

Sistema de navegación adaptativa para un vehículo autónomo en un entorno con establecimiento objetivo

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

Resumen—Este proyecto se centra en el desarrollo de un sistema de navegación para vehículos autónomos utilizando NEAT (NeuroEvolution of Augmenting Topologies). El objetivo principal es diseñar e implementar un vehículo capaz de desplazarse en un mapa en 2D, el cual consta de un punto inicial y un objetivo para el vehículo autónomo. A través de sensores virtuales, el vehículo es capaz de identificar la vía por la cual puede avanzar y cuales los los límites de esta. Se utilizan métricas de distancia como la Euclidiana, Manhattan y Chebyshev como sistema de recompensas, que constantemente calculan las posiciones de los agentes, entre más cercano esté el vehículo del objetivo, su fitness incrementa, de esa manera se establece la función aptitud. Se crea y se implementa un sistema de refuerzo forzado, que permite al vehículo avanzar en caso de que su velocidad sea menor a 0,1, evitando así que el vehículo se estanke. Los resultados obtenidos muestran que el vehículo autónomo es capaz de desplazarse de manera autónoma en el mapa, aumentando su aptitud en cada generación dependiendo de la métrica de distancia utilizada.

I. INTRODUCCIÓN

Uno de los mayores problemas del transporte moderno es el alto índice de accidentes en el tránsito terrestre, la gran cantidad de tráfico atribuida a factores como errores humanos, falla 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 eficiente [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 examinó cómo el algoritmo NEAT (NeuroEvolution of Augmenting Topologies) permite la optimización evolutiva de redes neuronales mediante 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 [12].

Este trabajo desarrolla un sistema de navegación para vehículos autónomos basado en NEAT (NeuroEvolution of

Augmenting Topologies). El vehículo, equipado con sensores virtuales, navega en un mapa 2D desde un punto inicial hasta un objetivo, utilizando métricas de distancia (Euclídea, 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 uso el entorno con la librería **Pygame** de python. Se asigna un mapa y un agente principal (Vehículo Autónomo). El entorno implementado es una fracción del mapa clásico del video juego **GTAII**, además la imagen del agente (ver Figura 1), y parámetros iniciales de configuración es proporcionada por el repositorio **NeuralNine**, también se hace uso de algunas imágenes que pertenecen al repositorio **Mihir Gandhi**. Se establecen los estados, un estado inicial donde se ubica el vehículo autónomo y un estado final. Se asignan pixeles verdes RBG (0, 0, 255) al mapa, que será el objetivo.

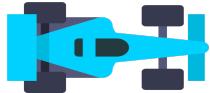


Figura 1: Agente previo a la aplicación de la red neuronal

II-A. NEAT-Python

El agente debe aprender a moverse en el entorno, NEAT permite que el agente recorra el espacio asignándole 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 estarán dadas por el color de cada pixel, dependiendo del color del pixel, 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 cuando este sale de las vías, y la 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 (Figura 1), ver Figura 2.

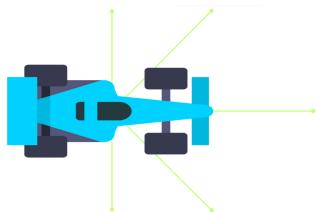


Figura 2: Asignación de sensores al agente

En la Figura 3, Hidden o capas ocultas, son nodos que aplican un procesado a la entrada, con cada generación las capas aumentan dependiendo de la ganancia, esta ganancia

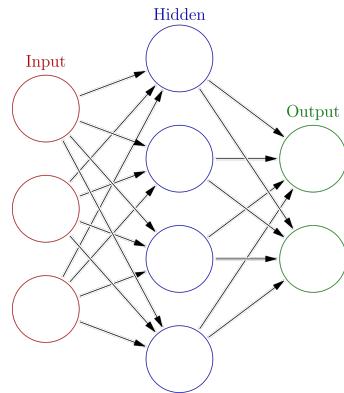


Figura 3: Estructura general de una red neuronal multicapa [14]

está dada por las acciones previas del agente. Si el agente se desplaza en el mapa sin exceder las limitaciones, se mantiene en la generación. Las generaciones en NEAT es el umbral con el que se desea finalizar la simulación, por ejemplo, si se establecen 50 generaciones cada valor del umbral se le asocian a los agentes creados por la red neural aplicando distintos valores de configuración, si en la primera generación todos los carros chocan, se pasa a la siguiente generación, en este caso la segunda, así sucesivamente hasta llegar a la última que sería la generación 50. En el entorno se crean distintos agentes, cada agente recorre el mapa, si un agente choca o se sale de la vía, esta configuración tendrá menos probabilidad de replicarse, de esta manera se garantiza una generación final donde el o los agentes aprenda a desplazarse por el entorno de forma autónoma.

En la configuración inicial, Hidden es igual a 0, puesto que se inicia con una configuración sencilla, con cada generación Hidden aumenta para que la salida dada la entrada sea la más óptima.

