Generación Efectiva de Controladores Difusos Evolutivos para Carreras en Simuladores de Coches

Antonio M. Mora
Depto. de Teoría de la Señal, Telemática y Comunicaciones
ETSIIT-CITIC, Universidad de Granada, España
amorag@ugr.es

Mohammed Salem
Department of Computer Sciences
University of Mascara, Algeria
salem@univ-mascara.dz

Juan J. Merelo
Depto. de Arquitectura y Tecnología de Computadores
ETSIIT-CITIC, Universidad de Granada, España
jmerelo@ugr.es

Pablo García-Sánchez
Departamento de Informática
Universidad de Cádiz, España
pablo.garciasanchez@uca.es

***Resumen*—Los simuladores de carreras de coches han sido utilizados durante mucho tiempo como un entorno para probar algoritmos de control autónomo de vehículos. Constituyen un entorno en el para evaluar todo tipo de algoritmos, incluyendo metaheurísticas, como por ejemplo Algoritmos Evolutivos. Sin embargo, el mayor desafío en este tipo de algoritmos evolutivos es diseñar un proceso de evaluación fiable y eficaz para los individuos, que se traducirá en la generación de buenas soluciones para el problema abordado: encontrar un controlador que sea capaz de ganar en una amplia gama de pistas y con distinto número de oponentes de diversa dificultad. Dicha evaluación no sólo implica el diseño de una función de aptitud (fitness) adecuada, que represente el nivel de calidad de los controladores de automóviles, sino también la selección de la mejor solución para la problema de optimización que se está resolviendo. Este paso puede resultar muy complejo, debido a la incertidumbre/ruido presente en el problema (condiciones meteorológicas y de la pista, comportamiento impredecible de otros conductores, etc). Por tanto, este trabajo presenta una serie de propuestas para realizar la evaluación de controladores difusos optimizados, analizando el rendimiento o calidad de los mismos en comparación con propuestas anteriores. Esto facilitará el diseño automático de un controlador autónomo para el simulador de carreras de coches TORCS. Para ello se parte de los resultados preliminares obtenidos en trabajos anteriores y se rediseña parte del procedimiento de evaluación del fitness. De modo que se estudian dos funciones de fitness diferentes en varios experimentos, junto con un novedoso enfoque basado en la competición para la selección del mejor individuo en cada generación.**

***Index Terms*—simuladores de carreras de coches,TORCS, controladores difusos, conductores autónomos, algoritmos genéticos, optimización, evaluación**

I. Introducción

La conducción autónoma es un problema que surge en muchos entornos, como los aviones, barcos, trenes o vehículos submarinos no tripulados. Recientemente están empezando a aparecer los primeros coches autónomos, en los que no es necesario un conductor humano. Todos estos sistemas se basan, en general, en una serie de entradas de sensores que incluyen velocidad real, obstáculos y otros vehículos. Considerando esas entradas, el controlador tendrá que tomar una decisión sobre la velocidad y dirección óptimas [7]. La prueba de diferentes metodologías de conducción autónoma en el mundo real suele estar reservada a unas pocas empresas importantes, por lo que las pruebas de algoritmos se realizan normalmente en entornos simulados, los cuales ofrecen, a su vez, el incentivo de poder comparar mediante competiciones distintas propuestas.

En este artículo, utilizaremos *The Open Racing Car Simulator (TORCS)* [19], un simulador de carreras muy realista que proporciona un gran entorno de pruebas para la implementación y evaluación de conductores/controladores autónomos. TORCS ha sido utilizado en diversas ocasiones para la celebración de competiciones de Inteligencia Artificial (IA), donde el objetivo es crear los mejores pilotos autónomos de carreras [10]–[12]. Además de evaluar nuestros controladores enfrentándolos contra otros, el simulador puede ser utilizado como un entorno aislado para optimizar la conducción en una carrera en solitario.

Los Algoritmos Evolutivos (AEs) [1] han sido aplicados frecuentemente como un método de optimización de propósito general en este área, los cuales son habitualmente combinados con motores de comportamiento que gobiernan diferentes partes del coche [6], [15], [16] e incluso con aproximaciones basadas en Deep Learning [8]. Dichos motores de comportamiento han sido modelados recientemente mediante controladores difusos [4], [9], [14], los cuales aplican Lógica Difusa (*fuzzy logic* o FL en inglés) [2]. Dicha técnica es bastante adecuada para definir este tipo de agentes autónomos, ya que está inspirada en razonamientos que los humanos aplicamos al conducir. Un controlador difuso funciona con variables lingüísticas, y por ejemplo, girará *ligeramente* a la derecha cuando la siguiente curva esté cerca; si bien estos controladores tienen que ser diseñados para asociar apropiadamente las entradas a las salidas deseadas en situaciones específicas.

