Navigation of Mobile Robots Using Neural Networks and Genetic Algorithms

David Abad Pérez, Dr. Basil Mohammed Al-Hadithi and Victor Cadix Martín

***Abstract*—The navigation of robots has been a subject of widespread interest over the last few decades. In the previous years, traditional methods based on mathematical equations were used, and there has been an evolution towards the use of methods based on artificial intelligence. Two of which have been used in this work: neural networks and genetic algorithms. Neural networks are used as a machine learning model to teach the robot to move from any starting point to a goal, avoiding obstacles along the way. However, this model needs an algorithm to learn how to carry out this activity, which is what the genetic algorithm will be used for. Furthermore, this method of navigation will be compared with the traditional method based on potential fields, where it can be observed how this new method based on artificial intelligence improves and solves some typical problems of the old methods, such as the tendency to get stuck in local minima.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/8506

***Index Terms*—Navigation, Robots, Artificial intelligence, Neural networks, Genetic algorithms, Potential fields, Local Minima.**

I. Introducción

La navegación robótica ha permitido la aparición de grandes avances en la actualidad. Sin embargo, la variedad de técnicas es tan grande que ha ido evolucionado desde modelos matemáticas como el álgebra lineal [1], el gradiente descendente [2] o, la teoría líder-seguidor [3], hasta técnicas basadas en inteligencia artificial [4] como la lógica difusa [5] o las redes neuronales.

Este trabajo utiliza esta última técnica debido a su potencial para revelar relaciones complejas y no lineales entre datos de entrada y de salida [6], lo que las hace muy útiles en campos como la detección de patrones [7], la economía [8] o la robótica [9], con ejemplos como brazos robóticos [10], robots alarma [11] o la navegación de robots voladores [12], acuáticos [13] o terrestres [14].

Sin embargo, estas redes neuronales necesitan pasar por una fase de aprendizaje antes de poder utilizarse [15], para la cual existen varios algoritmos de aprendizaje como la propagación hacia atrás [16], [17] o los algoritmos genéticos [18]. Se ha escogido esta segunda opción por su capacidad para optimizar cualquier tipo de proceso y, por tanto, por su posibilidad de poder combinarse con las redes neuronales y así poder enseñarlas a entender el entorno que la rodea, perfeccionar su funcionamiento y ajustar sus pesos en consecuencia.

Así, este trabajo estudia la navegación en robots a través de redes neuronales que aprenden a través de algoritmos genéticos y su posterior comparación con la navegación a través de campos potenciales. En el apartado II se explican los algoritmos de navegación, en el apartado III se justifican el algoritmo seguido por el robot y la programación del entorno, en el apartado IV se expone el desarrollo del programa, en el apartado V se enseña el ajuste de los hiperparámetros, en el apartado VI se muestran los resultados y, en el apartado VI se resumen las conclusiones obtenidas.

II. Algoritmos de Navegación

El objetivo es conseguir un algoritmo que permita a los robots aprender de manera autónoma a navegar en cualquier tipo de mapa [19] y, para ello, se debe de recurrir a la inteligencia artificial. Como se ha comentado anteriormente, se ha decidido combinar las redes neuronales y los algoritmos genéticos y, además, compararlo con la navegación a través de campos potenciales, con el objetivo de mostrar la mejora de un método robusto a otro con un bajo índice de éxito en mapas complejos debido a su tendencia a caer en mínimos locales.

A. Campos Potenciales

Los campos potenciales se han escogido por ser el método de navegación clásica por excelencia. Este método calcula el potencial resultante de sumar el campo potencial atractivo generado por la meta (1), el cual es inversamente proporcional de la distancia al robot, y los campos potenciales repulsivos generados por los obstáculos (2), los cuales son directamente proporcionales de la distancia al robot. De esta forma, el objetivo del robot es encontrar el punto de potencial o energía nulo (es decir, la meta).

$$\begin{cases} \nabla U_{att} = \xi * dist(q_{robot}, q_{meta}) \\ \qquad si\ dist(q_{robot}, q_{meta}) < dist_{meta} \\ \nabla U_{att} = \frac{\xi * dist_{meta}}{|mod(q_{robot}, q_{meta}) - r_{robot} - r_{meta}|} * dist(q_{robot}, q_{meta}) \\ \qquad si\ dist(q_{robot}, q_{meta}) \geqslant dist_{meta} \end{cases} \quad (1)$$

D. A. Perez is currently working at Alstom, Pinto, Spain (e-mail: david.abad.0713@gmail.com).

B. M. Al Hadithi is with the Intelligent Control Group, Universidad Politécnica de Madrid, Centre for Automation and Robotics UPM-CSIC. He is also with the School of Industrial Design and Engineering, Universidad Politécnica de Madrid, Madrid, Spain (e-mail: basil.alhadithi@upm.es).

V. C. Martín, is currently working at Perseo Techworks, Madrid, Spain (e-mail: victor.cadix.martin@alumnos.upm.es)

$$\nabla U_{rep} = \frac{2 * r_{robot} * \sigma * dist_{unit}(q_{robot}, q_{obs})}{|mod(q_{robot}, q_{obs}) - 7 * r_{robot} - r_{obs}|} \quad (2)$$

Dónde:

ξ y σ: Constantes del campo potencial atractivo y repulsivo.
dist (q_{robot}, q_{meta}): Vector distancia entre el robot y la meta.
mod (q_{robot}, q_{meta}): Módulo distancia entre el robot y la meta.
$dist_{meta}$: Distancia establecida para pasar de un campo otencial más fuerte a uno más suave.
$dist_{unit}$ (q_{robot}, q_{obs}): Vector unitario distancia entre el robot y el obstáculo.
r_{robot}: Es el radio del robot.
r_{meta}: Es el radio de la meta.
r_{obs}: Es el radio del obstáculo.

