evaluado tenga las mismas condiciones en cada prueba.

La información en ambos módulos muestra una tendencia que favorece a las redes cooperativas. Si bien los resultados presentan algunas diferencias éstas se deben a la naturaleza de aleatoriedad del primer módulo de posiciones aleatorias.

Módulo en posiciones Aleatorias.

El concepto más importante en los resultados, es el porcentaje máximo de eficiencia obtenido por cada estructura de red. El porcentaje máximo en el total de las pruebas se obtiene en las redes cooperativas con un 85.7% sobre el 81.0% de la red única, tabla 6.1.

Estructura de Red	% Máximo Alcanzado
Única	81.0
Cooperativas	85.7

Tabla 6.1 Mayor grado de eficiencia en posiciones aleatorias

Además se organiza la información por el porcentaje de pruebas de cada estructura de red, que lograron una eficiencia mayor a una serie de rangos. Tabla 6.2

	Porcentaje de Efectividad de las Pruebas (%)				
Estructura de Red	Mas del 25%	Mas del 50%	Mas del 60%	Mas del 70%	Mas del 80%
Única	77.1	37.5	35.4	27.1	06.3
Cooperativas	50.0	39.6	29.2	29.2	14.6

Tabla 6.2 Porcentaje de pruebas según porcentaje de efectividad posiciones aleatorias

De estos resultados se destaca lo siguiente:

- El porcentaje de las pruebas que superaron el 25% de efectividad en las redes cooperativas es de un 50% mientras que la red única es de casi un 80%.
- Al pasar del 25% al 50% la tendencia se reduce, un alto porcentaje de las pruebas de la red única se estanca.
- Del 50% al 70% el número de pruebas que no logran mejorar el resultado se reduce considerablemente.

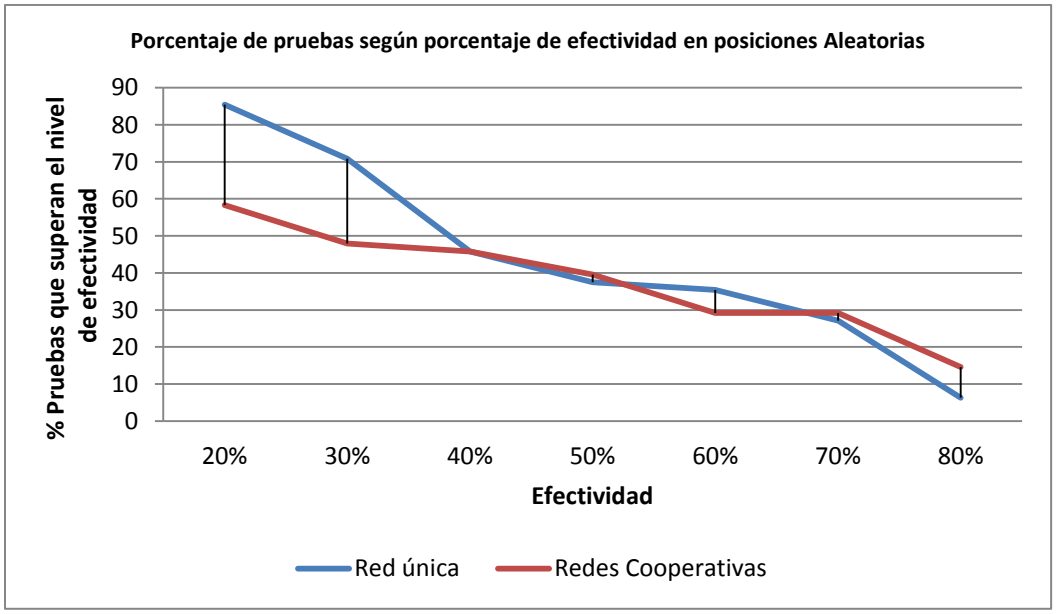


Figura 6.3 Porcentaje de pruebas según porcentaje de efectividad en posiciones aleatorias.

Durante esta prueba la eficiencia final la red cooperativa mantiene una tendencia semi-uniforme, mientras que la estructura única presenta una pendiente más pronunciada del 20% al 40% de eficiencia, figura 6.3.

Módulo en posiciones Fijas.

El porcentaje máximo en el total de las pruebas se obtiene en las redes cooperativas con un 76.0% sobre el 68.0% de la red única.

Estructura de Red	% Máximo Alcanzado
Única	68.0
Cooperativas	76.0

Tabla 6.3 Mayor grado de eficiencia en posiciones fijas

Aquí también se organiza la información por el porcentaje de pruebas de cada estructura de red, que lograron una mayor eficiencia a una serie de rangos. Tabla 6.4.

	Porcentaje de Efectividad de las Pruebas (%)				
Estructura de Red	Mas del 25%	Mas del 50%	Mas del 60%	Mas del 70%	Mas del 80%
Única	37.5	14.6	10.4	0.0	0.0
Cooperativas	41.7	25.0	16.7	4.2	0.0

Tabla 6.4 Porcentaje de pruebas según porcentaje de efectividad Posiciones fijas

De estos resultados se destaca lo siguiente:

- En el módulo de pruebas en posiciones fijas, a diferencia de los resultados que presenta el módulo de pruebas en posiciones aleatorias, las redes cooperativas superan a la red única en todos los rangos de efectividad.
- En ambas estructuras de red menos del 50% de la pruebas lograron dar individuos con un nivel de efectividad mayor al 25%, lo que indica que más del 50% se estanca en las primeras generaciones.

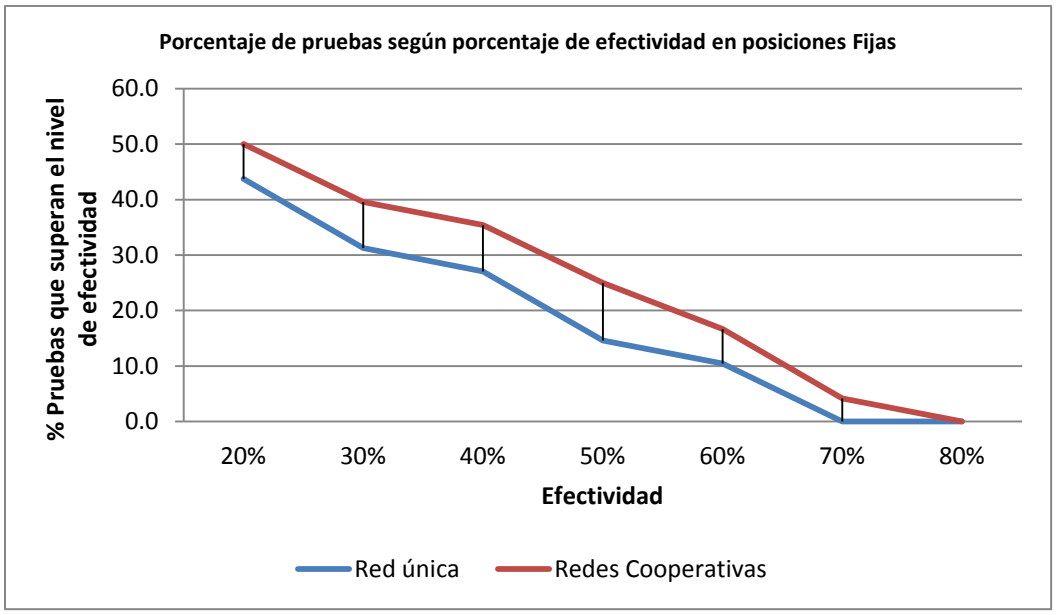


Figura 6.4 Porcentaje de pruebas según porcentaje de efectividad en posiciones fijas

En la gráfica de la figura 6.4 se puede ver como ambas estructuras siguen la misma tendencia con una diferencia en el porcentaje de pruebas de aproximadamente el 6% a favor de la red cooperativa.