II-B. Fitness function

Para evaluar que tan bien son los genomas (próximas generaciones) NEAT implementa una función que evalúa que tan bien resuelve el problema en cuestión, esta función es llamada *Fitness function*. 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 los signos de estas aptitudes no son importantes, solo sus valores relativos, los pasos estarían dados por:

- Crear una red neuronal: Basada en el genoma.
- Proveer entradas y calcular salidas: Por ejemplo, para cada caso en la tabla de verdad del XOR, se proporcionan las entradas a la red y se calcula la salida.
- Calcular el error: El error se calcula entre las salidas esperadas y las salidas reales de la red.
- Asignar aptitud: Si la red produce exactamente la salida esperada, su aptitud es 1. De lo contrario, es un valor menor a 1, disminuyendo más cuanto más incorrectas sean las respuestas de la red [15].

La ecuación de actitud sería:

$$Aptitud = 1 - \sum_i (e_i - a_i)^2,$$

donde, e_i (salidas esperadas): Son los valores que se esperan obtener de la red neuronal o agente autónomo. Las salidas esperadas son las posiciones ideales o los movimientos óptimos que el agente debería realizar para alcanzar el objetivo de la manera más eficiente posible.

a_i (salidas reales): Son los valores que realmente se obtienen de la red neuronal o agente autónomo. Estas son las posiciones o movimientos que el agente realiza durante su desplazamiento en el mapa.

Función de Aptitud. En el contexto de nuestro proyecto, la función de aptitud es fundamental para evaluar y guiar el proceso de evolución de los agentes autónomos (vehículos) controlados por redes neuronales generadas mediante el algoritmo NEAT.

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 Figura 4), recibiendo entradas del entorno (como las distancias detectadas por los sensores del agente) y produciendo acciones (como cambios en la dirección y velocidad del agente).
- **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. La función de aptitud implementada es:

$$Aptitud = R_d$$

donde R_d es la recompensa por distancia al objetivo.

- **Recompensa por distancia al objetivo (R_d):**

La recompensa por distancia es inversamente proporcional a la distancia d_{agent} entre el agente y el objetivo:

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

Donde para la distancia Euclíadiana d_{agent} se calcula como:

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

Para la distancia de Manhattan d_{agent} se calcula como:

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

El tercer y último método es la distancia de Chebyshev, esta es calculada de la siguiente manera

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



Figura 4: Mapa del entorno donde el agente autónomo realizará su aprendizaje

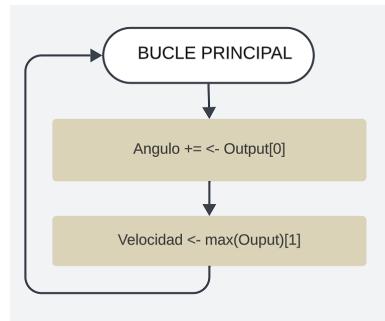


Figura 5: Bucle básico de como el agente autónomo es retroalimentado con las entradas

Aquí, (x_{agent}, y_{agent}) son las coordenadas actuales del agente, y (x_{goal}, y_{goal}) son las coordenadas de cada punto objetivo en la lista de objetivos. Esta recompensa incentiva al agente a acercarse lo más posible al objetivo.

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:

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

Entre mayor sea la aptitud, los genomas con sus características aumentan las probabilidades de ser seleccionados para la próxima generación.

II-C. Implementación de asignación entrada salida

Dadas las entradas de los sensores, se le asigna un angulo de rotación y una velocidad al agente autónomo, es un esquema básico, pero funcional ver Figura 5. Sin embargo, que pasa si las variaciones de los ángulos son muy altas, o tal vez muy bajas, o quizás la velocidad es muy lenta? Las altas variaciones de los ángulos generan inconvenientes, uno de ellos es causar que el agente gire sin parar. Si el agente recibe una entrada constante de ángulos en una misma dirección, ocasiona

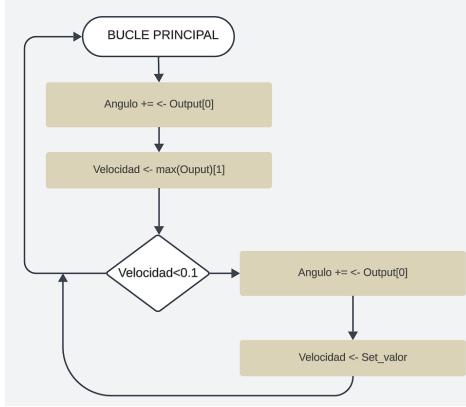


Figura 6: Diagrama de flujo de la función de aceleración o refuerzo forzado, diseñada para optimizar y reducir el estancamiento de algunos agentes en primeras generaciones.

que la suma constante de esos ángulos lo haga girar, más si dichas entradas se presentan en las primeras generaciones. Por contraparte, su desplazamiento se ve disminuido, lo que implica un estancamiento del agente. En el caso de que la entrada sea demasiada alterada pero iterando la dirección de los ángulos, su velocidad también es reducida, ya que si no lo hace, el carro será eliminado debido al inminente hecho de chocar con los bordes fuera de la zona establecida como carretera.

Para evitar alguna de las singularidades (asúmase singularidad como estancamiento, rotación sin parar) se asigna un **Set_Value** a la velocidad si la velocidad previa del agente es menor a 0.1, esto evita que el agente avance demasiado lento, optimizando en medida el tiempo de cada simulación, en este artículo, **Set_Value** es igual a 5. A esta asignación la denominamos **Refuerzo forzado** o **Método de aceleración**, y su diagrama es presentado en la Figura 6.

