Reglas Difusas para el Control de un Vehículo en la 2009 Simulated Car Racing Competition: un Enfoque Ganador

Enrique Onieva [1] **David A. Pelta** [2] **Joshué Pérez** [1] **Javier Alonso** [1] **Vicente Milanés** [1]

[1] Departamento de Informática Industrial. Instituto de Automática Industrial (CSIC).
{onieva, jperez, jalonso, vmilanes}@iai.csic.es
[2] Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada.
dpelta@decsai.ugr.es

Resumen

El área de los videojuegos es un campo de aplicación excelente para las técnicas de Soft Computing. En particular, las carreras de coches simulados están ganando el interés de la comunidad científica por ser un excelente marco de prueba para aprendizaje, adaptación, razonamiento, evolución, etc. En esencia se trata de controlar el comportamiento de un coche en una serie de carreras donde existen otros coches.
Este trabajo presenta los resultados de nuestra contribución en la 2009 '*Simulated Car Racing Competition*', en la que presentamos un coche conducido por conjunto de reglas difusas.

Palabras Clave: simulación de vehículos, carreras de coches, arquitecturas de control.

1 INTRODUCCIÓN

La Inteligencia Artificial se ha aplicado a los juegos ha desde sus inicios. Tradicionalmente, los juegos medían la habilidad del jugador para mover fichas en un tablero o jugar con cartas, todo ello, enmarcado en un conjunto de reglas que hacía que el proceso de generar un árbol de decisión fuera simplemente *cuestión de tiempo*. Desde la aparición de *Pong* en 1970 ha habido un fenomenal crecimiento en complejidad, diversidad y calidad de los videojuegos; tanto que, en muchos casos, los entornos virtuales planteados son muy similares al mundo real. Primeros ejemplos del interés de la AI por los juegos son los programas *Deep Blue* [2] para ajedrez y *Chinook* [10] para damas.

En los últimos años, el surgimiento de un gran número de interfaces de programación de aplicaciones (API) ha permitido probar técnicas de aprendizaje, planificación, control, etc. en entornos continuos de gran similitud con el mundo real [5]. En particular una nueva generación de juegos usa técnicas de Soft Computing [3] en la creación de contenidos, adaptación de luminosidad, control de cámaras o, como ejemplo más obvio, el control automático de jugadores (Non Player Characters, NPC) [13].

Un ejemplo que nos interesa por la cercanía a nuestra investigación es la simulación de carreras de coches, ya que provee un excelente marco para probar aprendizaje, adaptación, razonamiento, evolución, etc. Por otra parte, los entornos de simulación actuales implementan con gran detalle la inercia, aerodinámica y demás aspectos físicos, dotando así a los simuladores de gran realismo, lo que permite la realización de pruebas propias de una rama de investigación tan interesante e importante como son los sistemas de transporte inteligente y el guiado automático de vehículos [7], [9].

En este trabajo se presenta nuestra contribución al *2009 Simulated Car Racing Competition*: La creación de una arquitectura de control completa donde cada módulo será responsable de una acción considerada básica para el control de un coche en carrera: 1) Control de marchas, 2) Control de velocidad, 3) Determinación de la velocidad permitida, 4) Control del volante, 5) Gestión de oponentes y 6) Aprendizaje entre vueltas. Un elemento crítico es el control de la velocidad puesto que se debe ir lo más rápido posible, evitando salidas de pista, derrapes, etc. En nuestro caso se ha implementado un controlador difuso tipo Mamdani [6] con consecuentes de tipo singleton, El controlador de gran simplicidad dio buenos resultados.

El trabajo se estructura como sigue: la sección 2 hace una breve historia de la competición y describe el entorno de simulación TORCS. La sección 3 presenta la arquitectura de la última carrera, así como las mejoras realizadas con respecto a las dos anteriores. La sección 4 presenta los resultados en las tres carreras. Finalmente, la sección 5 presenta las conclusiones obtenidas del presente trabajo, así como líneas de investigación futuras.

