UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Facultad de Ciencias

Obtención de un Neurocontrolador para un Robot Autónomo usando Algoritmos Genéticos

T E S I S

Que para obtener el título de:

Licenciado en Ciencias
(Ciencias Computacionales)

Presenta:

Abraham Martínez Allende

Director de tesis: Dr. Bruno Lara Guzmán

Cuernavaca, Morelos, México. 20 de mayo de 2011

Resumen

La investigación presentada en este documento tiene como finalidad la obtención de un neurocontrolador para un agente artificial autónomo; este trabajo está ubicado en el marco de la robótica cognitiva y se utilizaron técnicas de evolución artificial para la obtención del controlador.

Dentro de la robótica evolutiva las redes programadas genéticamente son un modelo de reciente creación, éstas son estructuras que tienen asociado un programa en cada uno de sus nodos; así mismo favorecen la evolución de las conexiones entre los elementos de la red debido a que la configuración de la red cambia en función de la expresión sintáctica de los programas evolucionados.

En el presente trabajo mostramos los resultados experimentales de un conjunto de pruebas realizadas a dos tipos de arquitecturas de redes programadas genéticamente; éstas redes conforman el controlador de un agente que tenía como meta el deambular en su ambiente sin colisionar. Cada red fue codificada con una arquitectura inicial que cuenta con 8 entradas con las cuales recibirá información de su ambiente, dos nodos en la capa de salida y una capa oculta con uno ó cuatro nodos dependiendo de la prueba.

Utilizando esta arquitectura inicial generamos programas aleatorios y aplicamos el proceso evolutivo a los nodos de la red. Las ocho entradas de la red corresponden a sensores de distancia que se encuentran al frente del agente, la capa de salida produce los comandos motrices para los dos motores.

El proceso evolutivo se llevo a cabo en una arena con obstáculos fijos, Los controladores evolucionados se probaron en arenas distinatas a las utlizadas para por el proceso evolutivo. Ambos procesos fueron desarrollados bajo la plataforma de simulación MobileSim diseñada para el robot Pioneer P3-XD.

Índice general

Índice de figuras

Capítulo 1

Introducción

La obtención de un controlador para un robot autónomo es un problema ampliamente estudiado dentro del campo de la robótica, sin embargo, este problema aún está vigente y se exploran nuevas alternativas para solucionarlo. En este trabajo de tesis se busca desarrollar un neurocontrolador a fin de que el robot pueda deambular sin colisionar con los objetos de su ambiente.

El trabajo aquí presentado fue situado dentro del paradigma de la cognición embebida y se usaron técnicas de evolución artificial para el desarrollo del agente, el neurocontrolador fue codificado mediante el uso de una red neuronal artificial. Para este propósito utilizamos un tipo de redes neuronales artificiales conocidas como redes programadas genéticamente (GPN). Estas redes están situadas en el marco teórico de la evolución artificial, específicamente dentro de la robótica evolutiva, la cual nos brinda un paradigma en el que el diseñador no interviene en el desarrollo del agente.

Estas redes usan programación genética en sus nodos para generar aleatoriamente la red, conservando siempre la arquitectura inicial propuesta para la resolución del problema pero evolucionando las conexiones entre los elementos de la red. El uso de estas redes permite que el proceso evolutivo genere por si mismo una gran cantidad de arquitecturas diferentes sin ninguna restricción en el tipo de conexiones.

Los experimentos se llevaron a cabo utilizando el simulador del robot Pionner P3-DX (MobileSim) para el proceso evolutivo, posteriormente las pruebas a los individuos se realizarán utilizando ambientes diferentes a los utilizados en la fase de aprendizaje para comprobar el funcionamiento del controlador.

1.1. Inteligencia artificial tradicional

La inteligencia artificial nace como una disciplina que trata de reproducir las formas de pensamiento y comportamiento del ser humano. Durante sus inicios se pensaba que una computadora funcionaba de la misma forma que lo hace el cerebro, esta corriente es conocida como paradigma de procesamiento de información.

Esta corriente trabajó bajo la idea de que los procesos mentales podían ser situados al mismo nivel que los algoritmos o métodos computacionales, sin considerar al agente en el que se implementara el algoritmo; se pensaba que los procesos cognitivos en el cerebro funcionaban de una forma lineal, ante una entrada sensorial el cerebro tenia módulos de procesamiento de información que transformaban estas situaciones sensoriales hasta convertirlas en comandos motrices. Así se creía que sólo hacia falta que se desarrollara el software adecuado que modelara la información que los humanos poseen asi como la creación de reglas de producción para que las computadoras manipularan este conocimiento de la misma forma que los humanos lo hacen.

La meta de la inteligencia artificial tradicional era la de desarrollar un agente inteligente, para lo cual se puso como meta la resolución de tareas conocidas como de alto nivel; se sabe que éstas necesitan procesos cognitivos complejos; algunos ejemplos de ellas son: jugar ajedrez, resolver problemas, sistemas lógicos y demostraciones matemáticas.

Con el paso de los años, el campo de la inteligencia artificial tradicional fue capaz de resolver tareas de alto nivel de una forma muy eficiente, el mejor de ejemplo de esto se da cuando en 1996 la computadora de IBM Deep Blue le ganó un duelo al campeón mundial de ajedrez; sin embargo estos ordenadores sólo ejecutaban operaciones aritméticas y manipulaban símbolos de una forma muy eficiente sin que presentaran comportamiento inteligente.

En el año de 1990 el investigador americano Rodney Brooks publicó el artículo "Elephants don't play chess" [1] en el cual plantea una serie de argumentos en contra del paradigma del procesamiento de información; bajo estos argumentos se comienza a pensar que no era posible desarrollar un agente inteligente con las reglas definidas por el paradigma co el que se habia trabajado hasta entonces. En este artículo Brooks argumenta que el paradigma de procesamiento de información estaba basado en la manipulación de símbolos, así que los agentes hacían estas operaciones sin la necesidad de que supieran lo que realizaban o por qué lo realizaban.

Otro de los argumentos de Brooks era que se trataba de descomponer la inteligencia del ser humano en módulos de procesamiento de información que en conjunto nos llevaría a desarrollar un comportamiento deseado, es decir que para un comportamiento global el diseñador necesitaba inferir todos los comportamientos básicos necesarios para desarrollar un comportamiento más complejo.

El problema con este enfoque es que para tareas de alto nivel en ambientes dinámicos resulta muy complicado inferir todos los comportamientos necesarios para que el agente pueda desarrollar comportamiento inteligente. Brooks argumenta que el tiempo en términos evolutivos que le ha tomado a la naturaleza desarrollar las capacidades para realizar tareas de bajo nivel es mucho mayor del que nos ha tomado desarrollar las capacidades para aprender a resolver tareas de alto nivel [2].

Por lo tanto, Brooks propone que la investigación en el campo de la inteligencia artificial debería comenzar a estudiar estas tareas básicas que se creía que no tenían gran relevancia en los procesos cognitivos. Éstas también son conocidas como tareas de bajo nivel, y son todas aquellas que se realizan de forma automática tales como caminar, correr, evitar obstáculos o buscar una fuente de alimentación. Ahora la inteligencia artificial estudia los procesos cognitivos presentes en la resolución de tareas de bajo nivel para comprender como se llevan a cabo y después reproducirlos en agentes artificiales.

A partir de las ideas de Brooks y muchos otros investigadores y pensadores ([3] [4] [5], surgieron distintos argumentos en el enfoque de la inteligencia artificial tradicional que evidenciaban que este enfoque no sería capaz de lograr la meta de desarrollar un agente inteligente y aún cuando algunos de ellos continúan en discusión, estos argumentos han guiado a la inteligencia artificial a la creación de nuevos paradigmas de investigación.

1.1.1. Mundos virtuales contra el mundo real

Dentro de los argumentos que han surgido en la inteligencia artificial uno de los más conocidos tiene que ver con la complejidad que tiene un agente para desarrollarse y resolver problemas en un ambiente real o simulado.

Las tareas de alto nivel con las que se había trabajado en la inteligencia artificial eran problemas que podían modelarse de forma discreta y bien definida, por ejemplo, para un problema como el ajedrez podemos definir un mundo virtual discreto donde conocemos

estados (posiciones en el tablero), operaciones (movimientos) y restricciones (movimientos legales).

En la inteligencia artificial tradicional los agentes se enfrentaban con problemas reacitvos que ante una entrada producían una sola salida, sin embargo un agente con un problema en un ambiente real o simulado recibe información de forma continua de él y sus acciones inciden directamente en la forma en la que percibe el agente.

Un problema como el de la robótica móvil representa un mundo dinámico complicado de modelar por completo [6]. Es decir, cualquier agente físico con un problema situado en un ambiente está expuesto a fenómenos como ruido, obstáculos y descomposturas entre otros, todos estos son fenómenos que usualmente no están considerados en problemas abstractos [7].

1.1.2. El problema de la encarnación *(Embodiment and situatedness)*

Ya hemos señalado la necesidad de que los problemas dentro de la inteligencia artificial sean situados en un ambiente real o simulado debido a que sólo de el podemos obtener una representación válida para que el agente se desarrolle. El argumento de la encarnación nos dice que los algoritmos no son capaces de interactuar con el mundo real, sino que para desarrollar inteligencia en un agente artificial se requiere que el agente tenga un cuerpo.

Partiendo de esta necesidad, es indispensable que el agente esté situado, es decir que este debe desarrollarse en un ambiente con su propio cuerpo del cual pueda obtener información sensorial y que sus acciones afecten directamente su percepción del mundo. Actualmente se piensa que la autonomía de un agente en realidad es una propiedad emergente de su interacción motriz con el ambiente en el que se desarrolla [8].

Un agente situado es capaz de desarrollarse en su ambiente sin la intervención del diseñador, además que los agentes que tienen un cuerpo y que se desarrollan en él no sufren del problema de la cimentación de símbolos (symbol grounding).

1.1.3. Cimentación de símbolos (The symbol grounding problem)

Hasta el momento los sistemas que se han desarrollado para resolver problemas abstractos han logrado su meta de una buena manera, sin embargo estos sistemas traba-

jaban mediante la representación simbólica de mundos y problemas discretos, es decir el programador era el que interpretaba y daba significado a estos símbolos.

Dentro del campo de la inteligencia artificial, uno de los argumentos más conocidos es el del cuarto chino propuesto por John Searle [3]. Este argumento propone un experimento mental en el cual estamos situados en un cuarto cerrado y solo es posible comunicarnos con el exterior mediante dos orificios, un orificio de entrada en el cual recibimos caracteres en hojas y un orificio de salida por el cual transmitimos caracteres en hojas.

Para ello tenemos un libro con reglas de producción que nos dice que si recibimos un cierto conjunto de caracteres debemos transmitir por la ranura de salida otro conjunto de caracteres. Si en este experimento pensamos que los caracteres que entran en la ranura de entrada son caracteres en chino que forman preguntas en chino y que los caracteres que transmitimos por la ranura de salida forman respuestas a las preguntas en chino, para un observador externo será posible decir que el cuarto o lo que esté dentro del cuarto sabe chino, sin embargo es fácil darse cuenta que no es así.

Un agente que sólo manipula símbolos no los entiende ni tampoco es capaz de comprender la relación de estos con los objetos del mundo real, por lo tanto la propuesta de la nueva inteligencia artificial es que los agentes se desarrollen en su ambiente para que sean capaces de obtener una interpretación propia de los símbolos y la relación de estos con los objetos en su ambiente. Asi al desarrollarse en su ambiente el agente habrá creado un modelo interno propio del mundo en el que se desenvuelve y de los objetos con los que interactúa [9].

1.2. Nueva Inteligencia Artificial

El paradigma del procesamiento de información o inteligencia artificial tradicional dominó el campo de la inteligencia artificial durante sus primeros años, sin embargo los resultados arrojados por éste no mostraron grandes avances en la meta de desarrollar una agente inteligente.

La nueva inteligencia artificial como una rama de las ciencias cognitivas trata de comprender como funciona la inteligencia de los agentes biológicos utilizando técnicas computacionales como una forma de reproducirla en agentes artificiales. Este nuevo paradigma también es conocido como el paradigma de las ciencias cognitivas (embodied cognitive paradigm), dentro de las cuales encontramos a ciencias como la psicología, neurociencias y

filosofía de la mente. En conjunto éstas se encargan del estudio interdisciplinario de cómo es que el cerebro percibe, procesa y utiliza las situaciones sensoriales provenientes del ambiente en el que se desarrolla para producir comportamiento coherente.

