Universidad Zaragoza
1542

Trabajo Fin de Grado

Diseño y análisis de redes homeostáticas adaptativas

Design and analysis of homeostatic adaptative networks

Autor

Alberto Martínez Menéndez

Directores

Manuel Gonzalez Bedía

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. _____,

con nº de DNI _____ en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) _____, (Título del Trabajo)

_____

_____

_____

_____

_____

_____,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, _____

Fdo: _____

AGRADECIMIENTOS

Agradecimientos mas formales

Asier por aguantar Latex, a mi madre y a Bea por aguantarme a mi. A Manuel por su onvolucracion y su ayuda

Si nos ayuda alguien externo, tambien a el

Diseño y análisis de redes homeostáticas adaptativas

RESUMEN

Conseguir dotar a un agente artificial con la capacidad de adaptabilidad presente en los seres vivos le proporcionaria las habilidades necesarias para que pudiera realizar tareas para las que no ha sido entrenado o para realizar tareas para las que se le ha entrenado pero en un entorno cambiante o con incertidumbre. Los mecanismos necesarios para conseguir estas propiedades se llaman mecanismos homeostáticos.

En este trabajo se ha buscado diseñar e implementar un agente homeostático basado en redes neuronales recurrentes de tiempo continuo con capacidad de fototaxis (búsqueda y acercamiento a una serie de luces en un espacio). Este agente se corresponde con el mejor candidate de entre una cierta población de candidatos, el cual ha sido seleccionado mediante un algorítmo genético. El agente desarrolla el comportamiento homeostático a través de la evolución y de sus mecanismos de plasticidad, que le permiten modificar sus variables internas para alcanzar un estado estable y mantenerse en el mismo.

Una vez obtenido el agente con las propiedades de fototaxis y plasticidad, se ha buscado dotar al mismo de comportamientos sociales para estudiar como interacciona en situaciones o pruebas donde hay más de un agente involucrado. El comportamiento social se ha añadido al agente siguiendo dos aproximaciones diferentes. La primera, desarrollada por () defiende que el comportamiento social es una característica que se tiene de manera innata pero que se va entrenando con el tiempo. La segunda, desarrollada por () defiende que el comportamiento social es una característica añadida a la red neuronal e independiente de la misma. El objetivo es comparar estas dos aproximaciones y sacar conclusiones sobre ello creando un experimento en el que varios agentes tienen que interactuar de manera social para la obtención de un objetivo.

Design and analysis of homeostatic adaptative networks

ABSTRACT

El resumen pero traducido al ingles

Índice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Incorporar comportamiento adaptativo en agentes artificales es uno de los medios esenciales para que estos puedan desenvolverse en entornos con cambios o incertidumbre de manera satisfactoria.

En ingeniería, uno de los pioneros trabajos en este ámbito se debe a William Ashby ('A Design for a Brain'), en el que se interesó por investigar como funcionan los mecanismos de adaptación en sistemas vivos. La tesis de Ashby fue que todos los organismos tienen ciertas variables esenciales, que deben mantenerse dentro de unos límites, y que a menudo están vinculadas entre sí, provocando que cambios en algunas de ellas puedan afectar a las demás. El estado de supervivencia del organismo ocurre cuando un comportamiento no mueve ninguna variable esencial fuera de sus límites. Al conjunto de mecanismos que permiten esta estabilidad se lo conoce como regulación homeostática. En lugar de atribuir a los organismos conductas intencionales de búsqueda de objetivos, Ashby exploró estos mecanismos que permitían generar un comportamiento adaptativo autoinducido mediante el mantenimiento de la estabilidad interna de sus variables esenciales.

Trabajos recientes en neurociencia han mostrado que existen mecanismos reguladores homeostáticos de eficacia sináptica en el cerebro. Estos mecanismos entre neuronas están impulsados por la necesidad de adaptarse a los cambios que dependen de una actividad, al tiempo que mantienen la estabilidad interna.

Dotar a un agente artificial de propiedades de adaptación homeostática le permitiría adaptarse y resolver problemas para los que inicialmente no fue entrenado, permitiéndole alterar sus variables internas hasta obtener una estabilidad completa de su sistema al mismo tiempo que ejecuta las tareas encomendadas.

1.1. Contexto y motivación

Me planteo estudiar el comportamiento de agentes con objetivos de fototaxis y propiedades adaptativas basadas en la plasticidad en entornos coletivos, donde más de un agente participa en una determinada tarea u objetivo. Las capacidades sociales de los agentes se añadiran a los mismos siguiendo dos ideas distintas. La primera, de (NOMBRE DEL TIO), defiende que el comportamiento social es una capacidad innata que se va entrenando conforme pasa el tiempo. La segunda, de (NOMBRE DEL TIO), defiende que el comporamiento social es una capacidad externa a las capacidades innatas y que puede añadirse de manera independiente al individuo. Por tanto, se desarrollarán dos agentes, cada uno con las capacidades sociales añadidas siguiendo cada una de las anteriores ideas, para posteriormente someter a ambos a una prueba de la que se puedan obtener conclusiones que permitan comparar las dos aproximaciones.

1.2. Objetivos

En este trabajo de fin de grado se pretenden aplicar y analizar las ideas de ultrastabilidad de William Ashby en agentes artificiales. Dichos agentes serán diseñados mediante controladores basados en redes neuronales recurrentes a las que se les impondrá una regulación homeostática de su actividad sináptica (las neuronas tendrán funciones de activación adecuadas para generar estas estructuras de estabilidad). Los agentes serán evolucionados para solucionar un problema de fototaxis (búsqueda y acercamiento a una fuente de luz). Una vez programados agentes artificiales homeostáticos se buscará analizar:

- Si los agentes pueden adaptarse a cambios en las reglas del entorno, aunque no hayan evolucionado específicamente para resolver nuevas tareas. Se buscará probar que, tal como ocurre en organismos vivos, estos mecanismos de adaptación facilitan implícitamente el mantenimiento de estabilidad y el desarrollo de nuevos comportamientos.

- Si en entornos sociales de agentes artificiales, donde los comportamientos para resolver problemas se basen en estrategias colectivas, los patrones de estabilidad son más robustos. Para ello se someterán las configuraciones obtenidas a estímulos ruidosos o perturbaciones, y mediremos el grado de estabilidad de los patrones homeostáticos.

1.3. Estructura

La memoria cuenta con un primer apartado de introducción, seguido por otro del estado del arte, en los que se explica el contexto del problema a resolver, los principales objetivos, motivaciones y situación actual del ámbito en el que se desarrolla el trabajo. A continuación se encuentra el apartado de diseño e implementación, en el cual se habla sobre las distintas herramientas y técnicas utilizadas para la realización del trabajo. Por último, en la sección de experimentación se enuncian y se analizan los dos procesos que se han ejecutado para la realización de los dos objetivos anteriormente descritos.

En los anexos se puede observar el tiempo invertido en este proyecto junto con las tareas las que se dedicó. Además, se incluye el código correspondiente en el lenguaje de programación Python desarollado para la obtención de los resultados necesarios.

Capítulo 2

Estado del arte

Hábitos

En nuestro comportamiento diario desplegamos numerosos patrones de conducta que se han ido fortaleciendo con la experiencia, de un modo continuado, y sostenidos por la repetición. Por ejemplo, mirar hacia la izquierda o hacia la derecha antes de cruzar la calle, atarse los cordones de los zapatos o simplemente caminar, pueden entenderse como patrones anidados de coordinación sensomotora, consolidados en nuestra conducta por una historia de refuerzos: A estos esquemas de conducta les llamamos hábitos.