2 HISTORIA DE LA COMPETICIÓN Y ENTORNO DE SIMULACIÓN TORCS

Las primeras competiciones tuvieron lugar en 2007, en los congresos del IEEE *Congress on Evolutionary Computation* (CEC) y *Computational Intelligence and Games Symposium* (CIG). Las carreras usaron un modelo computacional gráfica y mecánicamente simple, lo que propició un alto grado de participación [3]. En 2008, en el *IEEE World Congress on Computational Intelligence* (WCCI), la competición se basó en TORCS (*The Open Racing Car Simulator*). TORCS permitió simular carreras complejas tanto desde el punto de vista de la competición, ya que todos los competidores corrían en la misma pista, como físico, ya que TORCS simula con un alto grado de realismo las dinámicas de un vehículo [4]. Otra competición similar se realizó en el IEEE-CIG de 2008.

La última competición, *2009 Simulated Car Racing Competition,* ha consistido en las tres carreras que se citan a continuación y el resultado final ha sido la suma de las puntuaciones obtenidas en cada una de ellas.

- *IEEE Congress on Evolutionary Computation* (CEC09), Trondheim (Noruega), mayo de 2009
- *Genetic and Evolutionary Computation Conference* (GECCO09), Montreal (Canadá), julio de 2009
- *IEEE Congress on Computational Intelligence and Games* (CIG09), Milán (Italia), septiembre de 2009

El simulador TORCS (Figura 1) está escrito en C++ y disponible bajo licencia GPL a través de la web[1]. Los aspectos que lo hacen interesante desde el punto de vista académico son:

- Es totalmente configurable y fácilmente extensible para implementar nuevas funcionalidades.
- Implementa una física motora (aerodinámica, consumo de combustible, tracción,…) altamente sofisticada, así como buenos gráficos 3D.
- Está diseñado para facilitar la implementación de controladores automáticos.

La organización puso a disposición de los participantes una arquitectura cliente-servidor que permite intercambiar información entre el controlador y el entorno Así el controlador recibe información sensorial del entorno (Tabla 1) y envía las órdenes de control.

Una descripción detallada de la arquitectura y de la información sensorial disponible se encuentra en la web[2]. Las variables que determinan las acciones de control son: *volante*, en [-1,1], *acelerador* y *freno* en [0,1] y *marcha* en -1 para marcha atrás, y {1, 2, 3, 4, 5, 6} para el resto de marchas.

[1] http://torcs.sourceforge.net
[2] http://cig.dei.polimi.it

Figura 1: Entorno de Simulación TORCS.

Tabla 1: Descripción de la información sensorial.

NOMBRE	DESCRIPCIÓN
Cabeceo [-π, π] rads	Ángulo formado por el vector director del coche y el eje central de la pista.
Distancia [0, -] m	Distancia desde la línea de salida hasta la posición actual del vehículo.
Marcha {-1, 1, 2, 3, 4, 5, 6}	Marcha actual.
RPM [0, 10.000]	Revoluciones por minuto del motor del vehículo.
Velocidad [0, 300] km/h	Velocidad actual del vehículo.
Ruedas [0, -] rad/s	Vector de 4 elementos con la velocidad angular de cada una de las ruedas.
Deriva	Desviación lateral normalizada del vehículo. En [-1, 1] si se encuentra dentro de la pista, otros valores si está fuera.
Pista [0, 100] m	Vector de 19 sensores de proximidad orientados cada 10º en [-90º, 90º]. Miden la distancia al borde de la pista hasta 100m. Estos valores no son válidos si el coche se encuentra fuera de la pista.
Oponentes [0, 100] m	Vector de 36 sensores de proximidad orientados cada 10º en [-180º, 180º]. Miden la distancia a un posible oponente hasta 100m.

Cada carrera del 2009, como las del 2008, tuvo dos fases: La clasificatoria, en la que un coche circula solo durante 200s en tres pistas, esta fase elige los ocho controladores que recorren mayor distancia. En la segunda fase los elegidos compiten entre sí en tres carreras, cada una de cinco vueltas al circuito, y repetida diez veces con diferentes parrillas iniciales. Los participantes puntúan en 10, 8, 6, 5, 4, 3, 2, y 1 respectivamente, del primero al último, sumando 2 puntos adicionales el que finalice con menos daños y otros 2 el que haga la vuelta más rápida.

