2018

MEJORAMIENTO EN APRENDIZAJE DE MOVIMIENTOS EN ROBOTS A PARTIR DE UNA RED t-HYPERNEAT

REYES ROBLES, PABLO GABRIEL

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE

"MEJORAMIENTO EN APRENDIZAJE DE MOVIMIENTOS EN ROBOTS A PARTIR DE UNA RED τ-HYPERNEAT"

PABLO GABRIEL REYES ROBLES

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO, MENCIÓN COMPUTADORES

PROFESOR GUIA: **MARÍA JOSÉ ESCOBAR**

PROFESOR CORREFERENTE: **GONZALO CARVAJAL BARRERA**

Agradecimientos

Quiero agradecer y dedicar este proyecto de titulación a cada persona que me acompañó en mi proceso universitario, principalmente a mi familia, amigos y mi pareja.

Agradezco de forma especial al equipo AIS RoboCup USM y sus integrantes, que han sido como una familia para mí durante los últimos años de carrera.

Resumen

Robots de extremidades móviles son recurrentes en temas de investigación relacionados a la robótica bio-inspirada e inteligencia artificial, teniendo como objetivo final poder otorgar autonomía en sus movimientos y acciones. En este proyecto de titulación se ha investigado sobre métodos de neuroevolución aplicados para la generación de caminatas en robots con extremidades móviles, específicamente el algoritmo Hipercubo basado en el Aumento de Topologías (HyperNEAT), que permite la evolución de redes de gran escala usando regularidades geométricas para un problema descrito haciendo uso del algoritmo Neuroevolución basado en el Aumento de Topologías (NEAT). A partir de HyperNEAT se desarrolló τ-HyperNEAT que agrega retardos de tiempo en la excitación de las señales en el substrato de la red. Ambos algoritmos han sido utilizado para la generación de caminatas en robots móviles, viendo en τ-HyperNEAT movimientos más naturales y complejos. En base a los resultados conocidos en trabajos previos esta memoria propone la implementación del método de Evolución de substrato en una red HyperNEAT (ES-HyperNEAT) que permite la generación de substratos que no posean topologías fijas a partir de una red NEAT. El propósito de este algoritmo pretende buscar estructuras de red ideales para la realización de caminatas en robots con extremidades móviles y evitar la información redundante en un substrato con nodos posicionados de forma manual. Finalmente los resultados de los entrenamientos utilizando ES-HyperNEAT demuestran una gran diversidad en los movimientos aprendidos por el robot alcanzando la solución al problema en menor cantidad de iteraciones que en su predecesor.

Palabras Clave:

Red neuronal artificial, Neuroevolución, HyperNEAT, τ-HyperNEAT, ES-HyperNEAT.

Abstract

Mobile limb robots are recurrent in research topics related to bio-inspired robotics and artificial intelligence, with the ultimate goal of granting autonomy in their movements and actions. In this degree project we have investigated neuroevolution methods applied to the gait generation in robots with mobile limbs, specifically the algorithm Hypercube based NEAT (HyperNEAT), which allows the evolution of large-scale networks using geometric regularities for a problem described using the Neuroevolution algorithm based on the Augmenting Topologies (NEAT). As HyperNEAT, τ-HyperNEAT was developed to add time delays in the excitation of the signals in the network substrate. Both algorithms have been used for the generation of gaits in mobile robots, seeing in τ-HyperNEAT more natural and complex movements. Based on the results known in previous works, this project proposes the implementation of the Evolvable substrate method in a HyperNEAT network (ES-HyperNEAT) that allows the generation of substrates that do not have fixed topologies from a NEAT network. The purpose of this algorithm is to search for ideal network structures for gait learning in robots with moving limbs and avoid redundant information in a substrate with manually positioned nodes. Finally the results of the training using ES-HyperNEAT demonstrate a great diversity in the movements learned by the robot reaching the solution to the problem in fewer iterations than in its predecessor.

Keywords:

Artificial neural network, neuroevolution, HyperNEAT, τ-HyperNEAT, ES-HyperNEAT.

Glosario

AI Artificial Inteligence

GA Genetic Algorithms

ANN Artificial Neural Network

CPPN Compositional Pattern Producing Networks

NEAT NeuroEvolution of Augmenting Topologies

HyperNEAT Hypercube-based Neuroevolution of Augmenting Topologies

ES-HyperNEAT Evolvable substrate in Hypercube-based Neuroevolution of Augmenting Topologies

Índice general

1. INTRODUCCIÓN Y OBJETIVOS

El proyecto “Mejoramiento en aprendizaje de movimientos en robots a partir de una red τ-HyperNEAT” tal como dice su título, pretende encontrar mejores resultados en términos de tiempo y desempeño en rutinas de aprendizaje de movimientos haciendo uso de algoritmos neuroevolutivos, específicamente HyperNEAT. Previo a este trabajo fue implementado τ-HyperNEAT[5] que incorpora conceptos temporales al substrato de la red, que como resultado se obtuvieron caminatas con movimientos más naturales, pero en términos de desempeño fueron muy similares. Los robots que fueron sometidos a entrenamientos tienen diferentes grados de libertad y estructura (fig. 1.1), pudiendo obtener resultados satisfactorios en ambos.

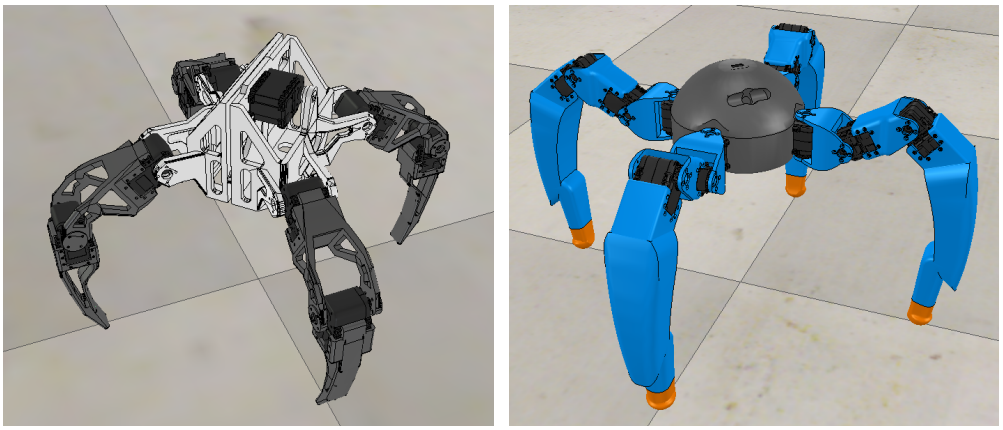
Como método para obtener los resultados deseados en este proyecto se implementará ES-HyperNEAT que construye un substrato a partir de una red NEAT, pero tomando en cuenta las conexiones salientes de los nodos de entrada y las conexiones entrantes en nodos de salida, pudiendo determinar en éstas información significativa o redundante para generar el substrato empleado en los ejercicios.

1.1. OBJETIVOS DEL PROYECTO

Objetivo 1 Codificar la implementación del algoritmo ES-HyperNEAT

Objetivo 2 Realizar entrenamientos de caminatas en robots con extremidades móviles haciendo uso de ES-HyperNEAT

Objetivo 3 Detectar diferencias y concluir resultados a partir de la comparación de las caminatas obtenidas entre diferentes técnicas de redes neuroevolutivas basa-



(a) Quadratot *(b)* ArgoV2

Fig. 1.1: **Robots sometidos a entrenamientos** Quadratot y ArgoV2, 9 y 12 grados de libertad respectivamente

das en HyperNEAT

1.2. TRABAJOS A DESARROLLAR Y RESULTADOS ESPERADOS

El trabajo a desarrollar en este proyecto se sustenta en resultados obtenidos de estudios previos sobre la generación de caminatas en robots con extremidades móviles a partir de métodos neuroevolutivos, específicamente HyperNEAT y la introducción de τ-HyperNEAT como proyección futura en los métodos de aprendizaje utilizados. Si bien la aplicación de ambos algoritmos en la búsqueda de soluciones al problema se han traducido en resultados exitosos, presentan un amplio espacio de mejora en su metodología. Aún más estos algoritmos dentro del estudio de la AI sostienen en su fundamento teórico la aplicación de GA en la evolución de las ANN obtenidas con ellos, siendo la principal característica el aumento de las topologías presentes en ellas a medida que progresan los entrenamientos. Tanto HyperNEAT como τ-HyperNEAT construyen un substrato de estructura fija a partir de NEAT, que complejiza la topología de las ANN que evoluciona desde de la estructura más simple posible, es por esto que en la búsqueda de encontrar soluciones ideales para la infinidad de esce-

narios posibles tratando de llevar un poco más lejos el fundamento de la evolución de topologías, es que en este proyecto se implementarán características evolutivas en la estructura del substrato. A partir de esto es que se propone la codificación de ES-HyperNEAT, una nueva versión de HyperNEAT que permite substratos variables respecto a una población de redes neuronales evolucionadas haciendo uso de NEAT y CPPN.

Este trabajo idealmente debiera consistir en agregar una nueva capa de procesamiento al algoritmo HyperNEAT ya existente, sin embargo las implementaciones conocidas y utilizadas en los ejercicios previos no ofrecen la compatibilidad necesaria para esta labor, por lo tanto se debe hacer una revisión a la codificación de HyperNEAT para hallar los posibles problemas que presentaría la construcción de un substrato dinámico y de qué forma resolverlos, luego de esto es posible realizar la codificación del algoritmo ES-HyperNEAT. Para comprobar el correcto funcionamiento del algoritmo se somete a ejercicios simples y conocidos como el OR exclusivo (XOR), de forma que la población de redes neuronales encuentren la solución al problema y converjan de forma promedio.

En HyperNEAT se definen los pesos de las conexiones en el substrato a través de la salida de una ANN, cuyas entradas son las coordenadas de los nodos de origen y destino. Esta red es capaz de reconstruir el substrato de forma idéntica en caso de que sea requerido una segunda o más veces. De manera similar en ES-HyperNEAT la estructura y los pesos del substrato son definidos por una ANN, que de usar una red igual o idéntica será posible reconstruir un substrato de características exactas al anterior. Esto nos indica que sólo es necesario exportar como archivo la información de la red CPPN-NEAT, pero además nos indica que el método evolutivo aplicado es clave para encontrar la solución al problema y lograr que la población de redes neuronales converja a ella, es por esto que junto con la implementación de ES-HyperNEAT se propone una mejora en la codificación del algoritmo NEAT, tanto en la forma en que se estructuran los organismos y especies, así como la adición de características evolutivas que permitan prevalecer los mejores organismos entre generaciones, potenciar

especies jóvenes dentro de la población, lidiar con el estancamiento en la búsqueda de soluciones, entre otros.

Los resultados de caminatas con HyperNEAT, τ-HyperNEAT y ES-HyperNEAT son comparados tanto a través de criterios cuantitativos utilizando funciones de desempeño, como a partir de criterios cualitativos en la observación de los movimientos realizados por los robots, pues una de las grandes características de τ-HyperNEAT es la obtención de caminatas más naturales como solución al sistema. Para apoyar la evaluación de este problema evitando subjetividades se observarán los gráficos de movimientos angulares de los motores de los robots a lo largo del tiempo.

2. ESTADO DEL ARTE DE MÉTODOS EVOLUTIVOS PARA LA CAMINATA DE ROBOTS

Dentro del área de neuroevolución y algoritmos genéticos es importante notar el trabajo del investigador norteamericano Kenneth O. Stanley, destacando su trabajo en [1] donde desarrolló el algoritmo NEAT, Neuroevolución a través del Aumento de Topologías, superando en pruebas específicas a diferentes ANN, sobre todo a aquellas de topologías fijas.

En el año 2007 Stanley se propone el método de codificación CPPN[2] o Compositional Pattern Producing Network que permite generar patrones a partir de un fenotipo codificado como una red de grafos. Las regularidades presentes en los patrones permiten explotar la geometría de un problema a resolver, y dada la similitud entre la red de grafos y las ANN tradicionales es posible aplicar métodos como NEAT para evolucionarlas y encontrar la relación óptima entre las entradas y salidas y los patrones que las relacionen.

Luego Kenneth O. Stanley presenta la evolución de NEAT, HyperNEAT [3] Hipercubo basado en Neuroevolución a través del Aumento de Topologías, que haciendo uso de CPPN, puede producir patrones con repeticiones y simetrías que interpreta el hipercubo, explotando la geometría de la tarea dentro de la topología de la red. En este caso la red que evoluciona es aquella que determina los pesos de las conexiones del hipercubo, cuya topología es fija.

En el año 2014 investigadores de la Universidad de la Coruña proponen una extensión del algoritmo NEAT llamado τ-NEAT [4], como respuesta a la inexistencia de elementos temporales explícitos en el algoritmo, lo que ocasionaba que tareas que

manejen variables temporales no siempre dieran el mejor resultado haciendo uso de NEAT. Es por esto que τ-NEAT permite agregar retardos temporales a las conexiones de la ANN.

Así como en el caso anterior, el ex-alumno de la Universidad Técnica Federico Santa María Óscar Silva Muñoz, presenta una extensión de HyperNEAT llamado τ-HyperNEAT[5], que permite que la ANN que se encarga de la evolución de los pesos de las conexiones en el hipercubo además entregue retardos en ellas. Tanto HyperNEAT y τ-HyperNEAT fueron utilizados para el entrenamiento de caminatas de robots con extremidades móviles donde τ-HyperNEAT demostró entrenar caminatas con movimientos más naturales y armónicos. Los entrenamientos fueron desarrollados en un entorno simulado gracias a una librería presentada en este trabajo llamada RobotLib, que permite conectar el software de simulación con las tareas de aprendizaje.

En el año 2012 Stanley propone una nueva evolución del algoritmo llamado ES-HyperNEAT [6], o Evolve substrate HyperNEAT, que tal como indica su nombre evoluciona el substrato del hipercubo deduciendo la geometría de sus nodos, basado en los patrones de los pesos de sus conexiones usando HyperNEAT, de esta forma se consigue la evolución de la totalidad de los nodos de la red neuronal artificial. Los resultados experimentales arrojan que en ejercicios específicos, como la búsqueda de salidas en laberintos o resolver dos tareas (*dual task*) con una misma red, ES-HyperNEAT supera significativamente a su predecesor.

El software de simulación que se usará para el desarrollo de esta memoria se llama V-REP [8], Virtual Robot Experimentation Plataform, desarrollado por Coppelia Robotics GmbH en Zurich, Suiza, el cual posee versiones tanto pagas como gratuitas. La versión educacional de este software es completamente gratuita y posee todas las funcionalidades del programa completo. Este software posee una API desarrollada en una gran variedad de lenguajes de programación, permitiendo así manejar todos los aspectos del programa y la simulación desde el exterior a través de sockets.

3. FUNDAMENTO TEÓRICO

En este capítulo se introducirá a los métodos neuroevolutivos aplicados para el aprendizaje de caminatas tales como NEAT y redes CPPN, HyperNEAT y su variación τ-HyperNEAT, para finalizar con el fundamento detrás de ES-HyperNEAT que fue implementado para ser utilizado en este proyecto.

3.1. NEAT

Los enfoques tradicionales en neuroevolución escogen una topología o estructura fija de red para realizar sus experimentos, siendo la más usual aquella con una única capa intermedia de neuronas interconectadas cada una con todas las neuronas en las capas de entrada y salida. Una topología más compleja puede ser escogida para una tarea en epecífico, como es el caso de los algoritmos de Aprendizaje profundo o *Deep Learning*, que a diferencia del caso anterior se aprecia una gran cantidad de capas intermedias cuyas conexiones son determinadas por la estructura escogida. Comúnmente estos enfoques tienen como meta permanecer con una topología fija optimizando los pesos de las conexiones que determinarán la funcionalidad de la red. Neuroevolución a través del Aumento de Topologías o NEAT[1], plantea a partir de estudios previos sobre la efectividad de hacer variaciones en la estructura de la red, la evolución de topologías junto con la variación de los pesos de sus conexiones para incrementar el desempeño del sistema.

La primera gran propuesta de NEAT es tomar ventaja de las estructuras de las ANN como forma de minimizar la dimensionalidad del espacio de búsqueda de pesos en la red. Para esto NEAT evoluciona una población de redes neuronales simples sin

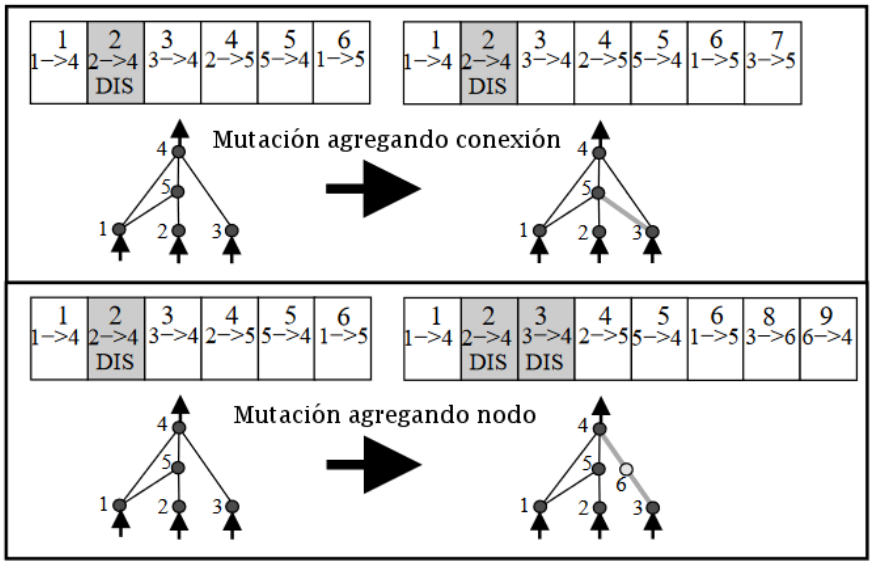


Fig. 3.1: **Mutaciones posibles en los organismos.** Se distinguen dos tipos de mutaciones, las de adición de conexiones y de nodos. En la segunda necesariamente deben crearse dos conexiones para sustituir aquella sobre la que fue posicionado el nuevo nodo. Imagen adaptada de [1].

capas intermedias, es decir los nodos de la capa de entrada interconectados con los nodos de la capa de salida, donde la única diferencia inicial entre ellas son los pesos de sus conexiones, aleatorizados en una primera etapa de inicialización. Posterior a esto y mientras dure el experimento, NEAT complejizará la estructura de las ANN de la población tanto como sea necesario para encontrar la solución al problema, permitiendo la adición de neuronas en la capa intermedia sobre conexiones existentes, las cuales permitirán a su vez adicionar nuevas conexiones en la medida que sea posible. Si lo anterior es bien aplicado las estructuras de las ANN que resuelvan el problema serán óptimas e idealmente minimizadas.