El nuevo enfoque de la inteligencia artificial trata de usar agentes que cuentan con un cuerpo para asignarles tareas de bajo nivel con la idea de comprender los procesos de cognición involucrados en la realización de estas tareas [2]. El abordar problemas de bajo nivel nos puede ayudar a la comprensión de los procesos de pensamiento involucrados al resolverlos, de forma tal que puedan aportarnos ideas para que en un futuro podamos trabajar problemas más complejos.

1.2.1. Definición de agente autónomo

La definición de agente autónomo nace de la observación del comportamiento de agentes biológicos, un agente biológico autónomo realiza diferentes tareas tales como desplazarse y alimentarse entre otras. Análogamente a lo observado en agentes biológicos, la definición de agente autónomo es la de un agente físico que sea capaz de realizar las tareas necesarias para sobrevivir en ambientes complejos sin ningún tipo de intervención externa [4].

En robótica un agente autónomo es aquel capaz de llevar a cabo la tarea que se le ha asignado sin intervención humana en el control o desarrollo del agente. La disminución de la intervención del diseñador en los procesos de desarrollo y aprendizaje elimina la necesidad de darle al agente una representación interna de la tarea que va a realizar, así la interpretación del mundo que el agente desarrolle será inherente a sus capacidades y limitaciones.

Si el diseñador no provee una representación interna de la tarea que va a realizar, entonces el agente habrá generado por sí mismo la representación más apropiada para la tarea que se le ha dado en función de sus necesidades.

1.2.2. Robótica Cognitiva

La nueva inteligencia artificial ha prestado atención al marco teórico que las ciencias cognitivas ofrecen, como una herramienta útil para lograr la meta desarrollar un agente inteligente. Las ciencias cognitivas están preocupadas por estudiar fenómenos presentes en

el cerebro humano, tales como intencionalidad, conciencia, inteligencia y percepción.

Los avances dentro de las ciencias cognitivas nos brindan herramientas para poder reproducir comportamiento inteligente en agentes artificiales, mientras que los estudios en el campo de la robótica cognitiva aportan nuevas ideas que puedan facilitar la comprensión de los procesos de cognición en el cerebro humano.

Estos procesos cognitivos se llevan a cabo en nuestro sistema nervioso central y en conjunto nos definen como agentes inteligentes, en particular la robótica cognitiva está interesada en la reproducción de fenómenos cognitivos en agentes artificiales [9]. Asi para modelar un proceso cognitivo primero es necesario que estos fenómenos puedan ser definidos de una forma correcta, lo que nos permitiría crear modelos capaces de reproducir estos fenómenos en un agente artificial.

1.2.3. Redes Neuronales Artificiales

Una red neuronal artificial (RNA) es un modelo simplificado de procesamiento inspirado en el funcionamiento del cerebro del ser humano, este modelo es una abstracción de la estructura interna del cerebro y de como es que se realiza la cooperación entre las neuronas en el para producir una salida. Este modelo ha tenido mucho auge en el campo de la inteligencia artificial ya que se cree que nos puede aportar ideas para comprender el funcionamiento del cerebro.

Como puede verse en la figura 1.1, una red neuronal se compone de unidades llamadas neuronas artificiales ó nodos, cada neurona recibe una entrada sensorial y toda la estructura coopera para emitir una salida. La salida en una red neuronal es establecida por tres funciones principales, la primera es conocida como función de propagación y está definida como la suma del producto de la entrada sensorial por un peso de interconexión, el cual modela la fuerza sináptica de la conexión entre dos neuronas de la red.

La segunda es la función de activación y la ultima es la de transferencia que se utilizan para ajustar los datos provenientes de las neuronas a el problema que buscamos resolver, en conjunto estas tres funciones completan el modelo de red neuronal artificial que se comporta como el cerebro humano.

El modelo de redes neuronales artificiales (RNA) ha demostrado su efectividad para la resolución de problemas en la inteligencia artificial [10][11][12], éste es uno de los métodos más usados por investigadores en todo el mundo. Es necesario considerar que el

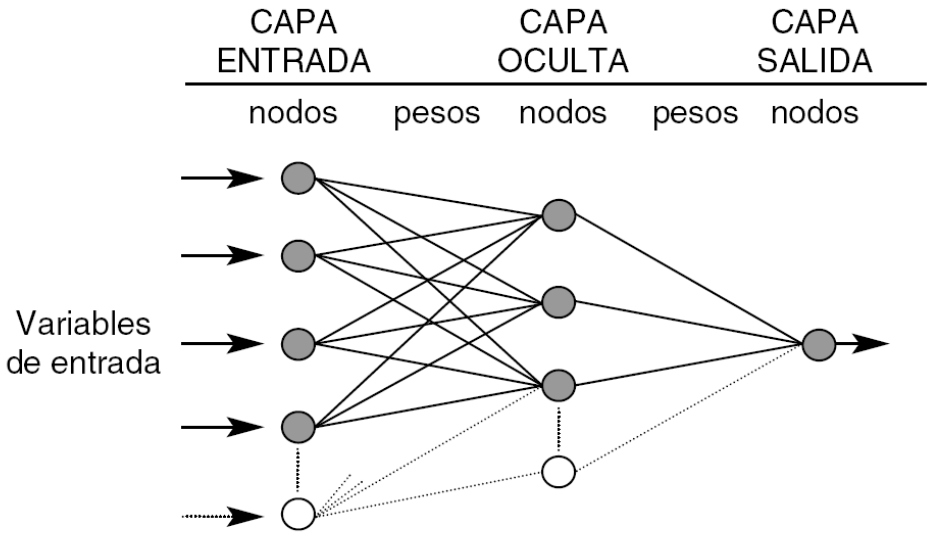


Figura 1.1: Ejemplo de una arquitectura de Red Neuronal artificial

algoritmo que se usa para resolver un problema mediante el uso de una RNA requiere que el investigador se involucre en el diseño tanto de la arquitectura como en el algoritmo de entrenamiento. A pesar de esto se ha demostrado que es posible obtener comportamiento autónomo para los controladores obtenidos con esta técnica, sin embargo la cantidad de supervisión necesaria para el desarrollo de estas redes nos deja con dudas sobre la válidez de la autonomía generada por estos algoritmos.

1.2.4. Robótica Evolutiva

En el campo de la robótica existe una gran discución sobre cuál es la cantidad ideal de supervisión para un agente si deseamos que su comportamiento sea autónomo, esto se debe a que podríamos estar incurriendo en el error de guiar el desarrollo y aprendizaje del agente.

Por lo tanto, uno de los retos para desarrollar un agente autónomo es disminuir la intervención del diseñador en su desarrollo. La robótica evolutiva es una disciplina que intenta desarrollar agentes artificiales y sus sistemas de control sensorimotrices a través de un proceso de diseño automatizado aplicando técnicas de evolución artificial.

La evolución artificial es un un método de optimización para la resolución de pro-

blemas que intenta emular el proceso de evolución natural en estructuras computacionales. La principal ventaja en su uso es que el aprendizaje está estrechamente relacionado con el desarrollo del agente en su ambiente, el proceso de selección de los mejores individuos se da en función de la eficiencia del agente para resolver una tarea mientras interactúa con su ambiente.

En un proceso evolutivo el papel del diseñador se enfoca en la definición del problema (arquitectura inicial, ambientes y función de aptitud), mientras que el desarrollo de los individuos y su aprendizaje a través de las generaciones es un proceso automático.

El diseñador propondrá una tarea como un instrumento para la evaluación de los individuos, mientras que el proceso evolutivo desarrollara los comportamientos básicos que en conjunto lo lleven a alcanzar dicha meta, de esta forma nos aseguramos de liberar al diseñador de la tarea de descomponer un comportamiento global en comportamientos básicos [8]. Bajo este enfoque el sistema nervioso del agente no es visto como un sistema de procesamiento de datos, sino que es visto como un sistema dinámico que junto al cuerpo del agente y el ambiente producen comportamiento real efectivo [13].

1.3. Estado del Arte

La búsqueda de un controlador para un robot autónomo es un problema que se ha reportado ampliamente en la literatura [14][15], en esta sección se presentará algunos trabajos realizados previamente.

1.3.1. Evolución de redes neuronales

En este primer conjunto de trabajos se buscó evolucionar una red neuronal como un método para la obtención de la estructura necesaria para resolver un problema [11][16][17], adicionalmente se evolucionan los pesos de la red para encontrar una configuración que permita a esta estructura obtener comportamiento autónomo que resuelva la tarea designada al agente. El principal problema que se presenta para poder evolucionar toda una red neuronal es la codificación del genotipo, es decir, como es que se va a representar la red neuronal de forma que sea posible aplicar los operadores genéticos a los individuos.

En el primer trabajo Yao [18] propone generar una matriz de incidencias para las conexiones de la red, para esto se genera una matriz binaria que codifica a la red completa, a continuación se deben listar las filas de la matriz para generar una cadena binaria y ésta es sometida al proceso evolutivo. Esta representación del genoma resulta fácil de implementar y utiliza pocos cálculos computacionales lo que reduce la complejidad del proceso evolutivo. En este mismo artículo se presenta una codificación de los pesos en cadenas binarias, así el proceso evolutivo es común tanto para la estructura como para los pesos de la red. El trabajo integra ambos métodos en un sistema que implementa un proceso de evolución que se extiende a todos los aspectos de una red neuronal.

En 2001 Frank Pasemann et. al. [19] proponen el algoritmo ENS^3 que es un método evolutivo para evolucionar redes neuronales utilizando probabilidad, es decir, se define un valor probabilista a las variaciones que se pueden hacer en la red. Estas variaciones pueden ser estructurales o paramétricas, permitiendo que el proceso evolutivo pueda ser extendido también para evolucionar los pesos de la red.

Este algoritmo es implementado para resolver un par de tareas, en la primera el agente tendrá que evitar obstáculos en una arena cuadrada con un arreglo de obstáculos aleatorios mientras que para el segundo de los experimentos el algoritmo se implementa para un agente seguidor de luz, es decir, se coloca una fuente de luz al centro de la arena y el robot tendrá que realizar una trayectoria para acercarse a ella iniciando en posiciones

aleatorias.

Stanley et. al. [16] proponen un genoma basado en el uso de listas ligadas, en ella se alistan todas las combinaciones que pueden existir de conexiones válidas entre los nodos de la red. En cada nodo se guarda el peso de la conexión y un bit que nos indica si la conexión existe en la red, el genoma de un individuo está representado sólo por las conexiones existentes dentro de la estructura de la red.

Este modelo fue aplicado a experimentos de balance de poleas y hasta el momento ha reportado una significativa disminución en el número de generaciones con respecto a otros métodos tradicionales, además de que el principal aporte de este articulo radica en la propiedad de que en la mayoría de las pruebas las estructuras resultantes son las estructuras más sencillas posibles.

1.3.2. Evolución utilizando programación genética

El último enfoque que hemos encontrado para la evolución de redes neuronales es la utilización de programación genética, de la cual se han efectuado varios estudios pero encontramos dos artículos que sobresalen. El primero de ellos es el realizado por Teller [20] en el cual desarrolla un modelo de memoria para las redes programadas genéticamente, para lo cual usa un arreglo de números enteros a los que se puede acceder por medio de funciones predefinidas por el usuario.

Éste trabajo es aplicado a un problema de robótica conocido como Tartarus en el cual el robot sólo puede realizar movimientos en posiciones discretas de una cuadrícula de $n * n$, el agente cuenta unicamente con dos movimientos (avanzar y girar 90°) y un número de obstáculos repartidos en el ambiente, el objetivo de éste consiste en mover los obstáculos a las casillas de la orilla del ambiente. En estas pruebas la diferencia entre la eficiencia de los controladores con memoria y sin ella es considerable, por lo que es posible deducir la importancia del manejo de memoria para la resolución de este tipo de problemas.

Otro de los estudios reportados en la literatura corresponde a Arlindo Silva et. al. en él se utilizan las redes programadas genéticamente para poder evolucionar un controlador para un agente autónomo en un ambiente discreto [21][22]. Para este trabajo se eligío el problema de la hormiga que trata de simular el comportamiento de una hormiga robótica en un mundo discreto, donde sólo realiza giros de 90° y avanza en línea recta.

