Departamento de Arquitectura y Tecnologías de Computadores
Escuela Técnica Superior de Ingeniería Informática
Universidad de Granada

APROXIMACION FUNCIONAL MEDIANTE REDES DE FUNCIONES DE BASE RADIAL, UNA ALTERNATIVA PARA LA PREDICCION EN EL PROCESO DE REDUCCION DE MINERAL DE LA TECNOLOGIA CARON DE PRODUCCION DE NIQUEL

TESIS DOCTORAL

DOCTORADO COOPERADO
UNIVERSIDAD DE HOLGUÍN – UNIVERSIDAD DE GRANADA

Realizada por:
M.Sc. Ing. Francisco Fernández Periche

Directores:
Dr. Julio Ortega Lopera y Dr. Ignacio Rojas Ruíz

Granada, Enero del 2008

Editor: Editorial de la Universidad de Granada
Autor: Francisco Fernández Periche
D.L.: Gr. 3026 - 2007
ISBN: 978-84-338-4724-9

APROXIMACION FUNCIONAL MEDIANTE REDES DE FUNCIONES DE BASE RADIAL, UNA ALTERNATIVA PARA LA PREDICCION EN EL PROCESO DE REDUCCION DE MINERAL DE LA TECNOLOGIA CARON DE PRODUCCION DE NIQUEL

Memoria presentada por

M.Sc. Ing. Francisco M. Fernández Periche

Para optar al grado de

DOCTOR EN INFORMATICA

Firmado por: Francisco M Fernández Periche

D. Julio Ortega Lopera, Catedrático de Universidad y
D. Ignacio Rojas Ruiz, Profesor Titular
del Departamento de Arquitectura y Tecnología de Computadores

CERTIFICAN

Que la memoria titulada:

APROXIMACION FUNCIONAL MEDIANTE REDES DE FUNCIONES DE BASE RADIAL, UNA ALTERNATIVA PARA LA PREDICCION EN EL PROCESO DE REDUCCION DE MINERAL DE LA TECNOLOGIA CARON DE PRODUCCION DE NIQUEL

ha sido realizada por D. Francisco M. Fernández Periche bajo nuestra dirección en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para optar al grado de Doctor en Informática.

Granada, Enero del 2008

Firmado por:

D. Julio Ortega Lopera

D. Ignacio Rojas Ruíz

Dedicatoria

A mis padres

A mi hija Ania

A mi esposa María Isabel

Pensamiento

Creo en el milagro de lo que puede hacer el trabajo,
de lo que puede hacer la ciencia,
de lo que pueden hacer los hombres.

Fidel Castro Ruz

Agradecimientos

Al Ministerio de Educación Superior de Cuba, a la Universidad de Holguín y al Ministerio de la Industria Básica por darme esta nueva oportunidad de superación profesional.

A la Junta de Andalucía, a la Universidad de Granada y a la Empresa Cmdte Ernesto Che Guevara, que han financiado parte de este Proyecto de Doctorado Cooperado entre la Facultad de Informática y Matemática de la Universidad de Holguín y la Escuela Técnica Superior de Ingeniería Informática.

A los profesores D. Julio Ortega Lopera y D. Ignacio Rojas Ruíz por la ayuda que me han prestado a lo largo de este período.

A todos los profesores del Doctorado, y en especial a Manuel Mariño, José Luis Verdegay y Rosa Urquiza, que siguen soñado con ver realizado este su proyecto.

Quiero agradecer al Dr. Alberto Guillén Perales, al Dr Antonio Jesús Rivera Rivas y al profesor Daniel Mendiola Ellis, por su colaboración más allá de las responsabilidades de sus trabajos.

A mis compañeros de Tecnologías de la Información, a los miembros del Consejo de Dirección de mi Empresa, y en especial a nuestro Director René Estévez Soto por su gran confianza y apoyo incondicional.

A mis dos familias, por su amor y alegría.

Por último, quisiera dar las gracias a todos mis amigos de allá y de aquí por lo mucho que han confiado en mí.

Resumen

La Reducción de Mineral es uno de los procesos más importante y complejo que tiene la tecnología lixiviación carbonato amoniacal de mineral reducido de producción de níquel (proceso CARON). Para favorecer el desarrollo de las distintas reacciones químicas que deben producirse, es necesario mantener un determinado perfil térmico y gaseoso dentro de los hornos mediante la combustión incompleta del petróleo.

Este tipo de proceso, por naturaleza, es un alto consumidor de portadores energéticos, entre los que sobresalen: aire, vapor, electricidad y petróleo tecnológico.

Conocer con antelación cómo se comportaría cualquiera de estas variables en el futuro, constituye una información muy valiosa y supone una indudable ventaja competitiva. Se trata de una herramienta de evidente interés tanto desde el punto de vista económico, como para la gestión y el control de procesos industriales.

En esta memoria de Tesis Doctoral se presenta una propuesta de solución que es capaz de ***predecir el consumo de petróleo en cámaras***, como parte de algunos de los procesos que están presentes en la Reducción de Mineral.

Basada en técnicas de Softcomputing, nuestro modelo ha incluido la hibridación e integración de diversas técnicas y paradigmas de programación tales como, algoritmos de búsqueda local, algoritmos evolutivos multiobjetivo y redes neuronales de funciones de base radial.

El empleo de las redes de funciones de base radial como aproximador funcional partiendo de un conjunto de datos reales, demuestra la utilidad de esta técnica en aplicaciones realistas de indudable interés socio económico.

Abstract

Ore reduction is CARON Technology most important and complex process for Nickel production. In order to support the development of the several chemical reactions that must occur in this process, it is necessary to keep a specific heat and gaseous profile inside the ore reduction roasters by means of oil incomplete combustion.

This kind of process, by nature, is a high consumer of energetic bearer, within which air, steam, power and technological oil are highlighted.

To know in advance the future behaviour of one of these variables, is very valuable information and it is supposed to have an undoubtedly competitive advantage. It is a tool of great interest from the economic point of view as well as for management and the control of industrial processes.

This Doctoral thesis memory contains a proposal for solution that is able to *predict oil consumption inside the chambers*, as part of some of the processes that are present in the Ore Reduction process.

It is based on Soft computing techniques and it integrates elements of dynamic neuroprogramming, evolutional optimization algorism and procedures for perdition or modeling based on neuronal networks and orthogonal transformations.

The use of Radial Base Function Networks as a functional approximation from a set of real data shows its feasibility for use in practical application.

The algorism used in the approximation module for prediction of the main function inside the system, is based on the genetic algorism with classification that is not designated in its second version (NSGA-II)

Indice General

Indice de figuras

Indice de tablas

Indice de algoritmos

Tenemos un problema ¡Qué fastidio!

Bueno, vamos a intentar solucionarlo. ¿Sabemos cómo hacerlo?

No. Entonces, a probar.

¿Podemos generar varias soluciones válidas a ese problema?

Ya, claro que podemos, pero van a ser todas muy malas soluciones.

Bueno, no importa. Las generamos: 40, 100, las que sean.

¿Alguna es mejor que otra? Todas son malas, pero unas son menos malas que otras.

Cogemos las mejores, las otras las eliminamos.

¿Y ahora qué?

Bueno, las podemos coger de dos en dos y mezclarlas a ver si sale algo bueno ¿Sí?

Bueno, a veces funciona. Vamos a hacerlo otra vez. Y otra.

También podemos mezclarlas de otra forma.

¡Oye esto se estanca, ahora son todas iguales!

Entonces, vamos a coger de vez en cuando alguna de las malas, no sólo de las buenas.

¿Y si hacemos pequeños cambios al azar en pequeñas zonas de alguna solución,
a ver si alguna da en el clavo?

Vale...

¡Oye, esto está mejorando! [29]

Capítulo 1

Introducción

En este primer capítulo se presentan los antecedentes, motivaciones, objetivo y metodología de investigación utilizada al abordar el problema objeto de estudio.

También se presenta brevemente la estructura de la Tesis, comentando el contenido de los diferentes capítulos.

Por último, luego del resumen de contenidos, se describen las principales contribuciones resultantes del trabajo realizado en estos años.

1.1 Antecedentes

La Universidad de Granada, España y la Universidad de Holguín, Cuba, conscientes de la importancia de afianzar las relaciones interuniversitarias, han convenido en aunar esfuerzos para llevar a buen término un *Programa de Doctorado en Informática* que contribuya a facilitar una formación avanzada y orientada a cualificar recursos humanos que impulsen la docencia y la suficiencia investigativa en el campo de la investigación educativa y la formación de profesorado, en las que ambas instituciones tienen intereses comunes.

Desde la reciprocidad y compromiso mutuos, y con la experiencia lograda en la formación de Doctores; ambas instituciones ponen a disposición sus cuadros académicos e investigadores, así como las posibilidades que hoy ofrece la avanzada tecnología en el campo de la enseñanza-aprendizaje de alto nivel y excelencia, para obtener los beneficios futuros que en los ámbitos profesionales e institucionales ambas anhelan.

Como novedad de este proyecto cooperado, cabe destacar, la privilegiada participación de la Empresa Ernesto Guevara, la cual viene mostrando un interés creciente en la resolución de problemas reales con métodos e investigaciones científicas, demostrando además que las mismas no son patrimonio sólamente de instituciones docentes de alto nivel o centros de investigación.

Esta empresa estatal, conocida también como Planta de Níquel de Punta Gorda, es un complejo minero-metalúrgico con 20 años de funcionamiento.

1.2 Motivación

Atendiendo a la idea que está en la base de este trabajo, la humanidad hace tiempo que busca mejores maneras de realizar las tareas cotidianas de la vida. A través de su historia, se puede observar la larga búsqueda de fuentes más efectivas de alimentos al comienzo y luego de materiales, energía y manejo del entorno físico. Sin embargo, relativamente tarde, comenzaron a formularse ciertas clases de preguntas generales de manera cuantitativa, primero en palabras y después en notaciones simbólicas. Un aspecto predominante de estas preguntas generales era la búsqueda de lo "mejor" o en su defecto, lo "óptimo".

Se han realizado grandes esfuerzos por describir complejas situaciones humanas y sociales. Para tener significado, esto debería escribirse en una expresión matemática que contenga una o más variables, cuyos valores deben determinarse. La pregunta que se formula, en términos generales es, ¿qué valores deberían tener estas variables para que esa expresión alcance el mayor valor numérico posible (maximización) o el menor valor numérico posible (minimización)? A este proceso general de maximización o minimización se le denomina "*Optimización"* [16].

En los problemas de decisión, que normalmente se presentan en cualquier ámbito industrial, empresarial o en la misma vida cotidiana, partimos de una serie de recursos, de unos requisitos mínimos que hay que cumplir, los cuales condicionan la elección de la mejor solución a nuestra decisión para alcanzar algún objetivo. Expresado en términos matemáticos se trata de optimizar una función objetivo (la decisión), sujeta ésta a una serie de restricciones (los recursos escasos o requisitos mínimos).

Los actuales intentos de Cuba de prosperar a partir de bases propias de desarrollo, se han extendido a todas las ramas de la economía a fin de lograr una pronta recuperación nacional. Para lograr este objetivo se hace imprescindible adoptar tecnologías que se encuentran adecuadas a las necesidades de equilibrio con el estado científico técnico del

mundo contemporáneo. Debido a esto, las estrategias y políticas seguidas por el estado en el campo de la información, orientan el desarrollo de las técnicas de computación al logro de elevados objetivos económicos y sociales en todas las manifestaciones. Dichas técnicas se encuentran llamadas a convertirse en la ciencia sobre la cual se erija necesariamente la sociedad. Por tanto, la introducción de éstas en el desarrollo económico social debe elevar el nivel de vida y cultural del pueblo, provocar la apertura de nuevas líneas de investigación y desarrollo, así como incrementar y diversificar la producción y la consolidación de nuevas relaciones internacionales. De esta forma se deberán ir creando las condiciones para lograr satisfacer las crecientes necesidades materiales y espirituales del país de forma armónica y proporcional [15].

Cuba posee crecientes y exclusivas ventajas naturales para la explotación del níquel, cuyo aprovechamiento adquirió una singular importancia después de los años 90. La etapa inicial del desarrollo de esta industria se remonta a principios del siglo XX, pero su consolidación como parte de la estructura económica de la isla ocurrió cuando se construyó la planta de Nicaro, la cual aportó las primeras producciones de óxido de níquel en diciembre de 1943. Parte de los yacimientos niquelíferos más importantes del mundo se encuentran en Cuba, los cuales representan aproximadamente el 37.3% de las reservas mundiales. Los más conocidos se localizan en la provincia de Holguín, constituyendo los mayores los de la región de Levisa y Mayarí, y los más ricos en mineral los ubicados en la región de Moa. De menor importancia los hay en la bahía de Tazo (al este Moa), la Meseta de Cajalbana en la occidental provincia de Pinar de Río, y la Meseta de San Felipe (al centro de Camagüey).

Uno de los principales retos a los que se enfrenta esta industria actualmente es la expansión de su producción con técnicas que ahorren gastos de portadores energéticos y generen mayor eficiencia metalúrgica [88].

La caracterización y el mejoramiento de las condiciones operacionales utilizando técnicas computacionales forman parte de las estrategias planteadas en ese sentido.

1.3 Objetivo

La utilización racional de los recursos es una condición indispensable de cualquier proceso productivo que pretenda insertarse, con un mínimo de competitividad, en la economía contemporánea. En este sentido, los procesos de producción de níquel, no son la excepción dentro de la manufactura moderna.

Prácticamente desde sus orígenes, los estudios se han centrado en aspectos de la mineralogía, siendo el tema metalúrgico menos tratado por su complejidad y altos costos asociados a la automatización en general.

Al revisar y consultar tanto la realidad actual, como la literatura especializada, se evidencia que en la práctica productiva está presente la siguiente ***situación problémica***: existencia de un volumen considerable de datos de proceso, así como la ausencia de herramientas informáticas de análisis de los mismos, que conlleva a que la organización no posea un control efectivo sobre el comportamiento futuro de muchos de los procesos que están presentes en su funcionamiento, implicando que en muchas ocasiones no se adecuan a las variadas condiciones de trabajo, impidiendo a su vez la toma de las mejores decisiones.

Por otro lado, las técnicas tradicionales de operación de estos procesos requieren de una actualización para lograr un mayor grado de afinación y seguimiento. La época en que las materias primas no reportaban grandes costos quedó en el pasado; incidiendo hoy de manera significativa en los resultados contable-financieros de la empresa.

La situación anterior es provocada por la existencia del siguiente ***problema científico***: los sistemas actuales que se utilizan para la supervisión y el control de la Reducción de Mineral no soportan los elementos necesarios que permitan formular e implantar una estrategia para analizar el progreso actual y definir la dirección futura del comportamiento de ciertos procesos claves en su operación.

Para dar solución a este problema, se plantea la siguiente ***hipótesis***: la aproximación funcional mediante redes de funciones de base radial permite crear modelos que son capaces de aproximar con aceptable precisión problemas del mundo real.

Para validar la hipótesis anterior, como ***objetivo general*** del presente trabajo se propone: *mostrar cómo la combinación de nuevas metaheurísticas y su aplicación en el Proceso de Reducción de Mineral puede ser una alternativa de solución para predecir el comportamiento a corto plazo del consumo en los subsistemas energéticos.*

En vista a cumplimentar el objetivo anterior, se han trazado las siguientes tareas:

1. Realizar un análisis bibliográfico asociado a nuevos procedimientos y modelos que pueden ayudar a la resolución de problemas del mundo real.
2. Conocer la estrategia de desarrollo y aplicaciones de los nuevos modelos de redes neuronales basados en RBF que forman parte de las líneas de investigación del

Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

3. Definición y planteamiento del caso seleccionado.
4. Establecimiento y caracterización de las funciones objetivos.
5. Conformar los juegos de datos que servirán de base a los experimentos.
6. Desarrollo y aplicación de un método basado en algoritmos genéticos, para la predicción del índice de consumo de portadores energéticos.
7. Ejecución del proceso de aproximación funcional y análisis de los resultados obtenidos.

1.4 Estructura General

El documento está compuesto por 6 capítulos y un apéndice, cada uno de los cuales queda resumido como sigue:

En el Capítulo 2 se hace referencia a un conjunto de técnicas que se reúnen bajo el nombre de SoftComputing, y su empleo en el tratamiento de la información. Las metodologías que se agrupan en torno a ella suponen un cambio con respecto al razonamiento clásico y modelos aproximados en los que se basa la lógica booleana, los modelos analíticos, clasificaciones precisas, y búsquedas deterministas respectivamente. Posteriormente se describen las Redes de Funciones de Base Radial, que se han elegido como el elemento computacional, que en el nivel más inferior se encarga de realizar el procesamiento de los datos de entrada, para obtener las correspondientes salidas del módulo de Aproximación Funcional del modelo propuesto. Esta descripción comienza, detallando las bases estructurales y de funcionamiento de estas redes, continuando con un estudio de las propiedades que las caracterizan. Seguidamente se hace un repaso de los métodos que se recogen en la bibliografía para diseñar y optimizar este tipo de redes.

En el Capítulo 3 se realiza una descripción a modo de resumen del Proceso Industrial de Punta Gorda. Es importante, para poder llegar al planteamiento del problema, obtener la mayor cantidad posible de información sobre el funcionamiento del sistema objeto de estudio. Todos los elementos recogidos en este capítulo han sido aportados por documentos elaborados por la propia industria, así como de la experiencia adquirida por 60 años de explotación de yacimientos de níquel. Los niveles de automatización actual de Punta Gorda, han permitido conformar una base de datos sobre la operación tecnológica de todas sus áreas y es en este capítulo donde con más precisión se hace referencia a la misma. Esos datos nos permitirán un acercamiento cualitativo al proceso de Reducción de Mineral. Se incluye luego el planteamiento general del problema desde el punto de

vista funcional. El problema de aproximación de una función desconocida al que nos enfrentamos, se puede plantear como un problema de optimización. De esta forma se tiene un espacio de búsqueda donde cada uno de sus puntos representa una solución a nuestro problema de aproximación.

En el Capítulo 4 se detalla la filosofía de funcionamiento del modelo propuesto, de igual manera se hace referencia a uno de los mejores algoritmos multiobjetivo desarrollados, para luego describir el algoritmo utilizado en la aproximación funcional.

En el Capítulo 5 se presentan los experimentos realizados con el conjunto de datos aportados por la operación del proceso de Reducción de Mineral, con los cuales se pretende adquirir conocimiento en relación con el comportamiento de la variable objeto de la predicción.

En el Capítulo 6 se presentan las conclusiones obtenidas, los principales aportes y se establecen las posibles líneas de investigación.

Por último se enumera la bibliografía usada como consulta para alcanzar los conocimientos básicos necesarios sobre los diferentes temas a que se ha hecho referencia a lo largo de esta memoria.

1.5 Alcance y contribuciones

Como resultado de nuestro trabajo de investigación se pueden considerar las siguientes aportaciones

1. El desarrollo de una metodología que utiliza por primera vez técnicas de Softcomputing para solucionar de problemas de predicción realistas planteados en la industria del Níquel en Cuba.

2. El análisis de las diferentes variables que se miden en los sistemas de supervisión y control de la planta de hornos de reducción con la finalidad de formular el problema de la *predicción del índice de consumo de petróleo en cámaras*.

3. El desarrollo, la aplicación y el análisis de resultados de aproximación funcional un procedimiento híbrido cuyos parámetros se han determinado medinate el uso de varias de las técnicas asociadas al SoftComputing, entre las que destacan los algoritmos evolutivos para aproximación multiobjetivo y la selección de variables.

4. Los resultados obtenidos han sido altamente satisfactorios, corroborando que la formulación del problema planteado ha sido enfocada de manera adecuada. Se logró obtener una aproximación funcional con un margen de error de un 2.1% aceptable para la operación de la industria.

5. Relacionado con el trabajo presentado se pueden citar las publicaciones descritas en el anexo A.

Capítulo 2

Soluciones a problemas del mundo real

Este capítulo hace breve referencia a un conjunto de técnicas que se reúnen bajo el nombre de Softcomputing (computación flexible), y su empleo en el tratamiento de la información. Las metodologías que se agrupan en torno a ella suponen un cambio con respecto al razonamiento clásico y modelos aproximados en los que se basa la lógica booleana, los modelos analíticos, clasificaciones precisas, y búsquedas deterministas respectivamente. La idea principal de todas ellas es "explotar la tolerancia de la imprecisión, la incertidumbre, la verdad parcial, los procesos estocásticos y la aproximación con el fin de conseguir un problema más tratable, robusto, con una solución menos costosa y con una mejor relación con la realidad" [134].

Concretamente se estudiarán las redes neuronales artificiales, la computación evolutiva y la lógica difusa. Incluso se abordará alguna extensión de éstas, como son los métodos coevolutivos cooperativos o la hibridación de algunas de estas técnicas como sería el desarrollo de redes neuronales mediante algoritmos evolutivos.

La capacidad del ser humano para predecir el comportamiento de su entorno, se ha ido incrementando con el paso del tiempo. De igual modo, ha comprendido que, si bien era capaz de controlar muchos aspectos de su vida, y su interacción con lo que le rodeaba, no lo era para otros tantos.

La inteligencia artificial es responsable de muchos de esos logros. Los pioneros de esta ciencia estaban tan interesados en la electrónica, como en la biología, y por eso sus aplicaciones iban desde calcular trayectorias de misiles, a tratar de modelizar el cerebro, de imitar el proceso de aprendizaje humano, y de simular la evolución biológica.

Los años ochenta en el siglo pasado marcan el florecimiento del interés de la comunidad científica por estos temas computacionales inspirados en la biología, que han visto como su desarrollo les llevaba a cotas inimaginables, primero en el campo de las Redes Neuronales, luego en el del Aprendizaje, y por último en lo que ahora se conoce como "Computación Evolutiva", de la que los algoritmos genéticos constituyen su máximo exponente [82].

El abundante trabajo de investigación con estas metodologías ha dado origen a un rápido aumento en la variedad y en la comprensión de los denominados sistemas inteligentes, con sus correspondientes aplicaciones tanto en productos para consumo doméstico, como en aplicaciones industriales.

2.1 Una Taxonomía para el Softcomputing

Varios son los trabajos presentados en relación a este conjunto de metodologías. Desde el primer artículo publicado con relación al Softcomputing hasta nuestros días, el concepto inicial ha sufrido muchos cambios y actualizaciones, pero se puede definir de manera muy general con la siguiente afirmación: *"es una asociación de metodologías de computación centradas en la Neurocomputación, en el cálculo mediante Algoritmos Genéticos y en la Lógica Difusa"* [66].

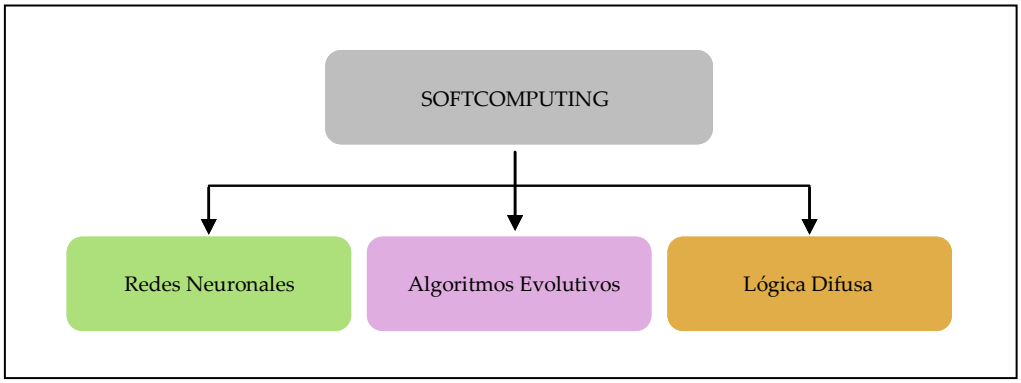


Figura 2-1: Una Taxonomía para el Softcomputing

Las redes neuronales son estructuras computacionales que pueden ser entrenadas para aprender una serie de patrones a partir de unos ejemplos, y posteriormente proporcionar resultados sobre nuevos datos.

Los algoritmos genéticos, metodología propuesta por Holland, nos proporcionan un mecanismo de búsqueda global en el espacio de posibles soluciones. En este espacio, una población de soluciones candidatas (posibles soluciones), codificadas como cromosomas,

son evaluadas mediante una función de aptitud (fitness) con relación a su comportamiento; las mejores soluciones evolucionan, pasando sus características a sus sucesores.

La lógica difusa, metodología introducida por Zadeh [85], nos proporciona un lenguaje, con una sintaxis y una semántica, que podemos utilizar para expresar nuestro conocimiento sobre el problema a resolver. Su principal característica es su robustez, el mecanismo de razonamiento que utiliza y lo fácil de entender por un ser humano.

Uno de los principales objetivos del Softcomputing es proponer y ofrecer una base para la concepción, diseño y aplicación de sistemas inteligentes que empleen metodologías inspiradas en el comportamiento de sistemas biológicos, y no metodologías aisladas de la realidad natural de los problemas que se intentan resolver.

Los sistemas convencionales de tratamiento masivo de la información han visto aumentada su capacidad de cálculo gracias a los avances obtenidos en microelectrónica, y en el diseño de computadores de carácter específico, lo que deriva a sistemas más potentes, obteniendo resultados con una precisión muy elevada; demasiado elevada si lo comparamos con la imprecisión innata del mundo real.

En este contexto, hay dos aspectos que han de ser considerados. Por un lado, existen muchos problemas que debido a su propia naturaleza, no pueden ser resueltos con las técnicas convencionales, principalmente porque o no se tiene la información necesaria para modelar el sistema o no se conoce lo suficiente. Este tipo de problemas es la norma general en el estudio de sistemas sociales a gran escala, en la determinación de planes económicos, y sistemas de toma de decisión en procesos industriales.

Otro tipo de problemas que presentan esta característica son aquellos típicos en inteligencia artificial, especialmente en donde se intenta utilizar un tipo de razonamiento cercano al humano.

El otro motivo del uso de las metodologías de Softcomputing, y quizás el más importante, es que, "en contraste al sistema tradicional de tratamiento de la información mediante ordenadores, Softcomputing es tolerante con la imprecisión, la incertidumbre y la verdad parcial" [25].

Softcomputing imita la capacidad humana de tomar decisiones en un entorno de incertidumbre e imprecisión.

2.2 Redes Neuronales Artificiales

2.2.1 Introducción

Una Red Neuronal Artificial (RNA) consiste en un conjunto de elementos de proceso, también conocidos como neuronas o nodos, los cuales se van a interconectar según distintos modelos.

Las RNAs suponen un nuevo modo de computación que propone un modelo de procesamiento distribuido y paralelo.

En este modelo, los elementos de proceso o neuronas realizan una función muy simple, suele existir gran cantidad de éstas y están masivamente interconectadas. Estas interconexiones van a estar caracterizadas por unos pesos que regulan los envíos de señales o datos, entre neuronas.

Una RNA adquiere su habilidad o se prepara para desarrollar su cometido, aprendiendo mediante ejemplos o patrones. Este aprendizaje consiste básicamente en la adaptación de los pesos que definen las interconexiones.

Así, una RNA, va a venir definida por su arquitectura (topología y celda) y su algoritmo de aprendizaje. La computación neuronal presenta además una serie de ventajas como por ejemplo: potencial para la computación paralela masiva, robustez ante la presencia de ruido o el fallo de alguno de sus componentes, capacidad de adaptación mediante aprendizaje, etc.