B. *Algoritmos Genéticos*

Los algoritmos genéticos [20] están basados en la teoría de la evolución de Darwin la cual dicta que, a través de las generaciones, sobreviven los mejores individuos de una población. En este caso, se trata de encontrar las mejores soluciones a un problema [21]: enseñar a la red neuronal de un robot a aprender a llegar a la meta evitando los obstáculos que se encuentre en su camino. Así, esta técnica ha sido elegida frente a otras por su buena sinergia con estas redes neuronales y por indagar en un campo poco estudiado hasta el momento.

Cada individuo está formado por una cadena de genes [21] (llamada cromosoma), los cuales representan la propia solución potencial al problema (red neuronal) que se está optimizando. Además, cada individuo está representado por un valor de aptitud, el cual indica cuanto de bueno es ese individuo dentro de la población [21]. Esta aptitud viene dada por la función de costo, una función implementada por el creador [22]. Así, el funcionamiento de los algoritmos genéticos se basa en cuatro procesos u operadores genéticos (Fig. 1):

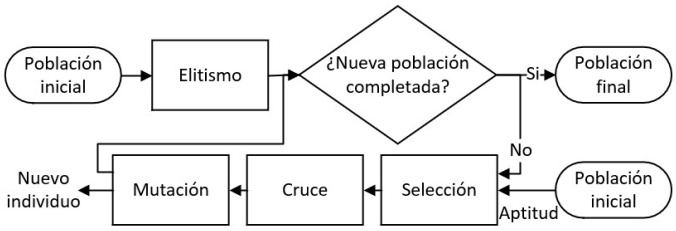


Fig. 1. Diagrama de bloques del funcionamiento de los algoritmos genéticos.

En primer lugar, el elitismo [23] consiste en copiar los mejores individuos de la generación actual a la siguiente con el objetivo de conservar los mejores genes. El segundo paso es la selección por ruleta [24], en el cual se escogen dos individuos padres entre toda la población inicial (cuanta más aptitud tenga el individuo, más posibilidades tiene de ser elegido). El tercer operador genético es el cruce multipunto [24], en el cual se divide en muchas partes los cromosomas de los individuos padres y el individuo hijo hereda alternadamente tramos de genes de cada uno de ellos. Finalmente, la mutación [24] se trata de cambiar aleatoriamente algunos de los genes con el objetivo de obtener nuevas soluciones potenciales al problema.

C. *Redes Neuronales*

Las redes neuronales artificiales [25] son un sistema que procesa información de manera similar al cerebro biológico y están basadas en modelos computacionales que estudian el comportamiento de un sistema a partir de un registro histórico del mismo [26].

Una red neuronal perceptrón multicapa [27] está formada por un conjunto de neuronas agrupadas en capas que pueden ser de tres tipos [28]: entrada (por donde se introducen los datos), salida (por donde se obtienen los resultados) y, ocultas (donde se procesan los datos). Las neuronas de una capa están conectadas con las neuronas de la capa anterior a través de unos enlaces llamadas pesos.

Por otro lado, para que una red neuronal pueda responder de manera automática ante situaciones diferentes a las aprendidas [29], debe de pasar una fase de aprendizaje. En dicha fase [30], la red neuronal utilizará los algoritmos genéticos y aprenderá, gracias a la función de costo, a comprender el entorno que le rodea y a realizar su tarea de la manera más eficaz posible [31]. Así, el funcionamiento de una red neuronal viene dada por tres etapas (Fig. 2):

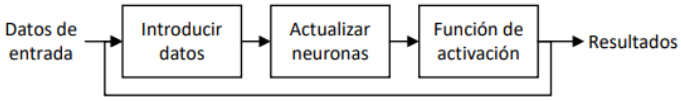


Fig. 2. Diagrama de bloques del funcionamiento de las redes neuronales.

En primer lugar, los datos son introducidos en cada una de las neuronas de la capa de entrada. A continuación [32], se calcula el valor del resto de neuronas (3).

$$F(X) = b_i + \sum_{i}^{n} w_i * n_i \quad (3)$$

Dónde:

F(X): Valor de la neurona.
$\sum w_i * n_i$: Sumatorio de todas las neuronas de la capa anterior (n_i) multiplicadas por los pesos (w_i) que los conectan a dicha neurona.
b: Sesgo de la capa anterior (peso que está conectado a una neurona adicional que siempre vale uno).

Finalmente, a estas neuronas se las aplica una función de activación [33]. Esta función escala los valores de las neuronas y, gracias a ello, ayuda a la red neuronal a entender el funcionamiento del sistema de una manera más sencilla. De esta forma, se obtienen los resultados de la red neuronal a dicha entrada, pero, como los datos de entrada del robot y, por tanto, de la red neuronal, cambian constantemente, este proceso se repite de manera continuada.

En este punto, se puede contemplar que, en una red neuronal, los únicos valores que se pueden modificar son los pesos, convirtiéndolos en el nexo entre las redes neuronales y los genes del algoritmo genético (permitiendo así su optimización) [34]. Así, cada gen del cromosoma de un individuo representa un peso de la red neuronal de un robot [35].