Se logra que algunos de los controladores encuentren el objetivo en un porcentaje alto de casos, sin embargo la efectividad del 100% no se da debido a que la pelota se ubica en lugares que no fueron probados durante el entrenamiento, figuras 6.5 y 6.6, o cuando son posiciones aleatorias, lugares de poco acceso al vehículo como son las esquinas o pegados a la pared.

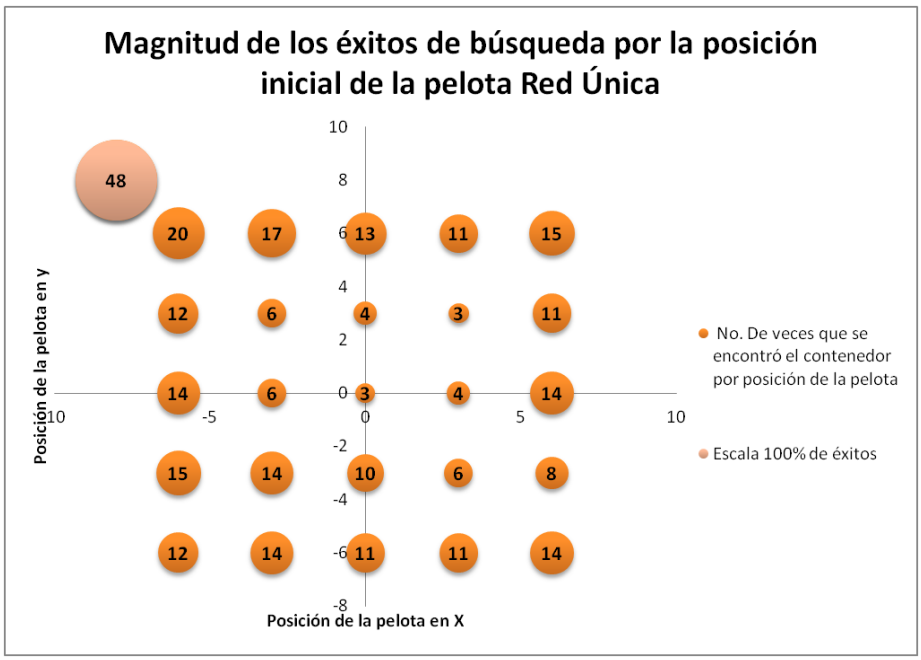


Figura 6.5 Magnitud de los éxitos por posición de la pelota, Red Única

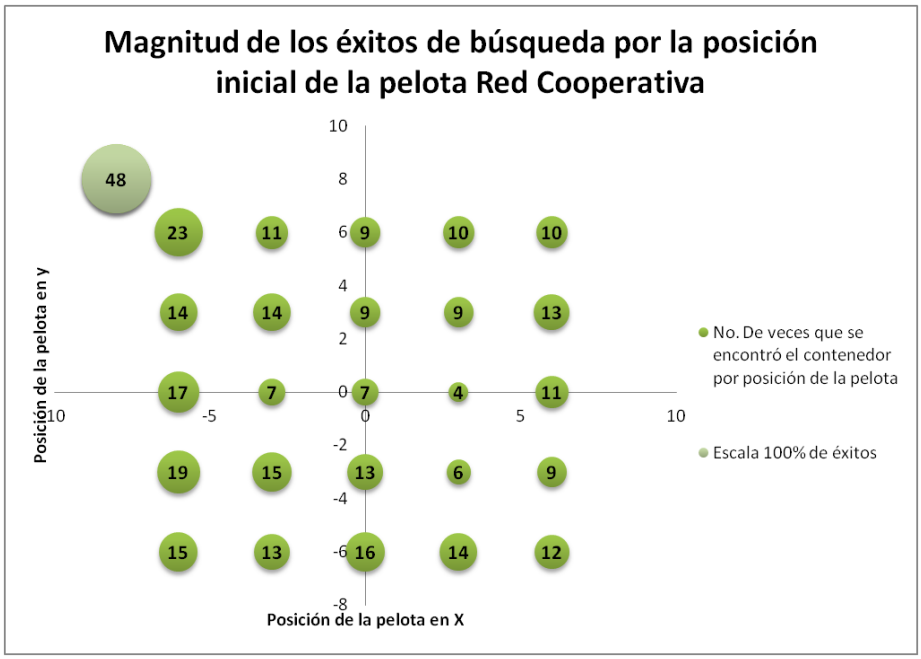


Figura 6.6 Magnitud de los éxitos por posición de la pelota, Red Cooperativa.

En las figuras 6.5 y 6.6 se ven las 25 posiciones iniciales de la pelota y se indica en cada posición el número de veces que se logró el objetivo de depositar la pelota en el contenedor. La posición que logra un mayor número de éxitos es la que se encuentra más

cercana al contenedor, de la misma forma el lado izquierdo obtiene el mayor número de éxitos en ambas estructuras, sin embargo en la red única el siguiente lado que obtiene mejores resultados en la parte superior mientras que la red cooperativa es la parte inferior. Como era de esperarse las posiciones que obtuvieron un menor número de éxitos son las posiciones centrales, esto debido a que el entrenamiento no considera una pelota en esta posición.

De manera general se generan dos comportamientos, en el primero el robot realiza círculos internos recorriendo la posición de las pelotas que se ubicaron en el entrenamiento. El segundo sigue un movimiento de pequeñas espirales mientras realiza vueltas internas. Figura 6.7.

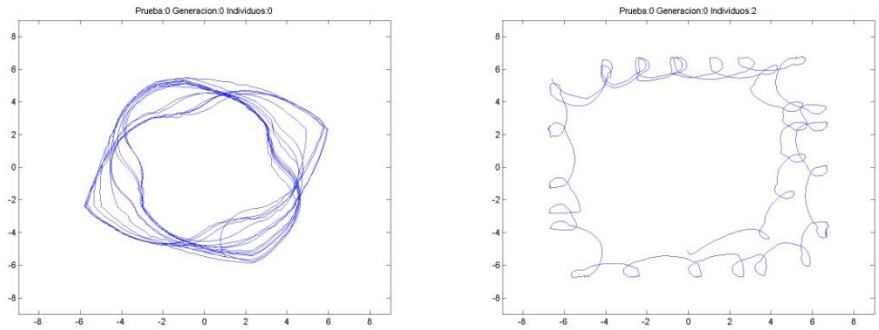


Figura 6.7 Rutas típicas del vehículo

El primer comportamiento es eficiente para encontrar las pelotas sin embargo no logra ubicar en ningún momento el contenedor, este es el punto donde se detiene el avance del algoritmo para encontrar mejores individuos.

El segundo comportamiento da los mejores resultados ya que hace un barrido general de la superficie cercana a las paredes, una de las desventajas que se muestra es que las

pelotas que se ubiquen muy cercanas al centro o en las esquinas, tienen pocas posibilidades de ser encontrados.

