Adaptive Navigation System for an Autonomous Vehicle in a Goal-Oriented Environment

Over Alexander Mejia-Rosado, Ronald Mateo Ceballos-Lozano, Rhonald Jose Torres-Diaz, and Juan Pablo Hoyos-Sanchez

***Abstract*—In the context of autonomous navigation, the development of systems that enable vehicles to operate independently in controlled environments is a crucial step toward advancing autonomous technology. This work presents the design, implementation, and validation of a navigation system for autonomous vehicles using NeuroEvolution of Augmenting Topologies (NEAT). The primary objective was to create a vehicle capable of navigating a 2D map with a defined starting point and target. Virtual sensors enable the vehicle to identify navigable paths and boundaries. Distance metrics such as Euclidean, Manhattan, and Chebyshev were employed as reward systems, continuously calculating agent positions. The closer the vehicle is to the target, the higher its fitness score, forming the basis of the fitness function. A forced reinforcement acceleration method was designed and implemented to ensure progress when the vehicle's speed fell below 0.1, preventing it from becoming stalled. Validation tests were conducted to evaluate the system's performance under varying conditions. Results demonstrate that the autonomous vehicle can navigate the map effectively, improving its fitness score in each generation depending on the distance metric used. Chebyshev performed best in obstacle-free environments, while Euclidean excelled in the presence of obstacles. The forced reinforcement method significantly reduced the time required to achieve the target fitness. These findings provide valuable insights for researchers aiming to develop NEAT-based navigation systems for autonomous vehicles.**

Link to graphical and video abstracts, and to code: https://latamt.ieeer9.org/index.php/transactions/article/view/9559

***Index Terms*—Autonomous vehicle, distance metrics, fitness function, NEAT, reinforcement learning.**

I. Introducción

UNO de los mayores problemas del transporte moderno es el alto índice de accidentes en el tránsito terrestre, la gran cantidad es atribuida a factores como errores humanos, fallas mecánicas y fenómenos naturales. Según la Organización Mundial de la Salud (OMS) en 2023, entre 2010 y 2021, el número de vehículos a nivel mundial se duplicó, superando los mil millones [1], con 1,19 millones de muertes en todo el mundo y una tasa de mortalidad de 15 personas por cada 100,000 habitantes. Además, cerca del 80% de las vías de tránsito a nivel mundial no cumplen con los estándares básicos de seguridad para peatones y ciclistas, lo que incrementa el riesgo para ciclistas y peatones, especialmente en entornos urbanos con alta densidad de tráfico, generando congestión y emisiones contaminantes [2], [3].

La automatización y los sistemas inteligentes de tráfico surgen como alternativas efectivas para reducir accidentes y optimizar el tránsito. Estos sistemas mejoran la seguridad y eficiencia tanto del vehículo como del transporte en general, enfocándose no solo en el control preciso del automóvil, sino también en la capacidad del sistema para adaptarse dinámicamente a condiciones variables y mantener una trayectoria segura para los pasajeros y otros agentes viales [4]. Los vehículos autónomos, apoyados en tecnologías avanzadas como la Programación Dinámica Aproximada (ADP), mejoran la seguridad y eficiencia al adaptarse a condiciones variables, optimizando trayectorias y minimizando errores de seguimiento. Este enfoque integra técnicas avanzadas de inteligencia artificial para garantizar un control preciso y decisiones en tiempo real [5].

Por otro lado, algoritmos clásicos como Dijkstra y A* se complementan con aprendizaje por refuerzo profundo para planificación de rutas más eficientes [6], [7], [8], [9], mientras que la Neuro-Evolución de Topologías Aumentadas (NEAT) optimiza redes neuronales adaptativas, destacándose como una herramienta poderosa para abordar los retos asociados. NEAT ofrece una potente herramienta de aprendizaje evolutivo en la creación de redes neuronales artificiales (ANN), utilizando algoritmos genéticos (GA) para optimizar las redes neuronales, aumentando gradualmente su complejidad según la necesidad de la tarea. Por ejemplo, en el estudio presentado en [10], se analiza un modelo de simulación de evacuación basado en agentes, en el cual se estudia la dinámica de diferentes vehículos y su proceso de aprendizaje mediante NEAT. Este enfoque refuerza la analogía entre los GA y la evolución biológica, permitiendo que las redes neuronales evolucionen de manera adaptativa para afrontar entornos dinámicos y optimizar soluciones de forma simultánea.

El rendimiento de los algoritmos de Neuro-Evolución se compara favorablemente con el de los algoritmos de retro-propagación basados en gradientes. Una característica clave de NEAT es que el proceso de evolución comienza a partir de redes neuronales muy simples, cuya topología aumenta gradualmente en complejidad a lo largo de la evolución [11]. Esta característica reduce la complejidad innecesaria de la red neuronal final, algo que no es posible lograr utilizando algoritmos basados en gradientes.

Este trabajo [12], examinó cómo el algoritmo NEAT permite la optimización evolutiva de redes neuronales mediante

The associate editor coordinating the review of this manuscript and approving it for publication was Alejandro Dzul (*Corresponding author: Over Mejia*).