El agente se desarrolló en un mundo en el cual tenemos una malla de $n * n$ estados

posibles teniendo como objetivo de este problema que la hormiga sea capaz de buscar y encontrar comida establecida en posiciones aleatorias. Los resultados de este trabajo muestran que el uso de GPN disminuye significativamente el número de individuos y generaciones necesarias para resolver el problema en comparación con otros métodos evolutivos.

Capítulo 2

Programación Genética

La programación genética es una técnica de la computación evolutiva que resuelve problemas automáticamente sin necesidad de que el diseñador conozca la forma o estructura de la solución [23], esta técnica es una variante de los algoritmos genéticos en la cual cada individuo es un programa.

La figura 2.1 muestra el algoritmo que se utiliza para programación genética, este es muy similar a el resto de los algoritmos genéticos, este busca optimizar una población de programas mediante un proceso de evolución artificial, la eficiencia de los individuos será medida con el uso de una función de aptitud que codifica la tarea designada por el diseñador.

2.1. Representación de un programa

En la programcion genética los individuos corresponden a programas, por lo que el proceso evolutivo se realiza sobre segmentos de código o expresiones válidas para un lenguaje. La forma más usada para representar los programas es mediante el uso de árboles, estas estructuras son fáciles de implementar y pueden evaluarse de una forma sencilla mediante un proceso recursivo.

Al evaluar el árbol se generan expresiones que respetan la sintaxis del lenguaje, por ejemplo al evaluarse el árbol de sintaxis de la figura 2.2 nos resulta la expresión $((x/3)+(y+5))-((x*7)*(z+9))$. En un árbol de sintaxis se debe distinguir entre dos tipos de datos definidos dependiendo de su posición en el árbol, estos dos conjuntos se definen para poder conservar la sintaxis válida de un programa o expresión sintáctica y se definen como:

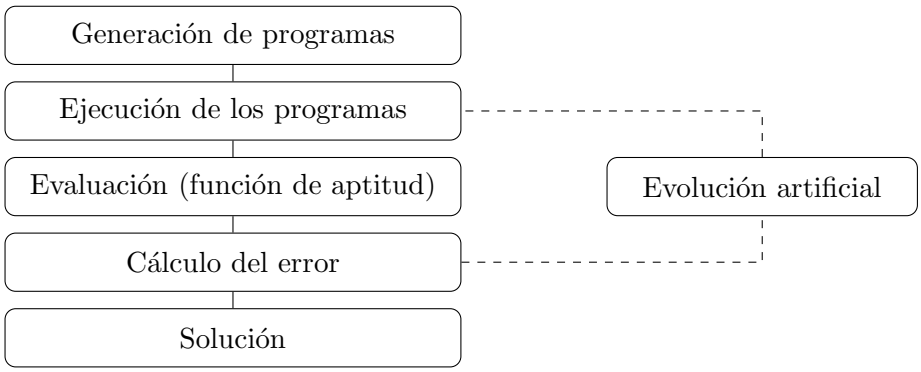


Figura 2.1: Algoritmo del proceso evolutivo en programación genética

- **Símbolos No Terminales:** Este conjunto contiene todos los operadores válidos que pueden ser utilizados durante el proceso evolutivo (aritméticos, lógicos, estructuras de flujo), estos elementos serán ubicados en los nodos internos del árbol.
- **Símbolos Terminales:** Este conjunto contiene todos los operandos válidos que pueden ser utilizados durante el proceso evolutivo (constantes, variables, funciones), estos elementos serán ubicados en los nodos hoja del árbol.

Con esta estructura los programas y expresiones podrán ser evaluados y sometidos a un proceso evolutivo de una forma muy sencilla, además de que el costo computacional de su implementación es muy bajo. El árbol como estructura de datos es muy versátil por lo cual es sencillo definir operaciones sobre su estructura que puedan actuar de una forma análoga a lo operadores genéticos encontrados en el proceso evolutivo natural.

2.2. Inicialización de una población

Como en cualquier otro algoritmo evolutivo es necesario que la población inicial sea creada de forma aleatoria, para el caso de la programación genética existen diferentes métodos para realizar esto, sin embargo en este trabajo examinaremos dos métodos principales.

Estos métodos son algoritmos que permiten el llenado de los árboles con elementos de los conjuntos de símbolos *Terminales* y *No Terminales* respetando la sintaxis del lenguaje.

El método *FULL* y el método *GROW* son métodos fáciles de implementar además

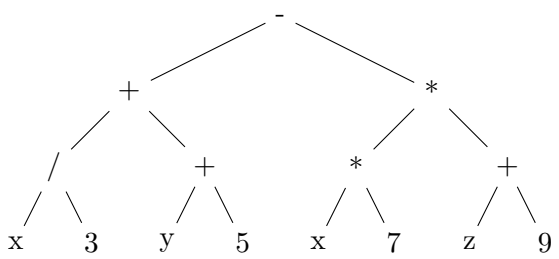


Figura 2.2: Árbol de sintaxis de un programa

de que ambos respetan una profundidad máxima para los árboles, con la ventaja de que la complejidad del computo que se necesita para su implementación es muy baja.

2.2.1. Método FULL

El método *FULL* tiene como principal ventaja el generar árboles completos, como puede observarse en la figura 2.3 este método va llenando el árbol sólo con elementos del conjunto de símbolos *No Terminales* para los nodos internos del árbol, al llegar al nivel de las hojas del árbol lo llena con elementos de conjunto de símbolos *Terminales*.

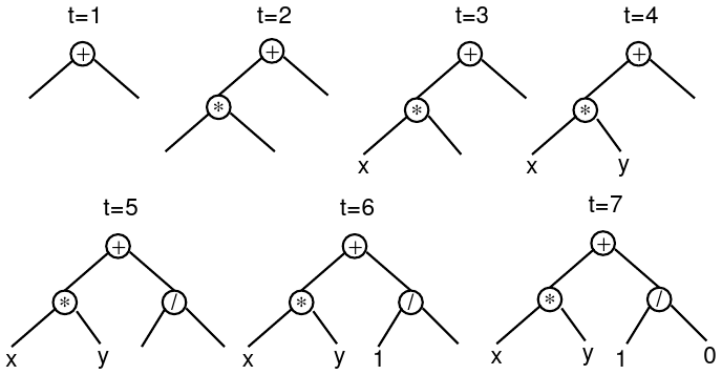


Figura 2.3: Creación de un árbol con profundidad 2 utilizando método FULL

2.2.2. Método GROW

A diferencia del método FULL este método permite generar árboles con mucha más variedad de formas y tamaños. Para este método se elige primero un elemento del conjunto de *No Terminales* para que sea la raíz, a continuación se hace un sólo conjunto definido como la unión de los conjuntos de símbolos *Terminales* y *No Terminales*.

Para cada nodo que se vaya creando se elije un elemento de este conjunto unión, si el elemento resulta ser del conjunto de *Terminales* la rama se da por terminada, en caso de ser un elemento del conjunto de *No Terminales* se continúa construyendo el subárbol debajo de este nodo, en la figura 2.4 se muestra un ejemplo de la inicialización de un árbol de profundidad 2 utilizando este método.

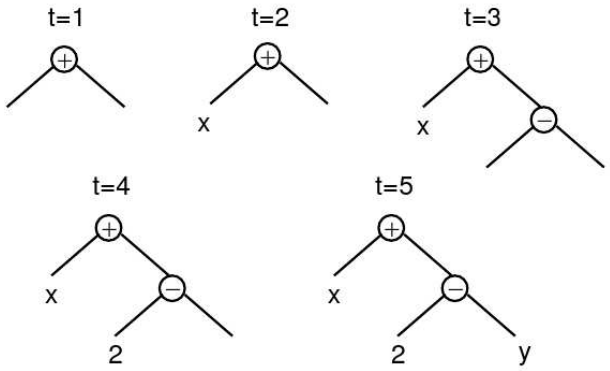


Figura 2.4: Creación de un árbol con profundidad 2 utilizando el método GROW

2.3. Operadores Genéticos

En el proceso de evolución biológica los operadores genéticos nos aseguran la diversidad de individuos, la combinación de genes entre los mejores individuos eventualmente resultará en poblaciones con individuos con valores mayores par la función de aptitud.

Una vez que tenemos la representación de los programas como árboles ahora debemos definir las operaciones o métodos computacionales para emular los operadores genéticos que conocemos. Así que para la programación genética principalmente se utilizan los operadores de selección, cruce y mutación.

2.3.1. Selección

El operador más importante dentro del proceso evolutivo es el operador de selección, éste asegura que los genes de los mejores individuos permanezcan con el paso de las generaciónes y determina cuales individuos son los más aptos. La forma de evaluar el comportamiento de un individuo mientras desarrolla una tarea es con el uso de una función de aptitud, esta será una medida obtenida en función de los comportamientos que realiza el robot y codifica la tarea que el diseñador busca que el agente desarrolle.

Para este proyecto utilizamos una selección por medio de ruleta, este operador asigna un valor de probabilidad para cada individuo en función del valor de su función de aptitud.

Esta probabilidad estará definida como :

$$P_i(t) = F_i(t) / \sum_{j=0}^{n} F_j(t)$$

- $P_i(t)$ es la probabilidad asociada al $i - esimo$ individuo
- $F_i(t)$ el valor de la función de aptitud del $i - esimo$ individuo al tiempo t
- $\sum_{j=0}^{n} F_j(t)$ la suma de los valores de la función de aptitud de los n individuos al tiempo t

Este operador asegura que el proceso evolutivo siempre tendrá la tendencia a individuos con valor más alto en la función de aptitud, la única observación es que se tiene que cuidar el tamaño de la población porque una generación con pocos individuos puede guiar el proceso evolutivo a un máximo local.

2.3.2. Cruce de subárboles

El método más utilizado de cruce en programación genética consiste en el cruce de subárboles, para este método se elige de forma aleatoria uno de los nodos de ambos árboles

(no puede ser la raíz) y se intercambia por completo todo el subárbol que tenía como raíz el nodo que elegimos, al nodo seleccionado se le conoce como punto de cruce (crossover point). En la figura 2.5 puede verse un ejemplo de como se realiza el proceso de cruce de subárboles.

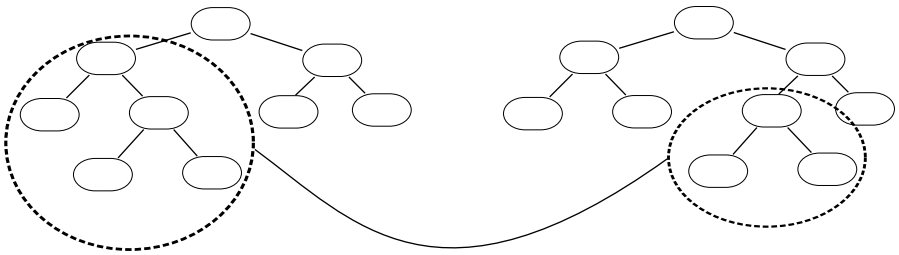


Figura 2.5: Ejemplo de cruce de subárboles

Como resultado tenemos dos árboles válidos con sintaxis válida que combinan a dos individuos para formar una nueva generación, el proceso evolutivo continuará hasta que eventualmente alcance una solución factible.

2.3.3. Mutación de subárboles

En la mutación se utilizó un método conocido como mutación de subárboles, como se muestra en la figura 2.6 este método se elije un nodo del árbol (excepto la ráiz) de forma aleatoria, posteriormente utilizando alguno de los métodos de inicialización se crea un árbol para reemplazar el que se encontraba por debajo del nodo que elegimos.

Como parte del proceso evolutivo se define una probabilidad de mutación, cabe señalar que según lo reportado en la literatura para programación genética esta probabilidad tendrá que estar entre el 1 % y el 5 %. La razón por la cual esta probabilidad es tan baja es que a diferencia de otros procesos evolutivos la mutación en programación genética representa cambios estructurales bastante significativos que pueden modificar drasticamente el comportamiento del controlador.