Los autores de este trabajo presentamos anteriormente una propuesta que combinaba dos controladores difusos especializados, diseñados a mano, que eran capaces de decidir el ángulo de giro adecuado del coche y la velocidad deseada en cada

punto (o en cada instante de juego, *tick*) durante una carrera [17]. Dicha propuesta fue revisada más tarde [18], optimizando los parámetros de sus funciones de pertenencia por medio de un Algoritmo Genético (AG) [3], mejorando el rendimiento del diseño manual, así como el de otros controladores de la literatura con los que se comparó.

Esto probó que los algoritmos evolutivos eran capaces de definir un conjunto de parámetros para el controlador difuso mucho mejor que los de un diseño hecho a mano, pero al mismo tiempo desveló varios desafíos. En general, los AEs optimizan la función de fitness que se utiliza, de modo que los controladores difusos evolucionados (en lo sucesivo, "FCs", o *Fuzzy Controllers*) serán finalmente tan buenos como lo permita dicha función. El problema es que en este caso no se puede considerar como función de aptitud la posición alcanzada por el FC en todas las carreras posibles en todas las pistas posibles y contra todos los oponentes posibles, así que tenemos que conformarnos con un *sustituto* de la función de fitness en un entorno muy limitado.

De modo que para mejorar el proceso de evaluación primero se optó por eliminar los oponentes, haciendo pruebas en carreras en solitario. Además se estudió qué factores relacionados con la velocidad, el daño y el tiempo por vuelta se deberían tener en cuenta para la evaluación de cada controlador (fueron incluidos en la función de fitness inicial).

Los resultados obtenidos en aquellos trabajos fueron alentadores, pero el modelo propuesto era algo ïmpreciso", ya que había que decidir cuál es la mejor pista para realizar el entrenamiento, así como los factores de carrera en solitario con mayor impacto en el rendimiento en una carrera real. En este sentido, en el presente estudio se ha elegido de forma analítica una pista concreta y representativa para los entrenamientos, la cual tiene una longitud media y combina tramos rectos con zonas de muchas curvas. A su vez se han combinado en la función de fitness, en dos aproximaciones distintas, los términos relacionados con la velocidad durante la carrera (a maximizar) y el daño recibido (a minimizar), que consideramos como los más relevantes. Además, se ha tenido en cuenta que el proceso de evaluación del fitness tiene un cierto grado de incertidumbre, como el hecho de que los daños y algunas condiciones de la pista pueden variar aleatoriamente en función del tipo de circuito. Por ello, en lugar de seleccionar directamente el mejor controlador en base a su valor de fitness como hicimos en los trabajos anteriores, en esta propuesta vamos a llevar a cabo una carrera entre los mejores individuos (controladores) para seleccionar al ganador.

El objetivo de estos tres factores (elección de pista, función de fitness y selección de ganador) es crear un algoritmo para crear controladores que sea más robusto y eficiente.

II. El simulador y los controladores

En esta sección se presentan el entorno de investigación considerado (el simulador de carreas de coches), y se describen los controladores definidos por los autores.

II-A. The Open Racing Car Simulator

TORCS [19] es un simulador de carreras de coches de código abierto, realista, multijugador y modular que permite a los usuarios competir contra otros oponentes controlados por ordenador. Es un entorno de pruebas bastante fiable y muy utilizado en investigación en inteligencia artificial.

Cada coche en TORCS manejará un conjunto de sensores y valores del entorno, por ejemplo distancias a bordes de la pista o a otros vehículos rivales, el combustible restante, la marcha actual, la posición en la carrera, la velocidad, o los daños, entre otros. Estos valores serán considerados por los conductores autónomos o *controladores*, para gestionar el coche utilizando los llamados *actuadores*: giro del volante, acelerador, freno y cambios de marcha.

II-B. Subcontroladores Difusos

El controlador diseñado por los autores se basa en el modelo de sensores y actuadores de la *Simulated Car Racing Competition* [12].

Sin embargo, la velocidad objetivo y el ángulo de giro de la dirección se calculan mediante dos sistemas modulares y especializados [17]. Estos subcontroladores incorporaron lógica difusa y consideran cinco sensores de posición. Partiendo de ellos, se aplicaron AGs para mejorarlos de manera automática.