Todas estas ventajas implican que las redes neuronales se apliquen o se puedan aplicar a prácticamente cualquier tipo de ciencias existentes en el mundo actual.

2.2.2 Fundamentos

Los fundamentos de las RNAs se encuentran en el cerebro humano o lo que es lo mismo en las redes neuronales naturales. El cerebro humano está compuesto por un gran número de neuronas, unos 10.000.000.000, las cuales están masivamente interconectadas.

Cada neurona es una celda especializada que puede propagar una señal electroquímica. Una neurona natural, figura 2-3, consta de tres partes:

1. Una estructura de entrada muy ramificada, denominándose a estas ramificaciones dendritas.
2. Un cuerpo.

3. Una estructura de salida, también ramificada en su parte final, denominada axón.

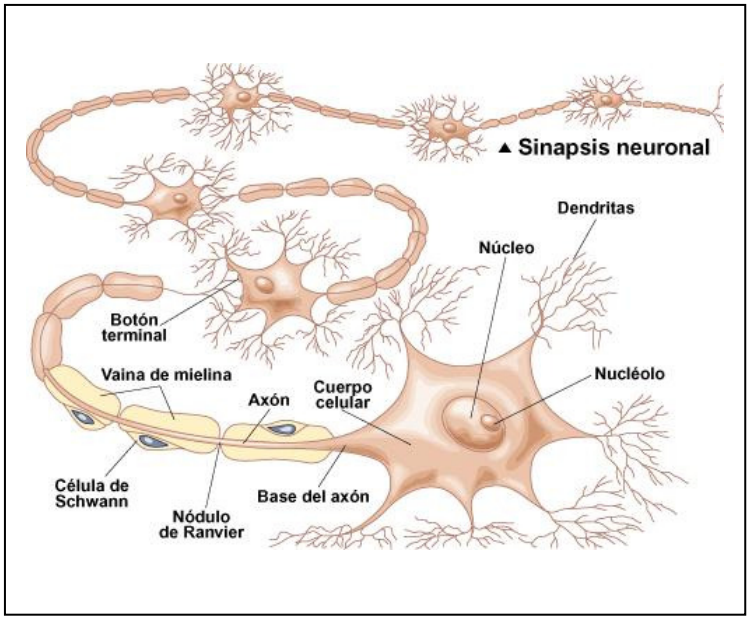


Figura 2-2: Neurona natural

El axón de una celda se conecta a las dendritas de otra mediante la sinapsis. Cuando una neurona se activa, envía una señal electro-mecánica a través del axón. Esta señal atraviesa la sinapsis de otras neuronas, y dependiendo de ciertos factores, puede a su vez activarlas. Una neurona se activa sólo si el total de señal recibida en su cuerpo, por las dendritas, supera un cierto nivel, también conocido como umbral de activación.

Por último resaltar la importancia de la sinapsis en este funcionamiento, ésta se podría describir como un espacio con neurotransmisores químicos que regularían la transmisión de la señal de una neurona a otra. De hecho, y según diversas investigaciones, durante el proceso de aprendizaje humano, se van determinando convenientemente estas sinapsis.

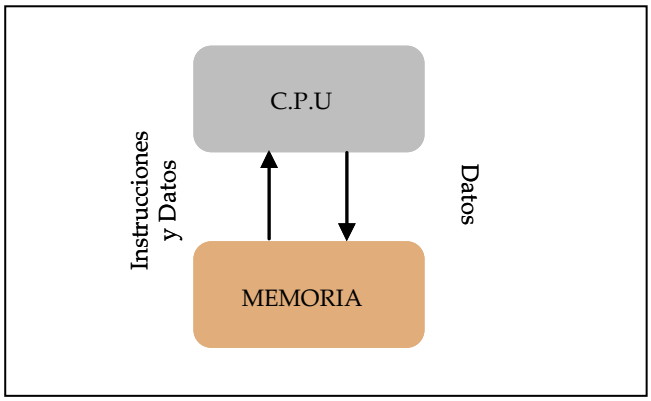


Figura 2-3: Máquina de Von Newman

2.2.2.1 Computación neuronal frente a la computación tradicional

La computación tradicional se va a identificar con aquella que realiza la conocida como máquina de Von-Newmann. Esta máquina, cuyo esquema se observa en la figura 2-4, ejecuta repetidamente los pasos del Algoritmo 2-1.

1. Cargar una instrucción de memoria.
2. Cargar de memoria los datos requeridos por la instrucción.
3. Ejecutar la instrucción (procesar los datos).
4. Guardar los resultados en memoria.
5. Volver al paso 1.

Algoritmo 2-1: Funcionamiento de la máquina de Von-Newmann

Este modelo de procesamiento de información implica que las computadoras tradicionales deban ser explícitamente programadas para resolver un problema, es decir, alguien debe de haber analizado el problema en profundidad y haber definido la serie de instrucciones (programa o algoritmo) que el computador debe seguir. El problema, usualmente tiene asociados un conjunto de datos en una estructura definida, y la computación consistirá en la manipulación de los datos dirigida por el programa.

Otra característica de esta computación es que hay una clara correspondencia entre las estructuras de datos tratados (números, matrices, registros,…) y el hardware de la máquina, guardándose cada uno en un bloque de memoria determinado.

Además la degradación o destrucción de unas pocas localizaciones de memoria suelen provocar que el programa deje de funcionar.

La computación neuronal difiere bastante de este tipo de computación. Desde el punto vista del procesamiento, se cuenta con un gran número de elementos de proceso, masivamente interconectados. A la hora de resolver problemas, el usuario debe permitir que la red adapte, por si misma, durante un periodo de aprendizaje. Este aprendizaje está basado en patrones, de modo que éstos, se le van pasando a la red, y los pesos de las interconexiones se van ajustando. Después del suficiente entrenamiento, la red está preparada para ofrecer soluciones viables a nuevos problemas.

Este tipo de computación presenta además como característica una robustez ante ruido en las entradas o ante ciertos fallos de hardware, de modo que estos factores afectan poco o muy poco al conjunto de los resultados.

Todo esto implica también cierta diferencia en los tipos problemas que resuelven mejor un tipo de computación u otra. A groso modo, la computación tradicional está especialmente indicada en problemas donde la solución se consigue ejecutando una serie de pasos, aquellos que tienen una solución matemática clara.

Por otro lado, será más idóneo usar computación neuronal en problemas que no están bien definidos, donde existen ambigüedades, y gran cantidad de información, por ejemplo en el reconocimiento de patrones.

2.2.3 Arquitectura

La neurona artificial es el elemento de proceso de este sistema de computación y en una red podemos encontrar cientos o miles de éstas. Una neurona o nodo, figura 2-4, va a constar de una serie de entradas (las cuales suelen tener asociado un peso), una salida y viene caracterizada por una función de activación.

Una neurona recibe datos o señales por sus entradas, las cuales pueden estar conectadas a las entradas del sistema o a otras neuronas. Estos datos suelen ser multiplicados por los pesos correspondientes, para emular el efecto de la sinapsis en las neuronas naturales. Así:

$$\eta = \sum_{i=1}^{n} w_i x_i \qquad 2\text{-}1$$

donde x_i es la entrada y w_i el peso. Partiendo de estos datos y dependiendo de la función de activación se obtendrá un valor u otro en la salida.

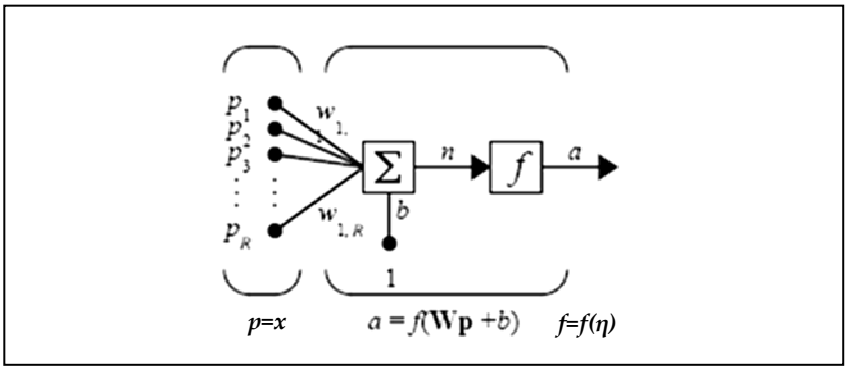


Figura 2-4: Neurona artificial

Existen funciones de activación con diferentes formas. Inicialmente se pude partir de una función básica que sólo tome dos valores en función de si se supera o no un umbral por parte de las entradas. Una de las funciones de activación más clásicas fue propuesta por [89] y su salida será igual a:

$$f(\eta) = \begin{cases} 1 & si \quad \eta > \theta \\ -1 & si \quad \eta < \theta \end{cases} \qquad 2\text{-}2$$

Denominándose a esta función paso (step-function ó hard-limiter), siendo θ el umbral de la función de activación.

Otra de las funciones de activación más conocida es la sigmoidal, la cual emula bastante bien el comportamiento de las neuronas naturales y utiliza la siguiente fórmula:

$$f(\eta) = \frac{1}{1 + e^{-(\eta - \theta)}} \qquad 2\text{-}3$$

Las redes utilizadas en este trabajo usan como función de activación una gausiana, cuya salida es la típica campana de Gauss. En este caso el concepto de función de activación cambia un poco, pues la salida de está depende de un centro c y un radio σ.

$$f(x) \quad = \quad e^{\{-(|x-c|/\sigma)^2\}} \qquad 2\text{-}4$$

En este caso tanto x, como c, pueden tener varias dimensiones y $|\cdot|$ es la norma euclídea.

2.2.4 Principales modelos de Redes Neuronales

Los distintos modelos de red neuronal pueden clasificarse de acuerdo con cuatro criterios básicos [59]:

1. Naturaleza de las señales de entrada y salida
2. Topología de la red
3. Mecanismo de aprendizaje que utilizan
4. Tipo de asociación de las señales de entrada y salida y la forma de representar estas señales.

Las distintas posibilidades de presentarse estos aspectos junto con las distintas funciones de activación y transferencia nos permiten la clasificación de los distintos modelos.

2.2.4.1 De acuerdo con su Naturaleza: Analógicas, Discrtetas, Híbridas

De acuerdo con la naturaleza de las señales de entrada y de salida podemos clasificar las redes neuronales en analógicas, discretas (generalmente, binarias) e híbridas.

Las redes analógicas procesan datos de entrada de naturaleza analógica, valores reales continuos, habitualmente acotados y usualmente en el compacto [-1,1] o en el [0,1], para dar respuestas también continuas. Las redes analógicas suelen presentar funciones de activación continuas, habitualmente lineales o sigmoides. Entre estas redes neuronales destacan las redes de Backpropagation, la red continua de Hopfield, la de Contrapropagación, la Memoria Lineal Asociativa, la Brain-State-in-Box, y los modelos de Kohonen (mapas auto-organizados (S.O.M.)y Learning Vector Quantizer, (L.V.Q.) .

Las redes discretas (binarias) procesan datos de naturaleza discreta, habitualmente {0,1}, para acabar emitiendo una respuesta discreta. Entre las redes binarias destacan la Maquina de Boltzman, la Máquina de Cauchy, la red discreta de Hopfield, el Cognitrón y el Neogognitrón.

Las redes híbridas, procesan entradas analógicas para dar respuestas binarias, entre ellas destacan el Perceptrón, la red Adaline y la Madaline.

2.2.4.2 De acuerdo con su Topología: Monocapa y Multicapa

Las neuronas de una RNA suelen estar masivamente interconectadas y se suelen agrupar en capas que indican su situación con respecto a las entradas o salidas de la RNA. Así las neuronas de la capa de entrada o las neuronas de la capa de salida tienen una conexión directa con la entrada o salida a la red, mientras que el resto de las neuronas se engloban dentro de la que se denominan capa ocultas.

El número de neuronas y capas que se necesitan dado un determinado problema dependen tanto del problema como del modelo de RNA que se esté utilizando.

Por lo que hace a la topología de la red, las redes pueden clasificarse de acuerdo con el número de capas o niveles de neuronas, el número de neuronas por capa y el grado y tipo de conectividad entre las mismas. La primera distinción a establecer es entre las redes Monocapa y las Multicapa.

Las redes Monocapa sólo cuentan con una capa de neuronas, que intercambian señales con el exterior y que constituyen a un tiempo la entrada y salida del sistema. En las redes Monocapa (red de Hopfield o red Brain-State-in-Box, máquina de Boltzman, máquina de

Cauchy), se establecen conexiones laterales entre las neuronas, pudiendo existir, también conexiones autorrecurrentes (la salida de una neurona se conecta con su propia entrada), como en el caso del modelo Brain-State-in Box.

Las redes Multicapa disponen de conjuntos de neuronas jerarquizadas en distintos niveles o capas, con al menos una capa de entrada y otra de salida, y, eventualmente una o varias capas intermedias (ocultas).

2.2.4.3 De acuerdo a la asociación de las entradas/salidas: Hacia delante y Hacia atrás

Normalmente todas las neuronas de una capa reciben señales de otra capa anterior y envían señales a la capa posterior (en el sentido Entrada - Salida). A estas conexiones se las conoce como conexiones hacia delante o feedforward. Si una red sólo dispone de conexiones de este tipo se la conoce como red feedforward.

Sin embargo, puede haber redes en las que algunas de sus neuronas presenten conexiones con neuronas de capas anteriores, conexiones hacia atrás o feedback. En tal caso hablaremos de una red feedback o interactiva.

Entre las primeras destacan los distintos modelos de Kohonen, aunque presentan conexiones laterales y autorrecurrrentes, el Perceptrón (multicapa) o M.L.P., las redes Adaline y Madaline, la Memoria Lineal Adaptativa y las Backpropagation.

Entre las segundas debemos mencionar el Cognitrón y el Neocognitrón, junto con los modelos de Resonancia y las máquinas multicapa de Boltzman y Cauchy.

1. Redes hacia delante (feed-forward): donde los datos o señales en la red viajan en un solo sentido desde las entradas, hasta la salida. Es decir, no existen ciclos y la salida de una capa no afecta a ella misma. Son ampliamente usadas en el reconocimiento de patrones.

2. Redes hacia atrás (feed-back): en este caso los datos o señales viajan en ambas direcciones, existiendo bucles en la topología. Pueden llegar a ser muy poderosas, pero también son más complicadas, dinámicas y cambiantes en cuanto a su punto de equilibrio.

2.2.4.4 De acuerdo a su Mecanismo de Aprendizaje: Supervisado, No supervisado, Por refuerzo

El entrenamiento de la red es muy importante ya que servirá para que posteriormente la respuesta del sistema sea la adecuada. Ello tiene mucho que ver con el aprendizaje humano. Ejemplo: cuando a un niño se le ordena coger un vaso, empieza moviendo el brazo de forma cuasi-aleatoria hasta que choca con el vaso y lo presiona con sus dedos. La próxima vez que se le ordene, éste alcanzará el vaso con mayor soltura y precisión. Este mismo modelo se ha ensayado en redes neuronales de características similares a las del niño.

Una vez que el brazo mecánico choca con la pieza y memoriza la secuencia, en posteriores ocasiones al brazo le cuesta menos realizar la misma operación se dice entonces que el sistema adquirió experiencia.

Mediante el aprendizaje, la RNA adquiere su habilidad para desarrollar su labor. El aprendizaje en RNAs se realiza mediante ejemplos.

Durante esta fase, también llamada entrenamiento, se ajustan los pesos de interconexión entre neuronas, para cada uno de los patrones del conjunto de entrenamiento, los cuales se dice que se van memorizando.

Básicamente el aprendizaje se puede dividir en tres tipos:

1. Aprendizaje supervisado (Supervised learning): se basa en la comparación directa entre la salida actual de la red y la salida correcta (valor objetivo) para un determinado patrón. Esta diferencia se formula muchas veces en términos de minimización del error, por lo que se utilizan algoritmos de optimización basados en gradiente descendiente para ajustar iterativamente los pesos de la red. Este tipo de entrenamiento es el más usado de los tres.

2. Un caso particular sería Entrenamiento reforzado (Reinforcement learning) donde mediante ciertas heurísticas se refuerzan ciertos comportamientos o salidas de la red.

3. Entrenamiento no supervisado (Unsupervised learning): en este caso no existe un valor objetivo que sirva de referencia para comparar la salida de la red, sino que la adaptación de la red se produce solamente utilizando los patrones de entrada. A este tipo de entrenamiento también se le suele denominar entrenamiento

competitivo (competitive learning), cuando se suelen considerar neuronas más importantes a aquellas que tienen más peso.

Conceptos implicados en el entrenamiento supervisado de una red son los de generalización, sobre-entrenamiento (overfitting) y bajo-entrenamiento (underfitting). Durante este entrenamiento las salidas de una red tienden a aproximarse a los valores objetivos de los patrones del conjunto de entrenamiento.

El objetivo final de ese comportamiento será conseguir la que la red generalice, es decir, que la red obtenga salidas próximas a valores objetivos para patrones que no pertenecen al conjunto de entrenamiento. Esta generalización no es siempre posible y para que se produzca se necesita tener cierto conocimiento a priori sobre la aplicación [136].

Por ejemplo, es necesario escoger un conjunto de patrones que sea relevante para que la red aprenda o que el conjunto de entrenamiento tenga cierta relación con el conjunto de patrones a generalizar.

Dos de los problemas con los que se encuentra la generalización son el bajo-entrenamiento y el sobre-entrenamiento [45]. Así una red que no es suficientemente compleja puede fallar cuando se le pasa un conjunto de datos complicado dando lugar a bajo-entrenamiento. Una red que es muy compleja puede llegar incluso a quedarse con ruido asociado a los patrones dando lugar a sobre-entrenamiento.

El sobre-entrenamiento es especialmente peligroso, en ciertas redes, ya que puede conducir a predicciones que están lejos del rango del conjunto de entrenamiento incluso aunque no haya ruido asociado a los patrones.

2.3 Computación evolutiva

Los principios fundamentales de los modelos de computación evolutiva se encuentran en la obra de Charles Darwin "El origen de las especies" que escribe en 1859. En este trabajo se introducen conceptos como individuo de una especie, nacimiento y desaparición de éste, lucha por la supervivencia, selección natural o herencia de un individuo a sus descendientes.

Todos estos conceptos se implementan a la hora de resolver un problema mediante computación evolutiva [9; 13]. Para esto, se mantiene una población finita de individuos que luchan por sobrevivir, donde cada individuo suele representar una solución al problema a resolver. La supervivencia de un individuo va a depender de cómo de buena

sea la solución que este simboliza, para resolver el problema. Esto se representa asociando a cada individuo un valor de adaptación (fitness), que se calcula mediante una función de evaluación. En función de estos valores se suelen ordenar los individuos, para posteriormente seleccionar un conjunto de estos. La obtención de descendientes de la población se va a conseguir aplicando a los miembros del conjunto, anteriormente citado, una serie de operadores. Para finalizar de entre los antiguos y nuevos miembros de la población se escogerán los miembros de la siguiente generación.

El uso de modelos de computación evolutiva tiene una serie de ventajas sobre otros métodos de resolución de problemas. En principio, se pueden aplicar cuando se tiene un conocimiento limitado sobre el problema a resolver. Por ejemplo, al contrario que otros métodos y a la hora de optimizar funciones no es necesario conocer de éstas, características como derivadas, discontinuidades, etc. Lo único que tenemos que poseer es una forma de medir para cada individuo, lo que se ha definido como valor de adaptación, o como de buena es la solución al problema que éste representa.

La segunda ventaja es que la computación evolutiva es menos susceptible de quedar atrapada en mínimos locales. Esto se debe a que esta computación mantiene una solución de poblaciones alternativas y un balance entre la explotación de regiones del espacio de búsqueda, donde se han establecido ya individuos, y la exploración de nuevas regiones.

La tercera ventaja es que la computación evolutiva puede ser aplicada en el contexto del ruido o de funciones objetivos no estacionarios. Esta ventaja hace que este modelo de computación se pueda aplicar a la resolución de problemas en un amplio rango de dominios, particularmente cuando el objetivo es construir un sistema que exhiba características bioinspiradas como inteligencia o adaptación a un contexto cambiante.

2.3.1 Orígenes y primeros desarrollos

La mayoría de las actuales implementaciones de algoritmos evolutivos descienden de tres desarrollos independientes pero relacionados: algoritmos genéticos, programación evolutiva y estrategias de evolución.

Los algoritmos genéticos fueron introducidos por Holland [68] y posteriormente estudiados por [27] y [52]. Originalmente fueron propuestos como un modelo general para procesos adaptativos, aunque después, sus aplicaciones más importantes se han desarrollado en el campo de la optimización. Para lograr esto, los algoritmos genéticos modelan el proceso evolutivo, ya esbozado anteriormente, a nivel del genoma. Antes de

que un algoritmo genético se use para resolver problemas, es necesario definir un código genético (representación de los individuos mediante cadenas de ceros y unos) y una relación entre el código genético y las soluciones del problema. Usando terminología biológica, al código genético de un individuo se le denomina genotipo y a la instanciación de este código, es decir a la solución que este representa, se le denomina fenotipo.

La programación evolutiva, introducida por [41], fue originalmente concebida como un intento de crear inteligencia artificial. Esta aproximación consistía en desarrollar máquinas de estados finitos para predecir eventos a partir de observaciones iniciales. Este tipo de máquinas abstractas transforman una secuencia de símbolos de entrada en una secuencia de símbolos de salida. Esta transformación depende de un conjunto finito de estados y de un conjunto finito de reglas de transformación entre estados. La eficiencia de una máquina de estados finita con respecto a su entorno puede medirse como la capacidad de predicción alcanzada.

Las estrategias de evolución fueron desarrolladas por [114] y [126]. Inicialmente fueron diseñadas con el objetivo de resolver problemas de optimización de parámetros. En este caso también se modela un proceso evolutivo, pero al contrario de los algoritmos genéticos que realizaban esta modelización a nivel del genotipo, las estrategias de evolución trabajan a nivel del fenotipo debido a los problemas para las que fueron concebidas.

2.3.2 Resolución del problema de optimización

En algunas situaciones del mundo real la función a optimizar f y las restricciones c_j no pueden ser tratadas analíticamente porque no se cumplan unas precondiciones (continuidad, diferenciabilidad, etc), por ejemplo si la definición de la función está basada en un modelo de simulación.

La aproximación tradicional en estos casos es desarrollar un modelo formal, que emule lo mejor posible las funciones originales, aplicándose más tarde métodos tradicionales como programación lineal o no-lineal. Esto conlleva, a que a menudo, se tenga que simplificar la formulación del problema original por lo que uno de los principales aspectos de la programación matemática será el diseño del modelo formal.

Esta aproximación, se ha mostrado satisfactoriamente en muchas aplicaciones, pero tiene ciertos problemas, sobre todo debido a las simplificaciones que se deben realizar, lo que implica que, muchas veces, las soluciones no resuelvan el problema original, llegando a

considerarse muchos problemas irresolubles. Es por esto, que se buscan nuevas aproximaciones donde la computación evolutiva es una de las más prometedoras.

El diseño de un algoritmo evolutivo que resuelva un problema de optimización concreto, se puede considerar como un problema de búsqueda en un espacio, donde cada punto representa una solución. Así, dado algún criterio de calidad, como complejidad, eficiencia, etc., el nivel en que todas las soluciones, cumplan este criterio formará una superficie en el espacio. El diseño de una solución óptima es equivalente a buscar el punto más alto (bajo) en esta superficie.

Una de las principales diferencias de la computación evolutiva es que el método empleado se debe adaptar al problema en sí, tal y como se mostrará en posteriores apartados. Pero incluso en el caso de que mediante las técnicas tradicionales de optimización, ésta sea imposible, mediante computación evolutiva se puede llegar a mejorar el conjunto de mejores soluciones conocidas lo que supone un gran éxito para problemas prácticos.

Otra opción es usar algoritmos híbridos: combinaciones de búsqueda evolutiva y tradicional como en las técnicas de búsqueda basadas en el conocimiento [26] y [115].

2.3.3 Estructura de un algoritmo evolutivo

Los algoritmos evolutivos emulan el proceso de la evolución natural, o proceso que conduce la emergencia de estructuras orgánicas complejas y adaptadas al entorno. De forma resumida y simplificada, la evolución es el resultado de la interrelación entre la creación información genética nueva, su evaluación y selección.

Un individuo de la población se relaciona con otros individuos de la población (por ejemplo al competir por la comida, mediante depredación o cruce para reproducirse) además de con su entorno (por ejemplo con factores como la comida o el clima). Mientras más grande sea la adaptación bajo esas condiciones, mayor será la probabilidad de que sobreviva durante más tiempo y de que genere descendientes que hereden parte de su información genética. A lo largo de la evolución esto conduce a una penetración de individuos que tengan una información genética cuyo valor de adaptación sea mayor. La naturaleza no determinística de la reproducción conduce a una permanente producción de información genética nueva y por lo tanto a la creación de descendientes diferentes.

Este modelo neo-Darwiniano de evolución orgánica se refleja en la estructura del Algoritmo 2-2. En este algoritmo G(t), es una población de μ individuos en la generación

t. Esta población será primeramente inicializada, siguiendo algún, método más o menos aleatorio.

1. t = 0.
2. Inicializar G(t)
3. Evaluar G(t)
4. Mientras no se cumpla la condición de fin
   a. P(t) = Seleccionar individuos para reproducción {G(t)}
   b. D(t) = Aplicar operadores {P(t)}
   c. Evaluar descendientes {D(t)}
   c. G(t+1) = Reemplazar [G(t) ∪ D(t)]
   d. t = t + 1

Algoritmo 2-2: Algoritmo evolutivo general

Después se evalúan los individuos de esta población, calculando su valor de adaptación a partir de una función de evaluación *f(x)*, para cada una de las soluciones *x*, representadas por cada uno de los individuos.

Posteriormente se entra en el bucle evolutivo principal del algoritmo. Este comienza, con la creación del conjunto de individuos *P(t)*, a los cuales se les va a aplicar un operador de reproducción. La pertenencia de un individuo a este conjunto se suele determinar en función del valor de adaptación de éste.

El número de individuos de *D(t)* es variable y puede ser igual al de *P(t)*. *D(t)* representa la población de descendientes, su tamaño es λ, y se genera mediante operadores que actúan sobre la información genética. Algunos de los operadores más conocidos son el del cruce o el de mutación que se aplicarán sobre individuos de la población *P(t)*.

Se pasará después a evaluar a los miembros del conjunto *D(t)*. Una vez determinado el valor de adaptación de estos individuos, se aplica una estrategia de reemplazo, que consistirá en sustituir algunos miembros del conjunto *G(t)* por algunos de los miembros del conjunto *D(t)*. Este reemplazo se suele realizar utilizando criterios como el valor de adaptación de los individuos u otros más o menos aleatorios.

2.3.4 Diseño de un algoritmo evolutivo

Tal y como se ha mencionado existen tres tipos principales de algoritmos evolutivos: algoritmos genéticos, programación evolutiva, y estrategias de evolución. Sin embargo y a partir de estas tres clases de algoritmos evolutivos, se pueden derivar muchas variantes. Las principales diferencias estriban en:

1. La representación de los individuos.
2. El diseño de operadores de reproducción.
3. Los mecanismos de selección y reemplazo.

