

Ganando la Carrera a la Incertidumbre: Bots Evolutivos Mejorados para un Simulador de Carreras de Coches

A.M. Mora¹, M. Salem², and J.J. Merelo³

¹ Depto. de Teoría de la Señal, Telemática y Comunicaciones
ETSIIIT-CITIC, Universidad de Granada, España.

amorag@ugr.es

² Dept. of Computer Sciences
University of Mascara, Argelia
salem@univ-mascara.dz

³ Depto. de Arquitectura y Tecnología de Computadores
ETSIIIT-CITIC, Universidad de Granada, España.

jmerelo@ugr.es

Abstract One of the biggest problems in the design through optimization of bots for driving cars in racing simulators, is the so-called 'noise' inherent in the process. That is, in addition to the fact that the fitness function itself is a heuristic based on what we believe (or an expert believes) that will be most important to get a competitive controller to win races, that is, a surrogate/substitute model; the fitness calculation itself has associated uncertainty due to the stochastic components involved in its computation, such as the behavior of the rivals or the race conditions (track, weather), which are not deterministic. In previous work we defined an evolutionary controller based on fuzzy logic for the TORCS simulator. It worked with two sub-controllers, one dedicated to decide on the steering wheel rotation and the other to determine the target speed at each simulation instant. Both were optimized by means of a Genetic Algorithm (GA) based on a fitness calculation focused on maximizing the average speed during the race and minimizing the damage to the vehicle. The mentioned noise required to maintain diversity in the search (in the GA population), so we added the Blend Crossover operator (BLX- α), which also allows to exploit the current results while exploring for new solutions. Along with this in this work we try to manage the uncertainty in the selection of the best individuals of the GA by applying a novel race-based parent selection policy, called pole-position based selection. That is, individuals are grouped and compete against each other in several races, so that only the top-ranked individuals will remain in the population to reproduce. We have conducted several experiments, testing the performance of this new method and comparing the optimized controllers with others in the state of the art, including one of the best from an edition of the prestigious international Simulated Car Racing Championship, which we have clearly beaten.

Keywords: Simulación de Carreras de Coches, TORCS, Controladores basados en Lógica Difusa, Conductores autónomos, Algoritmos Genéticos, Optimización, Cruce BLX- α , Selección basada en Carreras, Incertidumbre, Ruido

1. Introducción y descripción del problema

Los juegos son, en muchos casos, entornos cerrados y controlados, partes o mundos simulados completos que permiten probar técnicas que luego eventualmente se podrán aplicar en el mundo real, probablemente combinado con diferentes tecnologías para abordar su complejidad y variabilidad. Por ejemplo, la simulación de carreras de coches incluye muchos de los factores que están presentes en conducción autónoma: las pistas son muy diferentes y no se conocen por anticipado, hay otros vehículos presentes en la pista, las condiciones cambian según el clima y el vehículo se deteriora si sufre daños.

El coche simulado (o más bien, su conductor) puede ser ‘consciente’ de todo esto a través de un conjunto limitado de sensores, y deberá que tomar una decisión sobre la velocidad y dirección óptimas con varios objetivos diferentes [10], incluyendo, según el contexto, la posibilidad de vencer a un conjunto de oponentes en una carrera virtual.

Dado que probar diferentes metodologías de conducción autónoma en el mundo real está normalmente reservado a unas cuantas empresas en el mundo importantes, dichas metodologías y los algoritmos asociados se prueban habitualmente en entornos simulados. Estos entornos pueden ofrecer, además, la posibilidad de comparar nuestro sistema autónomo frente a otros. En este trabajo se usará uno de estos entornos, en concreto The Open Racing Car Simulator (TORCS) [25], un simulador de carreras realista que ofrece un gran banco de pruebas para la implementación y evaluación de conductores autónomos. TORCS se ha utilizado varias veces en competiciones de Inteligencia Artificial (IA), en las que el objetivo es crear el mejor conductor autónomo para competir en carreras [14,13,12]. Además de poder probar nuestro coche frente a otros que se han publicado, se puede utilizar como un entorno independiente para optimizar la conducción en carreras en solitario.

Por otra parte, los Algoritmos Evolutivos (EAs) [1] se han aplicado frecuentemente como un método de optimización de propósito general en esta área, generalmente combinado con motores de comportamiento que gobiernan diferentes partes del coche [19,9,18]. Dichos motores de conducción han incluido últimamente controladores difusos [6,17,11], que son aquellos que aplican Lógica Difusa [4], una técnica bastante adecuada para definir este tipo de agentes autónomos, ya que están en parte inspirados en el razonamiento humano al conducir. Un controlador difuso funciona con variables lingüísticas que, por ejemplo, permitirán girar *ligeramente* hacia la derecha cuando la siguiente curva está *cerca*, pero estos controladores deben diseñarse para mapear adecuadamente las entradas con las salidas deseadas en situaciones particulares.