Las diferentes pruebas que generaron este segundo comportamiento realizan las vueltas concéntricas con diferentes radios, las más cerradas hacen un barrido más completo pero su recorrido es más lento. También difieren en su distancia a la pared, lo que mejora el desempeño si se tiene una distancia apropiada, sin embargo al estar muy cerca no alcanza a cubrir las pelotas que se ubican al centro de la superficie de búsqueda. En el anexo “B” se puede ver otro grupo de comportamientos.

6.2. Conclusiones

Los mejores individuos de la estructura de red cooperativa obtuvieron un porcentaje superior de eficiencia que los mejores individuos de la red única, se supera en 4.7% utilizando posiciones aleatorias y en 8% utilizando posiciones fijas. Un rasgo a resaltar de los resultados es que la red cooperativa se muestra más robusta que la red única al lograr mejores resultados en el espacio central del área de búsqueda, donde no se posiciona una pelota durante el entrenamiento, como se ve en las figuras 6.5 y 6.6.

Otra particularidad es que, si bien el tiempo de procesamiento que requiere la red cooperativa es mayor (en 22 microsegundos), en promedio se requiere de un número inferior de pruebas para lograr encontrar un individuo "aceptable", lo que reduce el costo final de entrenamiento en la formación del controlador, ya que el tiempo de procesamiento de la red neuronal representa menos del 0.5% del total de la simulación.

De acuerdo a estas características se puede determinar que bajo las condiciones que se establecen en este trabajo se corrobora la hipótesis: "*Un controlador para un robot autónomo utilizando redes neuronales artificiales, es más eficiente con una estructura de redes neuronales distribuidas o independientes, que utilizando una estructura de redes monolítica"*.

La función para generar un retroceso al momento del choque se implementa para poder lograr el objetivo, ésta ocasiona que el vehículo retroceda lo suficiente para que pueda girar, de lo contrario se atora en la posición del choque. Esta rutina toma la forma de un

reflejo que complementa lo que se va aprendiendo, reflejo al que se le asigna una mayor prioridad que al conocimiento que se va generando, como un instinto de conservación.

Un aspecto poco favorable pero importante, es que la red única en promedio requiere de menos tiempo para lograr el objetivo, así que dependerá finalmente de la aplicación a utilizar el ponderar la importancia entre el tiempo y la efectividad.

Un área de oportunidad inmediata utilizando este mismo modelo es modificar el entrenamiento para mejorar la eficiencia en los espacios que lograron pocos éxitos. El modificar las posiciones de las pelotas, incluir un número mayor de posiciones así como incrementar el tiempo de cada entrenamiento para que se acerque al tiempo de prueba son algunos de estos cambios.

Se utilizan redes neuronales artificiales normalmente para realizar clasificaciones o generalizaciones, esta característica les permite soportar pequeñas fallas dentro del sistema en general. Esto ofrece una nueva área de prueba para el modelo, donde uno o más de los controles (sensores o actuadores) presente fallas, ya sea de manera intermitente o permanente.

Finalmente es posible trasladar este concepto a otro modelo con un número mayor de controles bajo las mismas condiciones de un vehículo móvil, o utilizando nuevas condiciones como puede ser un brazo robótico.

Anexo A Análisis de complejidad

En el campo del análisis de complejidad se estudia a los problemas de acuerdo a la dificultad que se tiene para resolverlos. La primera división se realiza en función del tiempo en que pueden resolverse los problemas, si su solución puede darse en tiempo polinomial o no, independientemente del algoritmo que se utilice. [Russell y Norvig 2004]

Un tiempo polinomial está en función de las variable implicadas en el problema, lo cual quiere decir que el tiempo pueda calcularse utilizando una formula polinómica que utiliza únicamente operaciones aritméticas de suma, resta, multiplicación y exponentes enteros positivos; como son por ejemplo:

$$A = \pi r^2 \qquad a^2 + b^2 = c^2$$

La clase de los problemas polinomiales, los que pueden resolverse en tiempo $O(n^k)$, se llaman **P**. En ocasiones se les llama problemas "Fáciles" sin embargo contiene problemas con tiempos de ejecución de $O(n^{1000})$ de manera que "fácil" no es un término muy literal.

Otra clase de problemas es **NP**, la clase de problemas no deterministas. Esta clase de problemas se resuelven en tiempos del tipo $O(2^n)$ donde sí existe un algoritmo que pueda averiguar la solución y a continuación verificar en un tiempo polinómico si la respuesta es correcta, pero para valores muy altos de n el tiempo que se requiere para resolverlos es mayor a la capacidad actual de procesamiento.

Anexo B Comportamientos Típicos

Estos son algunos de los comportamientos típicos que genera la red única.

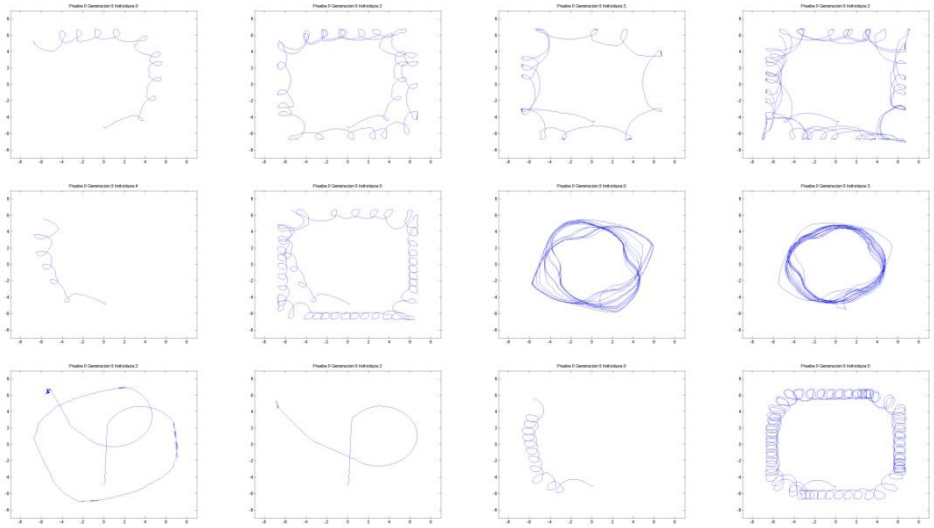


Figura B.1 Trayectorias típicas de la Red Única.

Estos son algunos de los comportamientos típicos que genera la red cooperativa.