III. Algoritmo Robótico y Programación del Entorno

Este trabajo utiliza un entorno de desarrollo integrado de código abierto que utiliza Java como lenguaje de programación. Este entorno ha sido escogido por su facilidad para desarrollar entornos virtuales a través de figuras geométricas.

A. Robots, Meta y Obstáculos

El objetivo de este apartado es crear los elementos necesarios para poder simular un entorno de navegación: una meta, distintos obstáculos y robots que tengan integrados los dos métodos de navegación. Estos robots (Fig 3.) tienen la opción de navegar a través de campos potenciales (se le otorgará a la meta un campo potencial atractivo y a cada uno de los obstáculos un campo potencial repulsivo) o a través de una red neuronal (la cual indica al robot lo que debe de hacer en todo momento, actuando como si fuera su cerebro).

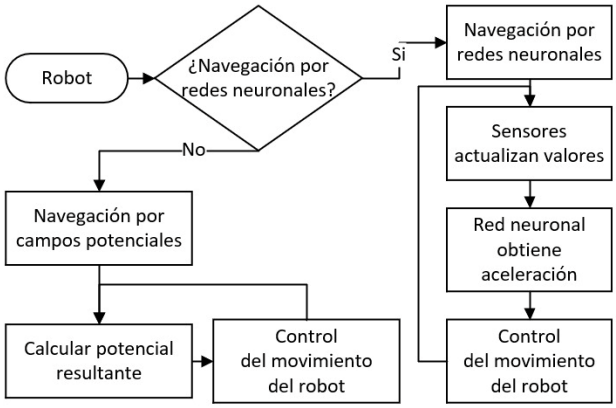


Fig. 3. Diagrama de bloques del funcionamiento del robot.

La navegación a través de redes neuronales se planificará en tiempo real, de manera que los robots disponen de los datos de entrada necesarios para alcanzar la meta y esquivar los obstáculos que se encuentren en su camino: su propia posición, la posición de la meta y la distancia captada por sus sensores al obstáculo más cercano [36]. Por otro lado, la salida de la red neuronal es la aceleración, la cual se deriva dos veces para obtener la posición final del robot. En este momento, se puede observar la relación inexistente entre los datos de entrada y salida y, por tanto, el potencial de las redes neuronales en este tipo de aplicaciones para encontrar relaciones no lineales y establecer una conexión entre ambos datos.

Por otro lado, la navegación a través de campos potenciales también se planifica en tiempo real, aunque el robot sólo debe de calcular la suma de todos los potenciales y, como con las redes neuronales, al ser una aceleración se deriva dos veces para obtener la posición final del robot.

Además, los robots cuentan con tres indicadores para poder examinarlos de una manera más sencilla: distancia recorrida, tiempo transcurrido y energía consumida. Finalmente, los robots genéticos y los robots elitistas están representados por círculos azules y por círculos amarillos respectivamente, la meta está representada por un círculo de color verde y los obstáculos por círculos de color rojo.

B. Mapas

El objetivo de la programación de los distintos mapas es estudiar el comportamiento, las fortalezas y las desventajas de los dos métodos de navegación en distintos ámbitos. Para ello, se ha creado un mapa sin obstáculos, un mapa sencillo con un rectángulo en el medio (Fig. 4), un mapa intermedio con numerosas paredes (Fig. 5) en el que el número y la orientación de estas paredes es aleatoria y, un mapa difícil que simula un laberinto (Fig. 6), el cual está dividido en tres secciones: una rectangular, una circular y una triangular. La posición y orientación de cada una de estas secciones es aleatorio. Además, la posición inicial de los robots y de la meta en los tres primeros mapas es aleatoria, mientas que en el último es fija.

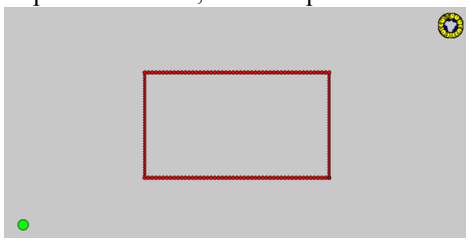


Fig. 4. Fotografía del mapa sencillo.

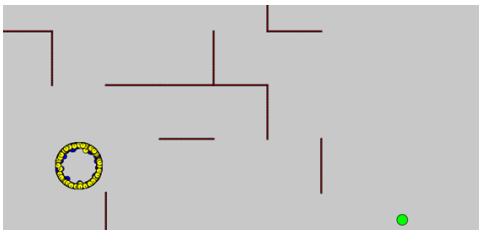


Fig. 5. Fotografía del mapa intermedio.

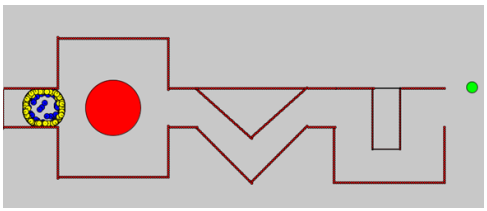


Fig. 6. Fotografía del mapa difícil.

IV. Desarrollo del Programa

El programa se inicializa creando numerosos robots que pueden navegar a través de campos potenciales o a través de redes neuronales. Si los robots navegan con este segundo método, se crea una población con el mismo número de individuos que de robots. Además, estos individuos tienen un cromosoma con el mismo número de genes que pesos de las redes neuronales.

A continuación, se inicia la simulación. Al principio de cada generación se crea el mapa escogido y, en caso de navegar a través de redes neuronales, cada uno de los individuos inserta sus genes en los pesos de la red neuronal de un robot.