Desde el punto de vista de la optimización, uno de los principales problemas es que el entorno siempre va a cambiar. Si esto está reflejado correctamente en el simulador utilizado, significará que la puntuación (y por lo tanto, la clasificación, si ese es el objetivo final) siempre cambiará, haciendo una selección probabilística del controlador *mejor o ganador*. Esta *incertidumbre* es un desafío a dos niveles: en primer lugar, controladores no óptimos (no ganadores) podrían seleccionarse por casualidad, ya que se les habría asignado una puntuación alta favorecida por dicha incertidumbre; en segundo lugar, una vez seleccionados, esto afectaría a que el algoritmo explore zonas alrededor de dichos controladores no óptimos y dejaría otros (que podrían ser prometedores) sin explorar.

Por estas dos razones, en este trabajo abordaremos dos desafíos: reducir la incertidumbre en la selección de los “ mejores” y mantener una alta diversidad en la población de soluciones (controladores) para no explotar solo las áreas alrededor de esos individuos, cuya puntuación podría haber sido favorecida en parte por el azar durante el proceso de evolución.

En trabajos previos, los autores presentaron un enfoque que combinaba dos sub-controladores difusos especializados, diseñados por un experto, que podían decidir el mejor ángulo de giro del volante y la velocidad deseada para el coche en cada punto (o tic de simulación) durante una carrera [22]. Este controlador fue posteriormente mejorado [23] optimizando los parámetros de sus funciones de pertenencia mediante un algoritmo genético [5]. Finalmente, los autores mejoraron el controlador en el último artículo [21] mediante la definición de nuevas funciones de fitness. La selección del mejor controlador al final de la evolución se basó en un conjunto de carreras entre las mejores 4 soluciones, consiguiendo un mejor piloto que en estudios anteriores.

Esto demostró que los algoritmos evolutivos son capaces de obtener los mejores parámetros difusos para los subcontroladores, pero al mismo tiempo reveló varios desafíos. En general, los algoritmos evolutivos optimizan la función de fitness definida, de modo que los controladores difusos evolucionados (en adelante, FCs) serán eventualmente tan buenos como lo permita la función de fitness. Pero en este caso particular no podemos utilizar como función de aptitud la posición obtenida por el FC en todas las carreras posibles en todas las pistas posibles con todos los posibles oponentes, por lo que tenemos que conformarnos con un *sustituto* (subrogado) del fitness en un entorno muy limitado. Primero optamos por eliminar oponentes y hacer evaluaciones en carreras en solitario. Después elegimos una pista en particular que combinaba segmentos rectos, así como algunas curvas, y finalmente tuvimos que decidir qué factores relacionados con la velocidad, el daño y el tiempo por vuelta iban a ser incluidos efectivamente en la función de fitness final.

De este modo, las dos nuevas técnicas aplicadas en este trabajo: *Selección basada en pole-position* (basada en carreras) y cruce $BLX - \alpha$, intentan mejorar los resultados anteriores confiando menos en la subrogación de la función de fitness para seleccionar las mejores soluciones (controladores). El mecanismo de selección utilizará una función de aptitud sin parámetros (parameter-less) para seleccionar algunos individuos que competirán entre sí, además las carreras *reducirán* la aleatoriedad en la aptitud al poner a los candidatos en un entorno más real, es decir, enfrentar coches en carreras uno contra uno ofrecerá un resultado mucho menos variable que simplemente comparar un número que represente su aptitud. Pero incluso en este caso, la incertidumbre estará presente al calcular la aptitud y debemos evitar una explotación excesiva de los resultados. El cruce de $BLX - \alpha$ que hemos introducido se ocupará de este aspecto.

Con estos métodos, nuestro objetivo es obtener controladores más confiables y competitivos, los cuales serán probados contra oponentes muy competitivos, incluyendo un controlador avanzado que obtuvo muy buenos resultados en varias ediciones de la competición internacional en TORCS.

2. El simulador y los controladores

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

2.1. The Open Racing Car Simulator

TORCS [25] 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 [12], 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 (Ver Figura 1).

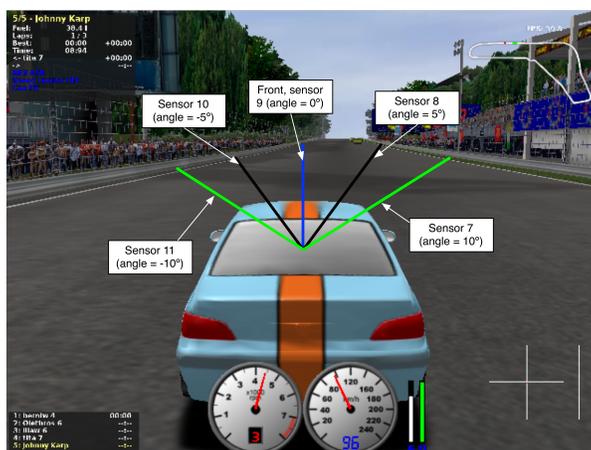


Figura 1. Captura de TORCS en la que se muestran varios de los sensores que considera el coche.