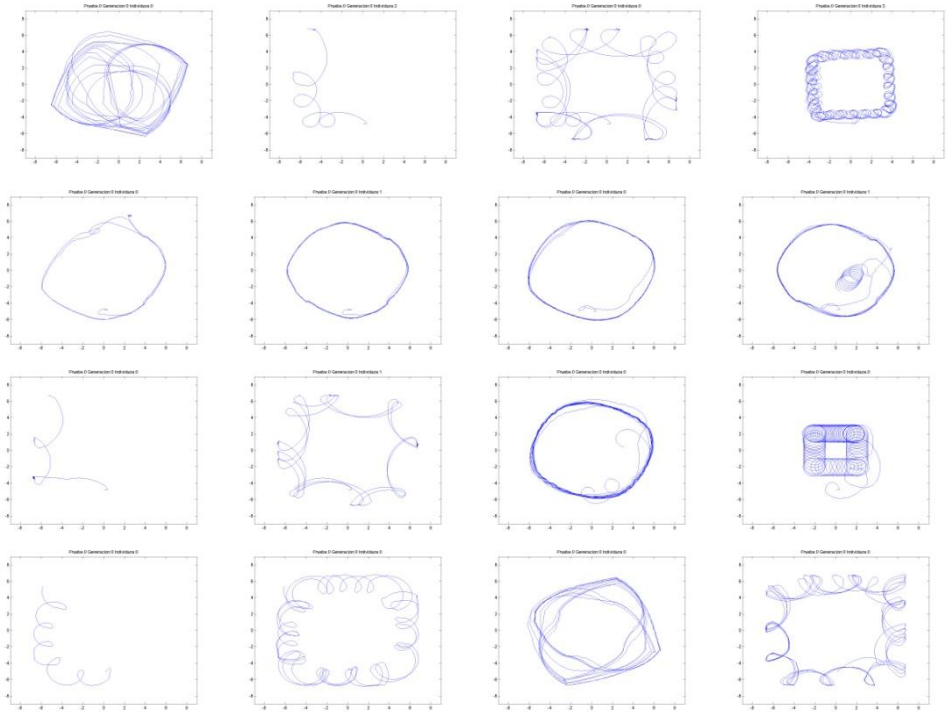


Figura B.2 Trayectorias típicas de la Red Cooperativa.

Anexo C Éxitos por posición inicial de la pelota.

Las gráficas muestran las posiciones de la pelota durante la prueba en posiciones fijas. Los círculos rojos indican el número de pruebas que encuentran la pelota y el contenedor no. El naranja y verde indican el número de pruebas que encuentran la pelota y posteriormente el contenedor. El color blanco indica el tamaño del círculo si el 100% de las pruebas tienen éxito.

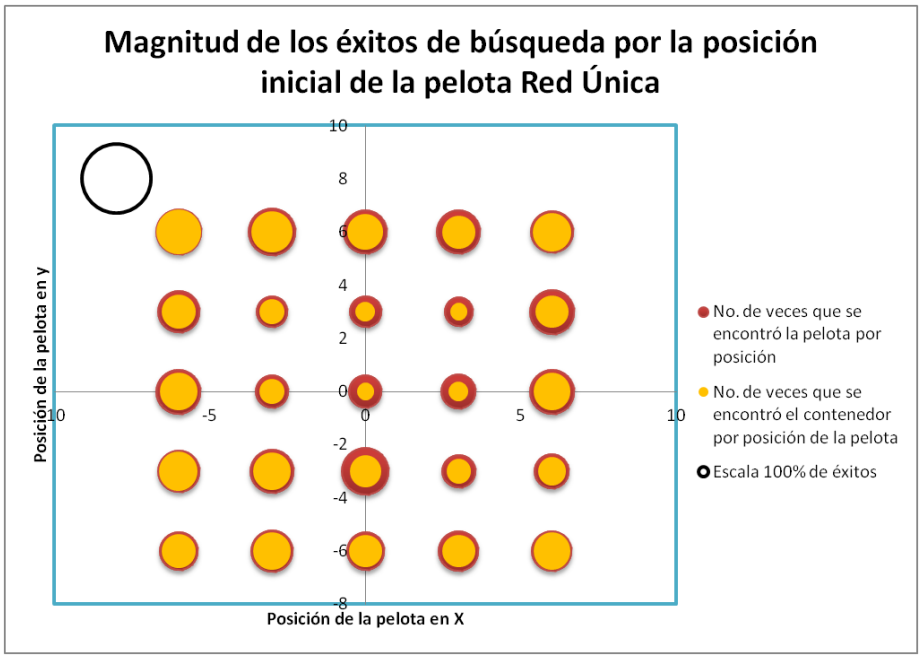


Figura C.1 Magnitud de los éxitos por la posición de la pelota, Red Única.

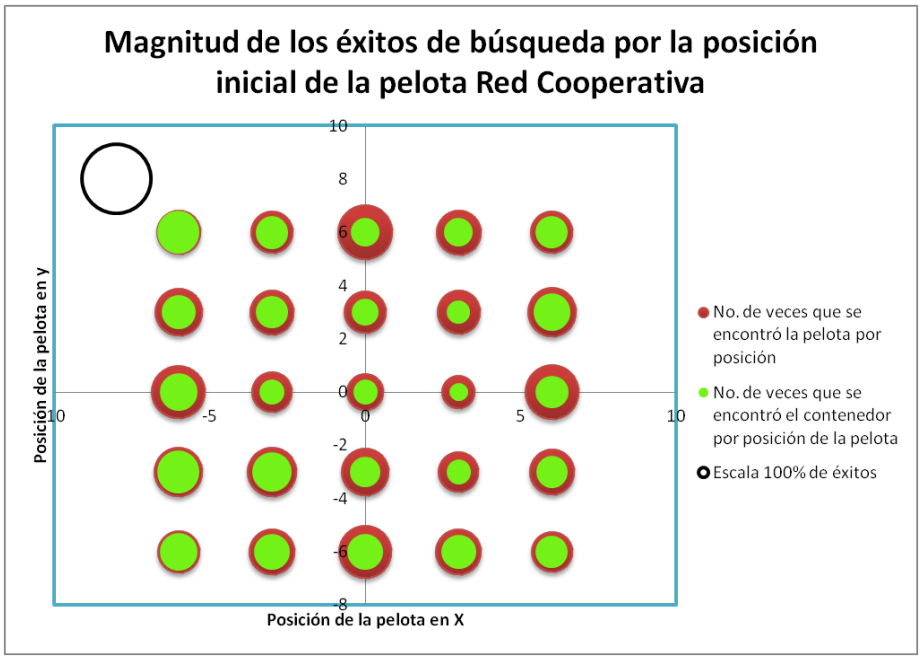


Figura C.2 Magnitud de los éxitos por la posición de la pelota, Red Cooperativa.

Anexo D Tiempo de búsqueda por posición de la pelota.

Las gráficas muestran las posiciones de la pelota durante la prueba en posiciones fijas. El tamaño del círculo indica el tiempo promedio que se tarda cada prueba en encontrar la pelota (rojo), o la pelota y después el contenedor (naranja y verde). Solo se toman los tiempos de las pruebas que tuvieron éxito. El círculo de color claro indica el 100% del tiempo, los círculos más pequeños indican menos tiempo.

