MEJORAS EN EL DISEÑO MULTIOBJETIVO DE REDES DE FUNCIONES DE BASE RADIAL

Pedro L. López[1] **Antonio J. Rivera**[1] **Cristóbal J. Carmona**[1] **María Dolores Pérez-Godoy**[1]

[1] Departamento de Informática. Universidad de Jaén {plopez, arivera, ccarmona, lperez}@ujaen.es

Resumen

En este artículo se presenta un nuevo elemento poblacional a considerar en el diseño de algoritmos evolutivos multiobjetivo para la optimización de Redes de Funciones de Base Radial. Concretamente, se divide la población en subpoblaciones virtuales, donde cada subpoblación está compuesta por individuos o redes con el mismo número de neuronas. El objetivo será mantener el mejor individuo (individuo élite) de cada subpoblación entre generaciones, consiguiendo aumentar la diversidad de la población. Este elemento se implementa en el algoritmo EMORBFN, ya presentado por los autores, y se aplica a la tarea de clasificación. En los resultados se puede observar cómo la introducción de este nuevo elemento mejora el comportamiento general del algoritmo.

Palabras Clave: Diseño Multiobjetivo, Redes de Funciones de Base Radial, Elitismo.

1 INTRODUCCION

El diseño de cualquier modelo o componente en cualquier área de ciencia e ingeniería se ha convertido en un problema donde se trata de buscar una solución de compromiso entre todos los objetivos que se intentan cumplir. En general, estos objetivos entran en conflicto y desaparece el concepto de diseño óptimo convencional. Estos problemas, denominados problemas de optimización multiobjetivo, son aquellos en los que no existe una única solución sino un conjunto de soluciones, cada una de ellas con sus ventajas frente a las demás, pero donde ninguna es mejor en todos los aspectos. A partir de este abanico de modelos se elige la solución o soluciones finales en función del conocimiento experto que se maneje.

En nuestro caso, los modelos a diseñar son Redes de Funciones de Base Radial (RBFNs), uno de los paradigmas más importantes dentro del campo de las Redes Neuronales. Este modelo de cómputo ha demostrado su solvencia a la hora de abordar problemas de aproximación de funciones [9], clasificación [4] o predicción de series temporales [21]. Las Funciones de Base Radial (RBFs) se usaron inicialmente en interpolación numérica y aproximación funcional [18]. A finales de los 80 tienen lugar las primeras investigaciones de redes neuronales basadas en RBFs [3][15].

Son muchas las características que despiertan el interés por este tipo de redes, entre las cuales cabe destacar: su topología simple con tan solo una capa oculta; su capacidad de ser un aproximador universal [16]; el ser de los pocos modelos de red neuronal interpretable, ya que pueden extraerse reglas difusas a partir de ellos [13], etc.

El diseño de una RBFN se puede abordar desde el punto de vista de la computación evolutiva [1][4]. En la mayoría de estas propuestas, un individuo representa una RBFN completa, que evoluciona intentando siempre optimizar su precisión.

Contemplar sólo el objetivo de la precisión a la hora de optimizar RBFNs, como en la computación evolutiva tradicional, puede conducir a obtener redes con un número elevado de RBFs. Esto se debe a que es más fácil reducir el error con RBFNs que contienen muchas RBFs que con RBFNs que contienen pocas RBFs. Para solventar este problema aparece la Optimización Multiobjetivo Evolutiva (EMO) [5]. En nuestro caso, y como suele ser habitual, además de optimizar la precisión de la red, se optimizará también la complejidad o número de neuronas de ésta.

Dentro de la EMO no son muchos los trabajos que se pueden encontrar en la literatura especializada en el campo del diseño multiobjetivo evolutivo de RBFNs. Algunos de estos trabajos se enumeran en la siguiente sección. Esta carencia de trabajos está relacionada con la

dificultad que encuentran este tipo de algoritmos a la hora de lograr diversidad en la población.

