Generación Eficiente de un Controlador Difuso con alta Interpretabilidad para Navegación de Robots Móviles*

Manuel Mucientes
Dept. Electrónica y Computación
Universidad de Santiago de Compostela
15782 Santiago
manuel@dec.usc.es

Jorge Casillas
Dept. Ciencias de la Computación e I.A.
Universidad de Granada
18071 Granada
casillas@decsai.ugr.es

Resumen

El trabajo presenta el diseño de un controlador difuso para el comportamiento de seguir contornos en robótica móvil utilizando la metodología COR con optimización mediante colonias de hormigas. El sistema ha sido evaluado en diferentes entornos simulados, y se ha comparado con otro controlador basado en algoritmos genéticos. La aproximación propuesta obtiene una base de conocimiento altamente interpretable en un tiempo reducido.
Palabras Clave: robótica móvil, seguir contornos, control difuso, interpretabilidad, colonias de hormigas.

1. Introducción

El campo de la robótica móvil se caracteriza por la gran cantidad de incertidumbre presente en los entornos reales. Además, la información que proporcionan los sensores es ruidosa y poco fiable. La lógica difusa ha demostrado ser una herramienta útil para tratar con toda esta incertidumbre, y ha sido ampliamente utilizada para el diseño de comportamientos en robótica [7]. El diseño de sistemas basados en reglas difusas requiere un profundo conocimiento de la tarea a controlar, y su ajuste requiere una gran cantidad de tiempo [5]. Por estos motivos, en los últimos años se ha generalizado el uso de métodos de aprendizaje para el diseño de controladores difusos.

En este artículo se presenta el diseño de un controlador difuso para el seguimiento de contornos en robótica móvil usando la metodología COR (Cooperación entre Reglas) [1, 2]. Las principales ventajas de la aproximación propuesta son la facilidad de diseño, la rapidez en la obtención de la base de conocimiento, y también la alta interpretabilidad de la misma. Finalmente, la utilización de una función que puntúa la acción del controlador sobre cada uno de los ejemplos del conjunto de entrenamiento permite que la calidad del comportamiento aprendido no dependa del entorno.

El artículo se organiza como sigue: en la sección 2 se explica el comportamiento de seguir contornos, y en la 3 la metodología utilizada. La sección 4 muestra los resultados y, finalmente, se exponen las conclusiones.

2. Aprendizaje del comportamiento de seguir contornos

El comportamiento de seguir contornos se utiliza habitualmente cuando el robot se encuentra explorando un área desconocida, o cuando se mueve entre dos puntos en un mapa. Un buen controlador para este comportamiento se debe caracterizar por tres aspectos: mantener una distancia adecuada con el contorno que se sigue, moverse a la máxima velocidad posible y, finalmente, tratar que los movimientos sean suaves y progresivos. En lo que sigue, se asumirá que el robot sigue el contorno situado a su derecha (para seguir el otro contorno basta con realizar un intercambio en las entradas sensoriales).

Las variables de entrada del sistema de control son la distancia derecha (DD), el cociente de distancias (CD) (muestra la posición relativa del robot en el entorno), la velocidad lineal del robot (VL), y la orientación del robot con respecto al contorno que sigue. Las variables de salida son la aceleración lineal y la velocidad angular.

$$CD = \frac{distancia\,izquierda}{DD} \tag{1}$$

Se ha escogido un conjunto de ejemplos (23085) para el aprendizaje de la base de conocimiento. Estos ejemplos cubren todo el universo de discurso de todas las

*Trabajo financiado en parte por el Ministerio de Ciencia y Tecnología mediante los proyectos TIC2002-04036-C05-01, TIC2003-00877 y TIC2003-09400-C04-03

variables de la parte antecedente de la regla, con una precisión p_n, donde n es la variable. La función SF, que puntúa la acción de la base de reglas sobre un ejemplo, se define como:

$$SF\left(RB\left(e^l\right)\right) = \frac{1}{\alpha_1 + \alpha_2 + \alpha_3 + 1} \quad (2)$$

donde α_1, α_2, y α_3 son respectivamente:

$$\alpha_1 = 100\frac{|DD - distancia\ de\ referencia|}{p_{DD}} \quad (3)$$

$$\alpha_2 = 10\frac{|velocidad\ máxima - VL|}{p_{VL}} \quad (4)$$

$$\alpha_3 = \frac{|orientación|}{p_{orientación}} \quad (5)$$

y p_{DD}, p_{VL}, y $p_{orientación}$ son las precisiones de las respectivas variables de entrada. Las precisiones son utilizadas en estas ecuaciones con el fin de evaluar las desviaciones en los valores de las variables respecto a los deseados, pero medidos en unidades de p_n. Esto hace que las desviaciones de las distintas variables sean comparables, con lo que es posible asignar un peso a cada desviación. Estos pesos (100, 10 y 1 para (3), (4), y (5)) han sido obtenidos de forma heurística.

El índice que mide la calidad global de cada conjunto de reglas es:

$$f(RB) = \frac{1}{2 \cdot NE}\sum_{l=1}^{NE}\left\{\left\{1 - \frac{SF\left(RB\left(e^l\right)\right)}{\text{máx}\left(SF\left(e^l\right)\right)}\right\} \cdot \omega\right\}^2 \quad (6)$$

donde NE es el número de ejemplos, ω es un factor de escala que toma el valor 1000, y máx $\left(SF\left(e^l\right)\right)$ es la máxima puntuación que una acción puede obtener para el ejemplo e^l. Estos valores han sido obtenidos antes del comienzo del algoritmo, puntuando todas las posibles acciones sobre cada ejemplo.

3. Metodología de aprendizaje basada en COR

El proceso seguido para aprender el controlador difuso se basa en la metodología COR (Cooperación entre Reglas) propuesta en [1] y extendida en [2]. Hemos seleccionado este proceso debido a sus buenas propiedades para obtener bases de conocimiento rápidamente con gran interpretabilidad. Los siguientes apartados describen la metodología de aprendizaje, un análisis de sus principales propiedades y el algoritmo propuesto basado en dicha metodología.

3.1. Metodología COR

En los últimos años se ha propuesto en la literatura especializada una familia de métodos simples y eficientes de extracción de reglas difusas guiados por criterios de cobertura de los datos del conjunto de ejemplos. Su simplicidad, además de su velocidad y fácil comprensión, hace a estos métodos muy adecuados para tareas de aprendizaje. Sin embargo, estos métodos generalmente buscan reglas difusas con el mejor comportamiento individual (por ejemplo, [8]) y, por tanto, no se considera la interacción global entre las reglas debida al razonamiento interpolado realizado por los sistemas difusos, provocando así una mala precisión en algunos casos.

Con la intención de paliar estos inconvenientes manteniendo las interesantes ventajas de estos métodos, en [1] se propone la metodología COR. En lugar de seleccionar el consecuente con el mejor comportamiento en cada subespacio, esta metodología considera la posibilidad de usar otro consecuente, diferente del mejor, si así se mejora el comportamiento global del sistema difuso gracias a tener una base de reglas con mejor cooperación.

COR se compone de los siguientes pasos: (1) *construcción del espacio de búsqueda*, se obtiene un conjunto de consecuentes candidatos para cada regla; y (2) *selección del conjunto de reglas más cooperativo*, se realiza una búsqueda combinatoria entre los conjuntos para encontrar la combinación de consecuentes con el mejor comportamiento global. En la figura 1 se muestra una descripción más detallada del proceso de generación de reglas basado en COR.

3.2. Ventajas de la Metodología COR para Aprender Controladores Difusos en Navegación con Robots Móviles

La metodología descrita anteriormente tiene varias ventajas interesantes que la hacen muy útil para aprender controladores difusos para navegación con robots móviles. En los siguientes apartados destacamos algunas de ellas.

3.2.1. Reducción del espacio de búsqueda

La metodología COR reduce el espacio de búsqueda basándose en información heurística. Este hecho diferencia a COR de otros métodos de aprendizaje de reglas difusas y le permite ser más rápido y realizar una mejor exploración de las soluciones. Este es un aspecto importante en el aprendizaje de controladores difusos, donde se suele utilizar un alto número de ejemplos. En el comportamiento de seguimiento de contornos presentado en este artículo, donde se consideran 23.085 ejemplos, la metodología propuesta empleó tan solo 20 minutos para obtener el controlador. A diferencia de esto, una solución basada en algoritmos genéticos con el mismo número de ejemplos tardaría varias horas.