Así, los robots se mueven siguiendo el potencial resultante de sumar el campo potencial atractivo con los campos potenciales repulsivos o siguiendo la aceleración proporcionada por las neuronas de la capa de salida de la red neuronal hasta alcanzar los 14 segundos.

En este momento, se finaliza la generación actual y el programa registra los datos más importantes: número de robots genéticos y número de robots elitistas que han alcanzado la

meta, número de robots que se han chocado con un obstáculo y la media de la distancia recorrida, distancia a la meta y energía consumida por los robots que han llegado a la meta, se han chocado o ni han llegado a la meta ni se han chocado. A continuación, se calcula la nueva aptitud de los individuos a través de la función de coste, la cual viene dada por los siguientes factores: distancia del robot a la meta, distancia recorrida por el robot hasta llegar a la meta, si el robot ha alcanzado la meta y si el robot se ha chocado con un obstáculo. Finalmente, los individuos se reproducen siguiendo los cuatro operadores genéticos del algoritmo genético.

De esta manera, termina la generación actual dando paso a una nueva generación en la que, antes de iniciar su simulación, se limpian los indicadores de todos los robots, se reinicia el mapa y se insertan los nuevos genes de los individuos en los pesos de la red neuronal de los robots. El programa realiza las generaciones indicadas hasta llegar a la última donde, además, guardará los pesos del mejor robot con el objetivo de poder simularle cuando se desee y, de esta forma, estudiar su comportamiento con mayor detenimiento. Finalmente, con el objetivo de realizar un estudio lo más exacto posible, el programa es ejecutado diez veces y obtiene una media de todos los datos guardados.

V. Ajuste de Hiperparámetros

En un primer momento, los valores óptimos de todos los hiperparámetros del programa son desconocidos, por lo que se deben de realizar numerosas pruebas para encontrarlos. Este ajuste se va a realizar en el mapa sencillo y los valores iniciales han sido escogidos intuitivamente.

A. Ajuste de Algoritmos Genéticos

Los hiperparámetros que deben ajustarse en este campo son: el número de individuos de la población, el número de individuos elitistas de la población, la probabilidad de mutación de cada uno de los genes de un individuo, el número de generaciones, el número de puntos de cruce en la reproducción entre dos individuos y, los parámetros de la función de costo.

Para el número de individuos, se ha escogido 100, 1.000 y 10.000 individuos, obteniéndose un porcentaje de robots que alcanzan la meta del 5%, 14% y 17,5% respectivamente. Así, se observa que, a mayor número de individuos, mayor número de genes potenciales, y, por tanto, mejores resultados. Aun así, se ha decidido continuar con 1.000 individuos, ya que el tiempo de ejecución del programa es mucho menor y el índice de éxito ya ha alcanzado su máximo exponencial. En cuanto al número de individuos elitistas, se han hecho pruebas copiando los 1, 10 y 25 mejores individuos y comprobando que el porcentaje de robots que consiguen su objetivo es del 10%, 20% y 28% respectivamente. Por lo tanto, se toman 25 individuos elitistas y se concluye que, a mayor número de individuos elitistas, mayor número de robots llegan a la meta.

Para la probabilidad de mutación de un gen se han tomado unos porcentajes del 0,1%, 0,5%, 1% y 10% y se puede observar (Fig. 7) que los mejores resultados se obtienen para un 0,5%. Por tanto, se puede concluir que es mejor mantener intactos la mayoría de los genes (los cuales, de generación en generación van quedando los mejores) pero que aun así es importante introducir un pequeño porcentaje de mutación para favorecer la introducción de nuevos genes potenciales.

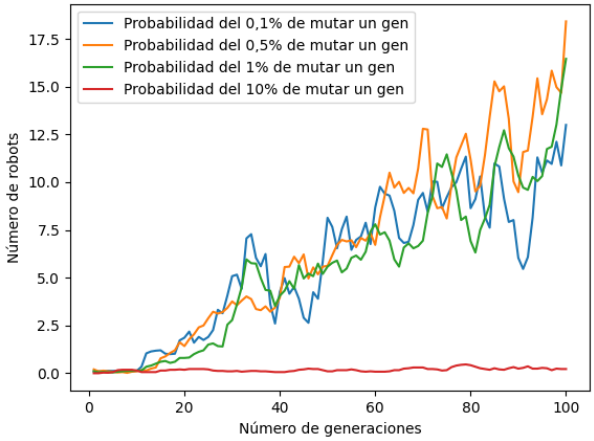


Fig. 7. Gráfica del número de robots que han alcanzado la meta en función de la probabilidad de mutación de un gen.

Para el número de generaciones se han realizado pruebas con 50, 100 y 200 generaciones, obteniéndose un resultado similar al del número de individuos. Cuantas más generaciones pasen, más tiempo para obtener los mejores genes y, por tanto, los robots serán más inteligentes y se obtendrán mejores resultados. Sin embargo, se ha decidido escoger 100 generaciones ya que, de esta forma, el tiempo de la simulación es mucho menor y, en torno a este número de generaciones, el ritmo de aprendizaje de los robots disminuye en gran medida.

Para el número de puntos de cruce se ha escogido 4, 10, 20 y 30. En este experimento, los resultados para los 4 casos son muy similares y, por tanto, se puede concluir que no es un parámetro muy crítico para el ajuste. Aun así, se ha escogido 20 puntos de cruce ya que, en este caso, los robots recorren una distancia algo menor al resto.