Estos valores serán considerados por los conductores autónomos o *controladores*, para gestionar el coche utilizando los llamados *actuadores* [12]: giro del volante, acelerador, freno y cambios de marcha.

2.2. Subcontroladores Difusos

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

Sin embargo, la velocidad objetivo y el ángulo de giro de la dirección se calculan mediante dos sistemas modulares y especializados [22]. 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 [8] con tres funciones de pertenencia (MF) trapezoidales para cada variable de entrada. En [23] 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 [22].

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 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 [22].

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 lingüísticos, 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 'funcionamiento'. De modo que se aplicó un AG para optimizar dichos

parámetros y determinar la partición difusa de la variable lingüística [24]. Las variables lingüísticas de entrada en nuestro problema serán *Front*, *M5* y *M10*.

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

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

Con:

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

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).

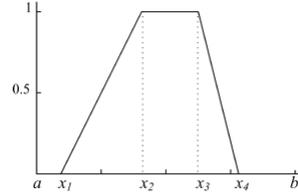


Figura 2. Función de pertenencia trapezoidal

En nuestra propuesta, una partición difusa con n MF trapezoidales se define mediante $2n$ variables ($a = x_1, x_2, \dots, x_{2n} = b$) (Ecuación 4). Con:

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

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

$$\mu_{Ai}(x) = \begin{cases} 0, & x \leq x_{2i-2} \\ \frac{x-x_{2i-2}}{x_{2i-1}-x_{2i-2}}, & x_{2i-2} \leq x \leq x_{2i-1}, n = 2, \dots, i-1 \\ 1, & x_{2i-1} \leq x \leq x_{2i} \\ \frac{x_{2i+1}-x}{x_{2i+1}-x_{2i}}, & x_{2i} \leq x \leq x_{2i+1} \\ 0, & x > x_{2i+1} \end{cases} \quad (4)$$

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

3. Optimización Mediante un 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.

De modo que cada individuo/cromosoma es un vector de 18 valores/parámetros, 6 por variable, como se muestra en la Figura 3.



Figura 3. Descripción de un cromosoma

La inicialización de los cromosomas (población inicial) se realiza asignando valores aleatorios en un rango de variación $[0, 100]$ [5], a fin de comenzar por valores válidos [22]. Dado que nuestro trabajo requiere precisión y el intervalo de variación de cada parámetro no es completamente conocido, hemos considerado codificación real [2] en el vector de variables a optimizar.

El proceso completo se puede ver en la Figura 3. Como se muestra, TORCS se usa en la fase de evaluación de cada individuo dentro del proceso evolutivo.

La evaluación de los individuos se ha realizado considerando la función de fitness que mejores resultados nos ha dado hasta este trabajo, usada en el artículo [21]. Ésta es:

$$f_{AVS} = \frac{AVG(Speed)}{Damage+1} \quad (5)$$

Es un enfoque sin parámetros (parameter-less) ya que no hay pesos en los términos [7], que se centra en los objetivos reales de un conductor en una carrera, en lugar de en el objetivo final de ganarla, a fin de conseguir controladores más humanos ('human-like'). La función depende únicamente de dos variables, por lo que se intenta conseguir conductores capaces de alcanzar la máxima velocidad media en la pista ($AVG(Speed)$), al tiempo que se evita que el coche sufra daños ($Damage$), ya que demasiado daño hará que el coche tenga que abandonar la carrera.

De modo que la aptitud de cada solución candidata se calcula 'inyectando' sus valores genéticos a los parámetros de las funciones de pertenencia 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, y los resultados (velocidad media y daño) se utilizan para calcular el valor de fitness. Como el objetivo del controlador del coche es ganar tantas carreras como sea posible, intentamos optimizar el caso más general realizando *carreras de entrenamiento* en solitario, que serán menos sensibles a la presencia de ruido/incertidumbre debido a la participación de otros controladores [15]. La pista seleccionada para esta evaluación será una que combine curvas y partes rectas para obtener un 'comportamiento todoterreno'.

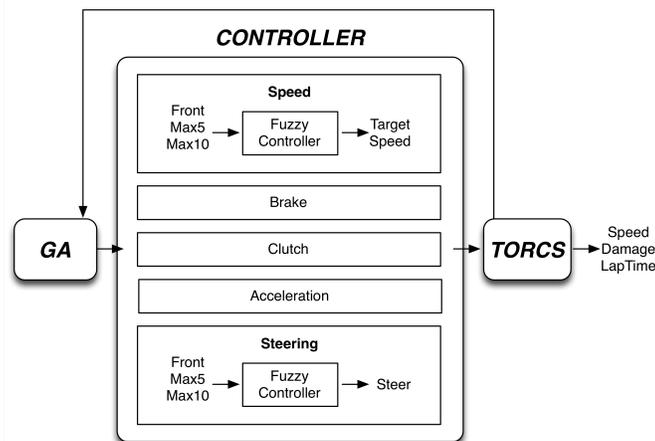


Figura 4. Diagrama de flujo del proceso de optimización de un controlador difuso en TORCS. Para evaluar a un individuo ponemos los valores de los parámetros de los dos subcontroladores en el cromosoma correspondiente, luego lanzamos una carrera en TORCS con esta configuración, obteniendo los valores resultantes de *Damage*, *TopSpeed* y *MeanLapTime*. El valor de fitness del individuo se calcula utilizando estos valores.