Durante un tiempo, los hábitos fueron la piedra angular en psicología hasta el surgimiento del computador en la década de los sesenta del siglo pasado. Desde entonces, los modelos psicológicos que se extendieron y que funcionaron como inspiración entre los ingenieros que desarrollaban máquinas inteligentes, asumieron típicamente modelos internos de procesamiento simbólico-lingüístico, relegando el concepto de hábito al de un simple automatismo de estímulo-respuesta, lo que supone una simplificación muy alejada de la esencia y la génesis de los hábitos.

¿A qué o cómo definiríamos un hábito en sentido estricto? Podríamos decir que es un patrón automantenido de conducta. Decimos auto-mantenido porque la estabilidad de la conducta esta acoplada con la estabilidad de los mecanismos que la generan. El hábito "llama" a realizar una conducta y la conducta refuerza el hábito.

En este trabajo, estamos interesados en analizar hábitos desde un punto de vista operativo para implementar agentes virtuales cuyo comportamiento se base precisamente en ellos. De modo que nuestras implementaciones deberán cumplir aspectos como que: (i) los hábitos no presupongan una prioridad causal entre percepción y acción, sino que integren a las dos; (ii) los hábitos sean plásticos y maleables (a diferencia de la rigidez que acompañan a las nociones de arco reflejo); (iii) los hábitos provean un sentido concreto de automantenimiento en el nivel de la conducta (es decir, sean formas de acción auto-reforzadas).

Agentes artificiales basados en hábitos: Operativización de la noción de homeostasis

Para modelar este tipo de hábitos en agentes artificiales que se desenvuelvan en entornos no conocidos y desarrollen tareas (es decir, para programar este tipo de patrones que se automantienen y fortalecen por la experiencia de practicarlos) nos interesaremos por las nociones que, en la década de los cuarenta, William Ross Ashby (1903-1972), desarrolló para operativizar e implementar en formar de artefacto, el concepto de homeostasis.

En general, la homeostasis es la capacidad de un organismo para mantener constante las propiedades físico-químicas de su medio interno. Es una habilidad que permite que su situación físico-química característica se conserve constante dentro de unos límites, aunque en el exterior existan fuentes que puedan ser motivo de alteración. Por ejemplo, el cuerpo humano, frente a fuentes de cambio externas, moviliza mecanismos de autorregulación (como el sistema nervioso central, sistema endocrino, sistema respiratorio, sistema circulatorio, etc.) para que se mantengan de forma constante sus condiciones de vida.

Aunque tradicionalmente es un término vinculado a la biología o la fisiología, otras ciencias y técnicas alejadas se han interesado por el fenómeno homeostático y han adoptado también este concepto a sus intereses. En su adopción técnica, la homeostasis es simplemente el rasgo de sistemas autorregulados que consiste en la capacidad para mantener ciertas variables en un estado estacionario, de equilibrio dinámico o dentro de ciertos límites, cambiando parámetros de su estructura interna. En este sentido, Ashby fue el primero que capturó operativamente estas ideas y construyó un homoestato artificial para demostrar las características de comportamiento de, lo que él llamó, un sistema ultraestable. El homeostato de Ashby respondía de manera aleatoria a las desviaciones de ciertos valores en sus parámetros esenciales y solo descansaba en su variación al azar cuando encontraba un comportamiento que mantenía los valores de esas variables críticas dentro de sus límites deseables. Las ideas de Ashby, desarrolladas en su famoso libro "Design for a Brain", dieron lugar al campo de estudio de los sistemas biológicos como sistemas homeostáticos y adaptativos en términos puramente de matemática de sistemas dinámicos.

Estas sugerentes ideas permiten operativizar nociones de automantenimiento biológico y fisiológico y llevarlas a un entorno de implementación de agentes artificiales. Como recordamos, nuestro objetivo era tener esquemas ultraestables en el ámbito de la conducta de un agente vitual. ¿Cuáles y cómo podrían ser los componentes de una organización neuronal-conductual para poder hacer una analogía entre la vida orgánica y la conducta de un agente programado?

Si, a partir de conceptos como la homeostasis, la supervivencia de la vida orgánica puede definirse como una red auto-mantenida de reacciones químicas alejadas del equilibrio, la conducta podría definirse análogamente como una red auto-mantenida de hábitos o estructuras neurodinámicas. El objetivo es extender las nociones fisiológicas a estructuras neuronales. Y, en este segundo caso, la conservación adaptativa de organizaciones neurodinámicas (en lugar de organizaciones fisiológicas) no vendría por mecanismos de tipo biológico sino mediante regulación sensomotora asociada a la conducta del agente (en lugar de basada en esquemas metabólicos).

Controladores con plasticidad homeostática

Esta analogía vida-conducta, que podríamos denominar interesada en nociones de "homeostasis en sistemas nerviosos artificiales", ha sido poco explorada[1, 2, 3, 4, 5]. Tomaremos la misma aproximación que otros investigadores han hecho y que se basa en el uso de redes recurrentes continuas para modelar el sustrato de un controlador neuronal sobre el que se incorporan mecanismos basados en plasticidad hebbiana para garantizar que regímenes de activación de las redes en su conjunto, tienen características homeostáticas que les permiten lograr condiciones de ultraestabilidad.

Esta "plasticidad homoestatica"no es solo un mecanismo formal sino que tiene evidencia empírica [6], pues se ha observado que las neuronas buscan estabilidad en su frecuencia de disparo. El modo de conseguir estabilidad conductual, por tanto, parece exigir que los modos de estabilidad en la conducta aparezcan acoplados a los mecanismos de estabilidad de los parámetros sinápticos, de manera que cuando la estabilidad conductual se pierde induce inestabilidad sináptica hasta recuperar la estabilidad conductual.

El método para implementar plasticidad homoestática normalmente se apoya en los resultados de la teoría Hebbiana, que describe un mecanismo básico de plasticidad sináptica en el que el valor de una conexión sináptica se incrementa si las neuronas de ambos lados de dicha sinapsis se activan repetidas veces de forma simultánea. Introducida por Donald Hebb, en 1949, es también llamada regla de Hebb"o "Teoría de la Asamblea Celular" y afirma que, en palabras del propio Hebb: «Cuando el axón de una célula A está lo suficientemente cerca como para excitar a una célula B y repetidamente toma parte en la activación, ocurren procesos de crecimiento o cambios metabólicos en una o ambas células de manera que tanto la eficiencia de la célula A, como la capacidad de excitación de la célula B son aumentadas».

De manera menos rigurosa, la teoría se resume a menudo como que "las células que se disparan juntas, permanecerán conectadas", aunque esto es una simplificación y no debe tomarse literalmente, así como no representa con exactitud la declaración

original de Hebb sobre cambios de la fuerza de conectividad en las células. Sin embargo, ese principio es que se suele utilizar en el ámbito del Aprendizaje automático y de los modelos en Neurocomputación. La teoría es comúnmente evocada para modelar algunos tipos de aprendizajes asociativos en redes neuronales artificiales en los que la activación simultánea de las células conduce a un pronunciado aumento de la fuerza sináptica (conocido como aprendizaje de Hebb).