Finalmente, se deben ajustar los 4 parámetros de la función de costo (distancia recorrida por el robot hasta llegar a la meta, distancia a la meta del robot, costo adicional si el robot se ha chocado con un obstáculo y costo adicional si el robot no ha alcanzado la meta). Para la distancia recorrida, se han realizado tres pruebas dividiendo la distancia por una constante de 100, 500 y 1.000, obteniendo unos resultados ligeramente favorables para el factor /500. Para la distancia a la meta, se han hecho pruebas multiplicando la distancia por una constante de 1, 5 y 10, obteniendo resultados prácticamente similares. Por ello, en este caso, el desempate ha sido la energía consumida por los robots donde el factor 10 tiene una ligera ventaja con respecto al resto. Por su parte, para el costo adicional por chocarse, se han hecho pruebas con un costo de 100, 1.000 y 10.000, destacando los buenos resultados obtenidos con el factor de 1.000. Finalmente, para el costo adicional por no llegar a la meta, se han hecho simulaciones con 100, 500, 1.000 y 1.500, obteniéndose mejores resultados para el factor de 1.000. Así, la conclusión de este bloque es que los costes adicionales por no lograr el objetivo principal (no llegar a la meta y chocarse con un obstáculo), tienen más importancia que los costes por realizar la tarea lo mejor posible (distancia recorrida y distancia a la meta), debido a que estos costes tienen un valor inferior a

100 y los costos adicionales tienen un valor de 1.000.

B. Ajuste de Redes Neuronales

Los hiperparámetros que deben ajustarse en las redes neuronales son: el número de capas y el número de neuronas por capas, la función de activación de las neuronas, la presencia de sesgo en las capas y, las características de los datos de entrada (número de sensores del robot, normalización de los datos y posibilidad de introducir el estado anterior del robot).

Para el número de capas y el número de neuronas por capa, se han hecho pruebas con 4 neuronas y 8 neuronas en 1 capa oculta y con 4 y 2 neuronas y con 8 y 4 neuronas en 2 capas ocultas respectivamente. (Fig. 8).

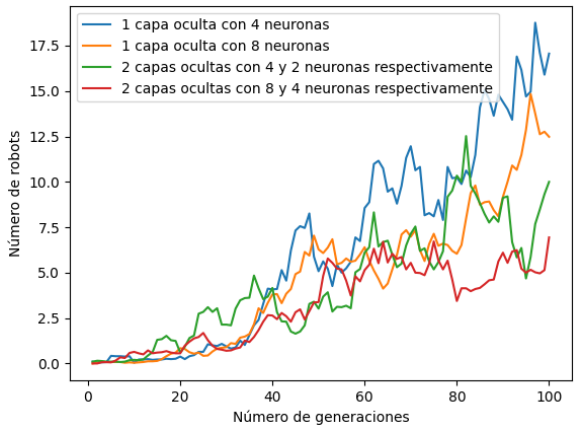


Fig. 8. Gráfica del número de robots que han alcanzado la meta en función del número de capas ocultas y del número de neuronas por capa.

La gráfica muestra cómo se obtienen mejores resultados con 1 capa oculta en lugar de 2 y con 4 neuronas en lugar de 8. Esto puede deberse a que, con muchas capas ocultas y muchas neuronas, la información se procesa en exceso y la eficiencia desciende. Para la activación de las neuronas, se han probado algunas de las funciones de activación más populares como son la función relu, softmax, sigmoide y tangente hiperbólica. Sólo esta última ha obtenido buenos resultados, ya que es la única función que tiene valores de salida positivos y negativos (necesarios para que el robot avance y retroceda). En cuanto al sesgo, se ha hecho una prueba rápida sobre si su presencia favorece o desfavorece el rendimiento. Como anticipaba el estado del arte del trabajo, el sesgo favorece el aprendizaje de la red neuronal, por lo que se ha introducido para el resto de las pruebas.

Finalmente, para los datos de entrada, se han hecho pruebas con dos posibles opciones para todos los casos: con 4 y 8 sensores, con y sin datos normalizados (es decir, todos los datos de entrada están ajustados a una misma escala) y, con y sin el estado anterior (es decir, el robot dispone de la información de la distancia captada por sus sensores, de su posición y de la posición de la meta en un instante anterior). Los resultados obtenidos muestran una clara ventaja para una de las opciones en todos los casos: 4 sensores, datos normalizados y sin el estado anterior. Esto puede deberse a que, si el robot dispone de más sensores o de su estado anterior, cuenta con demasiada información y puede saturarse. Por su parte, normalizar los datos siempre favorece su comprensión, ya que todos se encuentran dentro de un mismo rango pequeño.

C. Ajuste del Robot

El único hiperparámetro que se debe ajustar en el robot es la posición de los sensores: estos pueden estar fijos al robot (es decir, el sistema de referencia es el mapa y, por tanto, al ser este estático, los sensores apuntan siempre a un mismo ángulo) o rotando junto al robot (es decir, el sistema de referencia es el robot y, por tanto, los sensores giran junto a él). Los resultados obtenidos son mucho mejores para el primer caso que para el segundo (Fig. 9), lo cual puede deberse a que, con los sensores móviles, la información recopilada cambie demasiado rápido y el robot no tenga el tiempo suficiente para procesarla correctamente, provocando un mal rendimiento.

