ARTÍCULO

Optimización paralela multiobjetivo de conjuntos de perceptrones multicapa para clasificación de patrones

Pedro A. Castillo, Juan J. Merelo-Guervós

Departamento de Arquitectura y Tecnología de Computadores
ETSI Informática . Universidad de Granada
E. 18071 Granada (España)
{pedro,jmerelo}@geneura.ugr.es

Resumen

En los problemas de clasificación de patrones se busca minimizar el número de patrones mal clasificados (error de clasificación). Sin embargo, en muchas aplicaciones reales hay que tener en cuenta por separado el error tipo I (falsos positivos) y el tipo II (falsos negativos), lo que hace el problema más complejo ya que un intento de minimizar uno de ellos, hace que el otro crezca. Es más, en ocasiones uno de estos tipos de error puede ser más importante que el otro, y se debe buscar un compromiso que minimice el más importante de los dos. A pesar de la importancia de los errores tipo II, la mayoría de los métodos de clasificación sólo tienen en cuenta el error de clasificación global. En este trabajo se propone la optimización de ambos tipos de error de clasificación utilizando un algoritmo multiobjetivo en el que cada tipo de error y el tamaño de red son objetivos de la función de evaluación (*fitness*). Se ha utilizado una versión modificada del método G-Prop de diseño y optimización de perceptrones multicapa usando un algoritmo evolutivo para optimizar simultáneamente la estructura de la red neuronal y los errores tipo I y II. Debido a la carga computacional que supone la ejecución de un algoritmo evolutivo para el diseño de redes neuronales, se propone la paralelización utilizando el modelo isla como forma de distribuir la carga en una red heterogénea.

Palabras clave: Optimización Multiobjetivo, Algoritmos Evolutivos, Redes Neuronales Artificiales, Clasificación de Patrones.

1. Introducción

Se pueden cometer dos tipos de error al llevar a cabo la clasificación de patrones. El primero consiste en la detección (errónea) de un efecto que realmente no existe (error tipo I o falso positivo), mientras que el segundo consiste en pasar por alto (no detectar) un efecto que sí existe (error tipo II o falso negativo). Por ejemplo, si tras un experimento llegamos a la conclusión de que una sustancia es tóxica cuando realmente no lo es, habremos cometido un error de tipo I. Sin embargo, si llegamos a la conclusión de que la sustancia no es tóxica cuando realmente lo es, habremos cometido un error de tipo II [29]. Este ejemplo muestra que ambos tipos de error son diferentes en el mundo real, y que deberían ser tratados de forma separada.

Otro ejemplo es la predicción de quiebra de empresas: los falsos positivos pueden dar lugar a pleitos (por parte de una empresa cuya quiebra se ha predicho y que haya podido perder clientes por esa razón), mientras que los falsos negativos conducirán solamente a la pérdida de un cliente (el que ha confiado en la predicción). La medida estadística más utilizada, la significancia es-

tadística, es una medida del error tipo I (la ausencia de un efecto significativo no quiere decir que no exista). Por otro lado, el test de hipótesis de Neyman-Pearson se basa en minimizar el error tipo II tras haber impuesto un límite al error tipo I [64, 8]. Hay muchos ejemplos de conclusiones incorrectas debido al error tipo II, y se debe poner cuidado para no interpretar el "no encontrar evidencia de efecto" como "evidencia de que no se produce el efecto" [59].

Por tanto, un problema de diseño de clasificadores binarios se acaba convirtiendo en uno de optimización de dos objetivos, es decir, minimización de errores de los dos tipos. En ello coincide con la mayoría de los problemas reales, que tienen varias variables que deben ser optimizadas al mismo tiempo. Resolver estos problemas multiobjetivo en los que varios objetivos pueden entrar en conflicto es una tarea compleja y costosa. En estos casos, en lugar de buscar una sola solución, el método debe obtener un conjunto de soluciones; más tarde, una o varias de ellas se usarán para resolver el problema. Los métodos de optimización multiobjetivo van desde la suma ponderada de los objetivos, que hoy día no se considera un buen método, hasta técnicas basadas en el frente de Pareto [15, 16, 47, 68].

Este trabajo continúa la investigación presentada en [13, 57], donde se llevó a cabo una comparación entre métodos estadísticos (regresión logística) y un método evolutivo para el diseño de redes neuronales artificiales (G-Prop) para la predicción de la quiebra de empresas. Conocer el estado financiero de una empresa puede ser un dato importante para los directivos, que ante los primeros avisos pueden tomar decisiones por anticipado para evitar la quiebra. El problema de determinar si una empresa va a quebrar o no se ha abordado tradicionalmente mediante técnicas heurísticas que requerían un detallado conocimiento de la empresa, por lo que sólo podían llevarlo a cabo expertos. En estos trabajos se comprobó que las redes neuronales diseñadas obtenían mejores resultados al clasificar una clase, frente a la otra: se centraban en clasificar correctamente los patrones de una clase (minimizaban el error tipo I, no cometían falsos positivos), mientras que el error tipo II se incrementaba (cometían bastantes falsos negativos). Este problema también se da al clasificar datos médicos, como en la predicción del cáncer.

En este trabajo se presenta un método multiobjetivo basado en G-Prop [10, 11, 12, 14] para optimizar redes neuronales buscando la minimización de los falsos positivos y falsos negativos (y no sólo del error global). G-Prop aprovecha la capacidad de dos tipos de algoritmo: la habilidad del AE para encontrar una solución cercana al óptimo global, y la del algoritmo Quick-Propagation (QP) [26] para afinar una solución y alcanzar el óptimo local más cercano, gracias a la búsqueda local a partir de la solución encontrada por el AE.

Para evitar que la búsqueda se centre en una zona localizada y sí se explore correctamente el espacio de soluciones, en lugar de utilizar una estructura de red preestablecida, la población de individuos (perceptrones multicapa, MLP) se inicializa con tamaños de red diferentes y aleatorios, y posteriormente se utilizan operadores genéticos específicos durante la evolución. Se diseñaron los operadores de mutación, cruce multipunto, añadir y eliminar neuronas ocultas, y entrenamiento. De esta forma, el algoritmo evolutivo (AE) busca y optimiza la arquitectura (número de neuronas ocultas), los pesos iniciales para esa arquitectura y el parámetro de aprendizaje del MLP.

En este trabajo se propone la mejora del error tipo I y tipo II en clasificación utilizando un algoritmo evolutivo multiobjetivo llamado MG-Prop que optimiza tanto los errores como la estructura del MLP. El objetivo es diseñar redes neuronales pequeñas que generalicen correctamente (y produzcan errores tipo I y II pequeños). Aproximar este problema como una tarea de optimización multiobjetivo hace innecesario el tener que ponderar los diferentes objetivos, y además, las soluciones basadas en el concepto de Pareto-optimalidad garantizan la diversidad de la población final.

Además, se ha demostrado que una mezcla de predictores (conjunto o *ensemble*) mejora las clasificaciones [35] por lo que proponemos usar el frente de Pareto final para formar un conjunto y así mejorar la clasificación que daría una sola red.

En resumen, el problema multiobjetivo que pretendemos resolver tiene tres objetivos: minimizar el error tipo I, el error tipo II y el tamaño de la red. Para comprobar la capacidad del método propuesto en este trabajo, se usarán dos problemas reales de clasificación de patrones (cáncer de mama -*breast cancer*- [44], y quiebra de empresas [57]).

El diseño de conjuntos de redes neuronales artificiales (RNA) es una tarea compleja que requiere

una gran cantidad de recursos computacionales (tiempo de CPU y memoria) debido a la complejidad y coste de las evaluaciones de las redes. Así pues, se propone su paralelización para ejecutarlo en un conjunto de máquinas distribuidas siguiendo el modelo isla [6].

El resto del trabajo está organizado de la siguiente forma: las Secciones 2 y 3 exponen el estado del arte en optimización multiobjetivo de redes neuronales artificiales. La Sección 4 describe en detalle el método MG-Prop. A continuación, la Sección 5 describe los experimentos llevados a cabo para comprobar la capacidad del método propuesto, seguido de los resultados obtenidos con dos problemas de clasificación de patrones (Sección 6). Por último, la Sección 7 presenta una serie de conclusiones y propuestas de trabajo futuro.

2. Definición de un problema multiobjetivo

La solución a un problema multiobjetivo (PMO) se puede definir matemáticamente como [48]: *"un vector de variables de decisión que satisface un conjunto de restricciones y optimiza un vector función cuyos elementos representan las funciones objetivo. Estas funciones forman una descripción matemática de los criterios de rendimiento de un sistema determinado, que normalmente están en conflicto entre ellos. Así, el término 'optimizar' significa encontrar una solución que asigne a cada función objetivo un valor aceptable para el diseñador."*

Formalmente, un PMO minimiza $F(x) = (f_1(x), ..., f_k(x))$ sujeto a $g_i(x) \leq 0 \quad \forall i = 1...m$ donde $x \in \Omega$. La idea es minimizar $F(x)$, siendo x un vector n-dimensional $x = (x_1, ..., x_n) \in \Omega$

Normalmente los objetivos están en conflicto entre sí, de forma que la mejora de uno de ellos sólo se puede dar empeorando otro. De hecho, encontrar el óptimo global en un PMO es un problema NP-completo [7]. En la mayoría de los casos no existe un óptimo global (que todas las variables satisfagan las restricciones, y todas las funciones objetivo alcancen simultáneamente su óptimo global).

En el campo de los PMO, el concepto de óptimo tiene un significado diferente al dado en los problemas con un solo objetivo. Se suele utilizar el concepto de optimalidad propuesto por Edgeworth [24] y generalizado por Pareto [49]. Así, $x^* \in F$ es Pareto-óptimo si no existe $x \in F$ que $f_i(x) \leq f_i(x^2) \quad \forall i = 1, ..., k$ y $\exists j$ que $f_j(x) < f_j(x^*)$

En los problemas con un solo objetivo suele haber una solución única. Sin embargo, el concepto de Pareto-optimalidad usado en PMO resulta en un conjunto de soluciones (normalmente grande), de las que posteriormente hay que seleccionar varias. Este conjunto de soluciones es lo que se llama "conjunto de Pareto". A los vectores que representan esas soluciones se les llama "no dominados", y al grafo de las funciones objetivo que usan esos vectores se le llama "frente de Pareto" (ver Figura 1).