3 ARQUITECTURA DE CONTROL

La figura 2 muestra un esquema global de la arquitectura de control presentada al GECCO09 y CIG09. En el CEC09 se presentó una arquitectura preliminar a la descrita en este trabajo [8].

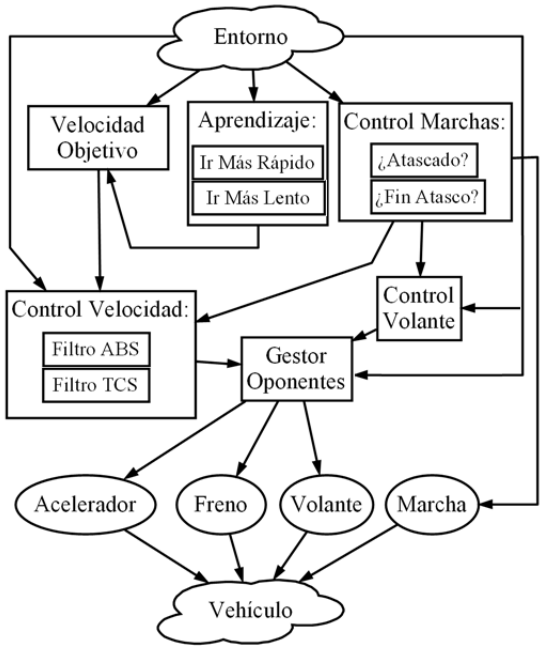

Figura 2: Arquitectura de Control.

En las siguientes secciones, damos una descripción detallada de cada uno de los módulos que la forman.

3.1. VELOCIDAD OBJETIVO

Tras las primeras pruebas, se concluyó que la clave para una conducción rápida era determinar correctamente la velocidad límite en cada uno de los tramos de pista. Esto permite no sólo recorrer la pista lo más rápido posible, sino, evitar salirse de ella, así como derrapar y perder el control del vehículo en una curva, chocar contra las barreras laterales,… El objetivo de este módulo es obtener ese valor de velocidad límite (V_{obj}, en adelante).

Es por ello que se optó por el uso de un controlador difuso para llevar a cabo esta tarea. El controlador se basó en un modelo computacional de coprocesador difuso llamado ORBEX [1] (Ordenador Borroso Experimental). ORBEX es capaz de definir sistemas difusos con funciones de pertenencia de tipo singleton. Sugeno [11] probó que el uso de singletons como consecuentes en las reglas difusas es adecuado para aplicaciones de control, donde el tiempo de respuesta es crucial. Es por ello el uso de este tipo de sistema difuso.

El controlador difuso utiliza como entradas tres de los 19 sensores de pista disponibles. Estos son, el sensor frontal, el máximo valor de los sensores que miden distancia en ±10º y el máximo valor de los sensores que miden distancia en ±20º. Por ello las variables de entrada se llamarán *FRENTE, MAX10* y *MAX20*, respectivamente.

Se optó por el uso de estas variables como entrada del sistema ya que nos proporcionarán, de cierta manera, una forma de predecir características de la pista con una antelación de hasta 100m (máximo valor posible para estos sensores). Un valor máximo de FRENTE indicará vía libre para circular a la máxima velocidad, mientras que, si este valor es bajo, y MAX10 es alto, indicará que el vehículo se está aproximando a una curva, el radio de dicha curva dependerá de los valores concretos de estas variables. El usar valores máximos entre dos sensores se debe a cuestiones de simetría, ya que la velocidad permitida debe ser similar ante curvas de igual radio, ya sean hacia la izquierda o hacia la derecha.

Para codificar las entradas se utilizaron tres funciones de pertenencia trapezoidales, llamadas *Bajo, Medio* y *Alto*. La figura 3 muestra una representación gráfica de dichas funciones.

