XXV Jornadas de Automática
Ciudad Real, del 8 al 10 de septiembre de 2004

UNA TÉCNICA HÍBRIDA DE MODELIZACIÓN NEUROBORROSA MEDIANTE COLONIA DE HORMIGAS

José Manuel Andujar, Juan Manuel Córdoba, Iñaki Fernández de Viana.
{andujar,juanmanuel.cordoba, i.fviana}@diesia.uhu.es

Departamento de Ingeniería Electrónica, Sistemas Informáticos y Automática
Universidad de Huelva

Resumen

Las colonias de hormigas han demostrado su utilidad en la resolución de problemas de optimización, como pueden ser el problema del viajante de comercio o la asignación cuadrática de tareas, donde el espacio de búsqueda es discreto. En este trabajo vamos a presentar cómo aplicar este tipo de metaheurística (OCH) al problema de la modelización borrosa de sistemas donde las variables a ajustar toman valores continuos. Se demuestra cómo la modelización híbrida de algoritmos OCH más gradiente descendente mejora notablemente la reducción en el error de ajuste de los modelos.

Palabras clave: Algoritmos Evolutivos, Colonias de hormigas, Modelado Borroso, TSK, Gradiente Descendente.

1. INTRODUCCIÓN

En los últimos años, las *Heurísticas basadas en la Naturaleza* o *Algoritmos bioinspirados* [7, 10] han sido unos de los más prometedores campos de investigación en el desarrollo de metaheurísticas. Dentro de estas metaheurísticas [9] se encuentran los Algoritmos Genéticos, Enfriamiento Simulado y Algoritmos de Optimización basados en Colonias de Hormigas (OCH), entre otras.

La metaheurística OCH [17], se basa en simular la forma en que las hormigas llevan la comida hasta el hormiguero recorriendo un camino de longitud mínima.

Con el desarrollo de la lógica borrosa, se han elaborado nuevos tipos de modelos borrosos [4, 5, 6, 11]. El objetivo de la creación de nuevos modelos es el de conseguir mayor precisión, reducción de la complejidad de elaboración y simplificación de la estructura del modelo. Esta necesidad de elaborar nuevos modelos es debida a la gran variedad de sistemas reales, cada vez de mayor complejidad, con diferentes formas de acceso a la información y diferentes formatos de representación de esta información.

La ventaja que presentan los modelos borrosos en comparación con los modelos matemáticos convencionales es que permiten la posibilidad de elaborar el modelo directamente a partir de los datos de entrada/salida obtenidos del sistema. Esta información además puede ser inexacta y poco concreta [25], lo cual facilita la implementación de controladores para plantas complejas [2, 3].

Los modelos Takagi-Sugeno (TS) fueron descritos por primera vez en [28]. Estos modelos también son conocidos como Takagi-Sugeno-Kang (TSK) [24, 29], modelos quasi-lineales o modelos borrosos lineales [4, 5]. El modelo TSK difiere del modelo Mamdani en la forma de las reglas. En el caso de un modelo Mamdani de un sistema con una entrada y una salida las reglas tienen la siguiente forma:

$$Si\ (x\ es\ A)\ Entonces\ (y\ es\ B)$$

donde B es un conjunto borroso. En el caso de un modelo TSK las reglas tienen la forma:

$$Si\ (x\ es\ A)\ Entonces\ (y{=}f(x))$$

Mientras que el modelo Mamdani se caracteriza por su interpretabilidad, el modelo TSK se caracteriza por su precisión. Debido a su precisión, los modelos TSK han sido ampliamente utilizados para el modelado de sistemas [21, 30].

El problema de la obtención de un modelo borroso se reduce al ajuste de los parámetros característicos del mismo. En la bibliografía, se han propuesto diferentes aproximaciones, aunque las más ampliamente utilizadas han sido aquellas basadas en el gradiente descendente [22] y en algoritmos evolutivos [20]. Independientemente de las aproximaciones propuestas, la solución conseguida suele depender de la estimación inicial de los parámetros del modelo. Para esta estimación incial, en la literatura se han propuesto varias metodologías como la de prueba y error [21], utilización de técnicas de clustering [12], o la simple asignación aleatoria [30].

El objetivo de este trabajo es demostrar cómo se puede utilizar los algoritmos OCH para el ajuste de los parámetros de un modelo borroso TSK.

El resto de este artículo se estructura de la forma siguiente: en la Sección II se introducirá el modelado borroso de sistemas. En la Sección III estudiaremos la metaheurística OCH. En la Sección IV, presentamos cómo aplicar las hormigas al modelado borroso. En la Sección V hablaremos de los ejemplos que hemos usado en la experimentación. En la Sección VI, describimos cómo se ha hecho la experimentación, qué resultados se han obtenido y a qué conclusiones se ha llegado. Finalmente, en la Sección VII, hablaremos de futuras líneas de investigación.