Over A. Mejia-Rosado, R. M. Ceballos-Lozano, R. J. Torres-Diaz, and J. P. Hoyos-Sanchez are with Universidad Nacional de Colombia, Sede De La Paz, Colombia (e-mails: omejiar@unal.edu.co, rceballosl@unal.edu.co, rhtorresd@unal.edu.co, and jhoyoss@unal.edu.co).

topologías dinámicas y mutaciones estructurales. La implementación de NEAT aborda problemas complejos como la clasificación y el control en tiempo real, empleando especiación y un registro histórico de innovaciones para mantener la diversidad y mejorar la precisión sin ajuste manual. También, implementando algoritmos evolutivos como Evolutionary Acquisition of Neural Topologies (EANT), los cuales pueden superar NEAT en temas como el rendimiento. En este trabajo se desarrolla un sistema de navegación para vehículos autónomos basado en NEAT. El vehículo, equipado con sensores virtuales, navega en un mapa 2D desde un punto inicial hasta un objetivo, utilizando métricas de distancia (Euclidiana, Manhattan y Chebyshev) como función de aptitud para evaluar y mejorar su desempeño. Para evitar estancamientos, se implementa un refuerzo forzado que asegura movimiento constante. Los resultados muestran que el vehículo mejora su capacidad de navegación autónoma con cada generación.

II. Metodología

Para la construcción del sistema de navegación adaptativa, se usó el entorno con la librería Pygame de Python. Se asigna un mapa y un agente principal. El entorno implementado es una fracción del mapa clásico del videojuego GTAII, la imagen de los agentes y parámetros principales de configuración se basan en los repositorios: NeuralNine y Mihir Gandhi. Se establecen los estados, un estado inicial donde se ubica el vehículo autónomo y un estado final. Se asignan píxeles verdes RGB (0, 255, 0) al mapa, que será el objetivo.

A. NEAT-Python

El agente debe aprender a moverse en el entorno, NEAT permite que el agente recorra el espacio asignándoles valores de salida dependiendo de los valores de entrada. Los valores de entrada son valores que se toman del entorno, para este caso en particular, las entradas estarían dadas por el color de cada píxel, dependiendo del color del píxel, el agente cambia o no de dirección, a esto se le conoce como acción o salida. NEAT le otorga al agente sensores que le permiten el reconocimiento del entorno, estos sensores son asignados para determinar cuándo este sale de las vías, y las distancias entre puntos [13]. En este artículo implementamos el uso de 5 sensores, asignados a una lista; -90°, -45°, 0°, 45° y 90°. Estas son las principales entradas para nuestro algoritmo NEAT aplicado al agente ver Fig. 1.

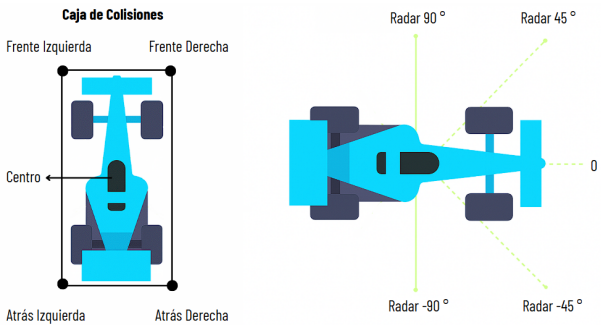


Fig. 1. Asignación de sensores al agente, caja de colisiones.

En NEAT, las generaciones determinan el momento en el que se finaliza la simulación. Por ejemplo, si se establecen 50 generaciones, este límite se aplica a los agentes creados por la red neuronal, utilizando distintos valores de configuración en cada ejecución. Si en la primera ejecución todos los vehículos chocan, se inicia una nueva ejecución; este proceso continúa hasta alcanzar la última, es decir, la generación 50. En el entorno, se crean varios agentes que recorren el mapa; si un agente choca o se sale de la vía, esta configuración tendrá una menor probabilidad de replicarse.

En la configuración inicial, el parámetro Capa oculta se establece en 0, ya que se comienza con una configuración sencilla. Con cada generación, el valor aumenta, lo que permite que la salida obtenida a partir de la entrada sea la más óptima.

En la Fig. 2 se observa que las capas ocultas son nodos encargados de procesar la información de entrada. Con cada generación, el número de estas capas aumenta en función de una ganancia, determinada por las acciones previas del agente. Si el agente se desplaza por el mapa sin exceder las limitaciones establecidas, permanece en la misma generación.

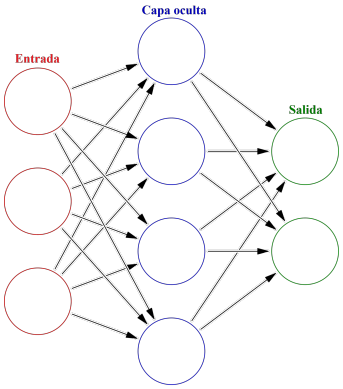


Fig. 2. Estructura general de una red neuronal multicapa [14].

B. Función Aptitud

Para evaluar qué tan efectivos son los genomas en las próximas generaciones, NEAT implementa una función que calcula qué tan bien se resuelve el problema en cuestión, denominada *Fitness function*, en español *Función aptitud*. Si un genoma A resuelve el problema de manera más exitosa que un genoma B, entonces el valor de aptitud de A debe ser mayor que el de B. La magnitud absoluta y el signo de estos valores no son relevantes; únicamente importan sus valores relativos.