El operador de mutación ha permanecido como en trabajos anteriores, es decir, **mutación no uniforme** [16].

Una nueva **política de selección basada en pole-position** ha sido implementada (o selección basada en carreras), con el objetivo de lograr que individuos/controladores mejores o más confiables sean padres de la siguiente población. Para ello, todos los individuos se organizan en grupos de 10, luego se simulan diferentes carreras de varias vueltas utilizando a cada individuo como controlador (con el mismo coche) en una pista de TORCS. Después de cada carrera, los participantes obtienen diferentes puntuaciones en función de su posición en la clasificación final. Los 5 mejores controladores tras la suma de puntuaciones de todas las carreras se seleccionan como padres para la siguiente descendencia.

De esta forma, se seleccionarán los mejores individuos para reproducirse con mayor probabilidad. No es posible asegurar que sean absolutamente los mejores, debido a la incertidumbre presente en este tipo de entornos, es decir, juegos en los que competimos contra oponentes no deterministas [15]. Sin embargo, creemos firmemente que esta política de selección será “menos sensible” a esa incertidumbre (o ruido) y, por lo tanto, será más justa y confiable que un enfoque basado puramente en los valores de fitness. Por tanto, pensamos que este mecanismo de selección propuesto sería beneficioso para conseguir un buen proceso de optimización.

Además, se ha aplicado el **operador de cruce BLX- α** [3], en lugar del anterior cruce en dos puntos. El *Blendcrossoveroperator* elige aleatoriamente un número en el intervalo $[x_i - \alpha(y_i - x_i) .. y_i + \alpha(y_i - x_i)]$, donde x_i e y_i son los i^{th} valores de paráme-

tros de las soluciones padres x, y , y además $x_i < y_i$. Vea la Figura 5 para comprender el proceso.

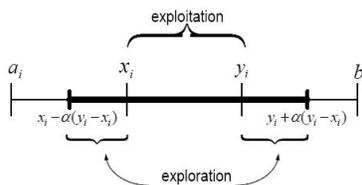


Figura 5. Operador de cruce ($BLX - \alpha$)

De forma que este operador se basa en la generación aleatoria de genes de la vecindad asociada a los genes en los padres. Se generan tres descendientes, con diferencias entre sí y también entre ellos y sus padres, lo que lleva a un factor de exploración superior en la generación de la descendencia. Este operador es normalmente usado en algoritmos genéticos con codificación real, en los que ha demostrado alcanzar un buen equilibrio entre exploración y explotación [3].

En este operador, el parámetro α controla la relación entre exploración y explotación, de modo que, para garantizar un equilibrio entre estos factores, tomará un valor $\alpha = 0,5$.

En los AGs, el proceso de búsqueda necesita una alta tasa de exploración en las primeras generaciones para evaluar múltiples partes del espacio de búsqueda (alta diversidad), pero en las últimas generaciones se prefiere una alta explotación para asegurar que se llega a una solución óptima.

Hemos considerado dos enfoques diferentes en los experimentos, uno en el que se usa un valor constante para α y otro en el que el valor es variable, de forma que éste decrece a medida que pasan las generaciones (obteniendo paulatinamente más explotación y menos exploración). De forma que el valor se calcula como:

$$\alpha = 1 - \frac{g}{g_{max}} \quad (6)$$

Donde g es la generación actual y g_{max} es el máximo número de generaciones.

4. Experimentos y resultados

Basándonos en los resultados de nuestro artículo anterior [21], la elección de una pista adecuado para el entrenamiento es un factor importante para obtener bots competitivos. De manera que hemos seleccionado el circuito *Alpine 2* para los experimentos, ya que combina múltiples giros con partes rectas (Ver Figura 6).

Además, al igual que en nuestros estudios previos, hemos considerado el coche *car1-tbr1* en nuestros controladores, dado que tiene un rendimiento moderado que per-

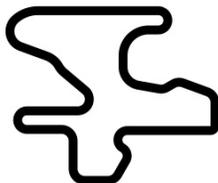


Figura 6. Alpine 2 Track: Circuito de montaña lento. Longitud: 3773.57m, Anchura: 10m

mitirá a nuestro controlador adaptarse a la conducción en casi todas las condiciones de pista.