Controladores diseñados mediante computación evolutiva

Hasta ahora hemos visto que sería posible implementar agentes cuya conducta esté basada en hábitos a partir de mecanismos neuronales con plasticidad hebbiana que garantizarían condiciones de homoestaticidad en la red como conjunto. Nos queda un último punto: seleccionar el tipo de red neuronal más adecuada para que puedan implementarse estos procesos y que suponga la base de un controlador de agentes virtuales. Para ello, nos fijaremos en aspectos de la metodología denominada Computación Evolutiva, que se usa para el diseño de controladores neuronales evolutivos genéticamente determinados. Dichos controladores son aplicados a agentes (reales o virtuales) tipo Khepera para llevar a cabo tareas (por ejemplo, despliegue de trayectorias en navegación) dentro de un entorno no conocido previamente. Los controladores están basados en redes neuronales, que mediante evolución de parámetros con algortimos genéticos, pueden ajustarse en forma on-line sin entrenamiento adicional a cambios del entorno, permitiendo un control apropiado de los este tipo de agentes robóticos. En los últimos años, en la mayoría de los experimentos de computación evolutiva son utilizados sistemas de control mediante redes neuronales con conexiones recurrentes, las cuales permiten que la red contemple aspectos de dinámica temporal[7].

La metodología empleada es la siguiente: una población de cromosomas artificiales es creada aleatoriamente y probada dentro del entorno del agente virtual. Concretamente, cada elemento de la población codifica el sistema de control de un robot. Cada robot es libre de actuar según al controlador utilizado, el cual fue genéticamente determinado, mientras es evaluado su desempeño (fitness) al realizar una tarea. Este proceso de generación de controladores robóticos y su evaluación en el entorno del robot, es llevado a cabo hasta satisfacer un criterio preestablecido (una fitness mínima que se supone aceptable por el programados) vinculado a la tarea a desarrollar.

En relación con la metodogía mencionada, el desarrollo de tales controladores neuronales presupone la aceptación de algunos criterios, como los que a continuación se describen:

– Tipo de controlador neuronal: por lo general, el tipo de controlador utilizado es una red recurrente continua en el tiempo (Continuous Time Recurrent Neural Network. CTRNN, en su acrónimo en inglés). Estas son redes continuas cuyas neuronas están totalmente conectadas (se explotan los diversos bucles retroalimentados) con la peculiaridad matemática de que pueden aproximar cualquier sistema dinámico [8].

– Codificación genética de los controladores: se usan controladores genéticamente determinados (los pesos sinápticos son determinados genéticamente) y, en ocasiones, controladores con sinapsis adaptativas (los pesos sinápticos son siempre aleatorios y posteriormente modificados según reglas de adaptación de las sinapsis, las cuales son determinadas genéticamente).

– Modelo del robot: el tipo de robot a utilizar debe poseer un conjunto de sensores, los cuales permitan la navegación autónoma basada en ellos. En su mayoría, los trabajos presentados utilizan un agente virtual inspirado en el robot denominado Khepera.

– Tarea a realizar por el robot: diferentes clases de problemas pueden ser afrontados al momento de evaluar un controlador como el aquí expuesto. Principalmente, algunos de estos problemas se enmarcan en Aprendizaje por Refuerzo, análisis de las capacidades de adaptación a los cambios, estudio de la influencia de cambios en el entorno, navegación autónoma, etc.

– Hipótesis a probar: Existe un amplio espectro de áreas de investigación que pretenden ser estudiadas con el enfoque aquí presentado. Dichas áreas involucran desde aspectos ingenieriles hasta científicos[9], algunos de los cuales son: controladores bio-inspirados, modularidad y plasticidad neuronal, la perspectiva comportamental, aprendizaje, etc.

Conclusiones

En conclusión, recopilando lo visto hasta ahora: (i) nos interesa el estudio de comportamiento de agentes basados en hábitos y es exige incorporar en nuestros modelos nociones de plasticidad homeostática; (ii) los modos de implementación se basarán en modelos de aprendizaje hebbiano y en redes recurrentes continuas como controladores de los agentes virtuales programados; (iii) metodológicamente, se diseñará una aquitectura neuronal con parámetros libres, se codificará genéticamente y se evolucionará en relación con la tarea a realizar, lo que implica definir una función

de evaluación o función de fitness, la cual determinará el grado de aceptación de cada controlador neuronal en el proceso evolutivo.

Capítulo 3

Fundamentos teóricos

3.1. Redes neuronales recurrentes de tiempo continuo (CTRNN)

En contraste con las redes neuronales feed-forward, las cuales soportan únicamente comportamientos reactivos, en las Redes Neuronales Recurrentes de Tiempo Continuo (CTRNN) [10] pueden existir ciclos en su estructura y la activación de sus neuronas es asíncrona y multiescalada en el tiempo. Este tipo de redes neuronales también facilita describir el agente como un sistema dinámico acoplado al entorno en el que está ubicado, ya que está demostrado que son el modelo más simple de red neuronal dinámica continua no lineal [8] Además, la interpretación neurobiológica de las CTRNN ha sido demostrada y puede consultarse en [10].

Descripción matemática de una CTRNN

Las CTRNN están formadas por neuronas cuyo comportamiento se describe en las ecuaciones 3.1 y 3.2

$$\dot{y_i} = \frac{1}{\tau_i} * \left(-y_i + \sum_{j=1}^{N} w_{ji} * \sigma\left(y_j + \theta_j\right) + I_i \right) \qquad i = 1, 2, ..., N \tag{3.1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

donde y_i es el estado de la neurona, w_{ji} es el peso de la conexión entre las neuronas i y j, θ es el término bias, I representa una entrada externa y τ hace que cada una de las neuronas dependa del tiempo, ya que para diferentes valores la caída del nivel de activación de la neurona es más rápida o lenta. En la fórmula 3.1 la velocidad de actualización de la red neuronal debe ser notablemente mayor (el intervalo entre dos actualizaciones será menor) que el valor de τ para no obtener comportamientos no deseados.

Valores de activación y de salida de la neurona de una CTRNN

Para poder entender cómo deben interpretarse la activación y la salida de una CTRNN, se va a utilizar como ejemplo una CTRNN formada por una única neurona autoconectada como la de la figura 3.1. El valor de salida o de una neurona será un valor real entre 0 y 1 obtenido al aplicar la función sigmoide (ecuación 3.2) a la suma del estado actual y de la neurona con su valor bias θ, tal y como puede verse en la figura 3.1.

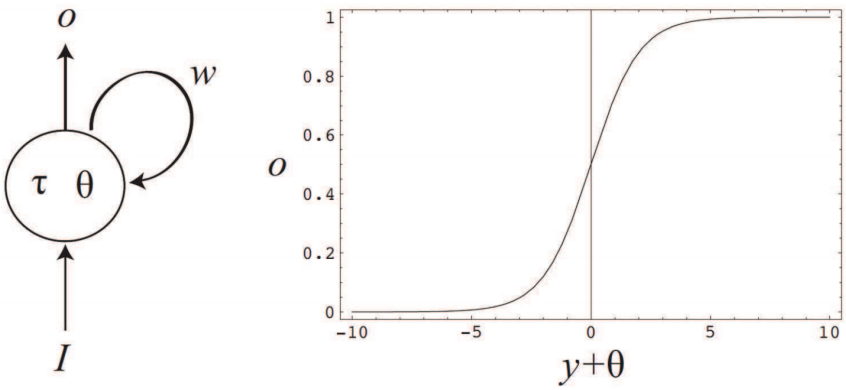

Figura 3.1: Valor de salida de una CTRNN. **Izquierda:** Un nodo autoconectado. **Derecha:** Función sigmoide aplicada para calcular la salida de una neurona.

En cuanto al valor de activación, a diferencia de una red neuronal feed-forward, la cual realiza un mapeo directo entre entrada y salida de la red, el comportamiento de una CTRNN corresponde al de un sistema dinámico [10], por lo que el valor de activación de la neurona convergerá a un punto de equilibrio. Para el análisis de las dinámicas del sistema formado por el agente controlado por la CTRNN y el entorno, se analizarán sus diagramas de bifurcación, los cuales muestran todos los puntos de equilibrio para la activación de las neuronas de la red.