La ecuación de actitud sería:

$$Aptitud = 1 - \sum_{i}(e_i - a_i)^2,$$

donde, e_i (salidas esperadas): son los valores que se espera obtener de la red neuronal o del agente autónomo. Estas salidas representan las posiciones ideales o los movimientos óptimos que el agente debería realizar para alcanzar el objetivo de la manera más eficiente posible. Por otro lado a_i (salidas reales): son los valores que se obtienen de la red neuronal o del agente autónomo, reflejando la respuesta del sistema ante

una determinada entrada. Los pasos para calcular la aptitud de cada genoma son los siguientes:

- **Crear una red neuronal**: Se genera una red neuronal basada en el genoma actual.
- **Simular el agente**: La red neuronal controla al agente autónomo en el entorno simulado (ver Fig. 3), recibiendo entradas del entorno (como las distancias detectadas por los sensores del agente) y produciendo acciones, cambios en la dirección y la velocidad.
- **Recopilar métricas**: Durante la simulación se registran métricas relevantes, como la distancia mínima al objetivo, el tiempo de supervivencia y si el agente ha colisionado.
- **Calcular la aptitud**: Se utiliza una función de aptitud definida para calcular el valor de aptitud del genoma, basándose en las métricas recopiladas.
- **Recompensa por distancia al objetivo** (R_d): La recompensa por distancia es inversamente proporcional a la distancia d_{agent} que separa al agente del objetivo. Se define de la siguiente manera:

$$R_d = \begin{cases} 10000, & \text{si } d_{agent} = 0, \\ 10000 - d_{agent}, & \text{si } d_{agent} > 0. \end{cases}$$

  El valor de d_{agent} se calcula utilizando distintas métricas de distancia:
- **Distancia Euclidiana:**

$$d_{agent} = \sqrt{(x_{agent} - x_{goal})^2 + (y_{agent} - y_{goal})^2}$$

- **Distancia de Manhattan:**

$$d_{agent} = |x_{agent} - x_{goal}| + |y_{agent} - y_{goal}|$$

- **Distancia de Chebyshev:**

$$d_{agent} = \max\left(|x_{agent} - x_{goal}|,\ |y_{agent} - y_{goal}|\right)$$

donde $(x_{\text{agent}}, y_{\text{agent}})$ representan las coordenadas actuales del agente, y $(x_{\text{goal}}, y_{goal})$ las coordenadas del objetivo. Esta recompensa incentiva al agente a acercarse lo máximo posible al objetivo, ya que al disminuir la distancia, el valor de la recompensa aumenta.

Combinando los términos anteriores, la función de aptitud se expresa como:

$$Aptitud = 10000 - d_{agent}$$

Además, para asegurar que el valor de aptitud no sea negativo, se aplica:

$$Aptitud = \max(0, Aptitud)$$

C. Implementación de la asignación entrada-salida

A partir de las entradas de los sensores se asigna un ángulo de rotación y una velocidad al agente autónomo. Se trata de un esquema básico, pero funcional. Sin embargo, ¿qué sucede si las variaciones de los ángulos son muy elevadas, o, por el contrario, demasiado bajas, o si la velocidad es insuficiente? Si el agente recibe de manera constante ángulos orientados en la misma dirección, la acumulación de dichos ángulos lo hace girar sin parar. Esto puede ocasionar el estancamiento del agente, especialmente en las primeras generaciones. Por otro lado, si la entrada es muy variable, pero con cambios iterativos en la dirección de los ángulos, la velocidad del agente se reduce; de lo contrario, el agente sería eliminado ante la alta probabilidad de chocar contra los bordes.

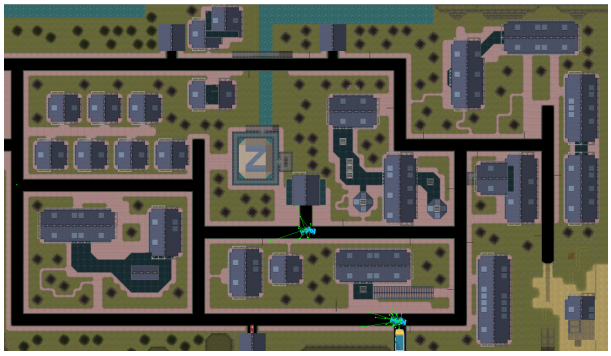


Fig. 3. Mapa del entorno en el que el agente autónomo realiza su aprendizaje.

Para evitar algunas de las singularidades (por ejemplo, estancamiento o rotación sin parar), se asigna un **Set_Valor** a la velocidad si la velocidad previa del agente es menor a 0.1. Esto previene que el agente avance de manera excesivamente lenta, optimizando el tiempo de cada simulación. En este artículo, **Set_Valor** se establece en 5. A esta asignación se la denomina *Refuerzo forzado* o *Método de aceleración*, y su diagrama se presenta en la Fig. 4.

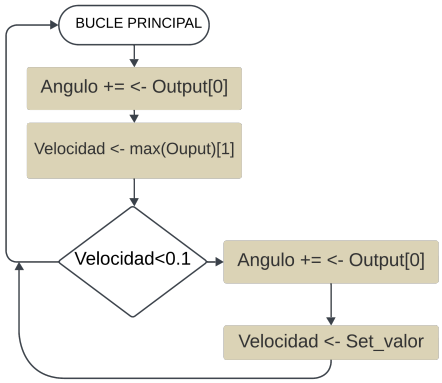


Fig. 4. Diagrama de flujo de la función de aceleración o refuerzo forzado, diseñada para optimizar y reducir el estancamiento de algunos agentes en primeras generaciones.