Una vez establecidos los parámetros iniciales, se realizan distintas simulaciones con el fin de recopilar datos para analizar el comportamiento de los agentes. Se realizaron un total de 48 simulaciones para el primer mapa (Figura 4), distribuidas en grupos de tres, 20 simulaciones para la distancia Manhattan, 20 para la Euclíadiana y 20 para la distancia Chebyshev. Se distribuyen de como se muestra en la Tabla I. A cada número de generación le corresponden 5 simulaciones, es decir en la distancia Euclíadiana según la tabla fue un total de $3 * 5 = 15$ simulaciones realizadas, donde 5 fueron las simulaciones realizadas con una generación de 20, otras 5 para la generación de 30 y por último otras 5 para la generación de 50; así para la distancia Manhattan y Chebyshev. Estas fueron realizadas con la aplicación del método **Refuerzo forzado** (ver Figura 6), además se realizaron tres simulaciones adicionales sin este método. Corresponden a una simulación de 50 generaciones para las tres distancias aplicadas, por ende el total de simulaciones son las 45 contadas anteriormente, más 3 simulaciones sin la aplicación de este método, como se observa en la Tabla II.

También se realizaron 15 simulaciones adicionales para un

Tabla I: Configuración de simulaciones del agente en el entorno por cada métrica

Simulaciones	Generaciones
Euclíadiana (5)	20, 30, 50
Manhattan (5)	30, 30, 50
Chebyshev (5)	20, 30, 50

Tabla II: Orden de simulaciones

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

segundo mapa ver Figura 7, en estas simulaciones solo se realizaron para una generación de 50, como se indica en la Tabla III, donde (5) representan las simulaciones realizadas, siguiendo el orden planteado anteriormente, 5 simulaciones realizadas para la distancia Euclíadiana con una generación de 50, y así para la distancia Manhattan y Chebyshev, para un total de $5 * 3 = 15$ simulaciones.

III. RESULTADOS

Los registros obtenidos de las simulaciones contienen el fitness promedio de los genomas, el mejor fitness de la simulación, por últimos las máximas y mínimas recompensas. Con estos datos, se obtiene la gráfica el fitness promedio con las desviaciones presentadas.

III-A. Configuración del mapa implementado en la simulación

Durante la preparación del mapa para la simulación, a los bordes principales de la carretera se les asigno un color RGB(20, 23.5, 21.6) (ver Figura 8.) Sin embargo, en la

Tabla III: Orden de simulaciones

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



Figura 7: Mapa con bloqueo de vía inferior

configuración para determinar los límites de la vía se opta por establecer que los límites son todos los colores distintos del negro.

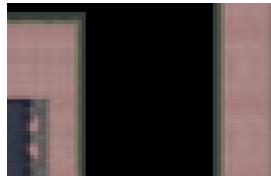


Figura 8: Bordes laterales de la vía usada en el mapa

III-B. Configuraciones principales del algoritmo NEAT

El archivo de configuración para NEAT consta de distintas secciones, cada una de ellas orientada a como será la creación de generaciones en cada simulación. En la sección [NEAT] IV, se pueden encontrar los parámetros fitness y el tamaño de la población en cada generación; en la sección [DefaultReproduction] V, podemos encontrar la configuración de elitism y survival threshold, que permiten determinar el numero de especies mas aptas de cada generación que se desea conservar, y la proporción de cada especie permitida para reproducirse en cada generación; en la sección [DefaultGenome] VI, se define los parámetros específicos para la estructura de los genomas (redes neuronales) que Neat evolucionará, como el modo de activación o activation default y activation mutate rate vea la tabla completa en Tabla S1T.

Tabla IV: [NEAT]

Variable	Valor
fitness_criterion	max
fitness_threshold	10000
pop_size	50
reset_on_extinction	True

Tabla V: [DefaultReproduction]

Variable	Valor
excess_coeff	1.0
disjoint_coeff	1.0
weight_diff_coeff	0.5
compatibility_threshold	3.0
elitism	5
survival_threshold	0.2

Tabla VI: [DefaultGenome]

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

III-C. Resultados de las simulaciones aplicando la distancia Euclidiana

Durante las simulaciones realizadas se almacenaron los datos del fitness correspondiente a esta distancia, las primeras

5 simulaciones realizadas correspondieron a un número de 50 generaciones, luego 5 para 30 generaciones y por ultimo 5 para 20 generaciones, un total de 15 simulaciones realizadas. Como resultados se obtienen las figura presentadas en la Figura 9. La zona sombreada corresponde a las desviaciones de cada simulación, estas desviaciones indican agentes alejados del promedio. Basándose en ello se puede observar que simulaciones obtuvieron la mayor cantidad de puntos.

Para los entrenamientos realizados con 50 generaciones, el entrenamiento 4 obtuvo el fitness máximo, con un valor exacto de 9990, como se aprecia en la Figura S1A-E. Ademas, este entrenamiento presentó la mayor desviación, como se observa en la Figura S2A-E.