2. MODELADO BORROSO DE UN SISTEMA

Considérese el sistema definido por:

$$\begin{array}{c} y_1 = f_1(x_1, x_2, \ldots, x_n) \\ y_2 = f_2(x_1, x_2, \ldots, x_n) \\ \ldots \\ y_n = f_n(x_1, x_2, \ldots, x_n) \end{array} \quad (1)$$

o de forma abreviada:

$$\mathbf{y} = f(\mathbf{x}). \quad (2)$$

Un modelo borroso equivalente al anterior sistema puede ser representado mediante el conjunto de reglas:

$$\begin{array}{c} R^{(l)} : \; Si\, x_1\, es\, A_1^l\, y\, x_2\, es\, A_2^l \ldots x_n\, es\, A_n^l \\ ENTONCES\, y^l\, es\, g^l(\mathbf{x}, \theta^l) \end{array} \quad (3)$$

donde $l = 1 \ldots M$ es el número de reglas del modelo borroso del sistema; A_k^l el conjunto borroso definido en el universo de discurso de la variable de estado x_k, con $k = 1 \ldots n$, y donde n es el número de variables de entrada.

Sea el consecuente de la ecuación (3) TSK [27, 28] con término afín:

$$g^l(\mathbf{x}, \theta^l) = a_0^l + a_1^l x_1 + \ldots + a_n^l x_n \quad (4)$$

donde a_k^l representa el coeficiente constante para la variable de estado x_k para la regla l, y el vector θ^l representa el conjunto de parámetros adaptables:

$$\theta^l = (a_0^l, a_l^l, \ldots, a_n^l). \quad (5)$$

Considerando un sistema borroso TSK con desborrosificador centro-promedio y método de inferencia producto, el sistema borroso equivalente a la expresión (1) puede ser descrito como [21]:

$$\hat{y}_i = \frac{\sum_{l=1}^{M} w^l g^l(\mathbf{x}, \theta^l)}{\sum_{l=1}^{M} w^l} \quad (6)$$

$\hat{y}_i$ es salida estimada del sistema, y w^l es el grado de cumplimiento de la regla l:

$$w^l = \prod_{k=1}^{n} \mu_{F_k}^l(x_k, \sigma^l) \quad (7)$$

donde σ^l son los parámetros característicos de la función de pertenecia $\mu_{F_k}^l(x_k, \sigma^l)$ que se define a partir del conjunto borroso A_n^l.

Si la función (1) se sustituye por la ecuación (6) entonces todos los parámetros de la base de reglas, σ^l y θ^l, pueden ajustarse para reproducir el comportamiento de un conjunto de entrada/salida. Después de adaptar los parámetros, el sistema borroso resultante representa un modelo equivalente al sistema real.

Con el fin de poder minimizar el error entre la salida del modelo borroso y la salida del sistema definido por la ecuación (1), se puede aplicar el método del gradiente descendiente para el ajuste de los parámetros del modelo borroso. Sea una función de error en la k-ésima iteración definida como el error cuadrático medio [30]:

$$J(k) = \frac{1}{2} \sum_{i=0}^{N} (y_i - \hat{y}_i) \quad (8)$$

donde y representa los datos de salida del sistema que se pretende modelar e $\hat{y}$ es el modelo borroso que se pretende obtener. El método del gradiente descendente minimiza la función de coste J ajustando cada parámetro, σ^l y θ^l, con una cantidad proporcional a la derivada de la función con respecto a cada parámetro. Aplicando este principio para minimizar la función J, la ley de adaptación de parámetros que define la parte precedente de la base de reglas responde a:

$$\sigma_i^l(k+1) = \sigma_i^l(k) - \eta \frac{\partial J(k)}{\partial \sigma_i^l(k)} \quad (9)$$

de forma similar, el consecuente de la base de reglas se adapta según:

$$\theta_i^l(k+1) = \theta_i^l(k) - \eta \frac{\partial J(k)}{\partial \theta_i^l(k)} \quad (10)$$

siendo η el factor de aprendizaje, un factor de escala del término derivativo que afecta al tiempo de convergencia del método.

Si usamos una función de pertenencia Gaussiana en la parte del antecedente de la regla:

$$\mu(x) = exp\left(\frac{-(x-\gamma)^2}{2\beta^2}\right) \tag{11}$$

se obtienen las siguientes reglas de adaptación para los centros γ de la función Gaussiana [1]:

$$\gamma_i^l(k+1) = \gamma_i^l(k) - 2\eta e(k) \cdot \frac{\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)}{\sum_{l=1}^{M}\left[\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)\right]} \frac{\left(x_i(k) - \gamma_i^l(k)\right)}{\left(\beta_i^l(k)\right)^2} \tag{12}$$

y los anchos β de la función de pertenencia Gaussiana:

$$\beta_i^l(k+1) = \beta_i^l(k) - 2\eta e(k) \cdot \frac{\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)}{\sum_{l=1}^{M}\left[\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)\right]} \frac{\left(x_i(k) - \gamma_i^l(k)\right)^2}{\left(\beta_i^l(k)\right)^3} \tag{13}$$

donde

$$e(k) = y(k) - \hat{y}(k). \tag{14}$$

Por $\mu_{F_i}^l$ se denota la función de pertenencia i-ésima de la regla l.