Con sus bases en los algoritmos genéticos NEAT describe la estructura de cada una de las redes de la población en un genotipo donde tanto los nodos de la ANN como sus conexiones son representados como genes, siendo la representación fenotípica la propia red que será usada para los cálculos correspondientes. La aplicación de esta metodología es conocida en métodos neuroevolutivos que hacen uso de algoritmos genéticos, sin embargo NEAT incorpora marcas históricas a cada nuevo gen para po-

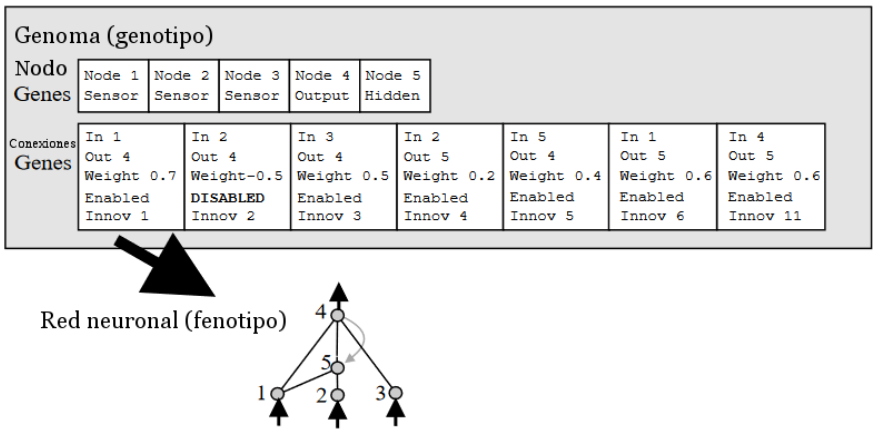


Fig. 3.2: **Repesentación de un fenotipo a partir de un genotipo.** Se mapean los genes para obtener una red neuronal a partir de su definición. Se observan 3 nodos de entrada, 1 nodo intermedio y 1 de salida con recurrencia hacia el nodo intermedio. Se observa en el genotipo toda la información de entrada y salida, los pesos de conexiones, si es que el gen se desactiva para el organismo y una marca histórica de innovación. Imagen adaptada de [1].

der realizar comparaciones entre individuos y establecer las reglas de reproducción. Cada elemento que es agregado al genotipo se le asigna un identificador global de innovación que se incrementa con cada nuevo gen que aparece, por lo que al avanzar en generaciones los organismos mutan o se reproducen con otros heredando el genotipo con el identificador de innovación en cada uno de sus genes, que ayuda reducir el análisis topológico entre individuos facilitando la detección de compatibilidad entre ellos y a su vez, la reproducción entre organismos.

El identificador de innovación como fue definido en el párrafo anterior ofrece la detección de compatibilidades entre individuos a partir de los rasgos evolutivos que presentan, NEAT hace uso de esta información para clasificar a la población en especies permitiendo que los individuos de una misma especie compitan entre ellos, dándoles tiempo de optimizar sus estructuras antes de competir con el resto de la población de ANN.

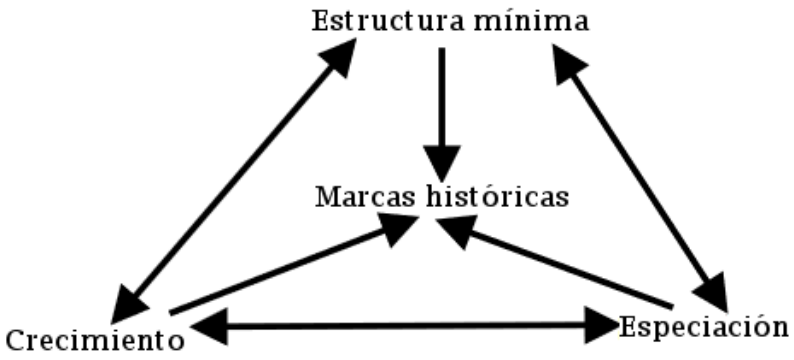


Fig. 3.3: **Dependencias del algoritmo NEAT.** Esquema que representa el cómo se relacionan los elementos en la neuroevolución de los organismos. Imagen adaptada de [1].

3.2. CPPN

En organismos naturales pueden apreciarse estructuras complejas que son codificadas gracias a un reducido número y eficiente uso de genes presentes en el genotipo del individuo. Haciendo uso además de las condiciones ambientales del organismo es posible obtener un fenotipo cuya estructura permite describir al individuo como tal. Sin embargo la naturaleza hace uso de más herramientas para poder ser eficiente en su codificación pudiendo identificar entre ellas el uso de patrones en el fenotipo que resultan en una gran cantidad de regularidades, tal como se puede observar en la naturaleza en general.

CPPN[2] es un método de codificación que permite hacer uso de relaciones estructurales de un organismo a través de una variedad de funciones. Para entender qué funciones y cómo utilizarlas en una estructura de red se identifican regularidades conocidas en la naturaleza de manera que puedan ser aplicados en fenotipos generados artificialmente:

- **Repetición:** Subestructuras repetidas en el organismo, como por ejemplo las células y neuronas.
- **Repetición con variaciones:** Frecuentemente estructuras se encuentran repeti-

das pero no de forma completamente idéntica. Por ejemplo en las vertebras de una columna o en los mismos dedos de una mano, cada una de sus componentes posee la misma estructura pero con distintas variaciones.

- **Simetría:** A menudo las repeticiones ocurren a través de las simetrías, por ejemplo los lados derecho e izquierdo del cuerpo humano, corresponden a una simetría bilateral.

- **Simetría imperfecta:** Normalmente las estructuras biológicas, muchas veces no son perfectamente simétricas. La simetría imperfecta es una característica común de repetición con variaciones, por ejemplo el cuerpo humano es simétrico en general, pero no es equitativo en ambos lados, como la aparición de órganos en diferentes lugares del cuerpo o un lado es generalmente dominante sobre el otro (diestros, zurdos).

- **Regularidades elaboradas:** A lo largo del tiempo hay regularidades que son más usadas y explotadas de forma que comienzan a recibir nuevos usos, como las aletas de los peces que terminaron por evolucionar en las extremidades de los nuevos mamíferos.

- **Preservación de regularidades:** Hay ciertas regularidades que son preservadas a medida que avanzan las generaciones y se transmiten a la descendencia, como el número de extremidades en animales cuadrúpedos.

Como ya fue mencionado los genes son codificados en un genotipo, describiendo un fenotipo que en la naturaleza representará una infinidad de regularidades observables, y son agrupados en los grupos previamente descritos, todo esto realizado a partir de la codificación eficiente de una pequeña cantidad de genes. Por lo tanto la naturaleza debe hallar la forma de complejizar los sistemas que se encuentran en constante evolución, y una forma de lograr esto es disminuir lo más posible la dimensionalidad de la solución al problema, pues mientras más grande sea más difícil será encontrar

una solución adecuada, es decir encontrar un fenotipo que pueda describir estructuras complejas con el genotipo de menor dimensión posible. La evolución supera este problema iniciando su espacio de búsqueda con un tamaño muy pequeño, pudiendo añadir nuevos genes al genoma y así agregando un nuevo nivel a su dimensión pero evitándonos comenzar la búsqueda de la solución final a partir de una dimensionalidad mayor y posiblemente, superior al óptimo.

La adición de genes al genoma se traduce automáticamente en el incremento de la dimensionalidad del espacio de soluciones, expresado fenotípicamente en el caso de neuroevolución basado en algoritmos genéticos corresponde a un nuevo nodo o conexión en la red neuronal a evaluar, y en este caso puede representar patrones de salida en la red extremadamente diversos dependiendo de la información del gen expresado en el fenotipo, como el peso de una conexión o como se verá en un par de secciones más adelante, la función de activación de un nodo adicionado.

Comienza a existir el cuestionamiento de cómo llevar a cabo la codificación de un fenotipo descrito por un genotipo, siendo una buena aproximación pensarlo como una distribución de puntos a lo largo de un espacio multidimensional, es decir el fenotipo puede ser descrito como una función

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

donde n indica la dimensionalidad del espacio físico. En la fig. 3.4 se observa una función con dos parámetros de entrada que describen puntos sobre un plano, esto es la ejemplificación de un fenotipo bidimensional e indica que la función por la cuál es sometido el espacio cartesiano de la figura es la encargada de producir un patrón una vez todos los puntos sean dibujados.

Si se toma la idea de la fig. 3.4 y la función que describe la distribución de puntos fuese una función lineal, se observaría que a medida que avanzamos en dirección positiva por el eje de las abscisas la concentración de puntos incrementaría de forma lineal sobre el eje de las ordenadas, si la representación del fenotipo en el plano

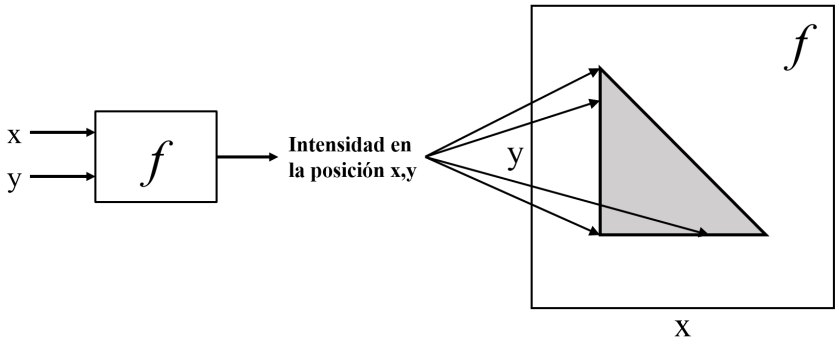


Fig. 3.4: **Una función produce un fenotipo.** Un espacio bidimensional es el dominio de una función o en este caso un fenotipo, cuya dimensionalidad de la solución se encuentra en $\mathbb{R}$. El resultado es una figura que es descrita por el fenotipo. Imagen adaptada de [2].

cartesiano tuviera repeticiones de puntos en posiciones equidistantes podría significar que la función utilizada por este fenotipo es del tipo sinusoidal, o si la representación de puntos tuviese una concentración acampanada es decir baja cantidad en los extremos y donde es apreciable una simetría bilateral, podría tratarse de una función Gaussiana. El hacer una composición de funciones, donde el recorrido de funciones correspondan al dominio de otras, genera una infinidad de soluciones y dependiendo de las funciones utilizadas puede obtenerse patrones bastante diversos, por ejemplo una simetría bilateral sobre el eje de las ordenadas con grupos de segmentos periódicos cuya orientación es opuesta, puede ser descrito por la composición de una función periódica cuya entrada es la salida de una función simétrica, como se observa en la fig. 3.5.

La composición de funciones puede ser representada como un grafo, con las conexiones de entradas y salidas correspondientes a cada una. Como se observa en la fig. 3.6 las funciones toman la labor de nodos en la topología descrita, donde los primeros nodos reciben las coordenadas de un sistema coordenado y sus salidas corresponden a las entradas de otros nodos de niveles superiores, hasta llegar a la salida del sistema que entregará la distribución de puntos en un plano cartesiano. En este punto es esperable que la salida no sea una figura geométrica conocida o la representación de

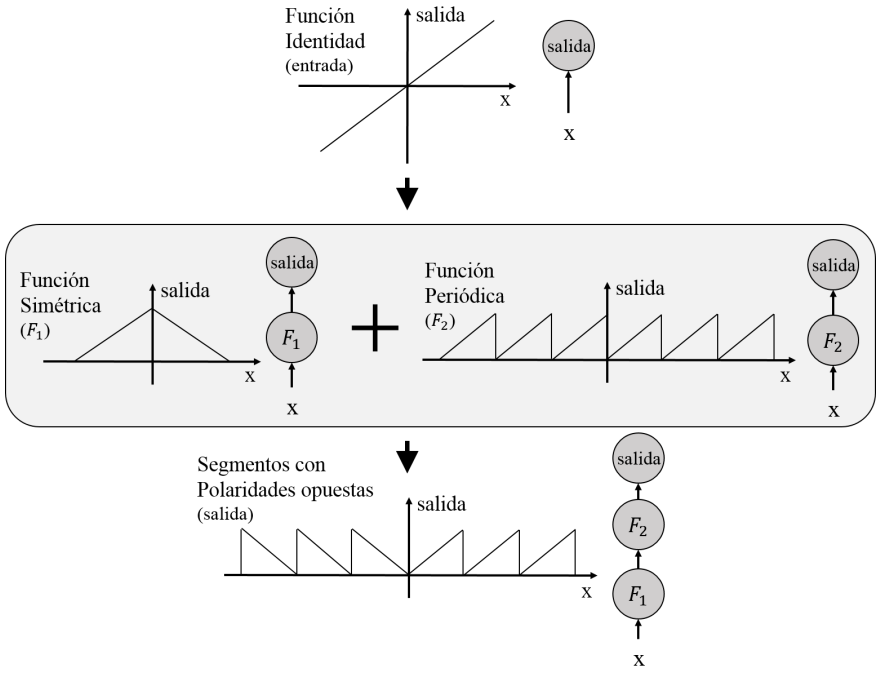


Fig. 3.5: **Composición de funciones.** Simetría bilateral sobre el eje de las ordenadas a partir de la comosición de una función simétrica y otra periódica. Imagen adaptada de [2].

una función lineal sino que la distribución de puntos responda a patrones claros con regularidades como simetrías y repeticiones a lo largo de todo el espacio definido por el dominio.

A lo largo de esta sección no se ha hecho más que describir lo que es "Compositional Pattern Producing Networks" (CPPNs) y cómo su método de codificación permite describir directamente relaciones estructurales de una topología a través de una composición de funciones.

Observando nuevamente la fig. 3.6 es posible encontrar similitudes entre el grafo y una ANN, donde la única diferencia entre una ANN tradicional es que las funciones de activación en sus nodos suele ser una función sigmoide, pudiendo entender a las CPPNs como una composición de funciones que es capaz de producir patrones regulares.

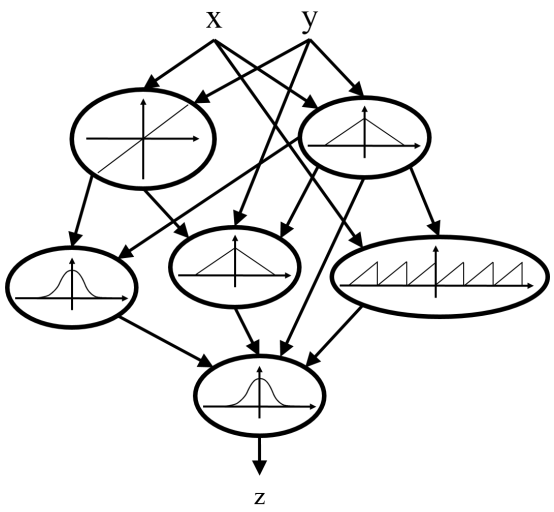


Fig. 3.6: **Composición de funciones en forma de grafo.** Grafo de conexión entre la coordenada (x, y) y la salida z. A cada conexión se le asigna un peso, el cual multiplica a la salida de la función del nodo entrante. Si múltiples conexiones entran a una misma función, esta recibe como entrada la suma de las salidas de cada una de las funciones entrantes multiplicadas por sus respectivos pesos de conexión. Se debe tener en cuenta que la topología no tiene restricciones y puede representar cualquier relación posible. Esta representación es similar al formalismo de las ANNs tradicionales con funciones de activación y topologías arbitrarias. Imagen adaptada de [2].

3.3. CPPN-NEAT

Desde un punto de vista estructural es posible establecer similitudes entre entre las redes CPPN y las ANN, por lo que sería posible aplicar algoritmos neuroevolutivos como NEAT sobre redes CPPN con algunas consideraciones especiales.

1. Tanto en NEAT como en los métodos neuroevolutivos tradicionales los nodos de capa intermedia de las ANN de una población sólo activan sus neuronas con funciones sigmoides, mientras que CPPN-NEAT crea nodos asignando aleatoriamente su función de activación a partir de un grupo determinado en la implementación del método, tales como funciones gaussianas, funciones periódicas o sigmoides.

2. En NEAT es necesario determinar la compatibilidad de organismos de una población para poder agruparlos en especies, por lo tanto al incorporar la aleato-

riedad en la selección de funciones de activación sobre las neuronas es necesario modificar el criterio por el cuál se determina las similitudes genotípicas que permitan determinar si pertenecen o no a una misma especie, de esta forma CPPN-NEAT considera por cuántas funciones de activación difieren los individuos además de los criterios ya aplicados en NEAT.

CPPN-NEAT incorpora elementos que no son comunes en los métodos tradicionales en neuroevolución, aprovechando el aumento de topologías en redes neuronales que NEAT ofrece. Además se incorpora la posibilidad de complejizar redes neuronales haciendo uso de funciones de activación diferentes en sus neuronas, esto permitiría dibujar a partir de la salida de una red una distribución de puntos con variadas características, concentrando el estudio en la búsqueda de patrones con regularidades visibles.

3.4. HYPERNEAT

En la sección anterior se describió como evolucionando redes CPPN con NEAT es posible obtener patrones en la salida del sistema, el problema siguiente corresponde a interpretar de forma correcta las regularidades obtenidas. HyperNEAT[3] aprovecha los patrones espaciales que puede generar las redes CPPN para mapear patrones de conectividad de una nueva red la que podrá resolver problemas desde su propia dimensionalidad.

