Evolución de Controladores Difusos Recurrentes Basados en Diagramas de Voronoi

Carlos Kavka, Patricia Roggero y Javier Apolloni

LIDIC
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950 - D5700HHW
San Luis - Argentina
email: {ckavka,proggero,javierma}@unsl.edu.ar

Resumen

Para muchos procesos del mundo real es posible diseñar un controlador difuso que provea buena regularidad usando sólo conocimiento experto. No obstante ello, para lograr un desempeño satisfactorio es necesario hacer uso de métodos más sofisticados. En este trabajo proponemos un modelo basado en sistemas difusos recurrentes, donde el antecedente de las reglas está determinado por una función de pertenencia multidimensional definida en términos de regiones de Voronoi. Las conexiones recurrentes permiten mantener una memoria que guarda información previa. Un algoritmo evolutivo puede evolucionar todas las componentes del sistema difuso recurrente. Además es posible insertar en el sistema conocimiento a priori de forma simple y efectiva. El modelo propuesto es evaluado sobre un problema de control de un robot móvil que debe avanzar evitando obstáculos y siguiendo una trayectoria dirigida por señales luminosas.

Keywords: sistemas difusos, redes neuronales recurrentes, algoritmos evolutivos, control.

1. Introducción

Los procesos de control y supervisión requieren modelos adecuados. Una de las áreas de aplicación de la lógica difusa es el control, donde los controladores difusos resultan ser muy efectivos en el contexto de control de procesos complejos [6].

Tanto los sistemas difusos como las redes neuronales son aproximaciones de inteligencia computacional (soft computing) que modelan comportamiento experto. El objetivo es imitar las acciones de un experto que resuelve problemas, es decir, en lugar de investigar el problema en detalle, se observa como un experto ataca el problema y se obtiene conocimiento por medio de instrucciones y/o aprendizaje. Si se tiene conocimiento expresado como reglas lingüísticas, se puede construir un sistema difuso. Por otro lado, si se tienen datos o se pueden aprender a partir de simulaciones o desde la tarea real, las redes neuronales son más apropiadas. Los méritos de ambos (sistemas difusos y redes neuronales) pueden ser integrados en aproximaciones neuro-difusas [10]. Los algoritmos evolutivos han sido utilizados con éxito para determinar el conjunto de pesos óptimo y la topología de redes neuronales, como así también los parámetros de los sistemas difusos.

En algunas aplicaciones de control, tales como control predictivo y control óptimo de procesos batch, los modelos de predicción multistep-ahead son los más apropiados, usualmente implementados utilizando redes neuronales dinámicas. Ejemplos de este tipo de redes son las redes neuronales globalmente recurrentes, las redes de Elman, las redes de propagación hacia adelante con filtros y las redes localmente recurrentes [13, 12, 2]. En las redes recurrentes las salidas son retroalimentadas a las entradas de la red a través de unidades de demora de tiempo.

En este trabajo, se propone el controlador difuso recurrente Recurrent Fuzzy Voronoi (RFV) basado en el modelo FV no recurrente propuesto en [5]. El modelo FV está basado en conceptos de geometría computacional para representar sistemas difusos, particionando el espacio de entrada en base a diagramas de Voronoi. El modelo RFV, a través de conexiones recurrentes, incorpora la habilidad de procesar secuencias de entradas temporales de longitud arbitraria, lo que permite construir modelos de predicción a largo plazo para procesos no lineales.

Un modelo de controlador difuso recurrente similar al que se propone en este trabajo es el modelo NFSLS propuesto por Mouzouris y Mendel [8], el cual provee una única unidad de retroalimentación. En la aproximación propuesta por Zhang y Morris [15], la red neuro-difusa recurrente está estructurada en cinco capas. La unidad de salida retroalimenta a la capa de entrada y a la capa de función, la que implementa los modelos lineales locales en las regiones de operación difusas. El modelo RSONFIN propuesto por Wu y Lin [14], es una red de inferencia neuronal difusa auto-organizativa recurrente. La estructura es de cinco capas, más una capa de retroalimentación que recibe como entrada las salidas de la capa cuatro (nodos términos de salida), y sus salidas se conectan a la capa dos (nodos términos de entrada). En el modelo DENFIS (Dynamic Evolving Neural-Fuzzy Inference) propuesto por Kasabov y Song [3], los nodos y conexiones son creados y conectados a medida que los ejemplos se presentan. Una capa de memoria de corto plazo se implementa a través de conexiones de retroalimentación desde la capa de nodos de reglas.

El trabajo está organizado de la siguiente forma: la sección 2 introduce la representación del modelo propuesto. La sección 3 explica de que forma se lleva a cabo la evolución de sistemas RFV. La sección 4 presenta el modelo RFV, su estructura y el conjunto de propiedades que dicho modelo posee. La sección 5 valida la aproximación propuesta con resultados experimentales para el problema de robótica planteado en el abstract. Finalmente, en la sección 6 se presentan las conclusiones.

2. Representación RFV

Esta sección introduce el modelo RFV para sistemas difusos recurrentes, el que puede ser considerado como una extensión del modelo propuesto en [5]. Las siguientes secciones introducen los conceptos básicos de geometría computacional necesarios, describen como estos conceptos son utilizados para modelar sistemas difusos recurrentes tipo Takagi-Sugeno y por último se detalla la estructura del modelo propuesto.