Para los datos recopilados con un número de generaciones igual a 30, la gráfica obtenida se observa en la Figura 9 sección (B) 30 Generaciones. Los entrenamientos que presentan las mayores desviaciones fueron el primer, tercer y quinto entrenamiento. El fitness máximo corresponde al tercer entrenamiento con 9986 como se aprecia en la Figura S5B-E, ademas presentó la mayor desviación de todas para esta sección, con un valor de 9400 (ver Figura S6B-E).

La gráfica general de las desviaciones y promedios obtenidos para 20 generaciones establecidas puede observarla en la sección (C) 20 Generaciones en la Figura 9. Como se puede observar, la desviación con mayor valor se obtuvo en el segundo entrenamiento, además su puntaje fitness fue el mas alto de todos los demás con un valor de 9881, con una desviación que no sobrepasa los 9100 puntos (Fig. S3C-E y S4C-E).

Con un fitness maximo total obtenido para la distancia Euclíadiana de 9990, se obtuvo en el entrenamiento 4 con 50 generaciones, seguido de este entrenamiento, el entrenamiento 3 obtuvo el segundo fitness mas alto con 9986 para 30 generaciones, y por último el entrenamiento 2 obtuvo el tercer fitness mas alto con 9881, con 20 generaciones. La diferencia en 50 generaciones y 30 generaciones es de 4 puntos, mientras que la diferencia del fitness maximo para las 50 generaciones y 20 generaciones es de 109 puntos.

La tabla VII contiene los resultados obtenidos para las simulaciones realizadas con la distancia Euclíadiana, donde se presentan los fitness maximos obtenidos para cada entrenamiento.

III-D. Resultados de las simulaciones aplicando la distancia Manhattan

Principalmente se iniciaron las simulaciones para las 50 generaciones, de estas simulaciones se graficaron los mejores promedios y las desviaciones de cada entrenamiento, de esta manera la grafica que reúne cada uno de los entrenamientos es la Figura 10 (A) 50 Generaciones. En el entrenamiento número 4 se presentó el fitness mas alto con 9978 junto con las desviaciones mas altas de todos los entrenamientos, de aproximadamente 9100 puntos, con una desviación cercana a los 9200 (Fig S7A-M y S8A-M).

En la gráfica general para las simulaciones con 30 generaciones se observa los fitness promedio de cada entrenamiento

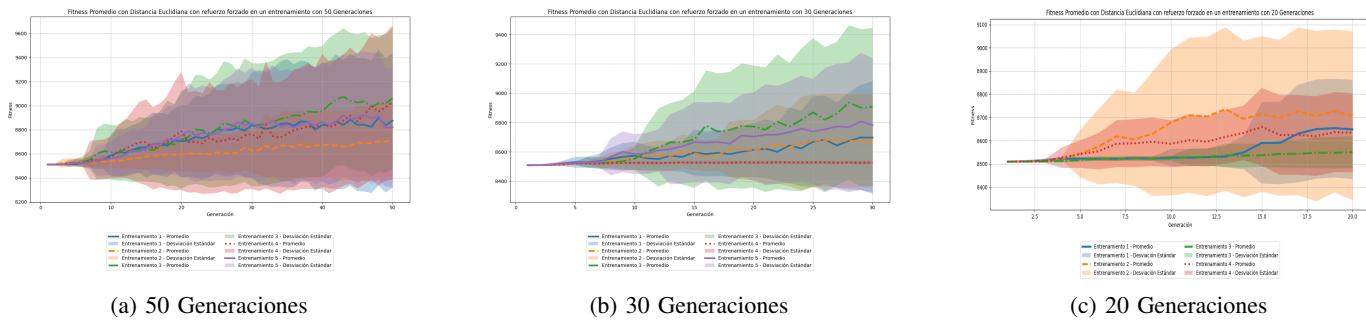


Figura 9: Gráficas por generaciones para la distancia Euclíadiana

Tabla VII: Resumen de Fitness por Generaciones y Entrenamientos para la distancia Euclíadiana

Generaciones y Fitness por Entrenamiento					
50 Generaciones		30 Generaciones		20 Generaciones	
Nº Entrenamiento	Fitness	Nº Entrenamiento	Fitness	Nº Entrenamiento	Fitness
1	9990	1	9981	1	9632
2	9626	2	9883	2	9881
3	9985	3	9986	3	8773
4	9990	4	8544	4	9237
5	9982	5	9984	5	8782

y las desviaciones ver la Figura 10 (B) 30 Generaciones. El primer entrenamiento presento el fitness mas alto de todos los entrenamientos, con un puntaje de 9990, con una desviación que supera los 9200 puntos (Fig. S1B-M y S2B-M).