A partir de la idea obtenida de la fig. 3.4 donde dos puntos de un plano bidimensional funcionaban como entrada a una función cuya salida representaba una distribución de puntos en el mismo plano cartesiano, se planteó la posibilidad de hacer composiciones de distintas funciones para obtener patrones en la distribución de puntos a lo largo del dominio del problema. Estos patrones espaciales pueden ser usados discretamente para definir patrones de conectividad en una nueva red, pudiendo explotar la geometría de una estructura propia del problema a resolver. Específicamente la idea consiste en usar como entradas de una red CPPN las coordenadas de un par de puntos

de una nueva red que describan una posible conexión entre ellos, siendo la salida de la red el peso de ella, lo anterior significa que a la hora de evolucionar esta nueva red se estará tomando en cuenta la geometría del problema a resolver. Es decir la red CPPN puede ser descrita como una función

$$\text{CPPN} : \mathbb{R}^4 \longrightarrow \mathbb{R}$$

cuya ecuación corresponde a

$$\text{CPPN}(x_i, y_i, x_j, y_j) = \begin{cases} \rho_{i,j} & \text{si } |\rho_{i,j}| > \rho_{min} \\ 0 & \text{e.o.c} \end{cases} \tag{3.1}$$

Como indica la ecuación [3.1] CPPN recibe de entrada las coordenadas (x_i, y_i) y (x_j, y_j) de los nodos i y j a conectar, siendo la salida de la función el peso $\rho_{i,j}$ de todas las conexiones posibles del sistema. Tal como es visto en otros métodos neuroevolutivos la conexión no se realiza si el valor absoluto del peso $\rho_{i,j}$ es menor al umbral mínimo ρ_{min} determinado. En caso de realizarse la conexión el peso es escalado entre cero y la magnitud máxima de la conexión ω_{max}, pudiendo dar versatilidad a la topología y sus valores numéricos en las salidas. La ecuación [3.2] muestra lo anterior considerando que los valores de $\rho_{i,j}$ han sido previamente normalizados entre $[-1, 1]$.

$$\omega_{i,j} = \omega_{max} \cdot \rho_{i,j} \tag{3.2}$$

Para ejemplificar lo anterior se considera un espacio bidimensional denominado de aquí en adelante como **substrato**, de 5×5 nodos, donde el nodo central del substrato corresponde a la coordenada $(0, 0)$ y el resto de los nodos se encuentran equiespaciados arbitrariamente por $0{,}5$ de unidad en cada eje. Luego las coordenadas de cada uno de estos nodos son ingresadas a una red CPPN y retorna el peso de la conexión. Las conexiones son dependientes de las posiciones de los nodos en el substrato por lo tanto la distribución de puntos en la salida de la red CPPN es dependiente de la geometría asignada al substrato, y posiblemente se obtendrán patrones regulares como

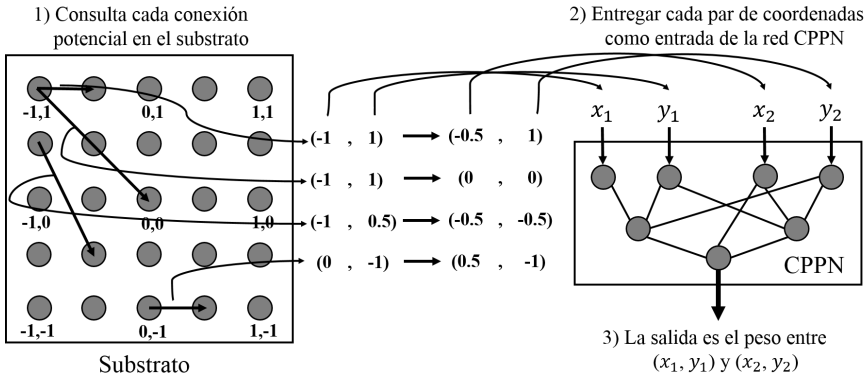
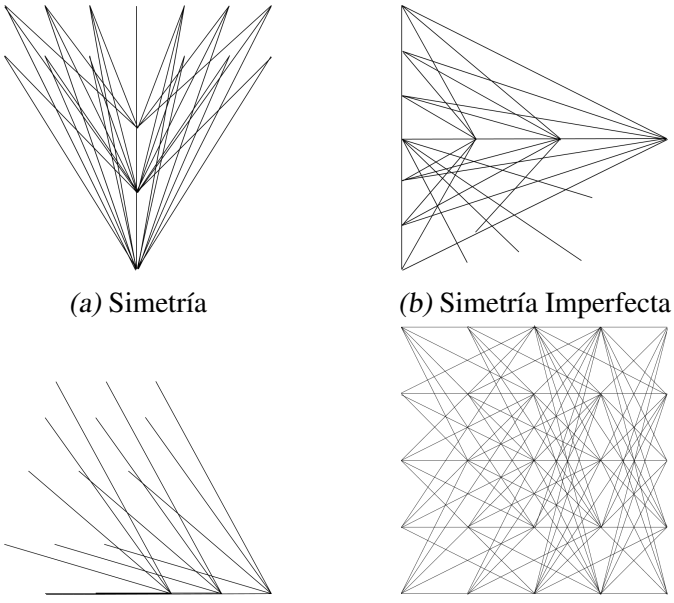


Fig. 3.7: **Interpretación del patrón geométrico de conectividad de un hipercubo.** La malla de nodos de 5×5 mejor llamada substrato, posee una serie de puntos cuyas coordenadas en un espacio bidimensional son ingresadas a una red CPPN en forma de pares para poder determinar una posible conexión entre ambos. La salida de la red CPPN corresponde al peso de la hipotética conexión a realizar, siguiendo las restricciones de un proceso neuroevolutivo.

los descritos en la sección 3.2. Gráficamente este ejercicio está plasmado en fig. 3.7.

Es esperable que el patrón de conectividad sea consecuencia del patrón espacial generado por la red CPPN, pudiendo encontrar un isomorfismo explicado a través de la dimensionalidad de ambos sistemas. Es posible descubrir subestructuras una vez son mapeadas las conexiones obtenidas por la red CPPN, por ejemplo es posible encontrar simetrías a lo largo en la conexión del substrato al aplicar funciones de activación Gaussianas en la red de entrenamiento, o repeticiones en el caso de hacer uso de funciones de activación de carácter sinusoidal. Un ejemplo de esto puede observarse en la fig. 3.8.

Una red CPPN puede producir una diversidad de patrones a partir de la composición de sus funciones. En la fig. 3.8 se muestran patrones de conectividad que han sido producidos a partir de las salidos a una red CPPN, donde se observa patrones ya mencionados como el simétrico en la fig. 3.8a, o una simetría imperfecta en la fig. 3.8b donde las funciones de activación de la red podrían estar compuestas por una función gaussiana y alguna que posea un comportamiento asimétrico. También es posible observar patrones con repeticiones en la fig. 3.8c, que puede haber sido obtenido por alguna función de tipo periódica o sinusoidal en la composición de la red CPPN,



(a) Simetría *(b)* Simetría Imperfecta

(c) Repetición *(d)* Repetición con Variaciones

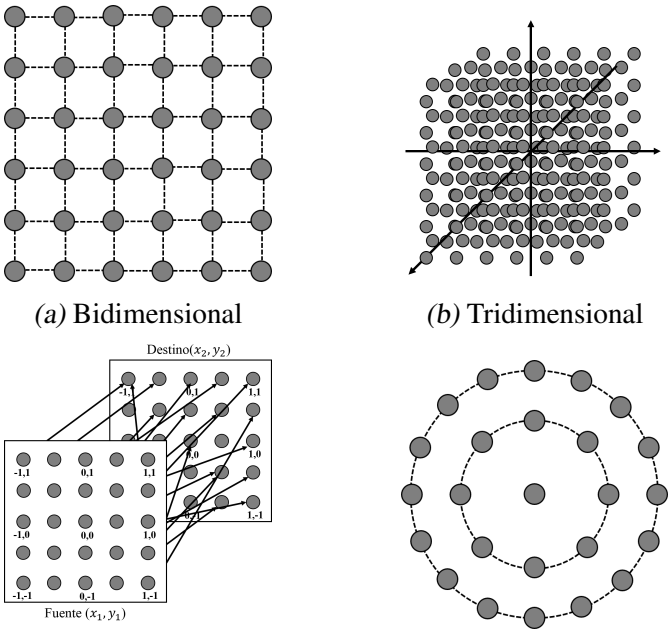
Fig. 3.8: **Patrones de conectividad a partir de redes CPPN.** A partir de neuroevolución es posible obtener patrones de conectividad con regularidades, tal como fueron descritas en la sección 3.2: (a) simetría bilateral, (b) simetría imperfecta, (c) repetición, y (d) repetición con variaciones.

o el patrón con repeticiones con variaciones de la fig. 3.8d pudiendo ser generada por una función periódica adicionada a una función cuya respuesta no posea repeticiones.

La configuración de nodos en el substrato de HyperNEAT no necesariamente es un plano de dos dimensiones, dependiendo del tipo de problema a abordar pueden adoptarse diferentes estructuras que se adecuen de mejor forma a la geometría del caso.

Una red CPPN podría producir patrones de conectividad tridimensionales como en la fig. 3.9b, si está definida a partir de

$$\text{CPPN} : \mathbb{R}^6 \longrightarrow \mathbb{R}$$



(a) Bidimensional *(b)* Tridimensional

(c) Capas *(d)* Circular

Fig. 3.9: **Alternativas de configuración del Substrato.** (a) configuración en un plano bidimensional, (b) una configuración tridimensional de nodos centrados en (0,0,0), (c) una configuración de capas donde las conexiones se ralizan entre nodos desde la capa de origan a la de destino, y (d) una configuración circular. Dependiendo de la geometría del problema pueden ser escogido algún substrato en específico

es decir se trata de un hipercubo de seis dimensiones cuya ecuación corresponde a

$$\begin{aligned} \text{CPPN}(\overrightarrow{P_i}, \overrightarrow{P_j}) &= \rho_{i,j} \\ \overrightarrow{P_i} &= (x_i, y_i, z_i) \\ \overrightarrow{P_j} &= (x_j, y_j, z_j) \end{aligned} \tag{3.3}$$

Es posible restringir la configuración de substrato para explotar los patrones de conectividad a topologías en específico, como es el caso de la configuración por capas interconectadas o *"state-space sandwich"*[9]. Tal como sucedió en el caso de las conexiones bidimensionales (fig. 3.9a) el patrón de conectividad es descrito por una función que recibe las coordenadas de dos puntos en la entrada, cuyo resultado es el peso de la conexión a establecer tal como se observó en la fig. 3.7, sin embargo los nodos se encuentran en planos bidimensionales separados, es decir se compone un

sistema de capas (fig. 3.9c) donde la principal restricción es que la dirección de las conexiones a establecer es única, por lo tanto teniendo

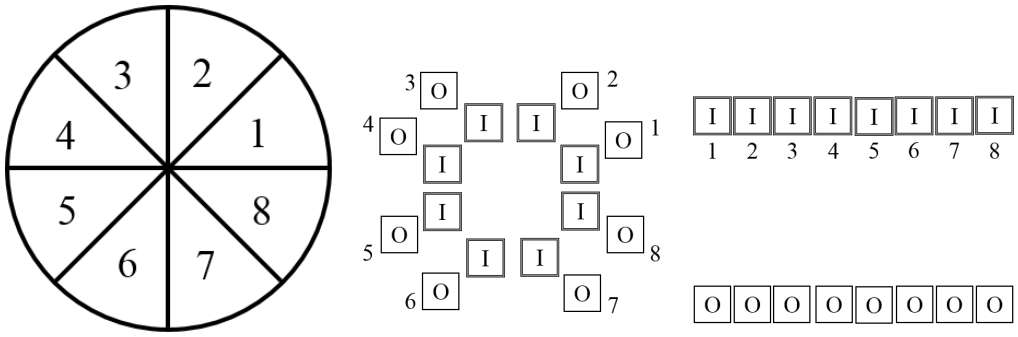
$$\text{CPPN}(x_i, y_i, x_j, y_j)$$

se establece que las coordenadas (x_i, y_i) corresponden a un nodo de la capa de origen y (x_j, y_j) la localización de un nodo de la capa destino, no dando posibilidad a bidireccionalidad. En este ejemplo se trata de un sistema de dos capas, pero en caso de que el problema se extienda a n capas la direccionalidad única debe mantenerse pudiendo complejizar aún más la solución.

La fig. 3.9d muestra el último ejemplo de configuración donde se explota una geometría circular, pudiendo tratarse de un problema específico a evolucionar. La conectividad de este patrón puede responder a la necesidad de evolucionar comportamientos en estructuras que posean distribuciones radiales en sus sensores o actuadores.

Gráficamente se entiende que el substrato de HyperNEAT puede explotar los patrones generados por una red CPPN, sin embargo esto cobra mayor potencialidad al encontrarse con problemas cuyas geometrías puedan ser expresadas en la configuración del substrato, mostrando ventajosos resultados a la hora de aplicar este método neuroevolutivo.

A partir de una disposición coherente de entradas y salidas la red CPPN podrá desarrollar patrones de conectividad en el substrato de la red que respondan a las necesidades del problema. Al hacer uso de NEAT para evolucionar redes CPPN se otorga aplican criterios de selección al proceso evolutivo pues aquellos patrones que terminen siendo ventajosos evolucionarán con mayor prioridad que aquellos que no han dado buenos resultados. Esto abre la posibilidad a diseñar substratos que describan la geometría del problema físico, por ejemplo para entrenar un robot que posee sensores interpretados como nodos de entrada y actuadores interpretados como nodos de salida, es posible describir espacialmente la posición de cada uno de ellos de una forma similar a la geometría física del robot. En la fig. 3.10 se observa cómo



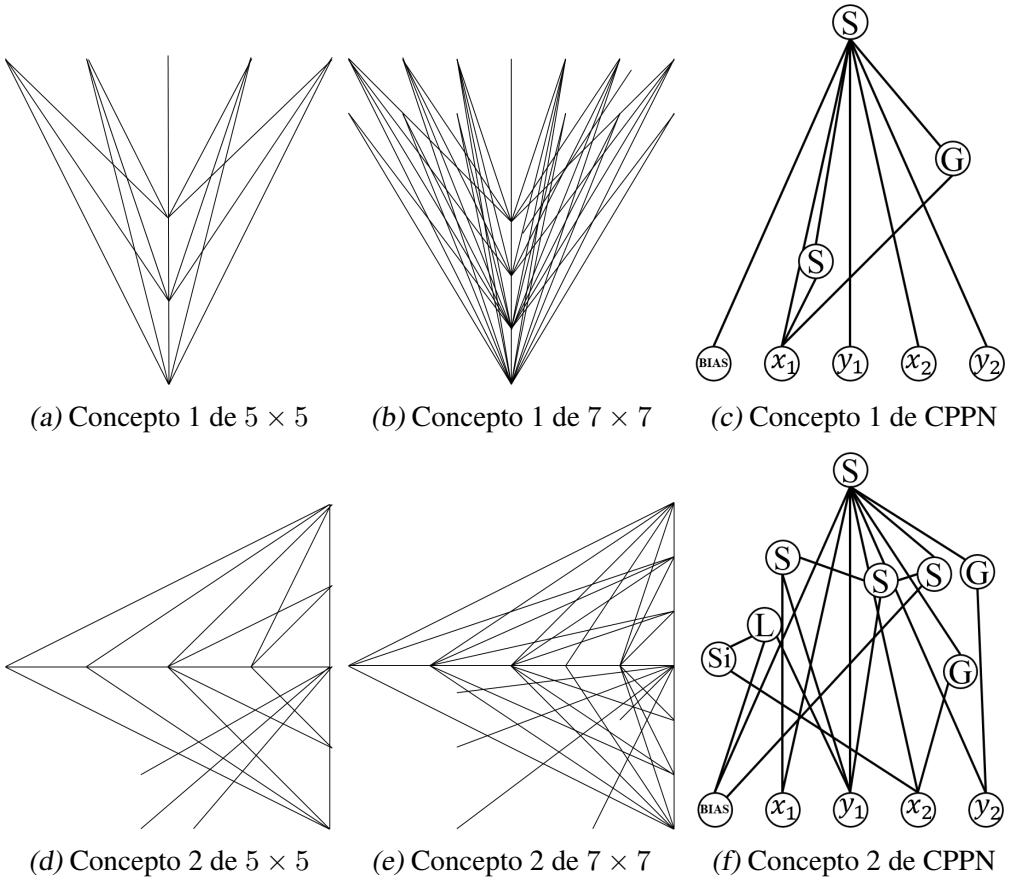
(a) Robot *(b)* Concéntrica *(c)* Paralela

Fig. 3.10: **Entradas y salidas en un substrato.** Un robot circular dividido en ocho secciones que poseen sensores y actuadores en una línea radial (a). En (b) y (c) los sensores representados como entradas ([I]) y los actudores como salidas ([O]), describen dos métodos de distribución de entradas y salidas en el substrato, el primero de forma radial y el segundo posicionando paralelamente las entradas y salidas de cada sección del robot. Ambos métodos permiten explotar la geometría del problema desde el principio del proceso evolutivo.

podrían ser posicionadas entradas y salidas en el substrato de la red HyperNEAT en el entrenamiento de un robot circular (fig. 3.10a), donde sus sensores y actuadores poseen ubicaciones radiales. Se presentan dos soluciones posibles para este problema (figuras 3.10a y 3.10b) para poder dar ventaja a la evolución desde el principio del entrenamiento.

En el arreglo de la fig. 3.10b se observa que las entradas están posicionadas en un círculo centrado en el origen del substrato, mientras que las salidas forman una circunferencia alrededor de las entradas, de forma que si la red CPPN desarrolla estructuras circulares en el patrón de conectividad, éstas pueden crear relaciones interesantes entre los sensores o entradas y actuadores o salidas. En el caso de la fig. 3.10c se observa que las entradas y salidas son posicionadas de forma horizontal y paralelas unas a las otras. Entendiendo que existe una correspondencia entre las entradas y salidas respecto a la posición física del robot, es posible que la red CPPN explote de manera significativa la relación que existe entre sensores y actuadores a partir de la distribución utilizada en el substrato.

La red CPPN es la encargada de determinar la conectividad de los nodos en un substrato, siendo la composición de las funciones de activación y sus relaciones matemáticas las que describirán un patrón en específico en las conexiones del substrato. La red CPPN describirá un concepto de conectividad respecto a su estructura y es posible apreciarlo incluso aún cuando la resolución de un substrato se viera alterada. La fig. 3.11 grafica lo anterior mostrando que a pesar de incrementar la dimensionalidad del substrato es posible encontrar similitudes en su conectividad si se hace uso de la misma red CPPN.



(a) Concepto 1 de 5×5 *(b)* Concepto 1 de 7×7 *(c)* Concepto 1 de CPPN

(d) Concepto 2 de 5×5 *(e)* Concepto 2 de 7×7 *(f)* Concepto 2 de CPPN

Fig. 3.11: **Conectividad equivalente en substratos de distinta dimensión.** (a) y (b) son substratos cuyas conexiones son determinadas por la red CPPN (c). A pesar de tener distintas dimensiones el concepto de conexión se mantiene, sólo ha incrementado la dimensión de puntos a graficar. Sucede de igual forma en los substratos (d) y (e), cuyas conexiones son creadas por la red CPPN (f). Las funciones de activación de las redes CPPN mostradas corresponden a G para funciones Gaussianas; S para Sigmoides; Si para Senos; L para funciones lineales.

3.5. τ-HYPERNEAT

τ-HyperNEAT[5] es una extensión de HyperNEAT que permite aplicar retardos temporales en las conexiones del substrato, generando el patrón de conectividad de la misma forma que en su predecesor, sin embargo en este caso la red CPPN en vez de sólo entregar el peso de la conexión a establecer asigna un nuevo valor correspondiente a un porcentaje de retardo $\tau_{i,j}$ sobre un τ_{max} o retardo máximo. La implementación de lo anterior se aplica por medio de un buffer cuyo largo es proporcional al retardo asignado. La función CPPN está definida por

$$\text{CPPN} : \mathbb{R}^4 \longrightarrow \mathbb{R}^2$$

Y se define en la ecuación [3.4]. La función CPPN recibe como entradas las coordenadas de un nodo de origen (x_i, y_i) y las coordenadas del nodo de destino (x_j, y_j), obteniendo como salida tanto el peso y porcentaje de retardo de la conexión. Si el peso de la conexión supera el umbral mínimo de pesos se escala la conexión y además se establece el retardo correspondiente obtenido, en caso contrario no se establece la conexión. Si la conexión es válida el porcentaje de retardo obtenido $\tau_{i,j}$ es multiplicado por τ_{max} que corresponde a un número entero que señala el tamaño máximo del buffer de retardo, obteniendo el tamaño final de buffer $t_{i,j}$ tal como se muestra en la ecuación [3.5].

$$\text{CPPN}(x_i, y_i, x_j, y_j) = \begin{cases} \rho_{i,j}, \tau_{i,j} & \text{si } |\rho_{i,j}| > \rho_{min} \\ (0,0) & \text{e.o.c} \end{cases} \tag{3.4}$$

$$\begin{aligned} \omega_{i,j} &= \omega_{max} \cdot \rho_{i,j} \\ t_{i,j} &= \tau_{max} \cdot \tau_{i,j} \end{aligned} \tag{3.5}$$

La fig. 3.12 muestra el funcionamiento de τ-HyperNEAT pudiendo establecer similitudes inmediatas entre éste y su predecesor HyperNEAT. La red CPPN recibe las coordenadas de puntos determinando a partir de la activación de la red completa el

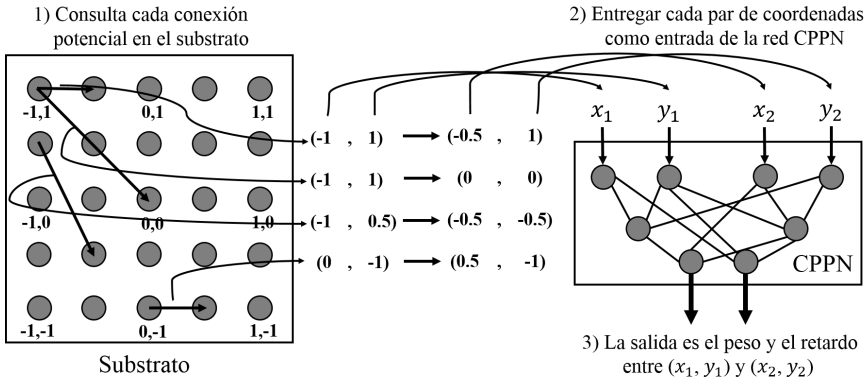
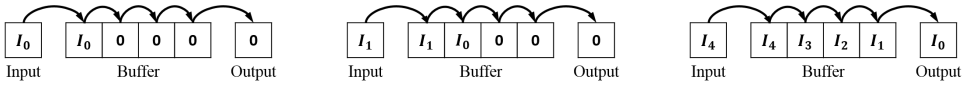


Fig. 3.12: **Obtención de patrón de conectividad de un hipercubo usando τ-HyperNEAT.** La obtención de pesos en la conexión del substrato se realiza de la misma forma que usando HyperNEAT, sin embargo la red CPPN posee una salida adicional que corresponde al porcentaje de retardo de la conexión.

valor de la conexión candidata. Sin embargo en este caso hay una salida adicional correspondiente al porcentaje de retardo de la conexión. Así como ocurre en HyperNEAT la distribución de pesos de las conexiones generan un patrón de conectividad, donde los retardos también estarán sujetos a la estructura que exhibe la red CPPN, y por lo tanto también al patrón generado.