2.1. Partición del dominio

La estrategia de partición del dominio está basada en diagramas de Voronoi. Un diagrama de Voronoi induce una subdivisión del espacio de entrada basada en un conjunto de puntos denominados *sitios*. Formalmente [1], un diagrama de Voronoi de un conjunto de p puntos $\mathcal{P} = \{P_1, \ldots, P_p\}$ es la subdivisión del plano en p regiones, una por cada sitio en $\mathcal{P}$, con la propiedad de que un punto M pertenece a la región correspondiente al sitio P_i sí y solo sí la distancia entre M y P_i es menor que la distancia entre M y todos los otros sitios P_j $(j \neq i)$. Formalmente, la región de Voronoi definida por el sitio P_i se define como: $\mathcal{V}(P_i) = \{Q \mid dist(Q, P_i) \leq dist(Q, P_j)\ \forall\ i \neq j\}$, donde $dist(x, y)$ es la

distancia Euclidiana entre x e y. Un concepto relacionado es la triangulación de Delaunay $\mathcal{T}$, la que se define como la máxima subdivisión planar (es decir, una subdivisión tal que no se puede agregar ningún arco que conecte dos vértices sin destruir la planaridad) del conjunto de vértices $\mathcal{P}$, de forma tal que ningún circumciclo de los triángulos de $\mathcal{T}$ contiene puntos de $\mathcal{P}$ en su interior. La figura 1 muestra un ejemplo de un diagrama de Voronoi y su correspondiente triangulación en $\mathbb{R}^2$. Notar que las definiciones se pueden extender a $\mathbb{R}^n$, con $n \geq 2$. Más detalles se pueden encontrar en [1].

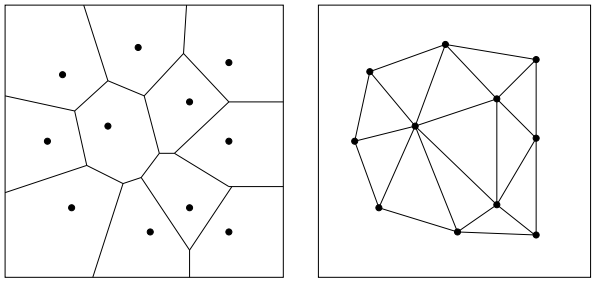


Figura 1: Un ejemplo de un diagrama de Voronoi (izquierda) y su correspondiente triangulación de Delaunay (derecha) para un conjunto de puntos en $\mathbb{R}^2$.

2.2. Representación Fuzzy Voronoi para sistemas difusos tipo Takagi-Sugeno recurrentes

Esta sección introduce la representación Recurrent Fuzzy Voronoi (RFV) para sistemas difusos tipo Takagi-Sugeno recurrentes. Un sistema difuso tipo Takagi-Sugeno recurrente tiene l variables de entrada $x_1, \ldots, x_l$, m variables de salida $v_1, \ldots, v_m$ y r variables $y_1, \ldots, y_r$ que definen las conexiones recurrentes. La regla R_k de un sistema difuso de estas características en RFV tiene la siguiente forma:

$$\begin{aligned} \textit{if } x : y^{t-1} \textit{ is } S_k \quad \textit{then} \quad & v_1 = a_{10} + a_{11}x_1 + \ldots + a_{1l}x_l + b_{11}y_1^{t-1} + \ldots + b_{1r}y_r^{t-1} \\ & \textit{and } \ldots \textit{ and} \\ & v_m = a_{m0} + a_{m1}x_1 + \ldots + a_{ml}x_l + b_{m1}y_1^{t-1} + \ldots + b_{mr}y_r^{t-1} \\ & \\ & y_1^t = c_{10} + c_{11}x_1 + \ldots + c_{1l}x_l + d_{11}y_1^{t-1} + \ldots + d_{1r}y_r^{t-1} \\ & \textit{and } \ldots \textit{ and} \\ & y_r^t = c_{r0} + c_{r1}x_1 + \ldots + c_{rl}x_l + d_{r1}y_1^{t-1} + \ldots + d_{rr}y_r^{t-1} \end{aligned} \tag{1}$$

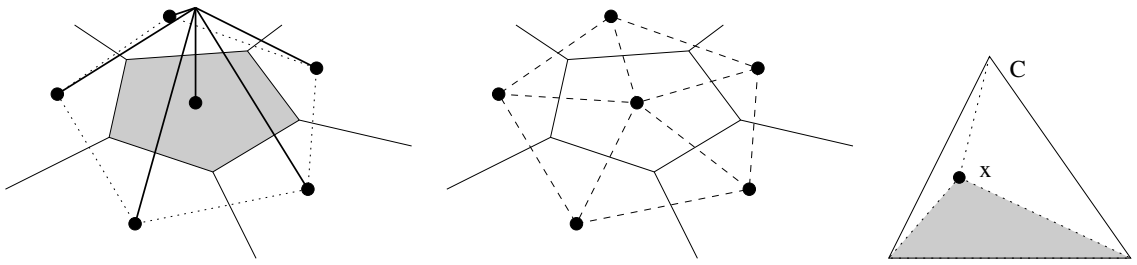
donde S_k es un conjunto difuso multidimensional, $x = (x_1, \ldots, x_l)$ es el vector de entradas, $y = (y_1, \ldots, y_r)$ es el vector de variables recurrentes, el operador : identifica la concatenación de vectores, los supraíndices t y $t-1$ identifican los valores de las variables en el tiempo t y $t-1$ respectivamente, y los valores a_{j0}, a_{ji}, c_{s0}, c_{si}, b_{js}, d_{js} $(1 \leq i \leq l, 1 \leq j \leq m, 1 \leq s \leq r)$ son parámetros reales que definen a las salidas como una combinación lineal de las entradas.