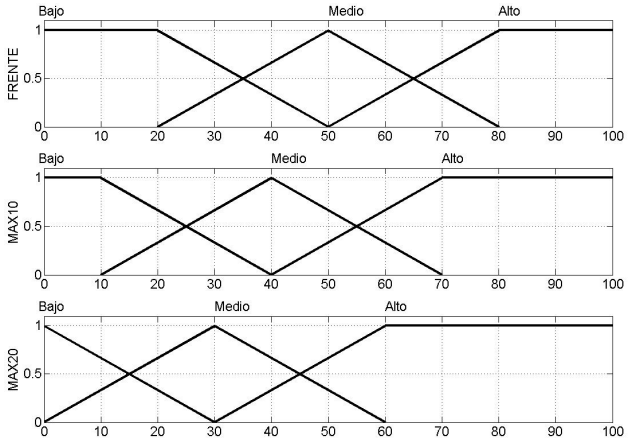

Figura 3: Funciones de pertenencia de las entradas.

Como consecuentes de las reglas se utilizaron los singletons mostrados en la figura 4.

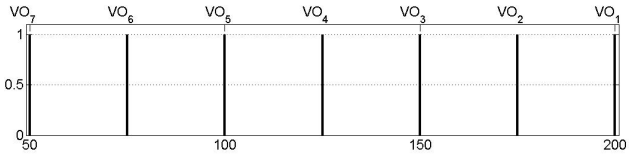

Figura 4: Singletons de salida.

La base de reglas está formada por 7 reglas difusas y una regla no difusa (8), que permite una velocidad máxima en tramos rectos; dicha regla es aplicada siempre que el módulo de aprendizaje entre vueltas no detecte riesgo de salida de la pista en el tramo.

La base de reglas utilizada está formada por el siguiente conjunto de reglas:

1. Si FRENTE es Alto Entonces V_{obj} es VO_1
2. Si FRENTE es Medio Entonces V_{obj} es VO_2
3. Si FRENTE es Bajo Y MAX10 es Alto Entonces V_{obj} es VO_3
4. Si FRENTE es Bajo Y MAX10 es Medio Entonces V_{obj} es VO_4
5. Si FRENTE es Bajo Y MAX10 es Bajo Y MAX20 es Alto Entonces V_{obj} es VO_5
6. Si FRENTE es Bajo Y MAX10 es Bajo Y MAX20 es Medio Entonces V_{obj} es VO_6
7. Si FRENTE es Bajo Y MAX10 es Bajo Y MAX20 es Bajo Entonces V_{obj} es VO_7
8. Si FRENTE=100 O MAX10=100 Entonces V_{obj}=300

El módulo de aprendizaje multiplica por un factor (1 en la primera vuelta) la salida de este módulo, en función de si la velocidad debe ser reducida o ampliada en un determinado tramo de la pista.

3.2. CONTROL DE MARCHAS

La funcionalidad de este módulo es doble. Primero, estará encargado de aumentar o disminuir la marcha actual en función del valor de *RPM*, y segundo, será el encargado de detectar si el coche requiere usar la marcha atrás (se ha quedado atascado, ha derrapado,…) así como cuándo debe dejar de usarse y volver a primera marcha para continuar la carrera.

Según la marcha actual, ésta será aumentada si *RPM* es superior a un valor entre 8000 y 9000 y será reducida si *RPM* es inferior a un valor entre 3000 y 3500. La detección de la necesidad de la marcha atrás se calculará en función del *cabeceo* y *deriva* del vehículo.

3.3. CONTROL DE VELOCIDAD

Por motivos prácticos y para evitar problemas, las salidas *acelerador* y *freno* se codifican como una única variable *pedales*, en [-1, 1] representando así, acciones sobre el freno cuando *pedales<0* y sobre el acelerador cuando *pedales>0*. El valor de dicha variable es una función exponencial de la diferencia entre *velocidad* y V_{obj}.

Los filtros ABS y TCL se añadieron tras la primera competición, al observarse que el vehículo a veces derrapaba perdiendo el control debido a que las acciones sobre los pedales eran muy bruscas, por lo que se implementaron funciones que reducían dichas acciones cuando la diferencia entre *velocidad* y las velocidades calculadas por los valores *ruedas* superaba un determinado umbral (1.5km/h). En ese caso, a la salida se disminuye un quinto de dicha diferencia.