Estas diferencias serán también los elementos clave a la hora de diseñar un algoritmo evolutivo. Así, a la hora de resolver un problema mediante computación evolutiva una de las tareas principales será elegir un esquema de representación para los individuos. Este esquema se diseñará en función del problema e influirá en los operadores de reproducción que más tarde se diseñarán. La importancia de estos operadores reside en el hecho de que serán los encargados de generar nuevos individuos o soluciones al problema dado. Por último resaltar también el papel de los mecanismos de selección y reemplazo dentro de la evolución de una población. De estos mecanismos, va a depender temas más importantes como la elección del subconjunto de padres o el mantenimiento de la diversidad de la población, y por lo tanto de su capacidad de búsqueda de nuevas soluciones.

Es necesario resaltar que los requerimientos para el diseño de aplicaciones evolutivas son modestos comparados con otras técnicas de búsqueda, siendo esta una de las principales ventajas de la computación evolutiva y una de las razones por la que se ha hecho tan popular, aunque el costo computacional es alto.

2.3.4.1 Inicialización de la población inicial

La creación de la población inicial se puede generar de forma aleatoria o mediante heurísticas, que van a situar a los individuos iniciales "mejor colocados", dentro del espacio de búsqueda.

En el caso de los algoritmos genéticos, existen estudios [92] que aseguran que, partiendo de cualquier inicialización, usan una estrategia casi óptima para alcanzar una solución en un tiempo finito. Sin embargo es evidente que una buena inicialización, en la que se utiliza conocimiento a priori del problema, ahorrará bastantes generaciones en el proceso de búsqueda o podrá disminuir el tamaño de la población. En conclusión se obtendrá una solución final con un menor coste de computación y tiempo.

Por otro lado, la aplicación de este tipo de heurísticas para la creación de la población inicial, debe realizarse con cierta precaución. Así, será necesario vigilar que ésta sea suficientemente diversa, o de lo contrario será fácil quedar atrapado en un óptimo local.

2.3.4.2 Condición de parada

El criterio de parada se establece cuando se da por concluido el proceso de búsqueda. Un criterio de parada ideal sería aquel que estudia la evolución de una población y detecta cuando se ha alcanzado una solución óptima. Una vez alcanzado este estado, no tiene sentido continuar la búsqueda, ya que no se va a mejorar la adaptación de la población. Este momento, es pues el adecuado para detener el proceso y ofrecer la solución más óptima encontrada.

Entre las condiciones de parada más usadas se encuentran:

2.3.4.3 Criterio del número máximo de generaciones

Este criterio termina el proceso de búsqueda cuando se han completado v generaciones. Dado un algoritmo evolutivo, diseñado para resolver un problema determinado, el valor de v se puede determinar de una manera empírica, de forma que pueda usarse en posteriores ejecuciones para la búsqueda de soluciones para el mismo problema.

Dentro del paradigma de los algoritmos genéticos es donde se encuentran más estudios teóricos sobre cual debe ser el número de generaciones adecuado, para un problema determinado. Así y según [1], este número dependerá del tamaño de la población. De este modo, si tenemos una población de tamaño n, y suponiendo una función de evaluación bien diseñada, el umbral para el número de generaciones sería:

$$v = log_2(n) \qquad 2\text{-}5$$

Otros autores sin embargo [12] fijan este umbral en función de la longitud del cromosoma. Así para un cromosoma de tamaño l se fija un número de generaciones de:

$$v = \mathrm{O}(\sqrt{l}) \qquad 2\text{-}6$$

En general y para cualquier paradigma de la computación evolutiva, el número de generaciones va a depender mucho del problema a tratar. Por tanto, estos se deben tener en cuenta sólo a un nivel orientativo, ya que un número insuficiente de generaciones puede que no llegue a proporcionar una buena solución. Por otro lado, si se sobrestima el valor de v, se emplean más generaciones de las necesarias que no mejoran la solución final, produciendo un algoritmo costoso en tiempo y en cálculo.

2.3.4.4 Criterio del número máximo de generaciones sin mejorar

Otra posibilidad para detener la ejecución del algoritmo consiste en hacer un estudio del estado de convergencia de la población durante la ejecución del algoritmo. De esta forma se puede terminar la búsqueda cuando se estime que la población ha convergido a una buena solución. Hay varias formas de hacer este estudio. Una de las más sencillas es contabilizar el número de generaciones en las que no se ha mejorado la aptitud del mejor cromosoma de la población y detener la búsqueda si este número de generaciones sobrepasa cierto límite v_0, propuesto a priori [117].

Al igual que en el criterio anterior, se debe escoger un valor adecuado de v_0, de forma que no se realicen más generaciones de las necesarias, lo que también va a depender del problema que se esté resolviendo.

2.3.5 La representación

En las aplicaciones del mundo real, el espacio de búsqueda se define mediante un conjunto de objetos, que poseen unos parámetros que están sujetos a optimización y constituyen lo que se denomina el espacio fenotipo. De otra parte los operadores evolutivos trabajan con objetos matemáticos abstractos, como cadenas binarias que representan a los objetos anteriores, y que determinarían el espacio genotipo. Obviamente se necesita una función de codificación entre el espacio del genotipo y el del fenotipo.

Los algoritmos genéticos clásicos usan, para un individuo, una representación binaria consistente en cadenas de longitud fija que utilizan el alfabeto {0, 1}, y a las que se les suele denominar cromosoma. La aplicación de una función de evaluación f, a una de estas cadenas nos devolverá el valor de adaptación de la solución que este individuo representa.

$$f : \{0,1\}^l \rightarrow \Re \qquad 2\text{-}7$$

Junto a la representación binaria, a menudo es de obligado uso la utilización de funciones de codificación y decodificación $h: M \rightarrow \{0, 1\}^l$ y $h': \{0, 1\}^l \rightarrow M$ que facilita la conversión de soluciones $x \in M$ a cadenas binarias $h(x) \in \{0, 1\}$ y a la inversa. En el caso de optimización de parámetros continuos, los algoritmos genéticos representan un vector real $\bar{u} \in \Re n$, mediante una cadena binaria $\bar{o} \in \{0, 1\}l$ de la siguiente manera: la cadena binaria se divide lógicamente en n segmentos de igual longitud l', por lo que $l = n \cdot l'$. Cada segmento es decodificado en su correspondiente parámetro real.

La preferencia por usar una representación binaria de las soluciones en algoritmos genéticos se deriva de la teoría de esquemas [68], donde se analizan los algoritmos genéticos en general, en función de esta teoría. El término esquema denota un patrón de similaridad que representa un subconjunto de {0, 1}l. En esta teoría, el teorema de esquemas de algoritmos genéticos afirma que los algoritmos genéticos canónicos proveen de una estrategia casi óptima para alcanzar una solución en tiempo finito [92] .

Por otro lado, también existen trabajos [36] y [73] en los que se concluye que esta codificación binaria de variables continuas, tiene ciertas desventajas. Según estos trabajos la función de codificación puede introducir cierta multimodalidad y hacer más complejo la resolución del problema en sí.

Existe otro tipo de estudio teórico sobre el funcionamiento de los algoritmos genéticos. En este estudio [99] se modelan los algoritmos genéticos mediante la teoría de cadenas de Markov. En esta aproximación se profundiza en las propiedades de convergencia de estos y en su conducta dinámica, desarrollando los modelos ejecutables que facilitan la simulación directa de los algoritmos genéticos mediante cadenas de Markov, para problemas de dimensión suficientemente pequeña.

Al contrario de los algoritmos genéticos, la representación en estrategias de evolución y programación evolutiva se basa directamente en la utilización de vectores reales que se corresponden directamente con los parámetros a optimizar.

$$f : M \subseteq \Re^n \rightarrow \Re \qquad 2\text{-}8$$

Además y para problemas reales con complejos espacios de búsqueda, cuya representación no es tan directa, se desarrollan otras variantes de representación.

Todo esto implica que los métodos de representación en estrategias evolutivas y programación evolutiva sean menos estándar y que existan menos estudios sobre estas representaciones que en algoritmos genéticos.

Una ventaja que si tiene la representación que utilizan estos métodos es que su función de codificación es más simple y por lo tanto, conlleva menos problemas asociados.

Si comparamos ambas aproximaciones, para la representación mediante cadenas de bits existen trabajos, tanto empíricos como teóricos que avalan la primera aproximación.

Por otro lado, la utilización de una función de codificación compleja en esta representación, puede introducir no-linearidades adicionales y otras dificultades matemáticas que pueden dificultar el proceso de búsqueda sustancialmente.

En cuanto a la representación mediante vectores de reales hay más resultados empíricos que teóricos que demuestran buenos resultados. Además su función de codificación, al ser más simple, ocasiona menos problemas de los citados anteriormente.

No hay pues, una respuesta general a la pregunta de cual de las dos aproximaciones mencionadas seguir dado un proyecto específico, pero muchas aplicaciones en la práctica han mostrado que se pueden obtener buenas soluciones después de imponer modificaciones a las representaciones tradicionales [92].

2.3.5.1 Operadores de reproducción

Los operadores de reproducción van a servir para crear individuos nuevos en la población, a partir de un subconjunto seleccionado de entre los individuos actuales. Con este, y otros elementos, se pretende mantener una diversidad en la población que explore nuevas zonas del espacio de búsqueda.

De modo que si los nuevos individuos aportan mejores soluciones, sustituirán a los ya existentes, con mucha probabilidad.

2.3.5.2 Mutación

Es uno de los operadores clásicos de variación de la información genética. En general el diseño de estos operadores debe de obedecer a las propiedades matemáticas de la representación elegida, aunque hay ciertos grados de libertad.

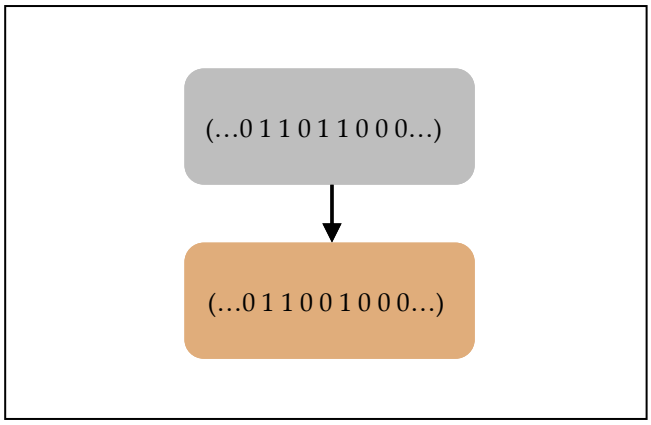


Figura 2-5: Mutación de un bit para un individuo en un algoritmo genético

La mutación en los algoritmos genéticos, inicialmente se desarrolla como un operador secundario, de pequeña importancia. En estos algoritmos este operador trabaja invirtiendo bits con probabilidad p_m, donde p_m suele ser un valor pequeño. Así es normal escoger $p_m \in [0.005, 0.01]$ o $p_m = 1/l$, donde l es la longitud de los cromosomas. Este valor además puede variar a lo largo de la evolución de manera que mejore los resultados de ésta. En [10] se decrementa el valor de p_m lo que ayuda en la convergencia y velocidad del algoritmo genético. En el apartado Adaptación de parámetros, se estudian la adaptación de parámetros en general en un algoritmo evolutivo.

Originalmente, la mutación en la programación evolutiva fue implementada como un cambio o cambios aleatorios en función de la descripción de las máquinas de estados finitos de acuerdo a cinco diferentes modificaciones: cambio de un símbolo de salida, cambio de un estado de transición, adición de un estado, eliminación de un estado o cambio de un estado inicial.

Las mutaciones fueron realizadas con probabilidad uniforme, y el número de mutaciones para un descendiente fue prefijado o elegido de acuerdo a una distribución de probabilidad. Hoy en día los esquemas de mutación frecuentemente utilizados son similares a los utilizados en las estrategias de evolución.

En estas estrategias, los individuos están formados por variables objeto. La mutación se realiza entonces independiente en cada vector elemento sumando un valor aleatorio que sigue una distribución normal con media 0 y desviación típica σ.

Así la mutación del valor de una determinada variable se realizaría de la siguiente forma:

$$x'_i = x_i + N(0, \sigma) \qquad 2\text{-}9$$

El control del tamaño de la mutación σ, también se trata en el apartado Adaptación de parámetros.

Para terminar, resaltar la importancia de este operador en las estrategias de evolución y en la programación evolutiva, hasta tal punto de que muchas veces, este es el único operador considerado.

2.3.5.3 Cruce

Este operador es uno de los más importantes en los algoritmos genéticos clásicos. Estos algoritmos proponen como operador de cruce clásico, el operador de cruce en un punto, que consiste en elegir dos individuos de la población aleatoriamente, determinar una posición en la cadena de bits, también aleatoriamente, denominada punto de cruce y generar el descendiente. Para esto, se concatena, la subcadena izquierda de un padre, con la subcadena derecha del otro padre.

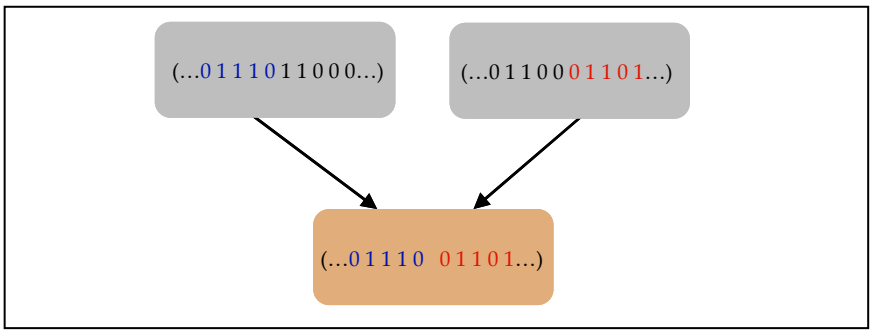


Figura 2-6: Cruce de dos individuos en un algoritmo genético

Existen además, numerosas variaciones de este operador, como por ejemplo: incremento del número de puntos de cruce [35], cruce uniforme [131] (donde cada bit del descendiente es elegido aleatoriamente de los padres), etc. En los algoritmos genéticos el operador de cruce se aplica normalmente con una probabilidad p_c, donde $p_c \in [0.75, 0{,}95]$.

En el caso de las estrategias de evolución, la utilización de este operador no está tan clara y no existen modelos de operadores predefinidos como en el caso de los algoritmos genéticos. En estas estrategias tras aplicar este operador en el bucle principal del Algoritmo 2-4, se genera una población intermedia de λ individuos, tras aplicar λ veces el operador a la población padre y crear un descendiente por aplicación a partir de δ padres ($1 \leq \delta \leq \mu$), donde μ es el tamaño de la población. Normalmente $\delta = 2$.

Existen trabajos también con operadores de cruce que utilizan múltiples padres, donde más de dos individuos, participan en la generación de un descendiente, representando esta generalización mejoras en la eficiencia en ciertas aplicaciones.

Los tipos de operador de cruce, para vectores reales, no son tan estándares y dependen más de la aplicación, aunque existen algunos ejemplos comunes [13] como por ejemplo cruce discreto, que consiste en la elección aleatoria de las variables de los padres, similar al cruce uniforme en algoritmos genéticos. Otro ejemplo sería cruce intermedio donde se utilizan medias aritméticas de los padres. Evidentemente, también se puede utilizar otros

operadores de cruce. El análisis teórico de este tipo de operador es un campo abierto de investigación.

Tal y como se ha comentado, dada una aplicación es difícil elegir a priori un operador de cruce patrón, con unos parámetros más o menos estándar con el que vayamos a tener éxito en la evolución de nuestra población, demostrándose que existen situaciones en las que puede llegar a darse divergencia de la población sino se eligen bien los parámetros de éste [80].

Con respecto a la programación evolutiva, basándose en la formación y semántica de los individuos se puede afirmar que este modelo no usa cruce. Según [40], antes de centrarse en un mecanismo de cruce, se debe examinar y simular su efecto funcional e interpretar una cadena de símbolos como una reproducción de la población.

2.3.5.4 Selección / Reemplazo

El mecanismo de selección se puede considerar como un operador que en vez trabajar con la información genética de los individuos trabaja con su valor de adaptación. En algoritmos genéticos, la selección se implementa como un operador de probabilidad, así dado un individuo ai su probabilidad de selección es:

$$p(a_i) = f(a_i) / \sum_{j=1}^{\mu} f(a_j) \qquad 2\text{-}10$$

denominándose a este tipo de selección, selección proporcional o ruleta. Este método requiere valores de adaptación positivos y una tarea de maximización, utilizándose funciones de escalado cuando sea necesario transformar el valor de adaptación convenientemente. Otro tipo de selección es la selección basada en ranking, que no usa valor de adaptación directamente sino el índice de individuos ordenados en función del valor de adaptación, para calcular la probabilidad de selección. La selección tournament trabaja tomando una muestra uniforme aleatoria de cierta cantidad $q > 1$ de la población, seleccionando el mejor de esos q individuos para sobrevivir en la próxima generación, repitiendo el proceso hasta que la población de descendientes se complete. Este método es muy usado porque, es fácil de implementar, computacionalmente eficiente, y permite ajustar la presión selectiva aumentando o disminuyendo el valor de q. Una descripción de los principales métodos de selección, incluyendo una caracterización de su presión selectiva en términos de medidas numéricas se encuentra en [11].

En las estrategias de evolución se utilizan los esquemas de selección / reemplazo (μ, λ). La notación (μ, λ) indica que μ padres crean λ hijos mediante recombinación y mutación, siendo $\lambda > \mu$. Los mejores μ individuos descendientes son determinísticamente seleccionados para reemplazar a los padres. Hay que destacar que con este esquema el mejor miembro de la población en la generación *t+1* puede tener peor valor de adaptación que en la generación *t*, a este tipo de método se le denomina no elitista, ya que puede permitir un deterioro temporal. Esta estrategia se lleva a cabo para ayudar salir de regiones de atracción de óptimos locales y alcanzar así óptimos globales. Por otro lado las estrategias (μ, λ) selecciona los μ supervivientes de entre la unión entre los padres y los descendientes, de esta manera el curso de la evolución es monótona

La utilización de la estrategia (μ, λ) o ($\mu + \lambda$) depende de la aplicación práctica con la que se esté trabajando, para de esta manera obtener mejores resultados. Además ambos esquemas se pueden interpretar como instancias de la estrategia general (μ, κ, λ) donde κ representa la vida máxima de un individuo en generaciones, pudiendo tomar valores entre $1 \leq \kappa \leq \infty$. Si κ=1 entonces el método de selección sigue la (μ, λ), mientras que si $\kappa = \infty$, entonces se sigue la estrategia ($\mu + \lambda$) [126].

Una pequeña diferencia entre la programación evolutiva y las estrategias de evolución consiste en una variante probabilística de la estrategia de selección ($\mu + \lambda$) en la programación evolutiva, donde cada individuo, fuera de los individuos padres o descendientes se evalúa frente a un número q, con $1 \leq q \leq 10$, de otros individuos pertenecientes al conjunto unión de padres e hijos. En cada una de estas comparaciones se determina un individuo ganador, que será aquel cuyo valor de adaptación sea mejor o igual que el de todos sus oponentes. Los μ individuos que hayan sido más veces ganadores serán los padres de la siguiente generación. Como se muestra en [13], este método de selección es una versión probabilística de la estrategia ($\mu + \lambda$) que es más determinística cuando el número q de competidores se incrementa. La determinación de si es mejor un esquema de selección más probabilístico que uno más determinístico es un campo de investigación abierto.

2.3.6 Adaptación de parámetros

El concepto de adaptación está implícito en la computación evolutiva. Su implicación más directa sería en el plano de encontrar la solución a un problema dado, aunque también aparece a la hora de ajustar los parámetros del algoritmo que resuelva el problema en cuestión [3] [33] [128].

Para resolver un problema en computación evolutiva, no solo se necesita elegir el algoritmo, la representación o los operadores a usar, sino también elegir o configurar los parámetros (probabilidad de operadores, condición de parada, etc.) que van a terminar de caracterizar nuestra estrategia general. El ajuste de la configuración o parámetros, puede determinar en gran medida la eficiencia final del método.

Tradicionalmente este proceso se ha realizado de manera manual lo que puede consumir muchos recursos, por lo que se buscan soluciones donde se automatice este proceso.

La ejecución de un algoritmo evolutivo es intrínsicamente un proceso adaptativo y dinámico. El uso de parámetros fijos, parece contradecir el espíritu evolutivo general de un algoritmo de este tipo. Además existen otra serie de inconvenientes a la aproximación tradicional:

1. Un error del programador al establecer los parámetros puede lugar a errores o alcanzar unos resultados no óptimos.
2. El ajuste de parámetros consume grandes cantidades de tiempo.
3. El conjunto de parámetros óptimo puede variar a lo largo de la ejecución.

Por lo tanto, parece natural pensar en modificar ciertos parámetros a lo largo de la ejecución de un algoritmo evolutivo. Esta tarea se realizará utilizando reglas, posiblemente heurísticas, que, o analizan lo ocurrido antes del estado actual, o emplean algún mecanismo auto-adaptativo. Estos cambios pueden afectar desde a una parte de un individuo, hasta a toda la población.

2.3.6.1 Diseño de parámetros adaptativos

A la hora de abordar un estudio sobre el diseño de un mecanismo adaptativo para los parámetros que intervienen en un algoritmo evolutivo es necesario fijar ciertos aspectos clave como:

1. Elemento/s cuyos parámetro/s se van a adaptar.
2. Tipo de adaptación a realizar. Estos tipos pueden ser: determinísticos, basados en el transcurso de la evolución o auto-adaptativos.
3. Nivel o alcance al que se realiza la adaptación: nivel de individuo, de población, etc.

Comenzando por el primer aspecto, a la hora de diseñar un algoritmo evolutivo es necesario elegir elementos o componentes de éste como: representación de los individuos, función de evaluación, operadores, etc. Para cada uno de estos elementos, y

dependiendo también del diseño, se va a poder elegir adaptar uno o varios parámetros. Así por ejemplo podemos pensar en adaptar según distintas variables de estado, el valor de la función de evaluación, la probabilidad de aplicación de los operadores, etc. Una completa revisión de este tipo de adaptaciones se encuentra en [33] .

En cuanto al tipo de adaptación a realizar y según [33], existen tres tipos principales:

1. Determinístico: este tipo de adaptación se produce cuando un parámetro se altera siguiendo una regla determinística. Esta regla modificará el parámetro sin usar ninguna variable de estado o realimentación del algoritmo evolutivo en sí. Este tipo de reglas suelen usar para su funcionamiento, el tiempo o generaciones transcurridas. Por ejemplo se podría actualizar la probabilidad de mutación de un algoritmo evolutivo p_m, de la siguiente manera:

$$p_m = 0.5 - 0.3 \cdot g / G \qquad 2\text{-}11$$

donde g es la generación actual y G es el número total de generaciones. De este modo la probabilidad de mutación variará de 0.5 a 0.2, conforme aumente el número de generaciones. Los primeros ejemplos de uso de este tipo de adaptación aparecen en [39].

2. Adaptativo: en este caso la adaptación se produce en función de ciertas variables de estado o existe cierta realimentación a partir del algoritmo evolutivo. Esta asignación puede influir en la asignación de crédito y su efecto se podrá propagar por toda la población. Ejemplo de este tipo de adaptación sería la regla de éxito 1/5 de [114] en las estrategias de evolución (1+1), cuya misión era variar el valor de la mutación. En este contexto se utilizan mutaciones gausianas que vienen caracterizadas por dos parámetros: la media que suele establecerse a cero y la desviación típica σ que puede interpretarse como el tamaño de la mutación, de este modo la mutación de un valor x_i vendría expresada cómo en(1-9), donde N(0, σ) es un número aleatorio gausiano con media 0 y desviación típica σ. La regla de éxito 1/5 establece que el porcentaje de mutaciones con éxito p_s, o mutaciones que producen un hijo mejor que el padre, debe ser 1/5 de todas las mutaciones, incrementándose el tamaño o desviación típica de la mutación si p_s es mayor de 1/5, y disminuyendo en caso contrario, lo que se expresa como:

Si (t mod n = 0) Entonces

$$\sigma(t) = \begin{cases} \sigma(t-n)/c & si \quad p_s > 1/5 \\ \sigma(t-n)\cdot c & si \quad p_s < 1/5 \\ \sigma(t-n) & si \quad p_s = 1/5 \end{cases} \qquad \text{2-12}$$

si no

$$\sigma(t) = \sigma(t-1)$$

1. La adaptación de σ, ocurrirá cada n iteraciones y $c \in [0.817, 1]$ según [13]. Usando este mecanismo se adapta el valor de σ, en función del estado de la ejecución del algoritmo evolutivo.

2. Auto-adaptativo: este tipo de adaptación se caracteriza porque se va a ver afectada por un proceso de evolución. Así por ejemplo, los parámetros que se van a adaptar se van a codificar, y se van a adjuntar al genotipo del individuo. De este modo, sobre estos parámetros también actuarán los operadores que a éste se le apliquen. Estos parámetros codificados no afectan al valor de adaptación del individuo directamente, pero mejores valores de adaptación conducirán a mejores individuos y estos individuos será más probable que sobrevivan y produzcan descendientes, y por lo tanto se propaguen los mejores parámetros. Un ejemplo de este método se puede encontrar en [126] donde se adapta el tamaño de la adaptación.

Por otro lado, si se considera ahora, el nivel o alcance de la adaptación realizada, se concluye que va a depender del componente del algoritmo evolutivo donde se realice la adaptación. Así, por ejemplo un cambio en el tamaño de la mutación, puede afectar a un gen, a un cromosoma, o a la población entera, de un algoritmo genético, dependiendo de la representación en particular.

Así, una adaptación de un coeficiente de la función de evaluación, siempre afectará a toda la población. Por lo tanto el nivel o alcance de una adaptación es secundario y usualmente, depende del componente tratado y de su implementación.

Por extensión se puede considerar el hecho de controlar o adaptar varios parámetros durante la ejecución de un algoritmo evolutivo. Esto, evidentemente, tiene el inconveniente de un aumento en las dependencias de los componentes elegidos, de los parámetros a adaptar, etc., y por lo tanto implica una mayor complejidad en el estudio.

Sin embargo se han realizado varios estudios satisfactorios, donde mediante la heurística se consiguen mejores resultados [67] [127].

Como conclusión, el campo de la adaptación de parámetros es un campo que presenta ciertos inconvenientes, sobre todo relacionados con el hecho de que los resultados dependen mucho de las elecciones hechas en la fase de diseño del algoritmo evolutivo. Esto implica que los resultados obtenidos dependen de la heurística y que no existen suficientes resultados teóricos o experimentales que permitan extraer conclusiones o estrategias generales, válidas para los algoritmos evolutivos en sí.

Sin embargo, se demuestra que con la adaptación de parámetros, se pueden conseguir mejoras de resultados, aunque sea a nivel local o del algoritmo evolutivo que se esté tratando. Para conseguir esto, lo normal es utilizar reglas basadas en la experiencia.

2.4 Modelos coevolutivos cooperativos

2.4.1 Introdución

Dentro de la computación evolutiva aparece como un nuevo paradigma, los modelos coevolutivos cooperativos [109]. Estos modelos surgen para intentar solucionar problemas, en los que la computación evolutiva tradicional encuentra ciertas complicaciones. Existen tres clases principales de estos problemas:

1. La primera clase incluiría problemas en los que se requiere, múltiples y distintas soluciones, como en el caso de la optimización de funciones multimodales.

2. La segunda clase la compondrían aquellos problemas que tienen una solución compuesta por subcomponentes más pequeños y especializados, como por ejemplo en sistemas basados en reglas o en redes neuronales artificiales. A este tipo de problemas se les suele denominar problemas de cubrimiento (covering problems).