La representación FV asocia a cada regla un único conjunto difuso multidimensional definido a partir de un diagrama de Voronoi correspondiente al conjunto de puntos $\mathcal{P} = \{P_1, \ldots, P_p\}$. Hay tantas reglas difusas como regiones de Voronoi. El conjunto difuso S_k está definido por una función de pertenencia μ_k que toma el valor máximo 1 en el sitio P_k, y decrementa el valor linealmente hasta alcanzar 0 en los centros de las regiones de Voronoi vecinas. Un ejemplo de un conjunto difuso asociado a una regla se muestra en la figura 2-a para $\mathbb{R}^2$.

Formalmente, la pertenencia de un vector de entrada x al conjunto difuso S_k se define como:

$$\mu_{S_k}(x) = \begin{cases} l_C(x) & x \in \mathcal{V}(k) \\ 0 & \text{en otro caso.} \end{cases} \tag{2}$$

donde $C = P_k$ es el sitio asociado a S_k que corresponde a la región de Voronoi $\mathcal{V}(k)$, y $l_C(x)$ es la coordenada baricéntrica de x en el simplex $T_C(x)$ de la triangulación de Delaunay de $\mathcal{P}$ que tiene a C como uno de sus vértices, con $x \in T_C(x)$. La figura 2-b muestra un ejemplo del diagrama de Voronoi y la triangulación de Delaunay asociada. En la figura 2-c, la coordenada baricéntrica $l_C(x)$ corresponde al área (normalizada) en color gris del sub-simplex formado por x y los vértices de $T_C(x)$ exceptuando a C.



(a) (b) (c)

Figura 2: Un ejemplo de un (a) conjunto difuso asociado a una región de Voronoi para $\mathbb{R}^2$ donde el valor de pertenencia está representado en el eje z, y de un diagrama de Voronoi (línea sólida) y (b) su correspondiente triangulación de Delaunay (línea punteada). El gráfico (c) muestra un ejemplo del cálculo de la coordenada baricéntrica. El triángulo exterior corresponde al simplex definido por la triangulación de Delaunay que contiene a x. El valor de pertenencia corresponde al área del triángulo sombreado. Notar que el valor del área es 1 cuando x es igual a C y desciende linealmente hasta alcanzar 0 en los lados del triángulo que están opuestos a C.

La representación FV pertenece a la categoría de los modelos aproximativos, en los que cada regla define sus propios conjuntos difusos. La salida del sistema es siempre continua y cumple con la propiedad de completitud a todo nivel.

3. Evolución de sistemas RFV

Se propone la utilización de algoritmos evolutivos para la optimización de sistemas RFV. Cada individuo está representado por un diagrama de Voronoi (de longitud variable) $\mathcal{P} = \{P_1, ...P_p\}$, donde cada P_i está definido por $l + r$ coordenadas (ver sección 2.2) y para cada $k \in [1, \ldots, p]$, el conjunto de coeficientes a_{j0}, a_{ji}, c_{s0}, c_{si}, b_{js}, d_{js} $(1 \leq i \leq l, 1 \leq j \leq m, 1 \leq s \leq r)$.

El algoritmo evolutivo está descripto en detalle en [11] [4]. El operador de crossover está basado en el intercambio geométrico de sitios de Voronoi entre los padres con respecto a un hiperplano aleatorio. El operador de mutación puede agregar o eliminar un sitio de Voronoi o modificar los parámetros de una regla particular usando mutación de Gauss. Detalles prácticos del algoritmo, incluyendo sus parámetros se dan en la sección 5.

4. Modelo RFV

4.1. Estructura del controlador RFV

La estructura del controlador RFV (ver figura 3) está formada por cuatro capas de unidades. La capa de entrada contiene una unidad por cada variable de entrada x_i $(1 \leq i \leq l)$ y una unidad por cada

conexión recurrente y_j $(1 \leq j \leq r)$. La segunda capa contiene una unidad por cada conjunto difuso S_i $(1 \leq i \leq \omega)$, inducido por la subdivisión del espacio de entrada definida por el diagrama de Voronoi. Cada una de las unidades de la capa uno está conectada con todas las unidades de la capa dos. La capa tres contiene una unidad por cada regla R_i $(1 \leq i \leq \omega)$ conectadas con la correspondiente unidad S_i de la capa dos. La capa cuatro contiene una unidad v_i $(1 \leq i \leq m)$ por cada variable de salida y una unidad y_j $(1 \leq j \leq r)$ por cada conexión recurrente. Cada unidad de la capa tres está conectada con todas las unidades de la capa cuatro. Cada unidad recurrente y_j $(1 \leq j \leq r)$ está conectada con su correspondiente unidad de la capa uno a través de una unidad de demora de tiempo.