Un PMO puede resolverse de diferentes formas. Históricamente, una aproximación directa para optimizar varios objetivos fue hacer una suma ponderada de los objetivos. Sin embargo, ese no es un método eficiente para resolver el problema [60], ya que la suma ponderada puede generar sólo una solución. El método más eficiente para atacar el problema es usar AE [15, 22]. Al estar basados en una población de soluciones candidatas, son capaces de generar un conjunto de soluciones Pareto-óptimas en cada ejecución del algoritmo.

En la bibliografía se pueden encontrar numerosas aproximaciones de AE multiobjetivo (MOEA) [68, 15]. Sin embargo, de acuerdo con el teorema de *No Free Lunch* [70], no existe una "mejor técnica" multiobjetivo.

A continuación se describen algunos de los más extendidos (que han sido utilizados con éxito con diversos problemas y aplicaciones reales) [15]:

- **MOGA** (Multi-Objective Genetic Algorithm). Implementado por Fonseca y Fleming [28]. Emplea un esquema de ranking basado en el frente de Pareto (el rango de un individuo viene dado por el número de individuos de la población que lo dominan). Incorpora la técnica de *fitness sharing*.
- **MOMGA** (Multi-Objective Messy Genetic Algorithm). Implementado por Van Veldhuizen [68]. Desarrollado para explorar la relación entre los bloques constructivos de las soluciones multiobjetivo (para reducir el número de bloques constructivos) y su uso en la búsqueda del MOEA. Incorpora *fitness sharing* y la selección de torneo propuesta por Horn et al. [34].

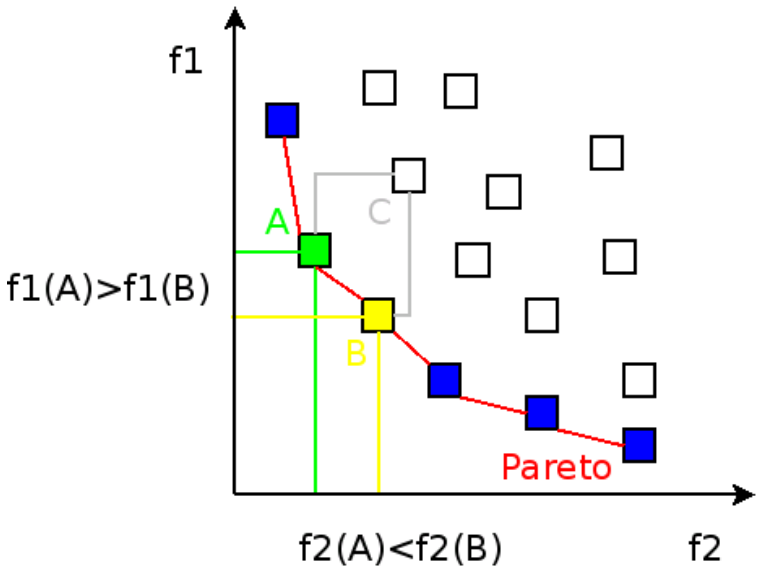


Figura 1: Ejemplo de frente de Pareto en un problema con dos funciones objetivo (f1 y f2). A y B son soluciones no dominadas, luego forman parte del frente. C es una solución dominada.

- **MOMGA-II**. Propuesto por Zydallis y Lamont [76]. Extiende el MOMGA usando la técnica *Probabilistically Complete Initialization* (PCI) y una fase de filtrado de bloques constructivos.
- **NPGA** (Niched-Pareto Genetic Algorithm). Implementado por Horn et al. [34]. Utiliza selección de torneo basada en el concepto de Pareto-optimalidad, y *fitness sharing*. Sobre el mismo algoritmo, Erickson et al. [25] proponen una mejora añadiendo *Pareto ranking* (el rango de un individuo está relacionado con el número de individuos que lo dominan).
- **NSGA** (Nondominated Sorting Genetic Algorithm). Implementado por Srinivas y Deb [63]. Emplea un esquema de ranking basado en el frente de Pareto, e incorpora la técnica de *fitness sharing*. Se basa en varias capas de clasificación de los individuos.
- **NSGA-II**. Propuesto por Deb et al. [21]. Mejora NSGA haciéndolo más eficiente computacionalmente. Incorpora elitismo y *crowding* para mantener la diversidad.
- **PAES** (Pareto Archived Evolution Strategy) y **PAES-II**. Implementado por Knowles y Corne [40]. Usa un solo individuo-padre que genera un solo descendiente (estrategia evolutiva). Utiliza un fichero externo y un procedimiento para mantener la diversidad. PAES-II utiliza regiones objetivo para seleccionar individuos.
- **SPEA** (Strength Pareto Evolutionary Algorithm). Implementado por Zitzler y Thiele [75]. Desarrollado para explorar el uso de los individuos en el frente de Pareto en la asignación generacional del fitness. Emplea un esquema de ranking basado en el frente de Pareto, e incorpora la técnica de *fitness sharing*. Usa una población secundaria en el proceso de asignación del fitness. Una mejora al algoritmo, propuesta por Zitzler et al. [74], utiliza asignación del fitness de grano fino, estimación de la densidad y truncación del archivo externo.
- **SFGA** (Single Front Genetic Algorithm). Desarrollado por De Toro et al. [19]. Usa un esquema de selección elitista conjuntamente con un método de aclarado que facilita una implementación eficiente.

Existen otros métodos en las que se proponen diferentes técnicas de selección, asignación del fitness o de *clustering*, aunque todos se basan en la misma idea general. En este trabajo se ha utilizado el SFGA debido a su eficiencia y buenos resultados obtenidos frente a otros [19, 20]. En la sección 4 se detallará el método seleccionado y las adaptaciones realizadas para la optimización de redes neuronales.

3. Optimización de redes neuronales artificiales usando métodos multiobjetivo

La evolución de RNA [71] es un método eficiente para resolver el problema de diseño de las redes.

Los AE recorren de forma más o menos aleatoria el espacio de soluciones, centrándose mediante la selección en aquellas zonas en las que el valor de la función de evaluación es máximo, mientras que los algoritmos de entrenamiento de RNA clásicos son algoritmos de descenso de gradiente, que iterativamente van haciendo disminuir el error hasta alcanzar un mínimo. Esto hace que, en este último caso, la búsqueda se concentre en una parte pequeña del espacio de soluciones, alrededor de la solución inicial de la que partió el método. En los AE, por el contrario, la búsqueda se lleva a cabo por todo el espacio, siendo posible además que en una población se encuentren soluciones que se sitúen cerca de varios óptimos.

La característica relevante del AE es que puede llevar a cabo una búsqueda global, situando la RNA cerca de un óptimo (global), a partir del cual, mediante el algoritmo de entrenamiento de la red (búsqueda local) se llegue rápida y eficazmente al óptimo [55]. Una ventaja añadida de estos métodos radica en que las dos formas de adaptación de estos sistemas, evolución y aprendizaje, hace que su capacidad de adaptación a entornos dinámicos sea efectiva.

Se puede hablar de tres enfoques principales para hacer evolucionar una RNA [71]: evolución de los pesos de conexión, de la arquitectura de red, y de las reglas de aprendizaje. La *evolución de los pesos de conexión* supone una aproximación adaptativa y global para realizar el entrenamiento, especialmente en aquellos casos en los que los algoritmos de entrenamiento basados en descenso del gradiente experimentan grandes dificultades. La *evolución de la arquitectura de red* hace que la propia red pueda adaptar su topología al problema sin intervención humana, lo cual representa una aproximación al diseño automático de RNA en tanto que los pesos y la arquitectura pueden hacerse evolucionar. La *evolución de las reglas de aprendizaje* significa una adaptación de las reglas de aprendizaje mediante evolución, en el sentido de buscar los parámetros de la regla de adaptación de pesos e incluso buscar nuevas reglas de adaptación de pesos.

La principal ventaja del diseño de RNA mediante un AE radica en la capacidad de éste para optimizar los diversos parámetros de la red (arquitectura, conectividad, pesos iniciales, regla de aprendizaje), y su paralelismo inherente (las diferentes redes pueden ser entrenadas simultáneamente en distintos procesadores).

Sin embargo, la optimización de todos los parámetros que definen la red puede representar varios objetivos diferentes que no se pueden minimizar por separado, ya que una mejoría en alguno de ellos supone un empeoramiento en el resto, así que se deben encontrar soluciones que sean sub-óptimas para cada uno de los objetivos por separado, pero de forma que la solución en conjunto sea *"aceptable"*, donde *"aceptable"* es un concepto totalmente subjetivo y dependiente del problema y del usuario final.

La optimización de varios objetivos opuestos simultáneamente se diferencia de la optimización de una sola función en que raramente admite una única, perfecta o utópica solución en la que todos los objetivos se satisfacen de forma plena. En cambio, los problemas multi-objetivo [48] se caracterizan por admitir un conjunto de soluciones alternativas que se deben considerar equivalentes en ausencia de información concerniente a la relevancia de unos objetivos sobre los otros.

Las técnicas de optimización convencionales, tales como el descenso en gradiente, u otras menos convencionales como el enfriamiento simulado (*simulated annealing*) presentan muchos inconvenientes para ser adaptadas a la resolución de problemas multi-objetivo, ya que no fueron diseñadas para ello y no están preparadas para albergar la posibilidad de que existan varias soluciones para un problema. Sin embargo, la mayoría de estos métodos calculan una suma ponderada de los objetivos (p.ej. minimizando el error y el número de pesos) o asignan una prioridad a alguno de los objetivos [10, 14] (dan más importancia al error que al tamaño de la red).