Para solucionar este problema se propone introducir un elitismo especial en la población a nivel de subpoblaciones, estando cada subpoblación definida por el número de RBFs de la red. Esto implica que todas las RBFNs con un número igual de RBFs estarán en la misma subpoblación y que al mejor individuo de cada una de esas subpoblaciones virtuales se le mantendrá entre generaciones.

Concretamente, en este trabajo se ha implementado una mejora para EMORBFN [14], un algoritmo multiobjetivo para el diseño de RBFNs, en la cual se mantienen estos individuos élite por subpoblación, aportando diversidad y una mejora de los resultados finales.

En la Sección 2 se describen las RBFNs y su optimización evolutiva. En la Sección 3 se detalla el funcionamiento del algoritmo EMORBFN. En la Sección 4 se muestran y analizan los resultados obtenidos. Para terminar, en la Sección 5 se exponen las conclusiones del trabajo.

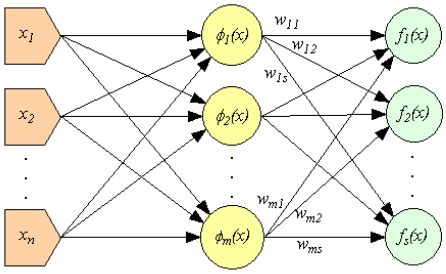

Figura 1: Topología de una RBFN

2 RBFNs Y SU DISEÑO

Una Red de Funciones de Base Radial es un tipo de red neuronal hacia delante con tres capas: la capa de entrada con *n* nodos, una capa oculta con *m* neuronas o RBFs, y una capa de salida con uno o varios nodos (Figura 1). Las *m* neuronas/RBFs de la capa oculta ofrecen una activación simétrica radial $\phi_i : R^n \rightarrow R$, que puede tomar diferentes formas. La más común es la función gausiana, que viene dada por la expresión $\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-\vec{c}_i\|/d_i)^2})$, donde $\vec{c}_i \in R^n$ es el centro de la función base ϕ_i, $d_i \in R$ es el radio y como $\|\ \|$ suele utilizarse la norma euclídea en R^n. Los nodos de salida implementan la siguiente ecuación:

$$f_j(\vec{x}) = \sum_{i=1}^{m} w_{ij}\phi_i(\vec{x}) \qquad (1)$$

En un escenario de clasificación, la RBFN debe establecer una relación entre un espacio de entrada X^n hacia un conjunto finito de clases *Y* con *k* clases. De esta manera un típico conjunto de entrenamiento *S* sería:

$$S = \left\{(\vec{x}_u, y_u) \mid x_u \in X^n,\ y \in Y,\quad u = 1,..., p\right\} \qquad (2)$$

donde $\vec{x}_u$ es el vector de características e y_u es la clase a la que pertenece. Normalmente, en un problema de clasificación, el número de salidas de la RBFN se corresponde con el número de clases (*k*). Para entrenar la RBFN, la pertenencia a la clase y_u se codifica en un vector binario $\vec{z}_u \in \{0,1\}^k$ a través de la relación $\vec{z}_u^{\,i} = 1$ sii $y_u = i$, y $\vec{z}_u^{\,i} = 0$ *en otro caso*. La clase obtenida por la red ante una determinada entrada será la salida con un mayor grado de activación.

En cualquier caso, el objetivo a la hora de diseñar una RBFN es determinar el centro y el radio que caracteriza cada RBF así como su peso para cada salida de la red. Las técnicas empleadas en el diseño de RBFNs son muy diversas.

El algoritmo típico de diseño de RBFNs tiene dos etapas. En la primera etapa se determinan los centros y los radios de las RBFs, mientras que en la segunda etapa se calculan sus pesos. Para determinar los centros y los radios se pueden emplear técnicas de clustering [17]. En la segunda etapa, para calcular los pesos, se pueden utilizar algoritmos como el LMS [22], SVD [8], etc.