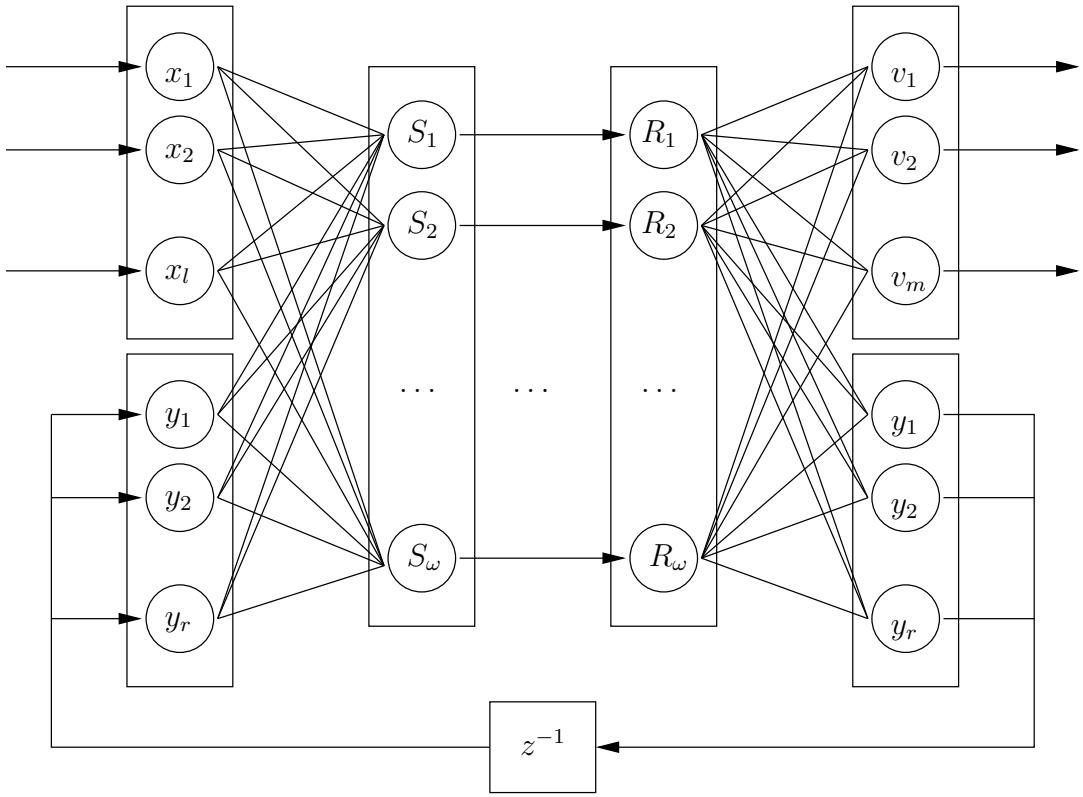


Figura 3: Estructura del controlador RFV.

4.2. Propiedades

El modelo RFV corresponde a un modelo difuso aproximativo en el que cada regla define sus propios conjuntos difusos. Presenta además las siguientes propiedades:

Posee variables internas: Las conexiones recurrentes determinan variables internas, las que no son variables de entrada ni variables de salida, como ocurre usualmente en otros modelos [8] [15]. La integración de este tipo de variables en el modelo permite la definición de reglas que definan los valores de estas variables internas y al mismo tiempo basen sus decisiones en valores obtenidos en un instante de tiempo anterior.

Cumple la propiedad de completitud: Un sistema difuso es ϵ-completo si toda entrada pertenece a algún conjunto difuso con un valor de pertenencia superior o igual a un cierto umbral ϵ. Los sistemas difusos definidos por el modelo RFV cumplen con esta propiedad para $\epsilon = \frac{1}{2}$ por definición de la función de pertenencia (ver ecuación 2), dado que el valor de $l_C(x)$ es mayor a 0.5 en la región de

Voronoi definida por el sitio C. Esta propiedad garantiza que siempre existe para cualquier entrada una regla que puede ser aplicada con un valor de pertenencia conocido.

Las reglas difusas son adaptativas: Los sistemas difusos definidos a través del modelo RFV están compuestos por reglas relacionadas y no reglas independientes, dado que el área de aplicación de las reglas difusas no depende sólo de las reglas individuales, sino de las reglas vecinas. El área de aplicación $\mathcal{A}_k$ de la regla R_k se define como la unión de todas las regiones de Delaunay que contienen al sitio P_k, centro de la regla R_k. Claramente, el área de aplicación de una regla depende de la posición de los centros de los conjuntos difusos definidos por las reglas vecinas. El algoritmo evolutivo evoluciona sistemas difusos definidos por conjuntos de reglas relacionadas en forma sinérgica, y no conjuntos de reglas independientes. Esto asegura además que se puede mantener siempre el nivel de completitud requerido.

Admite inserción de conocimiento: En la mayoría de los sistemas difusos, los usuarios pueden incorporar conocimiento previo en forma de reglas y particiones del espacio de entrada y/o salida. Sin embargo, este conocimiento previo introduce un sesgo que puede ser determinante en términos de la calidad del proceso evolutivo. Una de las mayores ventajas del modelo RFV (y del modelo FV [5]) es que el usuario no necesita especificar el área de aplicación de las reglas que determinan el conocimiento previo. Debido al efecto sinérgico explicado anteriormente, el proceso evolutivo puede modificar el área de aplicación de las reglas agregando, eliminando o modificando reglas vecinas.