Los controladores genéticos difusos (GFC) se han evaluado con la función de fitness mencionada en la sección anterior: f_{AVS} (Ecuación 5). Se ha considerado un tamaño de población de 60 individuos. El resto de parámetros han sido: Generaciones=50, Prob. Cruce=0,85, Prob. Mutación=0,09, y 10 ejecuciones diferentes por configuración.

La selección basada en pole-position se ha aplicado sobre la pista indicada, con 5 carreras de 20 vueltas cada una y una parrilla inicial (posición de partida) aleatoria. La función de puntuación se ha definido basándonos en el esquema habitual en la Fórmula 1, de modo que las puntuaciones obtenidas dependen del puesto del coche en el ranking final: 1 - 25 puntos, 2 - 18, 3 - 15, 4 - 12, 5 - 10, 6 - 8, 7 - 6, 8 - 4, 9 - 2, 10 - 1.

Al final de la evolución (en la última generación) un proceso de selección basado en carreras se ha aplicado de nuevo para elegir al mejor individuo de la ejecución. De modo que los 10 mejores individuos (en base a su fitness) de la población final han competido en 5 carreras de 5 vueltas en la misma pista. Se han calculado las mismas puntuaciones y el controlador con mayor puntuación ha sido elegido como el mejor.

El proceso evolutivo se ha aplicado en grupos separados de ejecuciones y se han obtenido los siguientes controladores:

- *GFC*: Controlador de nuestro trabajo anterior [21], en el que se usó el fitness f_{AVS} (Equation 5).
- *GFC - RS*: Un controlador obtenido aplicando el cruce de dos puntos, selección basada en pole-position cada 5 generaciones y el mismo fitness en las demás generaciones.
- *GFC - FA*: Un controlador obtenido aplicando $BLX - \alpha$ crossover con un valor constante de $\alpha = 0,5$ y selección basada en pole-position cada 5 generaciones y el fitness f_{AVS} en las demás.
- *GFC - VA*: Un controlador obtenido aplicando $BLX - \alpha$ con un valor variable de α usando la expresión 6 y selección basada en pole-position carreras cada 5 generaciones y el fitness f_{AVS} en las demás.

Una vez que las 10 ejecuciones han terminado, los mejores 10 controladores obtenidos compiten de nuevo en un conjunto similar de carreras al que se hace en la última generación, a fin de elegir al mejor controlador de cada esquema, es decir, el mejor *GFC - RS*, *GFC - FA* y *GFC - VA*.

Los mejores GFC finales (uno por esquema) se han evaluado en grupos de varias carreras, en una especie de *mini campeonato* de Fórmula 1, que consta de 10 carreras, cada una de 20 vueltas y con un total de 10 participantes por carrera: los 4 GFCs y 6 bots estándar de TORCS. Hemos elegido dos controladores de tipo *tita* (un conductor conservador), dos *berniw* (conocido por su agresiva política de adelantamiento) y dos *infern*o (el más rápido). Las primeras 5 carreras se llevaron a cabo en la pista *Alpine 2* (utilizada durante el entrenamiento/optimización); y las otras 5 carreras se realizaron en la pista *E-Track 5* (no entrenada para los nuevos controladores). Finalmente, para hacerlo más justo, hemos definido una *puntuación adicional*, por lo que el controlador que consigue la vuelta más rápida o el daño mínimo en cada carrera recibe 5 puntos extra. La parrilla de salida se estableció nuevamente al azar.

Los resultados de esta comparativa se muestran en la Tabla 1 y se resumen de manera gráfica en la Figura 7.

Tabla 1. Resultados del mini-campeonato con 10 controladores y 10 carreras en dos pistas diferentes. *tita*, *berniw* e *infern*o son controladores de ejemplo incluidos en TORCS [25]

Controlador	Carreras en <i>Alpine 2</i> (20 vueltas)					Puntuación Pista	Carreras en <i>E-Track 5</i> (20 vueltas)					Puntuación Pista	Puntuación TOTAL
	C1	C2	C3	C4	C5		C6	C7	C8	C9	C10		
<i>GFC</i>	6	8	8	15	15	52	12	15	10	10	15	62	134
<i>GFC - RS</i>	12	10	18	12	12	64	10	8	12	25	8	63	127
<i>GFC - FA</i>	18	12	15	10	10	65	15	12	25	15	25	92	157
<i>GFC - VA</i>	25	18	25	25	18	111	25	18	15	18	18	94	205
<i>tita1</i>	4	2	6	4	1	15	2	1	2	1	1	7	22
<i>tita2</i>	2	1	2	2	2	9	1	2	1	2	2	8	17
<i>infern</i> o1	8	4	1	1	6	20	4	4	6	8	6	28	48
<i>infern</i> o2	1	6	4	8	8	27	6	6	4	4	4	24	51
<i>berniw</i> 1	10	25	10	18	4	67	18	10	8	12	12	60	127
<i>berniw</i> 2	15	15	12	6	25	73	8	25	18	6	10	67	140