Hoy en día la computación evolutiva es uno de los paradigmas más importantes en el diseño de RBFNs [4][11]. Estos algoritmos suelen utilizar una codificación de tipo Pittsburg, en la que un individuo representa una red completa. La mayoría de estos algoritmos sólo optimizan la precisión de los individuos sin tener en cuenta la complejidad de las redes obtenidas. Como se ha comentado, contemplar un único objetivo puede conducir a obtener redes con un alto número de neuronas, ya que es más fácil reducir el error con un mayor número de neuronas que con un menor número de éstas. Es por esto que hay que tener en cuenta los paradigmas que nos proporciona la EMO para abordar este tipo de problemas.

Brevemente definiremos los conceptos de dominancia y de frente de Pareto dentro de la optimización multiobjetivo. Así se dice que un individuo A domina a otro individuo B si A no es peor que B en ninguno de los objetivos y A es mejor que B en algún objetivo. Dado un espacio de búsqueda el frente de Pareto óptimo se define como el conjunto de individuos no dominados de ese espacio de búsqueda.

Los Algoritmos Evolutivos Multiobjetivo (MOEAs) han sido usados en el diseño de RBFNs. Por ejemplo los

conocidos Algoritmos Genéticos Multiobjetivo (MOGAs) se han utilizado en [9][20] para la estimación o aproximación de funciones. En [19] se hibrida NSGA-II [6] con "rough-sets" para alcanzar un algoritmo multiobjetivo más eficiente. [12] usa "surrogates" distribuidos espacialmente para implementar las RBFNs. En [10] se presenta una aplicación que hace estimaciones en la reducción de un mineral. Hay también MOEAs específicos: en [23] se utiliza la técnica HRDGA (Hierarchical Rank Density Genetic Algorithm) para diseñar la topología y los parámetros de la red.

3 DESCRIPCIÓN DE EMORBFN

En este trabajo se presenta EMORBFN, un Algoritmo Genético Multiobjetivo para el diseño de RBFNs, donde los objetivos a optimizar son la precisión del modelo (maximizar el porcentaje de aciertos) y su complejidad (minimizar el número de RBFs). La principal característica a destacar de EMORBFN es que va a definir subpoblaciones virtuales de individuos con igual número de neuronas y que conservará entre generaciones al mejor individuo de cada subpoblación.

La justificacion de esta propuesta se encuentra en que, tras analizar la evolución de una población compuesta por individuos/RBFNs en el marco de un algoritmo evolutivo multiobjetivo, se puede observar que los individuos élite de las subpoblaciones de gran tamaño pueden ser dominados por individuos con menos neuronas, desapareciendo así de la población. Esto se debe a dos causas:

- Durante las primeras iteraciones del algoritmo, los parámetros de las redes no han sido ajustados aún y es de esperar que tengan un porcentaje de clasificación similar. Esto provoca que redes con pocas RBFs dominen rápidamente a los individuos de mayor tamaño, haciendo que estos últimos sean descartados durante el reemplazamiento de la población.
- Por otro lado, las RBFNs con un mayor número de neuronas son más difíciles de aprender, ya que contienen más parámetros a determinar. Por ello, se necesitan más iteraciones para obtener redes que sean competitivas en cuanto a precisión.

Estos dos factores implican una fuerte pérdida de diversidad en las primeras etapas del algoritmo, dificultando la obtención de un conjunto de soluciones adecuadamente distribuido a lo largo del frente de Pareto. El operador de elitismo garantiza que el individuo de cada número de RBFs con mayor precisión pase a formar parte de la población en la siguiente generación. Con ello se pretende evitar la mencionada pérdida de diversidad, dando lugar a un mejor comportamiento del método.

Concretamente, EMORBFN determina todos los parámetros del modelo: número de RBFs y centro y radio de cada una de ellas. Se utiliza codificación real y un enfoque Pittsburgh, donde cada individuo representa una RBFN al completo. El problema multiobjetivo se aborda en el paso 6 mediante un procedimiento de reemplazo basado en el algoritmo NSGA-II [6]. A dicho algoritmo se le añade un operador de elitismo que permite conservar al menos un individuo de cada número de RBFs; así se garantiza la evolución de todas las subpoblaciones. La Figura 2 muestra los pasos básicos del algoritmo, que se detallan a continuación.