5. Experimentos

5.1. Problema

Un problema de control en robótica evolutiva [9] ha sido seleccionado para validar el modelo RFV. Como base de experimentación se ha utilizado un simulador de robot Khepera [7], el cual tiene 8 sensores para medir proximidad de objetos y niveles de luz ambiente, y dos motores independientes para controlar la velocidad y dirección del robot. El problema consiste en dirigir al robot, partiendo de una posición inicial fija, para llegar a una posición de destino que depende de indicaciones luminosas presentes o no en la trayectoria, evitando todos los obstáculos que sea necesario. La presencia de una indicación luminosa indica que el robot debe doblar a la izquierda en la próxima intersección, y la ausencia de indicaciones luminosas indica que el giro debe ser a la derecha. El controlador requiere memoria, dado que la señal luminosa no está presente en la intersección, sino en un punto anterior de la trayectoria. Además, el comportamiento del robot en cada intersección es independiente del comportamiento en la intersección previa. Esto implica que el controlador debe aprender a *olvidar* indicaciones luminosas que afectan el comportamiento en situaciones previas y no determinan el comportamiento en la intersección actual.

El fitness de un controlador difuso recurrente se calcula en forma similar a lo establecido en [9], testeando el controlador en el simulador en e escenarios distintos. Los escenarios definen un punto de inicio, un punto de destino e incluyen intersecciones de caminos en los que las señales luminosas determinan la dirección adecuada. El fitness se acumula en cada paso del robot en forma proporcional a la velocidad de desplazamiento, inversamente proporcional a la distancia del punto de destino, y además disminuyendo cuando el robot se acerca a obstáculos de forma tal de garantizar que el robot circula lejos de ellos. La acumulación de fitness se detiene si el robot golpea una pared, o ha alcanzado un número predefinido de pasos s. El fitness total es el promedio de los valores obtenidos en los e

escenarios. Formalmente, el fitness a ser maximizado correspondiente a un individuo I es:

$$\text{fitness}(I) = \frac{1}{e} \sum_{i=1}^{e} \sum_{t=1}^{s} v(t) * (1 - a(t)) * (1 - d(t)) \tag{3}$$

donde t es el paso del tiempo, $v(t)$ es la velocidad normalizada hacia adelante del robot (suma de la velocidad de ambos motores), $a(t)$ es la activación máxima normalizada de los sensores (por ejemplo, $a(t) = 1$ implica una colisión) y $d(t)$ es la distancia normalizada hasta el destino. Esta función asigna valores mayores a individuos que viajan a la máxima velocidad posible, en línea recta la mayor parte del tiempo, lo más lejos posible de obstáculos y que llegan lo más cerca posible al destino.

5.2. Resultados experimentales

Los controladores se definen con cinco entradas, dos salidas y una variable interna. Las entradas son respectivamente, el promedio de los dos sensores izquierdos, los centrales, los derechos y los posteriores, y el valor promedio de luz ambiente. Las salidas corresponden a la velocidad de los motores. Notar que la presencia de la variable interna hace que las reglas tengan seis entradas y tres salidas. Los parámetros del algoritmo evolutivo se detallan en la tabla 1.

Parámetro	valor
Dimensión del espacio de entrada	5
Dimensión del espacio de salida	2
Número de variables internas	1
Tamaño de población	50
Número de generaciones	100
Selección - Reemplazo	Ruleta - Generacional
Número de pasos de tiempo	500
Tamaño mínimo y máximo de los individuos	10 - 30
Probabilidad de crossover (Voronoi)	0.8
Probabilidad de mutación (Gaussian - addition - deletion)	0.3 - 0.15 - 0.15

Cuadro 1: Parámetros del algoritmo evolutivo

El desempeño de los individuos se evalúa en $e = 4$ escenarios, en los que están presentes todas las posibles combinaciones con dos intersecciones, evaluando a lo sumo $s = 500$ pasos en cada uno. El desempeño de un controlador representativo obtenido luego del proceso evolutivo en los escenarios utilizados durante la evolución se muestra en la figura 4. El punto de partida es siempre el mismo en el extremo inferior, mientras que el punto de llegada está determinado por las indicaciones luminosas presentes o no en el camino.

Los experimentos también fueron repetidos utilizando conocimiento previo. El desempeño de un controlador representativo obtenido luego del proceso evolutivo en los escenarios utilizados durante la evolución se muestra en la figura 5. El conocimiento previo introducido se detalla en la tabla 2.