En la gráfica general para las simulaciones con 20 generaciones se observan los fitness promedio de cada entrenamiento y las desviaciones Figura 10 (C) 20 Generaciones. El entrenamiento número dos obtuvo el mejor fitness, con un puntaje de 9980, este entrenamiento también posee la mayor desviación de todas, sin embargo esta desviación no sucede los 8900 puntos (Fig. S3C-M y S4C-M). El entrenamiento número 4 obtuvo un fitness maximo de 8879, sin embargo su fitness promedio y desviación empezó a disminuir a partir de la generación numero 15 ver (Fig. S8C-M)

El valor de fitness maximo obtenido en la distancia Manhattan corresponde al entrenamiento numero 1 con 9990 para 30 generaciones, seguido de este entrenamiento, el entrenamiento 2 obtuvo el segundo fitness mas alto con 9980 para 20 generaciones, y por ultimo el entrenamiento 4 obtuvo el tercer fitness mas alto con 9978, con 20 generaciones. La diferencia entre el fitness maximo obtenido para 30 generaciones y 20 generaciones es de 10 puntos, mientras que la diferencia del fitness maximo para las 30 generaciones y 50 generaciones es de 12 puntos.

La tabla **VIII** contiene los resultados obtenidos para las simulaciones realizadas con la distancia Manhattan, donde se presentan los fitness máximos obtenidos para cada entrenamiento.

III-E. Resultados de las simulaciones aplicando la distancia de Chebyshev

Para las simulaciones realizadas con la distancia de Chebyshev, se obtuvieron los resultados de las gráficas ge-

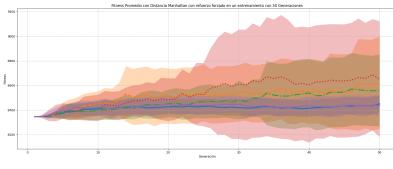
nerales para las 50 generaciones, 30 generaciones y 20 generaciones, como se observa en la Figura 11. La grafica que reúne cada uno de los entrenamientos realizados para las 50 Generaciones es la Figura 11 (A) *50 Generaciones*. En el entrenamiento número 1 se presentó el fitness mas alto con 9988 junto con las desviaciones mas altas de todos los entrenamientos, de aproximadamente 9400 puntos (Fig S1A-C y S2A-C).

En la gráfica general para las simulaciones con 30 generaciones se observa los fitness promedio de cada entrenamiento y las desviaciones ver la Figura 11 (B) 30 Generaciones. El segundo entrenamiento presento el fitness mas alto de todos los entrenamientos, con un puntaje de 9994, con una desviación que supera los 9400 puntos (Fig. S3B-C y S4B-C).

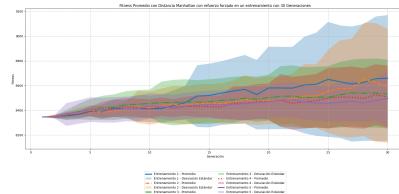
En la gráfica general para las simulaciones con 20 generaciones se observan los fitness promedio de cada entrenamiento y las desviaciones Figura 11 (C) 20 Generaciones. El entrenamiento número dos obtuvo el mejor fitness, con un puntaje de 9990, este entrenamiento también posee la mayor desviación de todas, sin embargo esta desviación no supera los 9100 puntos (Fig. S3C-C y S4C-C).

El valor de fitness maximo obtenido en la distancia Chebyshev corresponde al entrenamiento numero 2 con 9994 para 30 generaciones, seguido de este entrenamiento, el entrenamiento 2 obtuvo el segundo fitness mas alto con 9990 para 20 generaciones, y por ultimo el entrenamiento 1 obtuvo el tercer fitness mas alto con 9988, con 50 generaciones. La diferencia entre el fitness maximo obtenido para 30 generaciones y 20 generaciones es de 4 puntos, mientras que la diferencia del fitness maximo para las 30 generaciones y 50 generaciones es de 6 puntos.

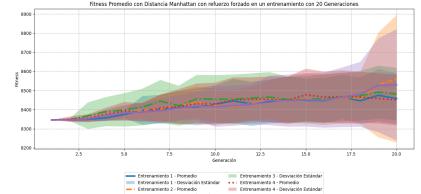
La tabla IX contiene los resultados obtenidos para las si-



(a) 50 Generaciones



(b) 30 Generaciones



(c) 20 Generaciones

Figura 10: Gráficas por generaciones para la distancia Manhattan

Tabla VIII: Resumen de Fitness por Generaciones y Entrenamientos para la distancia Manhattan

Generaciones y Fitness por Entrenamiento							
50 Generaciones		30 Generaciones		20 Generaciones			
Nº Entrenamiento	Fitness	Nº Entrenamiento	Fitness	Nº Entrenamiento	Fitness		
1	8531	1	9990	1	8766		
2	9688	2	9987	2	9980		
3	9621	3	9618	3	8771		
4	9978	4	9623	4	8799		
5	8677	5	9630	5	9360		

mulaciones realizadas con la distancia Manhattan, donde se presentan los fitness máximos obtenidos para cada entrenamiento.

III-F. Entrenamiento sin el Refuerzo forzado

Para esta sección solo se realizó un entrenamiento de 50 generaciones para cada métrica de distancia con el fin de analizar que tan óptimo era realizar las simulaciones sin la aplicación del método descrito en la Figura 6. En la aplicación de la distancia Euclídea el mayor fitness obtenido es de 8536 (Fig. S1-EN), por otra parte la desviación obtenida describe el comportamiento de los agentes en cada generación, cómo podrá observarse en la Figura 12, el mayor incremento de fitness solo se obtuvo en la generación 10, luego de la generación 12 no incrementaba el aprendizaje en los agentes.