La ANN activa sus neuronas de forma usual, es decir todas las conexiones entrantes a una neurona de destino son sumadas y este resultado es activado por la función correspondiente, sin embargo en este caso cada una de las conexiones posee un buffer de retardo provocando que la activación proveniente de cada neurona de origen se vea retrasada. El funcionamiento específico del buffer sigue un esquema FIFO (*first in, first out*) y es explicado a partir del siguiente ejemplo, suponer que la conexión entre el nodo de origen N_1 y el nodo de destino N_2 cumple con umbral mínimo de conexión y ésta se establece, asignando un buffer de retardo de largo l, sucederá que una primera activación I_0 de N_1 ingresará al buffer ocupando la primera posición de éste y una vez que N_1 sea nuevamente activada ingresará un nuevo valor al buffer permitiendo que el anterior se desplace una posición, por lo que la salida I_0 tendrá efecto en la entrada de N_2 una vez que N_1 sea activada $l+1$ veces. En la fig. 3.13 se ilustra el funcionamiento recién descrito.



(a) Primera Iteración *(b)* Segunda Iteración *(c)* Quinta Iteración

Fig. 3.13: **Buffer de retardo.** Cada conexión dentro del substrato posee un buffer de retardo, con un largo determinado por una red CPPN. En las imágenes se muestra un ejemplo de una conexión asignada con un buffer de tamaño 4, cuyos valores iniciales son cero. En una primera activación (a), la salida activada de la neurona de origen entrega un valor de entrada I_0 al buffer, colocándose al inicio de este y desplazando todos los demás valores contenidos en el buffer hacia la derecha. En la segunda activación (b), se entrega una nueva entrada I_1 al buffer volviendo a desplazar cada valor contenido en el buffer hacia la derecha. Una vez que se activa por quinta vez la neurona de origen (c), se ingresa un nuevo valor a la entrada del buffer y se obtiene a la salida el valor I_0 ingresado en la primera activación.

Este método permite aplicar conceptos temporales a la solución de un problema en específico haciendo uso de neuroevolución y algoritmos genéticos. Esto es buscado debido a que en la naturaleza ninguna acción es estrictamente instantánea y estamos rodeados de ejemplos. Para el caso de estudio de caminatas de robots ofrece resultados muy interesantes como fue estudiado en [5].

3.6. EVOLUCIÓN DE HYPERNEAT Y τ-HYPERNEAT A PARTIR DE CPPN-NEAT

Evolucionar substratos en HyperNEAT y τ-HyperNEAT sigue pasos idénticos en términos de construcción de conectividad entre nodos. τ-HyperNEAT agrega retardos a sus conexiones que no tienen efecto en la estructura del substrato, por lo tanto se explicará el método neuroevolutivo para HyperNEAT entendiendo que se aborda desde un punto de vista de conectividad.

El procedimiento del algoritmo sigue los siguientes pasos:

1. Se elige una configuración de substrato y se determina la posición de los nodos de entrada y salida. En caso de existir una configuración con una etapa intermedia de nodos también es establecida en este punto.

2. Se inicializa una población de redes CPPN a partir de NEAT, es decir una topología mínima con pesos aleatorios en sus conexiones.

3. Se repite hasta encontrar una solución o alcanzar número máximo de iteraciones:

   a) Por cada miembro de la población de redes CPPN:

      (I) Se determina el peso de las conexiones en el substrato, respetando la condición de superar el umbral mínimo para establecerse. Si el algoritmo utilizado es τ-HyperNEAT y la conexión es válida se asigna además un tamaño de buffer.

      (II) El substrato es utilizado como una ANN donde la salida del sistema completo es sometida a una función de desempeño para determinar si se ha encontrado una solución.

   b) Se reproduce y/o muta la población de CPPN utilizando el método NEAT pasando a la siguiente generación con un nuevo conjunto de redes CPPN

A medida que avanzan las generaciones del ejercicio los resultados pueden resultar siendo cada vez más complejos. Es posible detectar algunas simples regularidades en las primeras iteraciones y dependiendo de qué tan difícil sea de alcanzar la solución, los individuos de la población de redes CPPN complejizarán sus estructuras generando nuevos patrones de conectividad. Entendiendo el proceso que sigue, tanto en HyperNeat como en τ-HyperNEAT se puede observar que, si bien los patrones generados entre los nodos del substrato son variables respecto a la evolución de la población de CPPN, su estructura es fija. Se tuvo un primer acercamiento al aumento de topologías con NEAT, para aterrizar en la descripción de patrones sobre topologías fijas. Sin embargo la red CPPN utilizada puede definir información importante fuera del espacio de evaluación que ingresa a la red CPPN, o nodos contiguos pueden poseer información redundante para el sistema, es por esto que en la siguiente sección se presenta ES-Hyperneat o *Evolvable Substrate HyperNEAT* que además de aprovechar

las ventajas presentadas por su predecesor HyperNEAT permite que la descripción de conectividades de una red CPPN sea aprovechada de forma más óptima.

4. ES-HYPERNEAT

Una de las metas más ambiciosas de la neuroevolución es la de obtener, a partir de sus métodos, neurocontroladores similares al cerebro con millones de neuronas y conexiones. La escala de los desarrollos actuales se encuentra aún muy alejada del cerebro humano, sin embargo la funcionalidad de éste no responde simplemente a una dimensionalidad extremadamente elevada sino que también está muy relacionada a su organización estructural.

A medida que los algoritmos neuroevolutivos han logrado evolucionar estructuras grandes y complejas sobre ANN, esfuerzos se han concentrado en la codificación de ellas, como es el caso de NEAT donde un genotipo describe los nodos y conexiones en forma de genes que producen un fenotipo que expresa la ANN como tal. Esto ha dado paso a pensar en métodos de codificación indirecta como es el ejemplo del aumento de topologías que presenta NEAT y la búsqueda de la representación de patrones en función de parámetros de entrada a un sistema como es el caso de las redes CPPN, que en su conjunto permite a estos métodos la reutilización de genes y por ende la optimización de su codificación.

HyperNEAT muestra que la neuroevolución se beneficia de la geometría de un problema, haciendo uso de una red CPPN que genera patrones de conectividad en un substrato, que en términos prácticos actúa como la red neuronal con la que se evalúa la posible solución. Sin embargo el usuario de HyperNEAT debe escoger la posición de nodos en el substrato de forma manual, ya sea un substrato bidimensional, tridimensional o el tipo que sea conveniente para el problema. Esto presenta un problema de tipo restrictivo sobre los resultados que pueda encontrar la evolución de redes CPPN, pues los nodos posicionados de forma manual pueden no coincidir espacialmente con

la distribución de pesos que describe la estructura generada por la red CPPN. Aplicaciones como *developmental robotics* se verían directamente beneficiadas si, haciendo uso de un algoritmo neuroevolutivo como HyperNEAT, no estén sujetas a espacios reducidos de búsqueda sino que la misma exploración de cierta acción entregue pistas sobre los nodos y conexiones de un substrato, y no viceversa.

Es factible pensar que los patrones generados por redes CPPN contienen información implícita de dónde deben ser posicionados los nodos en el substrato, de forma que la información contenida en los patrones de conectividad sea aprovechada de manera óptima. Nuevos problemas se hacen presentes al intentar aplicar un método de estas características, como la redundancia de información, es decir que dos o más nodos contiguos posean la misma información o con muy pocas variaciones, forzando al argumento del método a tener en cuenta ésta y otra información que afecte la dimensionalidad del substrato de forma negativa. Sin embargo y como se verá a lo largo de esta sección, la información contenida en el patrón de conectividad de una red CPPN puede ser utilizada de forma tal que presente de forma clara dónde deben ser posicionados los nodos de un substrato a partir de las entradas y salidas del sistema, sin sobredimensionar el espacio de búsqueda y discriminando la información redundante.

ES-HyperNEAT[6] es capaz de determinar la geometría de posicionamiento de nodos en un substrato así como también su densidad basándose sólamente en información entregada por una red CPPN. En la siguiente sección es descrito el método de extracción de información sobre la red CPPN a partir de la estructura de nodos de entrada y salida, conocido de aquí en adelante como *quadtrees* [10].

4.1. EXTRACCIÓN DE INFORMACIÓN A PARTIR DE QUADTREES

Para encontrar la información implícita sobre la posición de nodos en el patrón generado por la red CPPN, ES-HyperNEAT hace uso de una herramienta de multi-resolución llamada *quadtrees* cuyo fundamento general consiste en sub-dividir re-

giones bidimensionales en 4 sub-regiones, de esta forma la descomposición obtenida puede ser representada como un árbol cuyo padre es la región original que fue sub-dividida. Es posible realizar divisiones en las regiones obtenidas hasta alcanzar una resolución deseada o hasta que no sea necesaria seguir sub-dividiendo.

El usuario escoge la posición de los nodos de entrada y salida en el substrato y, a partir de la posición de cada uno de ellos, ES-HyperNEAT descubre de forma iterativa las conexiones posibles de salida (*outgoing*) en el caso de nodos de entrada, o las conexiones entrantes en nodos de salida (*incoming*). Por ejemplo, dado un nodo de entrada de coordenadas (a, b) se obtiene el patrón de conectividad a partir de la función CPPN(a, b, x, y), donde el rango de posiciones está definido entre $[-1, 1]$. A partir de este punto el algoritmo trabaja en dos fases principales, cuya representación se observa en la fig. 4.2:

1. **Fase de división e inicialización:** En esta etapa es creado el *quadtree* subdividiendo un espacio bidimensional en 4 sub-regiones de manera recursiva hasta alcanzar una resolución r_t deseada, por ejemplo un espacio sub-dividido en 8×8 sub-regiones corresponde a un *quadtree* de profundidad 4 (fig. 4.1). Por cada cuadro del *quadtree* con centro en (x, y) se obtiene el peso de la conexión candidata a partir de la función CPPN(a, b, x, y).

   Una vez determinados los $(\omega_1, \omega_2, ..., \omega_k)$ pesos de los k nodos hoja de un sub-árbol de nodo p en los *quadtree*, se obtiene la media $\overline{\omega_p}$ para poder calcular la varianza σ_p^2 del nodo padre p respecto a los pesos de sus hojas, tal como se observa en la ecuación [4.1].

   $$\sigma_p^2 = \frac{1}{k} \sum_{1}^{k} (\overline{\omega_p} - \omega_i)^2 \tag{4.1}$$

   Si la varianza σ_p^2 es mayor a un umbral de división d_t entonces el proceso de sub-división puede ser aplicado de forma recursiva a los nodos hojas del sub-árbol teniendo las mismas consideraciones recién descritas, permitiendo así aumentar

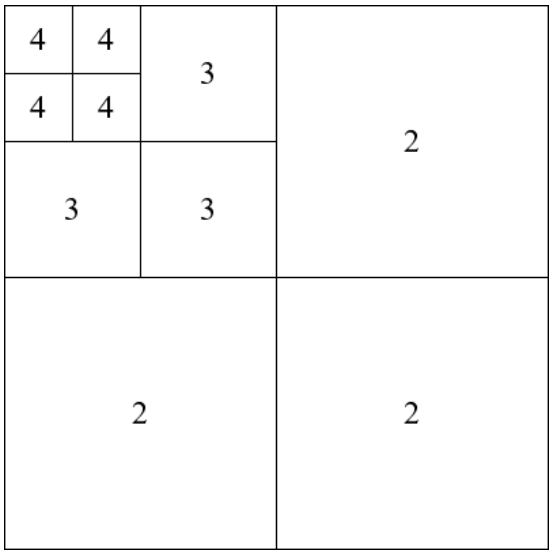


Fig. 4.1: **Representación de espacio dividido en quadtrees.** El espacio se sub-divide de forma recursiva en 4 sub-regiones hasta alcanzar *quadtrees* de profundidad 4. Al centro de cada sub-región obtenida se muestra el nivel de profundidad alcanzado usando este método.

la densidad de nodos candidatos. En caso contrario la sub-división llega hasta ese punto. Se define una variable de profundidad inicial que fuerza al algoritmo a al menos llegar a esa cantidad de iteraciones en la sub-division, pudiendo además definir una variable de resolución máxima r_m que permite al algoritmo explorar el espacio en caso de que la densidad de nodos pueda incrementarse y aprovechar aún más la información del patrón de conectividad.

2. **Fase de extracción y poda:** Las conexiones resultantes de la Fase de división e inicialización no completan aún el substrato utilizado en los entrenamientos de ES-HyperNEAT, pues el análisis de la varianza en los nodos generados ha sido evaluada sólo entre los nodos padres de cada sub-árbol y sus hojas, lo que nos dice que a lo largo del espacio bidimensional completo pueden aparecer zonas con alta varianza que no fueron consideradas en la primera etapa, como también puede encontrarse información redundante entre un sub-árbol y sus vecinos, esto puede observarse en la fig. 4.3a donde se aprecia una alta densidad de nodos una vez aplicada la fase de División e Inicialización sobre un nodo de entrada

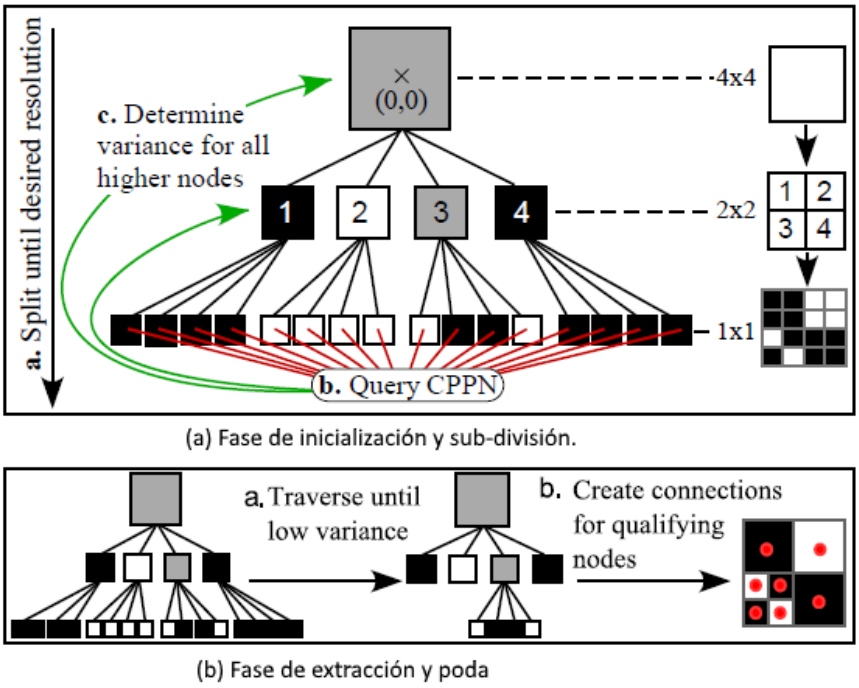
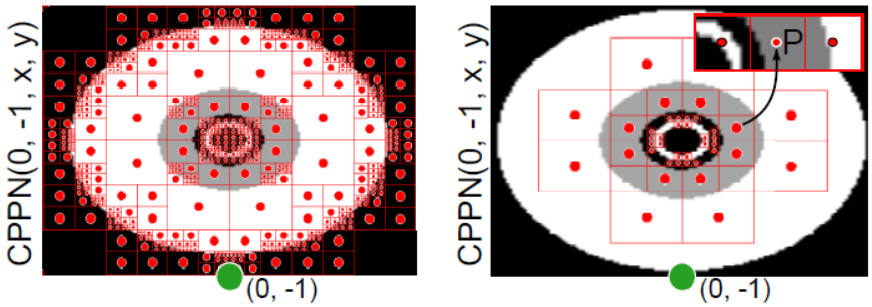


Fig. 4.2: **Fases de inicialización de substrato:** En la figura superior se observa la fase de división e inicialización cuyos pasos son (a) dividir hasta alcanzar resolución deseada, (b) obtener pesos de los puntos obtenidos y (c) determinar la varianza de nodos padres de sub-árboles. Mientras que la figura inferior muestra la fase de extracción y poda donde (a) se navega por los nodos padres descartando conexiones en nodos padres con baja varianza y (b) a través del método de bandas se determinan conexiones temporales que eviten redundancia de información. Imagen adaptada de [6]

de coordenadas $(0, -1)$. Para llevar a cabo esta segunda fase el algoritmo se encarga de evaluar recursivamente la varianza de cada nodo padre entre los sub-árboles generados. En caso de superar un umbral de varianza σ_t^2 se realiza el proceso de poda sobre él, aplicando este proceso recursivo hasta que no se hallen nodos hijos. Si σ_p^2 de coordenadas (x, y) supera σ_t^2 se procede a un sistema de evaluación de nodos vecinos denominados **bandas**, que consiste en puntos que son encerrados por al menos dos nodos vecinos en lados contrarios con diferentes niveles de activación CPPN. Los puntos en este caso corresponden a la ubicación del nodo p, el cual tiene un ancho (o alto para bandas evaluadas verticalmente) w definido por la profundidad en del sub-árbol en el que se encuentre. Una representación de lo anterior se puede observar en la fig. 4.3b en la



(a) Fase de inicialización y división *(b)* Fase de extracción y poda

Fig. 4.3: **Patrones de conectividad entre fases de inicialización:** (a) muestra un plano bidimensional con un patrón generado a partir de una función CPPN$(0, -1, x, y)$, con (x, y) los puntos explorados del plano en la fase de división e inicialización. (b) Muestra el mismo plano con una densidad de puntos mucho menor, esto gracias a la fase de extracción y poda que a partir del método de bandas discrimina los nodos que posean baja varianza para evitar la redundancia de información. Imagen adaptada de [6]

parte superior derecha muestra el punto P y dos vecinos horizontales. La banda del nodo p se calcula a partir de la ecuación [4.2], donde d_{left} corresponde a la diferencia entre la activaciones de una red CPPN del punto (a, b, x, y) y $(a, b, x - w, y)$, donde d_{top}, d_{bottom} y d_{right} son calculados de forma idéntica con el vecino superior, inferior y derecho respectivamente. Si la banda β es menor a β_t (umbral de banda) la conexión no es expresada, en caso contrario el nodo p es agregado a la capa intermedia de nodos y la conexión es expresada.

$$\beta = \text{máx}(\text{mín}(d_{top}, d_{bottom}), \text{mín}(d_{left}, d_{right})) \tag{4.2}$$

La diferencia entre la densidad de puntos en las figuras 4.3a y 4.3b muestra que la red CPPN tiene la capacidad de expresar información que es redundante para el ejercicio y con las fases expuestas en ES-HyperNEAT es posible explotar las regularidades de la red de forma muy eficiente.