Por otro lado, los algoritmos evolutivos no tienen ningún problema para afrontar este tipo de problemas, ya que al disponer de una población de posibles soluciones, pueden realizar una exploración de todos los objetivos en paralelo [15, 22, 68]. Además, al ser capaces de buscar soluciones para problemas complejos, incluyendo características tales como discontinuidades, multimodalidad, espacios de soluciones dispersos y fun-

ciones de evaluación ruidosas, refuerzan su eficacia a la hora de abordar problemas multi-objetivo [28, 56].

Después de los primeros estudios sobre evolución multi-objetivo aparecidos a mediados de los 80 [61], fueron apareciendo diferentes implementaciones de estos algoritmos [15, 22, 63, 68]. Posteriormente, estos enfoques (con algunas variaciones) se han ido aplicando con éxito a varios problemas multi-objetivo [28, 15]. Recientemente, algunos autores se han ido centrando en algunos aspectos particulares de la evolución multi-objetivo, como la convergencia de la frontera Pareto-óptima [58], la creación de nichos y el elitismo, mientras que otros se han concentrado en el desarrollo de nuevas técnicas multi-objetivo evolutivas [41, 46].

En general, las RNA actúan como una caja negra, dando su clasificación ante un patrón de entrada sin proporcionar ningún tipo de explicación. Esto puede representar un problema para un experto que usa un método basado en RNA, dada la dificultad para explicar cómo calcula el método su salida.

Para abordar el problema de la optimización de los parámetros de las RNA, algunos autores proponen utilizar algoritmos multiobjetivo tales como el *Pareto differential evolution* (PDE) [4]. Este algoritmo es una adaptación de otro algoritmo llamado evolución diferencial, propuesto por Storn y Price [65]. PDE se utilizó con éxito para realizar la evolución de RNA [3].

Recientemente, Abbas ha propuesto la optimización simultánea de la arquitectura de red y el error de entrenamiento mediante un método multiobjetivo [3], habiendo llegado a que combinar Back-Propagation (BP) con un MOEA, se consigue una reducción considerable del coste de computación. En un trabajo posterior, Abbas [2] propuso dos aproximaciones multiobjetivo para la formación de conjuntos de redes neuronales: en la primera, parte el conjunto de entrenamiento en dos subconjuntos disjuntos, y construye dos objetivos consistentes en minimizar el error de entrenamiento para cada subconjunto. La segunda aproximación consiste en añadir ruido aleatorio al conjunto de entrenamiento para formar un segundo objetivo.

Jin et al. [36] proponen una versión modificada de dos algoritmos de optimización multiobjetivo (*Dynamic Weighted Aggregation* y *NSGA-II*) para abordar el problema de la regularización de RNA desde un punto de vista multiobjetivo. El método propuesto trata de optimizar tanto la estructura como los parámetros de la RNA.

En los últimos años, la investigación sobre RNA se está centrando en las mezclas de predictores [51] ya que representan una buena herramienta para resolver problemas complejos. Muchos estudios [42, 43, 62] se han centrado en la construcción de conjuntos de RNA, ya que habitualmente se consigue una mejora en la capacidad de predicción [35].

A pesar de la atención prestada a los conjuntos de RNA, aún quedan cuestiones abiertas sobre su diseño, tales como determinar su tamaño (número de componentes), o decidir si una red debe incluirse. Algunos autores utilizan una aproximación evolutiva para determinar qué redes formarán el conjunto [42, 43], otros utilizan un enfoque coevolutivo [30], y otros un enfoque multiobjetivo basado en el frente de Pareto [3].

La mayoría de estos métodos sólo calculan el error de clasificación global. Sin embargo, en muchos problemas, tanto el error tipo I como el tipo II deberían tenerse en cuenta al realizar las predicciones, que es lo que se hará en el método MG-Prop, propuesto en este trabajo.

4. MG-Prop : Evolución multiobjetivo de MLP

En este trabajo se presenta un método para optimizar tanto el error tipo I como el tipo II y la estructura de red de un MLP. Puesto que los objetivos pueden entrar en conflicto, se ha desarrollado un método multiobjetivo.

Se basa en G-Prop [10, 11, 12, 14], un método evolutivo para optimizar MLPs, y en SFGA (Single Front Genetic Algorithm), un algoritmo genético multiobjetivo que usa un esquema de selección elitista, desarrollado por De Toro et al. [19, 20].

A continuación se describen en detalle ambos métodos.

4.1. El método G-prop

El método G-Prop está basado en un AE, que facilita el diseño de MLP buscando los parámetros de aprendizaje, los pesos iniciales y la arquitectura de red para un MLP que resuelva un proble-

1. Generar la población inicial con valores de pesos y tamaños de capas ocultas aleatorios.
2. Repetir durante N generaciones:
   a. Evaluar los nuevos MLP (individuos) entrenándolos con el conjunto de entrenamiento y obteniendo la capacidad de clasificación con el conjunto de validación.
   b. Seleccionar los mejores S individuos en la población según el valor de su función de evaluación.
   c. Aplicar los operadores genéticos específicos a copias de los individuos seleccionados y obtener nuevos MLP.
   d. Reemplazar los peores individuos de la población por los nuevos.
3. Utilizar el mejor individuo encontrado para obtener el error de generalización con el conjunto de test.

Figura 2: Pseudocódigo del AE del método G-Prop

ma de clasificación concreto. El método propuesto trata de aprovechar las ventajas que presentan tanto los AE como las presentadas por los algoritmos basados en descenso del gradiente, en este caso QP: la capacidad del AE para encontrar una solución cercana al óptimo global y la de QP para sintonizar la búsqueda, alcanzando el óptimo local más cercano a la solución encontrada por el AE.

La descripción completa del método y los resultados obtenidos ante problemas de clasificación de patrones y aproximación funcional se han publicado en [10, 11, 12, 14]. De forma resumida, el algoritmo utilizado queda especificado en la Figura 2.

En G-Prop, un individuo es un MLP completo. Un AE necesita que los individuos de la población estén codificados en los cromosomas para poder ser manejados por los operadores genéticos del AE. Sin embargo, G-Prop no usa codificación binaria ni de otro tipo; en lugar de ello, la estructura de datos que que representa los parámetros iniciales de la red (pesos iniciales y tasa de aprendizaje) se hace evolucionar usando los operadores genéticos diseñados. Estas características son posibles gracias a la filosofía de la biblioteca de Objetos Evolutivos [45], según la cual, cualquier objeto que posea una función de evaluación puede hacerse evolucionar. Los operadores genéticos actúan directamente sobre el objeto MLP, de forma que sólo los pesos iniciales y la tasa de aprendizaje inicial se hacen evolucionar, no los pesos o la tasa obtenidos tras el entrenamiento. Para calcular el valor de la función de evaluación de un individuo se crea una copia exacta del objeto MLP antes de entrenarlo, de forma que los pesos iniciales del MLP original no se modifican. Además, cuando modifican un MLP, acceden a éste de forma que cada neurona intermedia (y todos sus pesos) es tratada como un gen, de forma que cuando por ejemplo se cruzan dos MLP, realmente se intercambian neuronas de las capas ocultas completas (la neurona intermedia y sus pesos de entrada y salida se tratan conjuntamente), tal y como se propone en [66].

Se han desarrollado seis operadores para hacer evolucionar MLP:

- El **operador de mutación** modifica los pesos de ciertas neuronas, aleatoriamente, dependiendo del porcentaje de aplicación. Está basado en las ideas de Montana y Davis [17] y en el algoritmo presentado por Kinnebrock en [39]. El algoritmo presentado por éste se basa en aplicar un operador de mutación tras cada presentación del conjunto de datos de entrenamiento (al final de cada época). Dichas modificaciones consisten en cambiar los pesos de la red neuronal añadiendo un pequeño número aleatorio uniformemente distribuido en el intervalo $[-0{,}1, 0{,}1]$. La tasa de aprendizaje del MLP se modifica añadiendo un pequeño número aleatorio uniformemente distribuido en el intervalo $[-0{,}05, 0{,}05]$ (del orden de magnitud del valor de inicialización, de forma que no se varíe demasiado el valor de la tasa para evitar hacer que un salto brusco en el descenso de gradiente lo aleje de la "atracción" de un mínimo).

- El **operador de cruce** lleva a cabo el cruce multipunto entre dos MLP, de forma que se obtienen dos redes cuyas neuronas en las capas ocultas son una mezcla de las neuronas ocultas de las dos redes: algunas neuronas ocultas (junto con sus pesos de entrada y salida) de cada MLP padre forman un descendiente, y el resto de neuronas ocultas forman el otro. Este tipo de cruce es del tipo *uniforme*, por lo tanto necesita un parámetro que indique el número de "genes" que se van a intercambiar los dos MLP. Por último, las tasas de aprendizaje de los MLP son intercambiadas entre ambos descendientes.
- El **operador de entrenamiento** se usa para mejorar los individuos-MLP a través de una búsqueda local hecha con QP, tal y como se sugiere en [17] y [72]. Al aplicarlo, el operador toma un MLP y lo entrena un número de épocas especificado, devolviéndolo, con los pesos entrenados, a la población. QP es un método de ajuste adaptativo de la tasa de aprendizaje. Así pues, al aplicarlo, la tasa de aprendizaje del MLP se modifica, manteniéndose modificada al ser devuelto a la población. Se podría decir que esta estrategia implementa lamarckismo, ya que aquellas características adquiridas durante la vida del MLP se guardan en el "código genético" de forma que los descendientes pueden heredar esas características.
- El **operador de añadir neuronas** y el siguiente (eliminar neuronas ocultas) están pensados para atacar uno de los principales problemas de BP (y sus variantes): la dificultad de *adivinar* el número de neuronas en cada capa oculta. Añadiendo neuronas ocultas evitamos el problema de prefijar el tamaño del espacio de búsqueda del AE. Este operador lleva a cabo lo que se suele llamar "diseño incremental": comienza con estructuras pequeñas y las incrementa añadiendo nuevas unidades inicializadas aleatoriamente a las capas ocultas. En contrapartida, este incremento de tamaño puede hacer que los MLP tengan un tamaño excesivo: las redes grandes son más rápidas en la fase de aprendizaje [9], aunque una forma de obtener una buena generalización es usar la red más simple que aproxime los datos de entrada [54, 71].
- El complementario al anterior es el **operador para eliminar neuronas ocultas**. Está pensado para llevar a cabo lo que se llama "diseño decremental": poda ciertas unidades ocultas para obtener redes más pequeñas [50, 9]. En cierto modo, siguiendo este método se evita el problema de que las redes crezcan demasiado.
- El **operador para sustituir neuronas ocultas** elige una aleatoriamente, y la cambia por otra, inicializada con pesos aleatorios. Se puede considerar una especie de macromutación que afecta a un sólo "gen" en el MLP. La macromutación puede llegar a destruir lo aprendido, pero también puede tener un efecto "salto" en el espacio de búsqueda a otras zonas alejadas de la actual, de forma que dicho MLP puede caer en una buena zona y después, mediante QP o una mutación, obtener una buena solución.