El *subcontrolador difuso de velocidad objetivo* pretende estimar la velocidad objetivo óptima del coche, tanto en las partes rectas, como en las curvas de la pista. Para ello tiene en cuenta dos criterios: moverse lo más rápido posible y de la manera más segura (con el menor daño posible). Esta estimación se basa en dos casos generales: si el coche encara una línea recta, la velocidad objetivo tomará un valor máximo (*maxSpeed* km/h). Sin embargo, si está cerca de una curva, se disminuirá la velocidad actual a un valor incluido en el intervalo *[minSpeed, maxSpeed]* km/h.

Este controlador difuso tiene una salida, la velocidad, y tres valores de entrada (ver Figura 1):

- `Front` = Sensor_9: Distancia frontal al borde de la pista (ángulo 0°).
- `M5` = max (Sensor_8, Sensor_10): distancia máxima al borde de la pista con un ángulo de +5°y -5°con respecto a `Front`.
- `M10` = max (Sensor_7, Sensor_11): distancia máxima al borde de la pista con un ángulo de +10°y -10°.

Se trata de un sistema difuso de tipo Mamdani [5] con tres funciones de pertenencia (MF) trapezoidales para cada variable de entrada. En [18] se optimizaron con un AG los conjuntos de parámetros que definen las funciones de pertenencia, mejorando en gran medida los resultados obtenidos.

Además, el controlador está basado en un conjunto de reglas difusas, diseñadas para maximizar la velocidad del coche dependiendo de las distancias detectadas al borde de la pista. Dichas reglas pueden verse en [17].

El segundo es el *subcontrolador difuso para el giro del volante*, que pretende determinar el mejor ángulo de giro en base a una estimación de la posición objetivo del coche. Su

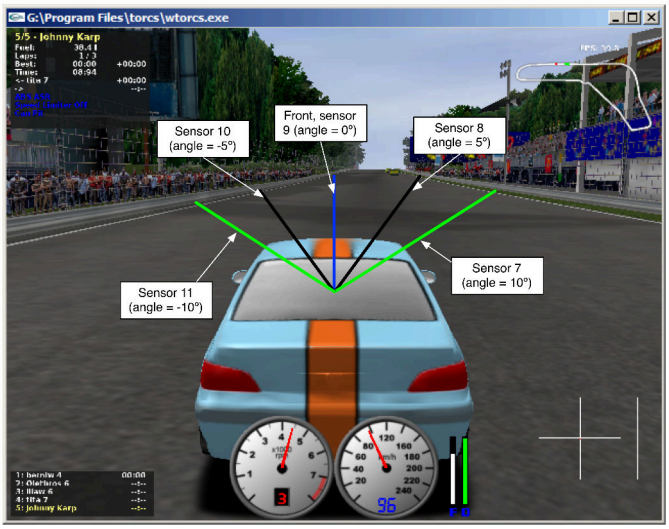

Figura 1. Sensores considerados de los 18 que tiene asociados cada coche en TORCS

estructura es similar a la del controlador anterior, basándose en los mismos sensores, pero considerando el giro como salida del mismo.

De modo que, como reglas generales: si el coche circula en línea recta, se fijará como posición objetivo el centro del carril por el que circula; mientras que, si el coche está cerca de una curva a derecha o izquierda, se acercará a la curva dejando un espacio entre el coche y el borde de la pista para evitar la pérdida de control.

Para detectar las curvas, el controlador revisa los valores de los sensores (`M10`, `M5` y `Front`), de modo que si el valor en el sensor frontal es el mayor, hay un tramo recto; mientras que si los valores de `M5` y `M10` con ángulos positivos (+5 y +10) son los mayores, habrá una curva a la derecha, y viceversa.

El controlador usa un conjunto de reglas que fue definido modelando el comportamiento de un conductor humano [17].

III. Algoritmo Genético

El algoritmo de optimización propuesto tiene como objetivo encontrar los parámetros óptimos de las funciones de pertenencia de los dos subcontroladores previamente introducidos.

El AG comienza creando una población inicial con valores aleatorios (distribución uniforme) para los parámetros en el rango definido $[0, 100]$. La idoneidad de cada solución candidata se calcula inyectando sus valores genéticos a los parámetros de las funciones de membresía de los dos subcontroladores difusos. El controlador autónomo definido se utiliza para conducir un coche en una carrera de 20 vueltas en un circuito sin oponentes en TORCS, y los resultados (velocidad máxima, mínima y media, junto con el daño obtenido) se utilizan para calcular el valor del fitness correspondiente.