1. P_t = Inicialización(Datos)
2. Entrenamiento/Evaluación(P_t)
3. C_t = Cruce(P_t)
4. M_t = Mutación(P_t)
5. Entrenamiento/Evaluación(C_t, M_t)
6. P_{t+1} = Reemplazo+Elitismo(P_t, C_t, M_t)
7. Si no se cumple la condición de parada, ir a 3.

Figura 2: Pasos básicos de EMORBFN

Inicialización. Se generan λ (tamaño de la población) RBFNs de manera que haya la misma cantidad de redes para cada número de RBFs; así, cada subpoblación tiene un tamaño similar. El centro $\vec{c}_i$ de cada RBF se determina asignando un ejemplo de entrenamiento aleatoriamente, teniendo en cuenta que las RBFs de una misma red deben estar igualmente distribuidas por todas las clases. Una vez asignados los centros, el valor del radio d_i es la mitad de la distancia media entre los centros de la red. Los pesos w_{ij} se inicializan a cero.

Entrenamiento/Evaluación. Se utiliza el algoritmo LMS para determinar los pesos w_{ij} de cada red. A continuación se evalúa cada RBFN y se obtiene su error de clasificación.

Cruce. Durante el cruce se genera un porcentaje *pc* de nuevas redes. Para ello se eligen dos RBFNs padre aleatoriamente y se combinan formando un único hijo, cuyo tamaño será un número aleatorio entre el mínimo número de RBFs de los padres y la suma del número de RBFs de los padres. El nuevo hijo se genera insertando RBFs alternativamente de cada padre. Cada vez que se selecciona una RBF se busca en el otro padre si existe una RBF que se solape con la seleccionada. En caso afirmativo se combinan sus valores (centros y radios) usando el cruce BLX-α [7]. En caso contrario, se introduce la RBF seleccionada aplicando una mutación (ver *ModRBF* a continuación).

Mutación. Se han implementado cinco operadores de mutación, distinguiendo entre mutaciones aleatorias y mutaciones con información. Las redes a mutar se eligen aleatoriamente. Los operadores aleatorios son:

- *EliRBFAleatorio*. Elimina un porcentaje *pm* de RBFs elegidas de forma aleatoria.
- *InsRBFAleatorio.* Inserta un porcentaje *pm* de RBFs generadas aleatoriamente.
- *ModRBF*. Modifica un porcentaje *pm* de las RBFs de una red. Para cada RBF a mutar, el centro y el radio se modifican en un porcentaje aleatorio entre 0 y *pr* del valor del radio actual.

Los operadores de mutación con información son:

- *InsRBFInf*. Inserta un porcentaje *pm* de RBFs de forma que no se solapen con el resto de RBFs de la red.
- *EliRBFInf*. Elimina un porcentaje *pm* de las RBFs con menor peso medio de la red.

Reemplazo+Elitismo. Durante esta fase se seleccionan sin reemplazo los λ individuos que formarán la nueva generación (P_{t+1}), considerando las RBFNs de la población actual (P_t) y las generadas mediante cruce (C_t) y mutación (M_t). La implementación se basa en el algoritmo NSGA-II y en los conceptos de dominancia de Pareto. Se construyen los frentes de no-dominancia $F=\{F_1,F_2,\ldots\}$, donde cada frente F_i está formado por individuos no-dominados entre sí y que dominan a todos los individuos de los frentes $F_{k>i}$.

En un primer paso, se recorren los frentes de no-dominancia de menor a mayor hasta que se encuentre un individuo de cada número de RBFs. Dicho individuo es considerado el mejor de su subpoblación, ya que las demás RBFNs del mismo tamaño estarán en frentes dominados por él, por lo que es incluido en la nueva población P_{t+1}. Así se asegura que las subpoblaciones conservan durante toda la evolución al mejor individuo encontrado hasta ese momento (elitismo).

Para completar la nueva población se van insertando los frentes F_1, F_2… hasta conseguir λ (tamaño de la población) individuos. Si al insertar un frente, éste tiene más individuos de los necesarios para completar P_{t+1}, se calcula la distancia de crowding entre los individuos del frente y se insertan en orden de mayor a menor distancia.