Gracias a esta fase se logra comprender la información implícita que posee una red CPPN en términos de valores de pesos, estructura y densidad de nodos a lo

largo de un substrato, pudiendo dirigir el algoritmo hacia soluciones de codificación cada vez más indirectas y naturales.

4.2. EVOLUCIÓN DE SUBSTRATO EN LA RED HYPERNEAT

El procedimiento de evolución de substrato involucra las dos fases previamente descritas o **fases de inicialización**, sin embargo las conexiones obtenidas son tratadas como temporales para adicionar un tercer paso después del proceso inicial.

En primera instancia se aplican las fases de inicialización en todos los nodos de entrada obteniendo así una lista de conexiones temporales desde las entradas a la capa intermedia, así como nuevos nodos adicionados. A continuación se aplica las fases de inicialización sobre los nodos de capa intermedia de forma reiterada hasta alcanzar un nivel de iteración i_t definido por el usuario, donde pueden ser creados nuevos nodos de capa intermedia y nuevas conexiones temporales. Tanto en el caso de nodos de entrada como intermedios, las fases de inicialización fueron aplicadas para nodos que expresaban conexiones salientes (*outgoing*). Sin embargo para la capa de nodos de salida al tratarse de conexiones entrantes (*incoming*), se aplican las fases de inicialización pero sólo son almacenadas como conexiones temporales aquellas cuyo origen sea un nodo ya existente, ya sea de entrada o capa intermedia.

Una vez que todos los nodos de la capa intermedia son descubiertos sólo aquellos que a través de sus conexiones puedan formar un camino hacia algún nodo de entrada y de salida son expresados. Además, las conexiones que terminan en puntos muertos tampoco son expresadas, conservando aquellas que aportan información desde las entradas a las salidas en el substrato. Este acercamiento iterativo permite reducir los costos de cómputo a la hora de activar las neuronas en el substrato. En la imagen 4.4 se ilustra el mecanismo de discriminación de nodos a partir de la información que acarrean.

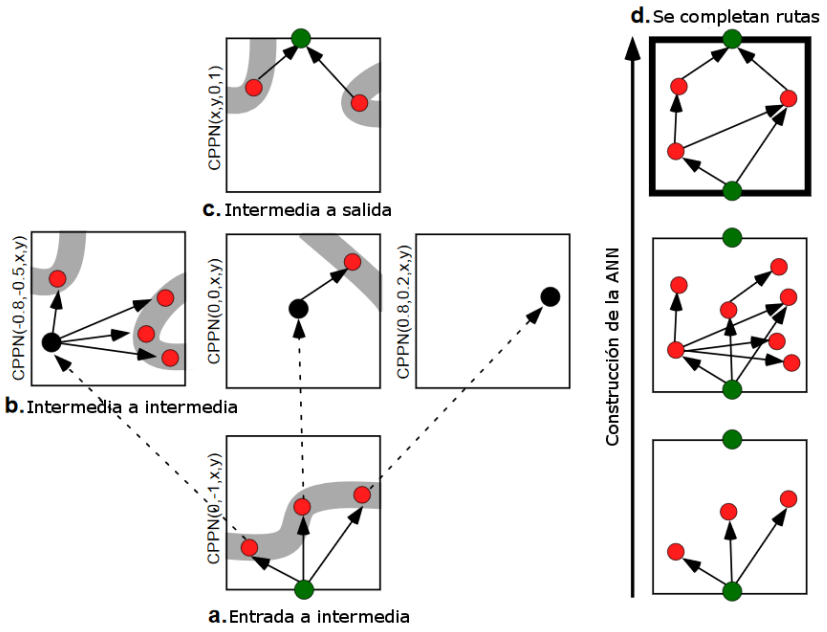


Fig. 4.4: **Determinación de nodos en el substrato:** Pasos finales en la creación del substrato de ES-HyperNEAT con la determinación de conexiones y nodos temporales en (a), (b) y (c). En la construcción de la ANN (d) se muestra que son descartados aquellos nodos cuyas conexiones no forman un camino entre los nodos dos de entrada y salida, es decir no aportan información en el cálculo de las salidas de la red. Imagen adaptada de [6]

5. IMPLEMENTACIÓN DE ES-HYPERNEAT

ES-HyperNEAT hace uso de redes CPPN para describir un patrón de conectividad en un substrato a construir, que son complejizadas y evolucionadas a partir de NEAT. Además se entiende que este nuevo algoritmo es el sucesor de HyperNEAT, por lo tanto con el propósito de continuar el trabajo previo realizado en [5] se utilizará la implementación de CPPN-NEAT existente, no sin antes revisar su funcionamiento e implementar las modificaciones pertinentes en su codificación que puedan optimizar el procedimiento de construcción de substratos.

En primer caso se observa la factibilidad de codificar el algoritmo ES-HyperNEAT haciendo uso de las declaraciones de substratos y nodos existentes en [5], codificado en C++. Se comprueba la velocidad de creación de substrato y la activación a lo largo de la red, cuyas conclusiones son profundizadas en la sección a continuación.

5.1. CLASE NEURALNETWORK

Al tratar a HyperNEAT como antecesor del algoritmo ES-HyperNEAT es deseable realizar una actualización en la forma en la que se describe el substrato. En el primero tanto los nodos de entrada y salida como los de la capa intermedia son posicionados de forma manual, mientras que ES-HyperNEAT pretende aprovechar los patrones expresados por la red CPPN a lo largo de un espacio bidimensional posicionando nodos de la capa intermedia done la información del patrón de conectividad sea significativa. Por lo tanto la situación ideal es reemplazar el posicionamiento de nodos de HyperNEAT que ahora son descritos en un archivo *json* y cargados en el programa, por un

posicionamiento inteligente a partir del algoritmo a codificar, esto quiere decir que el substrato comenzará sólo con los nodos de entrada y salida, y el resto de nodos se adicionarán a medida que transcurra el entrenamiento.

La implementación conocida de HyperNEAT utiliza un substrato descrito por una clase *Substrate*, cuyos nodos son codificados en una clase llamada *SpatialNode*. Los nodos del substrato se declaran en la clase *Substrate* como un vector de punteros a *SpatialNode*, donde los últimos poseen un vector de la misma clase que direcciona a los nodos con los que poseen conexiones en sus entradas. Un paso de la construcción del substrato es instanciar por cada nodo un objeto *SpatialNode*, ingresarlo a los vectores correspondientes y luego direccionarlos a partir de otros vectores hacia el resto de los nodos, que para una estructura fija como HyperNEAT no resulta en mayores dificultades a la hora de activar la red y obtener las salidas de los nodos, pero en el caso de ES-HyperNEAT no ha sido el caso. Al tratarse de una estructura de substrato dinámica utilizar punteros a objetos ha resultado en problemas para realizar las fases de inicialización, específicamente en la fase de Extracción y Poda (figura 4.3) y la eliminación de conexiones (figura 4.4), pues cada *SpatialNode* es direccionado en la misma clase hacia los nodos con los que está conectado. Diversos errores de acceso y almacenamiento en memoria fueron detectados, además de una excesiva lentitud para realizar los cálculos de la salida de la red en ejemplos simples como el entrenamiento de resolución de la operación lógica *xor*.

Tal como se ha manifestado a lo largo de este documento ES-HyperNEAT describe el substrato a partir de la información entregada por la red CPPN y es posible reconstruirlo de forma idéntica si la misma red es utilizada nuevamente. Por lo tanto en este desarrollo se propone las clases *Neuron* y *Connection* codificadas para implementar la clase *NeuralNetwork*, que corresponde a una red neuronal de bajo costo computacional que posee vectores de neuronas y conexiones descritas por las clases recién mencionadas, cuyas instancias mantienen la información justa y necesaria para realizar cálculos de activación en un tiempo muy reducido, esto pues las variables de cada una están pensadas exclusivamente para este propósito. Se llegó a esta determi-

nación a partir de la premisa expuesta al comienzo de este párrafo, pues reconstruir el substrato a partir de una instancia de la clase *NeuralNetwork* ha demostrado ser mucho más eficiente que el uso de las clases *Substrate* y *SpatialNode*. Sin embargo el alcance de esta clase no apunta sólo a la construcción de substratos pues a partir de un genoma evolucionado por el algoritmo NEAT es posible expresar su genotipo en esta misma clase, pudiendo externalizar el proceso de cálculo de la red CPPN a un sistema de bajo costo y eficiente. La clase *NeuralNetwork* ha demostrado además un muy buen comportamiento en las fases de inicialización del substrato en ES-HyperNEAT pudiendo realizar la fase de Extracción y Poda y el descarte de conexiones cuya información no sea relevante en la salida sin mayores problemas.

5.2. ADICIÓN DE FACTORES EVOLUTIVOS EN NEAT

Para la evolución de redes CPPN se hace uso de NEAT, recogiendo la implementación utilizada en [5] como un peldaño para aplicar sobre substratos en ES-HyperNEAT. En este caso el algoritmo es necesario para obtener patrones de conectividad en un espacio bidimensional, que son en realidad la expresión de una distribución de puntos que representan el peso de una conexión obtenidos a partir de una red CPPN. Utilizar la codificación actual de NEAT no ha representado un inconveniente para expresar los patrones y regularidades en el espacio bidimensional, pero si se han detectado tendencias poco favorables en la evolución de las redes, específicamente no se observa un privilegio sobre los organismos que ofrecen los mejores resultados, la descendencia de los organismos agrupados en especies no es determinada de forma eficiente privilegiando en ocasiones a individuos cuyo desempeño es extremadamente bajo y esto puede explicarse por una distribución de especies inadecuada y falta de factores determinantes que permitan converger a las poblaciones a la solución óptima. En base a esto se han adicionado factores evolutivos y procedimientos asociados a ellos que permiten a los entrenamientos converger en valores de desempeño promedio mucho más elevados.

Los factores adicionados en el funcionamiento del algoritmo tienen efecto en el método que se encarga de agrupar en especies a la población y reproducirlas a lo largo de generaciones en los entrenamientos, es decir el método *Epoch* (época en español). Ha sido necesario re-codificar por completo este método, junto con aquellos que se encargan de reproducir las poblaciones y luego agruparlas en especies. Un cambio adicionado importante es el ordenamiento de mayor a menor fitness de los individuos al interior de cada especie, aplicando luego un ordenamiento de mejor a peor especie, de esta forma en los arreglos que almacenan individuos y especies se encuentran en primera posición aquellos que han mostrado mejor desempeño en la generación actual, así es posible aplicar nuevos mecanismos para determinar herencia en base a los mejores resultados. Se enlista a continuación los factores y procedimientos adicionados para el correcto funcionamiento del re-diseño de los métodos recién mencionados:

- **Factor de des-estancamiento:** Tiene como objetivo medir el estancamiento de la evolución en el algoritmo NEAT comparando entre generaciones la evolución del mejor fitness de la población completa. Si a lo largo del entrenamiento el algoritmo detecta que el mejor fitness no ha mejorado por una cantidad de generaciones el factor de des-estancamiento es habilitado y se realiza un proceso de purga en la población, replicando los mejores organismos en la población de forma que la evolución continúe desde este punto. Este factor puede ser deshabilitado para impedir el proceso de purga pues en algunos ejercicios puede no ser deseado.

- **Cálculo de herencia de especie:** Uno de los mecanismos de NEAT es la determinación de fitness compartido de las poblaciones, que permite ponderar un valor de desempeño para los individuos respecto al tamaño de la especie a la que pertenecen. Calculando un fitness compartido promedio se decide la cantidad de herencia que tendrá cada individuo, dividiendo su fitness compartido por el recién calculado promedio de desempeño compartido de la población. Una vez obtenida la cantidad de descendencia de cada individuo, se suman a nivel

de especie para obtener la herencia final de cada una de ellas.

- **Umbral de supervivencia:** Consiste en un porcentaje de la población de una especie cuyos individuos serán seleccionados al azar para reproducirse. Esto permite a las especies seleccionar aunque de forma aleatoria a los mejores organismos que pertenecen a ella para reproducirse o mutar para pasar a las siguientes generaciones, en vez de hacer una selección aleatoria de todos los individuos que pertenezcan a ella. Este factor debe ser habilitado en la inicialización del entrenamiento y asignar el porcentaje correspondiente.

- **Factor de elitismo:** Es un porcentaje que determina la cantidad de población de una especie que pasará a la siguiente generación intacta, de esta forma los mejores individuos de cada generación no perderán incidencia en los resultados finales y en base a ellos puede encontrarse nuevas y mejores soluciones, ayudando al resto de la población a converger a mejores desempeños.

Los factores aplicados en su conjunto al método de cambio de época y los derivados de reproducción y especiación han demostrado ser muy eficientes en el incremento de fitness promedio de la población total, sin perder las bases neuroevolutivas que dicta el algoritmo NEAT, por lo tanto el siguiente paso es utilizar estos métodos para evolucionar redes CPPN y usarlas en los substratos de ES-HyperNEAT. En la siguiente sección se mostrará la codificación de los principales métodos del algoritmo y cómo son utilizados en la construcción del substrato.

5.3. MÉTODOS Y CLASES DE ES-HYPERNEAT

En esta sección se describen los métodos codificados para hacer uso de ES-HyperNEAT basándose en el fundamento teórico del capítulo anterior[1].

[1] `https://github.com/pabloreyesrobles/ESHyperNeat`

5.3.1. Clases

Los métodos utilizados a lo largo de la ejecución del algoritmo hacen uso de una variedad de objetos que son exclusivos de éste, y son principalmente instanciados en las fases de inicialización.

- **ES-HyperNEAT:** corresponde a la clase principal del algoritmo, contiene todos los métodos que corresponden a fases de inicialización y construcción de substratos. Además dentro de su codificación se incorporan las clases *QuadPoint* y *TempConnection*, pero éstas no heredan de la clase *ES-HyperNEAT*.

- **Quadpoint:** se utiliza para representar los puntos o nodos de los sub-árboles que son generados en las fases de inicialización, de forma que al sub-dividir el espacio de forma recurrente en 4 secciones sus centros son instanciados a través de los *Quadpoint*. Contiene información como las coordenadas del punto en el espacio, el ancho del espacio donde está contenido, la profundidad que se encuentra respecto a las etapas sub-división, un vector de *Quadpoint* que apunta a los nodos hijos y la varianza de pesos entre ellos.

- **TempConnection:** describe conexiones temporales que son realizadas en las fases de inicialización. Se utiliza esta clase en vez de aquella que representa las conexiones finales (clase *NeuralNetwork*) pues no tiene métodos y sus variables son elementales: coordenadas de nodo origen y nodo destino, y el peso de la conexión, por lo que instanciarla no representa un gran costo computacional.

5.3.2. Métodos

Son codificadas las fases de inicialización junto con un par de métodos que son de utilidad para la construcción de los substratos.

- **void ESHyperNEAT::DivideAndInitialize(vector <double> node, shared_ptr <QuadPoint> root, Genetic_Encoding *organism, bool outgoing):** recibe como argumentos las coordenadas del nodo de búsqueda (inicialmente entradas

y salidas), un puntero a *Quadpoint*, una red CPPN y si es que el nodo generará conexiones salientes o entrantes. Este método modifica el puntero a *Quadpoint* recurrentemente según sea descrito por las variables de resolución, para ser utilizado por la siguiente fase.

- **void ESHyperNeat::PruneAndExtraction(vector <double> node, shared_ptr <QuadPoint> root, Genetic_Encoding *organism, vector <TempConnection> &temporal_connections, bool outgoing):** recibe como argumentos las coordenadas del nodo de búsqueda, el puntero a *Quadpoint* que modifica el método *DivideAndInitialize*, una red CPPN, un vector de punteros a *TempConnection* y si es que el nodo generará conexiones salientes o entrantes. Se lleva a cabo la fase de Extracción y Poda como es descrito en el capítulo anterior iterando sobre el puntero a *Quadpoint*, por lo que si se supera el umbral de banda se genera una conexión temporal entre el nodo de búsqueda y el *Quadpoint* correspondiente. El método culmina con el vector de punteros a *TempConnection* con todas las conexiones temporales candidatas a pertenecer al substrato.

- **double ESHyperNeat::QuadPointMean(shared_ptr <QuadPoint> point):** método que recibe como argumento un puntero a *Quadpoint* y retorna el promedio de pesos en los nodos hijos.

- **double ESHyperNeat::QuadPointVariance(shared_ptr <QuadPoint> point):** método que recibe como argumento un puntero a *Quadpoint* y retorna la varianza de pesos en los nodos hijos.

- **void ESHyperNeat::Clean_Net(vector <Connection> &t_connections, unsigned int node_size, unsigned int io_count):** recibe como argumentos las conexiones resultantes del método *PruneAndExtraction* luego de evaluar si superan el umbral de conexión, la cantidad de nodos existentes y la cantidad de nodos de entrada y salida. Itera sobre las conexiones eliminando todas las conexiones que no pertenezcan a un camino formado entre algún nodo de entrada y

otro de salida.

- **bool ESHyperNeat::createSubstrateConnections(Genetic_Encoding *organism, NeuralNetwork &net):** es el método mayor de ES-HyperNEAT, dentro de su codificación se encuentran los métodos *DivideAndInitialize, PruneAndExtraction* ejecutados sobre etapas separadas según sea el tipo de nodo (entrada, intermedio o salida), agrega nodos en la capa intermedia en caso de las conexiones que los contengan sean válidas, construyendo el substrato a través de la clase *NeuralNetwork*, y finalmente llama a la función *CleanNet* para obtener el substrato final que genera el organismo o red CPPN.

A partir de este punto es posible utilizar los substratos que construyen la implementación de ES-HyperNEAT para evaluar el aprendizaje de caminatas en robots con extremidades móviles. En el siguiente capítulo se detallan las componentes y procedimientos utilizados para los entrenamientos.

6. APRENDIZAJE DE MOVIMIENTOS EN ROBOTS MÓVILES

Las redes de neuronas artificiales han sido protagonistas en el campo de la inteligencia artificial en los últimos años, gracias a ellas es posible encontrar soluciones a una infinidad de problemas de diversos tipos. Incontables algoritmos han sido desarrollados, cuyas topologías y criterios de evaluación de resultados varían según el contexto de aprendizaje al que está sujeto dicho algoritmo. Sin embargo cada uno de ellos debe realizar entrenamientos que permitan que las ANN evolucionen y encuentren la solución buscada para cada problema. En este capítulo se presenta el procedimiento adoptado para el entrenamiento de redes neuronales para el aprendizaje de caminatas en robots con extremidades móviles, posteriormente se presentan los resultados de aprendizaje y se realizan cuadros comparativos de desempeño entre ES-HyperNEAT, su predecesor HyperNEAT y su variación τ-HyperNEAT.