3. La tercera clase consistiría en problemas que se pueden descomponer en un determinado número de subtareas simples y que por lo tanto pueden ser resueltos usando la estrategia divide y vencerás. Este es el caso por ejemplo de problemas de planificación de rutas como en problemas del viajante de comercio, en el que se puede trabajar con rutas más simples. Otro ejemplo de aplicación se daría en el campo de aprendizaje de conductas que se presenta en el dominio de la robótica, donde conductas complejas pueden ser descompuestas en subconductas más simples.

Existen dos principales razones por las que los algoritmos evolutivos tradicionales tienen dificultades con este tipo de problemas.

1. La primera razón es que la población de individuos alcanzada con los algoritmos evolutivos tradicionales tiene una fuerte tendencia a converger. Esto se debe a que el trabajo de estos algoritmos se localiza en ciertas regiones del espacio de búsqueda donde el valor de adaptación medio es mayor. Esto, por ejemplo, supone una desventaja cuando se trabaja con problemas de optimización multimodal donde la solución necesita de más información que una simple zona del espacio. Esta convergencia también imposibilita el mantenimiento de subcomponentes coadaptados que se requieren en la resolución de problemas de cubrimiento o en la utilización de estrategias como divide y vencerás, ya que prácticamente, cualquier individuo, menos el más fuerte será eliminado.

2. La segunda razón es que un individuo de un algoritmo evolutivo tradicional representa una solución completa y se evalúa en solitario. Por tanto, partiendo de que las interacciones entre la población no se modelan, incluso aunque se mantuviera la diversidad en la población, el modelo evolutivo debería ser extendido para mantener conductas diferentes, que se puedan coadaptar.

La idea para solventar estos problemas es introducir en la computación evolutiva nociones sobre modularidad, para que de esta manera se puedan generar soluciones formadas por subcomponentes que coevolucionan, que se coadaptan y que interactúan cooperando. La dificultad está en buscar extensiones para los paradigmas evolutivos que permitan a los subcomponentes coevolucionar, sin ser diseñados a mano.

Para esto habrá que proveer un entorno que permita la identificación y representación de esos subcomponentes, donde además, éstos puedan interactuar, adaptarse, y donde la aportación de crédito de estos represente su contribución a la solución del problema. Todo esto se deberá realizar de la forma más independiente posible.

2.4.2 Diseño de algoritmos coevolutivos cooperativos

Uno de los aspectos básicos que deben ser tratados para resolver un problema mediante la coevolución cooperativa es determinar el número apropiado de subcomponentes que debe existir, así como el papel que cada uno debe desempeñar.

A esta tarea se le denomina descomposición del problema.

Esta descomposición se produce en un nivel macroscópico, teniendo en cuenta estrategias divide y vencerás, donde un problema complejo se divide en subtareas, que individualmente son más fáciles de resolver, a un nivel microscópico. La solución estará pues compuesta por un conjunto de subcomponentes simples.

Para algunos problemas puede ser más o menos sencillo conocer cual es su descomposición idónea. Por ejemplo en el caso de la optimización de una función de k variables independientes, una descomposición que a priori parece lógica, sería aquella en la que determinan k elementos independientes. Sin embargo, existen muchos problemas en que disponemos de poca o nula información sobre el número subcomponentes a obtener en la descomposición.

Por ejemplo en el problema de aprendizaje mediante reglas, será improbable que, a priori, se conozca el número de reglas necesario para cubrir el conjunto de ejemplos.

Así pues encontrar una descomposición adecuada no siempre es obvio y es muy importante trabajar este problema. En el caso ideal, el algoritmo debería de generar una descomposición adecuada como resultado de la evolución.

2.4.2.1 Interdependencias entre subcomponentes

Los subcomponentes en los que se descompone un problema son independientes cuando cada uno de ellos puede ser resuelto sin tener en cuenta a los demás. Gráficamente se podría representar de manera que cada subcomponente estaría intentado alcanzar la cima de su espacio de búsqueda independiente. Desafortunadamente, muchos problemas solo pueden ser descompuestos en subcomponentes entre los que existen interdependencias.

El efecto de cambiar uno de esos subcomponentes interdependientes es a menudo descrito, como una deformación del espacio de búsqueda de los otros subcomponentes. Por tanto es necesario que el diseño de nuestro algoritmo recoja esta característica.

2.4.2.2 Asignación de crédito

Cuando una tarea, es cubierta colectivamente por un conjunto de soluciones parciales, se llama asignación de crédito a la determinación de la contribución de cada solución parcial.

Uno de los principios de la evolución Darwiniana es que los individuos, cuyo valor de adaptación es mayor, van a pasar sus características de su genotipo a futuras

generaciones. Por tanto si el objetivo es desarrollar un modelo evolutivo para resolver problemas mediante una colección subcomponentes que cooperan, debe existir un proceso que mida la aportación de cada subcomponente a la solución alcanzada.

Esta medición va a depender del número de subcomponentes del problema, de las interdependecias de estos y en general del problema en sí.

2.4.2.3 Diversidad de la población

Cuando se usa un algoritmo evolutivo clásico, en los que se busca un individuo que represente una solución para un determinado problema, la diversidad en la población se mantiene durante un determinado tiempo en la ejecución, para realizar una exploración razonable del espacio de búsqueda. Esto se observa en la estructura de funcionamiento general de la computación evolutiva mostrada en el Algoritmo 2-2. En este algoritmo, y si se obvian los efectos de los operadores usados, se favorece la creación y sustento de individuos en ciertas zonas del espacio de búsqueda donde existe un valor de adaptación superior.

Este es el resultado de una continua presión selectiva que favorece la convergencia de la población.

Sin embargo, y cuando se soluciona un problema mediante una colección de subcompnentes que cooperan es necesario mantener la diversidad hasta el final. La diversidad en algoritmos evolutivos clásicos se introduce mediante los operadores usados.

Esta diversidad no es suficiente para los modelos de evolución cooperativa competitiva por lo que es necesario introducir otras estrategias que la mantengan. Así por ejemplo se pueden usar técnicas como la compartición del valor de adaptación, (fitness sharing) [51]o la técnica de nichos [5], para poblaciones simples, u otras que mantienen poblaciones aisladas, denominadas especies, cada una trabajando en una zona del espacio de búsqueda.

La técnica de fitness sharing fue desarrollada para mantener la diversidad en la resolución de problemas de funciones multimodales, y se basa en modificar convenientemente el valor de adaptación (fitness).

Así se define la siguiente función de compartición (sharing):

$$comp(d_{ij}) = \begin{cases} 1 \quad si \quad d_{ij} = 0 \\ 1 - \left(\dfrac{d_{ij}}{\sigma_s} \right) \quad si \quad d_{ij} < \sigma_s \\ 0 \quad en\ otro\ caso \end{cases} \qquad 2\text{-}13$$

donde dij representa la distancia entre dos individuos i y j, σs define un radio de grupo, y α controla la intensidad con la que varia esta función dependiendo de la distancia. La distancia se puede medir tanto en el espacio del fenotipo, como en el del genotipo. El valor de adaptación se modifica de la siguiente manera con la función de compartición:

$$f_i^{'} = \frac{f_i}{\sum_{j=1}^{n} comp(d_{ij})} \qquad 2\text{-}14$$

donde f sería el valor de adaptación inicial y n es el tamaño de la población. Cuando se aplica la compartición del valor de adaptación, la población se distribuye en un determinado número de agrupamientos de individuos alrededor de un máximo. A estas regiones se les denomina nichos. La determinación del valor de σ_s, va a depender del problema a resolver.

Posteriormente en el trabajo [5] se desarrolla la técnica secuencial de nichos, también para resolver funciones multimodales. Esta aproximación toma prestada, de la técnica de compartición del valor de adaptación, la idea de modificar el valor de adaptación basándose en distancias. Sin embargo, en vez estar continuamente modificando el valor de adaptación, basándose en la distancia entre individuos de la actual población, la técnica secuencial de nichos reduce el valor de adaptación de un individuo en función de su distancia a máximos encontrados en ejecuciones anteriores del algoritmo genético.

Esta técnica, si bien es menos costosa computacionalmente que la técnica de compartición del valor de adaptación, también es dependiente del valor de σs.

2.4.3 Trabajo previo

Las estrategias coevolutivas cooperativas son un paradigma nuevo dentro de la computación evolutiva, donde además las soluciones adoptadas dependen en gran medida del problema a resolver. Es por esto que no existen modelos de referencia que sirvan de base a la hora de abordar un nuevo problema. Por tanto se va a describir cuales son los trabajos más importantes realizados en este campo.

Una de las primeras extensiones al modelo evolutivo básico para soportar subcomponentes coadaptados es el sistema de clasificación de [69]. Este sistema de clasificación es un sistema basado en reglas, donde se desarrolla una población de reglas estímulo-respuesta usando un algoritmo genético. Las reglas individuales en la población trabajan juntas para formar la solución a un determinado problema. Estas reglas van a interaccionar entre ellas utilizando un modelo micro-económico y van competir por activarse. Para realizar la asignación de crédito se utiliza un algoritmo denominado bucket brigade que va a ir propagando un valor según una cadena de activación.

Otra aproximación al mantenimiento de una población de reglas coevolutivas se realiza en [49], mediante su sistema REGAL. Este sistema aprende reglas de clasificación a partir de ejemplos preclasificados. La descomposición del problema se realiza mediante un operador de selección, que agrupa a los individuos basándose en el cubrimiento que realicen de un subconjunto de ejemplos, elegido aleatoriamente. Una solución se forma seleccionando la mejor regla de cada conjunto. Un operador semilla mantiene la diversidad del ecosistema, creando los individuos apropiados, cuando no existe nadie, con las propiedades necesarias para cubrir algún elemento del subconjunto aleatorio. El valor de adaptación (asignación de crédito) de cada individuo en cada grupo es una función de la consistencia, con respecto a un grupo de ejemplos, y a su simplicidad.

Posteriormente la eficiencia de REGAL [48] se ha mejorado con el uso de islas de poblaciones semi-aisladas. A cada isla se asigna un conjunto de ejemplos preclasificados y se aplica el operador de selección y el operador semilla. Al final de cada generación ocurre una migración de individuos entre islas. La funcionamiento general se completa con la existencia de un proceso supervisor que maneja la asignación de ejemplos a las islas.

El trabajo de [135] es un ejemplo de desarrollo de redes neuronales, concretamente redes de funciones de base radial, mediante técnicas cooperativas competitivas. En este caso se utiliza un algoritmo genético básico al cual se le hacen las extensiones necesarias.

Así se parte de la población de individuos, correspondiendo un individuo a una neurona de la red, por lo que existe una función de evaluación que asigna un valor de adaptación a cada individuo/neurona. A estos individuos se les aplica la metodología evolutiva típica de un algoritmo genético. La selección, en este algoritmo, promueve de una parte, la competición entre individuos debido a la función de evaluación que se utiliza. De otra parte, también promueve la cooperación entre los mismos individuos que se van a

encargar de cubrir diferentes partes del dominio de actuación. El valor de adaptación se establece en función de la disposición espacial del vector de pesos.

Dentro del campo del diseño de redes neuronales artificiales otra de aproximación coevolutiva cooperativa es el sistema SANE [95]. En este sistema cada individuo en la población representa una neurona, especificando sus entradas, salidas y pesos asociados. Una red neuronal se constituye a partir de la selección de un conjunto de individuos. Por otro lado se mantiene una población aislada de redes (llamadas blueprints) constituidas por conjuntos de neuronas, que juntas, trabajan "bien".

La evaluación neuronal consiste en seleccionar neuronas de estos blueprints, formar redes, evaluar estas redes y asignar el valor de adaptación obtenido al blueprint.

El valor de adaptación de una neurona será la media de los valores de adaptación de las cinco mejores redes en las que participe. Como se puede observar es una medida de cómo una neurona trabaja con otras neuronas para la resolución de un problema. De esta manera se mantiene la población y se propicia formación de población de subcomponentes que cooperan.

Uno de los trabajos más importantes dentro de la coevolución cooperativa es el de modelo de [109]. En este trabajo propone un ecosistema donde existen dos o más especies aisladas. Cada especie funciona como una población individual donde se aplica el Algoritmo 1-4, o algoritmo evolutivo general instanciado a uno de los paradigmas clásicos. Así pues, entre los individuos de una población se producirá, reproducción, selección, etc, pero no existirá ninguna relación evolutiva entre los individuos de distintas especies.

Para obtener el valor de adaptación de los individuos de una especie, se establece una cooperación con representantes de las otras especies, determinando éste en función del resultado de esta colaboración. Existen diversos métodos para elegir los representantes de otras especies con los que se colaborará. En muchos casos el representante de una especie puede ser simplemente el que tenga mejor valor de adaptación.

También se puede elegir este representante de una forma más natural, escogiendo el representante de una manera aleatoria entre los mejores.

En [44] se presenta un interesante trabajo que usa la coevolución cooperativa. El modelo, llamado MOBNET, se utiliza para obtener tanto la topología como los pesos de una red. En este caso los subcomponentes con los que se trabaja son subredes que se combinan

para dar una red completa. La asignación de crédito a los subcomponentes se resuelve utilizando una técnica multiobjetivo donde se valoran eficiencia, complejidad, etc, de los subcomponentes. Los subcomponentes van evolucionando paralelamente en poblaciones aisladas. Una red completa se forma cogiendo un subcomponente de cada una de las poblaciones anteriores.

Además existe una población de redes completas que también van evolucionando.

2.5 Diseño evolutivo de redes neuronales artificiales

Las redes neuronales artificiales ofrecen un interesante paradigma para el diseño y análisis de sistemas inteligentes dentro del campo de la inteligencia artificial.

La extensa investigación realizada en el área ha conducido a lograr importantes resultados, tanto empíricos como teóricos, y al desarrollo de importantes aplicaciones prácticas. Sin embargo y normalmente, el diseño de una red neuronal para una determinada aplicación y bajo una serie de restricciones, suele ser un proceso heurístico, de prueba y error, basado en la experiencia con similares aplicaciones. Además, la eficiencia y complejidad de una red, para un problema particular, depende entre otras cosas, de la elección de las neuronas, la arquitectura de la red y el algoritmo de aprendizaje usado.

Por ejemplo se parte, de un típico algoritmo de aprendizaje, para establecer el conjunto de pesos de una red, dada su topología y un determinado conjunto de patrones de entrenamiento. Inicialmente, y para que este algoritmo tenga éxito, será necesario que este conjunto de pesos óptimo exista en el espacio de búsqueda determinado por las restricciones, fruto de las distintas elecciones realizadas (tipo de neuronas, topología de la red, etc). Por otro lado, evidentemente, el algoritmo de búsqueda debe de ser capaz, también de alcanzar este conjunto de pesos óptimo. Pero incluso aunque el algoritmo de búsqueda encuentre un conjunto de pesos adecuado, para el conjunto de patrones de entrenamiento, puede que la red no sea capaz de generalizar para patrones, que no se han usado durante el entrenamiento, o puede que la complejidad de la red obtenida no esté cerca de ser óptima. El conjunto de estos factores hacen que el diseño de una red neuronal artificial sea una tarea complicada.

Es natural, dadas estas premisas, que se investigue en técnicas para automatizar el diseño del conjunto de parámetros que define una red neuronal, para un problema determinado y bajo un conjunto de restricciones. Los algoritmos evolutivos ofrecen una aproximación atractiva y relativamente eficiente, para la resolución de este problema.

Estos devolverán, por su naturaleza, una solución casi óptima en una amplia variedad de dominios de problemas.

En un nivel abstracto, el problema del diseño de una red neuronal es esencialmente una instancia de los problemas que se resuelven en otras disciplinas de la inteligencia artificial [133] como el lambda cálculo, máquinas de Turing, etc.

Estas ofrecen aproximaciones para buscar soluciones a problemas complejos cuando ni la estructura, ni el tamaño de las soluciones se conoce de antemano. Sin embargo los algoritmos evolutivos parecen adaptarse mejor [4; 65] al diseño de estructuras de computo, masivamente paralelas, altamente interconectadas y con elementos de proceso relativamente simples, como son las redes neuronales. Incluso, un estudio sobre el diseño de una red neuronal conduce a una exploración de las respuestas a cuestiones fundamentales de interés de las neurociencias.

La evolución en las redes neuronales [4], [57], [65], [137]se suele realizar en tres niveles: evolución de los pesos, evolución de las arquitecturas, evolución de las reglas de aprendizaje.

Mediante la evolución de los pesos se pretende encontrar un conjunto de pesos casi óptimo, dada una determinada arquitectura. En este caso, como en otros, el paradigma evolutivo más utilizado son los algoritmos genéticos. Conviene también relacionar o comparar esta aproximación evolutiva con los algoritmos clásicos de entrenamiento basados en la minimización del error.

De la misma manera, con la evolución de la arquitectura de una red, se pretende conseguir que esta posea una arquitectura casi óptima. La arquitectura de una red va a determinar en gran medida la capacidad de proceso de ésta, lo que implica que este es uno de los campos de investigación más importante dentro del diseño de las redes neuronales.

Existen dos aspectos importantes en la evolución de las arquitecturas: la codificación de la arquitectura y el procedimiento de búsqueda a utilizar.

Por último, decir que el campo de la evolución de las reglas de aprendizaje es el menos investigado, ya que resulta complicado codificar de manera más o menos general estas reglas. Para solventar en cierta manera este problema, se suelen establecer una serie de restricciones.

2.5.1 Evolución de los pesos de una red

El aprendizaje de los pesos de una red, se formula usualmente como la minimización de una función de error, que suele venir dada como el error cuadrático medio total entre las salidas objetivo, dadas por los patrones de entrenamiento, y las salidas actuales ofrecidas por la red. La mayoría de estos algoritmos [65] de entrenamiento, como el de backpropagation o los algoritmos de gradiente conjugado, se basan en aplicar técnicas de gradiente descendiente de una manera iterativa. Estos algoritmos se aplican con éxito en ciertas ocasiones, aunque tienen ciertos inconvenientes debidos a su naturaleza. Uno de los inconvenientes más importantes es que pueden quedar atrapados en mínimos locales. Otro inconveniente es que su aplicación depende de la diferenciabilidad de la función. Así, en general la complejidad de estos métodos será proporcional al coste asociado a conseguir las derivadas de la función. En el caso más extremo, estos métodos no se pueden aplicar si la función no es diferenciable.

Una de las alternativas a las técnicas clásicas, y a sus inconvenientes intrínsecos, es formular este proceso de entrenamiento como la evolución de los pesos de la red, en el entorno determinado por la arquitectura. Con esto se tienen una serie de ventajas. La primera es que los algoritmos evolutivos proporcionan una estrategia global de búsqueda que devolverá un conjunto de pesos casi óptimos. El adjetivo de casi óptimos viene dado por la naturaleza en sí de los algoritmos evolutivos que devuelven una solución "cercana" al óptimo global. Otra ventaja es que el valor de adaptación de una red de este tipo, puede ser definido de acuerdo a diferentes necesidades. Además, y al contrario que en los algoritmos de entrenamiento basados en gradiente descendiente, la función del valor de adaptación no tiene por que ser diferenciable, ni incluso continua. Por otro lado, el mismo algoritmo evolutivo, se puede usar para entrenar diferentes tipos de redes neuronales: hacia delante, hacia atrás, etc.

Así mismo el entrenamiento evolutivo de una red, facilita la generación de redes neuronales con características especiales. Para esto, se suele tratar convenientemente la función de evaluación, o función que devuelve el valor de adaptación. De esta manera, se puede incluir por ejemplo en ésta, un término de penalización que disminuya la complejidad de la red y que aumente su capacidad de generalización.

Como inconveniente un entrenamiento evolutivo es generalmente más lento que las variantes más rápidas de un algoritmo back-propagation [38] o de un algoritmo de gradiente conjugado [93] , cuando la información de gradiente es fácil de obtener. A pesar de esto existen experimentos [111] en los que un algoritmo genético ha demostrado

ser más rápido que un algoritmo de gradiente descendiente. Este resultado se debe a la sensibilidad inicial de las condiciones iniciales

En resumen el uso de algoritmos evolutivos se adecúa a la hora de determinar los pesos de una red, ya que estos están especializados en trabajar con espacios de búsqueda grandes, complejos, que no tienen que ser diferenciables y que pueden ser multimodales, como son los espacios de búsqueda definidos por las funciones de error o las funciones definidas por el valor de adaptación.

2.5.2 Diseño del algoritmo evolutivo

En esta aproximación evolutiva para el entrenamiento de los pesos de una red, existen dos tareas principales. La primera consistiría en decidir la representación genotípica de los pesos de la red. En cuanto a la representación de los individuos de la población existen dos tipos:

1. Representación mediante cadenas de bits: esta representación sería similar a la utilizada en los algoritmos genéticos y por lo tanto está muy ligada al uso de estos como esquema general de evolución. Entre sus ventajas destacan la existencia de muchos estudios sobre este tipo de representación [68]. Su simplicidad y generalidad. La facilidad que ofrece a la hora de implementar operadores típicos de cruce o mutación. Entre sus inconvenientes se encuentran su pobre escalabilidad, por ejemplo, cuando la red a tratar posee muchas conexiones.
2. Representación mediante números reales: en este caso, el peso de una interconexión es representado directamente por un número real. El uso de esta representación implica que la utilización de operadores de reproducción estándar no sea tan directa y que haya que adaptarlos al esquema de representación empleado. Así un operador que se usa típicamente es un operador de mutación donde se suma una cantidad aleatoria gausiana. En cuanto a la implementación de un operador de cruce debe ser muy específica por lo que muchas veces, sólo se utiliza un operador de mutación. Todo esto implica que exista menos trabajo de referencia sobre este tipo de representación, lo que es un inconveniente. Como ventaja no se tendrían problemas de precisión, ni los acarreados por la conversión genotipo-fenotipo.

La segunda tarea sería decidir el modelo de evolución a utilizar. Partiendo del algoritmo general evolutivo Algoritmo 2-2, habría que fijar todos sus parámetros: operadores, esquema de selección, etc. Dependiendo de estas elecciones los resultados pueden ser

muy diferentes. En cualquier caso un individuo puede ser un conjunto de pesos de una red, o un peso de la red si se opta por una técnica coevolutiva cooperativa. Para evaluar estos individuos, lo normal será transformar los genotipos en fenotipos, aplicarlos a la red y medir el error que se produce. El valor de adaptación de cada individuo estará en función de este error.

2.5.2.1 Entrenamiento híbrido de los pesos de una red

Uno de los principales inconvenientes de los algoritmos evolutivos es que presentan cierta ineficiencia a la hora de afinar en una búsqueda local.

Por otro lado, hay que tener en cuenta que cuando se utiliza sólo un algoritmo matemático de minimización de error, se tiene que ejecutar varias veces en la práctica, partiendo de diferentes condiciones iniciales, para conseguir buenas soluciones.

Así pues, sería interesante mejorar la eficiencia de ambos, incorporando un procedimiento matemático de búsqueda local al final de una técnica de evolución. Con esto se conseguiría unir la habilidad de los algoritmos evolutivos para realizar búsqueda global con la característica de afinamiento de una búsqueda local.

os algoritmos evolutivos se usarían inicialmente para localizar una región del espacio idónea, donde debería encontrarse el extremo global.

Después, se aplicaría el procedimiento de búsqueda local para encontrar una solución, lo más cerca posible, a ese extremo global.

Como procedimiento de búsqueda local se puede usar un algoritmo de gradiente descendiente.

En la figura 2-7, se muestra un ejemplo de la importancia de partir de buenas condiciones iniciales, en un proceso de búsqueda del conjunto de pesos adecuado de una red.

Así si se parte del punto w_{i1} un algoritmo de gradiente descendiente tenderá a acabar y a dar como solución w_{f1}, mientras que si se parte de w_{i2}, la solución final aportada por el algoritmo será w_{f2}.

La misión, por lo tanto, del algoritmo evolutivo, será aportar como condiciones iniciales w_{i2}, en lugar de w_{i1}.

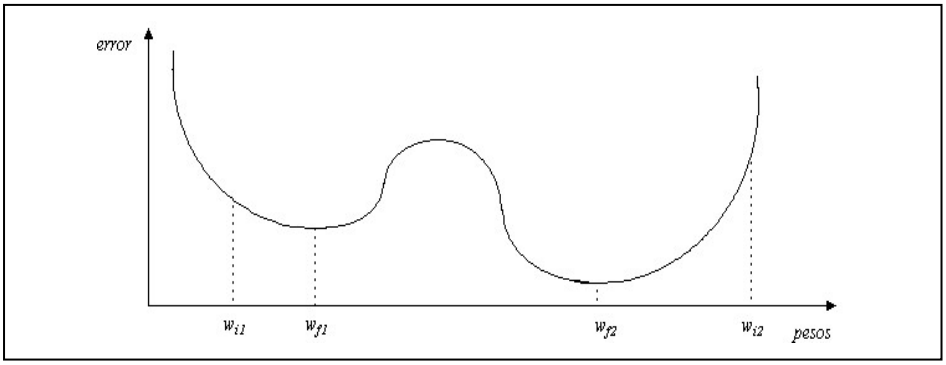


Figura 2-7: Espacio de búsqueda definido por conjunto de pesos de una red

En [6] se usa un algoritmo genético, que genera una búsqueda global que asigna a los pesos unos valores iniciales, y después se utiliza un algoritmo de backpropagation, que mediante una búsqueda local, afina la solución. En sus resultados se muestra que la aproximación genética/backpropagation es más eficiente que la utilización de un algoritmo genético o un backpropagation por separado.

2.5.3 Evolución de las arquitecturas

En el apartado anterior se ha descrito la técnica de evolución de los pesos de una red. En este caso se partía de que la arquitectura de esta red estaba prefijada y se mantenía fija durante toda la evolución. Ahora la idea es realizar un diseño evolutivo de la arquitectura de la red, lo que afectaría tanto a la conectividad de esta, como a las funciones de transferencia de cada nodo.

La arquitectura elegida o diseñada para una red neuronal es muy importante ya que va a definir las características de procesamiento de información de ésta y por tanto sus posibilidades y eficiencia. Dada una tarea de aprendizaje, una red neuronal con pocas conexiones y nodos lineales no será capaz de realizarla debido a las limitaciones en la capacidad de aprendizaje. Por otro lado, si la red tiene muchas conexiones y nodos no lineales, puede que incurra en problemas de sobreentrenamiento y poca generalización.

Hoy en día, el diseño de la arquitectura de una red neuronal suele ser un trabajo heurístico, que depende tanto del conocimiento experto de la persona que lo realiza como de procesos de prueba y error. No existe un método sistemático para diseñar una arquitectura de red para una aplicación determinada. Una aproximación al diseño automático de arquitecturas son los algoritmos constructivos o destructivos [38] [43]. En resumen, un algoritmo constructivo comienza con una red muy básica (con pocas capas ocultas, nodos y conexiones) y va añadiendo nuevas capas, nodos y conexiones conforme

van haciendo falta y durante el proceso de entrenamiento. Por otra parte un algoritmo destructivo hace lo contrario, es decir, empieza con una red muy compleja y va eliminando capas, nodos o conexiones que considera innecesarios.

Al igual que en el caso de la evolución de los pesos de una red, el diseño evolutivo de la arquitectura de ésta presenta una serie de ventajas [137] con respecto a otras técnicas. Estas ventajas están relacionadas con una mejor adaptación a las características del espacio de búsqueda que definen las arquitecturas de las redes neuronales. La primera característica es que la superficie de este espacio es infinitamente grande ya que el número de conexiones y nodos no está limitado a priori. Por otro lado la superficie es no diferenciable ya que los cambios en el número de nodos o conexiones son discretos y pueden tener un efecto discontinuo en la eficiencia de la red. Además la superficie es compleja y existe el ruido, debido a la conversión genotipo-fenotipo, y a la dependencia con el método de evaluación usado.