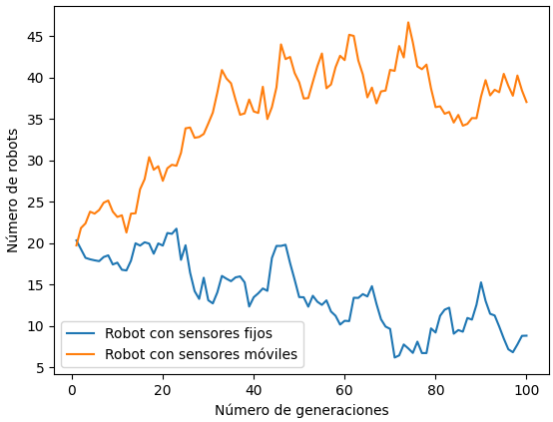


Fig. 9. Gráfica del número de robots que se han chocado con un obstáculo en función de la posición de los sensores del robot.

VI. Resultados

Una vez ajustados los hiperparámetros de la navegación a través de redes neuronales y algoritmos genéticos, se puede realizar su comparación con la navegación a través de campos potenciales en los distintos mapas construidos.

A. Pruebas

Las pruebas realizadas en los mapas sin obstáculos y sencillos (Fig. 10) muestran una clara superioridad de los campos potenciales sobre las redes neuronales.

La gran ventaja de los campos potenciales reside en la navegación en campo abierto (es decir, cuando hay pocos obstáculos estáticos), ya que es muy fácil para los robots encontrar la ubicación con energía nula al no haber apenas otros campos potenciales que interfieran en sus cálculos. Aun así, se puede observar cómo en el mapa sencillo, cuando sólo hay un obstáculo, también obtiene buenos resultados, pero éstos empiezan a empeorar al haber otro campo potencial que interrumpe en sus cálculos, provocando la aparición de los famosos mínimos locales (puntos con bajo potencial, pero no el punto con potencial nulo, que es la meta).

Por otro lado, se puede comprobar que, para las redes neuronales, se obtienen resultados similares en el mapa sin obstáculos y en el mapa sencillo, aunque este último sea más complejo. Esto se debe a que todos los hiperparámetros se ajustaron en el mapa sencillo y, con ello, se puede comprobar la importancia y la desventaja de los hiperparámetros en las

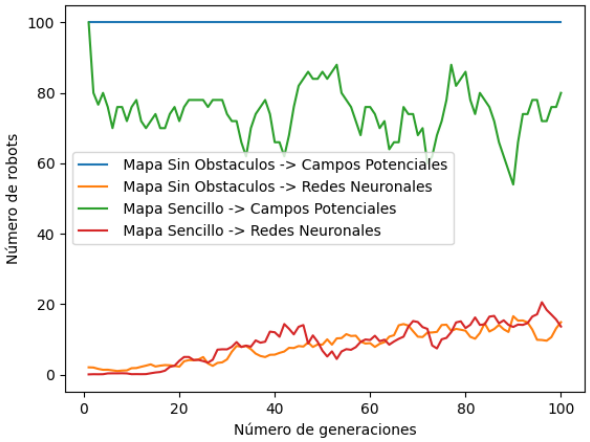


Fig. 10. Gráfica del número de robots que han alcanzado la meta en los mapas sin obstáculos y sencillo con los dos métodos de navegación.

redes neuronales, ya que, si no se ajustan en el caso de estudio, no se obtendrá su rendimiento óptimo.

Por otro lado, las pruebas realizadas en los mapas intermedio y difícil (Fig. 11) muestran un claro cambio de tendencia.

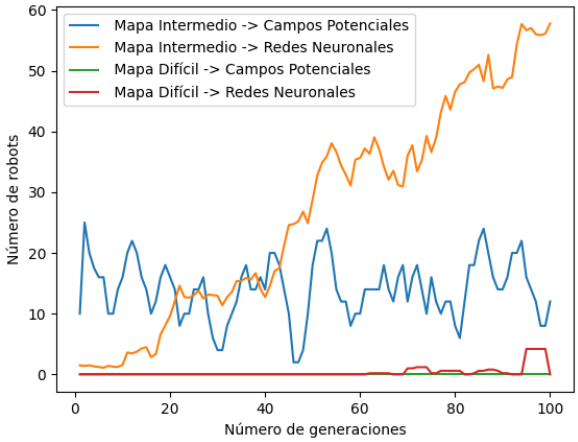


Fig. 11. Gráfica del número de robots que han alcanzado la meta en los mapas intermedio y difícil con los dos métodos de navegación.

Las redes neuronales han comenzado a superar a los campos potenciales. Con el aumento de la dificultad (es decir, más obstáculos), aparecen más mínimos locales en el mapa y, por tanto, más oportunidades para que los robots con campos potenciales caigan en estos puntos y no consigan llegar a la meta. Además, este experimento pone de manifiesto que muchos más robots alcanzan más la meta en el mapa intermedio que en el mapa sin obstáculos y sencillo. Este descubrimiento puede deberse a que la configuración inicial de los genes y su mutación posterior es un factor muy importante para obtener un método de aprendizaje rápido y exitoso.

B. Análisis Final

Los campos potenciales son más útiles en mapas con pocos obstáculos donde pueden calcular de manera precisa el punto de potencial nulo sin caer en mínimos locales. Además, es un método de navegación fácil de programar e implementar y, por tanto, se recomienda su uso en mapas sencillos que no tengan muchas complicaciones.