G-Prop no necesita que se establezca ningún parámetro del MLP a mano (salvo el número de épocas de entrenamiento para evaluar los individuos de la población, ya que dependiendo del problema a resolver, un mayor o menor número de épocas es necesario). Por supuesto, algunos parámetros de ejecución del AE (relativos a la población inicial y los operadores genéticos) deben ser establecidos de antemano. Este método ha sido presentado en trabajos anteriores, y aplicado con éxito a diferentes poblemas de clasificación de patrones y aproximación funcional [10, 11, 12, 14].

4.2. El método SFGA

Se trata de un algoritmo genético multiobjetivo que usa un esquema de selección elitista conjuntamente con un método de aclarado que facilita una implementación eficiente. Ha sido desarrollado por De Toro et al. [19, 20], y se basa en la idea de copiar en la siguiente población, en lugar del conjunto de élite completo (individuos no dominados) un conjunto reducido, con los individuos distribuidos homogéneamente en el espacio de búsqueda.

Si el número de individuos del primer frente de dominancia, debido al efecto acumulativo de sucesivas iteraciones, aumenta por encima de un determinado porcentaje del cardinal de la población total, se aplica entonces un procedimiento basado

```
1. Inicializar aleatoriamente la población Pt=P0
2. Evaluar Pt para cada uno de los objetivos
3. Calcular el primer frente de dominación de Pt,
   P_frente=NoDominados(Pt)
4. Obtener el conjunto de élite, Pt_elite=P_frente
5. Obtener el conjunto de élite diversificado,
   Pt_elite_diversificado=Aclarado(Pt_elite)
6. Incluir Pt_elite_diversificado en población de descendencia
   Dt=Pt_elite_diversificado
7. while(tamaño de Dt < Npob)
   a. elegir dos individuos al azar xi,xj de Pt_elite
   b. recombinar xi y xj generando el individuo xk
   c. mutar xk con probabilidad pm generando xk'
   d. hacer Dt=Dt U {xk'}
8. end while
9. Hacer t=t+1 y Pt=Dt
10. If(t<T) ir al paso 2
11. Hacer A=NoDominados(Pt)
```

Figura 3: Pseudocódigo del algoritmo SFGA

en un modelo de aclarado en el espacio de objetivos para obtener el conjunto de élite diversificado a partir del conjunto de élite de la población actual. El algoritmo SFGA utiliza elitismo de primer frente aplicado en dos vertientes: (1) el conjunto de élite diversificado se copia en la población siguiente; (2) la población de progenitores está formada por los individuos del conjunto de élite diversificado.

La Figura 3 ilustra el funcionamiento del algoritmo SFGA. Los parámetros del algoritmo son el tamaño de la población, el número de generaciones, el radio de aclarado, el umbral de aclarado y la probabilidad de mutación [19].

SFGA ha sido comparado con otros procedimientos multiobjetivo, como el SPEA, habiendo proporcionado mejores resultados en la mayoría de los casos [20].

4.3. El método propuesto: MG-Prop

El método propuesto se basa en el ciclo de ejecución del algoritmo SFGA, aunque ha sido adaptado para nuestro problema de diseño de conjuntos de RNA.

Tras un proceso de experimentación, se pudo ver que conforme avanza el proceso evolutivo, el número de individuos no dominados crece considerablemente. Es más, algunos individuos-MLP pueden clasificar perfectamente los patrones correspondientes a la clase A (cometen un 0 % de error), mientras que fallan cuando se les presenta un patrón de clase B (cometen el 100 % de error). Estos individuos no hábiles son realmente no dominados, pero no son adecuados ya que no representan soluciones buenas para el problema, por lo que deberían ser eliminados.

Por otro lado, los individuos seleccionados para reproducirse y generar la descendencia para la siguiente generación se toman de aquellos que forman el frente de Pareto. Los operadores genéticos para generar nuevos individuos son los mismos diseñados e incluidos en G-Prop (ver la Subsección 4.1).

El algoritmo desarrollado queda descrito en el pseudocódigo de la Figura 4, que es una adaptación del algoritmo G-Prop [10, 11, 12, 14].

Al terminar, se utiliza un conjunto de patrones no vistos anteriormente (conjunto de test) y se clasifican, obteniendo el error global, los errores tipo I y II, y el tamaño del MLP (número de pesos de conexión entre neuronas).

1. Generar una población de N individuos (Población).
2. Evaluar los N individuos: entrenarlos usando el conjunto de entrenamiento y obtener su fitness, tamaño de red y errores tipo I y II ante el conjunto de validación.
3. Repetir durante G generaciones:
   a. copiar Población en P0
   b. eliminar los individuos dominados de Población (obteniendo P1)
   c. eliminar los individuos no hábiles de P1 (obteniendo P2)
   d. podar P2 usando clustering para limitar el número de individuos
   e. seleccionar S individuos de P2 y aplicarles los operadores genéticos (a copias suyas) para obtener la Descendencia
   f. evaluar los nuevos individuos en Descendencia
   g. copiar en Población los individuos no dominados (P2) y la Descendencia
   h. si N es mayor que el número de individuos actual en Población, tomar individuos dominados (P0) para completar Población
4. Eliminar los individuos dominados y los no hábiles de Población.
5. Formar un conjunto con los individuos-MLP no dominados.
6. Clasificar con el conjunto formado los patrones de test para obtener el error total, tipo I y tipo II.

Figura 4: Pseudocódigo del algoritmo MG-Prop

Las redes del conjunto de Pareto (frente) serán potencialmente buenas, sin embargo, puesto que una sola red que ofrece un buen error ante el conjunto de entrenamiento no tiene por qué ser la mejor en términos de generalización (clasificando patrones no vistos durante la fase de entrenamiento), se propone utilizar todos los MLP en el frente de Pareto para formar un conjunto, tal y como se propone en [1, 2]. Este proceso queda descrito de forma gráfica en la Figura 5.

Se proponen tres métodos para obtener las clasificaciones para cada patrón de entrada: (1) la clase de salida se calcula por votación (mayoría); (2) la clase de salida se obtiene teniendo en cuenta la mayor activación de entre todas las salidas de todas las redes que forman el conjunto; (3) la clase de salida se calcula como la media de las salidas de todas las redes que forman el conjunto.

De esta forma, un experto que tiene que decidir si una persona está enferma (o una empresa está en quiebra) o no, dispone de más información sobre ese caso que si tuviera en cuenta la salida de una sola RNA.

El uso de conjuntos de MLP supone una buena opción para resolver problemas de clasificación, pero requiere una gran cantidad de recursos computacionales debido a la complejidad y coste de las evaluaciones de las redes. Es por esto que se propone la paralelización del algoritmo usando un modelo de grano grueso.

Este modelo (Figura 6) se basa en un sistema paralelo en el que la población está distribuida en un conjunto de máquinas heterogéneas unidas por una red ethernet, formando un anillo. De esta forma, la carga de trabajo queda repartida.

Cada nodo ejecuta una instancia de MG-Prop, de forma que en la práctica tenemos un sistema paralelo de grano grueso donde varias máquinas trabajan en paralelo buscando la solución, enviando/recibiendo individuos de las poblaciones cada cierto número de generaciones siguiendo un esquema de migración en anillo.

La implementación de las comunicaciones entre procesos se llevó a cabo utilizando el módulo `gSOAP` [1] [67] en C++, una implementación de SOAP (*Simple Object Access Protocol*) [69] protocolo estándar propuesto por el W3C que permite acceder a objetos remotos de forma independiente de la máquina y del lenguaje. Es uno de los protocolos estándar incluidos en Globus [31]. Un cliente usando SOAP puede enviar o recibir objetos, o acceder a un método de un objeto remoto. SOAP emite y recibe mensajes usando XML [53].

Es un protocolo de alto nivel, ligero, simple y extensible, pensado para comunicaciones entre

[1] http://sourceforge.net/projects/gsoap2

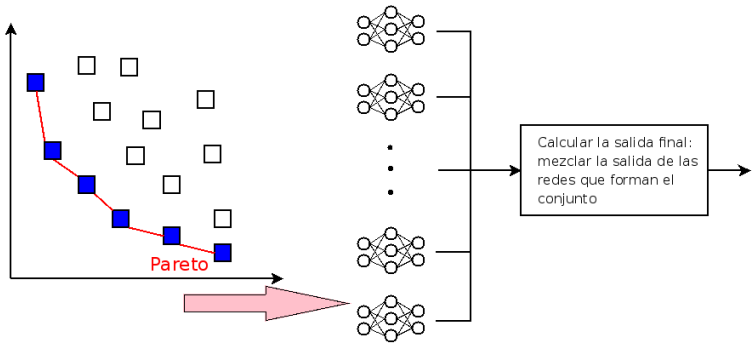


Figura 5: A partir de los individuos-MLP que forman el frente de Pareto al final del algoritmo, se forma un conjunto para llevar a cabo la clasificación de los patrones del problema.

aplicaciones. SOAP facilita al programador la tarea de distribuir objetos en diferentes servidores, sin tener que preocuparse por los formatos de los mensajes, ni, en muchos casos, la llamada explícita a servidores remotos.