3.2. Homeostasis

A pesar de que la homeostasis es una propiedad biológica, es posible representar un sistema lógico que se comporte de forma similar a como sistemas vivos con estas propiedades lo harían[11].

Un *Homeostato*, es un sistema artificial que presenta porpiedades de homeostasis. El Homeostato desarrolado por Ashby era un aparato electromagnético mientras que el que va a ser utilizado es un componente implementado computacionalmente.

Un Homeostato es un sistema de N neuronas completamente interconectadas entre

si. Cada neurona recibe N entradas, de si misma y del resto de neuronas, dependientes del peso de la fuerza de conexión de las uniones entre ellas (ver Figura 3.2).

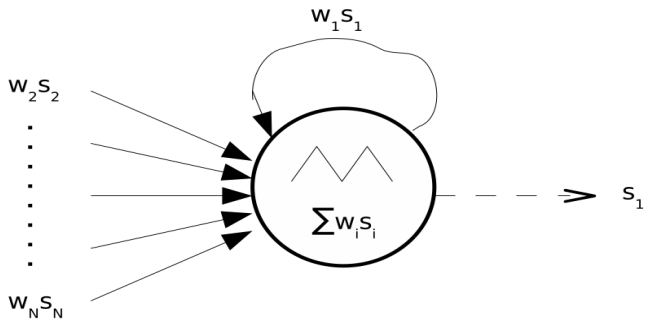

Figura 3.2: Esquema de un Homeostato individual. La unidad recibe las entradas del resto de unidades ($w_i s_i$) y de si misma ($w_1 s_1$). La salida s_1 es una función lineal por partes de la suma ponderada de las entradas.)

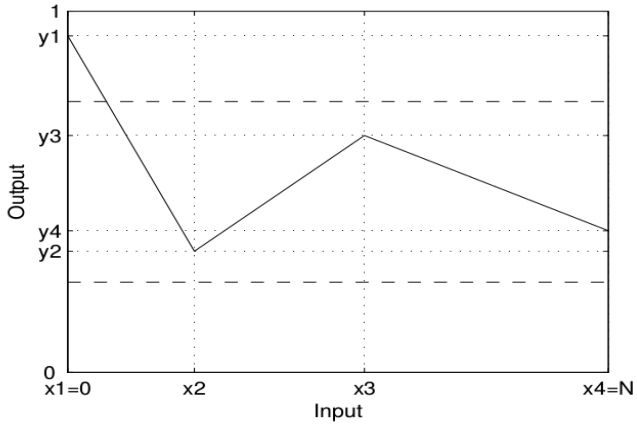

Figura 3.3: Ejemplo de función de transferencia de una unidad Homeostat. La función es lineal por partes con puntos en (x_1, j_1),...,(x_p, y_p) donde $x_1 = 0$ y $x_p = N$, para todas las unidades en una red de N unidades. Las lineas discontinuas representan el rango homeostatico de la red.

La suma ponderada I (ver equación 3.3) de las entradas de una unidad determina su salida s, tal y como esta especificado por la función linear de transferencia por partes F (equación 3.4 y figura 3.3).

$$I = \sum_{i}^{N} w_i s_i \tag{3.3}$$

$$s = F(I) = \begin{cases} y_1 + (y_2 - y_1)(\frac{I - x_1}{x_2 - x_1}) & : \quad x_1 \leq I < x_2 \\ y_2 + (y_3 - y_2)(\frac{I - x_2}{x_3 - x_2}) & : \quad x_2 \leq I < x_3 \\ y_3 + (y_4 - y_3)(\frac{I - x_3}{x_4 - x_3}) & : \quad x_3 \leq I \leq x_4 \end{cases} \tag{3.4}$$

Al inicializar la red, los pesos de las conexiones se aleatorizan a partir de una distribución uniforme de rango apropiado. El rango objetivo (o rango límite) $R = [0{,}5 - \delta, 0{,}5 + \delta]$ es especificado para la salida, donde δ determina la rigidez de la restricción homeostática. Si $s \in R$ la unidad es homeostática. Si $s \notin R$ significa que la homeostasis se ha perdido y se activan los mecanismos de cambio adaptativo.

Hay dos mecanismos de cambio adaptivo que se aplican a los parametros de las unidades no homeostáticas. El primer mecanismo asigna valores aleatorios a los pesos de las conexiones aferentes a la unidad (equación 3.5). El segundo mecanismo asigna nuevos valores aleatorios a los parametros que indican las coordenadas de la función de transferencia de la unidad (equación 3.6). Los rangos de estas reasignaciones son los mismos utilizados en la inicialización. Donde $rand(a, b)$ representa a la función que devuelve un número real aleatorio obtenido de una distribución uniforme en el rango $[a, b]$.

$$IF \quad (s \notin R) \quad THEN \quad [w = rand(0{,}00, 1{,}00) \quad \forall w \in \{w_1, ..., w_N\}] \tag{3.5}$$

$$\begin{aligned} IF \quad (s \notin R) \quad THEN \quad & [x = rand(0{,}00, N) \quad \forall x \in \{x_2, ..., x_{p-1}\}] \\ AND \quad & [y = rand(0{,}00, 1{,}00) \quad \forall y \in \{y_1, ..., y_p\}] \end{aligned} \tag{3.6}$$

3.3. Algoritmo genético

Los algoritmos genéticos son un tipo de algoritmos de optimización, lo que quiere decir que se utilizan para obtener la solución o soluciones óptimas de un problema computacional[12].

Estos algoritmos representan una rama dentro del campo conocido como "Computación evolutiva". En ella, se imitan los procesos biológicos de reproducción y selección natural con el fin de encontrar la mejor solución al problema para el que son creados. Al igual que en la evolución, muchos de los procesos de un algoritmo genético son aleatorios, aunque las técnicas de optimización permiten limitar el nivel de aleatoriedad y adaptar el control sobre el algoritmo que se considere necesario.

Este tipo de algoritmos permiten encontrar soluciones a problemas que otros algoritmos de optimización no pueden calcular por falta de continuidad, derivabilidad, linealidad u otras características.

3.3.1. Componentes, estructura y terminología

Dado que los algoritmos genéticos están diseñados para simular procesos biológicos, gran parte de su terminología ha sido tomado prestada de la biología. A pesar de ello,

las entidades a las que esta terminología se refieren en el entorno de los algoritmos genéticos son mucho más simples que sus equivalentes biológicos. Los componentes básicos de prácticamente todos los algoritmos genéticos son:

- Una función **fitness** para la optimización
- Una **población** de **cromosomas**
- Una función de **selección** para elegir los cromosomas que van a reproducirse
- Una función de **recombinación** (o *crossover*) para la producción de la siguiente generación de cromosomas
- Una función de **mutación** para aleatorizar algunos cromosomas de la nueva población

La *función fitness* es la función que el algoritmo está tratando de optimizar. La palabra "fitness"se toma prestada de la teoría evolutiva. Se utiliza porque la función de fitness testea y valora como de buena (como de "*fit*") es cada solución potencial del problema. El término *cromosoma* hace referencia al valor o valores numéricos que representan posibles soluciones candidatas al problema que se está intentando resolver. Cada cromosoma está formado por una lista de parámetros, llamados *genes*. Al conjunto de cromosomas que van a evaluarse como posibles candidatos a solución del problema se le llama *población*. Antiguamente los cromosomas eran codificados como listas de unos y ceros (codificación binaria), sin embargo, actualmente la computación moderna permite codificar los cromosomas como números reales u otros objetos.