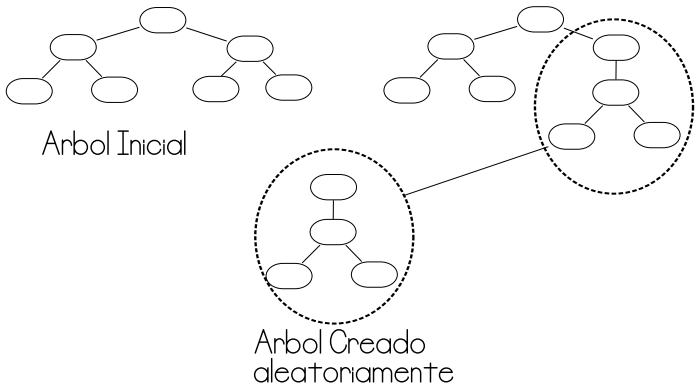


Figura 2.6: Ejemplo de Mutación de subárboles

2.4. Redes Programadas Genéticamente

Las Redes Programadas Genéticamente son un modelo de redes neuronales artificiales que usan la programación genética dentro de su arquitectura [22][21]. Para estas estructuras de reciente creación se han reportado en la literatura problemas discretos de la robótica móvil, en estos problemas el agente soló podía realizar movimientos predefinidos (giros de 90°, avanzar, retroceder).

Los resultados reportados por las redes programadas genéticamente redujeron el número de generaciones necesarias para la resolución de problemas en comparación con otras soluciones, además de que se reportó que los agentes resolvían las tareas asignadas de una manera satisfactoria.

Una GPN tiene como estructura básica un conjunto de entradas y salidas definidas para la red, además de un conjunto de nodos internos o programas sometidos a un proceso evolutivo. Los nodos internos de la red serán creados mediante técnicas de programación genética y todos lo operadores genéticos pueden ser aplicados a los nodos de la red.

En la figura 2.7 se muestra un esquema de una red programada genéticamente, los circulos representan entrdas sensoriales y las salidas a los actuadores del agente, mientras que los cuadros representan programas creados con técnicas de programación genética. En el esquema podemos ver la variedad de conexiones que se pueden generar, para este proyecto en especifico no existe ninguna restricción sobre el tipo de conexiones que s e puede generar.

Este proceso evolutivo genera expresiones sintácticas válidas en cada uno de los nodos que conforman la red, la configuración de estas expresiones definen las conexiones de

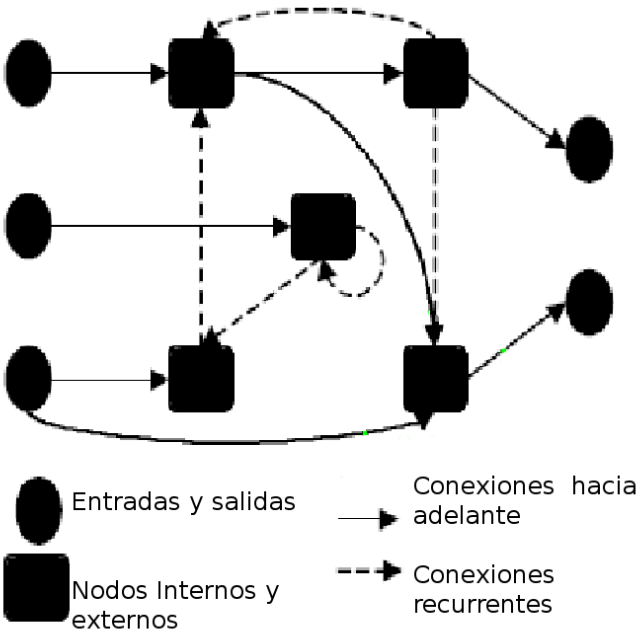


Figura 2.7: Redes Programadas Genéticamente

la red y por lo tanto su arquitectura.

2.4.1. Nodos y conexiones

En una Red Programada Genéticamente los nodos de la red constituyen una expresión sintáctica que podrá obtenerse producto de un recorrido "*in orden*" en el árbol, por lo tanto las conexiones de la red están definidas por los operandos definidos en estas expresiones sintácticas.

La arquitectura de la red está definida en función de las entradas o requerimientos que tienen los programas (nodos), es decir si el nodo i necesita como entrada la salida del nodo j entonces se realiza una conexión del nodo j al nodo i.

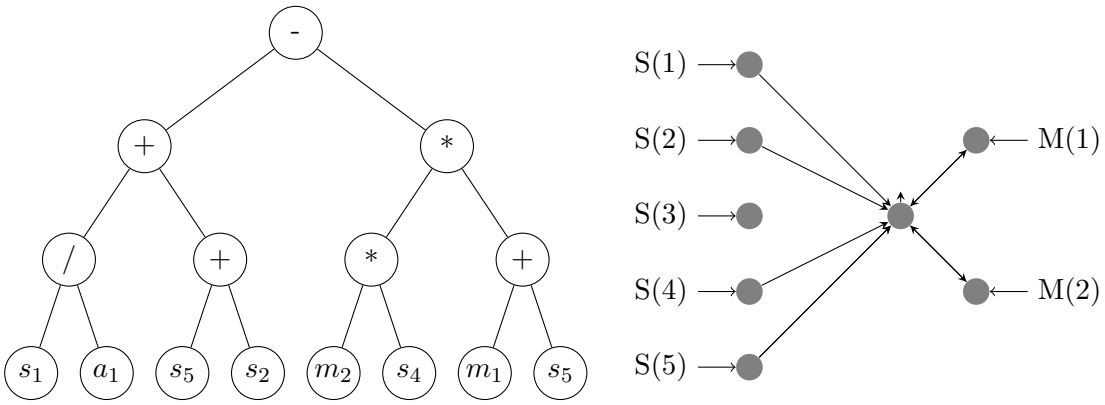


Figura 2.8: Ejemplo de una red de un nodo y la configuración de conexiones generada por su árbol de sintáxis

En la figura 2.8 se muestra una red con un solo nodo y el árbol de sintáxis de este programa, en esta se ilustra como es que las conexiones de la red están definidas por los elementos en las hojas del árbol.

Esta reconfiguración de las conexiones de la red representan la principal característica de estas redes debido a que podemos evolucionar en algún grado la arquitectura de la red. Por lo tanto el éxito de los individuos resultantes después del proceso evolutivo estará estrechamente realcionado con la arquitectura de red que produjo el proceso evolutivo y cobrará importancia el tipo de conexiones que estén presentes en las arquitecturas de los individuos más exitosos.

Capítulo 3

Planteamiento del problema

El objetivo de este proyecto es desarrollar un controlador para un agente autónomo, para lo cual proponemos una solución basada en técnicas de evolución artificial mediante el uso de redes programadas genéticamente.

Para evaluar la eficiencia de las redes programadas genéticamente utilizamos el problema clásico de la evasión de obstáculos en un ambiente no dinámico, cabe sealar que este método no ha sido reportado en la literatura como una solución a este problema.

Para iniciar el proceso evolutivo necesitamos definir una arquitectura inicial (figura 3.1), en esta arquitectura las entradas de la red serán los sonares del agente. La red inicial sólo nos proporciona una arquitectura válida para el problema que definimos, todas las conexiones serán inicializadas aleatoriamente así que el diseñador no propone ninguna conexión en la red. Los actuadores del agente están codificados en una capa de salida que cuenta con dos nodos que proporcionarán los comandos motrices a los dos motores del robot, para completar una arquitectura válida es necesario que definamos una capa oculta inicialmente cuenta con cuatro nodos.

El proceso evolutivo se llevará a cabo sobre los seis nodos de la red por lo tanto el conjunto de símbolos *Terminales* está constituido por las entradas sensoriales ($S_i(t)$), las salidas de los nodos de la capa oculta ($a_i(t)$) y las salidas de los nodos en la capa de salida($M_i(t)$). El conjunto de símbolos *Terminales* que hemos definido permiten que se pueda establecer cualquier tipo de conexión, dando gran variedad a las arquitecturas que se generan.

Para el conjunto de los símbolos *No Terminales* utilizamos los operadores aritmeticos de suma y resta, esto se debe a que buscamos que se realize una correlación con el

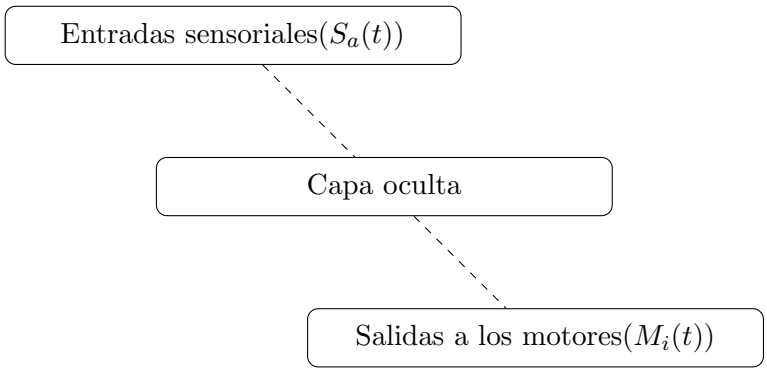


Figura 3.1: Red inicial para el proceso evolutivo, la capa de salida tiene dos nodos para todas las pruebas mientras que la capa oculta inicio con cuatro nodos en las primeras pruebas y luego se tomo solo un nodo para las restantes.

modelo de red neuronal artificial tradicinal. Usando solo de sumas y restas podemos emular el comportamiento de una red neuronal que usa pesos sinapticos con valor igual a 1 y una función de activación lineal.

Los ambientes utilizados para estos experimentos son ambientes con una configuración de obstáculos fija dentro de una arena cuadrada, los obstáculos son de tamaños diferentes y se mantendrán fijos durante el tiempo de vida del individuo.

El agente que hemos elegido para el proceso de evolución es una versión basada en el robot con el que cuenta el departamento de computación de la Facultad de Ciencias (Pionner P3-XD). Este robot cuenta con un arreglo de 8 sensores dispuestos al frente del robot en un rango de 180° como se muestra en la figura 3.2, estos sensores tienen un alcance de cinco metros que están expresados en valores enteros entre 0 y $5,000$.

Para el movimiento el robot cuenta con dos motores dispuestos a los lados del robot que funcionarán como actuadores de la red, además de una rueda loca para brindarle equilibrio a su movimiento. La tarea asignada al agente será la de sobrevivir en el ambiente que se le ha designado durante un tiempo de vida previamente definido, una colisión con alguno de los obstáculos es interpretada como un fallo a la tarea designada y por lo tanto el robot terminara su ejecución.

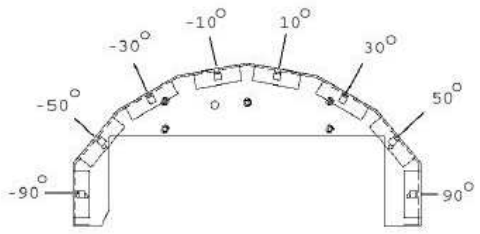


Figura 3.2: Esquema de la disposición de los sonares en el frente del agente que se utilizó

3.1. Metodología

El sistema que maneja el proceso evolutivo fue desarrollado en C++ debido principalmente a que la librería que crea la interfaz para comunicarse con el simulador también está implementada en este lenguaje, la librería Aria funciona como un software cliente-servidor para la comunicación eficiente entre el robot y el sistema que deseamos implementar en el robot.

Inicialización

El proceso evolutivo comienza creando n individuos para la primera generación, la creación de cada uno de los individuos consiste en una inicialización aleatoria de cada uno de los programas que componen la red programada genéticamente utilizando el método FULL para crear árboles con una profundidad de 5 niveles.

Evaluación

A continuación cada uno de los individuos de la generación 0 será ejecutado en el simulador durante un número de pasos (tiempo de vida). Para aumentar el carácter aleatorio del proceso cada individuo realizo un giro aleatorio al inicio de su tiempo de vida, lo cual es suficiente para evitar que se genere un comportamiento para una misma posición del ambiente. Durante su ejecución el agente será sometido a un proceso de evaluación para

el cual se utilizó una función de aptitud, que definimos en base a otros trabajos reportados en la literatura.

Selección

Al finalizar el proceso de evaluación de los individuos de una generación se usará un método de selección por ruleta, el cual asegura que éstos tendrán una probabilidad más alta de pasar a la siguiente generación. Favoreciendo aquellos individuos que tengan valores más altos durante el proceso de evaluación aseguramos que sus genes mejoren el desempeño de los individuos de la siguiente generación.

Proceso evolutivo

Una vez definida la siguiente generación mediante el proceso de selección estos individuos serán sometidos a dos operadores genéticos, primero un cruce de sub-árboles y posteriormente a una mutación de sub-árboles. El cruce de sub-árboles se realiza a todos los individuos de la generación, en particular el método aplica el operador de cruce a pares de nodos en las redes de dos individuos.

La mutación es un operador que realiza cambios que alteran significativamente la estructura del nodo y estos cambios fácilmente pueden alterar el comportamiento del individuo, por esta razón para el operador de cruce tenemos una probabilidad de 5 % basándonos en lo reportado por la literatura.