La superficie además da lugar a ambigüedades donde similares arquitecturas, puede tener diferentes eficiencias, y donde diferentes arquitecturas tienen una eficiencia similar.

2.5.4 Diseño del algoritmo evolutivo

Dentro del campo de la evolución de arquitecturas de redes neuronales [137], casi toda la investigación se refiere a la topología de la red, no habiéndose trabajado mucho en la evolución las funciones de transferencia de las neuronas.

En el diseño de un algoritmo evolutivo que obtenga una arquitectura de red adecuada existen dos aspectos principales, la representación del genotipo, y el proceso de evolución en sí.

En cuanto a la representación del genotipo, existen dos tipos principales [137]. Esta clasificación, que se describe a continuación, está basada en la cantidad de información que se va codificar.

1. Esquema de codificación directa o codificación fuerte. En este tipo de esquema se codifican prácticamente todos los detalles de la arquitectura de la red. Así por ejemplo se podría utilizar una matriz C, de $n \times n$ para codificar una red con n nodos, donde c_{ij}, indicaría la presencia o ausencia de conexión entre el nodo i y el j. Con este esquema, es sencillo extraer un sistema de codificación mediante cadenas de bits. Además permite establecer fácilmente restricciones sobre el tipo de arquitecturas permitidas, mediante el establecimiento de ciertas reglas. Este

esquema se adapta bien a arquitecturas pequeñas, o de pocos nodos y conexiones, pero no escala bien cuando las arquitecturas se hacen grandes. De todas formas, y partiendo de las restricciones de la red, también se puede disminuir el tamaño de las matrices.

2. Esquema de codificación indirecta o codificación débil. Ahora sólo se codifican los parámetros más importantes con los que se trabaja como por ejemplo: el número de capas, número de nodos por capa, etc. El resto de los detalles los decide el algoritmo de entrenamiento, mediante reglas de desarrollo, que indican por ejemplo, qué nodos se unen y como lo hacen. Este sistema de codificación más compacto y más cercano a la naturaleza, según las neurociencias. Existen diferentes formas de realizar esta codificación como por ejemplo:
   - Representación paramétrica: Consistiría en la codificación de los principales parámetros que definen una red como por ejemplo el número de sus capas, neuronas por capas, etc.
   - Representación de reglas de desarrollo: En este caso se codificarían las reglas de desarrollo que se usan para construir arquitecturas.
   - Representaciones coevolutivas: En [2] se usa una representación en la que cada individuo de una población representa una neurona de la red, en lugar de una red en sí. Para asignar el valor de adaptación, se utilizar técnicas de compartición de éste.

Con respecto al proceso de evolución, se instanciaría el Algoritmo 2-2, o algoritmo evolutivo general, definiendo todos los elementos que lo componen. Así, por ejemplo, un individuo será una red o un componente de ésta (neurona o conjunto de ellas) si se elige una estrategia coevolutiva. A continuación se formará la red correspondiente y se entrenará (si los pesos están sin determinar), para después medir el error producido por la red. Lo normal será usar este error, para calcular el valor de adaptación de un individuo. Este esquema de funcionamiento se define en la figura 2-8.

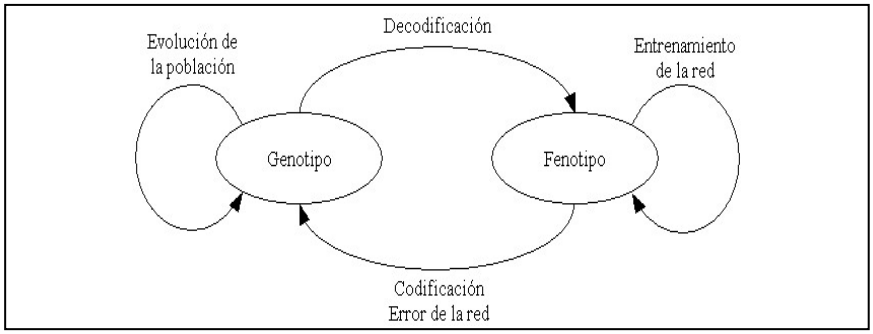


Figura 2-8: Ejemplo de un mecanismo evolutivo para el diseño de una red neuronal

En principio, la evolución de las arquitecturas de red es un proceso adaptativo donde no tiene porque existir un conocimiento a priori, pero su existencia en cualquiera de la etapas del algoritmo, facilita la eficiencia de la evolución. Con respecto a la definición de la función evaluación no existen tampoco limitaciones, en función de aspectos de continuidad, diferenciabilidad, etc. En esta función se suelen incluir sin mucha dificultad, parámetros relativos a eficiencia, complejidad, costes, etc. Todo esto va a mejorar los resultados obtenidos.

Uno de los problemas detectado en el diseño evolutivo de arquitecturas es el problema de la permutación, que ocurre cuando dos redes son funcionalmente equivalentes, pero tienen sus nodos dispuestos de diferente manera, y por lo tanto tienen dos genotipos diferentes. Esto implica se mantendrá en la población dos redes prácticamente iguales y que los descendientes de estas redes normalmente tendrán un valor de adaptación bajo. Esta es una de las razones por las que en muchos algoritmos evolutivos de este tipo, se omite un operador de cruce

2.5.5 Evolución de las reglas de aprendizaje

Un algoritmo de entrenamiento puede presentar diferentes resultados dependiendo de la arquitectura a la que se aplique. De hecho, el diseño de un algoritmo de entrenamiento, o de las reglas de aprendizaje, usadas para ajustar los pesos de una red, dependen de la arquitectura elegida para esta. El problema, es que este diseño se suele tornar complicado cuando hay poco conocimiento a priori, sobre la arquitectura de red y la aplicación determinada.

Surge pues la necesidad de desarrollar un método automático y sistemático para adaptar las reglas de aprendizaje, a la arquitectura de la red y aplicación elegidas. En la actualidad el diseño de estas reglas de aprendizaje es realizado por expertos (humanos) en la materia realizando una serie de suposiciones que suplen la falta de conocimiento antes mencionada, pero que muchas veces son erróneas. Lo ideal sería que el sistema adaptara sus reglas de aprendizaje en función de la red con la que se trabaje y a la aplicación asignada. Así pues parece normal pensar en técnicas evolutivas para conseguir esta adaptación.

El trabajo en este campo [137], está en sus primeras etapas, pero pone de manifiesto que las relaciones entre evolución y aprendizaje son bastante complejas. Incluso, yendo más allá, permitiría ayudar a entender como la creatividad aparece en sistemas artificiales, lo que puede servir de antesala para modelar el proceso creativo en sistemas biológicos.

Se pueden distinguir las siguientes aproximaciones en la evolución de reglas de aprendizaje:

1. Evolución de los parámetros de las reglas: En esta primera aproximación se propone un ajuste evolutivo de los parámetros que rigen las reglas y no de las reglas en su totalidad. En [6] se usa una técnica evolutiva para encontrar los mejores parámetros de las reglas de aprendizaje, manteniendo predefinida y fija la arquitectura de la red. Por otro lado en [63] se hace evolucionar tantos los parámetros de las reglas como la arquitectura de la red.
2. Evolución de las reglas de aprendizaje: En este caso se produce la adaptación de las reglas en sí. La adaptación de las reglas de aprendizaje hace que estas reglas y la red en general tengan una conducta dinámica. El problema es cómo codificar esta conducta dinámica de la red en genotipos estáticos. El intentar desarrollar un esquema de representación que pueda recoger esto es impracticable, por lo que es necesario imponer una serie de restricciones. Un ejemplo de estas restricciones es que la forma de adaptación sea a partir de información local o que las reglas de aprendizaje sean iguales para todas las conexiones de una red. Una vez prefijadas estas restricciones hay que decidir que parámetros van a intervenir en las reglas, decidir la representación de los genotipos y decidir las características del algoritmo evolutivo. Ejemplos de estos algoritmos se encuentran en [137].

2.6 Sistemas de Lógica Difusa

2.6.1 Introducción

La lógica difusa permite representar de forma matemática conceptos o conjuntos imprecisos, tales como días fríos, meses calurosos, personas altas, salarios bajos, consumos altos, etc. Pero hay que tener en cuenta que la idea en sí de que las cosas no son blancas o negras, sino que existen infinitos matices de grises viene ya desde la época de los primeros grandes filósofos.

Posteriormente, otros grandes pensadores apoyaban esta idea manteniendo que el razonamiento venía dado por las observaciones de las que somos testigos a lo largo de nuestra vida y la detección de algunos principios contradictorios en la lógica clásica.

Nace a mediados de los años 60, cuando Lotfi Zadeh publica su trabajo Conjuntos Difusos [139], por el cual se le considera el padre de este campo científico.

Este paradigma propone un nuevo modelo de inferencia más cercano al lenguaje natural y al pensamiento humano. Para esto, provee de un medio efectivo para capturar la natural inexactitud y aproximación del mundo real.

En el lenguaje humano, es normal, que se manejen sentencias como: "Juan es alto", "La velocidad es baja", "El agua está caliente", etc. Estas sentencias se caracterizan porque los adjetivos empleados para distintos elementos, (alto, baja, caliente…) no tienen una cuantificación exacta y son muchas veces subjetivos al sujeto que los emite.

Es frecuente, también que las personas empleen reglas del tipo "Si la velocidad es alta Entonces presiona moderadamente el freno", "Si la temperatura es baja Entonces eleva levemente la potencia del calentador", etc. Estos ejemplos, captan el funcionamiento de la lógica difusa, y revelan que para su implementación se necesitan elementos como:

1. Etiquetas lingüísticas frente a valores numéricos de una variable (muy caliente frente a 40ºC)
2. Cuantificación de variables lingüísticas (u1 puede tener número finito de etiquetas lingüísticas dentro de un rango: muy bajo, bajo, …, muy alto) mediante funciones de pertenencia difusas
3. Conexiones lógicas para variables lingüísticas (Y, O, ...)
4. Implicaciones (SI A ENTONCES B)
5. Combinaciones de reglas, etc.

A continuación se describen, más formalmente, todos los elementos necesarios para implementar un sistema de que permita realizar razonamiento utilizando lógica difusa, más conocido como Sistema de Lógica Difusa (SLD). Para esto se muestran los fundamentos de la lógica difusa, las bases del razonamiento difuso y conceptos sobre defuzzificación.

2.6.2 Fundamentos de la lógica difusa

2.6.2.1 Conjuntos clásicos y conjuntos difusos

Un conjunto nítido es un conjunto clásico que puede ser definido mediante el listado de todos sus elementos o mediante el establecimiento de una condición que exprese los elementos que pertenecen a ese conjunto, por ejemplo:

$$A = \{ x \mid x > 5 \} \qquad 2\text{-}15$$

Para este conjunto se puede definir una función de pertenencia cero-uno, también llamada función característica, denotada por $\mu A(x)$ de modo que:

$$\mu_A(x) = \begin{cases} 0 & si \quad x \in A \\ 1 & si \quad x \notin A \end{cases} \qquad 2\text{-}16$$

Dado un conjunto nítido, y cualquier elemento se sabe si este pertenece o no a al conjunto.

En un conjunto difuso la transición entre pertenecer o no pertenecer a él, es gradual, y esto es lo que caracteriza a este tipo de conjuntos. Esta transición progresiva va a venir definida por su función de pertenencia.

Un conjunto difuso D se puede definir en un universo de discurso U caracterizado por una función de pertenencia que toma valores en el intervalo [0, 1] y se denota como $\mu_F(u)$: $U \rightarrow [0,1]$.

Como se observa un conjunto difuso se puede ver como una generalización de un conjunto nítido, donde la función de pertenencia sólo tomaba los valores 0 ó 1.

El conjunto difuso F en U puede ser representado por un conjunto de pares ordenados compuestos por el elemento genérico u y su función de pertenencia.

$$F = \{ (u, \mu_F(u)) \mid u \in U \} \qquad 2\text{-}17$$

Cuando U es continuo, F puede ser escrito concisamente como:

$$F = \int_U \mu_F(u) / u \qquad 2\text{-}18$$

si U es discreto, el conjunto difuso F es representado como:

$$F = \sum_{i=1}^{n} \mu_F(u) / u \qquad 2\text{-}19$$

2.6.2.2 Operaciones teóricas de conjuntos

Dados A y B dos conjuntos difusos en U con funciones de pertenencia $\mu_A(x)$ y $\mu_B(x)$ respectivamente, las operaciones de unión, intersección y complemento se definen a partir de sus funciones de pertenencia.

T-conormas	
Suma algebraica	$x \oplus y = x + y$
Suma limitada	$x \oplus y = \min(1, x + y)$
Suma drástica	$x \oplus y = \begin{cases} x & si \quad y = 0 \\ y & si \quad x = 0 \\ 1 & si \quad x, y > 0 \end{cases}$
T-normas	
Producto algebraico	$x \otimes y = x \cdot y$
Producto limitado	$x \otimes y = \max(0, x + y - 1)$
Producto drástico	$x \otimes y = \begin{cases} x & si \quad y = 1 \\ y & si \quad x = 1 \\ 0 & si \quad x, y < 1 \end{cases}$

Tabla 2-1: Distintas T-conormas y T-normas.

2.6.2.3 Unión

La unión de *A* y *B*, se nota como $A \cup B$, y se define con respecto a la función de pertenencia siguiente, para todos los $u \in U$.

$$\mu_{A \cup B}(u) \quad = \quad \max\{ \mu_A(u) , \mu_B(u) \} \qquad 2\text{-}20$$

Intuitivamente se podría ver como el menor conjunto difuso que contenga tanto a *A* cómo a *B*. En general la unión se suele representar como:

$$\mu_{A \cup B}(u) \quad = \quad \mu_A(u) \oplus \mu_B(u) \qquad 2\text{-}21$$

donde a $\oplus$, se le denomina operador *t-conorma* ó *s-norma*. Existen varios tipos de t-conormas, además del máximo en (2-20), estas se muestran en la Tabla 2-1.

2.6.2.4 Intersección

La intersección de A y B, $A \cap B$, se define, para todos los $u \in U$, con respecto a la función de pertenencia:

$$\mu_{A \cap B}(u) = \min\{ \mu_A(u), \mu_B(u) \} \quad \text{2-22}$$

Esta operación se podría entender como el mayor conjunto difuso que es contenido tanto por A, como por B. También aquí se puede generalizar la intersección y expresarla como:

$$\mu_{A \cap B}(u) = \mu_A(u) \otimes \mu_B(u) \quad \text{2-23}$$

La intersección ahora se ha expresado en función de una *t-norma*. Además de la t-norma mínimo de (2-22), existen otras t-normas reflejadas en la Tabla 2-2.

2.6.2.5 Complemento

El complemento del conjunto difuso A, $\bar{A}$, para todos lo $u \in U$, se define con respecto a la función de pertenencia:

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u) \quad \text{2-24}$$

2.6.2.6 Funciones de pertenencia

Tal y como se ha definido una función de pertenencia, $\mu_A(x)$, del elemento x en el conjunto A, devuelve un valor en el intervalo [0, 1] que refleja el grado de pertenencia de x a A.

Existen diferentes tipos de funciones de pertenencia dependiendo de su forma.

2.6.2.7 Triangular

Una función triangular se define con respecto a tres parámetros {a, b, c} que determinan las coordenadas en x de sus esquinas:

$$triángulo(x; a, b, c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right) \quad \text{2-25}$$

2.6.2.8 Trapezoide

Se puede establecer una función trapezoide en función de sus cuatro parámetros {*a*, *b*, *c*, *d*}

$$trapezoide(x;a,b,c,d) \quad = \quad \max\left(\min\left(\frac{x-a}{b-a},\ 1,\ \frac{d-x}{d-c} \right), 0 \right) \qquad 2\text{-}26$$

Estas dos funciones son las más usadas debido a la simplicidad de sus fórmulas y a su eficiencia computacional, sobre todo en aplicaciones en tiempo real. Sin embargo, y por estar compuestas de segmentos no son de las más graduales que se pueden utilizar, para conseguir esta característica se usan las siguientes funciones:

2.6.2.9 Gausiana

Una función de pertenencia gausiana se caracteriza por un centro y un radio {*c*, σ}:

$$gausiana(x;c,\sigma) \quad = \quad e^{\{-((x-c)/\sigma)^2\}} \qquad 2\text{-}27$$

2.6.2.10 Campana generalizada

Como caso más general de función gradual podemos definir una campana con tres parámetros {*a*, *b*, *c*} que definen la amplitud de la campana *2a*, su centro *c*, y la pendiente lateral *–b/2a*:

$$campana(x;a,b,c) \quad = \quad \frac{1}{1+|(x-c)/a)|^{2b}} \qquad 2\text{-}28$$

Producto cartesiano, relaciones difusas y composición de relaciones

Dados los conjuntos difusos A_1, …, A_n en los universos de discurso U_1, …, U_n respectivamente, el producto cartesiano de A_1, …, A_n, es un conjunto difuso en el espacio $U_1 \times \ldots \times U_n$, con la función de pertenencia

$$\mu_{A_1 \times \ldots \times A_n}(u_1,\ldots,u_n) \quad = \quad \min\left\{ \mu_{A_1}(u_1),\ldots,\mu_{A_n}(u_n) \right\} \qquad 2\text{-}29$$

ó

$$\mu_{A_1 \times \ldots \times A_n}(u_1, \ldots, u_n) = \mu_{A_1}(u_1) \cdot \mu_{A_2}(u_2) \ldots, \mu_{A_n}(u_n) \qquad 2\text{-}30$$

Una relación difusa representa un grado de presencia o ausencia de asociación, interacción o interconexión entre dos o más conjuntos difusos. Algunos ejemplos de relaciones difusas son: *x* es mucho más grande que *y*, *y* está muy cerca de *x*, *z* es mucho más verde que *y*, … Es tipo de relaciones juegan un papel muy importante en un SLD.

Sean U y V dos universos de discurso, una relación difusa, $R(U, V)$ es un conjunto difuso en el espacio del producto $U \times V$, y se caracteriza por una función de pertenencia $\mu_R(x,y)$ donde $x \in U$ e $y \in V$, es decir:

$$R(U, V) = \left\{ \left((x, y), \mu_R(x, y) \right) \middle| \, (x, y) \in UxV \right\} \qquad 2\text{-}31$$

La generalización de una relación difusa a un espacio del producto cartesiano *n*-dimensional es directa.

Como las relaciones se pueden considerar como conjuntos difusos en un espacio producto, se pueden definir operaciones algebraicas y teóricas de conjunto para ellas, usando los operadores unión, intersección y complemento, definidos anteriormente. Dadas $R(x,y)$ y $S(x,y)$, dos relaciones difusas en el mismo espacio producto $U \times V$, la intersección y unión de *R* y *S*, que son composiciones de dos relaciones, se definen como:

$$\mu_{R \cap S}(x, y) = \mu_R(x, y) \otimes \mu_S(x, y) \qquad 2\text{-}32$$

$$\mu_{R \cup S}(x, y) = \mu_R(x, y) \oplus \mu_S(x, y) \qquad 2\text{-}33$$

donde ⊗es una t-norma y ⊕ es una t-conorma.

Se considera ahora la composición de relaciones de diferentes espacios productos que comparten un conjunto común, por ejemplo $R(U, V)$ y $S(V, W)$ (*x* es más pequeño que *y* e *y* está ceca de *z*). Asociado a la relación difusa *R* tenemos su función de pertenencia $\mu_R(x, y)$, donde $\mu_R(x, y) \in [0, 1]$ y asociada a la relación *S* tenemos su función de pertenencia $\mu_S(x, y)$, donde $\mu_S(x, y) \in [0, 1]$. Cuando *R* y *S* se definen en universos de discurso discretos, entonces la composición difusa de *R* y *S*, se escribe $R \circ S$.

Esta relación puede se descrita por un diagrama sagital, en que cada línea es etiquetada por el valor de su función de pertenencia, o por una matriz relacional difusa, en la que cada elemento es un número real en el intervalo [0,1]. Una forma de expresar matemáticamente la función de pertenencia de esta relación es mediante la conocida *composición sup-star* de *R* y *S*:

$$\mu_{R\circ S}(x,z) = \sup_{y\in V}[\mu_R(x,y)\otimes\mu_S(y,z)] \qquad 2\text{-}34$$

cuando U, V y W son universos de discurso discretos entonces la operación sup es el máximo. Como t-normas se suelen usar el mínimo y el producto.

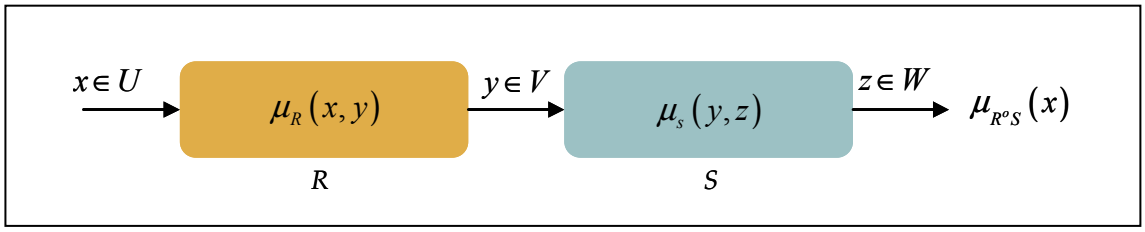


Figura 2-9: Composición sup-star

En la figura 2-9, se muestra un diagrama interpretativo para la composición sup-star, que sugiere una manera simple para componer relaciones difusas más complicadas. Si se considera la relación difusa R tan sólo como un conjunto difuso entonces $\mu_R(x, y)$ se convierte en $\mu_R(x)$, por ejemplo "y es medianamente grande y z es menor que y". Entonces $V = U$ y la figura 2-9 se transforma en la figura 2-10, y representa un conjunto difuso puede activar una relación difusa. Esta es una de las bases del funcionamiento del SLD.

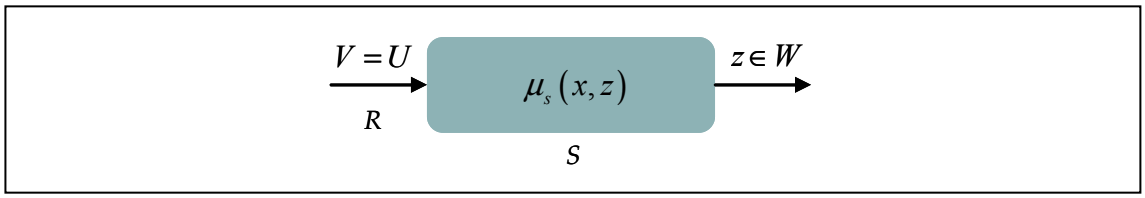


Figura 2-10: Composición sup-star, interpretando la primera relación como un conjunto difuso

Teniendo en cuenta que $V = U$:

$$\sup_{y\in V}[\mu_R(x,y)\otimes\mu_S(y,z)] = \sup_{x\in U}[\mu_R(x)\otimes\mu_S(x,z)] \qquad 2\text{-}35$$

que es sólo función de la variable de salida z, por lo que podemos simplificar la notación $\mu_{R\circ S}(x, z)$ a $\mu_{R\circ S}(x)$, así que cuando R es sólo un conjunto difuso:

$$\mu_{R\circ S}(z) = \sup_{x\in U}[\mu_R(x) \otimes \mu_S(x,z)] \qquad 2\text{-}36$$

2.6.2.11 Variables lingüísticas

Una variable lingüística se caracteriza por la quíntupla (x, $T(x)$, U, G, M), donde x es el nombre de la variable; $T(x)$ es el conjunto de términos o etiquetas de x; G es la regla sintáctica para generar los valores de x; y M es una regla semántica para asociar cada valor con su significado.

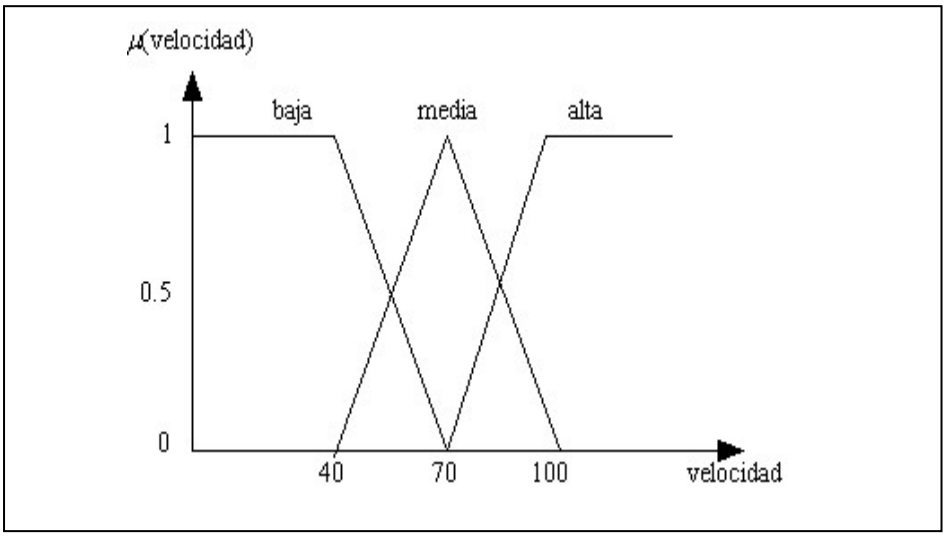


Figura 2-11: Representación de las etiquetas lingüisticas de la variable velocidad.

Por ejemplo, si velocidad es interpretada como una variable lingüística, entonces el conjunto de etiquetas T(velocidad) podría ser:

$$T(\text{velocidad}) = \{\text{baja, media, alta}\} \qquad 2\text{-}37$$

donde cada etiqueta en T(velocidad) se caracteriza por un conjunto difuso en un universo de discurso U = [0, 150]. Se puede interpretar *baja* como "una velocidad por debajo 40 km / h", *media* como "una velocidad en torno a 70 km / h y alta como "una velocidad superior a 100 km / h".

Estas etiquetas se pueden caracterizar como conjuntos difusos cuyas funciones de pertenencia se representan en la figura 2-11.

2.6.3 Lógica difusa y razonamiento difuso

Como se ha descrito hasta ahora, la lógica difusa toma prestados muchos conceptos de la lógica tradicional o clásica. Uno de estos conceptos es el que se conoce como reglas difusas, que se notan de la siguiente manera:

donde A y B son etiquetas lingüísticas definidas en universos de discursos X e Y respectivamente. A la parte "x es A" se le llama antecedente o premisa, mientras que a "y es B" se le llama consecuente o conclusión. Las reglas difusas son ampliamente utilizadas en las expresiones lingüísticas cotidianas: "SI la carretera es sinuosa ENTONCES la conducción es peligrosa"; "SI la velocidad es media ENTONCES presionar ligeramente el freno"...