5. Experimentos

Los tests utilizados para medir la capacidad de un método deben escogerse cuidadosamente ya que algunos de ellos (problemas de juguete) no son adecuados para medir ciertas capacidades de los algoritmos de entrenamiento, como la generalización. Según Prechelt [52], al menos dos problemas reales deberían utilizarse para comprobar el funcionamiento de un algoritmo, y en cualquier caso, el mejor método para comprobar las capacidades y las limitaciones de un algoritmo es aplicarlo a la resolución de problemas reales.

De acuerdo a lo expuesto, se seleccionaron dos problemas reales complejos de clasificación de patrones: predicción del cáncer de mama (*breast-cancer*) y un problema de detección de quiebra de empresas.

Para los problemas de clasificación, disponemos de un conjunto global de patrones, que se usarán para entrenar las redes, para asignar su valor de fitness, y para comprobar su capacidad de generalización. Así, para cada problema, el conjunto de datos inicial se dividió en tres partes disjuntas, para entrenamiento, validación y test. Para obtener el fitness, el MLP es entrenado con el conjunto de entrenamiento y su fitness se establece a partir del error de clasificación con el conjunto de validación. Una vez que el AE ha terminado, se calcula el error de clasificación final con el conjunto de test, que es el error mostrado en las tablas de resultados.

5.1. Problema 1: Predicción del cáncer de mama

Este problema, tomado de [52], consiste en la diagnosis del cáncer de mama, tratando de clasificar un tumor como benigno o maligno (a partir de la descripción de las células analizadas mediante un examen al microscopio).

El conjunto de datos, llamado *"Wisconsin breast cancer database"*, proviene del repositorio de pruebas para aprendizaje automático del *UCI machine learning dataset Wisconsin breast cancer database* [2]. Lo obtuvo el Dr. William H. Wolberg [44] en el Hospital Universitario de Wisconsin, Madison. Prechelt realiza un exhaustivo análisis de este conjunto de datos en [52].

Cada patrón tiene 10 atributos (código del patrón, grosor del grupo de células, uniformidad del tamaño de las células, uniformidad de la forma de las células, adhesión marginal, tamaño del epitelio de una sóla célula, núcleos desnudos, suavidad cromática, núcleos normales, mitosis) más la clase (benigno o maligno).

La distribución en clases es del 65,5 % como patrones de tumores benignos y el 34,5 % de tumores malignos. En la evaluación con este problema hemos utilizado el llamado ***Cancer1a***, que es el conjunto utilizado por Grönroos en [33] de los tres propuestos por Prechelt.

[2] http://www.ics.uci.edu/~mlearn/MLSummary.html

Figura 6: Modelo paralelo con esquema de migración en anillo propuesto para distribución de carga en la optimización de los conjuntos.

5.2. Problema 2: Predicción de la quiebra de empresas

El conjunto de datos se obtuvo de la base de datos de Infotel S.A. para los años 1998 y 1999[3]. La muestra empleada en el presente trabajo consta de 450 empresas no financieras, de las que la mitad son empresas fracasadas y la otra mitad empresas sanas. Además se han utilizado 76 empresas que no formaban parte de las anteriores y que constituye una muestra de validación de los modelos obtenidos. Las primeras se corresponden a las que se habían declarado en suspensión de pagos o quiebra legal, de acuerdo con la Ley Concursal española, mientras que las segundas se seleccionaron aleatoriamente de un total de 150.000 empresas.

La muestra es aleatoria, y dentro de ella se ha intentado incorporar un número suficiente de empresas quebradas, de tal manera que haya dos grupos, quebradas y no quebradas, similares.

La variable dependiente toma el valor 1 en el caso de quiebra legal de la empresa, y 0 en el caso de que la empresa esté sana. Las variables independientes seleccionadas para ser introducidas en el modelo son datos financieros cuantitativos junto con información cualitativa.

[3] http://www.axesor.com

5.3. Metodología

Los usuarios de AEs deben ajustar de forma manual ciertos parámetros, como son la probabilidad de cruce, la de mutación, el tamaño de la población, el número de generaciones, el porcentaje de selección[18]. Los valores para estos parámetros de ejecución se suelen asignar tomándolos de la bibliografía, o bien mediante procesos de ensayo y error [38, 32]. En cualquier caso, es muy importante conocer cuáles de los parámetros del AE tienen mayor influencia en su comportamiento, y para qué valores de esos parámetros, el funcionamiento es óptimo. Cuando se hace un análisis detallado de la influencia de cada parámetro, el diseñador debería prestar atención a los parámetros cuyos valores provocan efectos más significativos.

En [14] se llevó a cabo un estudio estadístico riguroso y detallado para determinar los parámetros más importantes del método G-Prop (en el sentido de la influencia sobre los resultados), y para establecer los valores más adecuados para dichos parámetros (y así obtener un funcionamiento óptimo). En dicho estudio se usó ANOVA (*ANa-*

Parámetro	Valor
número de generaciones	500
tamaño de población	500
porcentaje de selección	20 %
rango de inicialización de pesos	$[-0{,}05, 0{,}05]$
prioridad del operador de mutación	2.0
prioridad del operador de cruce	0.5
prioridad del operador de añadir neuronas	1.0
prioridad del operador de eliminación	0.5
prioridad del operador de entrenamiento	0.5
probabilidad de mutación	0,4
mutación de los pesos de conexión	$[-0{,}001, 0{,}001]$
mutación de la constante de aprendizaje	$[-0{,}010, 0{,}010]$
número de épocas de entrenamiento	500

Cuadro 1: Valores más adecuados para cada parámetro del método evolutivo G-Prop.

lysis Of the VAriance) [27], una herramienta estadística ampliamente utilizada, basada en el análisis de la varianza. Para cada conjunto de experimentos, se aplicó ANOVA para determinar si el efecto de esos parámetros sobre el error obtenido y el tamaño era significativo estadísticamente, y la herramienta ANOM (*ANalysis Of the Mean*) para obtener el valor más adecuado para cada parámetro.

Como resultado del estudio, se obtuvieron los valores mostrados en el Cuadro 1 para cada parámetro de ejecución del método evolutivo G-Prop.

Por supuesto, dependiendo de la dificultad del problema, el número de generaciones y el tamaño de población deberán ser mayores o menores, para conseguir una mayor diversidad.

Puesto que MG-Prop se basa en G-Prop, se han utilizado los resultados obtenidos en [14] para establecer los valores de los parámetros de ejecución (tras una experimentación exhaustiva se ha comprobado que son adecuados). Sin embargo, para evitar ejecuciones excesivamente largas, los experimentos se llevaron a cabo usando 200 generaciones, un tamaño de población de 200 individuos, y 500 épocas de entrenamiento.

Las tablas de resultados mostradas en la Sección 6 muestran valores en media y desviación estándar, obtenidas tras 30 ejecuciones para cada método. Se usaron tests estadísticos *t-Student* para evaluar las diferencias en las medias de los resultados, ya que este tipo de test puede usarse incluso con muestras pequeñas. Este método se usa para comprobar si las diferencias de las medias de dos muestras pequeñas puede, razonablemente, suponerse igual a cero, lo que implica aceptar la hipótesis nula ($H_0 \equiv$ medias iguales).

Para comparar resultados entre los diferentes modelos, los AE secuenciales se ejecutaron un número de generaciones y con tamaño de población tal que, en media, el número de individuos (MLP) evaluados en cada simulación fuese equivalente al número de individuos evaluados en las ejecuciones distribuidas.

La idea es suponer que el esfuerzo necesario para resolver el problema es constante, y debemos repartirlo entre las máquinas disponibles (de una a cuatro) para ejecutarlo eficientemente, obteniendo resultados similares o mejores, pero intentando reducir el tiempo de computación. De esta forma, el problema a resolver establece la carga computacional, y ésta se reparte entre los recursos disponibles. Puesto que la fase más costosa de un método evolutivo de diseño de RNA es el entrenamiento de las redes, la población inicial del algoritmo se distribuye equitativamente entre las subpoblaciones.

Los experimentos se ejecutaron en cuatro máquinas del propio departamento, cuyas velocidades están entre los 1700 y 2000 Mhz, y están conectados utilizando la red ethernet de 10Mbits de la Universidad, lo que supone una alta latencia en las comunicaciones.