Al término afín del consecuente TSK le corresponde la regla de adaptación:

$$\theta_0^l(k+1) = \theta_0^l(k) - 2\eta e(k) \cdot \frac{\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)}{\sum_{l=1}^{M}\left[\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)\right]} \tag{15}$$

El resto de los términos del consecuente TSK:

$$\theta_i^l(k+1) = \theta_i^l(k) - 2\eta e(k) \cdot \frac{\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right) x_i(k)}{\sum_{l=1}^{M}\left[\prod_{i=1}^{n} \mu_{F_i}^l\left(x_i(k), \sigma_i^l(k)\right)\right]} \tag{16}$$

3. ALGORITMOS OCH

En los siguientes puntos vamos a introducir el esquema general de funcionamiento de la metaheurística OCH y, posteriormente, estudiaremos algunos de sus algoritmos más representativos.

3.1. Introducción a la metaheurística OCH

La metaheurística OCH [18] puede englobarse dentro de las metaheurísticas bioinspiradas [7, 10]. Su modelo inicial fue desarrollado por Dorigo, Maniezzo y Colorni [19].

Se compone de algoritmos de búsqueda formados por poblaciones de agentes cooperativos que imitan a las hormigas reales. Concretamente, imitan el comportamiento que tienen estos insectos a la hora de buscar comida. En este proceso de búsqueda, las hormigas van depositando una substancia denominada feromona, siendo capaces de oler esta hormona y dirigir su búsqueda y, por tanto, la de la colonia, en función de las deposiciones de feromona. Cuando una hormiga llega a una intersección, decide el camino a seguir de un modo probabilístico en función de la feromona depositada en cada ramificación. Este movimiento continuado provoca que los caminos más cortos sean más frecuentados, por lo que reciben mayor cantidad de feromona, mientras que los más largos serán progresivamente abandonados y la feromona asociada acabará por evaporarse. De este modo, se acaba formando un camino de longitud mínima entre el hormiguero y la comida.

Para poder aplicar este tipo de algoritmos es necesario que una solución a un problema se pueda representar como un camino en un grafo. Este grafo lo representaremos como $G = (C, L)$, donde C es un conjunto de nodos y L es un conjunto de arcos que conectan los nodos de C. A este grafo G que representa el problema lo llamaremos mapa.

En cada uno de los arcos (r, s) de esta estructura de grafo se almacenará la siguiente información:

- *Información heurística* (τ_{rs}): que depende exclusivamente del problema que queramos resolver. Normalmente se calcula antes de comenzar la ejecución del algoritmo. Almacena información sobre la calidad el arco.

- *Información memorística* (η_{rs}): esta información se va modificando a lo largo de la ejecución del algoritmo y depende del número de hormigas que recorrieron cada arco en el pasado y de la bondad de las soluciones que generaron.

El funcionamiento básico de un algoritmo de OCH es el siguiente:

1. Se inicializa la información heurística del grafo.
2. Posicionamos a una población de m hormigas en el grafo.
3. Cada una de las m hormigas va construyendo progresivamente - según una regla de transición de estados que depende de la información existente - distintos recorridos por el grafo.
4. Se evalúan las soluciones obtenidas y, opcionalmente, se aplica una técnica de búsqueda local.
5. Actualizamos la información memorística.
6. Si no se da una condición de finalización, volvemos al punto 2.

Existen elementos que son propios del problema como:

- Búsqueda local.
- La inicialización de la información heurística.
- La inicialización de las hormigas.
- La obtención de la lista de nodos alcanzables.
- La evaluación de las hormigas.

y otros que dependen en exclusiva de la implementación concreta del algoritmo de OCH:

- Qué regla de transición de estados se utiliza.
- Cómo se actualiza la información memorística.

3.2. Modelos SH y SCH

En la literatura [13, 14, 15, 16, 26] existen distintas propuestas para la regla de transición de estados y varias formas de actualizar la información memorística.

A continuación se describen dos modelos caraterísticos de la OCH, que difieren tanto en la actualización de la feromona como en la regla de transición de estados. Estos son el SH y el SCH.

Sistema de Hormigas

El SH fue la primera propuesta de algoritmo OCH que se realizó [19].

Aunque existen distintas variantes de SH, todas comparten las siguientes características:

1. Todos los arcos del mapa tienen asociados una información de encaminamiento:

$$b_{rs} = \frac{[\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}{\sum_{u \in N(r)} [\tau_{ru}]^{\alpha} \cdot [\eta_{ru}]^{\beta}} \; \forall s \in N(r) \tag{17}$$

   donde $N(r)$ es el conjunto de nodos vecinos de r, α y β son constantes que ponderan la importancia que se da a la información memorística y heurística respectivamente contenida en el arco.

2. A cada uno de los posibles movimientos que una hormiga k puede realizar en un instante, se le asocia una probabilidad de selección:

$$p_{rs}^{k} = \frac{b_{rs}}{\sum_{u \in N(r)} b_{ru}}. \tag{18}$$

   Una vez calculadas estas probabilidades, la regla de transición selecciona una opción aleatoriamente.