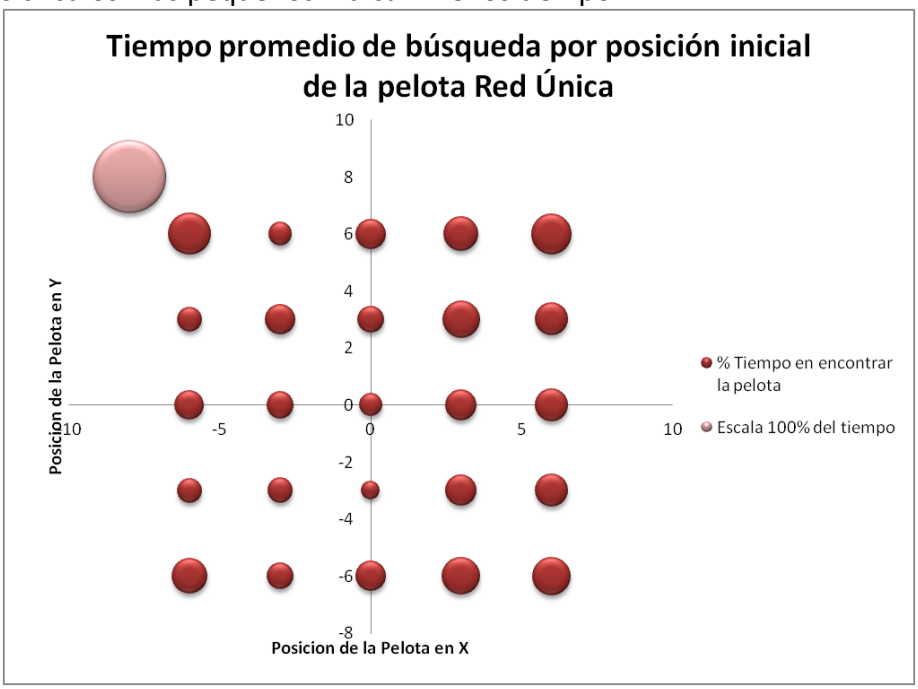


Figura D.1 Tiempo promedio de búsqueda de la pelota por posición inicial de la pelota, Red Única.

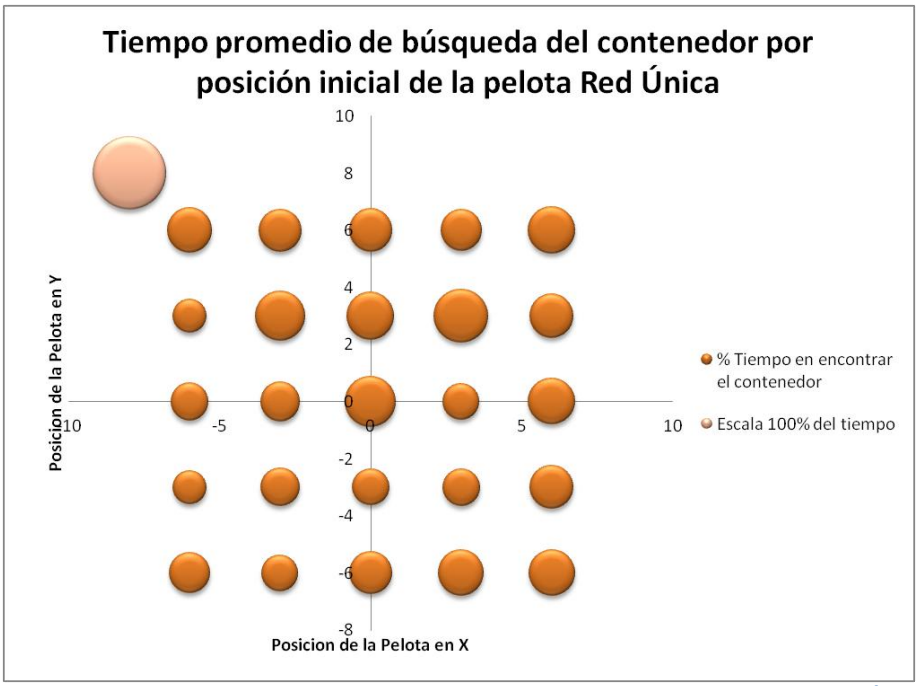


Figura D.2 Tiempo promedio de búsqueda del contenedor por posición de la pelota, Red Única.

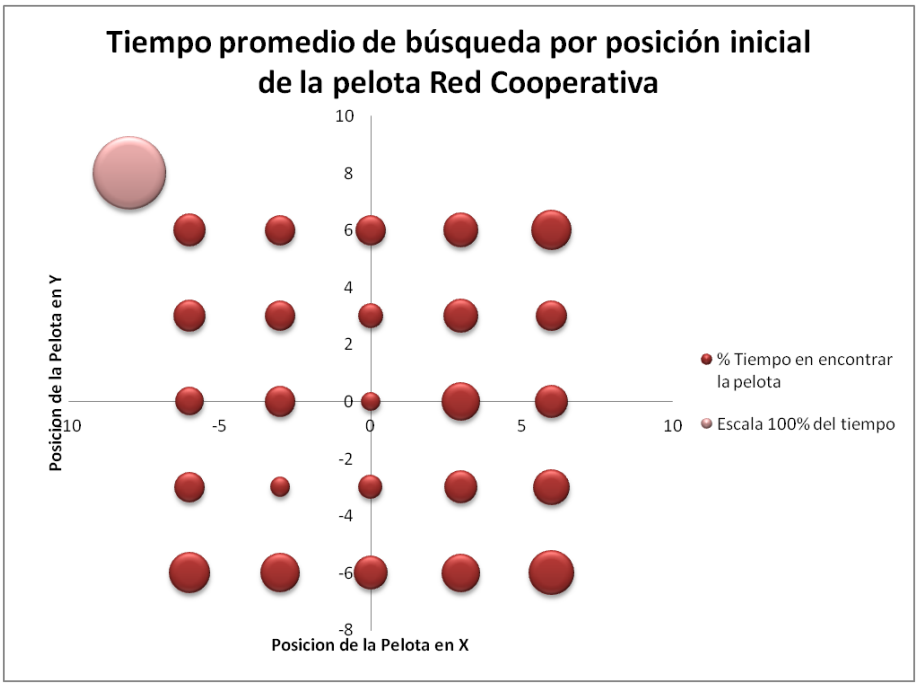


Figura D.3 Tiempo promedio de búsqueda de la pelota por posición inicial de la pelota, Red Cooperativa.

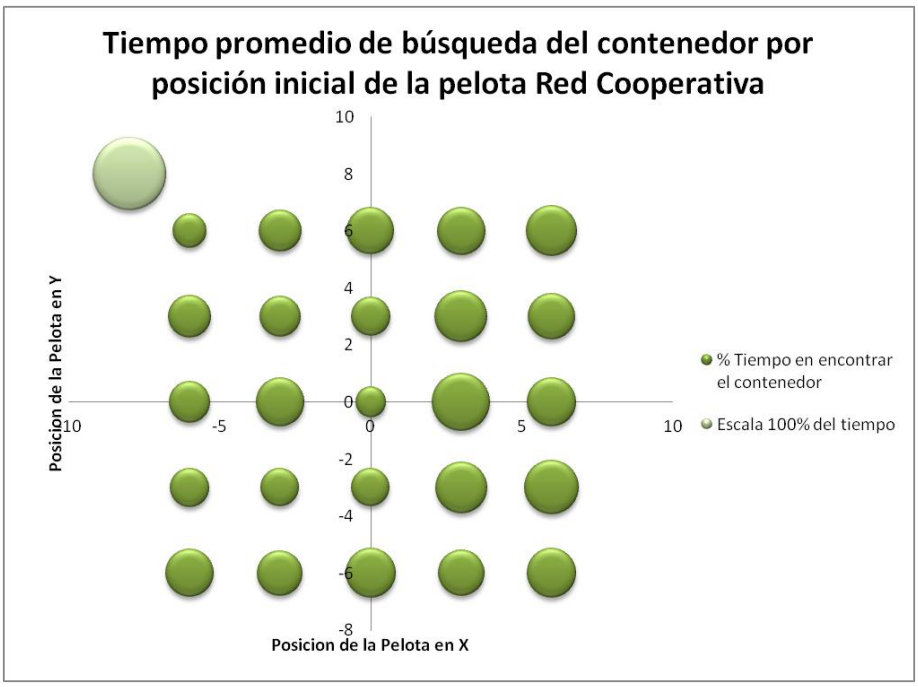


Figura D.4 Tiempo promedio de búsqueda del contenedor por posición de la pelota, Red Cooperativa.