Una vez establecidos los parámetros iniciales, se realizaron distintas simulaciones para recopilar datos y analizar el comportamiento de los agentes. Se llevaron a cabo un total de 63 simulaciones para el primer mapa (Fig. 3), distribuidas en grupos de tres correspondientes a las distancias Manhattan, Euclidiana y Chebyshev. A cada número de generación le correspondieron 5 simulaciones; es decir, en la distancia Euclidiana se realizaron un total de $3 \times 5 = 15$ simulaciones: 5 para la generación de 20, 5 para la generación de 30 y, por último, 5 para la generación de 50. Se aplicó el mismo esquema para las distancias Manhattan y Chebyshev. Estas simulaciones se realizaron utilizando el método propuesto *Refuerzo forzado*. Además, se llevaron a cabo tres simulaciones

adicionales sin este método, correspondientes a una simulación de 50 generaciones por cada una de las métricas aplicadas. Por ende, el total de simulaciones fue de 45, más 3 simulaciones sin la aplicación de este método, como se observa en la Tabla I. Adicionalmente, se realizaron 15 simulaciones para un

TABLA I
DISTRIBUCIÓN DE LOS ENTRENAMIENTOS

Simulaciones	Generaciones	Refuerzo forzado
Euclidiana (5)	20, 30, 50	SI
Manhattan (5)	20, 30, 50	SI
Chebyshev (5)	20, 30, 50	SI
Euclidiana (1)	50	NO
Manhattan (1)	50	NO
Chebyshev (1)	50	NO

segundo mapa (ver Fig. S1-MAP del material suplementario). En estas simulaciones se trabajó únicamente con la generación 50, como se indica en la Tabla II: se realizaron 5 simulaciones para cada una de las distancias Euclidiana, Manhattan y Chebyshev, lo que suma un total de $5 \times 3 = 15$ simulaciones.

TABLA II
DISTRIBUCIÓN DE LOS ENTRENAMIENTOS MAPA 2

Simulaciones	Generaciones	Refuerzo forzado
Euclidiana (5)	50	SI
Manhattan (5)	50	SI
Chebyshev (5)	50	SI

D. Navegación Adaptativa

Uno de los objetivos principales del proyecto es evaluar la capacidad de adaptación de los agentes autónomos a nuevos entornos sin modificar significativamente el algoritmo, optimizando su aprendizaje mediante el método propuesto de *Refuerzo forzado*. Para esto se realiza una comparativa con el artículo [15], en el que se desarrolló un sistema de navegación para rovers basados en NEAT, aplicados a la extinción de incendios, lo que permite a los rovers evitar obstáculos y localizar personas en situaciones de alto riesgo.

E. Configuraciones Principales del Algoritmo NEAT

El archivo de configuración para NEAT consta de distintas secciones, cada una de ellas diseñada a como se espera sea la creación de generaciones en cada simulación. En la Tabla III [NEAT], se pueden encontrar los parámetros de aptitud (fitness) con el tipo de criterio, en este caso el máximo y el tamaño de la población (pop_size) en cada generación, el parámetro reset_on_extinction indica que una vez eliminados todos los agentes en una generación, automáticamente se creen nuevos; en la Tabla [DefaultReproduction] S2T del material suplementario, se encuentra la configuración de elitism y survival threshold, que permiten determinar el número de especies más aptas de cada generación que se desea conservar, y la proporción de cada especie permitida para reproducirse en cada generación. En la Tabla IV [DefaultGenome], se definen los parámetros específicos para la estructura de los genomas (redes neuronales) que NEAT evolucionará, así como el modo de activación activation default (tanh) y otras opciones como lo son relu, sigmoid y mutate rate (la información completa se encuentra en la Tabla S3T del material suplementario).

TABLA III
NEAT

Variable	Valor
fitness_criterion	max
fitness_threshold	10000
pop_size	50
reset_on_extinction	True

TABLA IV
DEFAULTGENOME

Opciones de Activación de Nodos	Valor
activation_default	tanh
activation_mutate_rate	0.01
activation_options	tanh, relu, sigmoid

III. RESULTADOS

Los registros obtenidos de las simulaciones incluyen la aptitud promedio de los agentes, la mejor aptitud alcanzada durante la simulación y los valores de las recompensas máximas y mínimas. Los códigos desarrollados y el material suplementario se encuentran disponibles en https://github.com/OverAlexander-MR/Navegacion_Vehiculo_Autonomo. A partir de los datos recopilados, se generaron gráficas de la aptitud promedio junto con sus desviaciones, que se presentan a continuación.

A. Entrenamiento sin el Refuerzo Forzado

Para esta configuración se realizó un entrenamiento de 50 generaciones para cada métrica de distancia, con el fin de analizar qué tan efectivas son las simulaciones sin la aplicación del método descrito en la Fig. 4.

De la Tabla V se observa que, al utilizar la distancia Euclidiana, la mayor aptitud obtenida fue de 8536 (véase la Fig. S1-EN del material suplementario). Por otra parte, la desviación obtenida refleja el comportamiento de los agentes en cada generación, como se aprecia en la Fig. 5. Cabe destacar que el mayor incremento de aptitud se produjo únicamente en la generación 10; a partir de la generación 12, el aprendizaje de los agentes no mostró mejoras significativas.