Después de este proceso completo se constituye una nueva generación que será de nuevo sometida al proceso de evaluación y éste será repetido durante 50 generaciones. El sistema que se diseñó guarda en archivos de texto todos los individuos de cada una de las generaciones así como los valores de la función de aptitud que le corresponden, además calculamos el promedio de la función de aptitud por generación para observar su desarrollo a través del tiempo.

Pruebas

Una vez que terminó el proceso evolutivo, evaluamos el comportamiento de la función aptitud para buscar a los mejores individuos para realizar la fase de pruebas. Para

la fase de pruebas se realizó un sistema adicional que leyera de los archivos los individuos de una generación y que inicializará el simulador con cualquier mapa, de este modo podemos probar cualquier individuo de cualquier generación en cualquier mapa.

Durante la fase de pruebas evaluamos la eficiencia de los controladores en función del comportamiento que presentaron durante su ejecución y no por el valor de su función de aptitud, tomamos la generación con el máximo promedio y probamos sus individuos en ambientes diferentes a los que utilizaron para el proceso de evolución.

3.2. Función de aptitud

En cualquier método evolutivo la parte medular del proceso de optimización depende de la elección que el diseñador haga sobre la función de aptitud, esta función deberá codificar la tarea que se busca que los agentes desarrollen después de varias generaciones.

La función de aptitud elegida para este proyecto fue tomada de las descripciones en la literatura [24], esta es una de las funciones clásicas que describen un comportamiento deambulatorio para un agente. Ésta presentó muy buenos resultados en la resolución del problema de evasión de obstáculos, la función de aptitud es la siguiente:

$$F(t) = \alpha\|M_1(t) + M_2(t)\| - \beta\|M_1(t) - M_2(t)\|$$

- t Ciclo de ejecución en el que se encuentra el individuo
- $M_1(t)$ Valor del primer motor al ciclo t
- $M_1(t)$ Valor de segundo motor al ciclo t
- α, β Constantes de ponderación

La suma de los motores premia las trayectorias en las que el agente avanza en línea recta y la diferencia castiga las trayectorias con giros, esta función resultó en una gran variedad de comportamientos. En el sistema que se diseñó para el proceso evolutivo decidimos definir un máximo de 500 mm/s para los valores provenientes de los nodos en la capa de salida, por lo tanto el valor máximo de la función por cada uno de los pasos será de $1,000$, este valor en la función de aptitud significa las dos ruedas a avanzando a 500 mm/s en línea recta.

Cuadro 3.1: Tabla de parámetros del proceso evolutivo usando programación genéticamente

Población	30
Profundidad de los nodos	5
Generaciones	50
Tiempo de vida	250,500
Selección	Ruleta
Probabilidad Cruce	100
Probabilidad Mutación	5
Símbolos No Terminales	+,-
Símbolos Terminales	$S_i(t)$,$a_i(t)$,$M_i(t)$

Para esta función el máximo de cada uno de los ciclos que realizaba el proceso de evaluación del agente tiene un valor de $1,000$ y el máximo de la función será la multiplicaión de este valor por el número de ciclos en el tiempo de vida del agente.

Sin embargo tenemos que hacer notar que el valor máximo de la función no puede ser alcanzado para estas pruebas, un agente que obtenga el valor máximo de la función de aptitud durante su tiempo de vida tendria que tener un comportamiento de desplazamiento en linea recta con los dos motores al máximo de la velocidad durante su tiempo de vida, no es posible desarrollar este comportamiento bajo las condiciones de estas pruebas.

Así que tenemos que decir que en realidad el propósito principal de diseñar una función de aptitud es proveer al proceso evolutivo de una medida para realizar la selección de los individuos mas aptos, mientras que el éxito de las pruebas tendrá que evaluarse en función de los comportamientos resultantes y si estos son capaces de resolver la tarea que se les asigno.

3.3. Parámetros GP

Dentro de la programación genética existen un conjunto de parámetros que definen en gran medida los comportamientos que presentaran los individuos, como parte del proceso de diseño es importante definirlos para la primera prueba, sin embargo éstos pueden variar con el avance del proceso de experimentación.

El tamaño de la población, y el tiempo de vida fueron elegidos mediante algunas pruebas hechas con anterioridad y principalmente basándonos en los valores reportados en la literatura.

El principal motivo para elegir que los árboles tuvieran una profundidad de 5 niveles es que el número de hojas es el mas cercano al número de elementos del conjunto de los símbolos *Terminales* y necesitábamos que pudieran ser elegidos todos los elementos de este conjunto para no descartar la existencia de una red completamente conectada, además los reportes en la literatura presentaban profundidades similares.

Para el tiempo de vida se tomaron 250 y 500 ciclos que resultan en tiempos de vida entre 3 y 5 minutos, con esto la prueba del agente es lo suficientemente extensa y puede enfrentar obstáculos durante ella. Así podíamos tener una medida real de la eficiencia de los controladores para evitar colisiones, el valor de 500 ciclos para el tiempo de vida prolongaba el tiempo de las pruebas para 50 generaciones alrededor de seis ó siete días, motivo por el cual sólo una de las pruebas se realizó con el valor de 500 ciclos y las restantes con 250.

La razón por la cual elegimos una población inicial de 30 individuos es que con poblaciones menores la diversidad se reduce drasticamente y el proceso evolutivo convergía a máximos locales donde no se alcanzaban buenos controladores.

En el conjunto de los símbolos terminales decidimos contar con las estradas sensoriales ($S_i(t)$), las salidas de los nodos de la capa oculta ($a_i(t)$) y las salidas de los nodos en la capa de salida ($M_i(t)$). Esta configuración para el conjunto de símbolos terminales obedece a que no queríamos tener ninguna restricción en el tipo de conexiones que se obtuvieran durante el proceso evolutivo.

La elección de los símbolos *No Terminales* obedece principalmente a dos razones en específico, la primera de ellas tiene que ver con el marco teórico de la robótica cognitiva en el que está situado el proyecto. El uso de sumas y restas nos permite emular el comportamiento de una red neuronal con una función de activación lineal.

La segunda radica en el problema mismo, los datos provenientes de los sensores están definidos con números enteros desde 0 hasta $5,000$ y para poder realizar pruebas con algún otro tipo de operadores necesitamos cambiar la arquitectura básica de la red. Por ejemplo para manejar operaciones lógicas necesitamos que la red trabaje con números binarios lo que nos obligaría a cambiar la red por una que trabaje con datos binarios.

Estos cambios modifican la definición del problema por uno que maneje movimientos en un espacio discreto y responda con movimientos predeterminados como avanzar, girar a la derecha o girar a la izquierda, así que el poder emular redes neuronales nos da una ventaja para evaluar su comportamiento y abordar el problema de una forma contínua.

Por otro lado el manejar operadores aritméticos como la multiplicación y la división

nos obliga a un rango de datos mayor, debido a observaciones en los comportamientos de los agentes decidimos acotar la máxima velocidad a 500 mm/s para tener control de una velocidad máxima. Notamos que en los valores de salida cercanos a 10 mm/s para los motores del agente, este permanece prácticamente sin movimiento y por el contrario el comportamiento del robot con valores mayores a 500 mm/s se vuelve erratico. Por lo tanto los operadores de multiplicación y división nos llevarían a obtener datos de salida que podrían no ser significativos en términos de los comportamientos que deseábamos para los agentes.

Capítulo 4

Experimentos

Los experimentos aqui mostrados fueron diseñados para verificar si la arquitectura inicial propuesta de seis nodos ya definida era capaz de producir controladores eficientes, para la primera prueba se realizó con valor de 1 para los parámetros α y β de la función de aptitud.

La función utilizada es:

$$F_1(t) = \|M_1(t) + M_2(t)\| - \|M_1(t) - M_2(t)\|$$

Durante el desarrollo de esta prueba fue posible desarrollar comportamientos deambulatorios, sin embargo, en un fenómeno que no esperábamos los comportamientos que predominaban eran giratorios lo cual resultaba extraño debido a que la función estaba diseñada para castigarlos.

Después de un análisis a las pruebas realizadas pudimos darnos cuenta que los comportamientos giratorios en los agentes tenían una característica en común, todos ellos sobrevivían durante mas tiempo de la prueba en comparación con aquellos que presentaban un comportamiento deambulatorio con pocos giros.

En realidad ambos comportamientos obtenían una ganancia con cada uno de los ciclos y aunque los comportamientos giratorios tenían una ganancia menor por ciclo, esto se compensaba o incluso superaba a los démas comportamientos debido a que permanecían vivos por un mayor tiempo.

Debido a este fenómeno decidimos realizar la misma prueba con una variante en la función de aptitud, para esta segunda prueba decidimos tomar un valor de 2 para el

Cuadro 4.1: Resultados obtenidos del conjunto de experimentos.

Arquitectura	F(t)	Mejor	Promedio	Peor	Máximo
6 nodos	$F_1(t)$	$107,106$	$69,614$	$1,900$	$125,000$
	$F_2(t)$	$61,660$	$18,257$	0	$125,000$
3 nodos	$F_1(t)$	$208,690$	$75,887$	$8,744$	$250,000$
	$F_2(t)$	$74,506$	$38,018$	$74,506$	$125,000$

parámetro β lo cual duplicaba el castigo a los comportamientos giratorios.

La nueva función definida es:

$$F_2(t) = \|M_1(t) + M_2(t)\| - 2\,\|M_1(t) - M_2(t)\|$$

La razón para realizar esta segunda prueba era verificar si realmente estos comportamientos giratorios eran producto de este fenómeno observado o si en realidad el modelo no era capaz de resolver el problema de una forma eficiente. En contraste con la función anterior los comportamientos que se generaron fueron comportamientos deambulatorios que tenían pocos giros y pudimos encontrar comportamientos no esperados como giros de 90°.

Una vez que obtuvimos resultados con esta arquitectura inicial de 6 nodos en la red, ahora queríamos saber si una arquitectura de menor tamaño podía realizar la tarea. En la segunda fase de experimentos que definimos contaba con una nueva arquitectura que ahora cuenta sólo con un nodo en la capa oculta más los dos nodos de la capa de salida, esta arquitectura es la mínima posible definida por nuestro problema, ya que otro tipo de arquitecturas restringen el tipo de conexiones posibles en la red.

En total se realizaron cuatro experimentos para probar las dos arquitecturas con las dos funciones de aptitud, estos cuatro experimentos se elaboraron durante 50 generaciones, cabe señalar que en 50 generaciones ya se pudieron encontrar controladores eficientes.

Para cada uno de los experimentos se obtuvo el promedio del valor de la la función de aptitud de los individuos de la generación, además se obtuvo el valor de la función para los individuos con el máximo y mínimo, en la tabla 4.1 se ilustra los valores de estos individuos en conjunto con el promedio de toda la generación y el máximo posible de la prueba.

4.1. Arquitectura de 6 nodos

En el primero de los experimentos fue utilizada una arquitectura de seis nodos en la red, cuatro de ellos en la capa oculta más los dos nodos en la capa de salida que calculan los comandos motrices. Durante las primeras generaciones observamos que algunos de los agentes respondían ante la presencia de los obstáculos y alrededor de la décima generación, empezamos a observar los primeros comportamientos de evasión de obstáculos.

Utilizando la primera función de aptitud $F_1(t)$ el máximo posible de la función es de $125,000$, el mínimo para un individuo es de $1,900$ mientras que el máximo fue de $107,106$ que representa el $85\,\%$ del máximo posible, mientras que el promedio de la generación tiene un valor de $69,614$ que corresponde a un $55\,\%$ y se alcanzo en la generación 50. Los mejores individuos de la generación eran capaces de deambular en diferentes ambientes por tiempos mayores a los tiempos de vida utilizados para el proceso evolutivo.