3. Cuando cada hormiga ha generado una solución al problema, todas contribuyen a la de actualización de la feromona. El nuevo valor de la feromona para cada arco (r, s) es:

$$\tau_{rs} \leftarrow (1 - \rho) \cdot \tau_{rs} + \sum_{k=1}^{m} \triangle \tau_{rs}^{k} \tag{19}$$

   donde

$$\triangle \tau_{rs}^{k} = \begin{cases} f(C(S_k)), & si\ (r,s) \in S_k \\ 0 & otro\ caso \end{cases} \tag{20}$$

   y $f(C(S_k))$ es la cantidad de feromona depositada por la hormiga k, la cual depende de la calidad de la solución generada por la hormiga, $C(S_k)$. A esta forma de actualizar la feromona, después de haber obtenido soluciones al problema, se le denomina actualización en línea a posteriori o global.

Sistema de Colonias de Hormigas

El SCH [18] es una propuesta que difiere del SH en tres aspectos principalmente:

1. Se aplica la regla de transición proporcional pseudo-aleatoria. La probabilidad asociada

a cada nodo s incluido en la lista de nodos alcanzables desde r y no visitados por la hormiga k, $J_k(r)$, viene dada por:

Si $q \leq q_0$

$$p_{rs}^k = \begin{cases} 1, & si\ s = arg\,max_{u \in J_k(r)} b_{rs} \\ 0, & en\ otro\ caso \end{cases} \tag{21}$$

si no $(q > q_0)$

$$p_{rs}^k = \frac{b_{rs}}{\sum_{u \in J_k(r)} b_{ru}} \tag{22}$$

donde $q_0 \in [0,1]$ es un parámetro que especifica la relación exploración-explotación controlada por la regla y q un número aleatorio en [0,1].

Obsérvese que lo que se obtiene después de aplicar esta regla de actualización es que los arcos que pertenecen a la solución generada por la mejor hormiga reciben un aporte proporcional a la bondad de la solución, mientras que el resto no reciben ninguna aportación (su nivel de feromona decrece a causa de la evaporación).

2. Actualización global considerando únicamente la hormiga que generó la mejor solución hasta el momento, S_{mg}:

$$\tau_{rs} \leftarrow (1-\rho) \cdot \tau_{rs} + \rho \cdot f\left(C\left(S_{mg}\right)\right) \tag{23}$$

3. En este algoritmo también se aplica una actualización local o en línea paso a paso de la feromona. Cada vez que una hormiga recorre una transición (r,s) mientras construye su solución, se aplica la regla:

$$\tau_{rs} \leftarrow (1-\varphi) \cdot \tau_{rs} + \varphi \cdot \tau_0 \tag{24}$$

donde φ es un parámetro con características similares a ρ en la ecuación de actualización global y τ_0 es la cantidad de feromona inicial que permite diversificar las soluciones generadas por las hormigas en la iteración actual.

4. APLICACIÓN DE OCH AL MODELADO BORROSO

Para aplicar la metaheurística OCH al ajuste de los parámetros característicos de un modelo borroso TSK se ha de poder representar el problema mediante un grafo $G = (C, L)$. En la representación elegida de los parámetros de un modelo TSK mediante un grafo G, el conjunto de componentes C representará el conjunto de valores que pueden tomar los parámetros, y el conjunto de arcos L conectará cada subconjunto de nodos que representan valores de un parámetro determinado con otro subconjunto de valores de otro parámetro del modelo TSK. Así, el ajuste de los parámetros del modelo TSK puede ser interpretado como el recorrido de una hormiga por el grafo guiada por la feromona, donde cada nodo recorrido indica la elección de un valor para cada uno de los parámetros del modelo TSK. Adicionalmente, se pueden establecer restricciones en el recorrido de la hormiga con el fin de poder asegurar que el recorrido se corresponde con una solución correcta. En el caso del ajuste del modelo borroso TSK, estas restricciones, serían aquellas que impongan que no se pueden recorrer dos nodos que representen valores para un mismo parámetro, esto es, que a cada parámetro se le deberá asignar un sólo valor.

4.1. Representación del problema

Para una aplicación práctica de la metaheurística OCH al problema del ajuste de los parámetros de un modelo borroso TSK, es conveniente implementar el grafo G como un grafo dirigido organizado por capas. A la capa i-ésima la denotaremos por c_i. Cada capa c_i se asocia con un parámetro θ_i $(i = 1, \ldots, N)$ a ajustar del modelo borroso TSK. Cada capa c_i estará compuesta por un conjunto de nodos λ_{ij} ,con $j = 1, \ldots, V$, siendo V el número de nodos de la capa i-ésima; cada nodo λ_{ij} representa un valor asignable al parámetro θ_i. Un nodo λ_{ij} de la capa c_i estará conectado con todos los nodos de la capa c_{i+1} (figura 1).

En este tipo de representación una cuestión clave es la asignación de un valor $v_{ij} \in \mathbb{R}$ a cada nodo λ_{ij}; por tanto, si quisieramos representar todos los posibles valores v_{ij}, sería necesario tener en cada capa c_i un número infinito de nodos. Una de las posibles formas de obtener un número finito de nodos estaría basada en la discretización del espacio de búsqueda, dividiendo el rango de búsqueda del valor v_{ij} en I intervalos, obteniendo un valor v_{ij} representativo de cada intervalo.