Núm. Nodos	Método		Error global	Error Tipo I	Error Tipo II	Tamaño MLP	Tiempo (min.)
1	Prechelt [52]		1.2	-	-	-	-
1	Grönroos [33]		2.0 ± 0.6	-	-	-	-
1	G-Prop		1.2 ± 0.1	1.1 ± 0.1	1.3 ± 0.1	173 ± 22	25±5
1	MG-Prop	(V)	1.3 ± 0.5	1.5 ± 0.9	1.1 ± 0.6	*conjunto de*	25±4
		(M)	1.4 ± 0.4	2.0 ± 0.8	0.9 ± 0.5	17±4 *MLP de*	
		(A)	1.4 ± 0.4	2.1 ± 0.8	0.0 ± 0.8	125 ± 31 *pesos*	
2	MG-Prop	(V)	1.2 ± 0.3	1.4 ± 0.9	0.9 ± 0.6	*conjunto de*	15±3
		(M)	1.3 ± 0.4	1.9 ± 0.8	0.7 ± 0.5	26±5 *MLP de*	
		(A)	1.4 ± 0.5	2.0 ± 0.9	0.8 ± 0.7	123 ± 44 *pesos*	
3	MG-Prop	(V)	1.2 ± 0.3	1.4 ± 0.7	0.8 ± 0.6	*conjunto de*	10±3
		(M)	1.3 ± 0.4	1.8 ± 0.6	0.7 ± 0.4	38±6 *MLP de*	
		(A)	1.3 ± 0.5	1.8 ± 0.8	0.7 ± 0.5	129 ± 28 *pesos*	
4	MG-Prop	(V)	1.1 ± 0.2	1.3 ± 0.7	0.7 ± 0.4	*conjunto de*	8±2
		(M)	1.2 ± 0.4	1.7 ± 0.8	0.6 ± 0.3	46±4 *MLP de*	
		(A)	1.2 ± 0.3	1.7 ± 0.6	0.7 ± 0.4	135 ± 38 *pesos*	

Cuadro 2: Resultados obtenidos con el problema de predicción del cáncer de mama. Se muestra el error global de clasificación, el error tipo I, el tipo II, y el tamaño de los MLP. Los resultados se obtuvieron usando G-Prop y MG-Prop. La clasificación de los conjuntos se ha calculado mediante votación (V), media de las salidas (M) y mayor activación (A). El tamaño de los MLP se ha calculado en función del número de parámetros de la red, esto es, del número de pesos de conexión. En el caso de MG-Prop se obtiene un conjunto, de forma que se muestra tanto el número de componentes del conjunto como el tamaño de esos componentes. Las ejecuciones con MG-Prop se han llevado a cabo en un máximo de cuatro nodos, de forma que se muestran resultados para todos estos números de nodos.

Nodos	ganancia	eficiencia	fracción serie
1	1	100	-
2	1.75	88	0.14
3	2.52	84	0.09
4	3.16	79	0.088

Cuadro 3: Métricas de rendimiento para el problema del cáncer: ganancia en velocidad, eficiencia y fracción serie.

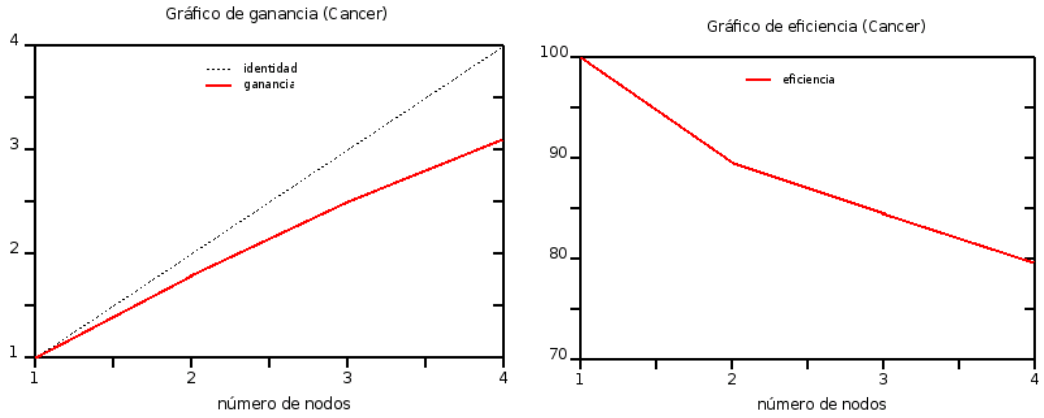
Para medir el rendimiento hemos usado la ganancia de velocidad y la fracción serie [37]: La ganancia de velocidad se ha tomado respecto al sistema más rápido [23], y para las versiones paralelas, se fue añadiendo la mejor máquina de las disponibles, hasta completar un máximo de cuatro nodos. De esta forma se puede garantizar que si la ganancia obtenida es superior a la unidad, el rendimiento del sistema paralelo es superior al de cualquier sistema monoprocesador [5]. La fracción serie es una medida que permite determinar si la pérdida de eficiencia ante un aumento del número de nodos se debe a las partes del programa que no se ejecutan en paralelo. Si la ganancia de velocidad es baja pero la fracción serie permanece constante, se puede afirmar que los resultados son buenos [37].

6. Resultados obtenidos

6.1. Problema 1: Predicción del cáncer de mama

Los resultados con el problema de la predicción del cáncer de mama se muestran en el Cuadro 2.

Se puede ver que el método multiobjetivo obtiene resultados comparables a los del G-Prop (método que da prioridad al error global de clasificación frente al tamaño de la red). Aunque los resultados de MG-Prop no son mucho mejores, sí se consigue una ligera mejora en la clasificación global, y las diferencias entre los errores tipo I y tipo II se minimizan. De alguna forma, la clasificación de



a b

Figura 7: Gráfica de la ganancia en velocidad (a) y eficiencia (b) para el modelo paralelo con el problema de predicción del cáncer de mama.

las diferentes clases es más homogénea.

En cuanto a las clasificaciones obtenidas por los diferentes tipos de conjuntos, la votación ofrece mejores resultados que tomar la media de las salidas o la mayor activación de los componentes del conjunto.

En cuanto al tamaño de las redes, puesto que el conjunto está compuesto por varios MLP, el número total de pesos es mayor que el obtenido usando G-Prop. En cualquier caso, el tamaño de los MLP individuales resulta algo menor que los obtenidos por G-Prop.

Por último, se utilizaron tests t-Student para verificar los resultados. Entre el método G-Prop y MG-Prop no se han encontrado diferencias significativas en las medias de los errores globales o del tipo I, salvo cuando aumentamos el número de nodos. Sin embargo, sí las hay en cuanto al error tipo II (al nivel del 99 %).

En cualquier caso, el problema de la predicción del cáncer de mama es fácilmente abordable usando G-Prop, ya que clasifica correctamente la mayoría de los patrones. Vemos que comete errores tipo I y tipo II pequeños, por lo que resulta difícil encontrar diferencias entre ambos modelos.

Tras analizar los resultados en cuanto a calidad de las soluciones, pasemos a analizar los resultados en cuanto a tiempos de ejecución y métricas de rendimiento. El Cuadro 3 y las Figuras 7 a y b muestran la gráfica de la ganancia en velocidad obtenidas con el problema de predicción del cáncer de mama. Vemos que el método paralelo obtiene buenos resultados en cuanto a ganancia en velocidad, con una pérdida moderada de eficiencia al incrementar el número de nodos. En este punto, la variación en los valores de la fracción serie indica esta pérdida de eficiencia se deberá a la parte secuencial intrínseca del programa. Se puede afirmar que el método propuesto escala hasta cuatro máquinas, mejorando los resultados en cuanto al tiempo de simulación, al dividir el problema entre varias máquinas que trabajen en paralelo, manteniendo la calidad de los resultados.

6.2. Problema 2: Predicción de la quiebra de empresas

El Cuadro 4 muestra los resultados obtenidos usando regresión logística, G-Prop y MG-Prop con el problema de la quiebra. Los resultados publicados previamente sobre este problema mostraron que la capacidad de predicción de los MLP era ligeramente inferior a la de la regresión logística, en cuanto al error tipo II y al global. Sin embargo, obtenía mejores errores tipo I.

En G-Prop, sin embargo, la medida del fitness sólo tenía en cuenta el error global y el tamaño de red, dando prioridad al primero; no utilizaba el error tipo I y tipo II.

Los resultados obtenidos usando MG-Prop son ligeramente mejores que los obtenidos previamente [57]. Como vimos anteriormente para el primer problema, las diferencias entre los errores tipo I y tipo II se minimizan, reduciéndose a la vez el error global.

Núm. Nodos	Método		Error global	Error Tipo I	Error Tipo II	Tamaño RNA	Tiempo (min.)
1	Logit [57]		15.92	17.24	14.48	-	-
1	G-Prop [57]		17.26	11.21	23.32	1042	35±6
1	MG-Prop	(V)	16 ± 4	16 ± 2	17 ± 3	*conjunto de*	36±4
		(A)	18 ± 1	18 ± 4	18 ± 3	19±4 *MLP de*	
		(B)	18 ± 2	19 ± 3	18 ± 4	1140 ± 40 *pesos*	
2	MG-Prop	(V)	14 ± 2	15 ± 2	14 ± 3	*conjunto de*	20±5
		(A)	16 ± 2	16 ± 3	16 ± 2	33±2 *MLP de*	
		(B)	17 ± 3	18 ± 3	17 ± 4	1120 ± 52 *pesos*	
3	MG-Prop	(V)	14 ± 4	15 ± 2	14 ± 4	*conjunto de*	14±3
		(A)	15 ± 2	15 ± 3	14 ± 4	48±5 *MLP de*	
		(B)	16 ± 3	16 ± 1	17 ± 3	1060 ± 61 *pesos*	
4	MG-Prop	(V)	12 ± 2	13 ± 3	12 ± 2	*conjunto de*	11±4
		(A)	13 ± 1	13 ± 2	13 ± 3	61±6 *MLP de*	
		(B)	15 ± 3	14 ± 2	16 ± 2	1114 ± 49 *pesos*	

Cuadro 4: Resultados obtenidos con el problema de la quiebra de empresas usando regresión logística (logit), G-Prop [57] y MG-Prop.

Nodos	ganancia	eficiencia	fracción serie
1	1	100	-
2	1.82	91	0.09
3	2.57	86	0.08
4	3.27	81	0.07

Cuadro 5: Métricas de rendimiento para el problema de la quiebra: ganancia en velocidad, eficiencia y fracción serie.

Se puede observar que los mejores resultados se obtienen al calcular la salida del conjunto usando la votación.

En cuanto a los tamaños de red, al igual que en el problema anterior, MG-Prop encuentra un conjunto de redes de mayor tamaño que el MLP encontrado por G-Prop, aunque los componentes son de un tamaño comparable.

Tras aplicar tests t-Student para verificar los resultados, observamos que existen diferencias significativas con niveles de confianza entre el 90 % y 95 %.