6.1. ENTORNO DE SIMULACIÓN Y CONECTIVIDAD MEDIANTE ROBOTLIB

El propósito de este proyecto de memoria es el aprendizaje de movimientos en robots con extremidades móviles a partir del entrenamiento de una ANN que de alguna forma describa el estado futuro de los movimientos de dicho robot a partir de su estado actual. Utilizar robots reales para este tipo de experimentos involucra consideraciones y costos que pueden acarrear resultados infructuosos en las labores de entrenamiento, además de significar un peligro innecesario tanto para el robot utilizado como para el operador del mismo, es por esto que se utiliza el entorno de simulación V-REP cuya API permitirá la comunicación de programas externos para el manejo de todo

tipo de aspectos dentro del programa. Para conectar el algoritmo de entrenamiento con V-REP se hace uso de RobotLib[1], una herramienta que permite la comunicación e interacción con el entorno de simulación. RobotLib está codificada en lenguaje C/C++ y puede ser utilizada como librería externa, pudiendo acceder desde el programa de entrenamiento a todo tipo de recursos para el manejo de elementos del robot simulado y almacenamiento de información relevante para el posterior análisis de resultados.

La codificación de RobotLib involucra dos grupos de clases, cuya distribución colabora a eficiente uso de recursos pudiendo incluso ejecutar simuladores simultáneos de forma simple y segura. Se presentan a continuación los grupos de clases que componen la librería:

- **Componentes de entorno:** corresponden a todo tipo de objetos a manipular como motores, extremidades, piezas o sensores. Al instanciar objetos de este grupo en el programa de entrenamiento se tiene acceso a la representación del mismo en el entorno de simulación, sin embargo sus estados no son controlados desde el mismo. Adicionalmente existe una clase en este grupo que permite representar objetos que puedan colisionar durante un experimento.

- **Controladores:** este grupo se compone de clases que una vez instanciadas permiten a un programa externo ejecutar acciones sobre los componentes de entorno, ya sea cambiar la posición de piezas, variar la posición o velocidad de algún motor, o realizar lecturas de sensores. Es posible instanciar múltiples objetos controladores para el manejo simultáneo de componentes de entorno, por ejemplo el manejo paralelo de motores tanto en el entorno de simulación como en robots físicos, a través de una de las clases agrupadas como controladores. Para los propósitos de esta memoria se desestima el uso de esta propiedad pues la validación de la misma se encuentra en resultados a través de V-REP.

Un programa correctamente codificado puede hacer uso de los grupos de clases

[1] `https://github.com/osilvam/RobotLib`

presentes en la librería, pudiendo realizar cualquier tipo de experimento con robots móviles, posea tanto locomoción a partir de ruedas como extremidades móviles.

6.2. SIMULACIÓN DE CAMINATAS

Conforme se desarrollan sistemas computacionales más rápidos, potentes y capaces, la búsqueda de la autonomía de sistemas a partir del aprendizaje de máquinas es cada vez más ambiciosa. La infinidad de algoritmos presentes al día de hoy permite a los investigadores testear diversos métodos de aprendizaje sobre una misma plataforma, pudiendo evaluar el comportamiento de cada uno de ellos sobre el problema descrito. Previo a este proyecto los algoritmos HyperNEAT y τ-HyperNEAT fueron utilizados para el entrenamiento de caminatas en robots con extremidades móviles, específicamente los robots Quadratot y ArgoV2. Con el fin de comprobar la efectividad en el aprendizaje de movimientos haciendo uso de ES-HyperNEAT, se entrenarán redes de éste en condiciones similares a las aplicadas con los algoritmos predecesores de forma que el desempeño de los organismos entrenados pueda demostrar cuantitativamente una mejoría o no en sus resultados. Asimismo se evaluará el comportamiento en sus movimientos, ya sea en términos de su desplazamiento en el espacio como del movimiento de sus motores, que idealmente deben tener comportamientos oscilatorios.

Los entrenamientos consisten en una serie de simulaciones donde el robot ejecutará movimientos determinados por la estructura del substrato evaluado. Cada simulación tiene una duración máxima de 6 segundos y al final de ésta se aplicarán los criterios de evaluación correspondientes a partir de funciones de desempeño. Adicionalmente el programa de entrenamiento discrimina las simulaciones que presenten colisiones de las partes del robot entre ellas o con el piso del escenario (salvo los extremos de las patas), terminando de forma abrupta la simulación actual y penalizando su desempeño con un puntaje mínimo.

ES-HyperNEAT tiene la particularidad de definir la estructura de un substrato a

partir de la información implícita contenida en el patrón de conectividad de la red CPPN utilizada. Evidentemente no son definidos a priori las posiciones de nodos de la capa intermedia del substrato, no así los nodos de la capa de entrada y salida, pues sus ubicaciones son escogidas convenientemente en un substrato de tipo malla (plano bidimensional), esperando que el diseño estructural determinado por la red CPPN determine las regularidades necesarias de forma autónoma. Cada nodo de entrada representa la posición actual de una articulación (motor) del robot sometido a entrenamiento, mientras que cada nodo de salida corresponde el estado siguiente de cada una de las articulaciones, es decir el substrato es el encargado de generar las señales de accionamiento de los motores en el robot a partir del estado actual de ellos. Se adiciona además a la capa de entrada dos nodos con señales senoidales desfasadas en 90 grados entre ellas (seno y coseno) que permitirán generar un comportamiento oscilatorio en las señales de los nodos de salida, y por ende en los estados siguientes de movimiento. Finalmente las señales que ingresan a los nodos de entrada del substrato corresponden a la realimentación de los nuevos estados generados por los nodos de salida del substrato.

La construcción del substrato varía para cada robot dependiendo de los grados de libertad que cada uno posea: QUADRATOT posee 9 grados de libertad y ARGOV2 cuenta con 12. Cada uno de los casos será abordado de forma específica en las próximas secciones, sin embargo en primera instancia será descrito los criterios de evaluación a utilizar que son idénticos para ambos.

6.3. EVALUACIÓN DE FUNCIONES DE DESEMPEÑO

Tal como fue descrito en la sección anterior con el fin de comparar la existencia o no de una mejoría en el uso de ES-HyperNEAT se utilizarán las funciones de desempeño (*fitness*) como criterio de evaluación en entrenamientos con HyperNEAT y τ-HyperNEAT sobre robots con extremidades móviles [5]. Dichas funciones miden la distancia recorrida por el robot y la frecuencia en el movimiento de sus motores en

los últimos 5 segundos de simulación, esto para evitar lecturas erróneas en el tiempo cero de ejecución.

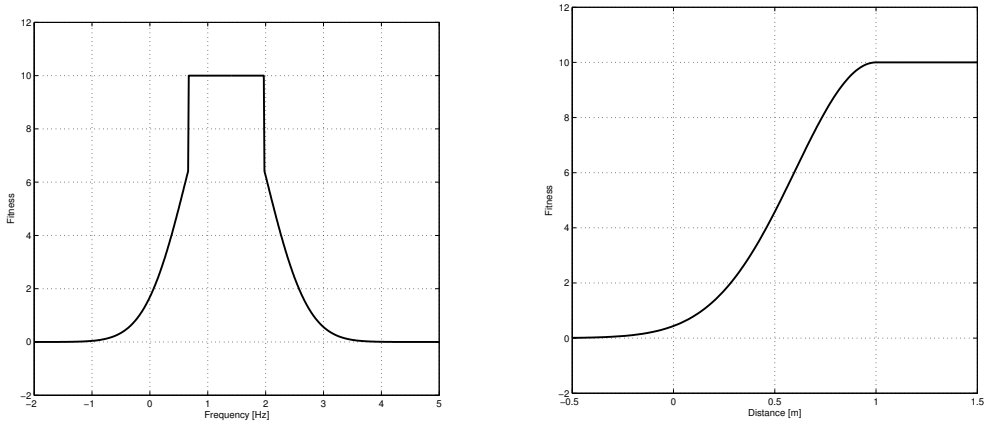
- **Evaluación de oscilaciones en extremidades:** para la evaluación de movimientos oscilatorios en los motores del robot se mide los cambios de signo en la pendiente de las señales generadas en la salida, de forma que dos cambios de signo consecutivos indicarán un período completo de oscilación en el movimiento del motor y por ende es posible medir su frecuencia. Así las frecuencias de los motores son promediadas por extremidad y posteriormente evaluadas según se define en la ecuación [6.1], cuya representación gráfica se muestra en la fig. 6.1a. Finalmente el *fitness* las extremidades son promediados según se define en la ecuación [6.2] entregando el resultado final del desempeño oscilatorio de los motores del robot. Las variables A, f y σ_f son constantes que se definen y describen en la tabla 6.1.

$$F_{frec}(f_i) = \begin{cases} A \cdot e^{-\dfrac{(f_i - f)^2}{2\sigma_f^2}} & \text{si } F_{frec}(f_i) < 6{,}5 \\ A & \text{si } F_{frec}(f_i) > 6{,}5 \end{cases} \tag{6.1}$$

$$\mathcal{F}_{frec} = \frac{1}{n} \sum_{i=1}^{n} F_{frec}(f_i) \tag{6.2}$$

- **Distancia recorrida en tiempo de simulación:** se calcula la diferencia entre la posición del robot en el segundo 1 de simulación y al término de la misma. Dicha distancia es evaluada en la ecuación [6.3] cuya representación gráfica se muestra en la fig. 6.1b. Las variables A, d y σ_d son constantes que se definen y describen en la tabla 6.1.

$$\mathcal{F}_{dist}(d_{robot}) = \begin{cases} A \cdot e^{-\dfrac{(d_{robot} - d)^2}{2\sigma_d^2}} & \text{si } d_{robot} < d \\ A & \text{si } d_{robot} > d \end{cases} \tag{6.3}$$



(a) Función de desempeño de oscilaciones. *(b)* Función de desempeño de distancia.

Fig. 6.1: **Funciones de desempeño para evaluar entrenamientos.** Representación gráfica de las funciones de desempeño utilizadas para evaluar el entrenamiento de caminatas.

f: Frecuencia objetivo [Hz]	1.32
d: Distancia objetivo [m]	1
A: Amplitud	10
σ_f: Desviación estándar de la frecuencia	0.7
σ_d: Desviación estándar de la distancia	0.4

Tab. 6.1: **Parámetros usados para las funciones de desempeño.**

$$\mathcal{F} = \min(\mathcal{F}_{dist}, \mathcal{F}_{frec}) \tag{6.4}$$

Finalmente el *fitness* de la iteración es determinado por el valor mínimo entre la función de desempeño de distancia y de frecuencias (ecuación [6.4]), esto permite forzar a las redes CPPN a mejorar el peor aspecto de cada ejercicio a medida que transcurren las generaciones en el entrenamiento. Cada valor de la función de desempeño es asignado al organismo correspondiente y una vez la población completa sea evaluada se avanza una generación, hasta completar la totalidad de iteraciones definidas por el usuario.

6.4. ENTRENAMIENTO DE REDES ES-HYPERNEAT

Un entrenamiento cuenta con una población de organismos que describen redes neuronales o en este caso redes CPPN, cuyo tamaño es definido por el usuario antes comenzar el ejercicio. Los organismos deben evolucionar haciendo uso del algoritmo NEAT a lo largo de una cantidad de generaciones determinada antes de comenzar el entrenamiento. En la generación cero las redes CPPN son inicializadas con una topología mínima de nodos de entrada conectados cada uno de ellos a los nodos de salida, cuyos pesos son aleatorizados. Cada organismo genera un patrón de conectividad en un substrato ES-HyperNEAT (siguiendo el método del algoritmo explicado en el capítulo 4), el cual es evaluado en la simulación actual y se determina el puntaje de desempeño obtenido según sea el caso. Una vez evaluada la población total se avanza de generación y las redes CPPN son evolucionadas por el algoritmo NEAT según se ha descrito en el capítulo 3. Esto se realiza para todos los organismos de una población y tantas veces como la cantidad de generaciones lo indique, por ejemplo si se define una población de 100 organismos y una cantidad de 100 generaciones quiere decir que la población completa debe ser evaluada cien veces, es decir 10000 iteraciones. Este proceso se realiza para cualquier ejercicio que involucre el aprendizaje de máquina a partir del algoritmo ES-HyperNEAT. A continuación se describe el proceso adoptado para la codificación de los entrenamientos de caminatas.

6.5. PROGRAMA DE ENTRENAMIENTO DE CAMINATAS

El programa de entrenamiento está codificado en C/C++ y posee etapas diferenciadas para su fácil uso y comprensión. El primer punto a abordar es el uso de hilos del procesador para el entrenamiento, esto pues dado que cada simulación toma 6 segundos y al realizar miles de evaluaciones se estima una extensión mínima de tiempo (para una población de 100 organismos y 100 generaciones) de aproximadamente 17 horas, por lo tanto la codificación del programa está pensada para aprovechar el

procesamiento paralelo de simulaciones a partir de hilos (*threads*), de esta forma el tiempo de entrenamiento se divide tantas veces como hilos sean utilizados. Luego se evidencian dos etapas que componen el programa final:

- **Inicialización de plataformas robóticas, redes y substratos:** En primera instancia se hace uso de la librería RobotLib para inicializar todos los objetos a utilizar en el entorno de simulación, que como fue descrito en secciones previas se cuenta tanto de estructuras físicas como controladores para el correcto manejo de las plataformas robóticas. Esto es replicado tantas veces como hilos de procesamiento se desee utilizar. Luego es inicializada una población de 100 organismos de redes CPPN de topología mínima y pesos aleatorios en sus conexiones. Finalmente y al igual que con RobotLib, se instancia tantas veces como hilos existan objetos ES-HyperNEAT y NeuralNetwork, donde el primero a partir de los patrones de conectividad que presenta el organismo CPPN evaluado modifica al segundo, el cual es utilizado para encontrar los estados de las extremidades en la simulación ejecutada.

- **Entrenamiento a lo largo de las generaciones:** Dependiendo de cuántos simuladores se inicialicen a partir de la cantidad de hilos a utilizar, se divide la población de forma que se distribuya equitativamente los organismos sobre cada simulador. Es decir si son inicializados dos simuladores para la población de tamaño 100, cada uno evaluará 50 organismos por generación.

  A medida que se itera sobre los organismos de la población, las redes CPPN construyen un substrato haciendo uso de ES-HyperNEAT el cual es utilizado para ejecutar y evaluar las caminatas efectuadas por cada robot. La configuración de nodos de entrada y salida varía para cada robot y será abordada de forma específica en las secciones a continuación. Una vez simulada la caminata de 6 segundos se obtiene el desempeño a través de la evaluación de la función *fitness*, y es asignado al organismo actual, y una vez que toda la población haya sido revisada se aplica el método *Epoch* de NEAT, evolucionando y reorganizando

las especies de redes CPPN, dando paso a una nueva generación de organismos. A medida que transcurre el entrenamiento es esperable que las redes CPPN produzcan patrones de conectividad en el substrato cuyo efecto en los robots sea generar caminatas que apunten a la solución del problema, es decir alcanzar la máxima distancia posible y que sus extremidades posean movimientos oscilatorios en dicha caminata. Una vez hayan transcurrido 100 generaciones se da por finalizada la rutina de aprendizaje.

Para evaluar el desempeño de los organismos haciendo uso de ES-HyperNEAT se someterá a entrenamientos a los robots presentados en el primer capítulo de esta memoria: ArgoV2 y Quadratot (fig. 1.1) con 12 y 9 grados de libertad respectivamente. El primero fue diseñado por un ex-estudiante Cristian Osorio Méndez de la carrera de Ingeniería en Diseño de Productos[2] de la Universidad Técnica Federico Santa María, mientras que Quadratot fue diseñado por el Dr. Juan Cristóbal Zagal[3] de la Universidad de Chile con el objetivo de ser utilizado para evaluar la generación de caminatas en robots móviles.

Además se realizarán entrenamientos con HyperNEAT y τ-HyperNEAT bajo las mismas condiciones de entrenamiento que ES-HyperNEAT (6.2) y haciendo uso de las mismas funciones de desempeño, para obtener un criterio de evaluación equitativo para todos los métodos. Además se aplican los umbrales de pesos para todos los algoritmos y retardos temporales, además de los parámetros necesarios para la ejecución de ES-HyperNEAT (6.3). En las siguientes secciones se muestran los resultados obtenidos para cada robot entrenado.

[2] `http://www.idp.usm.cl/`
[3] `http://www.jczagal.com/`

Número de redes CPPN por generación	100
Número de generaciones	100
Probabilidad de reproducción entre especies	0.001
Probabilidad de añadir un nuevo nodo para especies de tamaño grandes	0.0008
Probabilidad de añadir un nuevo nodo para especies de tamaño pequeñas	0.0006
Probabilidad de añadir una nueva conexión para especies de tamaño grandes	0.0008
Probabilidad de añadir una nueva conexión para especies de tamaño pequeñas	0.0009
Probabilidad de cambiar el peso de una conexión	0.3
Porcentaje de descendencia obtenida solo por mutaciones (sin entrecruzamientos)	25
Probabilidad de cambiar la función de activación de un nodo	0.1
Umbral de distancia entre especies	4
Umbral de supervivencia	0.3
Factor de elitismo	0.05

Tab. 6.2: **Parámetros de evolución en NEAT.** Una especie es considerada grande cuando la cantidad de organismos pertenecientes es mayor a 10.

	ArgoV2	Quadratot
$\rho_{\text{mín}}$	0.00833	0.0111
$\omega_{\text{máx}}$	0.0833	0.111
$\tau_{\text{máx}}$	10	10
d_t	0.08	0.08
σ_t	0.007	0.007
β_t	0.07	0.07
r_t	8	8
$r_{\text{máx}}$	10	10
i_t	1	1

Tab. 6.3: **Parámetros usados para la implementación de HyperNEAT, τ-HyperNEAT y ES-HyperNEAT**

6.6. QUADRATOT

A continuación se presentan los resultados obtenidos en entrenamientos de caminatas sobre la plataforma robótica QUADRATOT. En la fig. 6.2 se muestra la estructura que posee y la distribución y numeración de sus motores, que será utilizada para definir el substrato de la red.

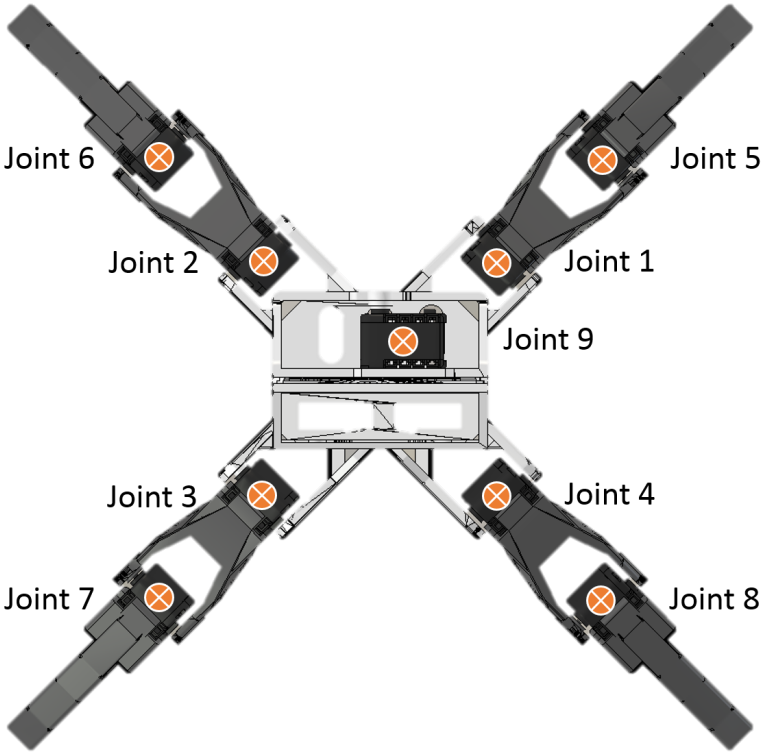


Fig. 6.2: **Diagrama de motores de Quadratot.** Distribución y numeración de motores de dicho robot

En la fig. 6.3 se muestra la distribución de nodos de entrada y salida en el substrato a partir de cómo han sido posicionado los motores en la plataforma robótica. Además se agregan como entradas dos nodos con señales senoidales desfasadas en $90°$, que producirán oscilaciones en los estados siguientes de cada motor. A partir de la descripción del substrato y del programa de entrenamiento la población de organismos evoluciona a medida que se obtienen fitness que indiquen un buen resultado en las caminatas.