Sea $[\psi_i, \vartheta_i]$ el rango de búsqueda de un valor para v_{ij}. Si dividimos el rango de búsqueda en I intervalos es posible definir un parámetro h_i como la precisión de búsqueda dentro del intervalo $[\psi_i, \vartheta_i]$ como:

$$h_i = \left| \frac{\vartheta_i - \psi_i}{I} \right| \tag{25}$$

si tomamos el valor medio como el valor representativo de cada intervalo, entonces podemos tomar el valor v_{ij} para el nodo λ_{ij} como:

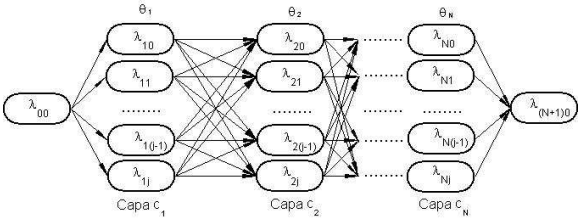

Figura 1: Representación del problema de ajuste de parámetros para el modelo TSK.

$$v_{ij} = \frac{(\psi_i + j \cdot h_i) + (\psi_i + ((j+1) \cdot h_i))}{2}. \quad (26)$$

Con el fin de facilitar los recorridos de las hormigas, definimos un nodo inicial y otro final de los recorridos, de tal forma que el nodo inicial λ_{00} se conecta a través de un arco dirigido con todos los nodos λ_{1j} de la capa c_1, y todos los nodos λ_{Nj} de la capa c_N se conectan con el nodo final $\lambda_{(N+1)0}$. La figura 1 representa un grafo G genérico para esta representación del problema. Notar que el valor j dependerá en cada capa del valor de h_i.

4.2. Construcción de la solución

Una solución S al problema del ajuste de los parámetros del modelo borroso TSK viene determinada en un algoritmo OCH mediante un recorrido de una hormiga k sobre el grafo G. Este recorrido comprende un nodo de cada capa. Las soluciones dadas por las hormigas tendrán la forma:

$$S_k = \{\lambda_{00}, \lambda_{1j}, \lambda_{2j}, \ldots, \lambda_{Nj}, \lambda_{(N+1)0}\} \quad (27)$$

donde j indica que el número de nodos para la capa c_i depende de la división en intervalos realizada.

En cada paso de construcción de la solución, el paso de la hormiga k por un nodo λ_{ij} significa que el parámetro θ_i queda ajustado con el valor v_{ij}. Por tanto, si tenemos una solución de la forma expresada en (27) significa que:

$$\{\theta_1, \ldots, \theta_i, \ldots, \theta_N\} = \{v_{1j}, \ldots, v_{ij}, \ldots, v_{Nj}\} \quad (28)$$

En cada paso, la elección del nodo $\lambda_{(i+1)j}$ después del paso por el nodo λ_{ij} se realiza en función de la expresión (18) y la vecindad $N^k(ij)$, que representa el conjunto de nodos a los que puede desplazarse la hormiga k desde el nodo λ_{ij}. En nuestro caso, la vecindad son todos los nodos de la capa $i+1$:

$$N^k(ij) = \{\lambda_{(i+1)j} | \, \lambda_{(i+1)j} \in c_{i+1}\} \quad (29)$$

4.3. Evaluación y actualización

Una vez obtenida una solución S se procede a evaluarla. La evaluación de la solución S es equivalente a evaluar la bondad de los valores $\{v_{1j}, \ldots, v_{ij}, \ldots, v_{Nj}\}$ asignados a los parámetros $\{\theta_1, \ldots, \theta_i, \ldots, \theta_N\}$. Esta evaluación se puede realizar mediante la expresión (8). En cada uno de los arcos que forman parte de la solución S se actualiza la feromona que aporta la hormiga k al camino recorrido como:

$$f(t) = \frac{1}{J(t)} = \frac{2}{\sum_{i=0}^{N}(y_i - \hat{y}_i)} \quad (30)$$

donde $J(t)$ es la correspondiente a la expresión (8).

4.4. Búsqueda local

En la bibliografía [23] se proponen algoritmos para mejorar las soluciones de las hormigas mediante búsqueda local. Estos algoritmos híbridos combinan la solución construida por las hormigas con un algoritmo de búsqueda local. Mas aún, se ha comprobado que búsquedas locales a partir de una solución incial generada de forma aleatoria proporcionan una considerable aproximación a la solución óptima. También se ha demostrado que la combinación de algoritmos probabilísticos con búsqueda local proporcionan importantes mejoras en las soluciones. Los algoritmos OCH actúan como las heurísticas de construcción adaptativas en el sentido de que la colonia de hormigas modifica la solución asignando mayor cantidad de feromona en aquellos recorridos que contienen una mejor solución. Durante la construcción de la solución, las hormigas se decantan por aquellos arcos que contienen una cantidad alta de feromona, y mediante la combinación de estos arcos, generan soluciones prometedoras para los algoritmos de búsqueda local. Como ventaja adicional del uso de algoritmos OCH es que, mediante la generación de buenas soluciones iniciales, la posterior búsqueda local necesita de menos iteraciones para llegar a un óptimo local. Así, en nuestro caso, el ajuste de parámetros con el algoritmo OCH se mejora con la aplicación posterior del conocido algoritmo de gradiente descendente.