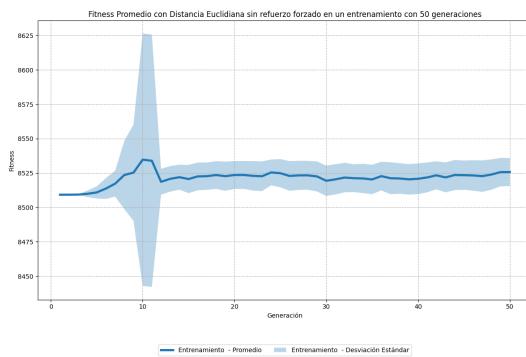


Figura 12: Desviación y promedio fitness para la distancia Euclídea sin la aplicación del Refuerzo forzado

Implementando la distancia Manhattan el puntaje solo se diferencia en 1 de la distancia Euclídea, con un puntaje de 8535, la grafica de desviación mantuvo el mismo comportamiento que la distancia Euclídea, sin embargo la desviación

en la generación 10 superó los 8625 puntos (Fig. S1-MN y S2-MN).

Por último, en esta sección la distancia de Chebyshev presentó los mejores resultados, con un puntaje fitness máximo obtenido de 9885 (Fig. S1-CN), y una desviación que ronda los 8800 puntos. La diferencia principal con respecto a las otras métricas de distancias radica en el aprendizaje constante de los agentes con cada generación, como se observa en la Figura 13.

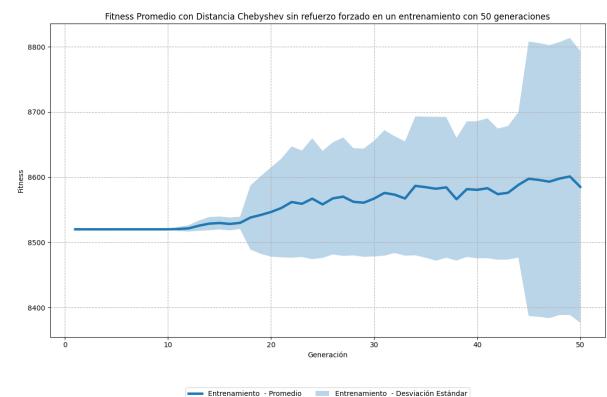


Figura 13: Desviación y promedio fitness para la distancia Chebyshev sin la aplicación del Refuerzo forzado

III-G. Simulaciones de 50 generaciones realizadas en el segundo mapa con el Refuerzo Forzado

Se realizaron 5 simulaciones de 50 generaciones para cada sistema de recompensas en el segundo mapa ver figura 7, Euclídeo, Manhattan y Chebyshev. Con los datos recopilados de estas simulaciones, se analizan las gráficas generales, y basándose en los resultados previos se hace una descripción

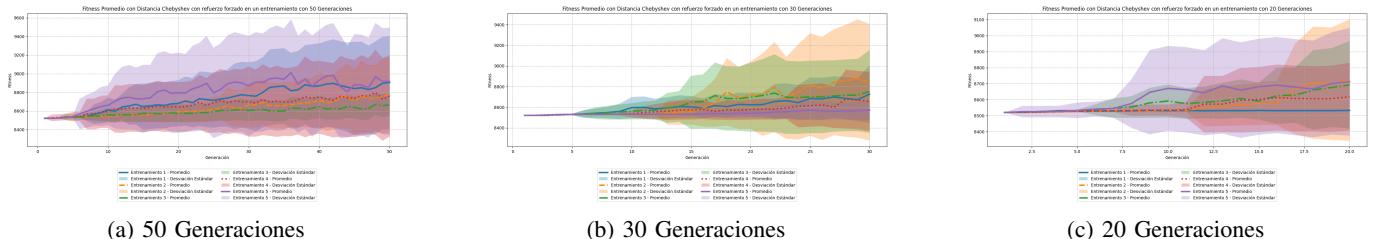


Figura 11: Gráficas por generaciones para la distancia Chebyshev

Tabla IX: Resumen de Fitness por Generaciones y Entrenamientos para la distancia Manhattan

Generaciones y Fitness por Entrenamiento					
50 Generaciones		30 Generaciones		20 Generaciones	
Nº Entrenamiento	Fitness	Nº Entrenamiento	Fitness	Nº Entrenamiento	Fitness
1	9988	1	9665	1	8543
2	9973	2	9994	2	9990
3	8777	3	9974	3	9906
4	9984	4	9884	4	9680
5	9986	5	8787	5	9882

III-G1. Distancia Euclíadiana: La gráfica general obtenida después de cada entrenamiento para la distancia Euclíadiana es la Figura 14 (a) *Distancia Euclíadiana*.

El mejor fitness obtenido es de 9684, por el tercer y quinto entrenamiento. El quinto entrenamiento en comparación con el tercero presenta una desviación que supera los 9100 puntos, por otra parte el tercero no supera los 9000 puntos en desviación, quedándose incluso por debajo del primer entrenamiento. El primer entrenamiento obtuvo un fitness de 9456 puntos, seguido del segundo con 9443, por último el menor fitness fue obtenido por el cuarto entrenamiento con 8933 puntos (Fig. S1-EM2 hasta S9-EM2)

III-G2. Distancia Manhattan: La gráfica general obtenida después de cada entrenamiento para la distancia Manhattan puede observarla en la Figura 14 (b) *Distancia Manhattan*.