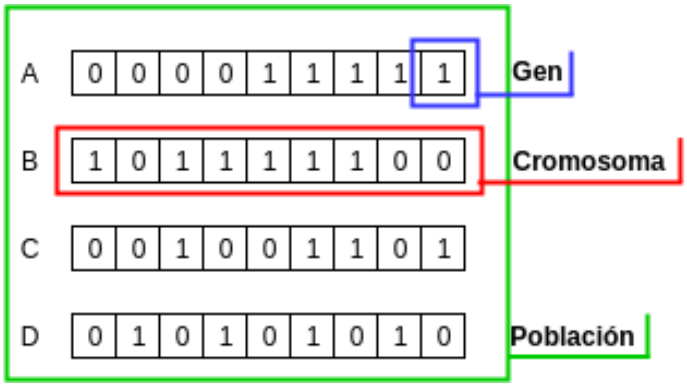

Figura 3.4: Ejemplo de estructura de población, cromosomas y genes.

Un algoritmo genético comienza con una *población inicial* de cromosomas, generados de forma aleatoria, que representan la primera generación de posibles soluciones. A continuación, se evalúa cada cromosoma de dicha población mediante

la función fitness para comprobar cómo de bien resuelve el problema planteado. Al finalizar, cada candidato tiene asignada una puntuación de fitness que representa lo bien (o lo mal) que lo ha hecho intentando resolver el problema. Una vez con la población inidical completamente evaluada, el *operador de selección* escoge de entre toda la población de candidatos algunos cromosomas para la reproducción (creación de una nueva generación). Existen muchos tipos de algoritmos de selección, pero generalmente en todos ellos se eligen los cromosomas que mejor puntuación fitness han obtenido, buscando que las siguientes generaciones creadas a partir de ellos den resultados como mínimo igual de buenos a los suyos. En el siguiente paso, el *operador de recombinación* simula las operaciones de recombinación de cromosomas presentes en las operaciones de división celular de tipo *meiosis*. En la figura 3.5 se puede observar un ejemplo de operación de recombinación. En este proceso se seleccionan dos cromosomas de la población y se recombinan para crear dos descendientes.

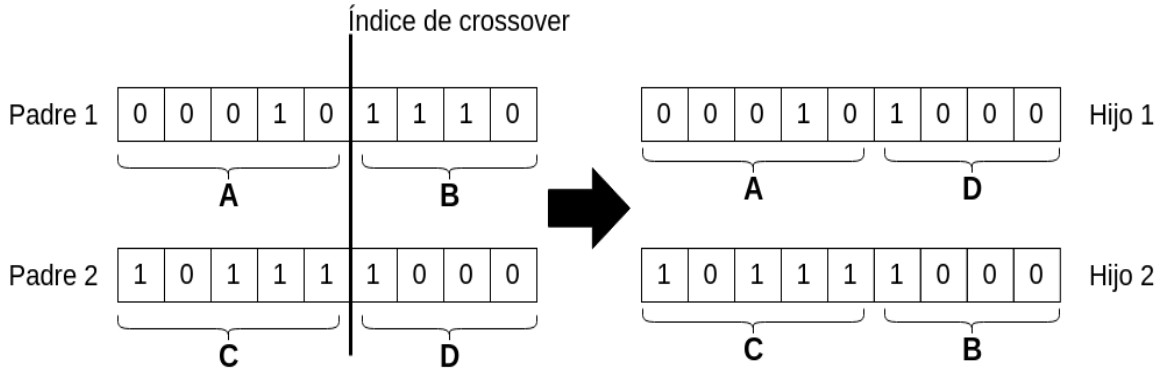

Figura 3.5: Ejemplo de operación de recombinación de dos cromosomas.

El *operador de mutación* modifica de forma aleatoria genes de los cromosomas descendientes (por ejemplo cambiando un 1 por un 0 y viceversa, en cromosomas binarios). Estas mutaciones ocurren generalmente con una probabilidad bastante baja. A primera vista, el proceso de mutación puede parecer inecesario e incluso entorpecedor. La realidad es que toma un papel muy importante. Las operaciones de selección y recombinación mantienen la información genética de los cromosomas con mejor puntuación fitness, pero estos cromosomos sólo son considerados los mejores respecto a la población de la generación en la que fueron elegidos. Esto puede provocar que el algoritmo converga demasiado rápido y de una solución final alejada de la solución máxima. En otras palabras, puede provocar que el algoritmo se quede atascado en un máximo local antes de encontrar el máximo global óptimo. El operador de mutación ayuda a proteger el algoritmo de este problema manteniendo la diversidad en la población, a costa de aumentar el tiempo de convergencia.

Lo normal es que la selección, recombinación y mutación continuen hasta que el

numero de descendientes es igual al número de cromosomas de la población inicial, para que esta segunda generación este compuesta en su totalidad por descendentienes y reemplace a la población anterior.

Ahora la segunda generación se evalúa mediante la función fitness y todo el proceso se repite. Es una práctica comúm el almacenar el cromosoma con la mayor puntuación de fitness junto con dicha puntuación para guardar un registro del mejor cromosoma hasta el momento.

Los algoritmos genéticos iteran hasta que la puntuación fitness del mejor cromosoma hasta el momento se estabiliza y no cambia en una serie de generaciones. Esto significa que el algoritmo ha convergido en una o solución. A todo el proceso de iteraciones se llama *carrera* (en inglés, *run*). Al final de cada carrera se tiene, normalmente, al menos un cromosoma que es una solución lo suficientemente buena (lo suficientemente *fit*) del problema. Dependiendo del tipo de algoritmo utilizado, la mejor solución puede ser el mejor cromosoma que ha aparecido en todas las iteraciones o el mejor cromosoma de la última generación.

El rendimiento de un algoritmo genético depende enormemente de los métodos utilizados para codificar posibles soluciones en cromosomas y el criterio utilizado en la función fitness para evaluar a los candidatos. Otros detalles importantes a tener en cuenta son las probabilidades de recombinación y mutación, el tamaño de la población y el número de iteraciones. Estos valores puedes ser ajustados convenientemente después de analizar el rendimiento del algoritmo tras unas carreras de prueba.

Los algoritmos genéticos tienen una gran variedad de aplicaciones. Algunos ejemplos destacados son la *programación automática* y el *"machine learning"*. Funcionan también especialmente bien para modelar distintos fenómenos en economía, ecología, el sistema inmunológico humano, genética y sistemas sociales.

Capítulo 4

Modelo e implementación

Para la realización de este trabajo, se han diseñado y desarrollado una serie de agentes a los que se les ha entrenado para la obtención de un comportamiento de fototaxis, es decir, de búsqueda y acercamiento a fuentes de luz.

Cada agente esta modelado como un círculo con dos sensores en la parte frontal que permiten recibir lecturas de los diferentes estímulos presentes (ya sean luces u otros agentes) y dos motores opuestos que permiten que el agente pueda moverse libremente por el espacio, como puede verse en la figura 4.1. El agente tiene en todo momento una posición en el espacio dada por unas coordenadas (x, y), así como una orientación respecto al eje X. Cada sensor esta separado 60º del eje de orientación del agente. Los sensores tienen un arco de visión de 160º, para representar que el propio cuerpo del agente pueda encontrarse entre el sensor y la luz, evitando las lecturas.