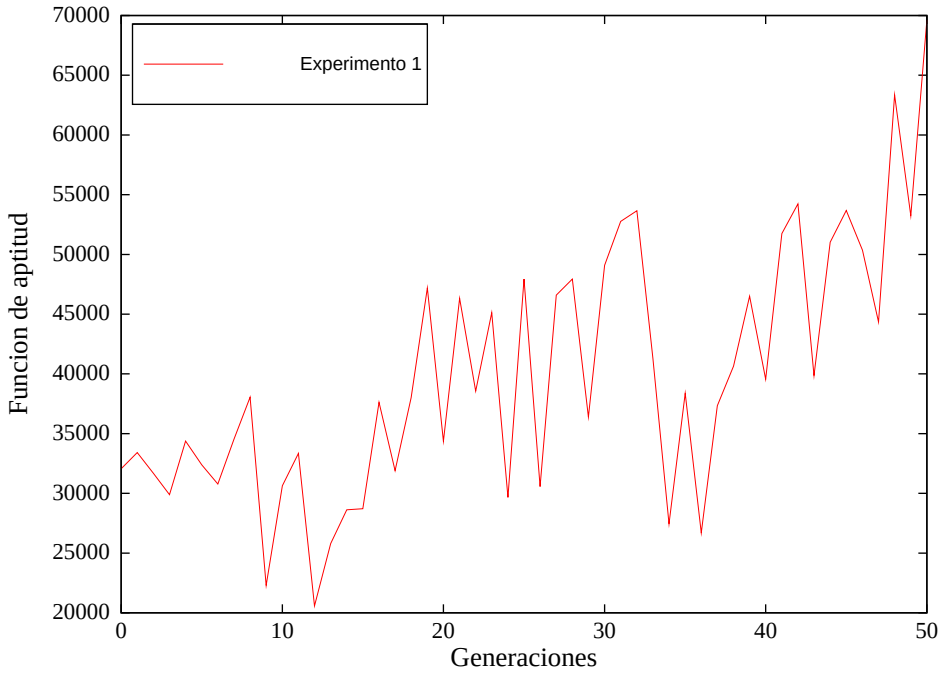


Figura 4.1: Gráfica del promedio de los valores de aptitud obtenidos por los individuos en la prueba con 6 nodos y función de aptitud $F_1(t)$

En el segundo de los experimentos implementamos la función de aptitud $F_2(t)$ previamente definida, todos los parámetros del proceso evolutivo y las condiciones del ambiente

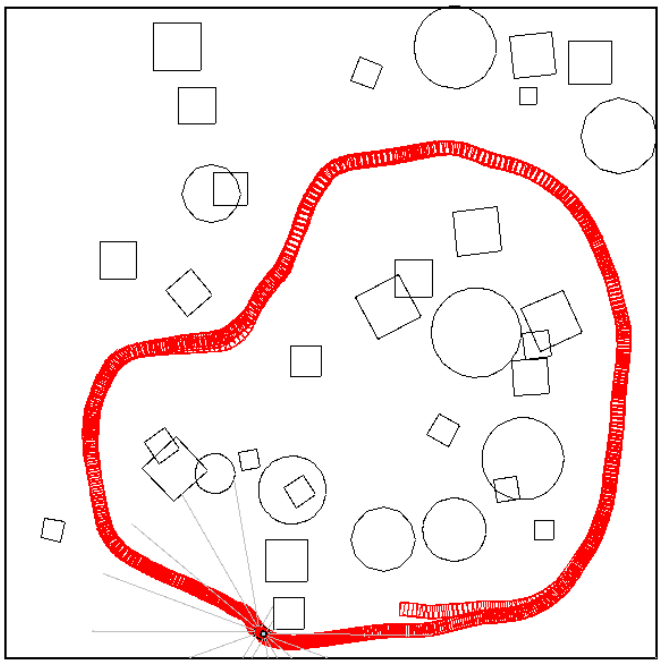


Figura 4.2: Trayectoria de un individuo para el experimento con 6 nodos y la función de aptitud $F_1(t)$.

se mantuvieron iguales al experimento anterior.

Así mismo los comportamientos de reacción ante los obstáculos del ambiente se presentaron de manera inmediata en las primeras generaciones y los primeros controladores exitosos se presentaron alrededor de la generación número 20 siendo la generación 39 la que obtuvo el valor máximo de la función.

Durante el proceso evolutivo pudimos darnos cuenta que eran más exitosos los comportamientos con poco tiempo de vida y trayectorias con pocos giros en comparación con trayectorias con tiempos de vida mayor pero comportamientos giratorios predominantes.

Al final los comportamientos de poco tiempo de vida fueron substituidos por aquellos comportamientos que eran capaces de deambular por más tiempo en el ambiente presentando pocos giros, al grado de que se presentaron algunos comportamientos no esperados como giros de 90°.

El valor máximo que puede alcanzar la función es de $125,000$ y la generación 39 presentó un promedio de $18,257$ que equivale al $14\,\%$, el mejor individuo tenia $61,660$ que corresponde al $49\,\%$ del valor máximo posible en la función de aptitud. Los valores de

aptitud obtenidos resultarón mas bajos de lo esperado sin embargo los comportamientos resultantes parecían comportarse de mejor manera, en la función de aptitud $F_2(t)$ el castigo sobre los giros es del doble en comparación con la función anterior, este factor disminuye drasticamente el valor de la función de aptitud así como el promedio. Algunos comportamientos giratorios representan valores de 0 para esta función, sin embargo cabe señalar que a la hora de examinar los comportamientos de este experimento en la fase de pruebas arrojo controladores exitosos en el cumplimiento de la tarea que se les asigno.

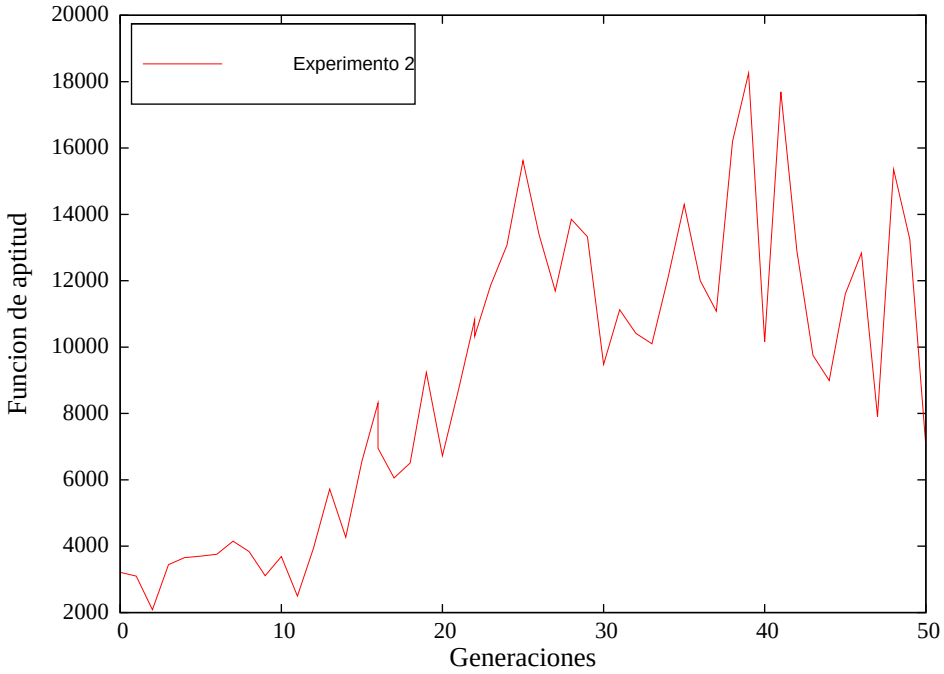


Figura 4.3: Gráfica del promedio de los valores obtenidos por los individuos en la prueba para el experimento con 6 nodos y función de aptitud $F_2(t)$

4.2. Arquitectura de 3 nodos

Una vez que se obtuvieron los resultados de las primeras pruebas decidimos disminuir el tamaño de la arquitectura a una con 3 nodos con la idea de que sería la arquitectura mínima para que resolviera el problema, ésta arquitectura fue probada con las dos funciones de aptitud que ya habíamos definido. Al igual que los controladores para las arquitecturas de 6 nodos, en estas pruebas observamos que los controladores respondían ante los obstáculos

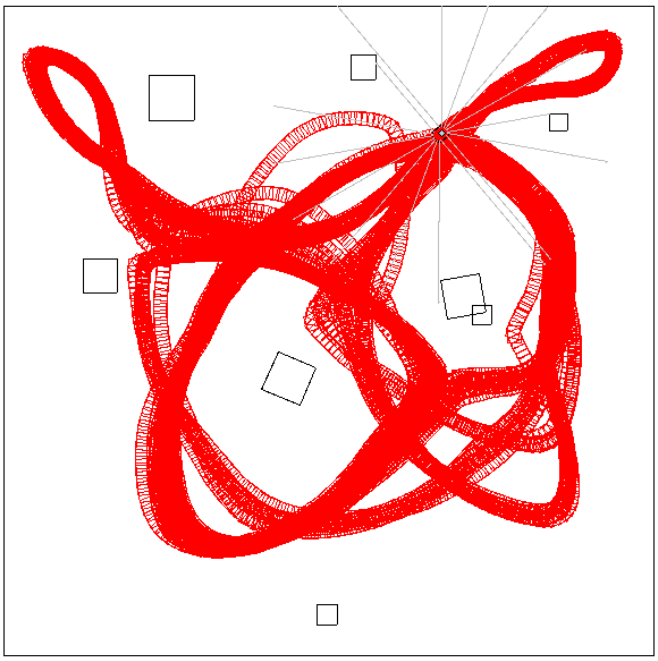


Figura 4.4: Trayectoria de un individuo con arquitectura del experimento con 6 nodos y función de aptitud $F_2(t)$

desde las primeras generaciones, de nuevo el máximo de la función siempre estuvo dentro de las primeras cincuenta generaciones.

El comportamiento de los controladores como el de las funciones de aptitud fue similar al de las pruebas que se habían hecho con una arquitectura de 6 nodos. Al final en las pruebas que realizamos los controladores generados con esta arquitectura también generaron comportamientos que resolvían exitosamente la tarea de evasión de osbstáculos al igual que los agentes con arquitectura de 6 nodos.

La primera de las pruebas fue realizada utilizando $F_1(t)$ pero a diferencia de las demás pruebas decidimos incrementar el tiempo de vida al doble (500 ciclos) para poder verificar que el tiempo de vida no fuera una factor decisivo en los comportamientos obtenidos por el proceso evolutivo.

Sin embargo, los comportamientos generados por los mejores individuos completaron de manera satisfactoria las pruebas en diferentes mapas a los que se les realizó el proceso evolutivo, el máximo en esta prueba se encontró en la generación 49.

La ultima de las pruebas utilizo la arquitectura de 3 nodos y la función de aptitud

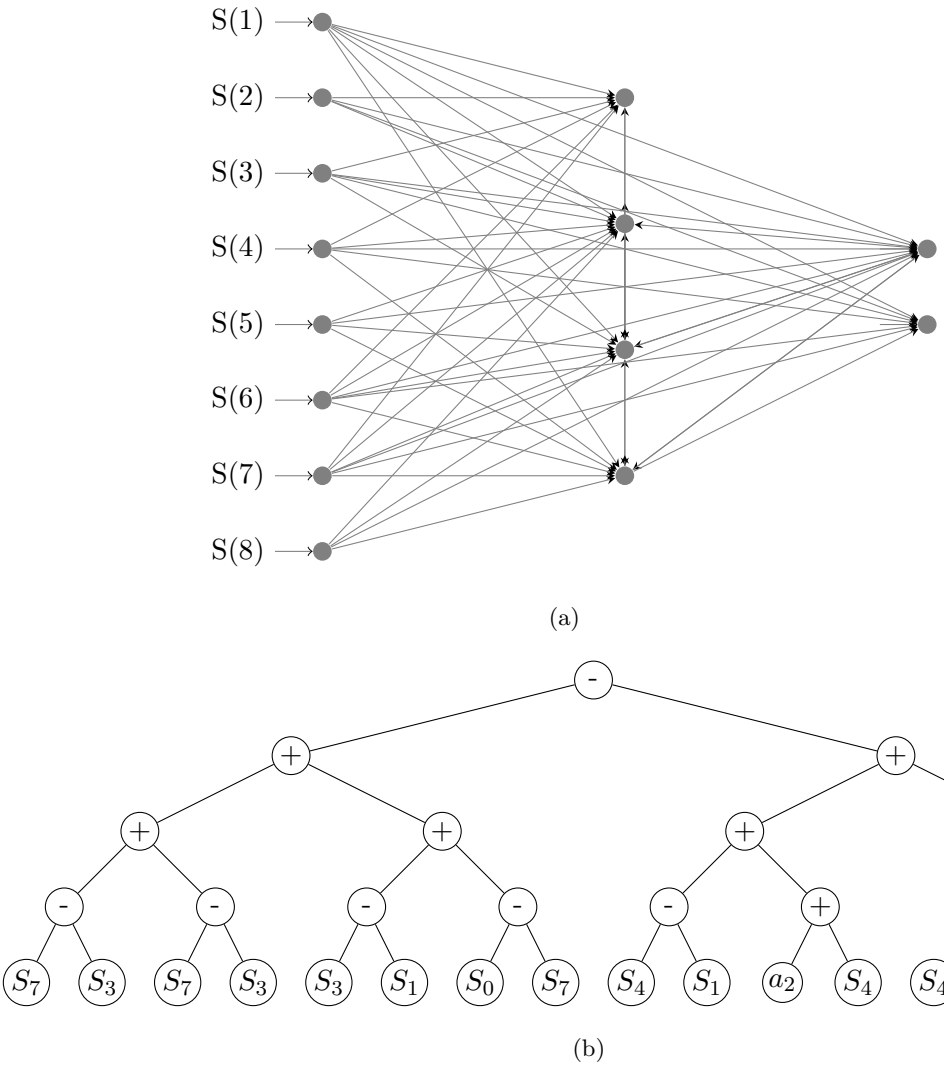


Figura 4.5: (a) Arquitectura del controlador de uno de los individuos del experimento realizado con una arquitectura de 6 nodos (b) Árbol de sintáxis para uno de los nodos de la red