El siguiente paso es formalizar una regla difusa "SI x es A ENTONCES y es B", la cual se representa abreviadamente cómo: $A \rightarrow B$. En esencia la expresión describe una relación entre dos variables x e y; esto sugiere que una regla difusa, se puede definir en términos de una relación difusa R en el espacio producto $X \times Y$. Además una regla difusa se va a caracterizar por una función de pertenencia $\mu_{A\rightarrow B}(x, y)$ donde $\mu_{A\rightarrow B}(x, y) \in [0, 1]$. $\mu_{A\rightarrow B}(x, y)$ mide el grado de verdad de la relación de implicación entre x e y. Ejemplos de interpretaciones de esta función de pertenencia son:

$$\mu_{A\rightarrow B}(x,y) = 1 - \min\left[\mu_A(x), 1-\mu_B(y)\right] \qquad 2\text{-}39$$

$$\mu_{A\rightarrow B}(x,y) = \max\left[1-\mu_A(x), \mu_B(y)\right] \qquad 2\text{-}40$$

$$\mu_{A\rightarrow B}(x,y) = 1 - \mu_A(x)\left(1-\mu_B(y)\right) \qquad 2\text{-}41$$

El *razonamiento aproximado* (o *razonamiento difuso*) es un procedimiento de inferencia usado para obtener conclusiones a partir de un conjunto de reglas difusas y una o más condiciones. Para esto se usa principalmente una generalización de la regla de inferencia de la lógica clásica *Modus Ponens*, a la que se le denomina *Modus Ponens generalizado* y cuyo funcionamiento se describe a continuación.

premisa 1:	x es A^*	
premisa 2:	SI x es A ENTONCES y es B	2-42
consecuencia:	y es B^*	

Esta regla de inferencia está basada en la regla de composición de inferencia para razonamiento aproximado aportada por [139]. La diferencia con la regla de inferencia

clásica estriba en la inclusión de los conjuntos difusos A^* y B^*. Donde el conjunto difuso A^* no es necesariamente igual al conjunto difuso antecedente A, y B^* no es necesariamente igual al conjunto difuso consecuente B. Esto implica que en lógica difusa, una regla se "dispara" en función de la similaridad entre la primera premisa y el antecedente de la regla. Mientras que el resultado del disparo de la regla es un consecuente que tendrá cierto grado de similaridad con el consecuente de la regla.

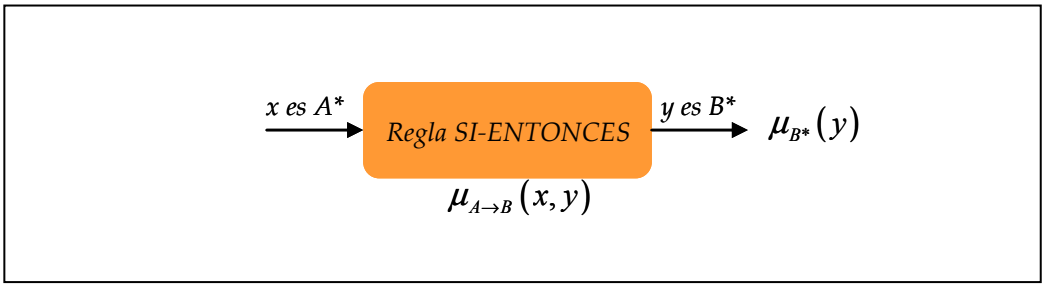


Figura 2-12: Interpretación de Modus Ponens generalizado

La figura 2-12 representa una interpretación para el Modus Ponens generalizado que como se puede observar coincide con la figura 2-10. De esto se puede concluir que el Modus Ponens generalizado es una composición difusa donde interviene una relación difusa.

$$B^* = A^* \circ R = A^* \circ (A \to B) \qquad 2\text{-}43$$

De la misma manera y para obtener $\mu_{B^*}(y)$ usaremos la composición sup-star, teniendo en cuenta las figuras y haciendo las transformaciones simbólicas apropiadas en (1-61):

$$\mu_{B^*}(y) = \sup_{x\in A}\left[\mu_{A^*}(x) \otimes \mu_{A\to B}(x,y)\right] \qquad 2\text{-}44$$

Hay diversas formas de implementar esa implicación. Entre las más comunes destaca la de[87]:

$$\mu_{A\to B}(x,y) = \min\left[\mu_A(x), \mu_B(y)\right] \qquad 2\text{-}45$$

o la de [58]

$$\mu_{A\to B}(x,y) = \mu_A(x)\mu_B(y) \qquad 2\text{-}46$$

lo que se puede expresar como una t-norma:

$$\mu_{A\to B}(x,y) \quad = \quad \mu_A(x) \otimes \mu_B(y) \qquad 2\text{-}47$$

Para trabajar de una forma general más general, se expresa (2-44) de la siguiente forma:

$$\mu_{B^*}(y) \quad = \quad \underset{x\in A}{\oplus}\left[\mu_{A^*}(x) \otimes \mu_{A\to B}(x,y)\right] \qquad 2\text{-}48$$

Vamos a considerar ahora diversos casos prácticos que se nos pueden presentar:

Un solo antecedente y un solo consecuente: En este caso se puede transformar (2-48) en la siguiente ecuación:

$$\mu_{B^*}(y) \quad = \quad \underset{x\in A}{\oplus}\left[\mu_{A^*}(x) \otimes \mu_A(x)\right] \otimes \mu_B(y) \quad = \quad w \otimes \mu_B(y) \qquad 2\text{-}49$$

Lo que implica que se buscará primero el grado de solapamiento máximo w entre $\mu_{A^*}(x) \otimes \mu_A(x)$ (el área sombreada en la parte antecedente de la Figura); entonces la función de pertenencia de B^* es igual que la función de pertenencia de B recortada por w, mostrada en la parte consecuente de la figura 2-13.

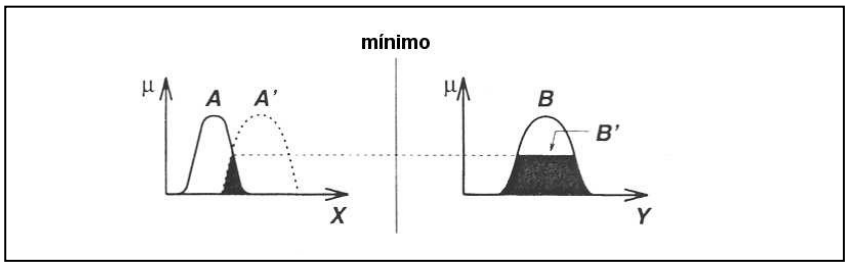


Figura 2-13: Razonamiento con un antecedente y un consecuente

Dos antecedentes y un consecuente: en este caso la regla se escribiría cómo "SI x es A Y y es B ENTONCES z es C". El problema a resolver mediante razonamiento aproximado se expresa:

premisa 1:	x es A^* Y y es B^*	
premisa 2:	SI x es A Y y es B ENTONCES z es C	2-50
consecuencia:	z es C^*	

La regla difusa de la premisa 2 se puede expresar como $A \times B \to C$. Intuitivamente esta relación difusa se puede expresar con la siguiente función de pertenencia:

$$\mu_{A\times B\to C}(x,y,z) \quad = \quad \mu_A(x)\otimes\mu_B(y)\otimes\mu_C(z) \qquad \text{2-51}$$

Y C* se puede expresar como:

$$C^* = (A^*\times B^*)\circ(A\times B\to C) \qquad \text{2-52}$$

entonces:

$$\begin{aligned}\mu_{C^*}(y) \quad &= \quad \underset{x,y}{\oplus}\left[\mu_{A^*}(x)\otimes\mu_{B^*}(y)\right]\otimes\left[\mu_A(x)\otimes\mu_B(y)\otimes\mu_C(z)\right]\\ &= \underset{x,y}{\oplus}\left[\mu_{A^*}(x)\otimes\mu_{B^*}(y)\otimes\mu_A(x)\otimes\mu_B(y)\right]\otimes\mu_C(z)\\ &= \left\{\underset{x\in A}{\oplus}\left[\mu_{A^*}(x)\otimes\mu_A(x)\right]\right\}\otimes\left\{\underset{y\in B}{\oplus}\left[\mu_{B^*}(y)\otimes\mu_B(y)\right]\right\}\otimes\mu_C(z)\\ &= w_1\otimes w_2\otimes\mu_C(z)\end{aligned} \qquad \text{2-53}$$

donde w_1 es el grado de máximo solapamiento entre A^* y A; w_2 es el grado de solapamiento entre B^* y B; y $w_1 \otimes w_2$ es el grado de disparo de la regla para el que se escoge una t-norma mínimo.

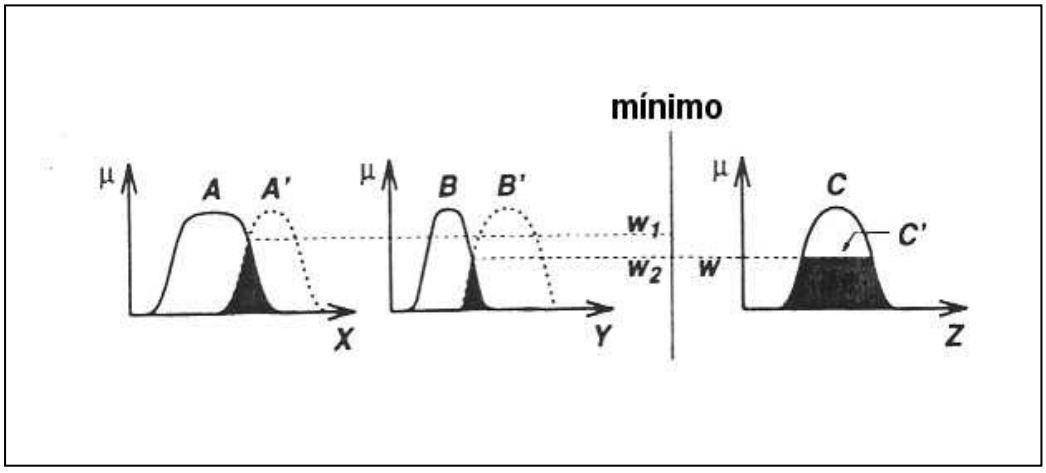


Figura 2-14: Razonamiento con dos antecedentes y un consecuente

En la figura 2-14, se muestra una interpretación gráfica de este procedimiento, donde la función de pertenencia de C^* es igual la función de pertenencia de C recortada por el grado de disparo de la regla. La generalización a más de dos antecedentes es directa.

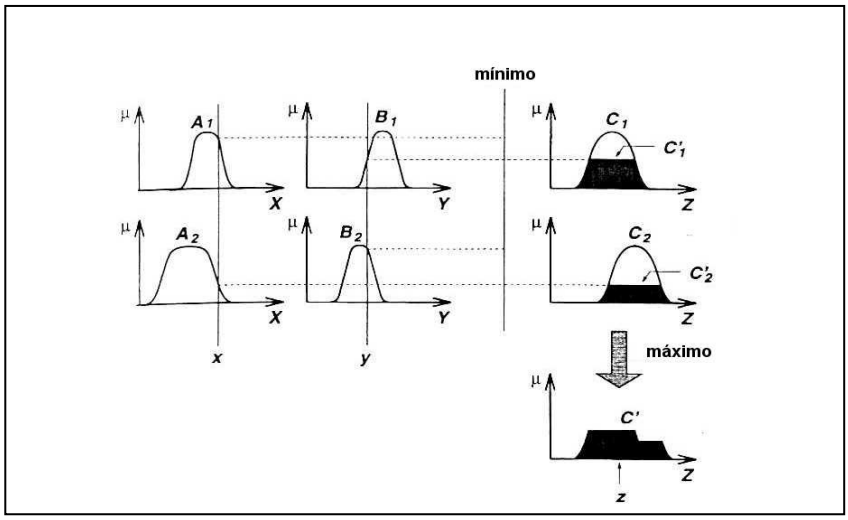


Figura 2-15: Razonamiento difuso (máximo-minimo) para múltiples reglas con múltiples antecedentes

Múltiples reglas con múltiples antecedentes: la interpretación de múltiples reglas se toma normalmente como la unión de relaciones difusas correspondientes a las reglas difusas. Por ejemplo dados los siguientes hechos y reglas:

premisa 1: x es A^* Y y es B^*

premisa 2: SI x es A_1 Y y es B_1 ENTONCES z es C_1

premisa 3: SI x es A_2 Y y es B_2 ENTONCES z es C_2

consecuencia: z es C^* 2-54

En este caso, y para obtener C^* se realizan las siguientes transformaciones aplicando la ley distributiva sobre el operador $\cup$.

$$\begin{aligned} C^* &= (A^* \times B^*) \circ \left[(A_1 \times B_1 \to C_1) \cup (A_2 \times B_2 \to C_2)\right] \\ &= \left[(A^* \times B^*) \circ (A_1 \times B_1 \to C_1)\right] \cup \left[(A^* \times B^*) \circ (A_2 \times B_2 \to C_2)\right] \\ &= C_1^* \cup C_2^* \end{aligned} \qquad 2\text{-}55$$

donde C_1^* y C_2^* son los conjuntos difusos para la regla 1 y 2, respectivamente. En la figura , se muestra gráficamente el razonamiento difuso para múltiples reglas con

múltiples antecedentes, utilizando como t-conorma el máximo, y como t-norma el mínimo.

Al tipo de razonamiento explicado se le denomina de [87]. En este modelo de razonamiento, se puede utilizar la operación producto en la implicación en vez de mínimo. Este modelo se muestra en la figura 2-16.

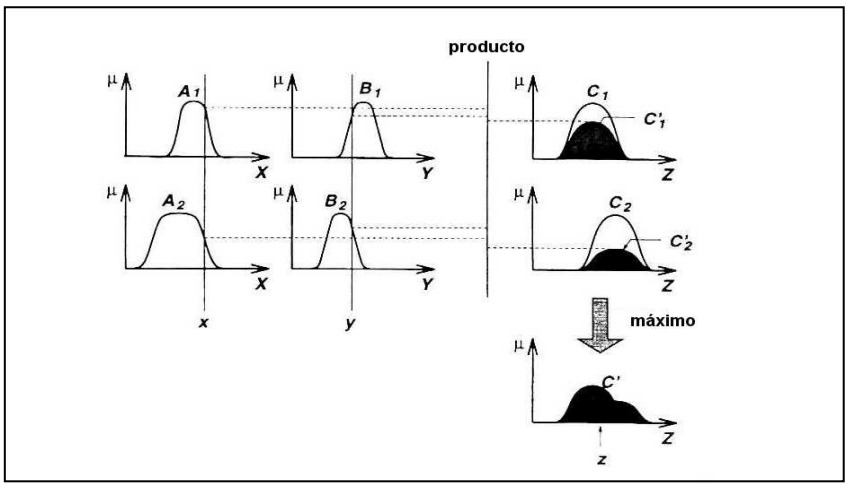


Figura 2-16: Razonamiento difuso (máximo-producto) para múltiples reglas con múltiples antecedentes

Existen otros modelos de razonamiento como el de [130], o el de [132] donde la salida se pone en función de la entrada.

2.6.4 Defuzzificación

Básicamente la defuzzifcación es una transformación de un espacio de decisión difuso de un universo de discurso en un espacio de decisión no difuso.

La defuzzificación sólo se utiliza, cuando se necesita una salida nítida del sistema.

Sin embargo, no existe un procedimiento sistemático para elegir una estrategia de defuzzificación. Las más utilizadas son:

1. *Criterio del máximo*. Este método produce el punto en que la salida del sistema alcanza su mayor valor, la primera vez (ó la última).

2. *Media del máximo*. Con esta técnica se genera un punto que es la media de todos los puntos cuya función de pertenencia alcanza el máximo.

   En el caso de que tengamos un universo de discurso discreto, esto se representa:

$$z_0 = \sum_{i=1}^{l} w_i / l \qquad 2\text{-}56$$

donde z_0 es la salida l es el número de puntos que alcanzan el valor máximo y w_i es cada uno de esos puntos.

3. *Centro del área*. Es una de las más usadas y produce punto centro de gravedad de la salida del sistema.

   En el caso de un universo de discurso discreto esto se representa como:

$$z_0 = \frac{\sum_{i=1}^{n} \mu(w_i) w_i}{\sum_{i=1}^{n} \mu(w_i)} \qquad 2\text{-}57$$

donde n es el número total de puntos del universo de discurso.

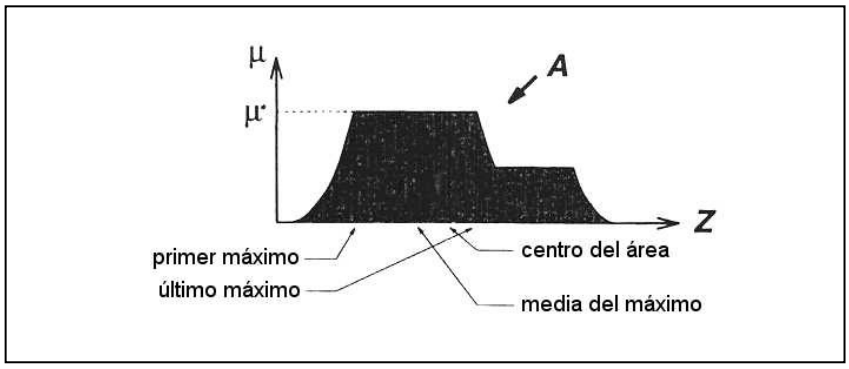


Figura 2-17: Estrategias de defuzzificación

En la figura 2-17 se muestran distintas estrategias de defuzzificación.

2.6.5 Sistemas de lógica difusa

Un Sistema de Lógica Difusa (SLD) figura 2-18, va a obtener unas determinadas salidas a partir de unas entradas dadas. Esta transformación entradas-salidas puede ser expresada como $y = f(x)$.

Un SLD consta de cuatro elementos: un fuzzificador, una base de conocimieto, un motor de inferencia y un defuzzificador.

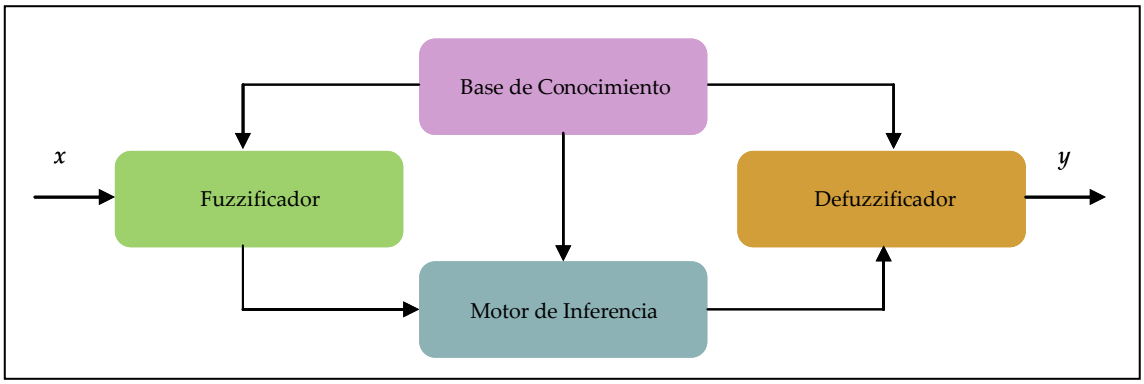


Figura 2-18: Sistema de lógica difusa

1. El fuzzificador tiene como entrada valores nítidos, los cuales evalúa, para realizar su transformación a valores del universo de discurso correspondiente. Esto implica implementar la función de difusión en sí, que consiste en convertir los valores de entrada en valores lingüísticos que pueden ser vistos como etiquetas de conjuntos difusos, y que más tarde se van utilizar para activar reglas.

2. La base de conocimiento representa el conocimiento del dominio de la aplicación y está compuesta por la base de datos y la base de reglas difusas.

   - La base de datos contiene las definiciones necesarias usadas para definir tanto las reglas difusas como la manipulación de datos difusos en el sistema.
   - La base de reglas difusas va a caracterizar el conjunto de objetivos comprendidos por las decisiones a tomar para solventar un problema dado. El conjunto de reglas suele ser suministrado por expertos en el campo en el que esté trabajando el SLD. Estas reglas suelen tener la sintaxis de las sentencias SI – ENTONCES (SI u_1 es alta Y u_2 es muy baja ENTONCES establecer v a un valor bajo).

3. El motor de inferencia de un SLD relaciona conjuntos difusos en conjuntos difusos, manejando la manera en que las reglas son combinadas. De igual manera que los humanos usan diferentes tipos de procedimientos de inferencia, existen diferentes procedimientos inferenciales de lógica difusa.

4. En muchas aplicaciones se necesita que la salida del SLD sea un valor nítido. Esta función la realiza el defuzzificador, que convierte conjuntos difusos de salida en estos valores nítidos.

A la hora de diseñar un SLD hay que llevar a cabo una serie de tareas o tomar una serie de decisiones que configuran las características finales de este. Los parámetros que van a definir en un SLD son:

1. Elegir la estrategia de fuzzificación y la interpretación del operador fuzzificador. Esta elección va a depender del tipo de datos a fuzzificar, del rango en el que están definidos, de si existe ruido, etc [20].

2. En cuanto a la base de datos hay que decidir conceptos que van a caracterizar las reglas de control difusas y la manipulación de datos difusos en un SLD. Estos conceptos son definidos subjetivamente y se basan en la experiencia. Los más importantes son:

    - Discretización / normalización del universo de discurso. En general la representación depende de la naturaleza del universo de discurso. Un universo de discurso puede ser continuo o discreto. Si el universo es continuo puede ser discretizado o puede ser normalizado teniendo en cuenta necesidades sobre todo de representación de la información.
    - Partición difusa de los espacios de entrada y salida. Una variable lingüística en un antecedente de una regla constituye un espacio de entrada, mientras que una variable lingüística en un consecuente de una regla constituye un espacio de salida. En general, una variable lingüística se asocia con un conjunto de etiquetas lingüísticas, definiéndose cada una de las etiquetas en el mismo universo de discurso. La partición difusa determina cuantas etiquetas lingüísticas existen en el conjunto de etiquetas lingüísticas. Este número va a determinar la granuralidad de las reglas y de las decisiones que se toman con el SLD.
    - Completitud. Un SLD cumple esta propiedad si para cualquier estado del sistema se puede inferir una acción a tomar. Para que se cumpla esta propiedad tenemos que tener en cuenta tanto el diseño de la base de datos, por ejemplo que cubra bien los espacios de entrada y salida, como el conjunto de reglas difusas, de modo que no existan condiciones que no se contemplen.
    - Funciones de pertenencia de conjuntos difusos. Dependiendo de si el universo de discurso es discreto o continuo, la pertenencia a este se puede definir de una manera numérica o funcional respectivamente. Si la función de pertenencia es numérica se indica para cada elemento del conjunto su grado de pertenencia. Si la función de pertenencia es funcional esta se

especifica mediante una función que dará el grado de pertenencia para cada elemento del conjunto.

3. El conjunto de reglas difusas de un SLD va a representar el conocimiento experto en un campo. Entre los elementos a decidir destaca la fuente y derivación de las reglas difusas. Existen diversas fuentes para derivar reglas difusas. Entre estas destacan la derivación de reglas a partir de conocimiento experto [87], o el auto-aprendizaje mediante diversas técnicas de las reglas [113]. Aunque suele ser normal la hibridación de estas técnicas para la consecución del conjunto final de reglas.

2.7 Aproximación Funcional mediante Redes RBF:

En este apartado se hace referencia a los conceptos generales de las Redes de Funciones de Base Radial, ya que se han elegido como el elemento computacional, que en el nivel más inferior se encarga de realizar el procesamiento de los datos de entrada, para obtener las correspondientes salidas del módulo de Aproximación Funcional del modelo propuesto.

Esta descripción comienza, detallando las bases estructurales y de funcionamiento de estas redes, continuando con un estudio de las propiedades que las caracterizan.

Dado que nuestro objetivo es diseñar y optimizar este tipo de redes, en una segunda parte, se hará un repaso de los métodos que se han utilizado en la bibliografía para este fin.

En esta exposición de métodos, no sólo se incluirán aquellos que están más relacionados con el softcomputing o tengan más características bioinspiradas, sino que también se detallarán métodos más numéricos, para de esta manera, tener una visión más completa de este campo y poder aplicar, un algoritmo de diseño y optimización más adecuado.

2.7.1 Redes RBF

Una Función de Base Radial (RBF), ϕ, se puede expresar como:

$$\phi_i(\bar{x}) = \phi(\| \bar{x} - \bar{c}_i \| / d_i) \qquad 2\text{-}58$$

Ésta se caracteriza porque su salida es simétrica, y se incrementa (o decrementa) de forma monótona, con respecto a un centro, $\bar{c}_i$, donde de forma general $\bar{c}_i \in \Re^n$. $d_i \in \Re$

es un factor de escala para $||\bar{x}_i - \bar{c}_i||$ o radio y como $||\cdot||$ se suele escoger la norma Euclídea en $\Re^n$. Para determinar la forma de una RBF existen distintas posibilidades, las más usuales se encuentran en la Tabla 2-2.

$\phi(x)$	Nombre
X	Función lineal
x^3	Función cúbica
$\text{Exp}(-x^2)$	Función Gausiana
$x^2 \log(x)$	Función Spline
$(x^2 + 1)^{1/2}$	Función Multicuadrática
$(x^2 + 1)^{-1/2}$	Función inversa multicuadrática

Tabla 2-2: Posibles elecciones para ϕ

En [120]se hace un estudio sobre las distintas formas que puede adoptar una RBF a la hora de formar parte de una Red de Funciones de Base Radial. En este trabajo se concluye que una RBF con forma gausiana es una de las mejores elecciones. La figura 2-19 muestra la forma de las funciones base gausiana y multicuadrática, con los parámetros $c = 0$ y $r = 1$.

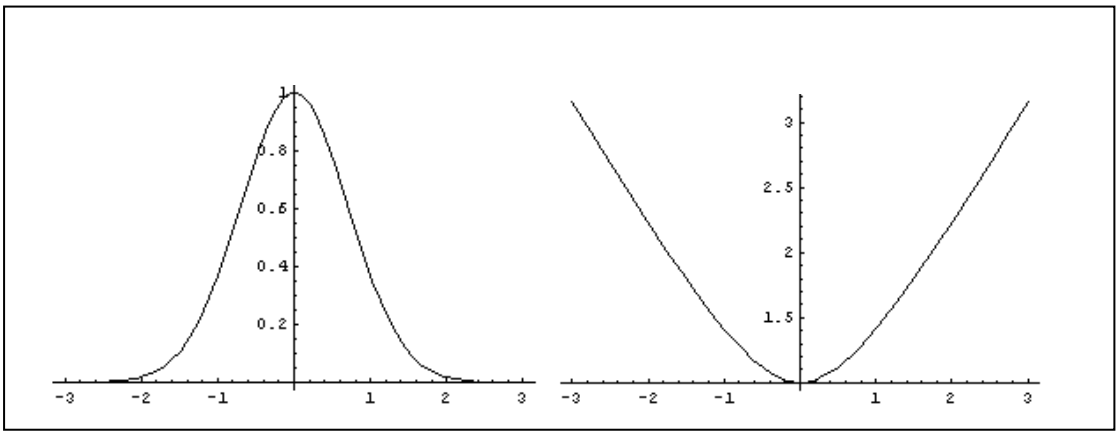


Figura 2-19: RBFs gausiana y mutlticuadrática con c = 0 y r =1.

Una Red de Funciones de Base Radial (RBFN) es una red hacia adelante (feedforward network) que contiene tres capas: las entradas, la capa oculta, el/los nodo/s de salida. Cada neurona de la capa oculta se caracteriza por que su salida viene dada por una RBF. La activación de estas funciones es proporcional a la cercanía, medida por la norma

euclídea, entre el patrón de entrada y centro c_i correspondiente. Si el patrón de entrada está cerca del centro de una RBF, la salida de está se acercará a 1, en caso contrario se acercará a 0.