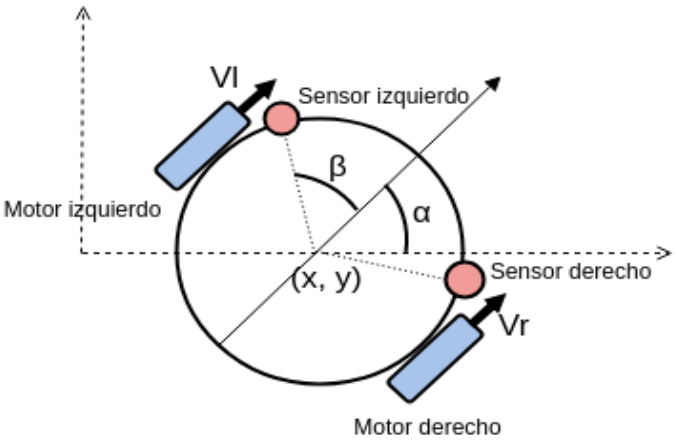

Figura 4.1: Esquema del agente utilizado.

El movimiento traslacional del agente se calcula a partir de la velocidad lineal (v) de su centro de masa (ecuación 4.1). El movimiento angular del agente se calcula obteniendo la velocidad angular (ω) de su centro de masa (ecuación 4.2).

$$v = \frac{Vr + Vl}{2} \tag{4.1}$$

$$\omega = \frac{Vr - Vl}{2R} \tag{4.2}$$

La nueva posición del agente para cada ciclo de ejecución se calcula mediante las ecuaciones (4.3, 4.4 y 4.5), donde x e y representan la posición actual del agente, θ representa la orientación actual del agente y δt representa el tamaño del tiempo de integración de ciclo.

$$x(t+1) = x(t) + v.cos(\theta).\delta t \tag{4.3}$$

$$y(t+1) = y(t) + v.sin(\theta).\delta t \tag{4.4}$$

$$\delta(t+1) = \delta(t) + \omega.\delta t \tag{4.5}$$

Para los experimentos realizados en este trabajo, el radio del agente siempre se ha tomado como **4.0**.

4.0.1. Plasticidad

La ultraestabilidad de los agentes desarrollados se ha conseguido dotando a los mismos de mecanismos de plasticidad, que les permiten modificar los valores de algunos de sus atributos internos. En este caso, los agentes pueden alterar los valores de los pesos w_{ij} de las conexiones entre las neuronas de la red cuando la frecuencia de activación de una neurona es demasiado baja o demasiado alta.

La homeostasis no se encuentra programada explícitamente, pero aparece de forma implícita debido a los mecanismos de evaluación de la evolución genética que recompensan con mejor puntuación a aquellos agentes que, durante la realización del entrenamiento, han mantenido un mayor número de sus neuronas estables.

La plasticidad de cada conexión esta gobernada por la actividad sináptica de la conexión junto con una regla de plasticidad codificada genéticamente. Estas reglas de plasticidad vienen dadas por las ecuaciones (4.6, 4.7, 4.8 y 4.9), donde Δw_{ij} es el incremento por unidad de tiempo de un determinado peso sináptico (w_{ij}), y z_i y z_j son las frecuencias de activación de las neuronas presinápticas y postsinápticas respectivamente.

Las reglas de plasticidad se construyen en torno a cuatro parámetros: la frecuencia de aprendizaje (n_{ij}), un valor límite (z_{ij}^{o}), el grado de facilitación plástica local (p_j) y un

factor de amortiguación linear que restringe el cambio dentro de los limites establecidos para los valores de los pesos sinápticos (δ).
R0: Sin plasticidad:

$$\Delta w_{ij} = 0 \tag{4.6}$$

R1: Aprendizaje Hebbiano acotado:

$$\Delta w_{ij} = \delta n_{ij} p_j z_i z_j \tag{4.7}$$

R2: Potenciación o depresión amortiguadas de la neurona presináptica cuando la eficacia sináptica es muy alta o muy baja:

$$\Delta w_{ij} = \delta n_{ij} p_j (z_i - z_{ij}^o) z_j \tag{4.8}$$

R3: Potenciación o depresión amortiguadas de la neurona postsináptica cuando la eficacia sináptica es muy alta o muy baja:

$$\Delta w_{ij} = \delta n_{ij} p_j z_i (z_j - z_{ij}^o) \tag{4.9}$$

El grado de facilitación plástica local (p_j) se ve incrementado linealmente (hasta un valor máximo de 1) cuando la frecuencia de activación aumenta hasta salirse de sus límites, facilitando así los cambios plásticos. Cuando la frecuencia de activación entra en sus límites, p_j decrece linealmente (hasta un valor mínimo de -1), disminuyendo la facilidad de los cambios plásticos.

El cambio en la conexión sináptica depende del signo de n_{ij}, de forma que cada neurona puede actuar de manera independiente facilitando los cambios plásticos en la dirección indicada.

El factor de amortiguación lineal (δ) asegura que los valores de los pesos se mantienen dentro de los límites establecidos para ellos.

El factor de valor límite (z_{ij}^o) depende linealmente del valor actual del peso que se está actualizando plásticamente, por lo que es calculado en cada modificación.

Los pesos son actualizados cada iteración (si precede, dependiendo de su regla de plasticidad) a partir de la ecuación 4.10.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \tag{4.10}$$

4.1. Agentes

Durante la realización de este trabajo se han diseñado e implementado tres tipos de agentes distintos. En esta sección se describe cada uno de ellos exponiendo la estructura de la red neuronal que actúa de controlador y las diferencias entre los controladores de cada agente.

4.1.1. Agente 0: agente sin comportamiento social

Este primer agente fue diseñado e implementado como punto de partida para probar tanto las funciones de evolución como el correcto funcionamiento de los sensores, motores, plasticidad y del comportamiento de fototaxis que sería utilizado posteriormente en los agentes con habilidades sociales.

El Agente 0 no cuenta con habilidades sociales. Su comportamiento se centra unicamente en la fototaxis, por lo que se acercará a las luces y se mantendrá cerca de ellas.

Controlador

El controlador de este agente consiste en una CTRNN de 4 neuronas completamente interconectadas entre sí y autoconectadas (conectadas con todas las demás y con ellas mismas). Como puede verse en la figura 4.2, las neuronas 1 y 2 estan conectadas a los sensores luminosos y son las encargadas de recibir las mediciones y procesarlas. Las neuronas 3 y 4 estan conectadas a los motores, y sus salidas se convierten en la velocidad de los motores, que permite al agente moverse libremente.

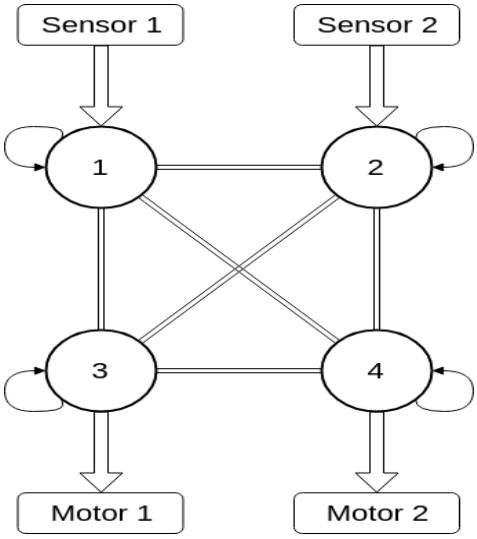

Figura 4.2: Esquema de la estructura del controlador del Agente 0.

4.1.2. Agente 1: agente colectivo según (Nombre del tio)

Este agente, al igual que el Agente 0, esta diseñado para seguir un comportamiento de fototaxis, pero además cuenta con habilidades sociales para interactuar con el resto de agentes. Las capacidades sociales del agente se han diseñado e implementado basandose en la teoría de (Nombre del tio), en la que defiende su creencia de que las habilidades cerebrales de los seres vivos pueden interpretarse como módulos independientes que se entrenan por separado. Es decir, que el comportamiento social

del agente se encuentra aislado del comportamiento de fototaxis del mismo y sus evoluciones son independientes.