Entradas:

- *Un conjunto de datos entrada-salida* —$E = \{e_1, \ldots, e_l, \ldots, e_N\}$, donde $e_l = (x_1^l, \ldots, x_n^l, y_1^l, \ldots, y_m^l)$, $l \in \{1, \ldots, N\}$, siendo N el tamaño del conjunto de datos, y n (m) el número de variables de entrada (salida)— *que represente el comportamiento del problema a resolver.*
- *Una partición difusa de los espacios de las variables.* Sea $\mathcal{A}_i$ el conjunto de términos lingüísticos de la i-ésima variable de entrada, siendo $i \in \{1, \ldots, n\}$, y $\mathcal{B}_j$ el conjunto de términos lingüísticos de la j-ésima variable de salida, siendo $j \in \{1, \ldots, m\}$, donde $|\mathcal{A}_i|$ ($|\mathcal{B}_j|$) es el número de etiquetas de la i-ésima (j-ésima) variable de entrada (salida).

Algoritmo:

1. *Construcción del espacio de búsqueda*:

   1.1. *Definir los subespacios de entrada difusos que contengan ejemplos positivos*: Para ello, dado el conjunto de ejemplos positivos $E^+(S_s)$ para cada subespacio de entrada difuso $S_s = (A_1^s, \ldots, A_i^s, \ldots, A_n^s)$ (con $A_i^s \in \mathcal{A}_i$):

   $$E^+(S_s) = \{e_l \in E \; tal \; que \; \forall i \in \{1, \ldots, n\}, \forall A_i' \in \mathcal{A}_i, \; \mu_{A_i^s}(x_i^l) \geq \mu_{A_i'}(x_i^l)\}, \tag{7}$$

   el conjunto de subespacios con ejemplos positivos se define como $S^+ = \{S_h \mid E^+(S_h) \neq \emptyset\}$.

   1.2. *Generar el conjunto de reglas candidatas en cada subespacio con ejemplos positivos*: En primer lugar se define el conjunto de consecuentes candidatos asociado con cada subespacio $S_h \in S^+$:

   $$\begin{aligned} C(S_h) = \{ \quad & (B_1^{k_h}, \ldots, B_m^{k_h}) \in \mathcal{B}_1 \times \ldots \times \mathcal{B}_m \; tal \; que \\ & \exists e_l \in E^+(S_h) \; donde \; \forall j \in \{1, \ldots, m\}, \forall B_j' \in \mathcal{B}_j, \; \mu_{B_j^{k_h}}(y_j^l) \geq \mu_{B_j'}(y_j^l) \quad \}. \end{aligned} \tag{8}$$

   A continuación se define el conjunto de reglas candidatas para cada subespacio como $CR(S_h) = \{R_{k_h} = [\text{SI } X_1 \text{ es } A_1^h \text{ y ... y } X_n \text{ es } A_n^h \text{ ENTONCES } Y_1 \text{ es } B_1^{k_h} \text{ y ... y } Y_m \text{ es } B_m^{k_h}] \; tal \; que \; (B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)\}$. Para permitir a la metodología COR reducir el número inicial de reglas difusas, se añade el elemento especial $R_\emptyset$ (que significa "no considerar") a cada conjunto de reglas candidatas, es decir, $CR(S_h) = CR(S_h) \cup R_\emptyset$. Si se selecciona este elemento, no se generará ninguna regla en el correspondiente subespacio de entrada difuso.

2. *Selección del conjunto de reglas difusas más cooperativo* — Esta fase se realiza ejecutando un algoritmo de búsqueda combinatoria para encontrar la combinación $RB = \{R_1 \in CR(S_1), \ldots, R_h \in CR(S_h), \ldots, R_{|S^+|} \in CR(S_{|S^+|})\}$ con el mejor comportamiento.