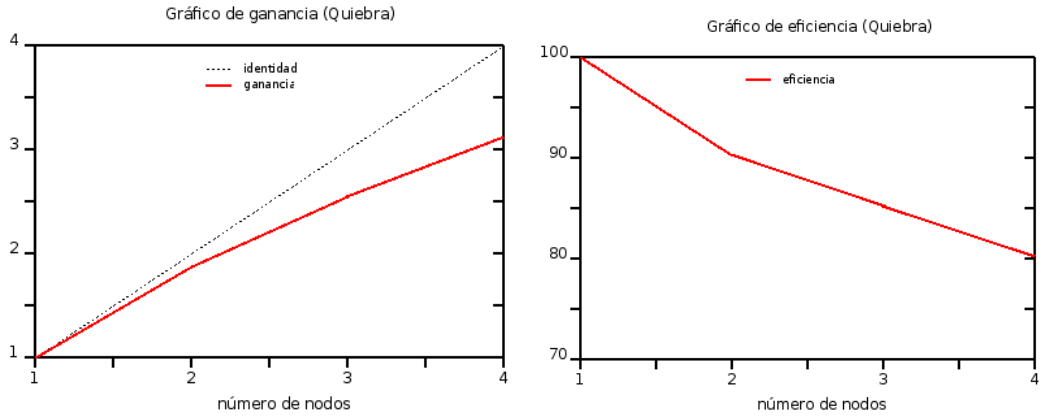
En cuanto a tiempos de ejecución y métricas de rendimiento, la ganancia en velocidad y eficiencias obtenidas con el problema de la quiebra de empresas se muestran en el Cuadro 5 y en las Figuras 8 a y b.

Se observa que el algoritmo puede obtener un conjunto cuyas clasificaciones sean similares e incluso mejores en un menor tiempo. Vemos que el método paralelo obtiene buenos resultados en cuanto a ganancia en velocidad, con una pérdida moderada de eficiencia. En cualquier caso, los valores de la fracción serie (valores muy pequeños) indican que los resultados de las configuraciones paralelas son buenos.

7. Conclusiones y trabajos futuros

En este trabajo se propone optimizar la capacidad de clasificación de MLP afrontándola como un problema de optimización multiobjetivo: se minimizan tanto el error tipo I, el error tipo II y el tamaño de la red. Un MLP solo puede no ser el mejor clasificador en términos de generalización. Por esto, se usan todos los MLP en el frente de Pareto al final del algoritmo evolutivo multiobjetivo como un conjunto para mejorar la capacidad de generalización.

El frente de Pareto establece el tamaño del conjunto y asegura que los componentes son diferentes y contribuirán a optimizar los objetivos.



a b

Figura 8: Gráfica de la ganancia en velocidad (a) y eficiencia (b) para el modelo paralelo con el problema de predicción de la quiebra de empresas.

MG-Prop calcula y tiene en cuenta los tipos de error, y obtiene resultados comparables (e incluso mejores) que G-Prop, que da prioridad a un objetivo sobre el resto. El método propuesto lleva a cabo un compromiso entre los objetivos, optimizando los tipos de error.

Debido a la cantidad de redes que se deben evaluar en una ejecución del AE, el método necesita una gran potencia computacional para diseñar eficientemente los MLP. Por eso se ha propuesto su paralelización usando el modelo de isla, de forma que hemos repartido la carga computacional entre varios procesadores.

Se han abordado dos problemas reales de clasificación de patrones: predicción del cáncer de mama y de quiebra de empresas. MG-Prop se ha mostrado competitivo con otros métodos y consigue reducir el error global y también las diferencias entre los errores tipo I y tipo II.

En cualquier caso, la idea de optimizar los errores tipo I y tipo II junto con la estructura del MLP como un problema multiobjetivo es interesante y podría aplicarse para mejorar otros métodos de clasificación.

En el caso de la predicción de la quiebra de empresas, el modelo se ha obtenido con datos de un año antes de la quiebra. Como trabajo futuro, sería interesante obtener más datos de la base de datos de Axesor S.A. y buscar modelos para realizar las predicciones dos o más años antes de la declaración de quiebra.

Por otro lado, Zhou et al. analizaron en [73] la relación entre el conjunto y las redes que lo componían. En sus resultados obtuvieron que usar sólo unas pocas redes de la población final para formar el conjunto daba mejores resultados que incluirlas todas. De acuerdo con esto, sería interesante utilizar métodos automáticos, tales como algoritmos coevolutivos, para llevar a cabo la selección de los componentes del conjunto.

A partir de los resultados obtenidos, la votación parece generar mejores resultados. En cualquier caso, sería interesante investigar nuevas formas de construir los conjuntos para llevar a cabo la clasificación.

Por último, en este trabajo se presenta un estudio preliminar de las posibles mejoras que la paralelización puede aportar. Sin embargo, surgen diversos temas de estudio que deben investigarse para ampliar el trabajo, como pueden ser estudios de los diversos tipos de migración, diferentes tipos de estructura de red entre las máquinas conectadas, y métodos para distribuir la carga de forma equilibrada.

Agradecimientos

Este trabajo ha sido financiado en parte por los proyectos CICYT TIC2003-09481-C04-04, TIN2007-68083-C02-01, de Excelencia de la Junta de Andalucía P06-TIC-02025, y el proyecto del Plan Propio de la Universidad de Granada PIUGR 9/11/06 con título *"Desarrollo en internet de servicios distribuidos en redes heterogéneas: aplicación a la resolución de problemas complejos me-*

diante co-evolución".

Referencias

[1] H.A. Abbass. Pareto neuro-ensemble. *The 16th Australian Joint Conference on Artificial Intelligence (AI'03), Perth, Lecture Notes in Artificial Intelligence LNAI 2903, Springer-Verlag, 554-566*, 2003.

[2] H.A. Abbass. Pareto neuro-evolution: constructive ensemble of neural networks using multi-objective optimization. *IEEE Congress on Evolutionary Computation (CEC2003), IEEE-Press, Vol. 3, pp. 2074-2080, Canberra, Australia*, 2003.

[3] H.A. Abbass. Speeding up back-propagation using multiobjective evolutionary algorithms. *Neural Computation, MIT Press, Vol. 15, No 11, pp. 2705-2726*, 2003.

[4] H.A. Abbass, R.A. Sarker, and C.S. Newton. PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. *In Proc. of the IEEE Congress on Evolutionary Computation (CEC2001), vol.2,pp.971-978,Piscataway,NJ, IEEE Press*, 2001.

[5] E. Alba and G. Luque. Evaluation of Parallel Metaheuristics. *Lecture Notes in Computer Science, Volumen: 4193, pp. 9-14, Reykjavik, Islandia*, 2006.

[6] E. Alba and J. M. Troya. A Survey of Parallel Distributed Genetic Algorithms. *Complexity*, 4(4):31–52, Marzo 1999.

[7] T. Bäck. Evolutionary algorithms in theory and practice. *Oxford University Press, New York*, 1996.

[8] R.J. Barton. Neyman-Pearson Hypothesis TestingSignal. *ECE 6333 - Signal Detection and Estimation*, 2005.

[9] G. Bebis, M. Georgiopoulos, and T. Kasparis. Coupling weight elimination with genetic algorithms to reduce network size and preserve generalization. *Neurocomputing 17 (1997) 167-194*, 1997.

[10] P. A. Castillo, J. Carpio, J. J. Merelo, V. Rivas, G. Romero, and A. Prieto. Evolving Multilayer Perceptrons. *Neural Processing Letters, vol. 12, no. 2, pp.115-127. October*, 2000.

[11] P. A. Castillo, J. J. Merelo, V. Rivas, G. Romero, and A. Prieto. G-Prop: Global Optimization of Multilayer Perceptrons using GAs. *Neurocomputing, Vol.35/1-4, pp.149-163*, 2000.

[12] P.A. Castillo, M.G. Arenas, J. J. Merelo, V. Rivas, and G. Romero. Optimisation of Multilayer Perceptrons Using a Distributed Evolutionary Algorithm with SOAP. *Lecture Notes in Computer Science, Vol.2439, pp.676-685, Springer-Verlag*, 2002.

[13] P.A. Castillo, M.G. Arenas, J.J. Merelo, V.M. Rivas, and G. Romero. Multiobjective Optimization of Ensembles of Multilayer Perceptrons for Pattern Classification . *Lecture Notes in Computer Science, Vol. 4193, pp.453-462. Springer-Verlag. Reikiavik. Islandia*, 2006.

[14] P.A. Castillo, J.J. Merelo, G. Romero, A. Prieto, and I. Rojas. Statistical Analysis of the Parameters of a Neuro-Genetic Algorithm. *in IEEE Transactions on Neural Networks, vol.13, no.6, pp.1374-1394, ISSN:1045-9227, november*, 2002.

[15] C.A. Coello Coello, D.A. Van-Veldhuizen, and G.B. Lamont. Evolutionary algorithms for solving multi-objective problems. *Kluwer Academic Publishers, New York, ISBN 0-3064-6762-3*, 2002.

[16] Carlos A. Coello Coello and Nareli Cruz Cortés. Solving Multiobjective Optimization Problems using an Artificial Immune System. *Genetic Programming and Evolvable Machines, Vol. 6, No. 2, pp. 163–190*, 2005.

[17] D.J. Montana; L. Davis. Training feedforward neural networks using genetic algorithms. *Proc. 11th Internat. Joint Conf. on Artificial Intelligence, 762-767*, 1989.

[18] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, EE. UU., 1991.

[19] F. de Toro, J. Ortega, J.Fernandez, and A.F Diaz. Parallel genetic algorithm for multiobjective optimization. *10th Euromicro Workshop on Parallel, Distributed and Network-based processing, IEEE Computer Society, pp. 384-391*, 2002.

[20] F. de Toro, J. Ortega, E. Ros, S. Mota, B. Paechter, and J.M. Martín. PSFGA: Parallel processing and evolutionary computation for multiobjective optimisation. *Parallel Computing, 30, pp.721-739*, 2004.