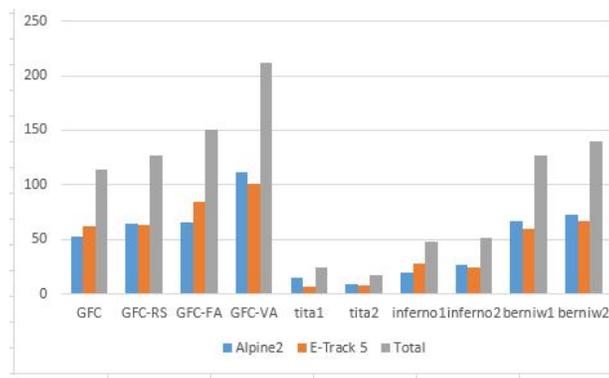


Figura 7. Puntuaciones obtenidas por los diferentes controladores difusos genéticos en dos pistas diferentes.

De la tabla y la figura se desprende claramente que el controlador $GFC - VA$ obtiene los mejores resultados. De hecho, el Dicho controlador ganó tres carreras en la pista Alpine 2 y se ha clasificado en segundo lugar en las otras dos pistas de carreras. En el circuito E-Track 5 ganó dos carreras, ha sido segundo dos veces y tercero en la última carrera.

El segundo controlador que usa $BLX - \alpha$ (valor constante de 0,5) quedó muy bien posicionado en todas las carreras. Ganó una carrera y se clasificó segundo en dos más y tercero otras. Las demás carreras las ganó el controlador *berniw*, siempre un duro rival. Podemos notar que los controladores basados en $BLX - \alpha$ ganaron tres de las cinco carreras en la pista Alpine 2 utilizada en la selección y se clasificaron al menos en el cuarto lugar. Se obtuvieron los mismos resultados o incluso mejores para la otra pista, la cual era desconocida para nuestros controladores.

Estos resultados confirman la eficacia y la solidez de la política de selección basada en pole-position utilizada para evaluar a los individuos, así como para seleccionar a los candidatos para el cruce. Aunque esta política se ha aplicado solo una vez cada 5 generaciones debido a que consume mucho tiempo, claramente ha afectado positivamente al rendimiento de los controladores, como se puede observar en la gran diferencia existente entre los resultados del controlador GFC frente a $GFC - RS$ por ejemplo.

A su vez, la política de selección propuesta, combinada con el operador $BLX - \alpha$, ha mejorado el rendimiento del controlador $GFC - FA$. La introducción de un parámetro variable α a lo largo de las generaciones en el bot $GFC - VA$ ha hecho posible controlar mejor la relación entre exploración/explotación durante el proceso evolutivo, permitiendo generar descendientes diferentes de sus padres y más eficientes que ellos.

Finalmente, para comprobar la bondad de nuestro mejor controlador, hemos realizado un experimento adicional. Hemos considerado un oponente del estado del arte, que participó en varias Competiciones de Carreras Simuladas en TORCS en ediciones pasadas. Fue propuesto por Pérez-Liébana, Sáez, Recio e Isasi [20] y luego refinado en el trabajo [11]. Lo hemos bautizado como PSRI en honor a los apellidos de sus autores.

Este controlador funciona principalmente mediante una máquina de estados finitos (FSM), definiendo los principales estados en los que puede encontrarse el conductor (por ejemplo, girando, adelantando a un rival, etc). Las transiciones en la FSM se rigen por un conjunto de reglas difusas, basadas en la información leída de diferentes sensores. También existe un módulo clasificador (árbol de decisión J48), capaz de analizar las entradas de algunos sensores con el fin de predecir partes de la pista, para anticipar las siguientes acciones a realizar. Las reglas difusas y también algunos parámetros del FSM se optimizaron mediante un algoritmo NSGA-II.

El controlador PSRI compitió en la edición 2009 del Simulated Car Racing Championship [13], donde ocupó el cuarto lugar considerando las puntuaciones obtenidas en tres competiciones diferentes (celebradas en las conferencias CEC, GECCO y CIG 2009). En promedio tuvo un gran rendimiento, alcanzando buenas puntuaciones y posiciones en varias carreras.

La Tabla 2 muestra una comparativa entre nuestros dos controladores genéticos difusos con $BLX - \alpha$: $GFC - FA$ y $GFC - VA$ y PSRI. Los resultados son los valores

medios de *Damage*, *MaxSpeed* y *Speed* para 10 carreras en la pistas *Alpine 2* y *E-Track 5*.