Los controladores difusos tienen funciones de pertenencia trapezoidales, que siguen la Ecuación 1. En un controlador de este tipo, las reglas difusas se aplican a términos lingüísicos, que califican las llamadas variables lingüísticas y que se definen mediante funciones de pertenencia que dependen de un conjunto de parámetros que determinan su forma y su 'funcionamento'. De modo que se aplicó un AG para optimizar dichos parámetros y determinar la partición difusa de la variable lingüística [20]. Las variables lingüísticas de entrada en nuestro problema serán `Front`, `M5` y `M10`.

Una función de pertenencia (MF) trapezoidal, se define como:

$$\mu_A(x) = \begin{cases} \frac{x-x_1}{x_2-x_1}, & x_1 \le x \le x_2 \\ 1, & x_2 \le x \le x_3 \\ \frac{x_4-x}{x_4-x_3}, & x_3 \le x \le x_4 \\ 0, & else \end{cases} \quad (1)$$

with:

$$x_1 \le x_2 \le x_3 \le x_4 \quad (2)$$

Como se puede ver, una MF está determinada por cuatro parámetros x_1, x_2, x_3 y x_4, los cuales tienen valores en el intervalo *[a, b]* (Figura 2).

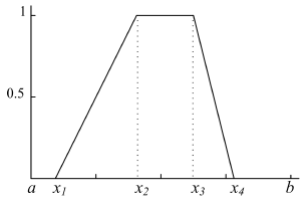

Figura 2. Función de pertenencia trapezoidal

De modo que una partición difusa con *n* MF trapezoidales se define mediante *2n* variables ($a = x_1, x_2, .., x_{2n} = b$) (Ecuación 4). Con:

$$a = x_1 \le x_2 \le ... \le x_{2n-1} \le x_{2n} = b \quad (3)$$

$$\mu_{A1}(x) = \begin{cases} 1, & x_1 \le x \le x_2 \\ \frac{x_3-x}{x_3-x_2}, & x_2 \le x \le x_3 \\ 0, & x > x_3 \end{cases}$$

$$\mu_{Ai}(x) = \begin{cases} 0, & x \le x_{2i-2} \\ \frac{x-x_{2i-2}}{x_{2i-1}-x_{2i-2}}, & x_{2i-2} \le x \le x_{2i-1}, n = 2, ..., i-1 \\ 1, & x_{2i-1} \le x \le x_{2i} \\ \frac{x_{2i+1}-x}{x_{2i+1}-x_{2i}}, & x_{2i} \le x \le x_{2i+1} \\ 0, & x > x_{2i+1} \end{cases}$$

$$\mu_{An}(x) = \begin{cases} 0, & x \le x_{2n-2} \\ \frac{x-x_{2n-2}}{x_{2n-1}-x_{2n-2}}, & x_{2n-2} \le x \le x_{2n-1} \\ 1, & x > x_{2n-1} \end{cases} \quad (4)$$

Cuando el número de parámetros es reducido y sus rangos de variación están bien definidos, un AG con codificación binaria será suficiente para encontrar sus valores óptimos. Sin embargo, dado que en nuestro problema las salidas deseadas requieren precisión y el intervalo de variación de cada

parámetro no esá bien determinado, hemos considerado una codificación real de los parámetros, que hemos dispuesto en un vector que incluye todas las variables a optimizar. De modo que cada individuo de nuestro AG será un vector con esa estructura, que tendrá 18 parámetros, 6 por variable.

La *inicialización de la población* de cromosomas/individuos se realiza asignando valores aleatorios en un rango de variación predefinido [3], a fin de comenzar la optimización desde un conjunto de valores prometedores [17].

Para llevar a cabo la evolución se han considerado: la selección de padres basada en *torneo*, un operador de *cruce aritmético simple en dos puntos* [21] y un operador de *mutación no uniforme* [13], por ser operadores genéticos bien contrastados en la literatura.

El objetivo principal del controlador autónomo en este entorno es ganar tantas carreras como sea posible. Sin embargo, tenemos que optimizar el caso más general mediante la realización de *carreras de entrenamiento en solitario*, que serán menos sensibles a la presencia de ruido/incertidumbre debido a la participación de otros controladores.

En propuestas anteriores nos enfocamos en la minimización del daño recibido ($damage$) y el tiempo de vuelta $LapTime$, a la par que intentábamos maximizar la velocidad máxima alcanzada $TopSpeed$. Sin embargo, en este estudio, nos hemos centrado en un .[en]foque más humano", es decir, tratar de conducir lo más rápido posible en cada una de las partes de la pista evitando los daños. Por lo tanto, hemos considerado:

- Velocidad mínima ($MinSpeed$): para mejorar la conducción en zonas difíciles del circuito, como las zonas de curvas.
- Velocidad Máxima ($MaxSpeed$): para optimizar la conducción en las zonas rectas o sencillas de la pista.
- Velocidad Media ($AVGSpeed$): que modelará el comportamiento general en la pista.
- Daño ($Damage$): con el objetivo de crear controladores seguros, que sean capaces de terminar la carrera en cualquier circunstancia.