Al utilizar la distancia Manhattan, el puntaje obtenido difiere en solo 1 punto respecto a la distancia Euclidiana, alcanzando un valor de 8535 (ver Tabla V). La gráfica de desviación mostró un comportamiento similar al de la distancia Euclidiana; sin embargo, en la generación 10, la desviación superó los 8625 puntos (véanse las Figuras S1-MN y S2-MN).

Con la distancia Euclidiana, los agentes no alcanzaron el objetivo, finalizando las 50 generaciones en un tiempo de 10 minutos, durante el cual se eliminaron manualmente aquellos agentes que quedaron estancados. El tiempo de entrenamiento para la distancia Euclidiana sin intervención

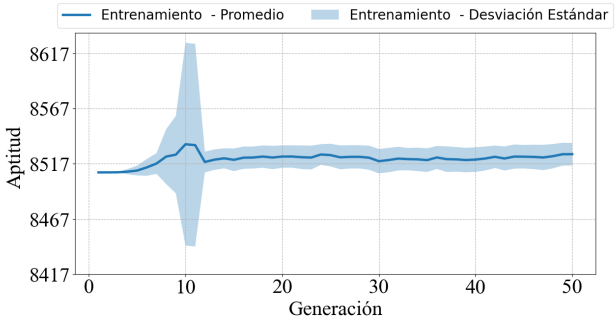


Fig. 5. Desviación y promedio de la aptitud para la distancia Euclidiana sin la aplicación del Refuerzo forzado.

fue de 7 horas y 56 minutos. En cambio, para la distancia Manhattan (sin la eliminación de los agentes estancados) el entrenamiento concluyó en 18 minutos. Finalmente, la distancia de Chebyshev presentó los mejores resultados, alcanzando una aptitud máxima de 9885 (ver Tabla V y Fig. S1-CN), y con una desviación que ronda los 8800 puntos. La principal diferencia respecto a las otras métricas de distancia, radica en el aprendizaje constante de los agentes a lo largo de cada generación, como se observa en la Figura 6. Usando la distancia Chebyshev, las simulaciones finalizaron en 10 minutos y 44 segundos.

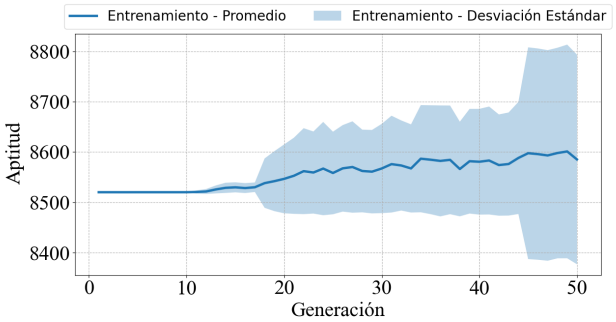


Fig. 6. Desviación y promedio de la aptitud para la distancia Chebyshev sin la aplicación del Refuerzo forzado.

TABLA V
APTITUDES MÁXIMAS Y PROMEDIOS SIN EL REFUERZO FORZADO

	N° Generaciones	Mejor Aptitud	Promedio
Euclidiana	50	8536	8525
Manhattan	50	8535	8423
Chebyshev	50	**9885**	**8585**

B. Resultados de las simulaciones con la aplicación del Refuerzo Forzado

La aplicación del método de Refuerzo Forzado mejora notablemente el desempeño del sistema de navegación. En líneas generales, se observa que:

- Los valores máximos de aptitud alcanzan cifras muy similares en todas las métricas, cercanas a 9990.
- El desempeño promedio varía, siendo la distancia Euclidiana la que presenta mejores resultados, superando en promedio a Manhattan por 340 puntos y a Chebyshev por 95 puntos.
- Además, la aplicación del Refuerzo Forzado reduce significativamente los tiempos de entrenamiento.

La Tabla VI resume las aptitudes máximas, promedios y mínimas obtenidos por cada métrica. A continuación se presentan en detalle los resultados para cada métrica de distancia.

1) Distancia Euclidiana: Para la distancia Euclidiana, la mayor aptitud máxima registrada fue de 9990, obtenido en el entrenamiento con 50 generaciones (ver Fig. S3A-E y S4A-E del material suplementario). En la Fig. 7 se observa que el entrenamiento número 4 alcanzó la mayor desviación en la última generación (**A1**). El segundo mejor resultado se obtuvo en el entrenamiento 3 con 30 generaciones, que presentó una aptitud máxima de 9986 (**A2** en Fig. 7; ver Fig. S1B-E y S2B-E). Finalmente, el entrenamiento 2 con 20 generaciones alcanzó una aptitud máxima de 9981 (**A3** en Fig. 7; ver Fig. S1C-E y S2C-E). La diferencia en la aptitud máxima entre 50 y 30 generaciones es de 4 puntos, mientras que entre 50 y 20 generaciones es de 109 puntos.