Tabla 2. Daño y Velocidades medios para 5 carreras en la pista **Alpine 2** y 5 carreras en la pista **E-Track 5**

Alpine 2				E-Track 5			
	<i>GFC - FA</i>	<i>GFC - VA</i>	<i>PSRI</i>		<i>GFC - FA</i>	<i>GFC - VA</i>	<i>PSRI</i>
Average Speed (km/h)	187,11	199,65	176,94	Average Speed (km/h)	161,11	170,23	160,89
Max Speed (km/h)	225,07	231,91	217,83	Max Speed (km/h)	262,88	270,17	266,54
Damage	126,82	117,55	131,99	Damage	18,12	14,67	28,09
Carreras ganadas	0	4	1	Carreras ganadas	1	3	1

Los resultados de los controladores *PSRI* y *GFC - FA* son similares, ganaron dos y una carrera respectivamente de las 10. Sus velocidades promedio son comparables pero en cuanto al daño, el controlador *GFC - FA* sufrió el mínimo al estar incluida la variable *Damage* en la evaluación de aptitud. Los resultados del controlador *GFC - VA* son aun más satisfactorios, ya que ganó 7 carreras y obtuvo el valor más bajo de daños 117, 55 y 14, 67 para ambos circuitos, así como la velocidad promedio más alta 199, 65 y 170, 23.

Observando estos resultados y los anteriores, podemos concluir que los controladores propuestos son más que competentes, debido a los nuevos mecanismos incluidos para hacer frente a la incertidumbre y realizar una búsqueda más adecuada del espacio de soluciones.

5. Conclusiones y trabajo futuro

En este artículo hemos intentado obtener, mediante la optimización evolutiva de sistemas difusos, controladores para coches de carreras competitivos mejorando el proceso de selección con el objetivo de eliminar (o al menos paliar) el efecto de la incertidumbre presente en este tipo de entornos. Para ellos se ha considerado evaluar a los controladores mediante carreras reales en lugar de seguir una evaluación basada en el cálculo de un fitness. Además, se ha controlado el equilibrio entre exploración y explotación durante el proceso evolutivo de optimización mediante el uso de un operador $BLX-\alpha$.

Cada controlador genético difuso está sujeto a incertidumbres en la pista, especialmente en caso de que haya rivales (con un comportamiento no determinista), por lo que para superar este problema y así diseñar un bot robusto y confiable, propusimos aplicar una *Política de selección basada en pole-position*, en la que la elección de los padres en el proceso evolutivo se lleva a cabo de acuerdo con los resultados de un conjunto de mini-campeonatos organizados entre los individuos de la población, de manera similar a un torneo de carreras de coches. Al mismo tiempo, y con el objetivo de intensificar el proceso de exploración en el espacio de búsqueda, usamos el operador de cruce $BLX - \alpha$ con valores decrecientes del parámetro α a lo largo de las generaciones.

La evaluación se realizó comparando el controlador propuesto con bots de la plataforma TORCS, arrojando muy buenos resultados. La otra evaluación de nuestro controlador fue una confrontación con un bot real (controlador *PSRI*), que participó en varias ediciones de campeonatos internacionales de carreras simuladas de coches. En este caso, el operador BLX y la nueva política de selección han tenido un gran impacto al ayudar a nuestro controlador a ganar tres cuartas partes de las carreras, obteniendo además los valores más bajos de daño, velocidad promedio y velocidad máxima.

Estos resultados nos permiten pensar que nuestro controlador podría haber alcanzado una muy buena clasificación en dicha competición, que lamentablemente ya no se celebra desde 2015. En cualquier caso, pensamos que los resultados de este estudio podrían aplicarse con éxito a otros simuladores de carreras de coches, como los que se utilizan en las competiciones de e-sports actuales, como iRace (<https://www.iracing.com/>).

Como líneas de trabajo futuro, este controlador se puede mejorar de varias formas: Podemos extender la política de selección a todas las generaciones superando el inconveniente del tiempo de cálculo mediante una implementación paralela. También podemos explorar otras funciones de fitness sin parámetros para evaluar a los individuos, incluyendo otros factores que afecten al rendimiento del coche. Otra perspectiva es utilizar múltiples pistas (en lugar de solo una) en el proceso de selección para obtener un controlador más general, capaz de lidiar con muchas situaciones diferentes.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los proyectos nacionales RTI2018-102002-A-I00 (Ministerio de Ciencia, Innovación y Universidades), TIN2017-85727-C4-2-P y PID2020-115570GB-C22 (Ministerio de Economía y Competitividad), así como los proyectos autonómicos B-TIC-402-UGR18 y P18-RT-4830 (FEDER y Junta de Andalucía).