   Se considera un índice $f(RB)$ para medir la calidad global del conjunto de reglas codificada en cada solución. Para obtener soluciones con gran interpretabilidad, se modifica la función original para penalizar excesivo número de reglas:

   $$f'(RB) = f(RB) + \beta \cdot f(RB_0) \cdot \frac{\#RB}{|S^+|} \tag{9}$$

   siendo $\beta \in [0, 1]$ un parámetro que regula la importancia del número de reglas, $\#RB = |\{R_h \in RB \; tal \; que \; R_h \neq R_\emptyset\}|$ el número de reglas usado en la solución a evaluar y RB_0 la base de reglas inicial.

Figura 1: Algoritmo COR

Esta reducción del espacio de búsqueda se realiza imponiendo dos restricciones:

1. *Número máximo de subespacios de entrada difusos*: El número máximo de subespacios de enterada difusos, y por tanto el número máximo de reglas difusas, se limita mediante el conjunto de ejemplos positivos. Las restricciones impuestas para construir $E^+(S_s)$ — ecuación (7)— dividen el espacio de entrada en una rejilla crisp mediante los puntos de corte entre los conjuntos difusos de las etiquetas y, por consiguiente, cada ejemplo contribuye a generar una sola regla. Esto es una selección del conjunto de espacios conservadora que genera el menor número posible de reglas que garantice una cobertura completa de los ejemplos.

En nuestro problema, dado que el conjunto de ejemplos está uniformemente distribuido en el espacio de entrada completo (como se describe en la sección 2), no se realiza una reducción del número máximo de subespacios de entrada difusos. Sin embargo, el hecho de asignar cada ejemplo a un solo subespacio implica reducir el número de consecuentes candidatos, ya que el conjunto de ejemplos positivos se reduce.

2. *Conjunto de reglas candidatas en cada subespacio*: Se realiza una segunda reducción del espacio de búsqueda restringiendo el conjunto de consecuentes posibles para cada combinación de antecedentes. De nuevo, usamos una condición restrictiva para construir $C(S_h)$ —ecuación (8)— que genera un menor número de reglas candidatas.

Para ilustrar el efecto de esta reducción del espacio de búsqueda podemos comprobar que, a partir del conjunto de datos de ejemplos propuesto en la sección 2, y usando el siguiente número de términos lingüísticos para cada variable de entrada/salida, $|\mathcal{A}_1| = 5$, $|\mathcal{A}_2| = 2$, $|\mathcal{A}_3| = 5$, $|\mathcal{A}_4| = 2$, $|\mathcal{B}_1| = 9$, $|\mathcal{B}_2| = 9$, nuestra metodología genera un espacio de búsqueda de $\prod_{S_h \in S^+} |C(S_h)| = 9{,}8\text{e}{+}89$ combinaciones, mientras que el total de posibles combinaciones (considerando los $|S^+| = 100$ subespacios de entrada analizados) es $(|\mathcal{B}_1| \cdot |\mathcal{B}_2|)^{100} = 7{,}1\text{e}{+}190$.

3.2.2. Propiedades de interpretabilidad

La metodología propuesta tiene también algunas ventajas interesantes desde el punto de vista de la interpretabilidad del modelo difuso obtenido. Este es un aspecto importante en control difuso para navegación de robots móviles ya que permite que las acciones del robot sean fácilmente compresibles y puedan ser comunicadas a otros módulos que supervisen el comportamiento de la arquitectura de control. Básicamente, podemos destacar dos aspectos:

• *La estructura del modelo y las funciones de pertenencia permanecen invariables para una excelente interpretabilidad*: La metodología COR es un esfuerzo por explotar la capacidad de precisión de los sistemas difusos lingüísticos centrándose exclusivamente en el diseño de la base de reglas. En este caso, las funciones de pertenencia y la estructura del modelo se mantienen invariables, resultando así en la máxima interpretabilidad.

• *Reducción de la base de reglas para mejorar la interpretabilidad y la precisión*: Al definir una base de reglas, es común incluir reglas redundantes o reglas que generan conflictos con otras en ciertas ocasiones. Además, una base con un alto número de reglas es difícil de interpretar, incluso cuando se considera una estructura de regla difusa lingüística.

Para afrontar este problema se suele realizar un *post-procesamiento*. Cuando se consideran restricciones de interpretabilidad, las reglas pueden ser fusionadas [6], generando así una estructura dispersa (no basada en una rejilla) donde cada regla usa diferentes conjuntos difusos para cada variable.