A la hora de insertar una RBFN en la nueva población P_{t+1} se implementa un operador que impide la introducción de individuos equivalentes. Consideramos que dos RBFNs son equivalentes cuando, teniendo el mismo tamaño, la distancia entre los centros de las dos RBFs más cercanas de cada red es menor que los radios de las mismas.

4 EXPERIMENTACION Y RESULTADOS

El objetivo de los experimentos llevados a cabo es comprobar la influencia del elitismo en el comportamiento de EMORBFN a la hora de obtener RBFNs de diferentes complejidades con una alta capacidad de clasificación.

Para evaluar el efecto del elitismo se ha experimentado con dos versiones del algoritmo: *EMORBFNBasico* y *EMORBFNElitista*. Ambas versiones son idénticas exceptuando el paso 6 (Figura 2), donde la implementación básica no incluye elitismo y selecciona los individuos atendiendo únicamente al frente de dominancia en el que se encuentran.

Se han seleccionado tres problemas de clasificación estándar (pima, wbcd y wine) obtenidos del UCI Machine Learning Repository [1]. Con el objetivo de acotar el espacio de búsqueda, se ha establecido el tamaño mínimo de una red al número de clases del problema y el tamaño máximo a cuatro veces dicho número de clases. Esta decisión se justifica porque tal y como se demuestra en los resultados, las RBFNs no necesitan un elevado número de neuronas para conseguir una precisión importante. Además permite mantener un número relativamente pequeño de individuos en la población. Para evaluar la precisión del método se ha utilizado la técnica de 10-fold cross validation. Cada experimento se ha repetido 5 veces, sumando 50 ejecuciones total.

La Tabla 1 muestra los parámetros utilizados en la configuración de EMORBFN. Los valores se han elegido entre los habituales para este tipo de algoritmos. Las Tablas 2-4 muestran los frentes obtenidos por ambos algoritmos en cada base de datos. Para cada número de RBFs se muestra la media del porcentaje de clasificación en test del mejor individuo de cada ejecución. Las Figuras 3-6 resumen gráficamente los resultados obtenidos.

Tabla 1. Parámetros de EMORBFN

Parámetro	*Valor*
Número de iteraciones	200
Tamaño de la población (λ)	40
Probabilidad de cruce	0,6
Probabilidad de mutación	0,1
pm	0,5
pr	0,5

Tabla 2. Resultados para el problema Pima.

Nº de RBFs	*% de Clasificación*	
	EMORBFNBasico	*EMORBFNElitista*
2	75,06	74,54
3	74,79	75,64
4	73,52	75,68
5	74,57	75,07
6	73,00	76,05
7	75,44	75,16
8	75,88	77,10

Tabla 3. Resultados para el problema Wbcd.

Nº de RBFs	*% de Clasificación*	
	EMORBFNBasico	*EMORBFNElitista*
2	95,16	96,28
3	96,37	96,79
4	95,84	96,61
5	96,50	97,23
6	96,18	96,30
7	95,84	96,86
8	95,87	96,90

Tabla 4. Resultados para el problema Wine.

Nº de RBFs	*% de Clasificación*	
	EMORBFNBasico	*EMORBFNElitista*
3	89,36	88,51
4	88,75	93,71
5	93,33	96,09
6	94,38	94,58
7	94,67	96,03
8	94,37	95,10
9	94,08	95,79
10	95,37	96,27
11	98,61	96,80
12	95,56	97,22

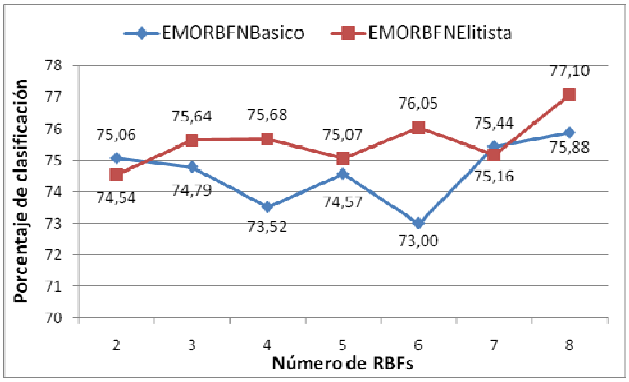

Figura 3. Resultados para el problema Pima.