La semántica de las reglas establecidas como conocimiento previo utilizan a la variable interna como un indicador de memoria cuyo valor es 1 cuando se ha detectado un indicador luminoso y 0 en caso contrario. Las primeras cuatro reglas corresponden a una situación en la que no hay obstáculos (el valor de las cuatro entradas que corresponden a los sensores de proximidad es 0) y tienen como objetivo establecer el valor de la memoria en base a la detección de luz ambiental y el valor previo

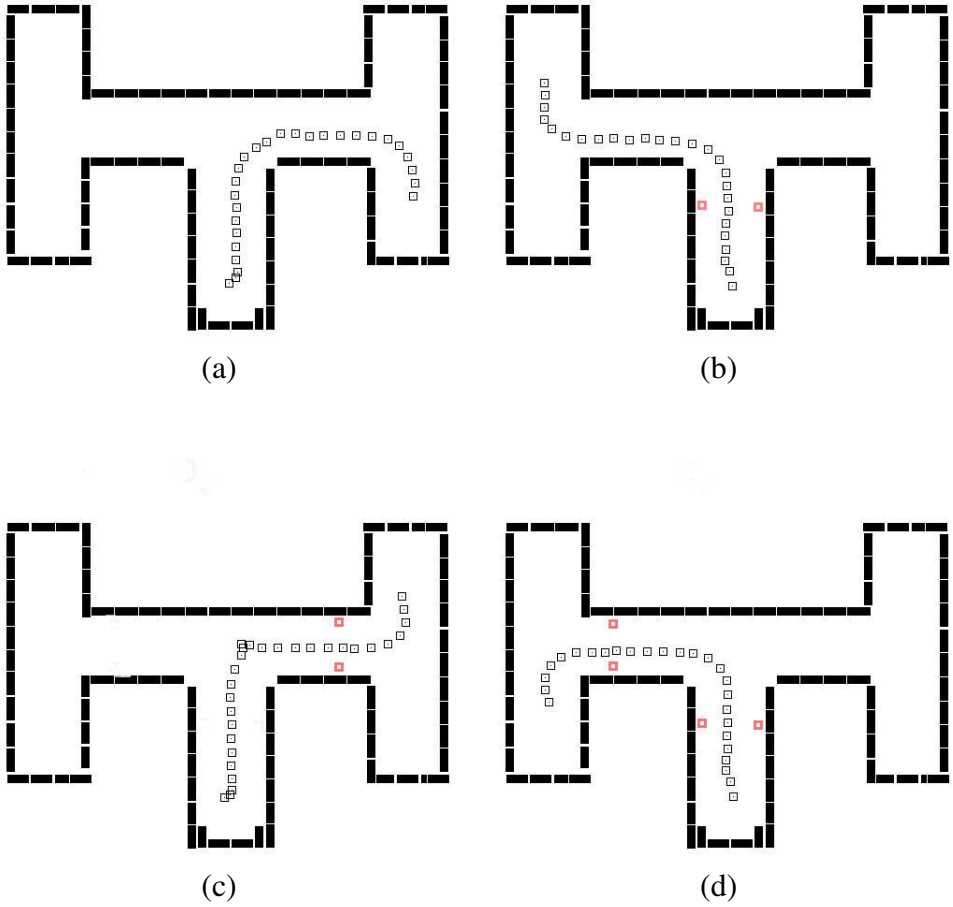


Figura 4: Los gráficos muestran el desempeño de un controlador representativo en los cuatro escenarios utilizados durante la evolución. En el escenario (a) no hay indicaciones luminosas, en el (b) hay una indicación luminosa en el segmento inferior, en el (c) una indicación luminosa en el segmento superior derecho y en el (d) tanto en el segmento inferior como en el superior izquierdo.

de la memoria. En estas cuatro reglas, el valor normalizado de la velocidad para ambos motores se establece en 1 (máxima velocidad), y el valor de la memoria en 0 (regla R_1) en el caso en que no haya luz ambiente ($L = 0$) y no se haya detectado previamente luz ($y_1 = 0$), y 1 en otro caso (reglas R_2 a R_4). Las reglas R_5 y R_6 se aplican sólo en intersecciones (obstáculo en el frente y libertad de girar a izquierda y derecha). La regla R_5 determina un giro a derecha cuando no se ha detectado luz ($y_1 = 0$) y la regla R_6 un giro a izquierda en caso contrario ($y_1 = 1$). Notar que la regla R_6 vuelve el valor de la memoria y_1 a 0, reflejando la situación de *olvido* mencionada en la sección 5.1.

La figura 6 muestra el cambio en el fitness en una ejecución típica del algoritmo evolutivo (promediado en cinco corridas) con respecto al número de generaciones, junto con las barras de error, cuando la evolución se realiza (a) sin y (b) con conocimiento previo. Se puede observar que la calidad de los controladores obtenidos al final del proceso evolutivo es comparable. Sin embargo, el error inicial promedio de los controladores es menor cuando la evolución se realiza con conocimiento previo.

Regla	*punto*	v_1	v_2	y_1
	(*I*,*C*,*D*,*P*,*L*,y_1)	(a_{10},a_{11},a_{12},a_{13},a_{14},a_{15},b_{11})	(a_{20},a_{21},a_{22},a_{23},a_{24},a_{25},b_{21})	(c_{10},c_{11},c_{12},c_{13},c_{14},c_{15},d_{11})
R_1	(0,0,0,0,0,0)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)	(0,0,0,0,0,0,0)
R_2	(0,0,0,0,0,1)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)
R_3	(0,0,0,0,1,0)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)
R_4	(0,0,0,0,1,1)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)	(1,0,0,0,0,0,0)
R_5	(0,1,0,0,0,0)	(1,0,0,0,0,0,0)	(0,0,0,0,0,0,0)	(0,0,0,0,0,0,0)
R_6	(0,1,0,0,0,1)	(0,0,0,0,0,0,0)	(1,0,0,0,0,0,0)	(0,0,0,0,0,0,0)