Anexo E Código Fuente

Operadores Genéticos

Red Única

```
// Algoritmo de Selección
void seleccion(void)
  {
    int i,j,k,l,d;
    for(i=0;i<Num_indiv;i++) //Todos los individuos a seleccionar de la generacion
      {
        k = (rand()/(RAND_MAX+0.01))*Num_indiv; //Seleccionar al primer individuo
        for(j=1;j<Torneo;j++)                   //Para los siguientes individuos
          {
            l = (rand()/(RAND_MAX+0.01))*Num_indiv; //Selec. Siguiente individuo
            if(NLestado[k]<NLestado[l])         //Dejar en K al mejor Calificado
              k = l;
          }
        for(d=0;d<Dimensiones;d++)//Traspasa pesos del indiv. seleccionado

          {NLp_sig[i][d] = NLpesos[k][d];
          }
        NLedo_sig[i] = NLestado[k];
      }
  }


// Algoritmo de cruce
void cruce(void)
  {
    double xnew1,xnew2;
    int i,j,k,d;
    float lambda;
    int h,g;
    for(i=0;i<(Num_indiv*Prob_Cruce/2);i++) //Para los individuos a cruzar
      {
         lambda = (rand()/(RAND_MAX+0.01)); //Punto de cruce del cromosoma
         j=(rand()/(RAND_MAX+0.01))*Num_indiv;  // Primer individuo
         k=(rand()/(RAND_MAX+0.01))*Num_indiv;  // Segundo individuo
         h=(rand()/(RAND_MAX+0.01))*Dimensiones/2;  //Numero de cromosomas a cambiar
         for(g=0;g<h;g++)
           {
             d=((rand()/(RAND_MAX+0.01))*Dimensiones);
             xnew1 = NLp_sig[j][d]*lambda+NLp_sig[k][d]*(1-lambda);
             xnew2 = NLp_sig[j][d]*(1-lambda)+NLp_sig[k][d]*lambda;
             NLp_sig[j][d] = xnew1;
             NLp_sig[k][d] = xnew2;
           }
      }
  }
```

```
// Algoritmo de mutación
void mutacion(void)
  {
    float r,h;
    int i,j,q;

    for (i=0;i<Num_indiv;i++)                          // Todos los individuos
      {
        r = (rand()/(RAND_MAX+0.01));                  //Número aleatorio
        if(r<=Prob_Muta*(1-(contadorgen/MAXIMAGEN)))  //Se aplica o no el operador
          {
             q = (rand()/(RAND_MAX+0.01))*Dimensiones;   //Cual gen mutar x[0]...x[r]
             h = (rand()/(RAND_MAX+0.01));               //Incremento o decremento
             if (h<0.5)
             NLp_sig[i][q]=NLp_sig[i][q]+((rand()/(RAND_MAX+0.01))*(NLp_sig[i][q]));
             else
             NLp_sig[i][q]=NLp_sig[i][q]-((rand()/(RAND_MAX+0.01))*(NLp_sig[i][q]));
          }
      }
  }
```

Red Cooperativa

```
// Algoritmo de Seleccion
void seleccionZ(void)
  {
    int i,j,k,l,d;
    for(i=0;i<Num_indiv;i++) //Todos los individuos a seleccionar de la generacion
      {
        k = (rand()/(RAND_MAX+0.01))*Num_indiv;  //Seleccionar al primer individuo
        for(j=1;j<Torneo;j++)                    //Para los siguientes individuos
          {
            l = (rand()/(RAND_MAX+0.01))*Num_indiv;  //Selec. Siguiente individuo
            if((NLestadoD[k]+NLestadoF[k]+NLestadoI[k])<(NLestadoD[l]+NLestadoF[l]+NLestadoI[l]))
              k = l;                             //Dejar en K al mejor Calificado
          }
        for(d=0;d<Dimensiones;d++)               //Traspasa pesos del indiv. seleccionado
          {NLp_sigD[i][d] = NLpesosD[k][d];
                   NLp_sigF[i][d] = NLpesosF[k][d];
                   NLp_sigI[i][d] = NLpesosI[k][d];
          }
        NLedo_sigD[i] = NLestadoD[k];
        NLedo_sigF[i] = NLestadoF[k];
        NLedo_sigI[i] = NLestadoI[k];
      }
  }
```

```
// Algoritmo de cruce
void cruceZ(void)
  {
    double xnew1,xnew2;
    int i,j,k,d;
    float lambda;
    int h,g;
    for(i=0;i<(Num_indiv*Prob_Cruce/2);i++) //Para los individuos a cruzar
      {
         lambda = (rand()/(RAND_MAX+0.01));              //Punto de cruce del cromosoma
         j=(rand()/(RAND_MAX+0.01))*Num_indiv;   // Primer Individuo
         k=(rand()/(RAND_MAX+0.01))*Num_indiv;   // Segundo Individuo
         h=(rand()/(RAND_MAX+0.01))*Dimensiones/2; //Numero de cromosomas a cambiar
         for(g=0;g<h;g++)
           {
             d=((rand()/(RAND_MAX+0.01))*Dimensiones);
             xnew1 = NLp_sigD[j][d]*lambda+NLp_sigD[k][d]*(1-lambda);
             xnew2 = NLp_sigD[j][d]*(1-lambda)+NLp_sigD[k][d]*lambda;
             NLp_sigD[j][d] = xnew1;
             NLp_sigD[k][d] = xnew2;

             xnew1 = NLp_sigF[j][d]*lambda+NLp_sigF[k][d]*(1-lambda);
             xnew2 = NLp_sigF[j][d]*(1-lambda)+NLp_sigF[k][d]*lambda;
             NLp_sigF[j][d] = xnew1;
             NLp_sigF[k][d] = xnew2;

             xnew1 = NLp_sigI[j][d]*lambda+NLp_sigI[k][d]*(1-lambda);
             xnew2 = NLp_sigI[j][d]*(1-lambda)+NLp_sigI[k][d]*lambda;
             NLp_sigI[j][d] = xnew1;
             NLp_sigI[k][d] = xnew2;
                         }
      }
  }


// Algoritmo de mutacion
void mutacionZ(void)
  {
    float r,h;
    int i,j,q;
    for (i=0;i<Num_indiv;i++)// Todos los individuos
      {
        r = (rand()/(RAND_MAX+0.01)); //Número aleatorio
        if(r<=Prob_Muta*(1-(contadorgen/MAXIMAGEN)))   //Decidir si se aplica el operador
          {
             q = (rand()/(RAND_MAX+0.01))*Dimensiones; //Decidir cual gen mutar x[0]..x[1]..x[r]
             h = (rand()/(RAND_MAX+0.01)); //Incremento o decremento
             if (h<0.5)
              { NLp_sigD[i][q]=NLp_sigD[i][q]+((rand()/(RAND_MAX+0.01))*(NLp_sigD[i][q]));
                NLp_sigF[i][q]=NLp_sigF[i][q]+((rand()/(RAND_MAX+0.01))*(NLp_sigF[i][q]));
                NLp_sigI[i][q]=NLp_sigI[i][q]+((rand()/(RAND_MAX+0.01))*(NLp_sigI[i][q]));
              }
             else
              { NLp_sigD[i][q]=NLp_sigD[i][q]-((rand()/(RAND_MAX+0.01))*(NLp_sigD[i][q]));
                NLp_sigF[i][q]=NLp_sigF[i][q]-((rand()/(RAND_MAX+0.01))*(NLp_sigF[i][q]));
                NLp_sigI[i][q]=NLp_sigI[i][q]-((rand()/(RAND_MAX+0.01))*(NLp_sigI[i][q]));
              }
          }
      }
  }
```