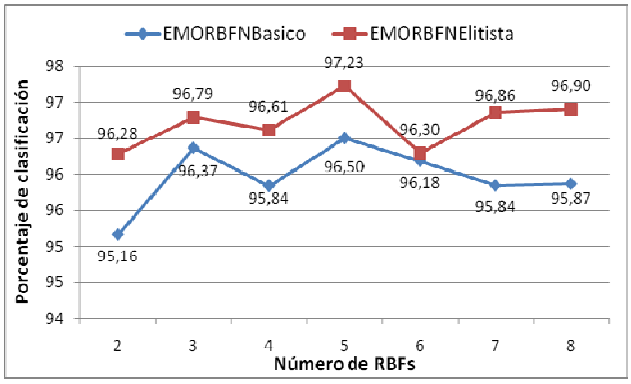

Figura 4. Resultados para el problema Wbcd.

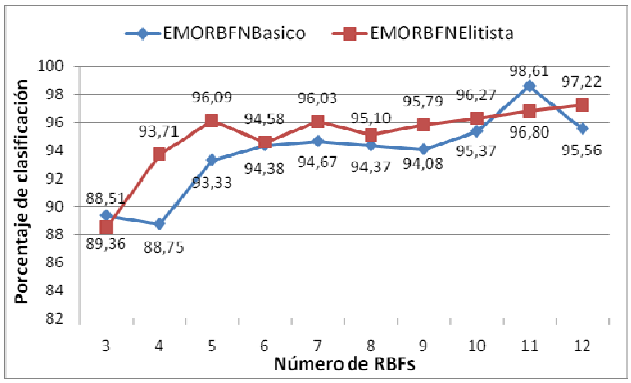

Figura 5. Resultados para el problema Wine.

En la evolución de las ejecuciones del algoritmo EMORBFNElitista hemos observado una mayor diversidad que para el algoritmo EMORBFNBasico. En las tablas de resultados observamos que EMORBFNElitista mejora con respecto a la versión básica. Exceptuando algunos casos puntuales, el operador de elitismo genera modelos con mayor precisión para redes con el mismo número de neuronas.

En general, los frentes obtenidos con EMORBFNElitista son más homogéneos, presentando altibajos menos pronunciados. Además, en los frentes de la versión elitista se observa de forma más clara la tendencia creciente en el porcentaje de clasificación a medida que aumentamos la complejidad de las RBFNs.

Si nos centramos únicamente en la precisión de las redes, EMORBFNElitista obtiene el mejor porcentaje de clasificación global tanto en Pima como en Wbcd. En cambio, en el problema Wine, EMORBFNBásico obtiene la mejor precisión global debido a la presencia de un pico en el caso de redes con 11 neuronas.

5 CONCLUSIONES

En este trabajo se presenta un nuevo elemento para aumentar la diversidad en el diseño multiobjetivo de RBFNs. Tras analizar el funcionamiento de estos

algoritmos se comprueba que, para la tarea de clasificación, sufren una fuerte pérdida de diversidad de individuos cuando el número de neuronas empieza a incrementarse. Una de las principales causas es que estos individuos necesitan un número mayor de generaciones para conseguir alcanzar un nivel adecuado de entrenamiento y eficiencia.

Para solventar este problema se ha propuesto definir un conjunto de subpoblaciones virtuales, donde cada subpoblación estaría compuesta por los individuos o redes de un determinado número de neuronas. El objetivo será mantener a los mejores individuos en eficiencia de cada subpoblación entre generaciones. Con esto hemos conseguido una mejora en la diversidad en la población y en la eficiencia general de los resultados alcanzados.

Agradecimientos

Este trabajo está soportado por el proyecto del Ministerio de Ciencia e Innovación TIN2008-06681-C06-02 y por el de la Junta de Andalucía TIC-3928.