5. Ejemplos

A continuación mostraremos los ejemplos que hemos usado para comprobar el ajuste, realizado, con un algoritmo OCH, de los parámetros de un sistema borroso. Partiremos de expresiones conocidas de las que se extraerán un conjunto de datos de entrada/salida para el entrenamiento del modelo borroso. Este conjunto de datos obtenidos supondremos que representan un sistema desconocido que queremos aproximar mediante un modelo borroso. Para poder evaluar los resultados obtenidos, se compararán las soluciones con las proporcionadas por un método de modelización borrosa ampliamente conocido como es el método del gradiente descendente.

5.1. Ejemplo 1

Dado un sistema definido por la expresión:

$$f(x) = x^2 + 3x + 6 \tag{31}$$

se obtendrá un conjunto de 100 valores aleatorios en el rango de entrada $[0, 1]$ para el conjunto de entrenamiento. Con el fin de poder comprobar la bondad del ajuste realizado se obtienen otros 100 valores que nos servirán como conjunto de test. La base de reglas utilizada para modelizar el sistema contiene 3 reglas con 3 funciones de pertenencia Gaussianas asignadas a la variable de entrada:

$$R^1: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_1)^2}{\beta_1^2}\right) \quad Entonces\, y_1 = \theta_{10} + \theta_{11}x$$

$$R^2: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_2)^2}{\beta_2^2}\right) \quad Entonces\, y_2 = \theta_{20} + \theta_{21}x$$

$$R^3: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_3)^2}{\beta_3^2}\right) \quad Entonces\, y_3 = \theta_{30} + \theta_{31}x$$

El número total de parámetros a ajustar es de 12, 6 parámetros en la premisa (no lineales) $\{\gamma_1, \gamma_2, \gamma_3, \beta_1, \beta_2, \beta_3\}$ y 6 parámetros en el consecuente (lineales) $\{\theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}\}$.

5.2. Ejemplo 2

Dado un sistema definido por la expresión:

$$f(x) = sen(x) + |x| \tag{32}$$

se obtendrá un conjunto de 100 valores aleatorios en el rango de entrada $[0, \pi]$ para el conjunto de entrenamiento, e igual número para el conjunto de test. La base de reglas utilizada para modelizar el sistema contiene 5 reglas con 5 funciones de pertenencia Gaussianas:

$$R^1: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_1)^2}{\beta_1^2}\right) \quad Entonces\, y_1 = \theta_{10} + \theta_{11}x$$

$$R^2: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_2)^2}{\beta_2^2}\right) \quad Entonces\, y_2 = \theta_{20} + \theta_{21}x$$

$$\text{factor}\ R^3: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_3)^2}{\beta_3^2}\right) \quad Entonces\, y_3 = \theta_{30} + \theta_{31}x$$

$$R^4: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_4)^2}{\beta_4^2}\right) \quad Entonces\, y_4 = \theta_{40} + \theta_{41}x$$

$$R^5: \quad Si\, x\, es\, exp\left(-\frac{(x-\gamma_5)^2}{\beta_5^2}\right) \quad Entonces\, y_5 = \theta_{50} + \theta_{51}x$$

El número total de parámetros a ajustar es de 20, 10 parámetros en la premisa (no lineales) $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5\}$ y 10 parámetros en el consecuente (lineales) $\{\theta_{10}, \theta_{11}, \theta_{20}, \theta_{21}, \theta_{30}, \theta_{31}, \theta_{40}, \theta_{41}, \theta_{50}, \theta_{51}\}$.

6. Experimentación y resultados

A continuación se muestra la experimentación realizada con los ejemplos expuestos en la sección anterior. Se ha utilizado el algoritmo SH. Los parámetros elegidos para la experimentación son aquellos propuestos en la literatura que han dado buenos resultados para otro tipo de problemas como el QAP (*Quadratic Assignment Problem*) [16], estos son:

Cuadro 1: Parámetros de la experimentación.

Parámetro	Valor
Hormigas (k)	5
Iteraciones	1000
ρ	0.8
α	2
β	0

En el caso del ejemplo 1, el rango de búsqueda para los parámetros de la premisa es [0,009861, 0,9883] y para los parámetros del consecuente [6,03, 9,942]. La precisión h utilizada es de 0,1. El número de iteraciones utilizadas en las pruebas realizadas ha sido de 1000, pero cabe decir que con 20 iteraciones se alcanzaban las soluciones iniciales proporcionadas por el algoritmo SH para la posterior optimización local con gradiente

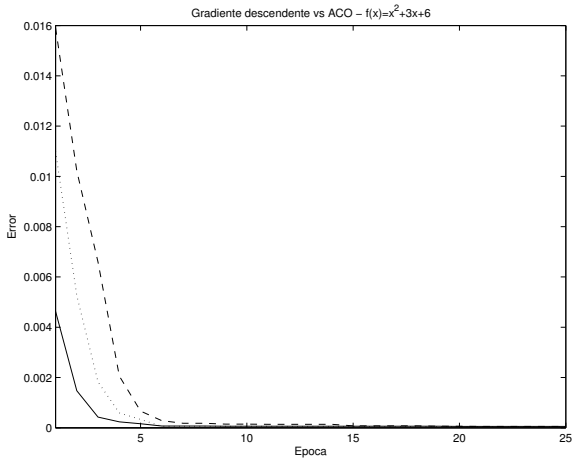

Figura 2: Error para $f(x) = x^2 + 3x + 6$ con gradiente descendente (línea continua), el peor caso con OCH (línea discontinua) y mejor caso con OCH (línea punteada).