Para la distancia Manhattan el fitness más alto obtenido en estas simulaciones realizadas para es para el entrenamiento numero 2, con un puntaje máximo de 9890 y una desviación que supera los 8900 puntos. Además obtuvo un incremento en el fitness promedio a partir de la generación 47. Con un fitness maximo de 9679, los entenamientos 4 y 3 obtuvieron los siguiente puntajes más altos. El cuarto entrenamiento empezó a disminuir su fitness promedio a partir de la generación 30, mientras que el tercer entrenamiento mantuvo un fitness ascendente. El quinto y primer entrenamiento obtuvieron los puntajes mas bajos, con 9507 y 8532 respectivamente. Las desviaciones presentadas por el quinto entrenamiento superan los 8800 puntos, mientras que el primer entrenamiento no pasa de los 8550 puntos (Fig. S1-MM2 hasta S10-MM2)

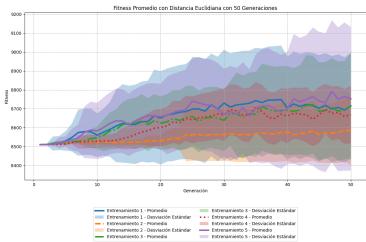
III-G3. Distancia Chebyshev: La gráfica general obtenida después de cada entrenamiento para la distancia Chebyshev es la Figura 14 (c) *Distancia Chebyshev*.

El entrenamiento que mayor fitness obtuvo en esta sección corresponde al cuarto entrenamiento con un puntaje de 9683, con una desviación que se acerca a los 9100 puttos. El segundo

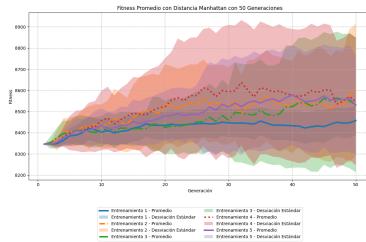
entrenamiento obtuvo el segundo mejor puntaje con 9671, con una desviación que no supera los 9000 puntos, sin embargo una característica particular de este entrenamiento es que no posee desviaciones elevadas hasta la cuarta generación, sin embargo a partir de la generación numero 5, la desviación se despiara a los 8700 puntos hasta llegar de forma gradual a los 8900. El entrenamiento numero 3 y 1 obtuvieron un puntaje fitness muy cercano entre ellos, solo con una diferencia de tres puntos. Para el tercer entrenamiento el fitness maximo obtenido es de 9764, mientras que para el primero fue de 9571 ademas de poseer una desviación mayor. Por ultimo el peor fitness obtenido lo obtuvo el entrenamiento numero 5 con 8985 puntos y un decrecimiento en la desviación a partir de la generación 30 (Fig. S1-CM2 hasta S10-CM2).

IV. CONCLUSIONES

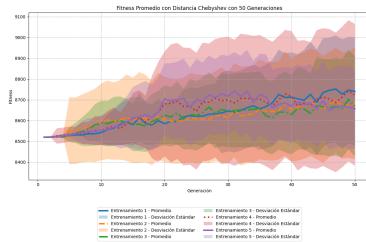
La implementación de la librería NEAT para un vehículo autónomo permitió analizar distintos aspectos del aprendizaje y desempeño del agente en diversos entornos. En este caso se realizaron dos pruebas, el entorno libre de obstáculos, mientras que en la otra se añadió un bloqueo de ruta. Se observó que el agente mostró un mejor rendimiento en el entorno libre de obstáculos, en comparación del entorno con la ruta bloqueada. Al agente le tomaba más generaciones aprender en comparación cuándo alguna vías era obstruida. En cuanto a las métricas de distancia, la distancia Chebyshev demostró ser más óptima que la Manhattan y Euclíadiana en entornos libres de obstáculos. Sin embargo, en escenarios con obstáculos, la distancia Euclíadiana ofreció un mejor desempeño obteniendo un alto valor fitness. La distancia Euclíadiana aumentaba al igual que la de Chebyshev con el pasar de las generaciones, sin embargo la distancia Manhattan obtuvo su mayor fitness con una generación de 30. Con los resultados obtenidos se puede concluir que para la distancia Manhattan y Chebyshev con 30 generaciones ya se obtiene un buen fitness.



(a) Distancia Euclidiana



(b) Distancia Manhattan



(c) 20 Generaciones

Figura 14: Gráficas generales de promedios fitness y desviaciones para el segundo mapa

Por otro lado, la implementación del refuerzo forzado generó mejoras importantes, como disminuir el tiempo requerido para que el agente alcanzara un fitness significativo. Sin la aplicación del refuerzo forzado, el agente no lograba obtener fitness altos, lo que se traduce en un aprendizaje más lento y menos eficiente. Sin embargo la distancia de Chebyshev demostró ser las más eficiente sin la aplicación del refuerzo forzado, obteniendo un fitness de 9885, superando a la Euclidiana y Manhattan. La métrica con los mejores resultados generales fue la distancia Chebyshev, demostrando el incremento del fitness en cada generación, con la aplicación o no del Refuerzo Forzado.