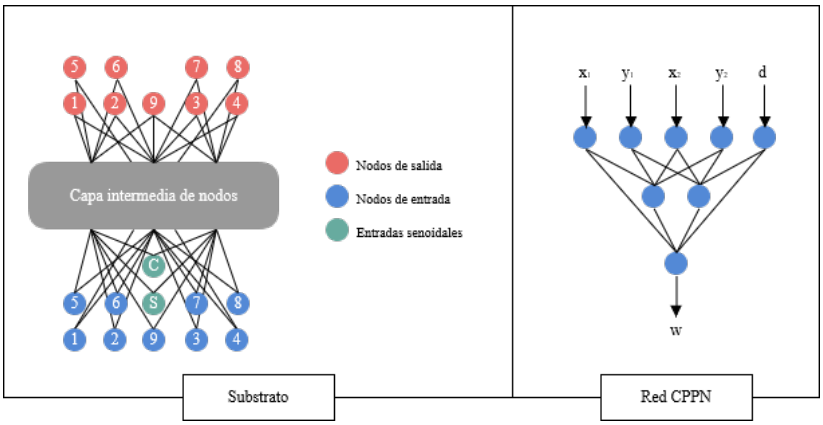


Fig. 6.3: **Diagrama de nodos en substrato de Quadratot.** De acuerdo a la distribución de motores en la estructura del robot se definen los nodos de entrada y salida del substrato. Se muestra un diagrama representativo de la red CPPN a utilizar para construir el substrato.

En la fig. 6.4 se muestra la red CPPN que construye el substrato de solución en el entrenamiento de caminatas. Se observa una topología mínima, sin nodos adicionales ni conexiones entre los que ya existen. Sin embargo las funciones de activación en nodos de entrada han sufrido mutación desde el algoritmo NEAT cambiando desde un par de funciones sigmoides a dos gaussianas que colaboran en la producción de simetrías en el substrato, mientras que el nodo de salida presenta un cambio de función de activación a una función seno, que permite la aparición de repeticiones en el patrón de conectividad generado.

El substrato obtenido como solución en los entrenamientos se muestra en la fig. 6.5. Si se divide justo a la mitad el plano de forma vertical se puede observar un patrón simétrico imperfecto, teniendo el lado izquierdo del substrato una mayor densidad de nodos. Esto es posible debido al patrón de conectividad generado por la red CPPN cuya composición de funciones ha dispuesto nodos de la capa intermedia de la forma en que se exhibe en la fig. 6.5, sin embargo en este caso puede no estar sujeto de forma absoluta a la expresión de la red CPPN en el plano bidimensional, pues el posicionamiento de los nodos también involucra el valor de los parámetros escogidos para la ejecución de ES-HyperNEAT, donde una resolución mayor de búsqueda de nodos puede aumentar aún más su densidad en el espacio dependiendo además del

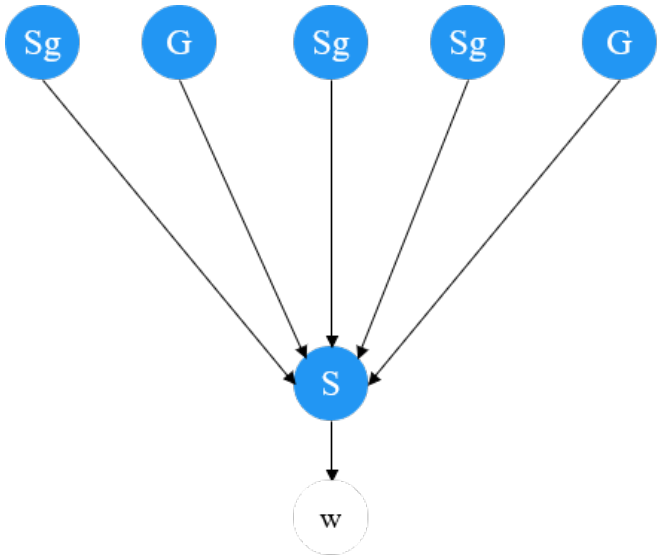


Fig. 6.4: Red CPPN campeona en entrenamiento de caminatas en Quadratot con ES-HyperNEAT.

resto de variables que discriminan la redundancia de información.

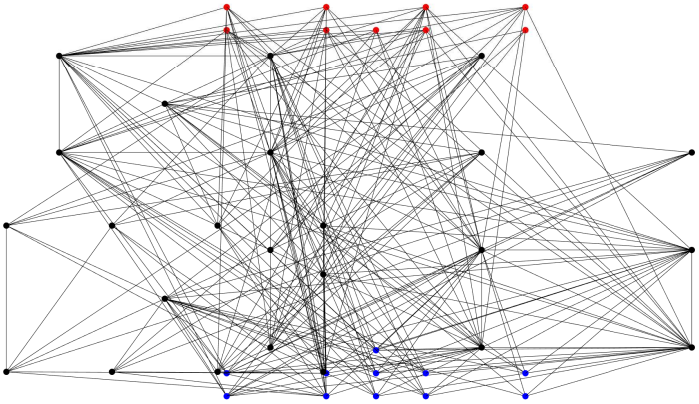


Fig. 6.5: **Substrato obtenido en entrenamiento con ES-HyperNEAT para el robot Quadratot.**

El substrato creado es una de las soluciones que ha encontrado el entrenamiento, por ende ha resultado en caminatas exitosas. En la fig. 6.6 se muestra los movimientos realizados por los motores del robot en la caminata que representa la solución. Según la figura se organizan las gráficas por extremidad, donde las primeras 4 filas corresponden a cada una, notando que en dos de las extremidades sus articulaciones sufren

cambios de fases en sus señales, mientras que las dos restantes presentan señales con fases aproximadas. Adicionalmente en la fig. 6.7 se grafica los movimientos de los motores de las extremidades, esta vez respecto a su posición en las mismas, es decir si se encuentran más cerca al cuerpo del robot (motor interno) y más alejados (motor externo), mostrando que los motores internos de las patas se encuentran en fase en grupos de a pares, pero cada uno de estos se desfasa casi en 90° respecto al otro, mientras que los motores externos de todas las patas poseen fases aproximadas. Se destaca además el motor que está anclado al torso del robot, cuya gráfica se ubica en la última fila de la figura, éste presenta unos picos a lo largo de su señal, esto puede ser un efecto provocado por el movimiento del resto de las extremidades del robot que provocaron un incremento en el ángulo del motor central.

Junto con el movimiento de cada motor se grafica además el desplazamiento del robot tomando en cuenta los ejes x e y en el espacio de trabajo (fig. 6.8). Es posible notar que el robot evidentemente ha podido desplazarse en el plano xy, mostrando poca fluidez principalmente por los movimientos realizados a lo largo del eje y, sin embargo una posible explicación es que los movimientos aprendidos por el robot a lo largo del entrenamiento provocan que a medida que avanza posea inclinaciones que son mucho más significativas a lo largo del eje y, tal como se aprecia en el gráfico superior derecho de la fig. 6.8.

Finalmente es necesario observar la evolución de la función de desempeño a lo largo del entrenamiento. Para esto se toma en cuenta el resultado promedio de desempeño de la población total y se grafica respecto a la generación respectiva. La función *fitness* sirve como un criterio cuantitativo de comparación entre algoritmos neuroevolutivos, es por esto que se ejecutaron entrenamientos de HyperNEAT y τ-HyperNEAT, cuyas funciones de desempeño son graficadas junto a la de ES-HyperNEAT para notar cuantitativamente si es que existe una mejoría o algún comportamiento importante a destacar. En la figura 6.9 se muestra los gráficos de las funciones de desempeño de los tres algoritmos utilizados, pudiendo notar un crecimiento importante al comienzo del ejercicio de HyperNEAT, mientras que ES-HyperNEAT y τ-HyperNEAT mues-

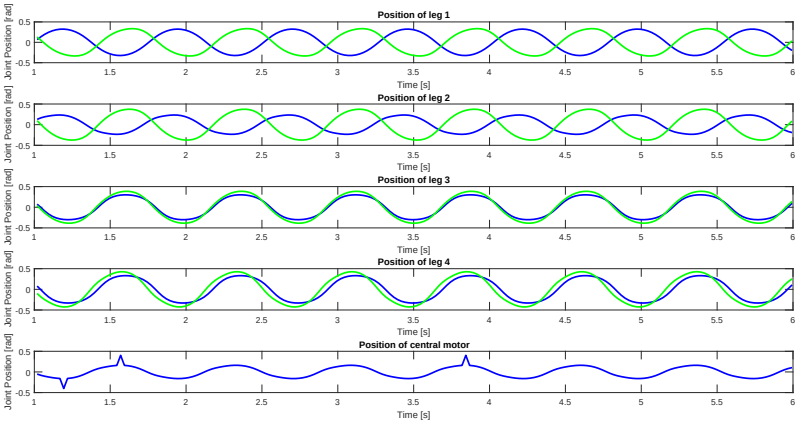


Fig. 6.6: **Posición de los motores a lo largo de la caminata en Quadratot.** Los motores presentan comportamientos oscilatorios en sus movimientos. Las primeras 4 filas representan extremidad y las gráficas de sus dos motores superpuestas (**azul:** motor interior, **verde:** motor exterior). La última fila representa el movimiento del motor ubicado en el cuerpo del robot.

tran un ascenso más lento, sin embargo a medida que el entrenamiento transcurre ES-HyperNEAT presenta un mejor desempeño promedio (aunque no de forma excesiva) de su población presentando incluso una mejoría cerca del final del ejercicio. HyperNEAT y τ-HyperNEAT cuentan con una distribución de nodos de la capa intermedia predefinida por el usuario por lo que muestra resultados más variados al comienzo del entrenamiento de caminatas, mientras que ES-HyperNEAT comienza con un substrato sin nodos en la capa intermedia, significando que en las primeras generaciones no exista mucha diferencia en el desplazamiento de los organismos evaluados, salvo aquellas redes CPPN que destacan y permiten la evolución del resto de la población.

La aleatoriedad que representan estos métodos evolutivos puede provocar que algunos entrenamientos encuentren soluciones más rápido que otros, por lo que es necesario observar una gran cantidad de ejercicios en promedio para poder tener un criterio de evaluación aún más eficiente. De hecho en [5] se comprobó que las funciones de desempeño de HyperNEAT y τ-HyperNEAT evolucionaban de forma muy similar en los entrenamientos, sin embargo la diferencia más notable fue en el movimiento del robot pues los retardos temporales en las conexiones del substrato permitieron generar

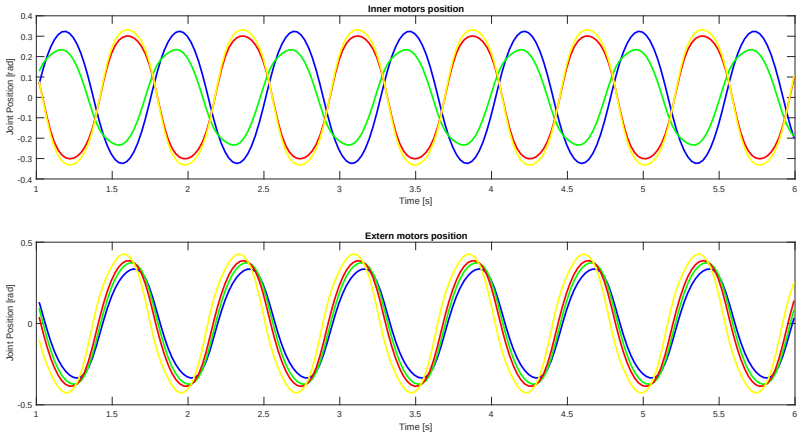


Fig. 6.7: **Posición de los motores por posición interna o externa en extremidades en Quadratot.** Gráficos superpuestos de movimiento de motores respecto a los roles por extremidad (**azul:** extremidad 1, **verde:** extremidad 2, **rojo:** extremidad 3, **amarillo:** extremidad 4).

movimientos más naturales en las caminatas de τ-HyperNEAT.

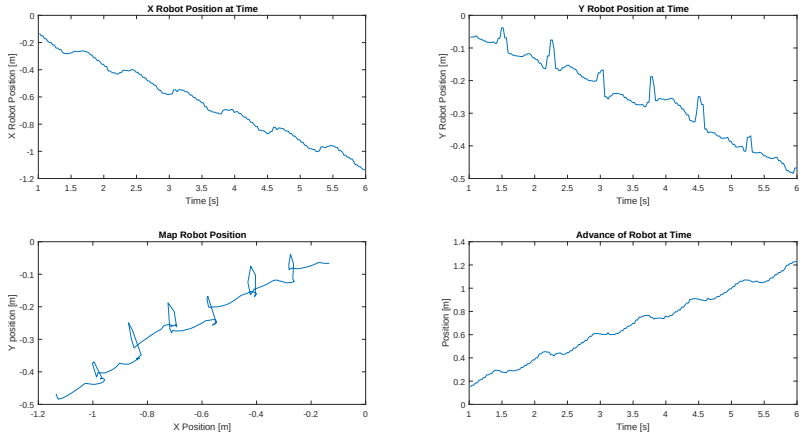


Fig. 6.8: **Desplazamiento de Quadratot en el escenario de entrenamiento** El gráfico superior izquierdo muestra la variación del desplazamiento en el eje x respecto al tiempo, análogamente el gráfico superior derecho toma en cuenta al eje y. El gráfico inferior izquierdo muestra el desplazamiento del Quadratot sobre el plano xy en el tiempo, mientras que el plano inferior derecho muestra la distancia alcanzada por el robot a lo largo de la simulación.

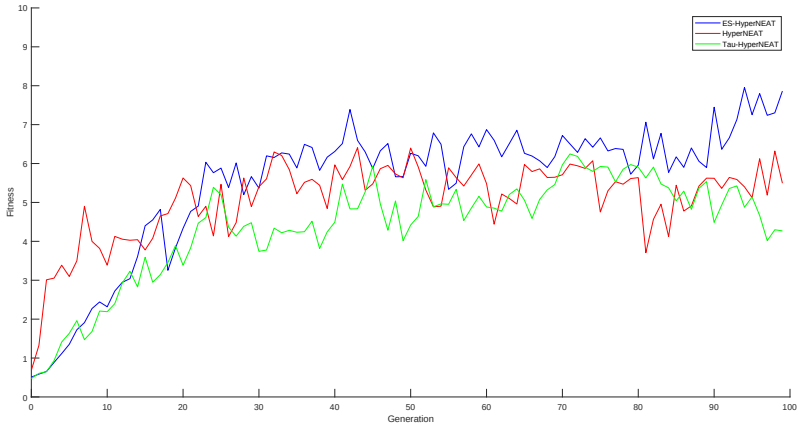


Fig. 6.9: **Función de desempeño promedio de los algoritmos evaluados.** El gráfico muestra la superposición de los gráficos de funciones de desempeño de los algoritmos utilizados en el entrenamiento de caminatas del Quadratot. Los gráficos de color azul, rojo y verde corresponden a los *fitnesses* de ES-HyperNEAT, HyperNEAT y τ-HyperNEAT respectivamente.

6.7. ARGOV2

A continuación se presentan los resultados obtenidos en entrenamientos de caminatas sobre la plataforma robótica ARGOV2. En la fig. 6.10 se muestra la estructura que posee y la distribución y numeración de sus motores, que será utilizada para definir el substrato de la red.

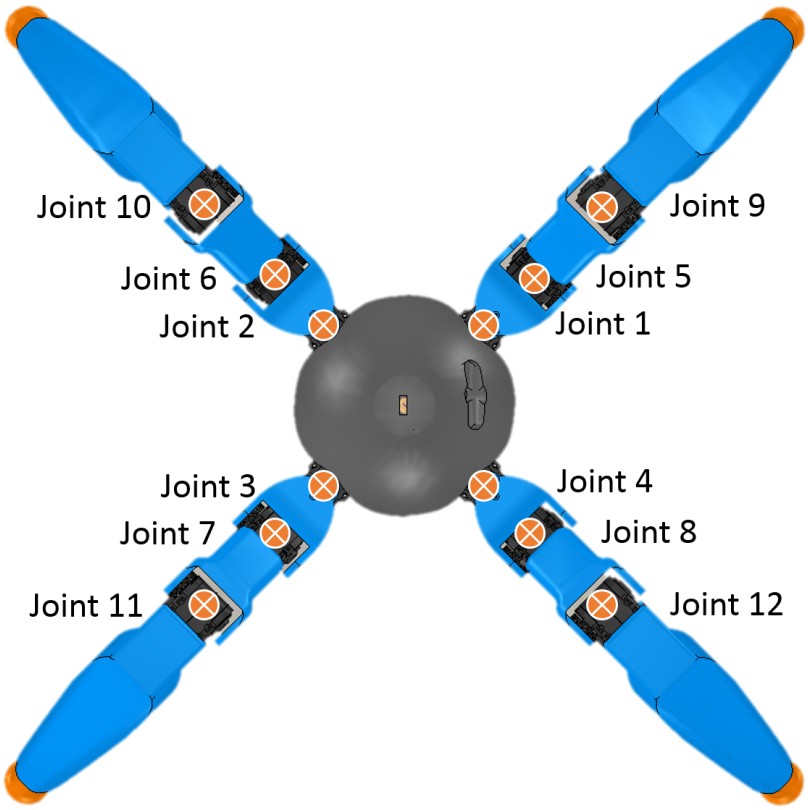


Fig. 6.10: **Diagrama de motores de ArgoV2.** Distribución y numeración de motores de dicho robot.

A partir de la configuración física de motores en la plataforma ArgoV2 se distribuyen los nodos de entrada y salida del substrato como se muestra en la fig. 6.11. A diferencia de Quadratot, ArgoV2 posee un grado de libertad extra en cada extremidad pero no posee un motor en su cuerpo, lo que generará diferencias en cuanto a los nodos de la capa intermedia que aparecerán en el substrato a medida que el entrenamiento progrese.