Cálculos de Red Neuronal

Red Única

```
// Cálculo de la neurona

  for (ii=0; ii< Noculta; ii++) {       // Para cada neurona de la capa oculta
    suma=0;
    for (jj=0; jj< Nentrada; jj++) {    // Se acumulan todas las entradas
       suma+=NeuronaE[jj]*NeuronaPO[jj][ii];
    }
    NeuronaO[ii]=(1.0 / (1.0 + exp(-suma))); // Se calcula el valor de salida de la neurona
  }

  for (ii=0; ii< Nsalida; ii++) {       // Para cada neurona de la capa de salida
    suma=0;
    for (jj=0; jj< Noculta; jj++) {     // Se acumulan todas las entradas
       suma+=NeuronaO[jj]*NeuronaPS[jj][ii];
    }
    NeuronaS[ii]=((1.0 / (1.0 + exp(-suma)))+(-0.5))*1.12;
                // Se calcula el valor de salida
                // como el valor de salida es un rango de 0 y 1 se resta 0.5 para obtener
                // valores negativos y positivos
                // Se multiplica por 1.12 para incrementar el valor final de salida.
  }
```

Red Cooperativa

```
// Cálculo de la red neuronal 1
  for (ii=0; ii< Noculta; ii++) {
    suma=0;
    for (jj=0; jj< Nentrada; jj++) { suma+=NF_E[jj]*NF_PO[jj][ii];}
    NF_O[ii]=(1.0 / (1.0 + exp(-suma)));
  }
  for (ii=0; ii< Nsalida; ii++) {
    suma=0;
    for (jj=0; jj< Noculta; jj++) { suma+=NF_O[jj]*NF_PS[jj][ii]; }
    NF_S[ii]=((1.0 / (1.0 + exp(-suma)))+(-0.5))*1.12;
  }

// Cálculo de la red neuronal 2
  for (ii=0; ii< Noculta; ii++) {
    suma=0;
    for (jj=0; jj< Nentrada; jj++) { suma+=NI_E[jj]*NI_PO[jj][ii];}
    NI_O[ii]=(1.0 / (1.0 + exp(-suma)));
  }
  for (ii=0; ii< Nsalida; ii++) {
    suma=0;
    for (jj=0; jj< Noculta; jj++) { suma+=NI_O[jj]*NI_PS[jj][ii]; }
    NI_S[ii]=((1.0 / (1.0 + exp(-suma)))+(-0.5))*1.12;
  }

// Cálculo de la red neuronal 3
  for (ii=0; ii< Noculta; ii++) {
    suma=0;
    for (jj=0; jj< Nentrada; jj++) { suma+=ND_E[jj]*ND_PO[jj][ii];}
    ND_O[ii]=(1.0 / (1.0 + exp(-suma)));
  }
  for (ii=0; ii< Nsalida; ii++) {
    suma=0;
    for (jj=0; jj< Noculta; jj++) { suma+=ND_O[jj]*ND_PS[jj][ii]; }
    ND_S[ii]=((1.0 / (1.0 + exp(-suma)))+(-0.5))*1.12;
  }
```

Función de Eficiencia

```
if ((e < 0.45) || (g < 0.95)) { // Al choca con la pared (estar muy cerca)
    progresofinalF-=0.1;        // Se da una calificación negativa
    contadorretro=10;           // Se inicia el contador para retroceder 10 tiempos
  }

  else{
     if (c < SENSORLON)         //Al detectar la pelota de frente
     {   if (fabs((NF_S[1]+NI_S[1]+ND_S[1])-(NF_S[0]+NI_S[0]+ND_S[0]))< 0.05)
         { progresofinalD+=0.1; }       //Si sigue de frente calificación positiva
         else
         { progresofinalD-=0.1; }       //Si va de lado calificación negativa
     }
     if ((sensorP[3] < SENSORLON) || (sensorP[5] < SENSORLON) || (sensorP[7] < SENSORLON))
     {                                  //Detecta pelota por el lado izquierdo
        if (((NF_S[1]+NI_S[1]+ND_S[1])-(NF_S[0]+NI_S[0]+ND_S[0]))< -0.01)
        {  progresofinalI+=0.1; }       // A la izquierda, calificación positiva
        else
        {  progresofinalI-=0.1; }       //No a la izquierda, calificación negativa
        }
     }
     if ((sensorP[4] < SENSORLON) || (sensorP[6] < SENSORLON) || (sensorP[8] < SENSORLON))
     {                                  //Detecta pelota por el lado derecho
       if (((NF_S[0]+NI_S[0]+ND_S[0])-(NF_S[1]+NI_S[1]+ND_S[1]))< -0.01)
        {  progresofinalI+=0.1; }       // A la derecha, calificación positiva
       else
        {  progresofinalI-=0.1; }       // No la derecha, calificación negativa
     }
  }

  if (c < 0.5 ) {                       //Si toma la pelota
    tomaPelota=1;
    progresofinalF+=1000;               // Recompensa neurona 1
    progresofinalI+=1000;               // Recompensa neurona 2
    progresofinalD+=1000;               // Recompensa neurona 3
    dBodySetPosition (pelotaC[0], PAREDL, PAREDL, PELOTAR);     // Retirar pelota de área
  }

  if (g < 1.1 && tomaPelota==1) {       //Si deposita en contenedor
    progresofinalF+=10000;
    progresofinalI+=10000;
    progresofinalD+=10000;
    detener=1;                          // Finaliza prueba
  }

  if (tomaPelota==1) {                  //Cuando ya tomó la pelota
     if (g < SENSORLON)                 //Al detectar el contenedor de frente
     {   if (fabs((NF_S[1]+NI_S[1]+ND_S[1])-(NF_S[0]+NI_S[0]+ND_S[0]))< 0.05)
         { progresofinalD+=0.1; }       //Si sigue de frente calificación positiva
         else
         { progresofinalD-=0.1; }       //Si va de lado calificación negativa
     }
     if ((sensorC[3] < SENSORLON) || (sensorC[5] < SENSORLON) || (sensorC[7] < SENSORLON))
     {                                  //Detecta contenedor por el lado izquierdo
        if (((NF_S[1]+NI_S[1]+ND_S[1])-(NF_S[0]+NI_S[0]+ND_S[0]))< -0.01)
        {  progresofinalI+=0.1; }       // A la izquierda, calificación positiva
        else
        {  progresofinalI-=0.1; }       //No a la izquierda, calificación negativa
     }
     if ((sensorC[4] < SENSORLON) || (sensorC[6] < SENSORLON) || (sensorC[8] < SENSORLON))
     {                                  //Detecta contenedor por el lado derecho
        if (((NF_S[0]+NI_S[0]+ND_S[0])-(NF_S[1]+NI_S[1]+ND_S[1]))< -0.01)
        {  progresofinalI+=0.1; }       // A la derecha, calificación positiva
       else
        {  progresofinalI-=0.1; }       // No la derecha, calificación negativa
     }
  }
```

Bibliografía

[Alavala 2008] Chennakesava R. Alavala. Fuzzy logic and Neural Networks, basic concepts & applications, Ed. New Age International Publishers, 2008.