REFERENCIAS

- [1] Organización Mundial de la Salud, *Informe sobre la situación mundial de la seguridad vial 2023*. Ginebra, Suiza: Organización Mundial de la Salud, 2023. dirección: <https://www.who.int/publications/item/9789241565684>.
- [2] R. Montezuma y J. Erazo, «El derecho a la vida en la movilidad urbana y el espacio público en América Latina,» *Intersecciones urbanas: origen y ...*, 2008. dirección: 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,» Tesis doct., CORE, 2015. dirección: <https://core.ac.uk/download/pdf/132743165.pdf>.
- [4] S. Arshad, M. Sualeh, D. Kim, D. V. Nam y G.-W. Kim, «Clothoid: An Integrated Hierarchical Framework for Autonomous Driving in a Dynamic Urban Environment,» *English, Sensors*, vol. 20, n.º 18, pág. 5053, 2020, ISSN: 1424-8220. DOI: [10.3390/s20185053](https://doi.org/10.3390/s20185053). dirección: <https://research.ebsco.com/linkprocessor/plink?id=0d501679-ad5e-3650-956f-f38dba4076eb>.
- [5] Z. Lin et al., «Policy Iteration Based Approximate Dynamic Programming Toward Autonomous Driving in Constrained Dynamic Environment,» *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, n.º 5, págs. 5003-5013, 2023. DOI: [10.1109/TITS.2023.3237568](https://doi.org/10.1109/TITS.2023.3237568).
- [6] I. Lamouik, A. Yahyaouy y M. A. Sabri, «Smart multi-agent traffic coordinator for autonomous vehicles at intersections,» en *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSiP)*, 2017, págs. 1-6. DOI: [10.1109/ATSiP.2017.8075564](https://doi.org/10.1109/ATSiP.2017.8075564).
- [7] R. Inamdar, S. K. Sundarr, D. Khandelwal, V. D. Sahu y 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, pág. 100810, 2024, ISSN: 2772-6711. DOI: <https://doi.org/10.1016/j.prime.2024.100810>. dirección: <https://www.sciencedirect.com/science/article/pii/S2772671124003905>.
- [8] Z. Huang, «Reinforcement learning based adaptive control method for traffic lights in intelligent transportation,» *Alexandria Engineering Journal*, vol. 106, págs. 381-391, 2024, ISSN: 1110-0168. DOI: <https://doi.org/10.1016/j.aej.2024.07.046>. dirección: <https://www.sciencedirect.com/science/article/pii/S111001682400766X>.
- [9] L. Luo, N. Zhao, Y. Zhu e Y. Sun, «A* guiding DQN algorithm for automated guided vehicle pathfinding problem of robotic mobile fulfillment systems,» *Computers & Industrial Engineering*, vol. 178, pág. 109-112, 2023, ISSN: 0360-8352. DOI: <https://doi.org/10.1016/j.cie.2023.109112>. dirección: <https://www.sciencedirect.com/science/article/pii/S0360835223001365>.
- [10] M. E. Yuksel, «Agent-based evacuation modeling with multiple exits using NeuroEvolution of Augmenting Topologies,» *Advanced Engineering Informatics*, vol. 35, págs. 30-55, 2018, ISSN: 1474-0346. DOI: <https://doi.org/10.1016/j.aei.2017.11.003>. dirección: <https://www.sciencedirect.com/science/article/pii/S1474034616304281>.
- [11] T. J. Ikonen e I. Harjunkoski, «Decision-making of online rescheduling procedures using neuroevolution of augmenting topologies,» en *29th European Symposium on Computer Aided Process Engineering*, ép. Computer Aided Chemical Engineering, A. A. Kiss, E. Zondervan, R. Lakerveld y L. Özkan, eds., vol. 46, Elsevier, 2019, págs. 1177-1182. DOI: <https://doi.org/10.1016/j.caep.2019.1177-1182>.

- 1016 / B978 - 0 - 12 - 818634 - 3 . 50197 - 1. dirección:
<https://www.sciencedirect.com/science/article/pii/B9780128186343501971>.
- [12] F. C. Guardo, «Evolución de redes neuronales mediante topologías aumentadas,» Trabajo Fin de Grado, Universidad Autónoma de Madrid, Madrid, España, jun. de 2019.
 - [13] R. Lauckner y H. Kolivand, «NEAT Algorithm in Autonomous Vehicles,» 2023, Preprint, not peer-reviewed. dirección: <https://ssrn.com/abstract=4644203>.
 - [14] J. Portilla, «A Beginner's Guide to Neural Networks in Python,» *Springboard Blog*, 2017. dirección: <https://www.springboard.com/blog/data-science/beginners-guide-neural-network-in-python-scikit-learn-0-18/>.
 - [15] L. CodeReclaimers, *Overview of the basic XOR example (evolve-feedforward.py)*, Accessed: 2024-11-18, 2019. dirección: https://neat-python.readthedocs.io/en/latest/xor_example.html#fitness-function.