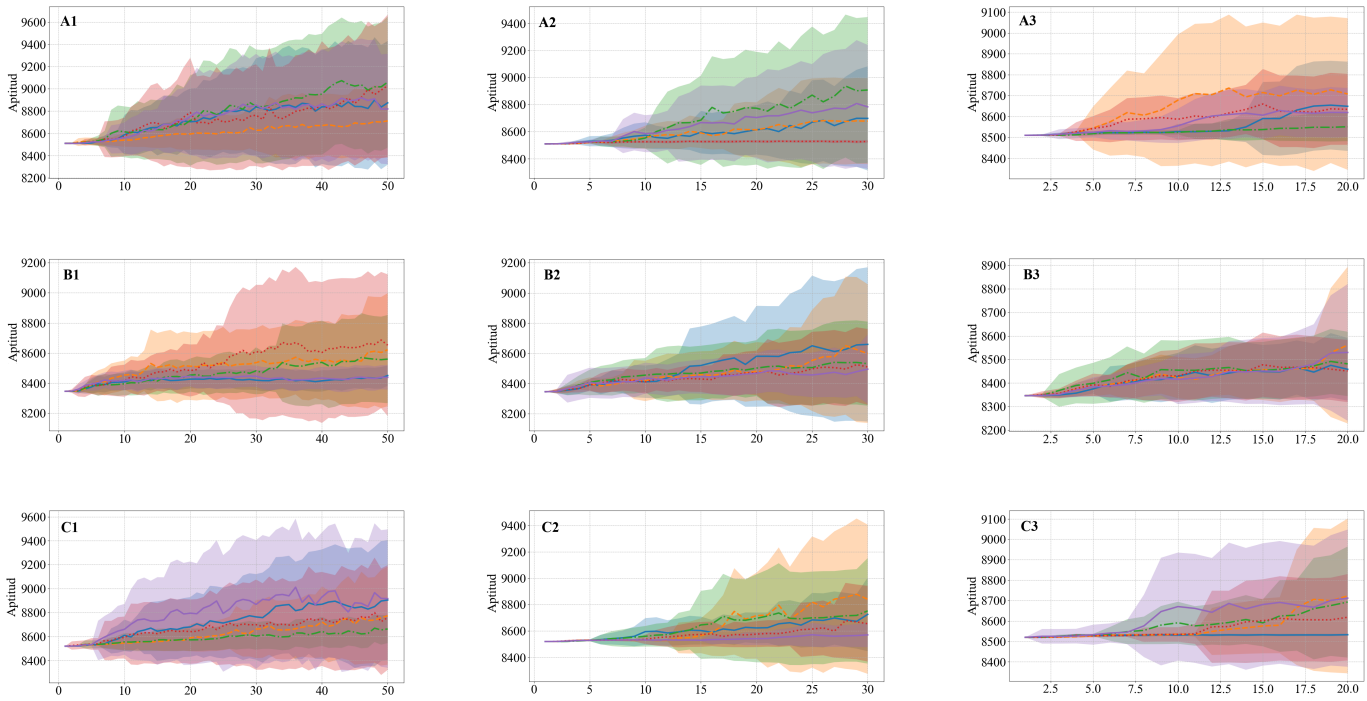
La Tabla S4T del material suplementario detalla estos resultados. El tiempo promedio de entrenamiento fue de 28 minutos para 50 generaciones, 17 minutos para 30 generaciones y 4.3 minutos para 20 generaciones.

2) Distancia Manhattan: En el caso de la distancia Manhattan, la mayor aptitud máxima se alcanzó en el entrenamiento número 1 con 30 generaciones, registrando un valor de 9990 (véanse Fig. S1B-M y S2B-M). Según la Fig. 7 (**B2**), en esta configuración la desviación se aproxima a los 9200 puntos. El segundo mejor resultado se obtuvo en el entrenamiento 4 con 50 generaciones, con una aptitud máxima de 9980 (**B1** en Fig. 7; ver Fig. S3C-M y S4C-M), y el tercer mejor en el entrenamiento 2 con 30 generaciones, alcanzando una aptitud máxima de 9978 (**B3** en Fig. 7; ver Fig. S3A-M y S4A-M). La diferencia entre la aptitud máxima obtenida para 30 y 50 generaciones es de 10 puntos, y entre 30 y 20 generaciones es de 12 puntos.

La Tabla S5T detalla estos resultados. El tiempo promedio de entrenamiento fue de 24 minutos para 50 generaciones, 8 minutos para 30 generaciones y 3 minutos para 20 generaciones.

3) Distancia Chebyshev: Para la distancia Chebyshev, la mayor aptitud máxima obtenida fue de 9994, alcanzado en el entrenamiento número 2 con 30 generaciones (**C2** en Fig. 7; ver Fig. S1B-C y S2B-C). Posteriormente, el mismo entrenamiento con 20 generaciones logró una aptitud máxima de 9990 (**C3** en Fig. 7; ver Fig. S3C-C y S4C-C), y finalmente el entrenamiento 1 con 50 generaciones presentó una aptitud máxima de 9988 (**C1** en Fig. 7; ver Fig. S1A-C y S2A-C). La diferencia entre la aptitud máxima de 30 y 20 generaciones es de 4 puntos, y entre 30 y 50 generaciones es de 6 puntos.

La Tabla S6T resume los resultados para esta métrica. El tiempo promedio de entrenamiento fue de 36.81 minutos para 50 generaciones, 7 minutos para 30 generaciones y 3 minutos para 20 generaciones.



(a) 50 Generaciones (b) 30 Generaciones (c) 20 Generaciones

Fig. 7. Promedios y desviaciones de cada entrenamiento con refuerzo para las métricas de distancia en un entorno sin obstaculos. **A1** hasta **A3** representan la distancia Euclidiana; **B1** hasta **B3** Manhattan; **C1** hasta **C3** Chebyshev. Las líneas continuas y punteadas representan los promedios, la sombra representa la desviación. Entrenamiento 1 , ; entrenamiento 2 , ; entrenamiento 3 , ; entrenamiento 4 , ; entrenamiento 5 , .