De modo que se han combinado estos factores en dos posibles funciones de evaluación/fitness:

GFC-MMS:

$$f_1 = \frac{(MinSpeed * MaxSpeed)}{Damage+1} \quad (5)$$

GFC-AVS:

$$f_2 = \frac{AVGSpeed}{Damage+1} \quad (6)$$

Como se puede observar, en la primera función el objetivo es maximizar las velocidades mínima y máxima, a la par que se minimiza el daño recibido. En la segunda, se intenta maximizar la velocidad media del controlador en el circuito completo.

La evaluación de cada solución candidata (individuo) durante la evolución, haremos que cada uno de ellos compita en una carrera de entrenamiento de 20 vueltas en un circuito de dificultad media sin rivales. Como se ha dicho, hemos omitido la presencia de oponentes para evitar incluir fuentes de incertidumbre adicionales al proceso de optimización. Para obtener controladores de comportamiento general, la pista seleccionada para este proceso tendrá una combinación de zonas de muchas curvas (difíciles) y partes rectas (sencillas).

Una vez que dicha carrera de prueba ha concluido, se tomarán los valores de salida: $Damage$, $MinSpeed$, $MaxSpeed$ y $AVGSpeed$ y se calculará el valor correspondiente del fitness aplicando la fórmula deseada.

IV. Experimentos y Resultados

Después de analizar la mayoría de las pistas disponibles, hemos seleccionado para estos experimentos el circuito **Alpine 2**. Éste es bastante complejo, con múltiples tipos de curvas, aunque también con ciertas partes rectas (Ver Figura 3).

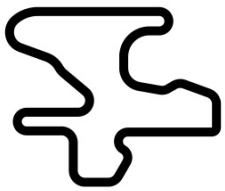

Figura 3. Circuito Alpine 2: Pista lenta de montaña. Longitud: 3773,57m, Anchura: 10m

Como coche para nuestro controlador, hemos utilizado *car1-tbr1*, ya que según experimentos anteriores [17], es la mejor opción debido a su rendimiento moderado y buen control, lo que lo hará adecuado para la gran mayoría de las pistas.

El controlador genético difuso (GFC) ha sido evaluado considerando las dos funciones de fitness propuestas: GFC-MMS (Ecuación 5) y GFC-AVS (Ecuación 6), comparando su rendimiento en carrera. Hemos ejecutado el algoritmo con una población de 50 individuos. El resto de parámetros son: Generaciones=50, Tasa de cruce=0,85, Tasa de mutación=0,1, y 10 ejecuciones diferentes por cada configuración.

Los dos procesos de optimización genética (uno por cada función de fitness) se han llevado a cabo de forma independiente. Sin embargo, a diferencia de trabajos anteriores en los que se seleccionaba el mejor en base a su valor para dicha función, en este estudio hemos buscado implementar una mejor metodología, la cual esperamos que produzca un controlador más competitivo.

Para ello, en ambas implementaciones, una vez finalizado el proceso evolutivo, los cuatro mejores individuos han competido juntos en 5 carreras (de 5 vueltas cada una) en la pista **Alpine 2** (utilizada durante la optimización) y 5 carreras (de 5 vueltas) en la pista **E-Track 5** (nueva para ellos).

Además, con el fin de mejorar la selección de los mejores, otros dos controladores son elegidos al azar para participar en la carrera, de entre una selección de bots de TORCS. Hemos implementado una competición basada puntos, que se basa en el esquema de la Fórmula 1, por lo que las puntuaciones obtenidas dependen de la posición del coche: 1 - 25 puntos, 2 - 18, 3 - 15, 4 - 12, 5 - 10, 6 - 8, 7 - 6, 8 - 4, 9 - 2, 10 - 1. Además, para incluir los términos de fitness en esta

selección, hemos definido una *puntuación extra*, de forma que el controlador que consiga el mejor tiempo o el daño mínimo en cada vuelta recibe 5 puntos extra.

Los resultados de estas ejecuciones se muestran en la Tabla I. Las "Mejores vueltasz "Daño mínimo"son las puntuaciones obtenidas por cada controlador en cada carrera al conseguir el mejor tiempo de vuelta y/o el daño mínimo de todos los contendientes. El símbolo '-' significa que el bot de TORCS no participa en la carrera.