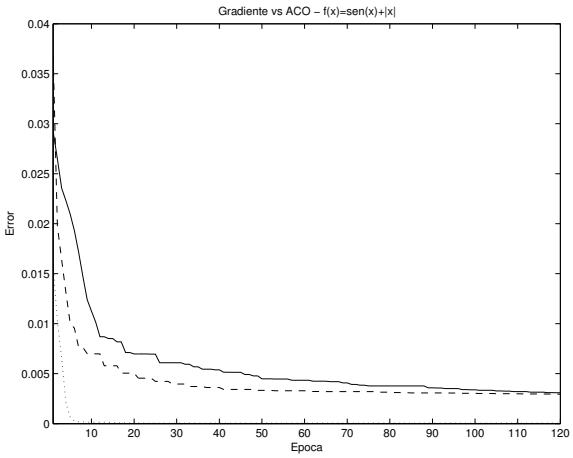

Figura 3: Error para $f(x) = sen(x) + |x|$ con gradiente descendente (línea continua), el peor caso con OCH (línea discontinua) y mejor caso OCH (línea punteada).

descendente. A continuación se muestran los resultados obtenidos para 10 ejecuciones del algoritmo aplicando entrenamiento mediante gradiente descendente durante 25 épocas. En las siguientes tablas se muestran la ejecución con gradiente descendente, la mejor ejecución con SH, la peor ejecución con SH y la media de las 10 ejecuciones:

Cuadro 2: Resultados de la experimentación para el ejemplo 1.

	Gradiente	OCH Mejor
Entrenamiento	$5{,}62478 \cdot 10^{-5}$	$4{,}7857 \cdot 10^{-5}$
Test	$6{,}59871 \cdot 10^{-5}$	$6{,}14781 \cdot 10^{-5}$

	OCH Peor	OCH Media
Entrenamiento	$5{,}44281 \cdot 10^{-5}$	$5{,}1247 \cdot 10^{-5}$
Test	$6{,}3341 \cdot 10^{-5}$	$5{,}94781 \cdot 10^{-5}$

En la figura 2 se muestra de forma gráfica la evolución del error para el ajuste mediante gradiente descendente y SH en el mejor y peor caso. Como se puede observar, el error se reduce aplicando SH pero la convergencia se produce de una forma más lenta.

En el caso del ejemplo 2, el rango de búsqueda para los parametros de la premisa es $[-3{,}2, 3{,}2]$ y para los parámetros del consecuente $[-0{,}3, 2{,}2]$. La precisión h utilizada es de 0,1. El número de iteraciones utilizadas en las pruebas realizadas ha sido de 1000, aunque con 100 iteraciones se alcanzaban las soluciones iniciales proporcionadas por el algoritmo SH para la posterior optimización local con gradiente descendente. A continuación se muestran los resultados obtenidos para 10 ejecuciones del algoritmo aplicando entrenamiento mediante gradiente descendente durante 120 épocas. En las siguientes tablas se muestran la ejecución con gradiente descendente, la mejor ejecución con SH, la peor ejecución con SH y la media de las 10 ejecuciones:

Cuadro 3: Resultados de la experimentación para el ejemplo 1.

	Gradiente	OCH Mejor
Entrenamiento	0,0039487	0,00054428
Test	0,0061236	0,00061341

	OCH Peor	OCH Media
Entrenamiento	0,0037002	0,002723
Test	0,0059841	0,0043998

En este caso, para una función más compleja, podemos observar como tanto la convergencia como la solución es mejor, muy buena en el mejor de los casos, y aún en el peor de los casos se mejora la proporcionada por la aplicación del gradiente descendente (figura 3).

7. Conclusiones y trabajos futuros

Se ha demostrado que se pueden aplicar los algoritmos SH al ajuste de los parámetros de un modelo borroso mediante la discretización del espacio de búsqueda de los valores de los parámetros, sin que esta discretización influya de forma negativa sobre los resultados. Mas aún, los resultados que se

obtienen mejoran sensiblemente los obtenidos con el método tradicional del gradiente descendente.

Se ha utilizado una versión simple de algoritmo SH para el problema del modelado, en futuros trabajos se utilizarán diferentes algoritmos OCH que han demostrado su validez en la aplicación a otros problemas.

La aplicación de la búsqueda local mediante gradiente descendente se ha aplicado después de obtener una solución preliminar mediante OCH, creemos que se pueden obtener mejores resultados si la aplicación de la búsqueda local se realiza en línea con la aplicación OCH.

Referencias

[1] J. M. Andújar. *Sistemas Borrosos Multivariables. Modelización Neuro-Borrosa, Control y Análisis de Estabilidad.* Tesis. Universidad de Huelva. España, 2000.

[2] J. M. Andújar, J. M. Bravo. *Multivariable fuzzy control aplied to the physical-chemical treatment facility of a cellulose factory,* Fuzzy Sets and Systems, disponible en linea desde el 22 de Abril del 2004 en http://dx.doi.org/10.1016/j.fss.2004.03.023.