Referencias

[1] Asuncion, A.; Newman, DJ. UCI machine learning repository. School of Information and Computer Science, University of California, Irvine, CA.http://www.ics.uci.edu/mlearn/MLRepository.html. 2007.

[2] Bäck, T.; Hammel, U.; Schwefel, H. Evolutionary computation: comments on the history and current state. *IEEE Trans. on Evolutionary Computation*, 1(1):3-17, 1997.

[3] Broomhead, D.; Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex System*. 2:321-355, 1988.

[4] Buchtala, O.; Klimek, M.; Sick, B. Evolutionary optimization of radial basis function classifiers for data mining applications. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 35(5):928-947, 2005.

[5] Deb, K. Multi-objective optimization using evolutionary algorithms. *Wiley*, 1st ed. 2001

[6] Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation* 6(2):182-197, 2002.

[7] Eshelman, LJ.; Schaffer, JD. Real-coded genetic algorithms and interval-squemata.Whitley, L.D. ed. Foundations of Genetic Algorithms 21:187-202. Morgan Kaufmann. 1993.

[8] Golub, G.; Van Loan, C. Matrix computations. J. *Hopkins University Press*, 3rd ed., 1996.

[9] González, J., Rojas, I., Ortega, J., Pomares, H., Fernández, J., Fco, A. Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation. *IEEE Transactions on Neural Networks*, 14(6):1478-1495, 2003.

[10] Guillén, A.; Pomares, H.; Rojas, I.; González, J.; Herrera, L.J.; Rojas, F.; Valenzuela, O.; Output value-based initialization for radial basis function neural networks. Neural Process Letters. (2007)

[11] Harpham, C.; Dawson, C.; Brown, M.; A review of genetic algorithms applied to training radial basis function networks. *Neural Computation Applications* 13:193-201. 2004

[12] Isaacs, A.; Ray, T.; Smith, W. An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization. *LNAI* 4828:257-268. 2007

[13] Jin, Y.; Sendhoff, B.; Extracting interpretable fuzzy rules from RBF networks. *Neural Process Letters* 17(2):149–164. 2003

[14] López, P.; Rivera, A.; Pérez-Godoy, M.; del Jesus, M.; Carmona, C. EMORBFN: An Evolutionary Multiobjetive Optimization algorithm for RBFN design. *Lecture Notes in Computer Science*. 5517: 752-759. 2009

[15] Moody, J.; Darken, C, J. Fast learning in networks of locally-tuned processing units. *Neural Computing*. 1:281-294, 1989.

[16] Park, J.; Sandberg, I. Universal approximation using radial-basis function networks. *Neural Comput.*, 3:246-257, 1991.

[17] Pedrycz, W. Conditional fuzzy clustering in the design of radial basis function neural networks. *IEEE Transactions on Neural Networks* 9(4):601-612, 1998.

[18] Powell, M. Radial basis functions for multivariable interpolation: A review. In *IMA. Conf. on Algorithms for the approximation of functions and data*, 143-167, 1985.

[19] Santana-Quintero LV, Coello CA, Hernández-Díaz AG. Hybridizing surrogate techniques, rough sets and evolutionary algorithms to efficiently solve multi-objective optimization problems. *GECCO'08* 763-764, 2007

[20] Teixeira, CA; Ruano MG, Ruano AE, Pereira WCA. A soft-computing methodology for non invasive time-spatial temperature estimation. *IEEE Transactions on Bio-Medical Engineering* 55:2:572-580, 2008

[21] Whitehead, B; Choate, T. Cooperative-competitive genetic evolution of Radial Basis Function centers and widths for time series prediction. *IEEE Transactions on Neural Networks*, 7(4):869-880, 1996.

[22] Widrow, B.; Lehr, M.A. 30 Years of adaptive neural networks: perceptron, madaline and backpropagation. *Proceedings of the IEEE*, 78(9):1415-1442, 1990.

[23] Yen GG. Multi-Objective evolutionary algorithm for radial basis function neural network design. *Studies in Computational Intelligence* 16:221-239, 2006