Según la tabla, el primer individuo de $GFC-MMS$ y el segundo de $GFC-AVS$ han ganado el mismo número de carreras, pero $GFC-AVS_2$ ha logrado mejores resultados en las carreras que no ganó. Hay que destacar que los resultados de los bots de TORCS no se han tenido en cuenta ya que sólo sirven para diversificar la selección y no participan en todas las carreras. Esta selección permite, por tanto, elegir el mejor individuo en varias carreras y de forma más robusta y estable, y así se evita la selección clásica por torneo donde se elige el ganador de una sola confrontación.

También se puede señalar que los controladores difusos genéticos obtienen el daño mínimo, incluso cuando no ganan la carrera. Este hecho justifica fuertemente el uso de daño en las funciones de fitness, que es un factor clave a tener en cuenta en las carreras reales (para terminarlas).

Los controladores obtenidos con la primera función de fitness también han sumado los puntos de las mejores vueltas en cinco de las diez carreras. Hay que tener en cuenta que la mejor vuelta es el resultado de un daño mínimo y una alta velocidad ($MaxSpeed$), ambos optimizados por dicha función de evaluación. En la misma línea, el segundo fitness intenta maximizar la velocidad media, pero no necesariamente $MaxSpeed$.

Para probar la efectividad del método, hemos elegido los dos mejores controladores difusos genéticos, $GFC-MMS_1$ y $GFC-AVS_2$, uno por función de fitness, obtenidos en las pruebas anteriores. Éstos han sido evaluados en un conjunto de carreras contra algunos oponentes seleccionados. Además, los dos mejores controladores evolutivos de nuestro anterior trabajo [18], $EVO1$ y $EVO2$, también han sido incluidos en la 'competición'.

Esta evaluación es un tipo de mini-campeonato, que también considera las puntuaciones de Fórmula 1, pero en este caso no hay puntos extra. Se realizaron 10 carreras, cada una de 20 vueltas, y con un total de 10 participantes por carrera: los dos mejores $GFC-MMS_1$ y $GFC-AVG_2$, $EVO1$, $EVO2$, y también 6 bots competitivos de TORCS escogidos de la literatura. Las primeras 5 carreras se realizaron en la pista **Alpine 2** (usada en el entrenamiento); y las otras 5 carreras tuvieron lugar en la pista **E-Track 5** (no utilizada por los nuevos controladores, pero usada en la evolución de los anteriores). La parrilla de salida (posiciones iniciales de los coches) en estas carreras se estableció al azar.

Los resultados se muestran en la Tabla II. Esta tabla muestra cómo uno de los controladores difusos que evolucionan usando los nuevos mecanismos de selección y evaluación, $GFC-MMS_2$, da los mejores resultados, obteniendo muy buenas clasificaciones en las carreras. El controlador Inferno1 también obtuvo muy buenos resultados, alcanzando 3 veces la mejor puntuación por vuelta, pero esos resultados están muy influenciados debido a que usó el coche más rápido. También podemos ver que nuestros dos controladores difusos genéticos sólo han ganado una carrera $GFC-MMS_1$ y dos carreras $GFC-AVS_2$, mientras que los bots $berniw2$ y $inferno1$ han ganado tres carreras cada uno.

Sin embargo, como se puede ver en la tabla, aunque no son capaces de ganar siempre, sí que han terminado en posiciones altas de la clasificación, lo que les ha ayudado a obtener puntos y ganar el campeonato finalmente. De modo que la combinación entre la minimización de daño y la búsqueda de la velocidad media ha sido el mejor enfoque.

V. Conclusiones y Trabajo Futuro

En este trabajo hemos presentado métodos para mejorar la generación efectiva de controladores para el simulador de coches TORCS mediante el uso de algoritmos evolutivos. Se parte de un controlador que hace uso de lógica difusa para calcular la velocidad objetivo (sub-controlador 1) y la dirección del coche (sub-controlador 2).

De modo que se han propuesto dos funciones de fitness más especializadas que las que usamos en trabajos anteriores [17], [18], enfocadas en la minimización de los daños obtenidos durante la carrera, así como la maximización de la velocidad máxima (en tramos sencillos), mínima (en tramos complejos) y media (en toda la pista).

Junto a esto, se ha postulado la selección heurística de pistas para el entrenamiento/evolución de los controladores, que contenga diversidad en sus tramos para obtener conductores autónomos más adaptables.