Por su parte, las redes neuronales junto con los algoritmos genéticos son más útiles en mapas con muchos obstáculos donde, de generación en generación, aprenden a utilizar la distancia captada por sus sensores y las posiciones de la meta y del propio robot. Por ello, este tipo de navegación es más robusto ya que, sin importar el mapa, se acostumbran y aprenden a navegar en él. Por otro lado, este método de navegación es complicado de programar e implementar y, por tanto, se recomienda su uso en mapas difíciles donde es necesaria la aparición de la inteligencia artificial para poder lograr el objetivo.

VII. CONCLUSIONES

En este trabajo se han desarrollado dos métodos de navegación: mediante campos potenciales clásicos y mediante redes neuronales que aprenden con algoritmos genéticos. Con ello, se ha realizado un estudio exhaustivo sobre el mejor ajuste de hiperparámetros para redes neuronales y algoritmos genéticos en el campo de la navegación robótica y se ha descubierto la importancia de tener una configuración inicial y una mutación posterior de los genes buena para obtener un método de aprendizaje rápido y exitoso. Además, se ha demostrado cómo un método basado en inteligencia artificial puede enseñar a un robot a llegar a cualquier punto desde cualquier posición inicial esquivando todo tipo de obstáculos (ya que con los campos potenciales los robots pueden quedar atrapados si existe un mínimo local entre ambas posiciones) y cómo es capaz de alcanzar la meta en mapas tan difíciles donde los métodos clásicos no tienen nada que hacer.

REFERENCIAS

[1] A. Rosales, G. Scaglia, V. Mut and F. di Sciascio, “Navigation in Mobile Robots in Unstructured Environments using Linear Algebra”, ISSN: 1697-7912, Vol. 6, No. 2, pp. 79-88, April 2009, doi: 10.1016/S1697-7912(09)70096-2.

[2] A.V. Rendón C., “Evaluation of autonomous navigation strategy based on reactive behavior for mobile robotic platforms”, Tekhnê Magazine, Vol.12, No. 2, pp. 75-82, December 2015.

[3] L.V. Calderita, V. Moreno and B. Curto, “Navigation of Robots in Training: a Reactive Approach with Constraints”, University of Salamanca, 2021.

[4] H.G. Acevedo and C.A.M. Castañeda, “Comparative study of three navigation techniques for mobile robots”, UIS Engineering, Vol. 6, No. 1, pp. 77-84, June 2007.

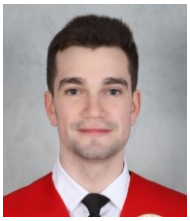
[5] Z. Yi and L. Yuan, “Application of fuzzy neural networks in data fusion for mobile robot wall-following”, 2008 7th World Congress on Intelligent Control and Automation, Chongqing, China, pp. 6580–6583, June 2008, doi: 10.1109/WCICA.2008.4593920.

[6] O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, N.A. Mohamed and H. Arshad, “State-of-the-art in artificial neural network applications: A survey”, Heliyon, November 2018, doi: 10.1016/j.heliyon.2018.e00938.

[7] O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, A.M. Umar, O.U. Linus, H. Arshad, A.A. Kazaure, U. Gana and M.U. Kiru, “Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition”, IEEE Access, Vol.7, November 2019, doi: 10.1109/ACCESS.2019.2945545.

[8] M.A. Boyacioglu, Y. Kara and O.K. Baykan, “A predicting bank financial failures using neural networks, support vector machines and multivariate statistical methods: a comparative analysis in the sample of savings deposit insurance fund (SDIF) transferred banks in Turkey”, Expert Systems with Applications, pp. 3355-3366, March 2009, doi: 10.1016/j.eswa.2008.01.003.