A partir de [8] y cuando una RBFN se utiliza en interpolación, aproximación de funciones o predicción de series temporales, consta de un solo nodo de salida que implementa una combinación lineal de las RBFs, por las que está compuesta. Por tanto la salida de la red se expresará de la siguiente manera:

$$f(\bar{x}) = \sum_{j=1}^{m} w_j \phi_j(\bar{x}) \qquad 2\text{-}59$$

donde w_j es el peso asociado a la neurona j. Este tipo de RBFN, se muestra en la figura 2-21, y será la que utilizaremos en nuestro trabajo . En cuanto a la elección concreta de la forma para las RBFs, la más utilizada en estos casos es la gausiana tal y como se describirá a continuación.

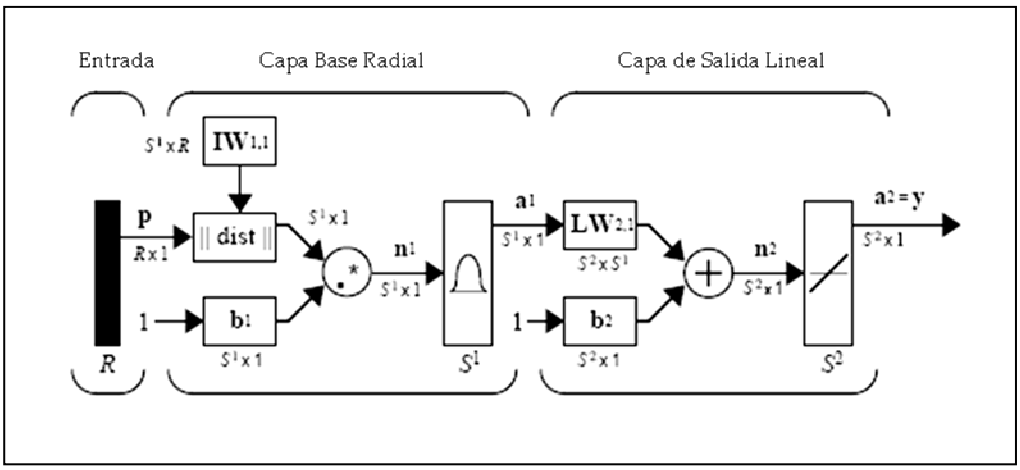


Figura 2-20: Red de Funciones de Base Radial

Haciendo un poco de historia, uno de los primeros trabajos que utiliza la superposición de funciones gausianas se realiza a principios de los sesenta por [90]. Después se demuestran las propiedades de estas a la hora de interpolar y aproximar funciones [110], [8]. Más tarde, se renueva el interés por estas al hacerse las primeras implementaciones de redes neuronales [83], [94], [107]. Parte de este interés se debía a las analogías con los campos receptivos localizados encontrados en muchas estructuras biológicas. Esto motivó el trabajo con funciones de forma de campana (gausianas), que se activaban en la vecindad de un punto.

Las propiedades de aproximador universal de la ecuación 2-59 se demuestran en [79], [102], [103]. En la actualidad las RBFNs, son uno de los modelos de red más usados en problemas de aproximación y clasificación, gracias a sus características. Los primeros problemas a los que se aplican las Redes de Funciones de Base Radial son la interpolación exacta y la aproximación de funciones.

2.7.2 Interpolación exacta

El problema de la interpolación exacta se formula de la siguiente manera: dado un conjunto de vectores de entrada $\bar{x} \in \Re^d$, y su salida $y \in \Re$, encontrar una función continua que:

$$h(\bar{x}_i) = y_i \quad i = 1, \ldots, n \tag{2-60}$$

La solución a este problema utilizando RBFs, consistiría en elegir un conjunto de n funciones base, centradas en los n puntos de datos, y usar la fórmula:

$$h(\bar{x}) = \sum_{i=1}^{n} w_i \phi_i(\| \bar{x} - \bar{x}_i \|) \quad i = 1, \ldots, n \tag{2-61}$$

Para conseguir una solución exacta, será necesario resolver las n ecuaciones lineales del sistema siguiente:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{n1} & \phi_{n2} & \cdots & \phi_{nn} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2-62}$$

donde se determinará el valor de los pesos y

$$\phi_{ij} = \phi(\| \bar{x}_i - \bar{x}_j \|) \quad i, j = 1, \ldots, n \tag{2-63}$$

La ecuación (2-62) se expresa de forma compacta como:

$$\Phi \overline{w} = \overline{y} \qquad 2\text{-}64$$

Según los trabajos [91] [84], existen muchos tipos de formas para la RBFs, como gausianas, splines o inversas multicuadráticas, para las que se puede resolver la ecuación siguiente y de esta manera conseguir una interpolación exacta. El vector de pesos se determinaría:

$$\overline{w} = \Phi^{-1} \overline{y} \qquad 2\text{-}65$$

También estos estudios muestran que muchas características de la función interpolada son relativamente insensibles a la forma de la función de la RBF. Sin embargo una de las formas más utilizadas son las gausianas, ya que su respuesta localizada ayuda en el proceso de aprendizaje, además de demostrar otras características beneficiosas [83], [120].

2.7.3 Aproximación funcional

En una gran mayoría de situaciones , no es recomendable realizar una interpolación exacta. Una de estas situaciones, sería por ejemplo cuando existe ruido, entonces la función de interpolación que pasara por los datos, podría conllevar sobre-entrenamiento y por tanto una pobre generalización. Otra razón, es que para garantizar una interpolación exacta, el número de funciones base requerido es igual al número de patrones en el conjunto de entrenamiento.

Por lo tanto, si el número de patrones es muy alto, el establecer la función de interpolación puede ser muy costoso.

En los trabajos [8] y [94] se obtiene un modelo, basado en RBFs, para la aproximación y generalización funcional modificando el procedimiento de interpolación exacta.

Este modelo obtiene una aproximación suave a los datos, a partir de un número reducido de funciones base.

El número de funciones base depende de la complejidad de la función a aproximar antes que del tamaño del conjunto de datos.

Las principales modificaciones inciden en los siguientes aspectos:

1. Número de funciones base: El número de funciones base m es normalmente menor que n.

2. Centros de las funciones base: Los centros de las funciones se determinan como parte de un proceso de entrenamiento, en vez de en función de un conjunto de patrones de entrada.
3. Radios de las funciones base: El proceso de entrenamiento también adaptará el valor del radio de cada función base, antes que asignar el mismo radio a todas las funciones base.
4. Introducción de un parámetro de adaptación (bias): De forma opcional se puede introducir un parámetro de adaptación para compensar la diferencia, si existe, entre la media de los datos de activación de las funciones base y los correspondientes valores a aproximar.

Aplicando estas modificaciones al proceso de interpolación exacta, y si se empieza introduciendo el parámetro de adaptación, la salida de la red quedaría:

$$f(x) = w_0 + \sum_{j=1}^{m} w_j \phi_j(x) \qquad 2\text{-}66$$

El parámetro de adaptación w_0, se puede incorporar en la sumatoria de la ecuación anterior, introduciendo una función base extra ϕ_0, y estableciendo su salida al valor constante 1, con lo que quedaría:

$$f(x) = \sum_{j=0}^{m} w_j \phi_j(x) \qquad 2\text{-}67$$

En [79] se prueba que la superposición lineal de funciones base gausianas, donde los radios se traten como parámetros ajustables, es un *aproximador universal*. En [102] y [103] se muestra que sólo con leves restricciones en la forma de las funciones base, estas propiedades en la aproximación se mantienen. Sin embargo en los trabajos, no se ofrecen datos ni sobre el número de funciones base a introducir, ni sobre procedimientos para construir la red. De todas formas, si se ofrecen fundamentos teóricos en los que apoyar las aplicaciones prácticas.

Así por ejemplo, según [107], una RBFN posee la propiedad de mejor aproximador, lo que quiere decir que existe un único conjunto de parámetros que consigue la mejor aproximación de una función determinada. Esta propiedad, por ejemplo, no la muestran los perceptrones multicapa.

2.8 Diseño de Redes de Funciones de Base Radial

2.8.1 Conceptos y técnicas básicas

Según la ecuación (2-59), para diseñar una RBFN, se tendrán que determinar varios parámetros como el tipo de funciones base ϕ, el número de estas, m, su radio, d_i, la localización de su vector de centros, $\bar{c}_i$, y los pesos, w_i, asociados a las neuronas.

En nuestro caso, como tipo de funciones base se usarán las gausianas, viniendo también predefinido en muchas ocasiones, el número de estas funciones, que va a intervenir en la red. Esto implica que, normalmente y cuando se diseña una RBFN, se aporten estrategias para el establecimiento de los conjuntos de radios, centros y pesos de una red.

Como punto de partida, y para diseñar una RBFN, se pueden utilizar técnicas de gradiente descendiente.

De la misma manera que se usan en el diseño de otras redes, éstas realizan la actualización de parámetros de forma iterativa, cada vez que se muestra un patrón del conjunto de entrenamiento. Para esto se define una función de costo a minimizar:

$$e = \sum_{i=0}^{n} (y_i - f(x_i))^2 \qquad 2\text{-}68$$

Así, y si se usan funciones gausianas, se pueden utilizar las siguientes ecuaciones [47], para actualizar los conjuntos de parámetros:

$$\Delta w_j = \alpha_1 (y_i - f(\bar{x}_i))\phi(\bar{x}_i) \qquad 2\text{-}69$$

$$\Delta \bar{c}_j = \frac{\alpha_2 (y_i - f(\bar{x}_i))\phi(\bar{x}_i) \| \bar{x}_i - \bar{c}_j \| w_j}{d_j^2} \qquad 2\text{-}70$$

$$\Delta d_j = \frac{\alpha_3 (y_i - f(\bar{x}_i))\phi(\bar{x}_i) \| \bar{x}_i - \bar{c}_j \|^2 w_j}{d_j^3} \qquad 2\text{-}71$$

donde α_1, α_2 y α_3 definen la variación a realizar o la velocidad del aprendizaje. Estas ecuaciones representan una particularización del algoritmo LMS para las RBFNs.

2.8.2 Cálculo de parámetros de una RBFN mediante métodos numéricos

En este apartado se van a presentar algunos de los métodos más usados para la determinación de los pesos de una RBFN mediante métodos matemáticos. Hay que tener en cuenta que una RBFN es una red simple en cuanto a su estructura, ya que posee una sola capa oculta, y que su salida se puede expresar de forma lineal. Esto implica, que el problema de la determinación de un conjunto de parámetros de una RBFN pueda ser enunciado muchas veces, como la resolución de un sistema de ecuaciones.

El siguiente apartado describe una serie de métodos utilizados para la determinación de los pesos de una red, en la que el resto de los parámetros ya están fijados. Por último se referenciarán los métodos de regularización.

2.8.2.1 Cálculo de los pesos de una RBFN mediante métodos numéricos

Estos métodos se usarían cuando estuvieran ya establecidos, mediante algún otro algoritmo de diseño, los parámetros que definen a las distintas RBFs. En este caso se parte de que el problema consiste encontrar los pesos en la expresión 2-59. Nuestro problema se puede expresar como:

$$\overline{y} = \Phi \overline{w} \qquad 2\text{-}72$$

si se tienen en cuenta todas las muestras del conjunto de patrones.

Esto implica que los métodos que se utilizarán a continuación para determinar el vector $\overline{w}$, son métodos usados en la literatura para la resolución de sistemas de ecuaciones lineales. Concretamente los métodos que más se han utilizado para esta tarea en el campo del diseño de las RBFN son la descomposición de Cholesky, la descomposición en valores singulares (SVD) y el método de los mínimos cuadrados ortogonales (OLS).

2.8.2.2 Descomposición de Cholesky

La descomposición de Cholesky [53] [112] es el más rápido de los tres métodos para resolver sistemas de ecuaciones lineales pero solo se puede aplicar a sistemas descritos por una matriz cuadrada, simétrica y definida positiva. El sistema de ecuaciones a resolver (2-72) habría que transformarlo antes ya que la matriz Φ no cumple ninguna de las restricciones anteriores.

Cuando se realizan las transformaciones, la matriz de activación debe verificar otra serie de condiciones, cuyo cumplimiento depende de las características de las RBFs que haya definidas. Así las RBFs deberían estar colocadas de forma que se cubra todo el espacio de entrada y ser linealmente independientes. Esto implica que redes con funciones base "mal colocadas", bien porque alguna no se active con ningún punto de entrenamiento o porque exista demasiado solapamiento entre las mismas, pueden producir matrices de activación singulares que el método de Cholesky sería incapaz de resolver, o bien casi singulares, para las que el método anterior proporcionaría una solución indeseable.

2.8.2.3 Descomposición en valores singulares

La descomposición en valores singulares (Singular Value Descomposition, SVD) [53] es un primer método para solucionar el problema de que existan RBFs mal colocacadas, algo que por otro lado es normal que ocurra durante el diseño de una RBFN. Como se ha mencionado en estos casos se producen matrices de activación singualares o casi singulares. Ocurre que en la matriz de activación cuando dos funciones son casi idénticas, se producen dos columnas prácticamente iguales, mientras que si una función base no se activa para casi ningún punto se producirá una columna casi nula en Φ.

La descomposición en valores singulares facilita una solución para cualquier sistema de ecuaciones, con la que se obtiene una reducción del error. Además el método muestra que, si se elimina de la red alguna función base cuyo valor singular asociado tuviera una magnitud pequeña el error de aproximación no se vería prácticamente afectado, y al ser la red más pequeña se gana simplicidad y se pueden conseguir mejoras en la aproximación. Este suele ser otro de los usos del método SVD, es decir la identificación en una red, de funciones base que aportan poco al funcionamiento general de ésta. El inconveniente que plantea es que la forma en que selecciona las funciones base más importantes de la red no tiene en cuenta la salida esperada del sistema para cada vector de entrada, de forma que si en una red existieran dos funciones base muy solapadas, probablemente sacrificaría a una de las dos sin importar la influencia que dicha modificación tenga en el error de aproximación de la red.

2.8.2.4 El método de los mínimos cuadrados ortogonales (OLS)

Otra posibilidad para resolver el sistema de ecuaciones planteando es el algoritmo OLS. Este método, al igual que el anterior, es aplicable aunque las funciones base estén “mal colocadas”, es decir que pueda existir alguna RBF que no se active con ningún punto de entrenamiento o que exista demasiado solapamiento entre algunas de éstas. Además no tiene tampoco el inconveniente del método anterior. Este inconveniente consistía en que

al intentar detectar funciones base poco útiles para la red y encontrarse con funciones base muy solapadas, se eliminaría seguramente alguna de éstas, aunque ésta tuviera importancia en el error de aproximación obtenido por la red.

El método OLS es un algoritmo iterativo que selecciona en cada iteración la columna de la matriz de activación Φ que más contribuya a la disminución del error de aproximación al modelo. Así este va transformado las columnas de la matriz de activación Φ en un conjunto de vectores ortogonales. A estas columnas se les aplica otra serie de operaciones hasta obtener unos coeficientes que se denominan radios de reducción del error. Este radio proporciona una forma simple de escoger un conjunto de regresores importantes de forma directa. Dichos regresores se corresponden con las funciones base más importantes de la red y si se decide reducir el modelo a *r* funciones base ($r < m$), se deben seleccionar las *r* funciones base con un mayor radio de reducción del error.

Posteriormente se han presentado hibridaciones del método OLS con técnicas de regularización [100] produciendo el algoritmo ROLS [17], e hibridaciones de este último con algoritmos genéticos [17].

2.8.2.5 La técnica de la regularización

La regularización [100] es una técnica que favorece unas soluciones sobre otras añadiendo un término de penalización a la función de costo o en nuestro caso a la función de error a minimizar. Dependiendo de la función de costo y del término de penalización añadido, se definen diferentes técnicas para solucionar problemas. Entre estas se podría destacar por su importancia la regresión límite (ridge regression).

Una de las herramientas más importantes de las que se utilizan en la técnica de la regularización es la matriz de proyección P, que se define como:

$$P = I_n - \Phi A^{-1} \Phi^T \qquad 2\text{-}73$$

donde In es la matriz identidad con dimensión n y A = ΦTΦ.

P es una matriz que representa la proyección de vectores de un espacio *n*-dimensional, donde *n* era el número de patrones del conjunto de entrenamiento en un subespacio *m*-dimensional donde *m* es el conjunto de funciones base. En función de esta matriz se van a poder definir muchos conceptos importantes, como por ejemplo los relacionados con la reducción del error, con los efectos de la inclusión o eliminación de funciones base, etc.

2.6.2.5 Regresión límite

La técnica de la regresión límite (ridge regression) [101] utiliza el concepto de regularización consistente en incluir un término de penalización en la función de error a minimizar. Así, la función de costo a minimizar que se define es:

$$C = \sum_{i=1}^{n} (y_i - f(\bar{x}_i)) + \lambda \sum_{j=1}^{m} w_j^2 \qquad 2\text{-}74$$

donde λ es el parámetro de regularización. Con la inclusión de este término está claro que se penalizarán redes con pesos muy altos, también dependiendo del valor de λ. El objetivo final será obtener una red con una salida suave, al tener los pesos de ésta un valor bajo o moderado. Con estas premisas y usando la matriz de proyección se define el vector de pesos óptimo como:

$$\overline{w} = A^{-1} \Phi^T \overline{y} \qquad 2\text{-}75$$

siendo ahora A:

$$A = \Phi^T \Phi + \lambda I_n \qquad 2\text{-}76$$

2.8.3 Algoritmos de Clustering

Los algoritmos de clustering se desarrollaron inicialmente como técnica de clasificación [64] aunque después se han aplicado en los campos de los sistemas difusos y redes neuronales artificiales [7], [78].

Un algoritmo tradicional de clustering va a intentar dividir un conjunto de vectores $X = \{ \bar{x}_i : i = 1, \ldots, n\}$, en un número c, determinado a priori, de subconjuntos o grupos que van a proporcionar una partición de Voronoi $P = \{P_1, \ldots, P_c\}$, donde para cada grupo (cluster) P_j, se definirá un vector representante o prototipo $\overline{p}_j$. Los vectores se agruparán en estos grupos en función de la definición de alguna o varias características.

Dentro del campo de diseño de una RBFN, estos algoritmos se encuadrarían dentro de una fase de preproceso de datos de entrada de la red. Concretamente, el objetivo de esta fase sería intentar revelar la estructura interna de este conjunto de datos de entrada, agrupando o identificando grupos de estos datos que tengan una o varias características comunes. Una vez realizada esta fase, lo normal sería insertar una RBF en el centro

geométrico de cada uno de los grupos identificados. La determinación de las RBFs insertadas termina con el establecimiento de un radio para de cada una ellas, relacionado con el espacio ocupado por el grupo en el que se encuentran ubicadas.

De todas formas es importante resaltar que los algoritmos de clustering no se suelen contemplar como estrategia única en el diseño de una RBFN. Así lo normal es que estos formen parte de una fase inicial a la hora de diseñar una red de este tipo. Tal y como se ha comentado, esta primera fase se suele corresponder con un estudio preliminar de las características del conjunto de datos, que sirve para realizar una primera determinación o inicialización de los centros, e incluso radios de las RBFs. De este modo, el proceso de diseño se complementaría con otras estrategias que refinan mucho más los parámetros finales de las RBFs en particular y de la RBFN en general.

Dentro de los algoritmos de clustering podemos establecer una clasificación en la que se distinguirían los siguientes tipos:

1. Algoritmos de clustering no supervisados: en este tipo de algoritmos la característica común es que el mecanismo de aprendizaje (asignación de grupos) no tiene en cuenta ninguna información aportada por la/s variable/s de salida.
2. Algoritmos de clustering supervisados: se introducen para mejorar a los algoritmos de clustering no supervisados y se caracterizan porque van a tener en cuenta la información de que suministren la/s variable/s de salida. Esta mejora, parece razonable ya que las redes neuronales pretenden modelar las relaciones entre las entradas y salidas de los conjuntos de entrenamiento. La forma utilizar esta información que aportan la/s variable/s depende del algoritmo en sí, como se describirá a continuación.

Seguidamente se describirán algunos de los algoritmos de clustering más importantes. De estos algoritmos el algoritmo de las c medias, el algoritmo de las c medias difuso y el algoritmo ELBG, se clasifican como algoritmos de clustering no supervisados. Por otro lado el algoritmo de clustering difuso condicional, el algoritmo de estimación de grupos alternantes y el algoritmo de clustering para aproximación de funciones se clasifican como algoritmos de clustering supervisados.

2.8.3.1 Algoritmo de las c medias

Una primera aproximación a los algoritmos de clustering, fue el propuesto inicialmente por [32] y conocido como algoritmo de las c medias (ó K means algorithm). Esta técnica, mostrada en el Algoritmo 2-5 realiza una partición del conjunto de vectores en *c* grupos.

En este caso, se establece como característica de referencia para formar los grupos la distancia entre los vectores. Así, se define como función objetivo a minimizar *J*:

$$J = \sum_{j=1}^{c} \sum_{i=1}^{n} \| \bar{x}_i - \bar{p}_j \| \qquad 2\text{-}77$$

donde $\bar{x}_1$, …, $\bar{x}_n$ son los vectores agrupar, y $\bar{p}_1$, … , $\bar{p}_c$ serían los representantes o prototipos de cada partición o grupo P_1, … , P_c, $\| \cdot \|$ sería la distancia euclídea. Además se define una función de pertenencia $\mu_{P_j}(\bar{x}_i)$ del vector de entrada $\bar{x}_i$, al grupo representado por el prototipo $\bar{p}_j$, que está definida en el rango de salida {0, 1}, como:

$$\mu_{P_j}(\bar{x}_i) = \begin{cases} 1 & \| \bar{x}_i - \bar{p}_j \|^2 < \| \bar{x}_i - \bar{p}_l \|^2 \quad \forall l \neq j \\ 0 & \textit{en otro caso} \end{cases} \qquad 2\text{-}78$$

Esta función de pertenencia produce una partición de Voronoi *P* del conjunto de vectores de entrada de la forma:

$$P = \bigcup_{j=1}^{m} P_j \quad con \quad C \cap C = \emptyset \quad \forall j \neq i \qquad 2\text{-}79$$

donde *Pj* se define como:

$$P_j = \left\{ \bar{x}_i : \mu_{P_j}(\bar{x}_i) = 1 \right\} \qquad 2\text{-}80$$

Como se deduce cada uno de los vectores de entrada sólo va a ser asignado a un único grupo.

Así una partición se va a poder describir mediante la matriz de partición *U* de dimensión $c \times n$, cuyos elementos son las funciones de pertenencia $\mu_{P_j}(\bar{x}_i)$.

De este modo se cumple que:

$$U = \left\{ \mu \in \{0,1\} \mid \sum_{i=1}^{c} \mu_{P_i}(\bar{x}_k) = 1 \ \forall k \quad y \quad 0 < \sum_{k=1}^{n} \mu_{P_i}(\bar{x}_k) < n \quad \forall i \right\} \qquad 2\text{-}81$$

Concretamente el algoritmo empezaría con una serie de inicializaciones generales, entre las que destaca la asignación aleatoria de los prototipos $\bar{p}_j$ de las particiones.

Después se entra en el ciclo principal del algoritmo donde inicialmente se determinará a que partición pertenece cada vector $\mu_{P_j}(\bar{x}_i)$ utilizando (2-78).

A continuación se vuelven a recalcular los prototipos de las particiones utilizando:

$$\bar{p}_j = \frac{\sum_{i=1}^{n} \mu_{P_j}(\bar{x}_i)\bar{x}_i}{\sum_{i=1}^{n} \mu_{P_j}(\bar{x}_i)} \qquad 2\text{-}82$$

La condición de parada del algoritmo consiste en chequear la distorsión δ de la partición actual, con la partición de la partición de la iteración anterior δ_{mt}.

De modo que si el cambio es menor que un ε, prefijado, se terminará el algoritmo.

El valor δ se calcula:

$$\delta = \sum_{j=1}^{c} \delta_j \qquad 2\text{-}83$$

donde δ_j, es la distorsión producida en cada cluster y se calcula como:

$$\delta_j = \sum_{\bar{x}_i \in P_j} \| \bar{x}_i - \bar{p}_j \|^2 \qquad 2\text{-}84$$

1. Realizar la asignación δ = ∞
2. Asignar aleatoriamente el conjunto inicial de prototipos
3. Asignar δant = δ
4. Calcular las funciones de pertenencia usando (2-78)
5. Calcular los prototipos mediante (2-82)
6. Calcular δ usando (2-83)
7. Si |δant - δ|/δ < ε entonces terminar, si no volver a 3.

Algoritmo 2-3: Algoritmo de las c-medias

Como se ha comentado, este algoritmo muestra la estructura interna del conjunto de datos de entrada. De este modo datos similares, o que estén cerca geométricamente pertenecerán al mismo grupo, mientras que será difícil que un dato lejano al prototipo de un grupo, formase parte de él.

En cualquier caso el algoritmo muestra una serie de inconvenientes como son:

1. No detecta grupos vacíos o degradados.
2. Puede acabar con varios prototipos idénticos.
3. Como solución final encuentra el mínimo local más cercano en el espacio a la partición inicial.

Evidentemente, el más serio de los inconvenientes es el último, ya que como se deduce, la solución final, no solo va a depender de la partición inicial sino que además no asegura llegar a la partición más óptima.

2.8.3.2 Algoritmo de las c medias difuso

El algoritmo de las c medias difuso (fuzzy c means algorithm) fue propuesto por [7] y es una generalización del algoritmo de las c medias. La principal modificación que se hace es con respecto a los valores que la función de pertenencia μ, puede alcanzar. Así, mientras en el algoritmo de las *c* medias la función μ estaba definida en el rango {0, 1}, ahora cada grupo se considera como un conjunto difuso, por lo que μ se define en el rango [0, 1]. Esto implica, que un vector de entrada pueda ser asignado a varios grupos, con valores de pertenencia distintos, que informarán de la cercanía del vector de entrada a sus respectivos prototipos.

Como salida se devuelve un conjunto de *c* prototipos, cada uno de los cuales es el prototipo de los vectores de entrada que han sido asignados al grupo que representa.

Los pasos del algoritmo de las *c* medias difuso son básicamente iguales a los del algoritmo de las *c* medias mostrados en el algoritmo 2-3, cambiando sólo el cálculo de algunas funciones. En principio la función de pertenencia del vector de entrada $\bar{x}_i$, al grupo representado por el prototipo $\bar{p}_j$, vendría dada por la siguiente ecuación:

$$\mu_{P_i}(\bar{x}_k) = \frac{1}{\sum_{i=1}^{c} \left(\frac{\| \bar{x}_k - \bar{p}_i \|}{\| \bar{x}_k - \bar{p}_j \|} \right)^{2/(\rho-1)}} \qquad \text{2-85}$$

Como se observa, además de los parámetros que se manejaban en el algoritmo anterior, ahora se introduce un nuevo parámetro ρ que indica el grado de "difusión" de la partición del conjunto de entrada. Así, si $\rho \to 1$, la partición obtenida se aproxima a una

partición nítida conseguida con el algoritmo anterior, mientras que si $\rho \to \infty$, el algoritmo produce una partición en la que todos los puntos pertenecen a todos los grupos.