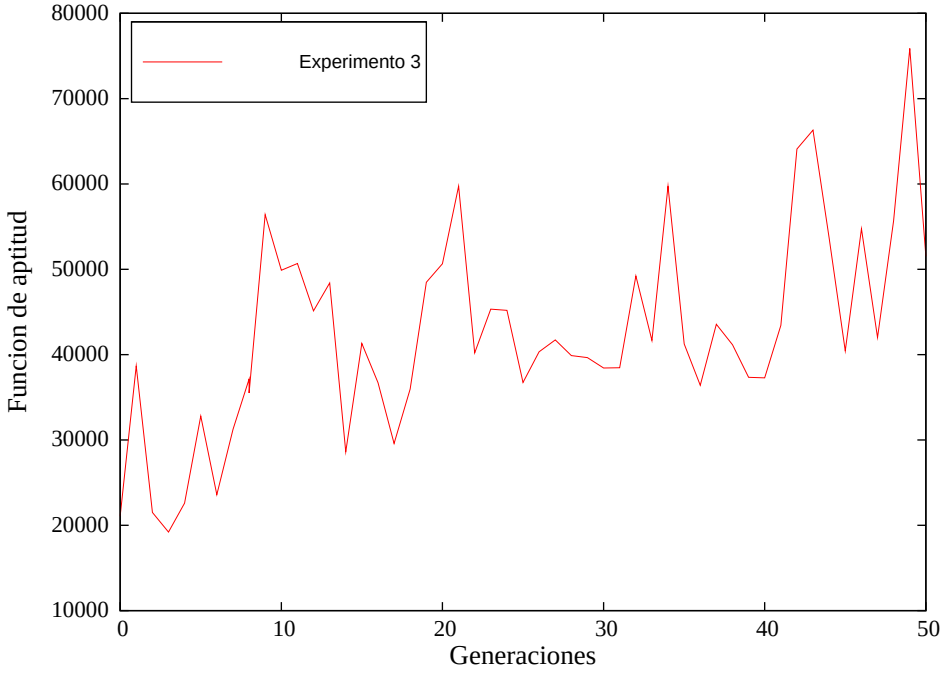


Figura 4.6: Gráfica del promedio de los valores de aptitud obtenidos por los individuos en la prueba con una arquitectura de 3 nodos y utilizando la función de aptitud $F_1(t)$

$F_2(t)$, esta prueba presentó los mejores resultados en el conjunto de comportamientos producidos por el proceso evolutivo, el máximo de la función se alcanzó en la generación 44 y el mejor individuo tuvo un valor de $74,506$ que corresponde $59\,\%$ mientras que el individuo con valor mínimo fue de $41,89$ que corresponde al $3\,\%$, el promedio que alcanzó la función fue de $38,018$ siendo un $30\,\%$ del total de la función.

Es importante destacar que cuando analizamos los datos de la función de aptitud no coincidían con nuestra percepción sobre el éxito de esta prueba, de hecho los datos arrojaban que en realidad la primera de las pruebas que corresponde a una estructura de 6 nodos utilizando $F_1(t)$ era la mejor del conjunto completo de pruebas.

Por lo cual decidimos realizar las pruebas a los individuos de esta generación en ambientes más complicados, creamos un ambiente con una configuración diferente en forma de un laberinto, en esta prueba tomamos el mejor de todos los agentes de cada generación y lo ejecutamos en él.

Sin que los agentes estuvieran entrenados para este tipo de ambiente fueron capaces de deambular en él, sin embargo como se puede ver en la figura 4.11 el único que completo

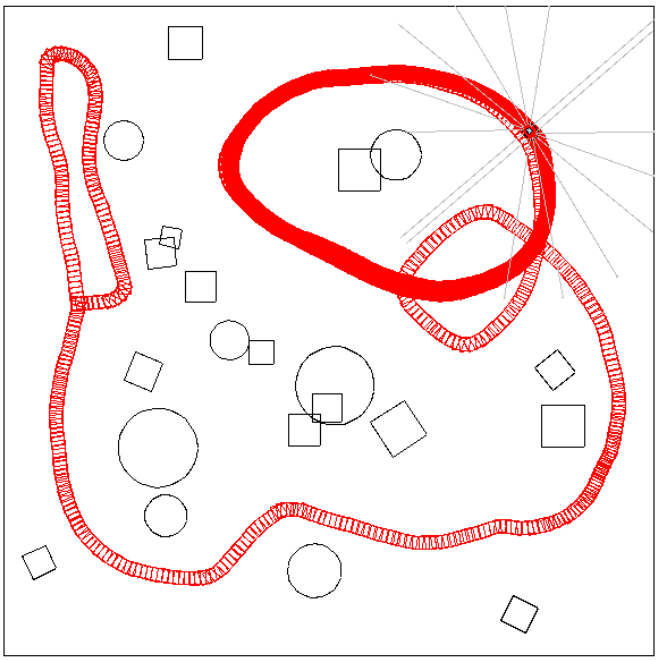


Figura 4.7: Trayectoria de un individuo con arquitectura de 3 nodos y utilizando la función de aptitud $F_1(t)$

todo el recorrido fue el que corresponde a una arquitectura de tres nodos con $F_2(t)$. Con esto no hablamos de un comportamiento emergente para resolver tareas más complicadas como llegar de un punto a otro, lo que sí podemos concluir es que cuando se enfrentó a un ambiente con una configuración diferente esta arquitectura fue la que produjo un comportamiento mas robusto ante diferentes configuraciones en el tipo de ambiente al que se enfrentaba.

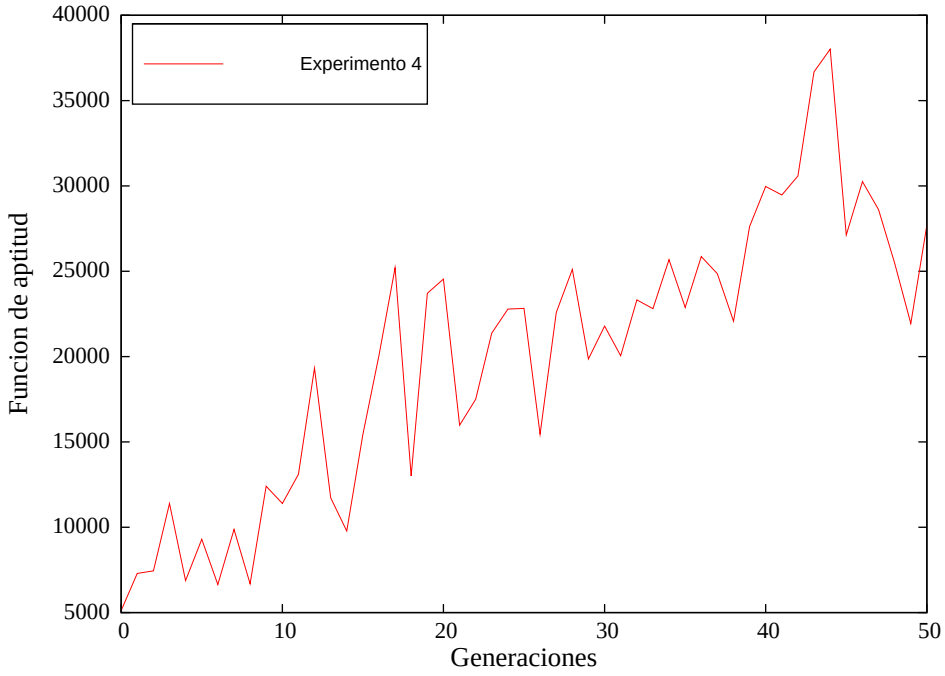


Figura 4.8: Gráfica del promedio de los valores de aptitud obtenidos por los individuos en la prueba que cuenta con una estructura de 3 nodos y $F_2(t)$ como función de aptitud.

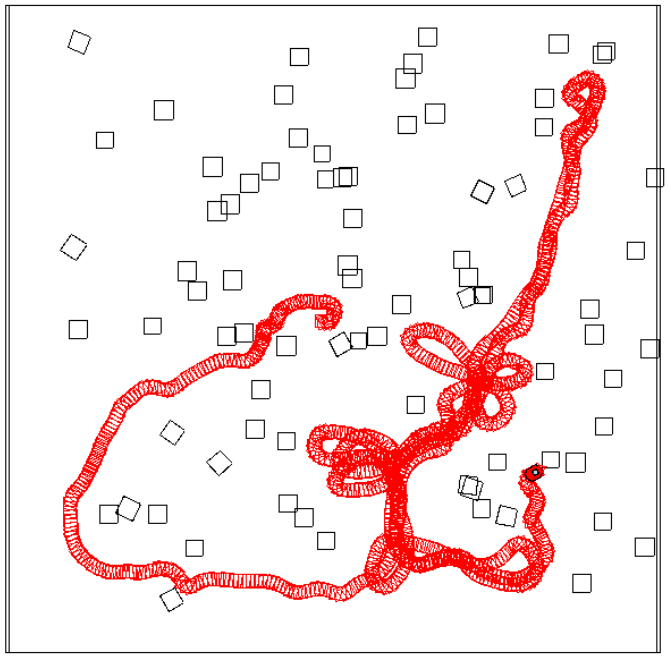
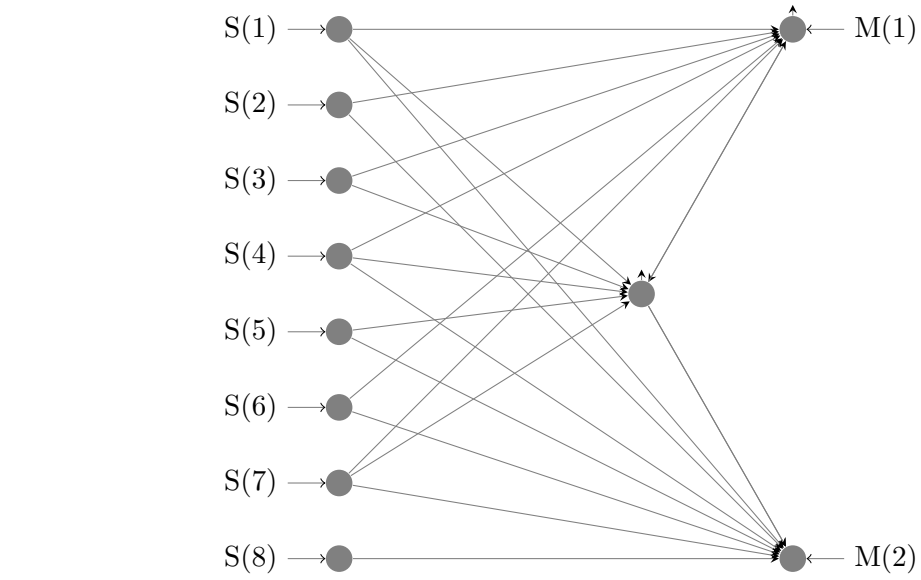
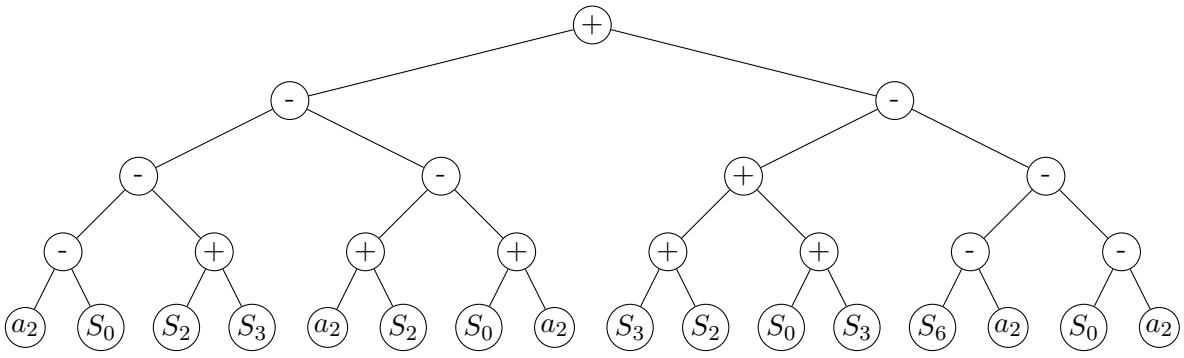