Otra posibilidad si queremos obtener reglas difusas lingüísticas con gran interpretabilidad es realizar un proceso de selección para obtener un subconjunto de la base de reglas original. Sin embargo, este enfoque no parece apropiado para generar un conjunto de reglas preciso ya que no se considera la interdependencia existente entre las tareas de aprendizaje y reducción. Es decir, es seguro que después de la reducción del conjunto de reglas, el nuevo conjunto de reglas que mejor cooperan será diferente.

La metodología COR realiza el proceso de reducción al mismo tiempo que el aprendizaje con la intención de mejorar la precisión (eliminando reglas conflictivas y erróneas) y la interpretabilidad. Este proceso se realiza añadiendo la regla nula ($R_\emptyset$) a cada conjunto de reglas candidato correspondiente a cada subespacio, como se muestra en el paso 1.2. de la figura 1. Además, se considera una función objetivo modificada, ecuación (9), para penalizar un alto número de reglas y favorecer aún más la reducción.

3.3. Metodología COR con Optimización Basada en Colonias de Hormigas

Dado que el espacio de búsqueda manejado en el paso 2. de la metodología COR es grande, resulta necesario usar técnicas de búsqueda aproximadas. En [1] se obtuvieron modelos lingüísticos con buena precisión utilizando enfriamiento simulado. Sin embargo, dado que una de las restricciones de nuestro problema es el manejo de una función de evaluación computacionalmente costosa, en este artículo se propone el uso de optimización mediante colonias de hormigas (OCH) [4]. Esta es una técnica de búsqueda bio-inspirada que considera información heurística para permitir obtener buenas soluciones con rapidez. Esta sección describe los componentes del algoritmo propuesto.

3.3.1. Representación del problema basado en COR

Para aplicar OCH en la metodología COR es conveniente verla como un problema de optimización combinatoria con la capacidad de ser representado en un grafo ponderado. De esta forma, se puede afrontar el problema considerando un número fijo de subespacios e interpretando el proceso de aprendizaje como la forma de asignar vectores de consecuentes —es decir, etiquetas de las particiones difusas de las variables de salida— a estos subespacios según un criterio de optimalidad (esto es, siguiendo la metodología COR).

Por tanto, de acuerdo con la figura 1, cada nodo $S_h \in S^+$ se asigna a cada consecuente candidato $(B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)$ y al símbolo especial "no considerar" ($R_\emptyset$) que indica la ausencia de reglas en el correspondiente subespacio.

3.3.2. Información heurística

La información heurística sobre la preferencia potencial de elegir un vector de consecuentes específico, B^{k_h}, en cada combinación de antecedentes (subespacio) se determina de la siguiente forma para $B^{k_h} =$

$(B_1^{k_h}, \ldots, B_m^{k_h}) \in C(S_h)$:

$$\eta_{hk_h} = \max_{e_l \in E^+(S_h)} Min\left(\mu_{A^h}(x^l), \mu_{B_j^{k_h}}(y^l)\right). \quad (10)$$

Si $B^{k_h} \notin C(S_h)$, hacer $\eta_{hk_h} = 0$.

Posteriormente, para el símbolo "no considerar", hacer lo siguiente:

$$\eta_{h,|\mathcal{B}_1|\cdot\ldots\cdot|\mathcal{B}_m|+1} = \frac{1}{\max\limits_{k_h \in \{1,\ldots,|C(S_h)|\}} \eta_{hk_h}}. \quad (11)$$

3.3.3. Inicialización de la feromona

El valor inicial de la feromona para cada asignación se obtiene de la siguiente forma:

$$\tau_0 = \frac{1}{|S^+|} \sum_{S_h \in S^+} \max_{B^{k_h} \in C(S_h)} \eta_{hk_h}. \quad (12)$$

Así, la feromona inicial será el valor medio del camino construido tomando el mejor consecuente en cada regla de acuerdo con la información heurística (se trata, por tanto, de una asignación voraz).