TABLA VI
APTITUDES MÁXIMAS, PROMEDIOS Y MÍNIMAS CON REFUERZO

	Gen.	Mejor Aptitud	Promedio	Menor Aptitud
Euclidiana	50	**9990**	**8898**	**9626**
	30	9986	8718	8544
	20	9981	8631	8773
Manhattan	50	9980	8544	**9621**
	30	**9990**	**8558**	9618
	20	9978	8489	8766
Chebyshev	50	9988	**8803**	8777
	30	**9994**	8708	**8787**
	20	9990	8655	8543

C. Resultados de las Simulaciones en Mapa con Obstáculo Utilizando el Refuerzo Forzado

La distancia Manhattan presentó la aptitud máxima con un valor de 9890, mientras que la distancia Chebyshev obtuvo el mejor promedio con 8694, una diferencia de 160 respecto a la distancia Manhattan. La Tabla VII resume las aptitudes máximas, promedios y mínimas para cada métrica. A continuación se detallan los resultados obtenidos para cada sistema de recompensas.

Se realizaron 5 simulaciones de 50 generaciones para cada sistema de recompensas (Euclidiana, Manhattan y Chebyshev) en el segundo mapa con obstáculo (ver Fig. S1-MAP del material suplementario).

La gráfica general obtenida tras cada entrenamiento para las distintas distancias se muestra en la Fig. S1-EM2.

TABLA VII
APTITUDES MÁXIMAS, PROMEDIOS Y MÍNIMAS EN EL SEGUNDO MAPA

	Gen.	Mejor Aptitud	Promedio	Menor Aptitud
Euclidiana	50	9684	8686	8933
Manhattan	50	**9890**	8534	8532
Chebyshev	50	9683	**8694**	**8985**

1) Aptitud para la Distancia Euclidiana, Manhattan y Chebyshev: Para las simulaciones realizadas con la distancia Euclidiana, la mejor aptitud obtenida es de 9684, por el tercer y quinto entrenamiento. La menor aptitud fue obtenida por el cuarto entrenamiento con 8933 puntos (Fig. S6-EM2, S7-EM2, S8-EM2 y S9-EM2).

Para la distancia Manhattan, la aptitud más alta se obtuvo en el entrenamiento número 2, con un valor de 9890 y una desviación que supera los 8900 puntos. El primer entrenamiento obtuvo la aptitud más baja, con 8532 puntos. Las desviaciones del primer entrenamiento no superan los 8550 puntos (Fig. S3-MM2, S4-MM2, S1-MM2 y S2-MM2). El entrenamiento que mayor aptitud obtuvo en la distancia Chebyshev, corresponde al cuarto entrenamiento con un valor de 9683 puntos, con una desviación que se aproxima a los 9100 puntos. La aptitud más baja obtenida le corresponde al entrenamiento número 5 con 8985 puntos y un decrecimiento en la desviación a partir de la generación 30 (Fig. S7-CM2, S8-CM2, S9-CM2 y S10-CM2).

D. Evaluación del Sistema de Navegación Adaptativa

Para evaluar la adaptabilidad de nuestro algoritmo, se realizaron dos simulaciones con 32 generaciones y la distancia Euclidiana en el entorno mostrado en la Fig. 1 que corresponde al primer mapa del artículo [15]. Se estableció un valor de 1000 en la función R_d para ajustarse a la resolución del mapa. En estas simulaciones, los agentes inician aproximadamente en el mismo punto de partida de los rovers, como se observa en el mapa de calor de la Fig. 7 (*Heatmap of the Rover after 500 generation simulation under map1*) [15].

En el artículo [15] los agentes empiezan a tener un aprendizaje significativo a partir de 100 generaciones, como se observa en Fig. 5 [15]; mientras que con la aplicación del *Refuerzo forzado* se observa a partir de la generación 5 (ver Fig. 8). Sin la aplicación del método propuesto, los agentes son capaces de tener un buen aprendizaje a partir de la generación 21 (ver Fig. 8).

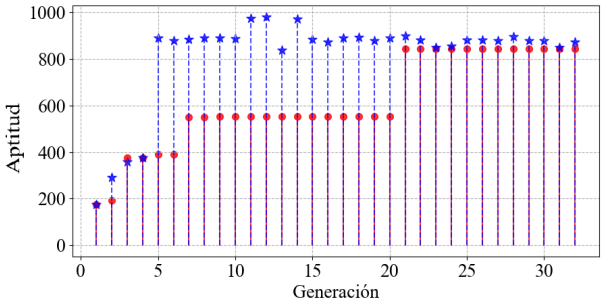


Fig. 8. Comparativa de las aptitudes máximas por generación, con la aplicación del *Refuerzo forzado* ⋆ y sin la aplicación del método ●.

IV. Conclusiones

El sistema de navegación adaptativa desarrollado demostró que la implementación del método de *Refuerzo forzado* propuesto mejoró el desempeño de vehículos autónomos en los entornos simulados, superando tanto en los valores máximos como en el promedio de desempeño obtenido sin el refuerzo. Además, redujo los tiempos de entrenamiento de 7 horas a menos de 35 minutos, mientras mantenía un alto desempeño en la navegación autónoma.