Cuadro 2: Conocimiento previo introducido antes del proceso evolutivo en término de reglas difusas recurrentes. El valor del *punto* corresponde al centro de la región de Voronoi definido por la regla y está especificado por los valores normalizados de la activación de los sensores izquierdos (I), centrales (C), derechos (D) y posteriores (P), el valor normalizado de la luz ambiental (L) y el valor de la variable interna y_1. Los valores a_{10},...,a_{15},a_{20},...,a_{25},b_{11},b_{21},c_{10},...,c_{15} y d_{11} corresponden a las constantes utilizadas en la definición de las reglas para el cálculo de las salidas v_1 y v_2 y de la variable interna y_1.

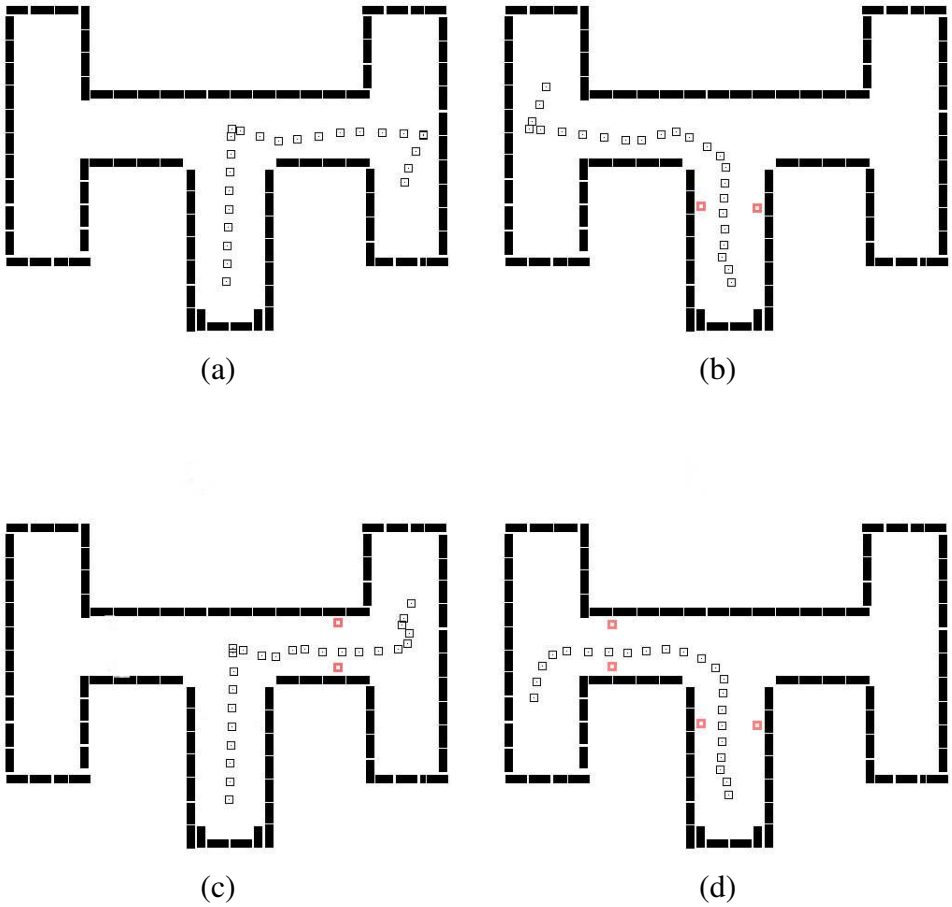
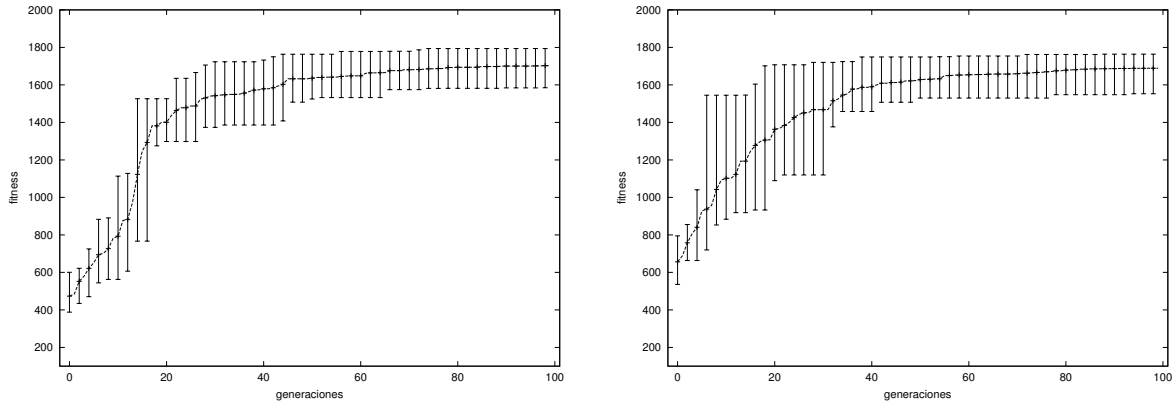


Figura 5: Los gráficos muestran el desempeño de un controlador representativo en los cuatro escenarios utilizados cuando la evolución se realiza con conocimiento previo.