Referencias

1. Bäck, T.: Evolutionary algorithms in theory and practice. Oxford University Press (1996)
2. Elsayed, S.M.M., Sarker, R., Essam, D.L.: A genetic algorithm for solving the CEC2013 competition problems on real-parameter optimization. In: IEEE Congress on Evolutionary Computation, CEC 2013. pp. 356–360. Cancun, Mexico (21-23 June 2013 2013)
3. García-Martínez, C., Lozano, M., Herrera, F., Molina, D., Sánchez, A.: Global and local real-coded genetic algorithms based on parent-centric crossover operators. European Journal of Operational Research 185 (3), 1088–1113 (2008)
4. Godil, S., Shamim, M., Enam, S., Qidwai, U.: Fuzzy logic: A ‘simple’ solution for complexities in neurosciences? Surg Neurol Int. pp. 2 – 24 (2011)
5. Goldberg, D.E.: Genetic Algorithms in search, optimization and machine learning. Addison Wesley (1989)
6. Guadarrama, S., Vazquez, R.: Tuning a fuzzy racing car by coevolution. In: Genetic and Evolving Systems, GEFS 2008 (March 2008)
7. Harik, G.R., Lobo, F.G.: A parameter-less genetic algorithm. In: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1. pp. 258–265. GEC-CO’99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)

8. Iancu, I.: A Mamdani Type Fuzzy Logic Controller, pp. 325–352. InTech (2012)
9. Kim, T.S., Na, J.C., Kim, K.J.: Optimization of an autonomous car controller using a self-adaptive evolutionary strategy. *International Journal of Advanced Robotic Systems* 9(3), 73 (2012)
10. Kolski, S., Ferguson, D., Stacniss, C., Siegwart, R.: 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)
11. Liébana, D.P., Recio, G., Sáez, Y., Isasi, P.: 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. pp. 263–270 (2009)
12. Loiacono, D., Cardamone, L., Lanzi, P.: Simulated car racing championship competition. software manual. TORCS news (2013)
13. Loiacono, D., Lanzi, P.L., Togelius, J., Onieva, E., Pelta, D.A., Butz, M., Lonnerker, T.D., Cardamone, L., Perez, D., Saez, Y., Preuss, M., Quadflieg, J.: The 2009 simulated car racing championship. *IEEE Trans. Comput. Intell. AI Games* 2(2), 131–147 (2010)
14. Loiacono, D., Togelius, J., Lanzi, P.L., Kinnaird-Heether, L., Lucas, S.M., Simmerman, M., Perez, D., Reynolds, R.G., Saez, Y.: The wcci 2008 simulated car racing competition. In: 2008 IEEE Symposium On Computational Intelligence and Games. pp. 119–126 (Dec 2008)
15. Merelo, J., Chelly, Z., Mora, A., Fernández-Ares, A., Esparcia-Alcázar, A.I., Cotta, C., de las Cuevas, P., Rico, N.: A statistical approach to dealing with noisy fitness in evolutionary algorithms. In: *Computational Intelligence*, pp. 79–95. Springer (2016)
16. Neubauer, A.: 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)
17. Onieva, E., Alonso, J., Pérez, J., Milanés, V.: Autonomous car fuzzy control modeled by iterative genetic algorithms. In: *Fuzzy Systems*. pp. 1615 – 1620 (2009)
18. Onieva, E., Pelta, D., Godoy, J., Milanés, V., Rastelli, J.: An evolutionary tuned driving system for virtual car racing games: The autopia driver. *International Journal of Intelligent Systems* 27, 217–241 (2012)
19. Onieva, E., Pelta, D.A., Alonso, J., Milanés, V., Pérez, J.: A modular parametric architecture for the torcs racing engine. In: Proceedings of the 5th IEEE Symposium on Computational Intelligence and Games (CIG'09). pp. 256–262. IEEE Press, Piscataway, NJ, USA (2009)
20. Perez, D., Saez, Y., Recio, G., Isasi, P.: Evolving a rule system controller for automatic driving in a car racing competition. In: 2008 IEEE Symposium On Computational Intelligence and Games. pp. 336–342 (Dec 2008)
21. Salem, M., Mora, A.M., Merelo, J.J.: The evolutionary race: Improving the process of evaluating car controllers in racing simulators. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG). pp. 1–8 (Aug 2018)
22. Salem, M., Mora, A.M., Merelo, J.J., García-Sánchez, P.: Driving in TORCS using modular fuzzy controllers. In: Squillero G., S.K. (ed.) *Applications of Evolutionary Computation. EvoApplications 2017*, LNCS, vol 10199. pp. 361–376. Springer, Cham (2017)
23. Salem, M., Mora, A.M., Merelo, J.J., García-Sánchez, P.: Evolving a TORCS modular fuzzy driver using genetic algorithms. In: Sim, K., Kaufmann, P. (eds.) *Applications of Evolutionary Computation*. pp. 342–357. Springer International Publishing, Cham (2018)
24. Thang, H.D., Garibaldi, J.M.: A novel fuzzy inferencing methodology for simulated car racing. In: IEEE International Conference on Fuzzy Systems, Hong Kong, China, 1-6 June, 2008, Proceedings. pp. 1907–1914. IEEE (2008)
25. Wyman, B., Espie, E., Guionneau, C., Dimitrakakis, C., Coulom, R., Sumner, A.: TORCS the open racing car simulator (2000), <http://www.torcs.org>