3.4. CONTROL DEL VOLANTE

Para el control del volante, se distinguen tres casos:

1. Marcha normal, si el vehículo avanza dentro de la pista. En este caso se utiliza una media ponderada entre el sensor de *Pista* con mayor valor y sus dos adyacentes, logrando así, dirigir al coche en la dirección con mayor distancia libre.
2. Si el vehículo se encuentra fuera de la pista los sensores de pista no dan valores válidos, en esta situación se usa una función proporcional al *cabeceo* y *deriva* del vehículo.
3. Si el vehículo va marcha atrás, únicamente se utiliza una función inversa al *cabeceo*, para recuperar la orientación y poder continuar la marcha.

3.5. GESTOR DE OPONENTES

Este módulo se introdujo porque TORCS retira de la carrera y pone en la última posición al vehículo cuyos daños superan un determinado umbral. Por ello conviene modificar levemente las acciones de control sobre los pedales y el volante cuando hay riesgo de colisión. Tiene tres objetivos bien diferenciados:

1. Para hacer adelantamientos. Se modifica la acción inferida por el controlador del volante cuando se detecta un oponente en un determinado radio. El valor de la modificación dependerá cuál de los sensores de oponentes ha detectado un valor inferior al umbral, y variará en ±0.1 y ±0.15.
2. Para evitar colisiones. En caso de detectar un vehículo a una distancia frontal inferior a 10 m, se actúa sobre el freno (*pedales=-0.5*) hasta reducir la velocidad al 75% de V_{obj}, para minimizar los daños.
3. Se mejoró el controlador presentado al CEC09, añadiéndole una nueva función que implementa un efecto de "volantazo". Para ello, a la acción sobre los pedales, se le añade una acción sobre el volante, más brusca que la primera (±0.25), logrando así obtener un efecto *volantazo* ante una posible colisión.

3.6. APRENDIZAJE ENTRE VUELTAS

Este módulo se añadió tras la primera competición, para evitar salidas sistemáticas en pistas con curvas muy pronunciadas donde el tiempo de reacción es mínimo. El módulo, usando *distancia,* reconoce los tramos en los que el vehículo no ha tenido tiempo de reacción y se ha salido de la pista y, antes de volver entrar en ellos (con una antelación de 200m) reducirá V_{obj} al 80%.

Además, reconoce largos tramos rectos para dar el valor máximo a V_{obj} (300km/h), sabiendo que no hay ninguna curva próxima.

4 RESULTADOS OBTENIDOS

Los resultados que se muestran han sido obtenidos en los contextos de los congresos CEC09, GECCO09 y CIG09. La arquitectura de control presentada recibe el nombre de *Onieva&Pelta,* en adelante.

En el CEC09 se presentó un prototipo preliminar de arquitectura, se presentaron 6 controladores, incluyendo al ganador de la anterior edición de la competición, llamado *Cham2008*. Las tres pistas utilizadas en esta carrera se muestran en la figura 5.

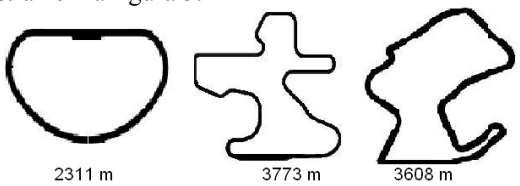

Figura 5: Pistas usadas en CEC09.

En la fase clasificatoria, nos clasificamos en el tercer, segundo y cuarto puesto, para cada una de las pistas. Los resultados de la fase final aparecen en la tabla 2, donde se muestran, para cada pista, la media de las puntuaciones obtenidas en las 10 carreras realizadas para cada uno de los competidores, así como la puntuación final.

Tabla 2: Resultados finales obtenidos en CEC09.

	Pista1	**Pista2**	**Pista3**	**Total**
Cobostar	12	7.5	9	**28.5**
Onieva&Pelta	**8**	**9**	**5**	**22**
Champ2008	5	10	5	**20**
Mr. Racer	3	6	10	**19**
Perez&Saez	6	4	6	**16**
RedJava	5	5	4	**14**

En la primera carrera obtuvimos la segunda posición. La mala puntuación en la tercera pista se debe a que el vehículo siempre se salía de la pista en tres lugares, eso motivó la incorporación del módulo de aprendizaje. Por otra parte, el exterior de la tercera pista es de arena, por lo que al salirse el coche acababa derrapando y perdiendo el control, lo que motivó el añadir filtros para suavizar las acciones sobre los pedales en dichas situaciones.