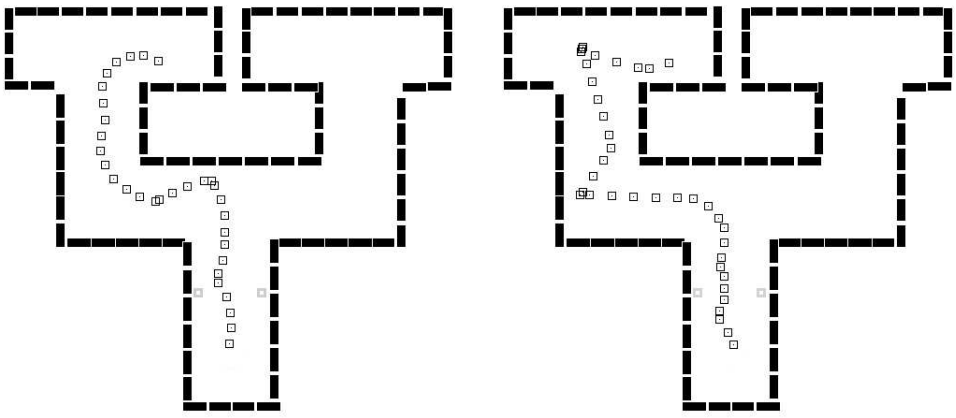
El punto más importante de la utilización de conocimiento previo radica en que es posible establecer una *interpretación semántica* de las variables internas. En este ejemplo, la variable interna y_1 tendrá el valor 1 para indicar que se ha detectado luz ambiente y 0 en caso contrario. En los controladores obtenidos sin conocimiento previo, el comportamiento de las variables internas es impredecible.



(a) (b)

Figura 6: Gráfica del fitness con respecto al número de generaciones cuando la evolución se realiza (a) sin conocimiento previo y (b) con conocimiento previo.

En la figura 7 se muestra el desempeño de un controlador representativo obtenido con y sin conocimiento previo, en un escenario distinto no utilizado durante la evolución. Se puede apreciar que el controlador ha aprendido exitosamente las reglas de navegación determinadas por el problema.



(a) (b)

Figura 7: Desempeño de un controlador representativo en un escenario no utilizado durante el proceso evolutivo cuando la evolución se realiza (a) sin conocimiento previo y (b) con conocimiento previo.

6. Conclusiones

En este trabajo se ha propuesto un modelo de sistemas difusos recurrentes, particularmente adaptado para evolución de controladores, que provee memoria a través del uso de variables internas. El

modelo es una extensión del modelo FV basado en diagramas de Voronoi. La interpretación geométrica de las reglas permite el uso de operadores genéticos geométricos, y permite representar sistemas difusos recurrentes, con reglas sinérgicas que cumplen la propiedad de completitud, proveyendo además una forma simple de introducir conocimiento previo. El modelo ha sido validado en un problema de control de robots móviles utilizando un simulador. Trabajos futuros incluyen experimentos utilizando un robot real.

Referencias

[1] M. de Berg, M. van Kreveld, M. OVermars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer Verlag, 1998.

[2] P. Frasconi, M. Gori, and G. Soda. Local feedback mulilayered networks. *Neural Computing*, 4:120–130, 1992.

[3] Nikola Kasabov and Qun Song. Denfis: Dynamic evolving neural-fuzzy inference systems and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, April 2002.

[4] C. Kavka and M. Schoenauer. Voronoi diagrams based function identification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2003.

[5] C. Kavka and M. Schoenauer. Evolution of voronoi based fuzzy controllers. In *Proceedings of the Parallel Problem Solving in Nature*, 2004.

[6] C. Lee. Fuzzy logic in control systems: Fuzzy logic controller - part i. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–418, March/April 1990.

[7] O. Michel. Kephera simulator package version 2.0: Freeware mobile robot simulator. http://wwwi3s.unice.fr/ om/khep-sim.html.

[8] George Mouzoris and Jerry Mendel. Dynamic non-singleton fuzzy logic systems for nonlinear modeling. *IEEE Transactions on Fuzzy Systems*, 5(2):199–208, May 1997.

[9] S.Ñolfi and D. Floreano. *Evolutionary Robotics, The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Books, 2000.

[10] S. K. Pal and S. Mitra. *Neuro-fuzzy Pattrn Rocognition: Methods in Soft Computing*. New York Wiley, 1999.

[11] M. Schoenauer, F. Jouve, and L. Kallel. Identification of mechanical inclusions. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*. Springer Verlag, 1997.

[12] A. Tsoi and A. Back. Locally recurrent globally feedforward networks: A critical review of architecture. *IEEE Transactions on Neural Networks*, 5:229–239, 1994.

[13] P. Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78:1550–1560, 1990.

[14] Gin-Der Wu and Chin-Teng Lin. A recurrent neural fuzzy network for word boundary dtection in noisy environment. *International Journal of Fuzzy Systems*, 2(1):31–38, March 2000.

[15] Jie Zhang and Julien Morris. Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE Transactions on Neural Networks*, 10(2):313–326, March 1999.