El agente obtiene lecturas de los mismos sensores usados para la fototaxis, sobre las posiciones del resto de agentes que puede ver, pero estas lecturas son procesadas por una red independiente y por tanto no conectada al resto de la red.

Controlador

El controlador de este agente es similar al del Agente 0, una red CTRNN de 4 neuronas completamente interconectadas. Aunque al contar con habilidades sociales, se ha añadido una capa independiente de 2 neuronas que será la encargada de procesar las lecturas de los sensores sobre las posiciones de los agentes y de transmitir la señal adecuada a los motores del agente. Como puede verse en la figura 4.3, las neuronas 1 y 2 estan conectadas a los sensores y se encargan de recibir mediciones de luz y de procesarlas. Las neuronas 5 y 6 estan conectadas a los motores, igual que en el Agente 0. Las neuronas 3 y 4 se encargan de procesar las señales de los sensores sobre el posicionamiento del resto de agentes, procesarlas y transmitir su salida a los motores. Los motores juntan las salidas de las neuronas 3, 4, 5 y 6 para generar una velocidad.

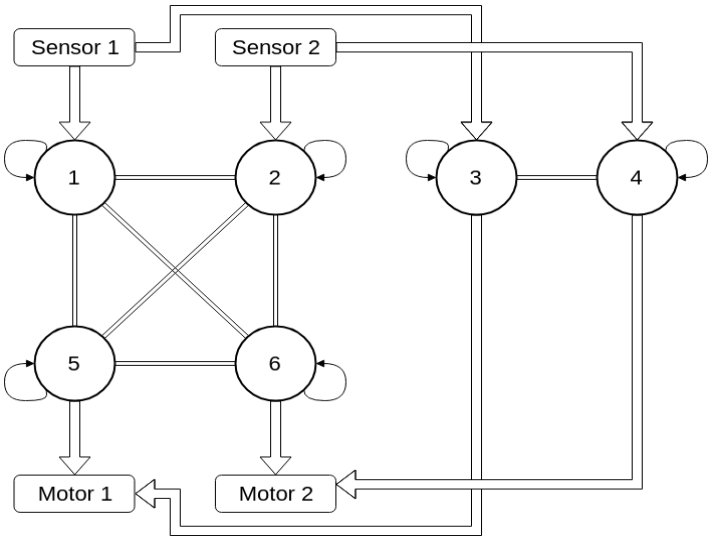

Figura 4.3: Esquema de la estructura del controlador del Agente 1.

4.1.3. Agente 2: agente colectivo según (Nombre del tio)

El controlador de este agente es una extensión del controlador del Agente 0. A la red CTRNN de 4 neuronas se le han añadido 2 neuronas más, las cuales estan conectadas a los sensores y se encargaran de procesar las mediciones sobre el posicionamiento del resto de agentes. La diferencia con el Agente 1 es que estas dos neuronas extra estan interconectadas completamente con el resto de la red, no son independientes.

Esta implementación consiste en la teoría de (nombre del señor), que defiende que las capacidades cerebrales de los seres vivos pueden aprenderse y evolucionar en momentos diversos de su vida, pero esta evolución afecta al resto de las neuronas haciendolas cambiar también. Por tanto, en el Agente 2, el comportamiento social no se encuentra aislado del comoportamiento de fototaxis.

Controlador

El controlador de este agente es similar al del Agente 1, salvo porque la cada extra de 2 neuronas no se encuentra aislada del resto, sino mezclada con el resto. Como puede verse en la figura 4.4, las neuronas 1 y 2 estan conectadas a los sensores y se encargan de recibir mediciones de luz y de procesarlas. Las neuronas 3 y 4 estan conectadas a los sensores y se encargan de procesar las señales de los sensores sobre el posicionamiento del resto de agentes, pero se encuentran completamente interconectadas con el resto de neuronas. Las neuronas 5 y 6 generan las salidas que los motores convierten en velocidades.

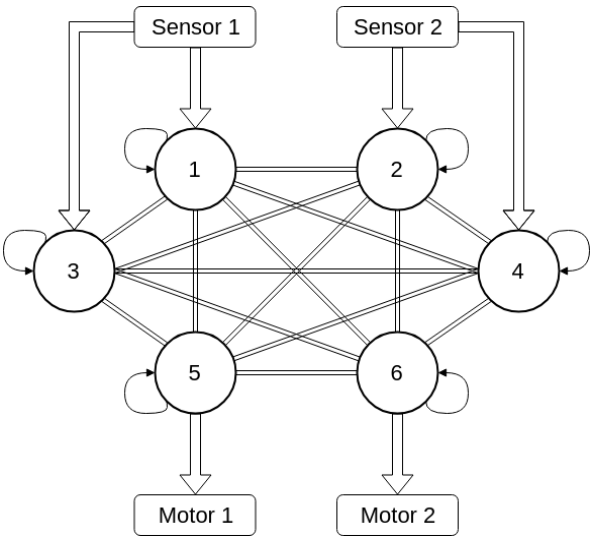

Figura 4.4: Esquema de la estructura del controlador del Agente 1.

4.2. Evolución

Para la evolución de los tres tipos de agentes anteriormente descritos se ha implementado un algoritmo genético. El algoritmo es común para los tres agentes, pero algunas de las funciones varían entre los tres tipos para adaptarse a sus estructuras, características y objetivos propios. De forma común, la estructura del algoritmo es la siguiente:

1. Se crea una población inicial de agentes candidatos a ser solución. Los valores de los atributos de estos agentes son completamente aleatorios. Constituyen la primera generación.

2. Se evalúan los agentes de la poblacion inicial aplicándoles la función fitness. Se obtiene como resultado un valor de fitness para cada agente de la primera generación.

3. Se comprueba si algun agente ha alcanzado un valor fitness lo suficientemente alto como para darlo por entrenado y así finalizar el experimento.

4. Si no se encuentra un agente con un valor fitness satisfactorio se procede a crear otra generación.

5. Se aplica la función de selección a la generación anterior, obteniendo una nueva población de agentes.

6. Se aplica la función de recombinación (*crossover*) sobre la nueva población para generar nuevos agentes.

7. Se aplica la función de mutación sobre los agentes para crear pequeñas variaciones sobre los mismos que den más variedad a la generación.

8. Se evalúa esta nueva generación de agentes aplicándoles la función fitness. Se obtiene como resultado un valor de fitness para cada agente de esa nueva generación.

9. Volvemos al punto número 3.

4.2.1. Codificación de los agentes

Cada agente candidato es codificado como un vector que contiene cada uno de sus atributos. El Agente 0 se codifica como un vector de 46 componentes formado por 42 componentes reales y 4 componentes enteras, como puede verse en la figura 4.5.

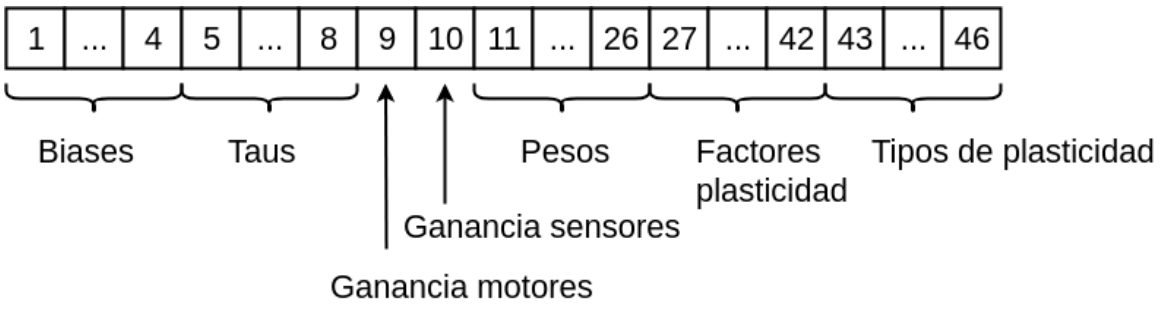

Figura 4.5: Representación codificado de un cromosoma que representa un agente de tipo 0.