En GECCO09 se presentó la arquitectura tal y como se describe en el presente trabajo. Participamos 11 competidores, las pistas utilizadas se muestran en la figura 6.

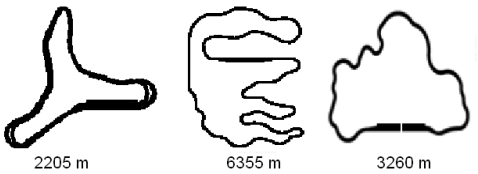

Figura 6: Pistas usadas en GECCO09.

En la fase clasificatoria obtuvimos respectivamente el primer, segundo y segundo puesto. Hay que notar que la primera de las pistas es de tierra, por lo que los filtros sobre las acciones de los pedales jugaron un importante papel en esa primera clasificación. Únicamente 8 se clasificaron para la fase final. Los resultados de esta fase se resumen en la tabla 3.

Tabla 3: Resultados finales obtenidos en GECCO09.

	Pista1	**Pista2**	**Pista3**	**Total**
Onieva&Pelta	**12**	**10**	**10**	**32**
Champ2008	8	7	8	**23**
Cobostar	3.5	5.5	7.5	**16.5**
Epic	4	5.5	3.5	**13**
Simplicity	5	3.5	4	**12.5**
DRT	2.5	3.5	5	**11**
Perez&Saez	3.5	4	3.5	**11**
Jorge Fuentes	3	3	2.5	**8.5**

En esta carrera fuimos los primeros, a notable distancia incluso de *Champ2008* y de *Cobostar*, que en la primera habían sido nuestros principales oponentes.

Al CIG09 acudimos con el mismo controlador usado en la carrera anterior. Hubo 13 participantes y 8 clasificados para la fase final. La figura 7 muestra las pistas utilizadas.

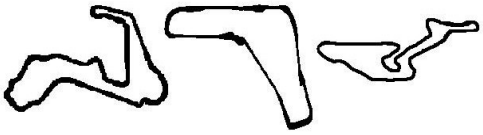

Figura 7: Pistas usadas en CIG09.

En la fase clasificatoria obtuvimos el segundo lugar en las tres pistas. Los resultados de la fase final se resumen en la tabla 4. En la carrera obtuvimos un segundo puesto, a únicamente un punto de *Cobostar*, y con una gran diferencia con el resto de competidores.

Tabla 4: Resultados finales obtenidos en CIG09.

	Pista1	**Pista2**	**Pista3**	**Total**
COBOSTAR	11	7	12	**30**
Onieva&Pelta	**9**	**12**	**8**	**29**
Wolf-Dieter	4	5.5	6	**15.5**
Jorge Muñoz	6	3.5	5	**14.5**
Champ2008	4.5	4	4	**12.5**
Mr. Racer	3.5	6	2.5	**12**
RedJava	2.5	4	4	**10.5**
Perez&Saez	3	2.5	4	**9.5**

Finalmente, en la tabla 5 se muestran los resultados finales del campeonato, siendo este la suma de las puntuaciones obtenidas a lo largo de las tres competiciones. Siendo el resultado obtenido el de campeones del *2009 Simulated Car Racing Championship.*

Tabla 5: Resultados y clasificación final. (NP: No presentado, NC: No clasificado)

	CEC 09	GECCO 09	CIG 09	Total
Onieva&Pelta	**22**	**32**	**29**	**83**
COBOSTAR	28.5	16.5	30	75
Champ2008	20	23	12.5	55.5
Perez&Saez	16	11	12.5	36.5
Mr. Racer	19	NC	12	31
RedJava	14	NC	10.5	24.5
Jorge Muñoz	NP	8.5	14.5	23
Wolf-Dieter	NP	NP	15.5	15.5
Epic	NP	13	NC	13
Simplicity	NP	12.5	NC	12.5
DRT	NP	11	NC	11
Witold	NP	NC	NC	0
Ebner&Tiede	NP	NP	NC	0

5 CONCLUSIONES Y TRABAJO FUTURO

Este artículo presenta nuestra contribución a las carreras de coches simulados celebradas en los tres congresos internacionales que forman la *2009 Simulated Car Racing Competition*. Se ha presentado una arquitectura de control completa, cuyos módulos controlan aspectos básicos de la conducción. Debido a la importancia de determinar la velocidad permitida, se ha utilizado un controlador difuso, técnica no utilizada hasta ahora por ninguno de los participantes, ni de esta, ni de la pasada edición.