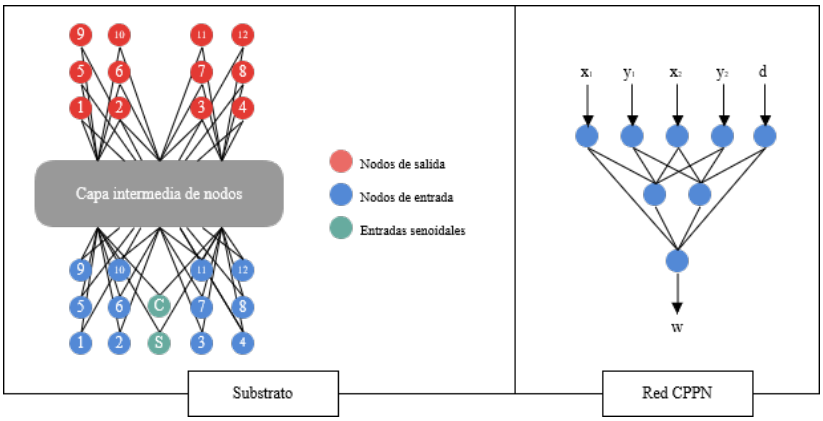


Fig. 6.11: **Diagrama de nodos en substrato de ArgoV2.** De acuerdo a la distribución de motores en la estructura del robot se definen los nodos de entrada y salida del substrato. Se muestra un diagrama representativo de la red CPPN a utilizar para construir el substrato.

En la fig. 6.12 se muestra la red CPPN que construye el substrato de solución en el entrenamiento de caminatas. El algoritmo encontró en la red CPPN que no fue necesario incorporar nodos pero si adicionar un par de conexiones entre las entradas, además de realizar mutaciones de funciones de activación en una de las entradas y en la salida, pasando desde funciones sigmoides a senoidales, que permiten la aparición de repeticiones en el patrón de conectividad generado.

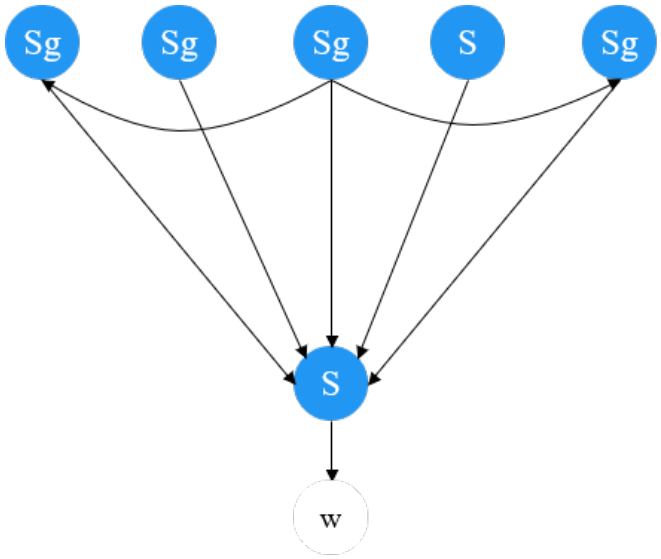


Fig. 6.12: Red CPPN campeona en entrenamiento de caminatas en ArgoV2 con ES-HyperNEAT.

El substrato obtenido como solución en los entrenamientos se muestra en la fig. 6.13. En este caso también se ha obtenido una simetría imperfecta pero con mucho más similitudes que el substrato de solución del Quadratot (fig. 6.5) entre las zonas izquierda y derecha, donde la primera posee un nodo más que la izquierda. Es posible observar que los patrones de conectividad son eficientemente utilizados y que es posible explotar la información implícita en la red CPPN sobre el patrón de conectividad que debe ser generado.

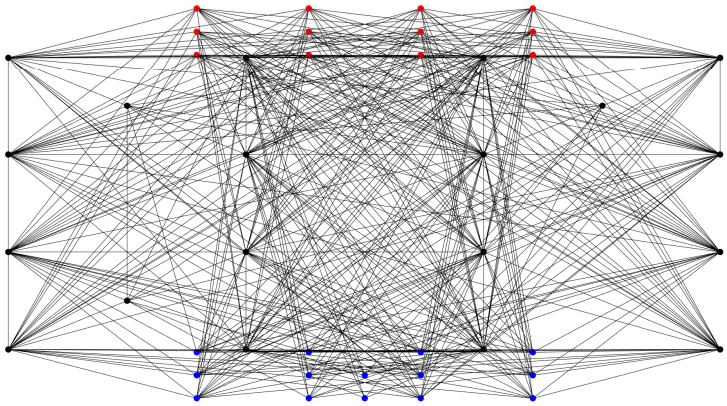


Fig. 6.13: **Substrato obtenido en entrenamiento con ES-HyperNEAT.**

A partir de la solución encontrada por el algoritmo, se grafica las señales de los motores a lo largo de la caminata generada. En la fig. 6.14 se muestra lo anterior, pudiendo notar los movimientos oscilatorios obtenidos en cada una de las extremidades, donde las extremidades 1 y 3 muestran cambios de fase en uno de sus motores, y las extremidades 3 y 4 tienen todos sus motores en fase. Se muestra además en la fig. 6.15 el comportamiento de los motores respecto a su posición en las extremidades. En este caso cada pata tiene 3 grados de libertad pudiendo observar que la solución encontrada muestra que los motores internos de la extremidad 4 se encuentra en desfase de $90°$ respecto al resto, mientras que los motores medios de las extremidades 1 y 2 se encuentran en fase pero en desfase con los de las extremidades 3 y 4, coincidiendo estas dos últimas. Finalmente los motores externos se encuentran en fase salvo en la

extremidad 2.

Adicionalmente se grafica el desplazamiento sobre el plano xy y el efecto en cada uno de los ejes que lo componen. Lo anterior se muestra en la fig. 6.16 evidenciando a un desarrollo relativamente fluido a lo largo de los ejes x e y, pudiendo observar a partir del gráfico inferior izquierdo de la figura un desplazamiento que busca tender a lo lineal sobre el plano xy.

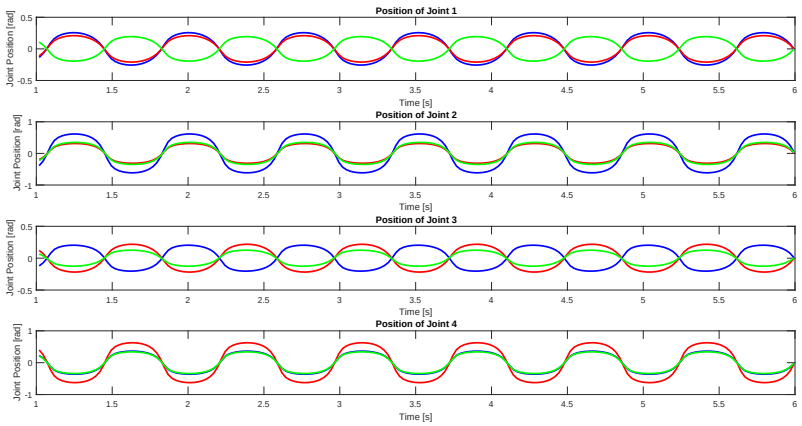


Fig. 6.14: **Posición de los motores a lo largo de la caminata.** Los motores presentan comportamientos oscilatorios en sus movimientos. Las 4 filas representan cada una delas extremidades con gráficos de sus motores superpuestos (**azul:** motor interior, **rojo:** motor medio, **verde: motor externo).**

La fig. 6.17 muestra las funciones de desempeño promedio obtenidas de los entrenamientos con HyperNEAT, τ-HyperNEAT y ES-HyperNEAT. Es posible notar que los tres algoritmos logran buenos resultados en términos de desempeño a lo largo del ejercicio. En las primeras generaciones HyperNEAT demuestra una ventaja incrementando rápidamente su desempeño promedio, tal como sucedió en el entrenamiento de Quadratot, pudiendo notar que el desempeño ES-HyperNEAT crece de forma sustancial una vez comienza a encontrar la solución. Un aspecto positivo por notar de ES-HyperNEAT es que mantiene un desempeño promedio relativamente constante a partir de la generación 20 indicando la consistencia en el uso de este algoritmo para la resolución de aprendizaje de caminatas.

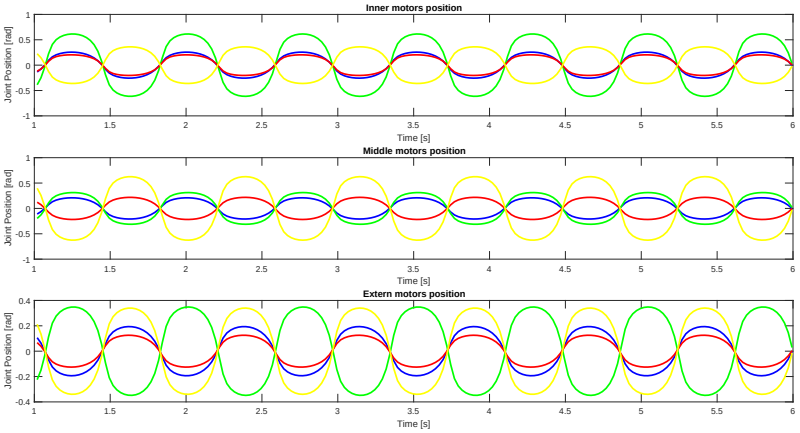


Fig. 6.15: **Posición de los motores por posición interna o externa en extremidades en ArgoV2.** Gráficos superpuestos de movimiento de motores respecto a los roles por extremidad (**azul:** extremidad 1, **verde:** extremidad 2, **rojo:** extremidad 3, **amarillo:** extremidad 4).

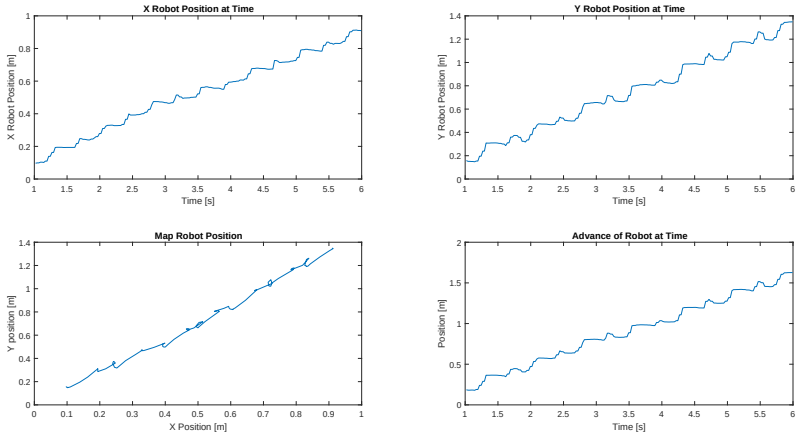


Fig. 6.16: **Desplazamiento de ArgoV2 en el escenario de entrenamiento.** Los gráficos superiores izquierdo y derecho muestran el desplazamiento del robot sobre los ejes x e y en el tiempo, respectivamente. El gráfico inferior izquierdo muestra el desplazamiento del ArgoV2 sobre el plano xy en el tiempo y el plano inferior derecho muestra la distancia lograda por el robot a lo largo de los 6 segundos de simulación.

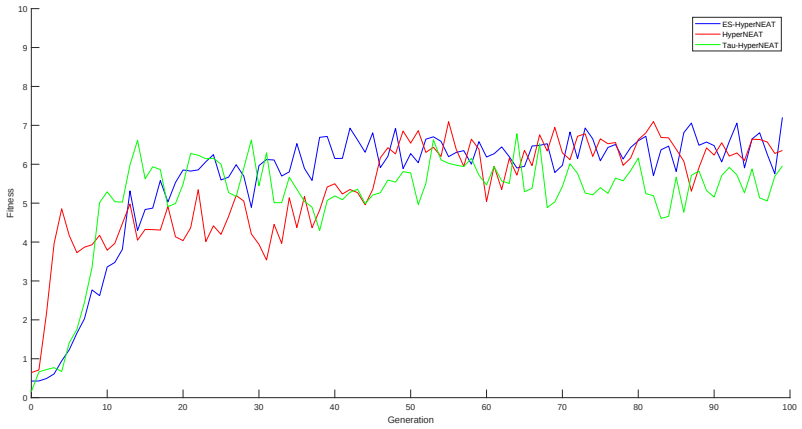


Fig. 6.17: **Función de desempeño promedio de los algoritmos evaluados en ArgoV2.** Se superponen los gráficos de desempeño promedio de los algoritmos utilizados para el aprendizaje de caminatas en ArgoV2. Los gráficos de color azul, rojo y verde corresponden a los *fitnesses* de ES-HyperNEAT, HyperNEAT y τ-HyperNEAT respectivamente.

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1. DISCUSIÓN SOBRE RESULTADOS OBTENIDOS

En este trabajo de memoria se ha implementado el método neuroevolutivo ES-HyperNEAT, desde la motivación de entrenamientos de aprendizaje de caminatas realizados con algoritmos predecesores a este método. Precisamente este trabajo de memoria ha establecido la búsqueda de elementos comparativos entre HyperNEAT, τ-HyperNEAT y ES-HyperNEAT, centrándose principalmente en la forma en que los substratos determinan los movimientos a realizar por las plataformas robóticas y cómo se determina la conectividad entre las capas asociadas.

A partir de los resultados obtenidos en las caminatas aprendidas es posible determinar un par de conclusiones en términos comparativos:

1. Se han encontrado diversos trabajos que han hecho uso de HyperNEAT como herramienta de aprendizaje para la generación de caminatas en robots móviles, encontrándose con ningún uso de ES-HyperNEAT para estos propósitos. Evidentemente los resultados con ES-HyperNEAT son positivos pero no han demostrado una mejoría sustancial en términos cuantitativos, esto no quiere decir que el algoritmo no sirva como herramienta neuroevolutiva para la generación de caminatas, pues hay una serie de parámetros cuyas modificaciones representaron una amplia gama de resultados, pero bajo las condiciones que han sido presentadas ES-HyperNEAT ha demostrado resultados positivos sostenidos en el tiempo con un amplio espacio de mejora.

2. Los parámetros de ES-HyperNEAT son de alta prioridad para generar diversi-

dad de soluciones en los entrenamientos. Por ejemplo el parámetro r_t que indica la división en *quadtrees* del espacio donde se posiciona el substrato provocó que ante valores pequeños ($r_t = 3$) se lograra que la función de desempeño escalara rápidamente a valores cercanos al máximo pero obteniendo caminatas que visualmente eran poco naturales, mientras que al usar valores un tanto más elevados ($r_t = 8$) pudiese aumentar la densidad de nodos que puedan posicionarse en el substrato generando caminatas visualmente atractivas pero con un crecimiento mucho más lento en términos de función de desempeño.

3. ES-HyperNEAT ha resultado una efectiva herramienta para la generación de caminatas, pero carece de elementos que aporten el aprendizaje de caminatas armónicas y naturales, como la incorporación de retardos temporales en conexiones del substrato hecho por τ-HyperNEAT. Si bien ES-HyperNEAT ha mostrado ser levemente superior que τ-HyperNEAT en términos de desempeño, el segundo ha logrado entrenar caminatas cuyos movimientos se asemejan más a movimientos naturales hechos por insectos en la naturaleza por ejemplo, aunque todo está sujeto a percepciones. Posiblemente la aplicación de retardos temporales en un substrato que evolucione en el tiempo traiga resultados aún más positivos que los ya encontrados.

Queda un amplio espacio de exploración con la herramienta ES-HyperNEAT. En su estado actual es posible realizar diversas variaciones en sus parámetros que permitirán una diversidad de soluciones a los problemas planteados, asimismo hay una gran cantidad de adiciones disponibles para realizar como trabajo futuro y que posiblemente permitirán encontrar soluciones cuyo camino es el aprovechamiento de la geometría del problema a resolver.

7.2. TRABAJO FUTURO

Se han detectado algunos cambios a implementar sobre los métodos utilizados en este proyecto, tanto sobre la codificación y teoría del algoritmo ES-HyperNEAT como modificación sobre el método de entrenamiento de caminatas aplicado.

- Formular y probar nuevas funciones de desempeño para lograr clasificar de manera óptima la correcta ejecución de una caminata. Junto con esto aprovechar las distintas configuraciones de parámetros de ES-HyperNEAT para observar la evolución de soluciones obtenidas.

- Poder probar los resultados obtenidos en el aprendizaje en plataformas reales, entendiendo que el comportamiento real de motores es considerablemente distinto al que se aprecia en entornos simulados. En [11] se ha presentado el comportamiento de motores *Dynamixel* sobre el trabajo en robots articulados que participan en la competencia *RoboCup* y han desarrollado un firmware para el uso de los mismos motores. Es importante contar con las consideraciones de hardware asociadas al uso de este tipo de motores por lo que un trabajo dedicado en esta área es necesario para el futuro de la robótica autónoma.

- Incorporar retardos temporales en las conexiones de ES-HyperNEAT para comprobar el comportamiento de las caminatas aprendidas en entrenamientos con este nuevo algoritmo.

- Implementar distintas configuraciones de substrato en ES-HyperNEAT. HyperNEAT presenta un rápido crecimiento en los resultados de la función de desempeño haciendo uso de substratos tipo *state-space sandwich*, por tanto es necesario aplicar soluciones de este tipo con substratos generados a partir de ES-HyperNEAT y comprobar el comportamiento de sus soluciones en entrenamiento de caminatas.

- Incorporar variabilidad temporal a lo largo de los entrenamientos en parámetros

como r_t e i_t, el primero para aumentar o disminuir la resolución del substrato en la búsqueda de soluciones y el segundo permitirá observar topologías similares a algoritmos de *deep learning* en caso de incrementos. Es posible sofisticar el algoritmo de ES-HyperNEAT para que adapte sus resoluciones según lo requiera el problema.

Bibliografía

[1] Stanley, K.O., Miikkulainen, R, Evolving neural networks through augmenting topologies. Evolutionary Computation 10 (2), (2002), 99?127.

[2] STANLEY, K.O.(2007). "Compositional pattern producing networks: A novel abstraction of development". Genetic Programming and Evolvable Machines Special Issue on Developmental Systems, 8(2):131?16.

[3] Stanley, K.O., D?Ambrosio, D.B., Gauci, J., A hypercube-based encoding for evolving large-scale neural networks. Artificial Life 15 (2), (2009), 185?212

[4] Caamaño, P., Bellas, F., Duro, R., τ-NEAT: Initial experiments in precise temporal processing through neuroevolution, International Joint Conference on Neural Networks (2014)

[5] Silva O., Sigel P. and Escobar M.J. (2017). Time Delays in a HyperNEAT Network to Improve Gait Learning for Legged Robots. Oral Presentation. 30th International Joint Conference in Neural Networks (IJCNN2017), Anchorage-Alaska, USA.

[6] Sebastian Risi and Kenneth O. Stanley. An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density and Connectivity of Neurons In: Artificial Life journal. Cambridge, MA: MIT Press, 2012

[7] Haasdijk E., Rusu A.A., Eiben A.E. (2010) HyperNEAT for Locomotion Control in Modular Robots. In: Tempesti G., Tyrrell A.M., Miller J.F. (eds) Evolvable Systems: From Biology to Hardware. ICES 2010. Lecture Notes in Computer Science, vol 6274. Springer, Berlin, Heidelberg

[8] Virtual Robot Experimentation Plataform, Coppelia Robotics, Switzerland. http://www.coppeliarobotics.com/

[9] CHURCHLAND, P.M. (1986). Some reductive strategies in cognitive neurobiology. Mind, 95:279-309.

[10] R. Finkel and J. Bentley. Quad trees: A data structure for retrieval on composite keys. Acta informatica, 4(1):1(9), 1974.

[11] Rémi Fabre, Quentin Rouxel, Grégoire Passault, Steve N'Guyen, Olivier Ly. Dynaban, an Open-Source Alternative Firmware for Dynamixel Servo-Motors (2016)