Los agentes de tipo 1 y 2 estan formados por 6 neuronas, a diferencia de los agentes de tipo 0 que están compuestos por 4 neuronas. Esto hace que el vector de un Agente 1 o de un Agente 2 este formado por 92 elementos, siendo 86 de ellos reales y 6 enteros, como puede verse en la figura 4.6

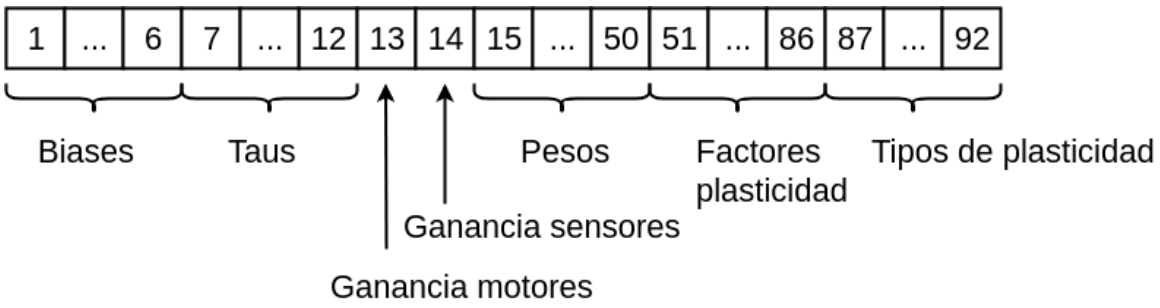

Figura 4.6: Representación codificado de un cromosoma que representa un agente de tipo 1 o 2.

Todas las componentes reales de los agentes codificados estan formadas por valores entre [0, 1], los cuales serán escalados a sus rangos establecidos a la hora de evaluar el agente. Las componentes enteras estan formadas por valores del 0 al 3, y representan los tipos de plasticidad nombrados en el apartado de plasticidad de este documento.

4.2.2. Creación de la población inicial

Para la creación de la población inicial se llama N veces a la función de creación de candidato. Esto da como resultado una población de candidatos aleatorios que constituyen la primera generación.

Se ha seleccionado un tamaño de población inicial (N) de **60** candidatos. Este tamaño se mantiene para el resto de las poblaciones de las siguientes generaciones.

4.2.3. Función de selección

La función de selección es la encargada de elegir los candidatos que serán utilizados para engendrar una nueva generación. Existen muchas formas de seleccionar estos candidatos (de manera aleatoria, los k mejores, los k peores, etc.). En este caso se ha elegido utilizar la llamada "selección por torneo". Este tipo de selección consiste en elegir de manera aleatoria un número determinado de candidatos de entre la población inicial para "participar en el torneo", el ganador del torneo es el candidato con mejor fitness de entre los participantes, el cual es seleccionado para engendrar la nueva generación. El torneo se repite hasta que se han elegido ganadores suficientes como para formar una nueva generación.

Se ha elegido este tipo de selección ya que, por una parte selecciona a aquellos candidatos de la generación actual que mejor fitness han conseguido (quedándote con los mejores), mientras que permite que candidatos menos buenos puedan ser también elegidos como ganadores (por ejemplo, si los participantes elegidos aleatoriamente son aquellos con peores valores de fitness). El permitir que no solamente los candidatos mejores engendren dota de cierta riqueza al algoritmo genético, permitiendo explorar un mayor abanico de posibilidades que, quedandonos solo con los mejores, no sería posible explorar.

En nuestro algoritmo genético, el número de participantes en los torneos se ha fijado en **10** candidatos.

4.2.4. Función de recombinación

La función de recombinación o *crossover* se encarga de crear nuevos individuos a partir de los candidatos seleccionados con la función de selección. En nuestro caso, esta recombinación se aplica con una probabilidad del **50 %**. En ella se eligen dos candidatos aleatorios de la nueva generación y se aplica el llamado "*crossover* uniforme". En este tipo de recombinación, para cada componente del vector de componentes, los dos candidatos elegidos intercambian sus valores con una cierta probabilidad. El resultado son dos nuevos candidatos formados por componentes de los elegidos.

En nuestro caso, la probabilidad de que los dos candidatos elegidos intercambien el valor de una componente es del **0.5**.

4.2.5. Función de mutación

La función de mutación se encarga de aplicar un cierto nivel de aleatoriedad en los candidatos, dando lugar a una mayor riqueza de individuos y una mayor amplitud de resultados posibles. En nuestro caso se ha optado por una mutación básica, la cual

se da con una probabilidad del **50 %**. La mutación consiste en recorrer el vector de componentes del candidato elegido para mutar, generando un nuevo valor (dentro de sus rangos establecidos) aleatorio para cada componente con una cierta probabilidad.

En nuestro caso, la probabilidad de generar un nuevo valor para una componente real es del **0.5**, mientras que para una componente entera es del **0.1**.

4.3. Simetría

Las redes CTRNN que actúan como controladores de los agentes cuentan con simetría en las ganancias de los motores y los sensores. Esto quiere decir que ambos motores comparten el mismo valor para la ganancia y que ambos sensores comparten el mismo valor para la ganancia.

4.4. Parametros y rangos

Para todos los agentes anteriormente descritos, los rangos establecidos para sus variables internas pueden verse en la tabla 4.1.

Parametro	**Descripción**	**Valor mínimo**	**Valor máximo**
τ_i	Constante de decaimiento	0.4	4.0
θi	Bias	-3.0	3.0
$Gain$	Ganancia motora y sensora	0.1	10.0
w_{ij}	Peso de la conexión sináptica	-10.0	10.0
n_{ij}	Ritmo de plasticidad	-0.9	0.9

Tabla 4.1: Rango de valores para los parámetros de la red

Capítulo 5

Experimentos

Capítulo 6

Conclusiones

Capítulo 7

Bibliografía

[1] Di Paolo. Homeostatic adaptation to inversion of the visual, field and other sensorimotor disruptions. 2000.

[2] Di Paolo. Organismically-inspired robotics: homeostatic adaptation and teleology beyond the closed sensorimotor loop. 2003.

[3] Di Paolo. Evolving robust robots using fast homeostatic oscillators. 2002.

[4] Mathayomchan and R. D. Beer. Centre-crossing recurrent neural networks for the evolution of rhythmic behaviour. 2002.

[5] Hoinville and Henaff. Evolving plastic neural controllers stabilized by homeostatic mechanisms for adaptation to a perturbation. 2004.

[6] G. G. Turrigiano. Homeostatic plasticity in neuronal networks: The more things change, the more they stay the same. *Trends Neuroscience*, 21:221–227, 1990.

[7] Jesper Blynel and Dario Floreano. Levels of dynamics and adaptive behavior in evolutionary neural controllers. *In B. Hallam, D. Floreano, J. Hallam, G. Hayes, and J.-A. Meyer, editors. Recompilation of papers*, 2002.

[8] K. Funahashi and Y. Nakamura. Aproximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1996.

[9] Nolfi S. and D. Floreano. Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. *MA: MIT Press/Bradford Books*, 2000.

[10] R. D. Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 3(2):459–509, 1995a.

[11] Hywel Thomas Parker Williams. Homeostatic adaptive networks. pages 26–27, 2016.

[12] Jenna Carr. An introduction to genetic algorithms. 2014.

Anexos