Los resultados muestran claramente la utilidad de los controladores difusos en el mundo de los videojuegos, en concreto, en las carreras de coches. A partir de ahora estamos en condiciones de mejorar y ajustar los distintos módulos con técnicas evolutivas por ejemplo, así como de aplicar técnicas de Soft Computing en el diseño automático de arquitecturas alternativas.

Por otra parte, módulos de la arquitectura son susceptibles de ser mejorados o modificados, así queda abierta la posibilidad de incluir nuevos comportamientos y estrategias de carrera, tales como razonamiento temporal para predecir situaciones futuras, o aprendizaje para mejorar a lo largo de la carrera.

Agradecimientos. Este trabajo se ha realizado gracias a los proyectos TRÁNSITO (TRA2008-06602-01, ENVITE Ministerio de Fomento (T7/ 2006), GUIADE (Exp. Nº P9/08), el proyecto TIN2008/01948 del ministerio de Ciencia e Innovación y el proyecto P07-TIC02970 de la Junta de Andalucía.

Referencias

[1] R. García, T. de Pedro. Modeling a Fuzzy coprocessor and its programming language, *Mathware and Soft Computing*, vol. 5(2-3), Pág. 167-174, 1998.

[2] F. Hsu. Behind Deep Blue. *Princeton Univ. Press,* 2002.

[3] J.E. Laird. Research in human-level AI using computer games. *Commun. ACM,* Vol. 3(8), Pág 32-35. 2002.

[4] D. Loiacono, J. Togelius, P.L. Lanzi, L. Kinnaird-Hetter, S.M. Lucas, M. Simmerson, D. Pérez, R.G. Reynolds, Y. Sáez. The WCCI 2008 Simulated Car Racing Competition. *Proceedings of the IEEE Symposium on Computational Intelligence and Games.* Pág. 119-126. 2008.

[5] S. M. Lucas. Computational Intelligence and AI in Games: a New IEEE Transactions. *IEEE Trans. on Computational Intelligece and AI in Games.* Vol 1 (1). 2009.

[6] E. H. Mamdani, S. Assilian. Applications of fuzzy algorithms for control of simple dynamic plant, *Proc. IEEE*, vol. 121, Pág. 1585-1588. 1974.

[7] M.C. Marques, R. Neves-Silva. Traffic simulation for intelligent transportation systems development. *Proceedings 2005 IEEE Intelligent Transportation Systems*. Pág. 320-325. 2005.

[8] E. Onieva, D. A. Pelta, J. Alonso, V. Milanés, J. Pérez. A Modular Parametric Architecture for the TORCS Racing Engine. *Proc. IEEE Symposium on Computational Intelligence and Games,* Pág. 256-262. 2009.

[9] L. Qi. Research on Intelligent Transportation System Technologies and Applications. *2008 Workshop on Power Electronics and Intelligent Transportation System*. Pág. 529-531. 2008.

[10] J. Schaeffer, N. Burch, A.K.Y Björnsson, M. Mueller, R. Lake, P. Lu, S. Sutphen. Checkers is solved, *Science,* vol. 317, Pág. 1518-1522. 2007.

[11] M. Sugeno, On Stability of Fuzzy Systems Expressed by Fuzzy Rules with Singletons Consequents. *IEEE Trans. Fuzzy Systems.* Vol. 7(2), Pág 201-224. 1999.

[12] J. Togelius, S. Lucas, H. Duc Thang, J.M. Garibaldi, T. Nakashima, C. Hiong Tan, I. Ihanany, S. Berant, P. Hingston, R.M. MacCallum, T. Haferlach, A. Gowrisankar, P. Burrow. The IEEE CEC Simulated Car Racing Competition. *Genetic Programming and Evolvable Machines*, vol 9, Pág. 295-329, 2008.

[13] G. Yannakakis and J. Hallam, "Evolving opponents for interesting interactive computer games," From Animals to Animats, vol. 8, pp. 499–508, 2004.