3.3.4. Esquema de OCH: sistema de la mejor-peor hormiga

Una vez diseñados los componentes previos, se debe definir un algoritmo OCH para resolver el problema. En esta contribución consideramos el algoritmo sistema de la mejor-peor hormiga (SMPH) [3]. Su esquema global se muestra en la figura 2, para más detalle refiérase a [3].

1. Dar un valor inicial de la feromona, τ_0, para cada arco.
2. Mientras (*condición_terminación* no se cumpla) hacer:
   - a) Realizar el camino de cada hormiga mediante el **proceso de construcción de la solución**.
   - b) Aplicar el **mecanismo de evaporación de feromona**.
   - c) Aplicar el **proceso de búsqueda local** sobre la mejor solución actual.
   - d) Actualizar $S_{mejor\ global}$ y $S_{peor\ actual}$.
   - e) Aplicar la **regla de actualización de feromona Mejor-Peor**.
   - f) Aplicar la **mutación de feromona**.
   - g) Si (*condición_estancado* se cumple), aplicar **reinicialización**.

Figura 2: Algoritmo SMPH

4. Resultados

El controlador aprendido ha sido probado en cuatro entornos simulados diferentes utilizando el software del robot Nomad 200. Estos entornos incluyen todo el abanico de situaciones a las que el robot puede enfrentarse durante la navegación. Ninguno de los entornos de prueba ha sido utilizado durante el aprendizaje, puesto que en esta etapa se ha empleado un conjunto de ejemplos escogidos de forma que se cubra todo el espacio de entradas con una precisión adecuada. Estas condiciones garantizan que la calidad del controlador obtenido no dependa de los entornos en los que se ha entrenado el sistema, y además que el robot sea capaz de enfrentarse a cualquier situación.

La figura 3 muestra el camino seguido por el robot a lo largo de uno de los entornos de prueba (entorno D). La trayectoria del robot se representa mediante círculos. Una mayor concentración indicará menor velocidad. El controlador aprendido tiene 77 reglas y ha sido obtenido en menos de 20 minutos. Si en una determinada situación no se dispara ninguna regla, se seleccionará aceleración lineal y velocidad angular nulas. La máxima velocidad del robot es de 61 cm/s, y la distancia de referencia a la que se debe seguir la pared es de 51 cm. Se han llevado a cabo 10 pruebas para cada uno de los entornos de test. En la tabla 1 se muestran los valores medios de algunos parámetros que reflejan el rendimiento del controlador. Estos parámetros son la distancia media a la pared derecha (el contorno que se sigue), la velocidad lineal media, el tiempo empleado en recorrer el camino, y el cambio medio en velocidad entre dos ciclos consecutivos (refleja la suavidad del comportamiento).

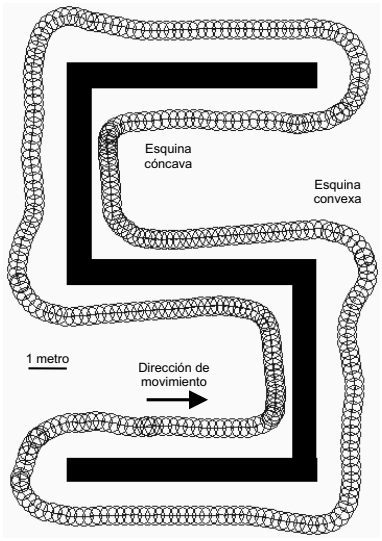

Figura 3: Recorrido del robot en el entorno D.

Con el fin de mostrar la calidad del controlador se va a describir el recorrido del robot en el entorno D (figura 3). Este entorno es bastante complejo, pues tiene cuatro esquinas cóncavas y seis convexas a lo largo de sus 59 metros. En estas situaciones el robot deja de

detectar la pared correctamente en algunos instantes, y además debe reducir la velocidad significativamente. A pesar de ello la velocidad media obtenida ha sido alta, y la distancia a la que se ha seguido el contorno se encuentra próxima a la distancia de referencia. La diferencia es debida al elevado número de esquinas de este entorno, ya que en ellas la orientación del robot es mala, y se da prioridad al giro rápido antes que a una distancia correcta.