Además se ha propuesto un mecanismo de selección de los mejores controladores, una vez concluida la evolución, basándose en la realización de carreras y eligiendo al ganador en ellas, en lugar de ceñirnos al valor obtenido en la función de fitness. De este modo, se elegirá a los mejores de forma más justa y robusta.

Los experimentos realizados nos dejan clara la efectividad de las propuestas, puesto que se han enfrentado los controladores obtenidos contra rivales de nivel medio/alto, incluyendo los de trabajos anteriores de los autores, en un campeonato en el que han resultado ganadores los nuevos.

Como trabajo futuro, nos centraremos primeramente en la evaluación de los distintos enfoques aplicados, para identificar aquel que tiene mayor influencia en los resultados. Respecto al algoritmo genético empleado, se podría intentar implementar un enfoque multiobjetivo, que considere los distintos factores de la función de fitness como independientes.

Agradecimientos

Este trabajo ha sido financiado en parte por el Ministerio Español de Economía y Competitividad con los proyectos TIN2014-56494-C4-3-P (UGR-EPHEMECH), TIN2017-85727-C4-2-P (UGR-DeepBio) y TEC2015-68752 (también financiado por FEDER).

Tabla I
PUNTUACIONES OBTENIDAS EN LA SELECCIÓN BASADA EN CARRERAS PARA LAS DOS IMPLEMENTACIONES EN DOS PISTAS DIFERENTES

Controlador	5 carreras en la pista Alpine 2 (5 vueltas cada una)							5 carreras en la pista E-Track 5 (5 vueltas cada una)							
	C1	C2	C3	C4	C5	Mejores vueltas	Daño mínimo	C1	C2	C3	C4	C5	Mejores vueltas	Daño mínimo	Total
$GFC-MMS_1$	**25**	18	8	15	12	15	10	12	**25**	18	**25**	18	10	10	**221**
$GFC-MMS_2$	12	**25**	15	12	15	0	5	8	15	15	4	10	0	0	136
$GFC-MMS_3$	6	6	10	10	8	0	0	15	10	10	18	6	0	5	104
$GFC-MMS_4$	2	8	4	4	6	0	0	10	1	2	1	2	0	0	40
$GFC-AVS_1$	1	4	6	2	4	0	0	4	2	12	10	4	0	0	49
$GFC-AVS_2$	15	10	18	**25**	18	5	10	**25**	18	**25**	15	15	5	10	206
$GFC-AVS_3$	10	2	1	6	2	0	0	2	6	4	6	1	0	0	41
$GFC-AVS_4$	4	1	2	1	1	0	0	1	4	8	2	8	0	0	32
$bt1$	-	-	-	8	-	0	0	-	8	6	8	-	0	0	-
$inferno1$	18	-	12	-	-	0	0	18	12	-	-	-	0	0	-
$berniw2$	8	15	-	18	-	0	0	-	-	-	12	12	5	0	-
$berniw3$	-	12	-	-	**25**	5	0	4	-	-	-	**25**	5	0	-
$damned1$	-	-	**25**	-	10	0	0	-	-	8	-	-	0	0	-

Tabla II
RESULTADOS DEL MINI-CAMPEONATO CON 10 CONTROLADORES Y 10 CARRERAS EN DOS PISTAS DISTINTAS. `tita`, `berniw` E `inferno` SON CONTROLADORES INCLUIDOS CON EL SIMULADOR TORCS [19]

Controlador	Carreras en **Alpine 2** (20 vueltas cada una)						Carreras en **E-Track 5** (20 vueltas cada una)						Puntuación Total
	C1	C2	C3	C4	C5	Puntuación por pista	C6	C7	C8	C9	C10	Puntuación por pista	
$GFC-MMS_1$	25	10	18	12	10	75	18	12	15	15	12	72	147
$GFC-AVS_2$	15	18	25	15	15	**88**	25	18	18	12	18	91	**179**
$tita1$	4	2	1	2	2	11	4	2	1	4	6	17	28
$tita2$	2	1	2	1	1	7	1	1	2	1	2	9	16
$inferno1$	12	15	12	18	18	75	12	15	25	25	15	**92**	167
$inferno2$	10	12	4	10	25	61	10	10	4	2	8	34	95
$berniw1$	18	25	15	8	6	72	8	8	6	10	10	42	114
$berniw2$	8	8	10	25	12	63	15	25	10	8	25	83	146
$EVO1$	6	6	8	4	8	32	2	6	12	8	4	32	64
$EVO2$	1	4	6	6	4	21	6	4	8	6	2	26	47

REFERENCIAS

[1] T. Bäck, *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.