Figura 4.9: Trayectoria de un individuo de la prueba con una arquitectura de 3 nodos y función de aptitud $F_2(t)$



(a)



(b)

Figura 4.10: (a)Arquitectura del controlador de uno de los individuos del experimento con 3 nodos y la función de aptitud $F_2(t)$ (b) Árbol de sintáxis de uno de los nodos de esta red

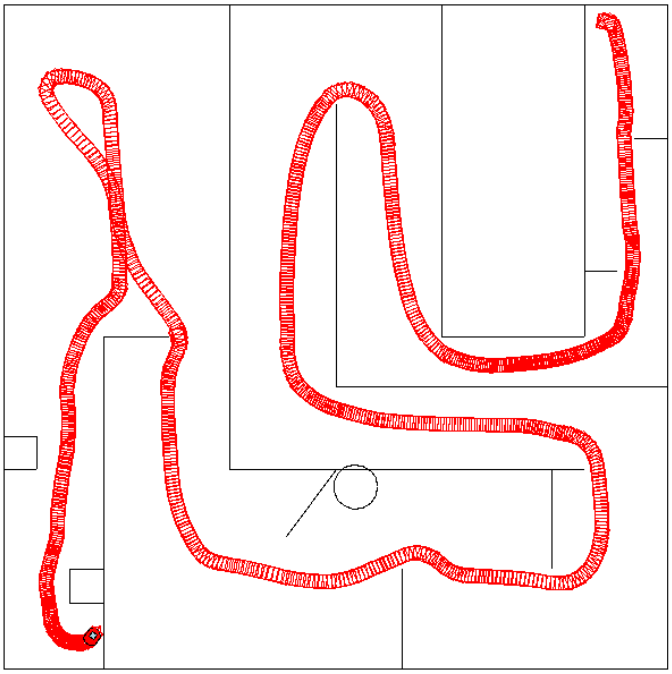


Figura 4.11: Evaluación de uno de los controladores exitosos del experimento con 3 nodos y $F_2(x)$ en un ambiente con una configuración distinta a los ambientes de entrenamiento y prueba.

Capítulo 5

Conclusiones

El principal propósito del proyecto era evaluar el comportamiento de las redes programadas genéticamente ante un problema en el campo de la robótica que pudiera resultar más complicado de los que se habían reportado en la literatura, en especial verificar su desempeño en un problema clásico de evasión de obstáculos en un ambiente continuo.

Los resultados que pudimos obtener en el marco de los experimentos generaron comportamientos de evasión eficientes para cada una de las pruebas, estos se presentaron en pocas generaciones en comparación con otras soluciones que se han propuesto para este problema tan estudiado en la literatura.

En especial, resulta interesante notar que los controladores obtenidos observaban una aparente reacción a los obstáculos desde las primeras generaciones y se necesitaron pocas generaciones para observar controladores que resolvieran el problema en una forma eficiente.

En la última prueba decidimos tomar a los mejores individuos que encontramos y probarlo en un ambiente con una configuración distinta, para lo cual utilizamos un ambiente en forma de un laberinto que representa una configuración diferente de obstáculos en comparación con los usados en el proceso evolutivo. A pesar de que los individuos no fueron evolucionados para recorrerlo por completo, su controlador fue capaz de llevarlos a través del ambiente y uno de ellos fue capaz de recorrerlo por completo, demostrando que puede ser también usado para problemas con diferentes metas dentro del campo de la robótica evolutiva.

De especial interés para un trabajo a futuro es el estudio de la eficiencia de las funciones de aptitud que hemos definido, debido a que muchas veces los individuos generan

comportamientos que optimizan la función sin embargo no producen los comportamientos que buscamos.

Nos hemos encontrado con casos en los que el mejor de la generación tiene comportamientos que no resultan tan eficientes como los que presentaron individuos con valores de aptitud más bajos, estos casos disminuyeron con el uso de $F_2(t)$, sin embargo, aún en esta prueba pudimos encontrar casos similares.

Por otra parte creemos que es importante estudiar la forma de las soluciones generadas, la variedad de conexiones que hemos podido desarrollar y el tipo de arquitecturas tienen una característica en común, ninguna de las redes de estos individuos estaba completamente conectada, a pesar de que todos los elementos del conjunto de los *Terminales* están presentes en la red, de hecho ni siquiera existe un nodo que tome todos los $S_i(t)$.

También pudimos notar que los sensores dispuestos al frente del agente fueron los más recurrentes en los individuos e incluso estos se repiten varias veces en algunos de los nodos, esta característica nos puede indicar una zona en la que la información sensorial es mas importante en comparación con otras zonas.

En los individuos más exitosos de las pruebas se encontraron conexiones de la capa de salida a la capa oculta que representan conexiones recurrentes, además de conexiones con nodos de la misma capa e incluso conexiones en ciclo de un nodo a si mismo.

Según lo reportado en la literatura este tipo de conexiones representan un modelo de memoria en las redes neuronales artificiales, por lo que podemos decir que el proceso evolutivo llevó a los agentes a desarrollar algún tipo de modelo de memoria para resolver el problema de la evasión de obstáculos. Como trabajo a futuro resulta interesante evaluar la importancia de estas conexiones para el desarrollo de controladores y cual es su aportación real en la eficiencia obtenida de controladores que resuelven el problema de la evasión de obstáculos.

La variedad de comportamientos que hemos obtenido mediante este proceso de evolución artificial fue muy amplia por lo que es importante señalar la importancia en el buen desarrollo de la función de aptitud. Hemos encontrado muchos comportamientos que a la vista de los diseñadores resultaban comportamientos malos sin embargo, éstos optimizaban la función de aptitud que les hemos proporcionado.

Cabe destacar que con los resultados obtenidos creemos que estas redes pueden ser aplicadas a problemas más complejos dentro del campo de la robótica cognitiva. Como parte

de un trabajo a futuro ya se tienen varios esquemas de evolución incremental tomando como base los controladores de los mejores individuos para resolver problemas más complejos.

En especial uno de estos esquemas de evolución incremental puede ser implementado para poder desarrollar una arquitectura que sea capaz de inferir el estado sensorial siguiente del robot utilizando sus propios comandos motrices y las entradas sensoriales.

Bibliografía

[1] R. A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, 1990.

[2] R. A. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[3] J. R. Searle, "Minds, brains and programs," 1980.

[4] R. Pfeifer and C. Scheier, *Understanding Intelligence*. Cambridge, MA, USA: MIT Press, 2001. Illustrator-Follath, Isabelle.

[5] T. Ziemke, "Rethinking grounding," in *In*, pp. 177–190, Plenum Press, 1999.

[6] J. Urzelai and D. Floreano, "Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments," *Evol. Comput.*, vol. 9, no. 4, pp. 495–524, 2001.

[7] O. Miglino, H. H. Lund, and S. Nolfi, "Evolving mobile robots in simulated and real environments," *Artif. Life*, vol. 2, no. 4, pp. 417–434, 1995.

[8] S. Nolfi, "Evolutionary robotics: Exploiting the full power of self-organization," *Connect. Sci.*, vol. 10, no. 3-4, pp. 167–184, 1998.

[9] G. Hesslow, "Conscious thought as simualtion of behaviour and perception," *Trends in Cognitive Science*, vol. 6(6), pp. 242–247, 2000.

[10] N. S. and P. D., *Handbook of brain theory and neural networks, Second Edition*, ch. Evolution of artificial neural networks, pp. 418–421. Cambridge, MA: MIT Press, 2002.

[11] M. Hlse, B. Lara, F. Pasemann, and U. Steinmetz, "Evolving neural behaviour control for autonomous robots," 2001.

[12] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, pp. 1118–1121, November 2006.

[13] R. D. Beer, "Toward the evolution of dynamical neural networks for minimally cognitive behavior," 1996.

[14] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada, "How to evolve autonomous robots: Different approaches in evolutionary robotics," in *4th International Workshop on Artificial Life* (R. A. Brooks and P. Maes, eds.), MA: MIT Press, 1994.

[15] D. Floreano, P. Drr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[16] K. O. Stanley and R. Miikkulainen, "Efficient reinforcement learning through evolving neural network topologies," in *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002*, Morgan Kaufmann, 2002.

[17] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[18] X. Yao and Y. Liu, "Towards designing artificial neural networks by evolution," *Applied Mathematics and Computation*, vol. 91, pp. 83–90, 1996.

[19] F. Pasemann, M. Hlse, , U. Steinmetz, and B. Lara, "Robot control and the evolution of modular neurodynamics," 2001.

[20] A. T. Department and A. Teller, "Learning mental models," in *Advances In Genetic Programming*, pp. 199–219, MIT Press, 1993.

[21] A. Silva, A. Neves, and E. Costa, "Building agents with memory: An approach using genetically programmed networks," in *In the Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99*, pp. 6–9, Morgan Kaufmann, 1999.

[22] A. Silva, A. Neves, and E. Costa, "Evolving controllers for autonomous agents using genetically programmed networks," in *Proceedings of the Second European Workshop on Genetic Programming*, (London, UK), pp. 255–269, Springer-Verlag, 1999.

[23] J. R. Koza, "Introduction to genetic programming: tutorial," in *GECCO (Companion)*, pp. 2299–2338, 2008.

[24] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robot. Auton. Syst.*, vol. 57, pp. 345–370, April 2009.

[25] S. J. Blakemore, D. Wolpert, and C. Frith, "Why can't you tickle yourself?," *Neuroreport*, vol. 11, pp. 11–16, 2000.

[26] S. J. Blakemore, S. J. Goodbody, and D. M. Wolpert, "Predicting the consequences of our own actions: The role of sensorimotor context estimation," *The Journal of Neuroscience*, vol. 18, no. 18, pp. 7511–7518, 1998.

[27] C. Zhang and H. Shao, "An ann's evolved by a new evolutionary system and its application," 2000.

[28] G. L. Osella Massa, H. Vinuesa, and L. Lanzarini, "Modular creation of neuronal networks for autonomous robot control," *Inteligencia Artificial, Revista Iberoamericana de IA*, 2007.

[29] A. C. Slocum, D. C. Downey, R. D. Beer, and A. D. Beer, "Further experiments in the evolution of minimally cognitive behavior: From perceiving affordances to selective attention," in *In*, pp. 430–439, MIT Press, 2000.

[30] D. M. Wolpert, R. C. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends in Cognitive Sciences*, vol. 2, no. 9, pp. 338 – 347, 1998.

[31] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*.

[32] H. Hoffmann, "Perception through visuomotor anticipation in a mobile robot," *Neural Netw.*, vol. 20, no. 1, pp. 22–33, 2007.

[33] T. Ziemke, D.-A. Jirenhed, and G. Hesslow, "Internal simulation of perception: a minimal neuro-robotic model.," *Neurocomputing*, vol. 68, pp. 85–104, 2005.

[34] A. Weitzenfeld, "From schemas to neural networks: A multi-level modelling approach to biologically-inspired autonomous robotic systems," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 177–197, 2008.

[35] J. R. Koza and J. P. Rice, "Automatic programming of robots using genetic programming," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 194–201, The MIT Press, 1992.

[36] S. Harnad, "The symbol grounding problem," 1990.

[37] S. Harnad, "Why and how we are not zombies," 1995.

[38] S. Harnad, "Minds, machines and searle," 1989.