[9] C. Torras, "Applications of neural networks in robotics", Polytechnic University of Catalonia, pp. 479.498, 1995.
[10] C.L.C.D. de León, S.V. Limon, J.M. Gonzalez-Calleros and M.A.D.V. Treviño, "Artificial neural network for the extraction of dynamic parameters of robots from incomplete information about their movement", Colombian Computing Magazine, Vol. 22, No. 2, pp. 37-47, December 2021, doi: 10.29375/25392115.4298.
[11] S. Arun, G. Harish, K. Salomon, R. Saravanan, K. Kalpana and Dr. J. Jaya, "Neural networks and genetic algorithm based intelligent robot for face recognition and obstacle avoidance", International Conference on Current Trends in Engineering and Technology, pp. 356-361, July 2013, doi: 10.1109/ICCTET.2013.6675985.
[12] H. Nguyen, S.H. Fyhn, P. De Petris, and K. Alexis, "Motion Primitives-based Navigation Planning using Deep Collision Prediction", Norwegian University of Science and Technology, January 2022, doi: 10.1109/ICRA46639.2022.9812231.
[13] M. Anthony Lewis, Andrew H. Fagg and Alan Solidum, "Genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot", IEEE International Conference on Robotics and Automation, pp. 2618-23, 1992.
[14] X. Liu, D. Jiang, B. Tao, G. Jiang, Y. Sun, J. Kong, X. Tong, G. Zhao and B. Chen, "Genetic Algorithm-Based Trajectory Optimization for Digital Twin Robots", Frontiers in Bioengineering and Biotechnology, Vol. 9, January 2022, doi: 10.3389/fbioe.2021.793782.
[15] Mª C. M. Provecho, R.G. Martínez and R.A. Rodríguez, "Self-guiding of mobile robots using neural networks", XXV Automation Conference, September 2004.
[16] L.H. Rios G., M. Bueno L. and S. Sanchez A., "Generation of trajectories for a mobile robot using neural networks", Scientia Et Technica, Vol. 14, No. 39, pp. 94-99, September 2008.
[17] A. Hossian, L. Cejas, R. Carabajal, C. Echeverría, V. Olivera and M. Alvea, "Development of Experiments of a Navigator Robot with Neural Networks in a Structured Environment: Behavioral Programming with Backpropagation Algorithm", Latin American Journal of Software Engineering, pp. 41-46, 2015, doi: 10.18294/relais.2015.41-46.
[18] D.J. Montana and L.Davis, "Training Feedforward Neural Networks Using Genetic Algorithms", BBN Systems and Technologies Corp, pp. 762-767.
[19] T. Dewi, P. Risma, Y. Oktarina and M.T. Roseno, "Neural Network Controller Design for a Mobile Robot Navigation; a Case Study", Proc. EECSI 2017, September 2017, doi: 10.1109/EECSI.2017.8239168.
[20] S. Forrest, "Genetic Algorithms", ACM Computing Surveys, Vol. 28, No.1, March 1996, doi: 10.1145/234313.234350.
[21] B.M. Batista, J.A.M. Pérez and J.M.M. Vega, "Genetic Algorithms. A practical vision", Journal of Mathematics Didactics, Vol. 71, pp. 29-47, September 2009.
[22] J. Lee, B.-Y. Kang and D.-W. Kim, "Fast genetic algorithm for robot path planning, Electronics Letters", Vol. 49, No. 23, pp. 1449-1451, November 2013, doi: 10.1049/el.2013.3143.
[23] F.P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley and J. Clune, "Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning", Uber AI Labs, April 2018.
[24] M. Gestal, D. Rivero, J.R. Rabuñal, J. Dorado and A. Pazo, "Introduction to Genetic Algorithms and Genetic Programming", Digitalia, January 2010.
[25] H. Chiroma, A.S.M. Noor, S. Abdulkareem, A.I. Abubakar, A. Hermawan, H. Qin, M.F. Hamza and T. Herawan, "Neural Networks Optimization through Genetic Algorithm Searches: A Review", Applied Mathematics & Information Sciences, No. 6, pp. 1543-1564 November 2017, doi: 10.18576/amis/110602.
[26] K. Gurney, "An introduction to neural networks", UCL Press Limited: New Fetter, 1999.
[27] N.H. Singh and K. Thongam, "Neural network-based approaches for mobile robot navigation in static and moving obstacles environments", Intelligent Service Robotics, September 2018, doi: 10.1007/s11370-018-0260-2.
[28] C.A. Ruiz and M.S. Basualdo, "Neural Networks: Basic concepts and applications", National Technological University, March 2001.
[29] F.J.G. Quesada, M.A.F. Graciani, M.L. Bonal and M.A. Díaz-Marta, "Learning with artificial neural networks", Polytechnic University School of Albacete, pp. 169-179, 2020.
[30] E. Serna M., "Development and innovation in engineering", 2017.
[31] J. Cheng and Q.S. Li, "Reliability analysis of structures using artificial neural network based genetic algorithms", Comput. Methods Appl. Mech. Engrg., Vol. 197, pp. 3742-3750, August 2008, doi: 10.1016/j.cma.2008.02.026.
[32] A.J. Serrano, E. Soria and J.D. Martín, "Artificial Neural Networks", University of Valencia, OpenCourseWare, 2010.
[33] X.B. Olabe, "Artificial neural networks and their applications", Bilbao Higher School of Engineering, October 2021.
[34] L.V. Fausett, "Fundamentals of Neural Networks", Architectures, Algorithms and Applications, Prentice-Hall, 1994.
[35] S. Nolfi, D. Floreano, O. Miglino and F. Mondada, "How to Evolve Autonomous Robots Different Approaches in Evolutionary Robotics", Proceedings on the Artificial Life IV Conference, May 1994.
[36] A. J. Barreto-Cubero, A. Gómez-Espinosa, J.A.E. Cabello, E. Cuan-Urquizo and Sergio R. Cruz-Ramírez, "Sensor Data Fusion for a Mobile Robot Using Neural Networks", Sensors, December 2021, doi: 10.3390/s22010305.



David Abad Pérez, Madrid, Spain. B. Sc. in Electrical Engineering and Industrial Electronics and Automation Engineering from the Universidad Politécnica de Madrid (UPM) (Spain) in 2023. He is currently working as an engineer at Alstom.

Basil Mohammed Al-Hadithi got the title of B. Sc. in control and system engineering in 1983 and the M. Sc. in control and instrumentation engineering in 1988. He received a PhD in process control and artificial intelligence in 2002 from Universidad Politécnica de Madrid (UPM) (Spain) with a thesis on analysis, design, and stability of fuzzy slide-mode control systems. He is a full professor at UPM. His teaching activity covers control engineering and analogue electronics, being an author and co-author of seven textbooks and having supervised and co-supervised several B. Sc. final year projects, M. Sc. theses and PhD theses. He is a researcher at the Centre for Automation and Robotics UPM-CSIC. His interest is mainly focused on fuzzy control and slide mode control. He has several publications (JCR), book chapters and conference papers. Moreover, he has participated in several research projects and industrial contracts with companies. He is a board member and reviewer of several international scientific societies and International journals in modelling and designing control systems.

Victor Cadix Martín got the title of B. Sc. in Industrial Electronics and Automation Engineering in 2019 and the M. Sc. In Automation and Robotics in 2021 from the Universidad Politécnica de Madrid (UPM) (Spain). His interests are mainly focused on systems control, navigation, legged robots and swarm robotics. He is currently working at Perseo Techworks.