Para entornos sin obstáculos, la distancia Euclidiana mostró el mejor desempeño promedio (8898 puntos), seguida por Chebyshev (8803) y Manhattan (8558). Para entornos con obstáculos, Manhattan alcanzó la aptitud máxima (9890), mientras que Chebyshev demostró mayor consistencia con el mejor promedio. Por tanto, la implementación del refuerzo forzado no solo optimizó los tiempos de entrenamiento, sino que también mejoró significativamente los valores de aptitudes máximas y promedios en comparación con el sistema base, demostrando su efectividad para prevenir el estancamiento de los agentes.

Estos resultados validan la robustez y capacidad del sistema para adaptarse dinámicamente a condiciones variables, sugiriendo su potencial aplicabilidad en diversos escenarios de navegación autónoma.

References

[1] Organización Mundial de la Salud, *Informe sobre la situación mundial de la seguridad vial 2023*. Ginebra, Suiza: Organización Mundial de la Salud, 2023.

[2] R. Montezuma and J. Erazo, "El derecho a la vida en la movilidad urbana y el espacio público en américa latina," *Inter/secciones urbanas: origen y …*, 2008. [Online]. Available: https://www.flacsoandes.edu.ec/sites/default/files/agora/files/1218665734.ponencia_final_de_ricardo_montezuma_2.pdf

[3] D. Díaz, "Establecimiento de una metodología para el diseño de infraestructuras seguras para ciclistas en entornos urbanos," Ph.D. dissertation, Universidad de Málaga, 2015. [Online]. Available: https://core.ac.uk/download/pdf/132743165.pdf

[4] S. Arshad, M. Sualeh, D. Kim, D. V. Nam, and G.-W. Kim, "Clothoid: An integrated hierarchical framework for autonomous driving in a dynamic urban environment," *Sensors*, vol. 20, no. 18, p. 5053, 2020.

[5] Z. Lin, J. Ma, J. Duan, S. E. Li, H. Ma, B. Cheng, and T. H. Lee, "Policy iteration based approximate dynamic programming toward autonomous driving in constrained dynamic environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 5, pp. 5003–5013, 2023.

[6] I. Lamouik, A. Yahyaouy, and M. A. Sabri, "Smart multi-agent traffic coordinator for autonomous vehicles at intersections," in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2017, pp. 1–6.

[7] R. Inamdar, S. K. Sundarr, D. Khandelwal, V. D. Sahu, and N. Katal, "A comprehensive review on safe reinforcement learning for autonomous vehicle control in dynamic environments," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 10, p. 100810, 2024.

[8] Z. Huang, "Reinforcement learning based adaptive control method for traffic lights in intelligent transportation," *Alexandria Engineering Journal*, vol. 106, pp. 381–391, 2024.

[9] L. Luo, N. Zhao, Y. Zhu, and Y. Sun, "A* guiding dqn algorithm for automated guided vehicle pathfinding problem of robotic mobile fulfillment systems," *Computers & Industrial Engineering*, vol. 178, p. 109112, 2023.

[10] M. E. Yuksel, "Agent-based evacuation modeling with multiple exits using neuroevolution of augmenting topologies," *Advanced Engineering Informatics*, vol. 35, pp. 30–55, 2018.

[11] T. J. Ikonen and I. Harjunkoski, "Decision-making of online rescheduling procedures using neuroevolution of augmenting topologies," in *29th European Symposium on Computer Aided Process Engineering*, ser. Computer Aided Chemical Engineering, A. A. Kiss, E. Zondervan, R. Lakerveld, and L. Özkan, Eds. Elsevier, 2019, vol. 46, pp. 1177–1182.

[12] F. C. Guardo, "Evolución de redes neuronales mediante topologías aumentadas," Madrid, España, 6 2019.

[13] R. Lauckner and H. Kolivand, "Neat algorithm in autonomous vehicles," *Liverpool John Moores University*, 2023.

[14] J. Portilla, "A beginner's guide to neural networks in python," *Springboard Blog*, 2017. [Online]. Available: https://www.springboard.com/blog/data-science/beginners-guide-neural-network-in-python-scikit-learn-0-18/

[15] D. Shrestha and D. Valles, "Evolving autonomous navigation: A neat approach for firefighting rover operations in dynamic environments," in *2024 IEEE International Conference on Electro Information Technology (eIT)*, 2024, pp. 247–255.

Over Alexander Mejia-Rosado currently is a student of Mechatronic Engineering at the Universidad Nacional de Colombia, De La Paz campus. Engaged in research groups within the university campus, passionate about Computational Neural Networks, Artificial Intelligence, and the digital world.

Ronald Mateo Ceballos-Lozano currently is a Mechatronics Engineering student at the National University of Colombia, De La Paz campus. He actively participates in research groups focused on Artificial Intelligence, Microcontrollers, and Control Systems.

Rhonald Jose Torres-Diaz currently is a Mechatronics engineering student at Universidad Nacional de Colombia. Currently a student and researcher at the De La Paz campus. His research in data processing and AI in search of a better tomorrow.

Juan Pablo Hoyos-Sanchez received the Engineer, Magister in Electronics and Telecommunications and Doctor in Electronic Sciences degrees from Universidad del Cauca, Popayan, Colombia, in 2010, 2016, and 2018 respectively. He is currently an assistant professor at the Universidad Nacional de Colombia, De La Paz campus. His research interests are in signal processing, artificial intelligence, and convex optimization.