[3] J. M. Andújar, J. M. Bravo, A. Peregrín. *Stability analysis ans synthesis of multivariable fuzzy systems using interval arithmetic.* Fuzzy Sets and Systems, disponible en linea desde el 24 de Febrero del 2004 en http://dx.doi.org/10.1016/j.fss.2004.01.008.

[4] R. Babuska, H. B. Verbruggen. *A new identification method for linguistic fuzzy models*, Proc. FUZZ-IEEE/IFES'95, Yokohama, Japan, pp. 905-912, 1995.

[5] R. Babuska. *Fuzzy modeling a controll engineering perspective*, Proc. FUZZ-IEEE/IFES'95, Yokohama, Japan, pp. 1897-1902, 1995.

[6] R. Babuska, H. B. Verbruggen. *Identification of composite linear models via fuzzy clustering*, Proc. 3^{rd} European Control Conference, Rome, Italy, pp. 1207-1212, 1995.

[7] P. J. Bentley. *Digital Biology.* Healine 2001.

[8] J.L. Bently. *Fast algorithms for geometric travelling salesman problem. ORSA Journal on Computing*, 4(4), pp. 387-411, 1992.

[9] C. Blum, A. Roli. *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison.* Technical report 2001-13 IRIDIA , 2001

[10] E. Bonabeon, M. Dorigo, G. Theraulaz. *Swarm Intelligence. From Natural to Artificial Systems*, Oxford University Press, 1999.

[11] M. Brown, C. Harris. *Neurofuzzy adaptative modelling and control*, Prentice Hall, 1994.

[12] S. L. Chiu. *Fuzzy model identification based on cluster estimation*, Journal of Intelligent and Fuzzy Systems, 2(3), 1994.

[13] O. Cordón, F, Herrera, L. Moreno. *Integración de Conceptos de Computación Evolutiva en un Nuevo Modelo de Colonias de Hormigas*, en Actas de la CAEPIA'99. Seminario Especializado sobre Computación Evolutiva, Vol.II pp. 98-105, 1999.

[14] O. Cordón, F. Herrera, T. Stutzle. *A Review on then Ant Colony Optimization Metaheuristic: Basic, Models and New Trends*, Mathwork & Soft Computing, 9, 2002.

[15] O. Cordón, I. Fernández de Viana, F. Herrera, L. Moreno. *A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System. Proc. of ANTS'2000*, Ant Colonies to Artifitial Ants: Second International Workshop on Ant Algorithms, Brussels, Belgium, September 7-9, pp. 22-29, 2000.

[16] D. Corne, M. Dorigo, F. Glover (Ed.). *New Ideas in Optimization*, McGraw-Hill, 1999.

[17] M. Dorigo, G. Di Caro. *The Ant Colony Optimization Meta-heuristic*, D. Corne, M. Dorigo, F. Glover (eds), New Ideas in Optimization, pp. 11-32, McGraw-Hill, 1999.

[18] M.Dorigo, L. Gambardella. *Ant Colony System: A Cooperative Learning Approach to the Travelling Saleman Problem*, IEEE Transactions on Evolutionary Computation 1(1), pp. 53-66, 1997.

[19] M. Dorigo, V. Maniezzo, A. Colorni. *The Ant System: Optimization by Colony of Cooperating Agents*, IEEE Trans. on Systems, Man, and Cybernetics, Part B, 26:1 pp. 29-41, 1996.

[20] F. Hoffmann. *Incremental tuning of fuzzy controllers by means of an evolution strategy*, GP'98 Conf., Madison, Wisconsin, 1998.

[21] J-S. R. Jang, C-T. Sun, E. Mizutani. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*, Prentice-Hall, 1997.

[22] J.-S.R. Jang, C-T Sun. *Neuro-fuzzy modeling and control*, Proceedings of the IEEE , 83(3), pp. 378-406, March 1995.

[23] D. S. Johnson, L.A. McGeoch. *The Traveling Salesman Problem: A Case Study in Local Optimization*, E.H.L. Arts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, pp. 215-310. John Wiley & Sons, 1997.

[24] T. H. Nguyen, M. Sugeno, R. Tong, R. R. Yager. *Theoretical aspects of fuzzy control*, John Wiley Sons, 1995.

[25] A. Piegat. *Fuzzy Modeling and Control*, Physica-Verlag, 2001.

[26] T. Stützle, Holger H. Hoos. *MAX-MIN Ant System*, Future Generation Computer Systems Journal 16(8), pp. 889-914, 2000.

[27] M. Sugeno, G. T. Kang. *Structure identification of fuzzy model*, Fuzzy Sets and Systems, 28, pp. 15-33, 1988.

[28] T. Takagi, M. Sugeno. *Fuzzy identification of systems and its applications to modeling and control*, IEEE Transactions on Systems, Man, and Cybernetics, 15:116-132, 1985.

[29] R. Yager, D. Filev. *Essentials of fuzzy modeling and control*, John Wiley and Sons, 1994.

[30] J. Yen, R. Langari. *Fuzzy logic: intelligence, control, and information*, Prentice-Hall, 1999.