[Amalendu y otros 2004] Amalendu Patnaik, Dimitrios E. Anagnostou, Rabindra K. Mishra, Christos G. Christodoulou, and J. C. Lyke. Applications of Neural Networks in Wireless Communications,IEEE Antennas and Propagation Magazine, Vol. 46, No. 3, June 2004

[Amigoni y Gallo 2005] Francesco Amigoni and Alessandro Gallo. A Multi-Objective Exploration Strategy for Mobile Robots, Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference.

[Anderson 2007] James A. Anderson, Redes Neuronales. Alfaomega 2008.

[Arbib 1995] Arbib, M. Handbook of brain theory and neural networks. The MIT Press, 1995.

[Avila 2008] Victor Germán Avila Cienega. TESIS, Generación de caminos en mapas 2D utilizando técnicas heurísticas sobre mallado automático no estructurado, Instituto Tecnológico de León, Octubre 2008.

[Baase 1998] Sara Baase. Computer Algorithms Introduction to design and analisis, second edition. Addison-Wesley Publishing Company, 1998. Pags, 323-329.

[Bellas 2003] Bellas Bouza, Fancisco Javier. Mecanismo cognitivo darwinista para agente autónomos, Tesis Doctoral, Julio de 2003, Universidad de Coruña.

[Ramon y Cajal 1906] Santiago Ramón y Cajal. The structure and connexions of neurons. Nobel Lecture, December 12, 1906

[Darwin 1859] On the Origin of Species, Charles Robert Darwin, 1859.

[Dawkins 1986] Richard Dawkins. The Blind Wathmaker. Longman Scicentific & Technical. Longman Group UK Limited 1986.

[De Jong 2004] De Jong, E., Thierens, D., and Watson, R. Defining modularity, hierarchy, and repetition. In Proceedings of the GECCO Workshop on Modularity, regularity and hierarchy in open-ended evolutionary computation, 2004, pages 2–6.

[Díez 1998] F. J. Díez. Introducción al Razonamiento Aproximado, Dpto. Inteligencia Artificial UNED, Octubre 1998.

[Duda y otros 2001] Duda Richard O., Hart Peter E. y Stork David G., Pattern Classification, Ed. John Wiley & Son, Inc. 2001.

[Engelbrecht 2005] Andries P. Engelbrecht. Fundamentals of Computational swarm intelligence. Ed. John Wiley & Son, Ltd, 2005.

[Fuller 1991] James L. Fuller. Robotics, Introduction programing and projects, Prentice-Hall / Englewood Cliffs,NJ 07632, 1991.

[Heaton 2008] Heaton, Jef. Introduction to neural network for java , second edition. Heaton Res Inc. 2008.

[Hogg y otros 1991] Hogg David W., Martin Fred y Resnick Mitchel. Braitenber Creatures, Massachusetts Institute of technology epistemology and learning group, E&L Memo No. 13 June 1991.

[Holland 2001] Holland, J.H., Adaptation in Natural and Artificial System. MIT Press, 2001. Primera Edición 1975, The University of Michigan.

[Jakimovski 2011] Bojan Jakimovski. Biologically Inspired Approaches for Locomotion, Anomaly Detection and reconfiguration for Walking robots, (pags 24-25) Springer-Verlang 2011 Berlin Heidelberg.

[Jamshidi 2002] Jamshidi, M., et al., Robus Control System With Genetic Algorithms. Control Series, Ed. R.H. Bishop. 2002, USA:CRC Press. 210.

[Jordan 1990] Michael I. Jordan. Artificial neural networks: concept learning table of contents. IEEE Press Piscataway, NJ, USA ©1990, páginas 531 a 546.

[Jordan y Jacobs 1991] M. I. Jordan and R. A. Jacobs. A competitive modular connectionist architecture. In Advances in Neural Information Processing Systems 3, pages 767–773, San Maeto, CA, 1991. Morgan Kaufmann Publisher Inc.

[Kim y Hallam 2001] DaeEun Kim and John Hallam. Mobile Robot Control Based on Boolean Logic with Internal Memory, ECAL '01 Proceedings of the 6th European Conference on Advances in Artificial Life, 2001. Springer-Verlag London, UK ©2001

[Maroto y otros 2002] Concepción Maroto Alvarez, Javier alcaraz Soria, Ruben Ruiz Garcia. Investigación operativa: Modelos y Técnicas de Optimización, Editorial Universidad Politécnica de Valencia, 2002.

[Martin y Sanz 2007] Martín del Brío Bonifacio y Sanz Molina Alfredo. Redes Neuronales y Sistemas Borrosos. Tercera edición, Alfaomega / Ra-Ma, 2007.

[Michalewicz 1999] Zbigniew Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer 1999.

[Montes y otros 2010] Montes González Fernando, Ochoa Ortiz Zezzatti Carlos Alberto, Marín Urías Luis Felipe, y Sánchez Aguilar Jöns. Un Enfoque Híbrido en el Desarrollo de

Robótica Basada en el Comportamiento. Instituto Politécnico Nacional Revista: Computación y Sistemas Vol. 13 No. 4, 2010.

[Russell y Norvig 2004] Russell Stuart, Vorvig Peter. Inteligencia Artificial un enfoque modern. Pearson / Prentice Hall. 2004.

[Sanchez y Alanís 2006] Edgar Nelson Sánchez Camperos y Alma Yolanda Alanís Garcia. Redes Neuronales (Conceptos fundamentos y aplicaciones a control automático), Prentice hill – Pearson 2006.

[Smith 2004] Russell Smith. OPEN DYNAMICS ENGINE V0.5 USER GUIDE, MAY, 2004.

[Stuart-Norvig2008] Stuart Russell y Peter Norvig. Inteligencia Artificial un enfoque moderno, Editorial Pearson-Prentice hall, 2008.

[Tellez 2010] Ricardo A. Tellez. Distributed neural control for complex autonomous robots. Tesis Doctoral, Universidad Politécnica de Cataluna, Febrero 2010.

[Tellez y Angulo 2004] Ricardo A. Tellez y Cecilio Angulo. Generando un Agente Robótico autónomo a partir de la evolución de sub-agentes simples cooperativos, In W WORKSHOP EN AGENTES FISICOS [84-933619-6-8], paginas 113-118. 2004.

[Toal 1996] Daniel Toal, Colin Flanagan, Caimin Jones y Bob Strunz. Subsumption architecture for the control robot, University of Limerick. 1996

[Tsoukalas y otros 1997] Lefteri H. Tsoukalas, Robert E. Uhring, Lotfi A. Zadeh. Fuzzy and Neural Approaches in Engineering. Editorial John Wiley & Sons, Inc., 1997. Pagina 281.

[Turhan 1995] Dr. M. Turhan (Tury) Taner. Neural Networks and computational of neural network weights and biases by the generalized delta rule and back-propagation of errors. Rock Solid Images (RSI), Neural Networks and their supervised training, 1995.

[Zadeh 1994] Zadeh, L.A., “Soft Computing and Fuzzy Logic”. IEEE Software, 1994. 11(6): Páginas 48-56.