Ent.	DD (cm)	Vel. (cm/s)	Cambio vel. (cm/s)	Tpo (s)
A	65	55	5.73	62
B	57	52	5.75	103
C	55	50	4.10	84
D	55	54	4.27	112

Tabla 1: Valores medios de algunos parámetros (77 reglas lingüísticas).

Para evaluar el controlador obtenido, se ha realizado una comparación con otro diseñado utilizando algoritmos genéticos [6]. El diseño en [6] constaba de dos etapas: aprendizaje de la base de datos y una base de reglas general (aproximación Pittsburgh), y reducción de la base de reglas generada mediante la fusión de reglas. Esta reducción provoca una pérdida de interpretabilidad en la base de conocimiento final, pues las etiquetas usadas en cada regla no son generales, sino dependientes de la misma. Este controlador tiene 46 reglas (en la tabla 2 se muestran los valores medios de algunos parámetros para los entornos de prueba). El tiempo empleado en el aprendizaje es elevado (varias horas) si se compara con los menos de 20 minutos necesarios para la obtención del controlador presentado en este trabajo.

Ent.	DD (cm)	Vel. (cm/s)	Cambio vel. (cm/s)	Tpo (s)
A	59	46	8.69	72
B	54	43	9.70	120
C	54	36	7.75	114
D	54	46	7.18	127

Tabla 2: Valores medios de algunos parámetros para el sistema descrito en [6] (46 reglas difusas)

El controlador descrito en este trabajo presenta un mayor número de reglas (77 vs. 46), pero mejora los resultados obtenidos en [6]. La velocidad media es mayor en todos los entornos, reduciendo el tiempo de navegación. La distancia derecha media es similar en ambos controladores y, finalmente, el cambio medio en velocidad se reduce bastante en este controlador, lo que refleja la calidad del comportamiento obtenido en comparación con [6]. Por otra parte, la interpretabilidad de las reglas es alta, en contraposición a [6].

5. Conclusiones

En este trabajo se ha presentado el diseño de un controlador difuso para el seguimiento de contornos utilizando la metodología COR. Las principales ventajas de la aproximación propuesta son la facilidad de diseño, la rapidez en la obtención de la base de conocimiento, el alto grado de interpretabilidad de la misma y, finalmente, que la calidad del comportamiento obtenido no depende del conjunto de entrenamiento.

El controlador ha sido probado en varios entornos simulados mostrando valores adecuados en los valores medios de algunos parámetros que reflejan la calidad del comportamiento. El sistema ha sido también comparado con un diseño previo basado en algoritmos genéticos, incrementando la calidad e interpretabilidad del mismo.

Referencias

[1] J. Casillas, O. Cordón, F. Herrera. COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules. *IEEE Trans. Syst., Man, Cybern. B*, 32(4):526–537, 2002.

[2] J. Casillas, O. Cordón, F. Herrera. COR methodology: a simple way to obtain linguistic fuzzy models with good interpretability and accuracy. In J. Casillas, O. Cordón, F. Herrera, L. Magdalena (Eds.), *Accuracy improvements in linguistic fuzzy modeling*. Springer, Heidelberg, Germany, 2003.

[3] O. Cordón, F. Herrera, I. Fernández de Viana, L. Moreno. A new ACO model integrating evolutionary computation concepts: the best-worst ant system. En *Proc. 2nd Int. Workshop on Ant Algorithms*, 22–29, Brussels, Belgium, 2000.

[4] M. Dorigo, G. Di Caro. The ant colony optimization meta-heuristic. En D. Corne, M. Dorigo, F. Glover (Eds.), *New ideas in optimization*, 11–32. McGraw-Hill, New York, USA, 1999.

[5] M. Mucientes, R. Iglesias, C. V. Regueiro, A. Bugarín, S. Barro. A fuzzy temporal rule-based velocity controller for mobile robotics. *Fuzzy Sets and Systems*, 134:83–99, 2003.

[6] M. Mucientes, D. L. Moreno, C. V. Regueiro, A. Bugarín, S. Barro. Design of a fuzzy controller for the wall-following behavior in mobile robotics with evolutionary algorithms. En *Proc. IPMU'2004*, aceptado, 2004.

[7] A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(4):180–197, 1997.

[8] L.-X. Wang, J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. Syst., Man, Cybern.*, 22(6):1414–1427, 1992.