[2] S. Godil, M. Shamim, S. Enam, and U. Qidwai, "Fuzzy logic: A 'simple' solution for complexities in neurosciences?" *Surg Neurol Int.*, pp. 2 – 24, 2011. [Online]. Available: https://doi.org/10.4103/2152-7806.77177

[3] D. E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley, 1989.

[4] S. Guadarrama and R. Vazquez, "Tuning a fuzzy racing car by coevolution," in *Genetic and Evolving Systems, GEFS 2008*, March 2008. [Online]. Available: https://doi.org/10.1109/GEFS.2008.4484568

[5] I. Iancu, *A Mamdani Type Fuzzy Logic Controller*. InTech, 2012, pp. 325–352.

[6] T. S. Kim, J. C. Na, and K. J. Kim, "Optimization of an autonomous car controller using a self-adaptive evolutionary strategy," *International Journal of Advanced Robotic Systems*, vol. 9, no. 3, p. 73, 2012. [Online]. Available: https://doi.org/10.5772/50848

[7] S. Kolski, D. Ferguson, C. Stacniss, and R. Siegwart, "Autonomous driving in dynamic environments," in *In Proceedings of the Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[8] J. Koutník, J. Schmidhuber, and F. Gomez, "Evolving deep unsupervised convolutional networks for vision-based reinforcement learning," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14, 2014, pp. 541–548.

[9] D. P. Liébana, G. Recio, Y. Sáez, and P. Isasi, "Evolving a fuzzy controller for a car racing competition," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games, CIG 2009, Milano, Italy, 7-10 September, 2009*, 2009, pp. 263–270. [Online]. Available: https://doi.org/10.1109/CIG.2009.5286467

[10] D. Loiacono, P.-L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. . Butz, T. D. Lonneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadflieg, "The 2009 simulated car racing championship," *IEEE Trans. Comput. Intell. AI Games*, vol. 2(2), pp. 131–147, 2010.

[11] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez, "The wcci 2008 simulated car racing competition," in *2008 IEEE Symposium On Computational Intelligence and Games*, Dec 2008, pp. 119–126.

[12] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Simulated car racing championship: Competition software manual," *CoRR*, vol. abs/1304.1672, 2013. [Online]. Available: http://arxiv.org/abs/1304.1672

[13] A. Neubauer, "A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1997. [Online]. Available: https://doi.org/10.1109/ICEC.1997.592275

[14] E. Onieva, J. Alonso, J. Pérez, and V. Milanés, "Autonomous car fuzzy control modeled by iterative genetic algorithms," in *Fuzzy Systems*, 2009, pp. 1615 – 1620.

[15] E. Onieva, D. Pelta, J. Godoy, V. Milanés, and J. Rastelli, "An evolutionary tuned driving system for virtual car racing games: The autopia driver," *International Journal of Intelligent Systems*, vol. 27, pp. 217–241, 2012.

[16] E. Onieva, D. A. Pelta, J. Alonso, V. Milanés, and J. Pérez, "A modular parametric architecture for the torcs racing engine," in *Proceedings of the 5th IEEE Symposium on Computational Intelligence and Games (CIG'09)*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 256–262.

[17] M. Salem, A. M. Mora, J. J. Merelo, and P. García-Sánchez, "Driving in TORCS using modular fuzzy controllers," in *Applications of Evolutionary Computation. EvoApplications 2017, Lecture Notes in Computer Science, vol 10199*, S. K. Squillero G., Ed. Springer, Cham, 2017, pp. 361–376.

[18] ——, "Evolving a TORCS modular fuzzy driver using genetic algorithms," in *Applications of Evolutionary Computation. EvoApplications 2018, LNCS*, K. S. et al., Ed. Springer, 2018, p. To appear.

[19] Sourceforge, "Web torcs," Web, Nov. 2016, http://torcs.sourceforge.net/.

[20] H. D. Thang and J. M. Garibaldi, "A novel fuzzy inferencing methodology for simulated car racing," in *IEEE International Conference on Fuzzy Systems, Hong Kong, China, 1-6 June, 2008, Proceedings*. IEEE, 2008, pp. 1907–1914. [Online]. Available: https://doi.org/10.1109/FUZZY.2008.4630630

[21] S. G. Varun Kumar and R. Panneerselvam, "A study of crossover operators for genetic algorithms to solve VRP and its variants and new sinusoidal motion crossover operator," *International Journal of Computational Intelligence Research*, vol. 13 (7), pp. 1717–1733, 2017.