[21] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186(2/4):311-338*, 2000.

[22] K. Deb. Multi-objective optimization using evolutionary algorithms. *John Wiley and Sons, New York*, 2001.

[23] V. Donaldson, F. Berman, and R. Paturi. Program speedup in a heterogeneous computing network. *Journal of Parallel and Distributed Computing, 21:316-322*, 1994.

[24] F.Y. Edgeworth. Mathematical Physics. *P. Keagan, London, England*, 1881.

[25] M. Erickson, A. Mayer, and J. Horn. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. *In Zitzler, E., Deb, K., Thiele, L., Coello, C.A.C., and Corne, D., editors, 1st Intl. Conf. on Evolutionary Multi-Criterion Optimization, pp.681-695. Springer-Verlag. Lecture Notes in Computer Science, vol.1993*, 2001.

[26] S.E. Fahlman. Faster-Learning Variations on Back-Propagation: An Empirical Study. *Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann*, 1988.

[27] R.A. Fisher. The Comparison of Samples with Possibly Unequal Variances. *Annals of Eugenics, 9, pp.174-180*, 1936.

[28] C. M. Fonseca and P. J. Fleming. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A unified Formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):26–37, Enero 1998.

[29] J.A. Freiman, T.C. Chalmers, and H. Smith. The importance of beta, the type II error and sample size in the design and interpretation of the randomized control trial. *New England Journal of Medicine, 299:690-694*, 1978.

[30] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. Evolutionary Computation 9(3): 271-302*, 2005.

[31] Globus. Globus Toolkit version 4. *Disponible en http://www.globus.org*, 2007.

[32] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. Systems, Man, and Cybernetics, SMC-16(1):122-128*, 1986.

[33] M.A. Grönroos. Evolutionary Design of Neural Networks. *Master of Science Thesis in Computer Science. Department of Mathematical Sciences. University of Turku.*, 1998.

[34] J. Horn, N. Nafpliotis, and D. E. Goldgerg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In Z. Michalewicz, J. D. Schaffer, H. P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation, ICEC'94*, pages 82–87, Orlando, Florida, EE. UU., 1994. IEEE Computer Society Press.

[35] H. Ishibuchi and T. Yamamoto. Evolutionary multiobjective optimization for generating an ensemble of fuzzy rule-based classifiers. *In Proc. of the Genetic and Evolutionary Computation Conference (GECCO2003), Lecture Notes in Computer Science (LNCS), pp.1077-1088, Chicago, IL*, 2003.

[36] Y. Jin, T. Okabe, and B. Sendhoff. Neural network regularization and ensembling using multiobjective evolutionary algorithms. *Congress on Evolutionary Computation, CEC2004. Vol.1, pp.1-8. ISBN:0-7803-8515-2*, 2005.

[37] A. H. Karp and H. P. Flatt. Measuring parallel processor performance. *Communications of the ACM, 33(5):539-543*, 1990.

[38] D. Kim and C. Kim. Forecasting time series with genetic fuzzy predictor ensemble. *IEEE Transactions on Fuzzy Systems, vol.5,no.4,pp.523-535*, 1997.

[39] Werner Kinnebrock. Accelerating the standard backpropagation method using a genetic approach. *Neurocomputing, 6, 583-588*, 1994.

[40] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation, 8(2):149-172*, 2000.

[41] M. Laumanns, G. Rudolph, and H. P. Schwefel. A Spatial Predator-Prey Approach to Multi-objective Optimization: A Preliminary Study. In A. E. Eiben, T. Bäck, M. Schoenauer, and H. P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 241–249, Amsterdam, Holanda, 1998. Springer-Verlag.

[42] Y. Liu and X. Yao. Ensemble leagning via negative correlation. *Neural Networks, 12(10):1399-1404*, 1999.

[43] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation, 4(4):380-387*, 2000.

[44] O. L. Mangasarian, R. Setiono, and W.H. Wolberg. Pattern recognition via linear programming: Theory and application to medical diagnosis. *Large-scale numerical optimization, Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30*, 1990.

[45] J. J. Merelo, M. G. Arenas, J. Carpio, P. A. Castillo, V. M. Rivas, G. Romero, and M. Schoenauer. Evolving objects. *In M. Graña, editor, FEA2000 (Frontiers of Evolutionary Algorithms) proceedings. Proc. JCIS'2000. P.P.Wang(ed). Editorial Association for Intelligent Machinery. ISBN:0-9643456-9-2. Vol I, pp.1083-1086. Atlantic City, NJ, Feb. 27-March 3.*, 2000.

[46] S. Obayashi, S. Takahashi, and Y. Takeguchi. Niching and Elitist Models for MOGAs. In A. E. Eiben, T. Bäck, M. Schoenauer, and H. P. Schwefel, editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 260–269, Amsterdam, Holanda, 1998. Springer-Verlag.

[47] J. Olvander. Robustness considerations in multi-objective optimal design. *Journal of Engineering Design, Vol. 16, No. 5, pp. 511–523*, 2005.

[48] A. Osyczka. Multicriteria optimization for engineering design. *In John S. Gero, editor, Design Optimization, pp.193-227. Academic Press*, 1985.

[49] V. Pareto. Cours d'economie politique. *volume I and II. F. Rouge, Lausanne*, 1896.

[50] M. Pelillo and A. Fanelli. A Method of Pruning Layered Feed-Forward Neural Networks. *Lecture Notes in Computer Science, Vol. 686, pp. 278-283, Springer-Verlag*, 1993.

[51] M.P. Perrone and L.N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. *Neural Networks for Speech and Image Processing, R.J.Mammone, Ed., pp.126-142*, 1993.

[52] L. Prechelt. PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, Septiembre 1994.

[53] Erik T. Ray. *Learning XML: creating self-describing data*. O Reilly, January 2001.

[54] R.D. Reed and R.J. Marks. *Neural Smithing*. Bradford. The MIT Press, Cambridge, Massachusetts, London, England., 1999.

[55] J.M Renders and S.P. Flasse. Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics, Vol.26, No.2, pp.243-258*, 1996.

[56] I. Rojas, J. González, H. Pomares, F. J. Rojas, F. J. Fernández, and A. Prieto. Multidimensional and Multideme Genetic Algorithms for the Construction of Fuzzy Systems. *International Journal of Approximate Reasoning*, 26(3):179–210, Abril 2001.

[57] I. Roman, J.M. de la Torre, M.E. Gomez, P.A. Castillo, and J.J. Merelo. Bankruptcy prediction adapted to firm characteristics. an empirical study. *26th Annual Congress European Accounting Association. Congress Book. pp. A-108. Sevilla, April*, 2003.

[58] G. Rudolph. On a Multi-objective Evolutionary Algorithm and its Convergence to the Pareto Set. In D. B. Fogel, editor, *Proceedings of the Fifth IEEE International Conference on Evolutionary Computation, ICEC'98*, pages 511–516, Anchorage, Alaska, EE. UU., 1998. IEEE Computer Society Press.

[59] J. Savulescu, I. Chalmers, and J. Blunt. Are research ethics committees behaving unethically? Some suggestions for improving performance and accountability. *Br. Med. J., 313, 1390-1393*, 1996.

[60] Y. Sawaragi, H. Nakayama, and T. Tanino. Theory of multiobjective optimization. *Volume 176 of Mathematics in science and engineering. Academic Press Inc, Harcourt Jovanovich*, 1985.

[61] J. D. Schaffer. *Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, Tennessee, EE. UU. UMI Order Number: AAI8522492, 1984.

[62] A.J.C. Sharkey. On combining artificial neural nets. *Connection Science, 8:299-313*, 1996.

[63] N. Srinivas and K. Dev. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Otoño 1995.

[64] Jonathan A.C. Sterne. Teaching hypothesis tests - time for significant change? *STATISTICS IN MEDICINE. 21:985-994 (DOI: 10.1002/sim.1129)*, 2002.

[65] R. Storn and K. Price. Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical Report TR-95-012, International Computer Science Institute, Berkeley. http://http.icsi.berkeley.edu/~storn/tr-95-012.ps.gz*, 1995.

[66] D. Thierens, J. Suykens, J. Vandewalle, and B. De Moor. Genetic weight optimization of a feedforward neural network controller. *In Proceedings of the Conference on Artificial Neural Nets and Genetic Algorithms, pp. 658-663. Springer-Verlag*, 1993.

[67] R.A. Van-Engelen. gSOAP and Web Services. *C/C++ Users Journal, vol.23, no.2, pp.67-73*, 2005.

[68] D.A. Van-Veldhuizen and G.B. Lamont. Multiobjective evolutionary algorithms: analyzing the state-of-the-art. *Evolutionary Computation 8(2): 125-147*, 2000.

[69] Don Box; David Ehnebuske; Gopal Kakivaya; Andrew Layman; Noah Mendelsohn; Henrik Frystyk Nielsen; Satish Thatte; Dave Winer. Simple object access protocol (soap) 1.1, w3c note 08 may 2000. Available from `http://www.w3.org/TR/SOAP`.

[70] D.H. Wolpert and W.G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation, 1(1):67-82*, 1997.

[71] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE, 87(9):1423-1447*, 1999.

[72] Xin Yao and Yong Liu. Towards Designing Artificial Neural Networks by Evolution. *Applied Mathematics and Computation, 91(1):83-90*, 1998.

[73] Z.H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence, vol.137, no.1-2, pp.239-253*, 2002.

[74] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the Strength Pareto Evolutionary Algorithm. *Technical Report 103. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich*, 2001.

[75] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation, 3(4):257-271*, 1999.

[76] J.B. Zydallis, D.A. Van-Veldhuizen, and G.B. Lamont. A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. *In Zitzler, E., Deb, K., Thiele, L., Coello, C.A.C., and Corne, D., editors, 1st Intl. Conf. on Evolutionary Multi-Criterion Optimization, pp.226-240. Springer-Verlag. Lecture Notes in Computer Science, vol.1993*, 2001.