Mientras que la matriz de partición *U* quedaría ahora de la siguiente forma:

$$U = \left\{ \mu \in \{0,1\} \mid \sum_{i=1}^{c} \mu_{P_i}(\bar{x}_k) = 1 \ \forall k \quad y \quad 0 < \sum_{k=1}^{n} \mu_{P_i}(\bar{x}_k) < n \quad \forall i \right\} \qquad 2\text{-}86$$

La ecuación para el cálculo de los prototipos también cambia para adaptarse a la forma de la nueva función de pertenencia, quedando:

$$\bar{p}_j = \frac{\sum_{i=1}^{n} \mu_{P_j}(\bar{x}_i)^{\rho} \bar{x}_i}{\sum_{i=1}^{n} \mu_{P_j}(\bar{x}_i)^{\rho}} \qquad 2\text{-}87$$

La distorsión se sigue calculando igual que en (2-83)

De todas formas el algoritmo presenta los siguientes inconvenientes:

1. Pueden aparecer de nuevo grupos idénticos.
2. De nuevo, el algoritmo termina en el mínimo local más cercano a la partición inicial.

Como se observa, aunque desaparezca el inconveniente de crear grupos vacíos, se siguen manteniendo, los otros dos problemas. Aunque también hay que resaltar que, con respecto al algoritmo anterior, este algoritmo es más costoso en tiempo de ejecución debido al incremento de cómputo que aparece en las nuevas fórmulas.

2.8.3.3 Algoritmo ELBG

Con este algoritmo se pretenden solucionar los problemas que presentan los algoritmos anteriores. Estos problemas son debidos principalmente a que estos algoritmos sólo producen cambios locales, hasta alcanzar el mínimo local más próximo al punto de partida.

Como ejemplo se va a partir de una situación como la que se muestra, en la que los círculos blancos indicarían los vectores $\bar{x}_i$, mientras que los negros indicarían la disposición inicial de los prototipos $\bar{p}_j$.

En este caso el prototipo aislado no se movería en todo el proceso mientras, ya que no está influenciado por ningún vector de entrada. Por otro lado si a la zona donde se encuentra el prototipo $\overline{p}_1$, se traslada algún prototipo, incluso de la zona donde se encuentran los prototipos $\overline{p}_2$ o $\overline{p}_3$, se produciría un mejor resultado ya que el conjunto de entrada, quedaría mejor cubierto. El problema es que debido al funcionamiento local de los algoritmos anteriores, esto tampoco va a ocurrir.

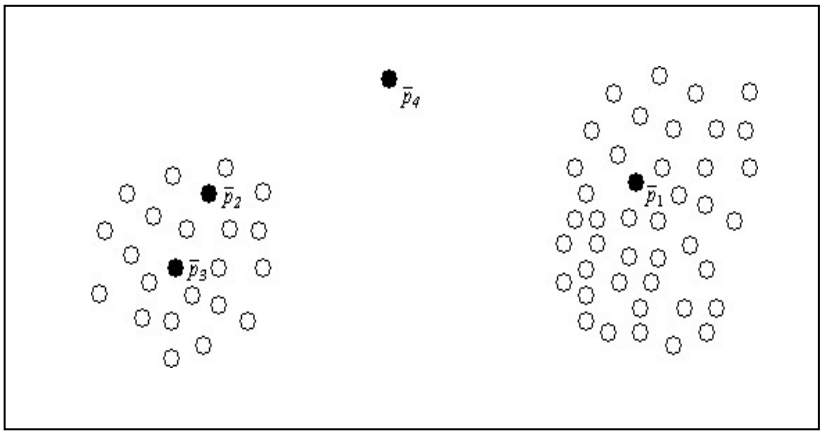


Figura 2-21: Ejemplo de prototipos mal colocados

El algoritmo ELBG propuesto por [123] intenta aportar una solución a estos inconvenientes. Este algoritmo es una extensión del de las *c* medias y se basa en el teorema de distorsión total enunciado por [46] que dice que: "Cada grupo hace una contribución igual a la distorsión total en una cuantización óptima de alta resolución".

Basándose en este teorema se define el concepto de utilidad de un prototipo $\overline{p}_j$ como:

$$u_j = \frac{\delta_j}{\overline{\delta}} \qquad 2\text{-}88$$

dónde $\overline{\delta}$, representa la distorsión media de la partición:

$$\overline{\delta} = \frac{1}{c}\sum_{i=1}^{c}\delta_i \qquad 2\text{-}89$$

El teorema de la distorsión total implica que en una partición óptima del conjunto de entrada, todos los clusters tendrían una utilidad igual. Así el algoritmo ELBG tratará de

llegar a esta condición, mediante migraciones de prototipos con utilidad menor que 1, a grupos donde los prototipos tengan una utilidad mayor que 1.

En el ejemplo esto implicaría que el grupo $\overline{p}_4$ tiene una utilidad 0, que las utilidades de los grupos $\overline{p}_2$ y $\overline{p}_3$ son menores que 1 y que la utilidad del grupo $\overline{p}_1$ es mayor que 1. Por tanto el algoritmo ELBG, intentaría desplazar el grupo $\overline{p}_4$ y alguno de los grupos $\overline{p}_2$ o $\overline{p}_3$ a la zona donde está el grupo $\overline{p}_1$.

1. Alojar en el grupo de llegada tanto el nuevo prototipo como el prototipo actual de ese grupo.
2. Recalcular posiciones de los prototipos para generar dos grupos.
3. Recalcular posiciones de los prototipos en zonas de salida.

Algoritmo 2-4: Rutina de migración de prototipos en ELBG

En general y para realizar la migración de un prototipo que tiene baja utilidad a un grupo que tiene mayor utilidad se realizarían los pasos mostrados en el algoritmo 2-4.

El primer paso consistiría en calcular una posición inicial dentro del grupo de llegada P_b, tanto para el prototipo que llega $\overline{p}_a$, como para el actual $\overline{p}_b$. Para esto se considera al paralelepípedo que contiene al grupo P_b, alojando a $\overline{p}_a$ y a $\overline{p}_b$, en su diagonal principal. Para esto la diagonal se divide en tres segmentos de forma que el segmento central tenga una longitud igual al doble de la longitud de los otros dos segmentos y se colocan los prototipos en los extremos del segmento central.

Después de realizar esta primera inicialización de los prototipos $\overline{p}_a$ y $\overline{p}_b$, se ajustan aplicando el algoritmo de la c medias de forma local la grupo P_b, dividiéndolo en los grupos P'_a y P'_b.

El último paso consiste en fusionar los grupos P_a y P_c en un nuevo grupo P_c y actualizar $\overline{p}_c$ teniendo en cuenta los puntos de las dos particiones a fusionar.

Cuando acaba el proceso de migración, se compara la distorsión de los grupos afectados antes del desplazamiento:

$$\delta_{ant} = \delta_a + \delta_b + \delta_c \qquad 2\text{-}90$$

con su distorsión después del desplazamiento:

$$\delta_{pos} = \delta'_a + \delta'_b + \delta'_c \qquad 2\text{-}91$$

Si $\delta_{pos} \leq \delta_{ant}$ se acepta la migración, en caso contrario se rechaza.

Con este algoritmo ya se resuelven los inconvenientes de los algoritmos anteriores.

2.8.3.4 Algoritmo de clustering difuso condicional

Este es el primer algoritmo, de los descritos, que se considera como un algoritmo supervisado. Esto supone que al realizar la partición en grupos, no sólo se tienen en cuenta las variables de entrada, sino también la/s de salida.

Las bases del algoritmo de clustering difuso condicional (Condicional Fuzzy Clustering Algorithm) [104] se encuentran en el algoritmo de las *c* medias difuso, suponiendo una extensión de éste. La parte condicional del mecanismo de agrupación, reside en las variables de salida y_1, …, y_n de las correspondientes variables de entrada. Es decir la variable de salida y_k del vector de entrada $\bar{x}_k$, describe el nivel en que este vector interviene en la construcción del grupo. Esto se realiza definiendo, una etiqueta lingüística en el espacio de salida, que se va a corresponder con un conjunto difuso *B*, *B*: $\Re \rightarrow [0, 1]$. De este modo $y_k = B(f_k)$, donde f_k la salida de la red para el vector $\bar{x}_k$, expresa el grado de pertenencia de y_k a *B*.

El problema de agrupamiento se reformula como "Agrupar los datos considerando que *y* es *B*". La forma en que y_k se asocia a las funciones de pertenencia de la variable $\bar{x}_k$, $\mu_{P_1}(\bar{x}_k)$, …, $\mu_{P_c}(\bar{x}_k)$ no es única. Sin embargo si se tiene que cumplir que f_k se distribuya aditivamente a lo largo de las entradas de la columna *k* de la matriz de partición, lo que significa que:

$$\sum_{i=1}^{c} \mu_{P_i}(\bar{x}_k) = f_k \quad k = 1,\ldots,n \qquad 2\text{-}92$$

Esto implica que si el vector de entrada $\bar{x}_k$, es poco significativo en el contexto del conjunto difuso *B*, entonces $B(f_k) = 0$, lo que implica que ese patrón de entrada será excluido del proceso de clustering o agrupamiento ya que $\mu_{P_i}(\bar{x}_k)$ para cualquier valor de *i*. Por otro lado si $B(f_k) = 1$, el vector de entrada $\bar{x}_k$, contribuirá de forma máxima al proceso de clustering. Estos cambios implican que las matrices de partición queden:

$$U = \left\{ \mu \in \{0,1\} \mid \sum_{i=1}^{c} \mu_{P_i}(\bar{x}_k) = f_k \ \forall k \quad y \quad 0 < \sum_{k=1}^{n} \mu_{P_i}(\bar{x}_k) < n \quad \forall i \right\} \qquad 2\text{-}93$$

$$\mu_{P_i}(\bar{x}_k) = \frac{f_k}{\sum_{i=1}^{c} \left(\frac{\| \bar{x}_k - \bar{p}_i \|}{\| \bar{x}_k - \bar{p}_j \|} \right)^{2/(\rho-1)}} \qquad 2\text{-}94$$

Una importante consecuencia de este método es que se reduce el costo computacional al dividir el problema original, en una serie de problemas condicionados por el contexto.

2.8.3.5 Algoritmo de estimación de grupos alternante (ACE)

En los algoritmos de clustering presentados hasta ahora, existe una determinación inicial de los parámetros que caracterizan los métodos, como por ejemplo, la forma de las funciones de pertenencia, la actualización de los prototipos de los grupos, etc. En [122] se presenta el algoritmo estimación de grupos alternante (Alternating Cluster Estimation ACE) o una herramienta donde los parámetros citados anteriormente se pueden directamente por el usuario.

En [122] se presentan ejemplos del modelo donde no se impone como restricción la optimización de una función objetivo en particular. Así, si el usuario selecciona ecuaciones que no están relacionadas con una función objetivo, las funciones de pertenencia y los grupos se obtienen actualizando alternativamente conjuntos de parámetros.

Así el concepto de función objetivo se reemplaza por un nuevo concepto denominado estimación de grupos alternantes. Las funciones de pertenencia que aquí se proponen se inspiran en sistemas usados en sistemas neuro-difusos.

2.8.3.6 Algoritmo de clustering para aproximación de funciones

La mayoría de los métodos presentados anteriormente, han sido diseñados para la resolución de problemas de clasificación, esto hace que no se adapten bien al problema de la aproximación de funciones. Algunas de las diferencias [55] más importantes, que se resaltan entre estos problemas son:

1. El espacio de salida de ambos problemas es muy diferente. Mientras en clasificación se puede considerar como discreto (conjunto de etiquetas), en aproximación de funciones es continuo.

2. En la aproximación de funciones no se persigue una coincidencia máxima entre la salida del modelo y la salida objetivo, sino que se tolera cierto error. Sin embargo en problemas de clasificación si se persigue esta coincidencia.

3. En los problemas de clasificación no se tienen en cuenta propiedades de interpolación que si se tienen en cuenta en la aproximación de funciones.

El algoritmo de clustering para aproximación de funciones (clustering for function approximation) [55], describe un método que va a incorporar una serie de características que van a ayudar en la tarea de la aproximación de funciones.

De todas formas su objetivo final no será un modelo de aproximación tan exacto como por ejemplo se puede conseguir utilizando métodos matemáticos de minimización tradicionales como: el del gradiente conjugado, Newton-Rapson o Levenberg-Marquardt [30], [72], [108]. Estos pueden conseguir soluciones prácticamente óptimas si parten de un conjunto de parámetros inicial adecuado. Así el objetivo de este algoritmo será obtener una configuración lo más cerca posible del óptimo global.

Una de las bases de este algoritmo será partir del análisis de la variabilidad de la forma de función. Así, en zonas donde función se mantenga constante, introducirá sólo un prototipo, mientras que en zonas donde la función sea más variable será necesario introducir más prototipos.

Para conseguir un compromiso entre estos parámetros se siguen los pasos mostrados en el Algoritmo 2-5.

1. Realizar la asignación $\delta = \infty$
2. Asignar aleatoriamente el conjunto inicial de prototipos
3. Asignar $\delta_{ant} = \delta$
4. Calcular las funciones de pertenencia usando 2-78
5. Calcular los prototipos mediante 2-82
6. Calcular δ_j usando 2-95
7. Calcular δ usando 2-83
8. Realizar migración
9. Si $|\delta_{ant} - \delta|/\delta < \varepsilon$ entonces terminar, si no volver a 3

Algoritmo 2-5: Algoritmo de clustering para aproximación de funciones

Como se puede observar este algoritmo utiliza elementos tanto del algoritmo de las *c* medias como del algoritmo ELBG. Así y tras una inicialización aleatoria se entra en el bucle principal donde, al igual que en el algoritmo de las c medias, se realiza el cálculo de $\mu_{P_j}(\bar{x}_i)$ y de $\bar{p}_j$ como en el algoritmo de las *c* medias. Después y para calcular δ_j, un elemento del algoritmo ELBG, utilizaría la siguiente expresión:

$$\delta_j = \sum_{i:\mu_{ij}=1}^{n} \| \bar{x}_i - \bar{p}_j \|^2 \cdot (f(\bar{x}_i) - f(P_j))^2 \qquad 2\text{-}95$$

donde $f(P_j)$ es la media de la salida de la función objetivo para todos los vectores de entrada o entrenamiento que pertenecen al grupo P_j:

$$f(P_j) = \frac{\sum_{\bar{x} \in P_j} f(\bar{x}_i)}{|P(j)|} \qquad 2\text{-}96$$

Una vez calculados los valores δ_j, δ se calcula como en 2-83.

En el último paso del ciclo se realizaría una migración de prototipos de forma similar a como se ha descrito para el algoritmo ELBG.

2.8.4 Algoritmos para la inicialización de radios de RBFs

Para la determinación del radio inicial de una RBF existen varias técnicas clásicas. La misión de estas técnicas será encontrar unos valores iniciales para los radios de las funciones base, de forma que se cubra convenientemente el espacio de entrada. Estos métodos se caracterizan porque se usan una vez que se ha fijado el centro de la función

base a tratar con respecto a los centros del resto de las RBFs. Otra característica de estos métodos es que suelen trabajar con la distancia, normalmente euclídea, entre las RBFs.

Conviene decir que estas técnicas no sólo se pueden tener en cuenta la hora de diseñar una primera fase de inicialización. Así, también se pueden usar como referencia, cuando se introducen RBFs en fases posteriores, para las que las que normalmente habrá que establecer un radio.

Si se hace una recopilación del funcionamiento de estos algoritmos, la primera solución, consistiría en el establecimiento de un radio similar para todas las funciones base.

Esta técnica, a pesar de su simplicidad se ha probado suficiente para obtener un aproximador universal en [102] y [103].

Sin embargo parece más lógico establecer un radio individual y adaptado para cada función base. En [96], se demuestra que con esta variante se mejoran las prestaciones. El objetivo que persigue esta segunda opción, es conseguir cierto grado de solapamiento entre las funciones base vecinas para que interpolen de forma suave y continua las regiones del espacio de entrada que representan.

Entre las heurísticas más usadas se encuentran:

2.8.4.1 Heurística de los k vecinos más cercanos (KNN):

Esta heurística fija el radio de cada función base a un valor igual, a la distancia media a los centros de k funciones base más cercanas [94]. Esta heurística pasa a ser la heurística del vecino más cercano si $k = 1$.

2.8.4.2 Heurística de la distancia media de los vectores de entrada más cercanos (CIV):

En este caso se determina el radio de la RBF j-ésima de acuerdo con la siguiente ecuación:

$$d_j = \frac{\sum_{\bar{x}_i \in C_j} \bar{x}_i}{|C_j|} \qquad 2\text{-}97$$

donde C_j es el conjunto de vectores de entrada que están más cerca de $\bar{c}_j$ que de cualquier otro centro y $|C_j|$ representa el número de vectores que lo componen. Esta heurística produce menor solapamiento que la anterior aunque $k = 1$.

2.8.4.3 Algoritmos incrementales/decrementales

En este apartado se describirán los conocidos como métodos incrementales y/o decrementales. Un método se define como incremental cuando partiendo de una red prácticamente sin neuronas, va a ir agregándolas siguiendo algún criterio. Por el contrario un método se define como decremental cuando parte de una red compleja en cuanto su número de neuronas, y las va eliminando siguiendo también algunas directrices.

En este apartado se comenzará con la descripción del algoritmo RAN (Resource Allocation Network), para continuar después con sus posteriores modificaciones. El algoritmo RAN [106] intenta mejorar la investigación en el diseño de RBFNs que había hasta la fecha. En esta época por ejemplo los trabajos [94] consistían en utilizar un algoritmo de clustering, como por ejemplo el de las *c* medias (descrito anteriormente), para determinar los centros de las RBFs. El radio de una RBF determinada, se establecía a partir del cálculo de la RBF más cercana a ésta, mientras que para determinar los pesos se utilizaba el típico algoritmo LMS.

El método RAN es un método incremental, que por lo tanto parte de un número bajo de RBFs y que utiliza información local para ir añadiendo RBFs, y así alcanzar el diseño RBFN. La información local que se analiza cíclicamente son las muestras o patrones $(\bar{x}_k, y_k)$ del conjunto de entrenamiento. Dada una muestra, si se verifica que se cumplen dos condiciones, se introducirá una nueva RBF.

La primera condición que se tiene que verificar es que:

$$\| \bar{x}_k - \bar{c}_n \| > \delta(t) \qquad \text{2-98}$$

donde $\bar{x}_k$ es el vector de entrada actual, $\bar{c}_n$ es el centro de la RBF ϕ_n, más cercana según la distancia euclídea $\| \cdot \|$, al vector $\bar{x}_k$. $\delta(t)$ es un umbral de resolución variable con el número de patrones que se han presentado, donde *t* indica el número de patrón actual. Inicialmente $\delta(t) = \delta_{max}$, es decir el valor máximo que puede tomar y puede ser igual al tamaño del espacio de entrada. La distancia $\delta(t)$, va disminuyendo hasta que alcanzar un valor δ_{min}. En general y para disminuir el valor de $\delta(t)$ se utiliza la siguiente expresión:

$$\delta(t) = \max[\delta_{\max} \exp(-t/\tau), \delta_{\min}] \qquad \text{2-99}$$

donde τ es un constante.

La segunda condición que se tiene que verificar es que la diferencia entre la salida de la red $f(\bar{x}_k)$, y el valor objetivo dado por el patrón y_k, sea superior a un determinado valor, es decir:

$$| y_k - f(\bar{x}_k) | > \varepsilon \qquad 2\text{-}100$$

donde ε es un umbral que va a representar la precisión de la salida de la red.

Si no se cumplen las condiciones anteriores entonces se utilizan algoritmos tipo LMS para ajustar los pesos y centros de la red. Con esto el sistema comenzaría ofreciendo una representación de la función con mucho error con respecto a la original, para luego irla refinando mediante la inclusión de RBFs con menor radio.

Más tarde aparece el trabajo [75] que mejora la eficiencia del método RAN utilizando un método de minimización denominado Filtro de Kalman Extendido (Extended Kalman Filter, EKF) en lugar algoritmos LMS para ajustar los parámetros de la red.

El método general se denomina RANEKF y demuestra obtener mejores resultados en la aproximación de funciones y la predicción de series temporales.

Otro enfoque parecido es el propuesto por el algoritmo Growing Radial Basis Networks (GRBN) [78]. En este algoritmo se plantea un mecanismo de división de funciones base que cometan más error. Esto implica que la complejidad de la red va creciendo continuamente hasta que se cumpla una condición de parada que evalúa el compromiso entre el error de aproximación y la complejidad de la red.

Hasta ahora los métodos descritos solo tienen en cuenta la adicción de funciones base, pero nunca la eliminación de éstas, si en un momento determinado aportan poco al funcionamiento de la red. Para solventar este inconveniente aparecen los siguientes algoritmos.

El algoritmo Minimal RAN (M-RAN) [138] incorpora un mecanismo de poda de funciones contribuyan poco a la salida de la red. Para esto se determina una ventana o grupo de muestras del conjunto de entrenamiento, para las que se analiza la salida de cada función base, eliminado aquellas que no superan un umbral. Otra diferencia con respecto al método RAN es que para introducir una nueva RBF se tiene que cumplir también que el error cuadrático medio para un grupo de patrones del conjunto entrenamiento sea superior a un valor mínimo.

El algoritmo RAN se sigue modificando en [121], donde se incluyen la descomposición QR de Givens, la cual se usa para calcular los pesos de la red (RAN-GQRD) y para realizar la de poda de funciones base (RAN-P-GQRD) reduciendo de esta manera, la complejidad de la red final.

Otra posibilidad para implementar un mecanismo de poda en el algoritmo RAN es usar la descomposición SVD de la matriz de activación de la red para determinar las funciones base menos relevantes [124].

En [119] se presenta el algoritmo PG-BF Sequential Learning Algorithm donde se emplean hasta tres criterios para identificar funciones base que no contribuyan a la salida de la red.

También existe la metodología opuesta es decir partir de una red completa e ir eliminando RBFs menos relevantes hasta alcanzar algún criterio de parada. Para determinar este tipo de funciones se puede utilizar métodos como el OLS [18] el SVD [77]. Incluso han aparecido hibridaciones del método OLS con algoritmos evolutivos [19].

2.8.5 Métodos evolutivos

A continuación se describirán unos métodos que utilizan una estrategia evolutiva para el diseño de una RBFN. Estos métodos van a tener como característica común que van a implementar la evolución de una población, donde un individuo de la población representa una red completa.

Concretamente el primer método utilizará un algoritmo genético para el diseño de la red. El segundo método utiliza también como base un algoritmo genético, pero lo complementa con otras técnicas típicas en el desarrollo de redes neuronales como por ejemplo algoritmos de clustering, técnicas SVD, OLS, minimización de error, etc.

2.8.5.1 Una estrategia genética

En [116] se presenta un típico algoritmo genético que evoluciona una población de individuos, donde cada individuo representa una RBFN.

Una característica diferenciadora de este método es que un individuo no se representa mediante cadenas de bits o vectores de reales, si no que utiliza una serie de objetos predefinidos. Otras características de este método es que el tamaño de la población es fijo, se usa selección tipo tournament y un reemplazo elitista.

Para calcular los pesos se utiliza SVD. El algoritmo termina cuando se alcanzan un determinado número de iteraciones.

Los principales pasos de éste se muestran en el algoritmo 2-6.

Los operadores utilizados y el cálculo de la función de evaluación se muestran en los siguientes apartados.

2.8.5.2 Operadores utilizados

Los operadores que se utilizan son:

1. Operador de cruce binario: este operador trabaja cogiendo una serie consecutiva de neuronas de cada uno de los padres e intercambiándolas. No es necesario que estas secuencias de neuronas sean de igual tamaño.
2. Operador de mutación de centros: este operador cambia en un porcentaje los elementos del vector centro de una neurona. Para esto se suma a cada uno de estos elementos un valor aleatorio que sigue una función de probabilidad gausiana de media cero y desviación 0.1.
3. Operador de mutación radios: consiste en la suma de un valor aleatorio, que sigue una función de probabilidad igual que la definida anteriormente, una RBF de una determinada red.
4. Operador de inclusión de funciones base: este operador duplica cada una de las neuronas con una probabilidad dada (la misma para todas). Cuando una neurona se duplica los valores del centro y del radio se modifican mediante una función gausiana.
5. Operador de eliminación de funciones base: elimina un número aleatorio de funciones base.

1. Inicialización
2. Repetir mientras no se cumpla condición
   a. Seleccionar individuos
   b. Eliminar al resto de la población
   c. Generar nuevos individuos mediante el operador de cruce
   d. Aplicar el resto de operadores a los nuevos individuos
   e. Calcular el peso de los nuevo individuos utilizando SVD
   f. Eliminar neuronas cuyos pesos sean muy cercanos a cero
   g. Evaluar las redes

Algoritmo 2-6: Pasos principales de un algoritmo genético típico

2.8.5.3 Función de evaluación

Como función de evaluación o función que calcula el valor de adaptación de un individuo/red se usa:

$$V.A. = \frac{1}{\sqrt{\frac{\sum_{i=0}^{n-1}(y_i - f(x_i)}{n}}} \qquad 2\text{-}101$$

Los valores que se pasan para el cálculo de esta función son los relativos a un conjunto de validación con el que no se ha estado entrenando.

2.8.5.4 Un método evolutivo multiobjetivo

En [54] se describe un algoritmo evolutivo multiobjetivo para el diseño y optimización de los parámetros de una RBFN a partir de un conjunto de datos de entrenamiento. Las RBFNs diseñadas se aplican al problema de la aproximación de funciones.

La base es un algoritmo evolutivo, concretamente un genético, que evoluciona una población donde cada individuo representa una RBFN. El valor de adaptación que se establece para cada individuo es la raíz cuadrada del error cuadrático medio normalizado. Una vez que los individuos han sido evaluados se usa una técnica multiobjetivo, que trata las soluciones en función de dos objetivos: complejidad y error de una red. Con esto el algoritmo puede optimizar redes con distinto número de funciones base a la vez.

A continuación se detallarán las principales características del algoritmo.

2.8.5.5 Creación de la población inicial

Dado que el algoritmo va a tener la capacidad de evolucionar redes de distintas capacidades a la vez, se diseña un algoritmo de inicialización que cree individuos con diferentes características. Para eso este algoritmo combina diferentes algoritmos de agrupamiento o clustering para la inicialización de centros como son: el algoritmo de las *c* medias, el ELBG o el CFA. Para asignarle los radios a estas RBFs se utilizan algoritmos de inicialización de radios como el de los *k* vecinos más cercanos (KNN) con $k = \{1, 2, 3\}$ y de la distancia media de los vectores de entrada más cercanos (CIV), para un rango de tamaños de red comprendidos entre dos umbrales $[m_{min}, m_{max}]$.

Si hacen falta más redes para completar la población inicial, se obtendrán mediante la aplicación de cambios aleatorios a las redes creadas anteriormente. Si por el contrario el tamaño de la población inicial es mayor de lo necesario, se ordena a los individuos según su rango y se eliminan las peores soluciones.

2.8.5.6 Operadores evolutivos

Se utilizan una serie de operadores evolutivos para cambiar la estructura de la red añadiendo o eliminando funciones base. Para realizar estas variaciones en el número de RBFs que forman una red se estudiarán aspectos como utilidad de las RBFs o zonas del espacio de entrada que necesitan mal cubiertas por la red. Los operadores utilizados son:

1. AME (Adición de una función base en el punto de mayor error). Su funcionamiento es sencillo y consiste en añadir una función base con centro en el patrón de entrenamiento en el que la red produzca el mayor error de aproximación. El radio de la RBF se fijará utilizando las técnicas de asignación de radio citadas anteriormente. Una vez fijados el centro y el radio de esta nueva función base, los pesos de la nueva red se calculan mediante el método de Cholesky.

2. DIV (División de funciones base). El funcionamiento de este operador consiste en detectar la función base que produzca un mayor incremento del error de aproximación y dividirla en dos funciones base para que la red cubra mejor esta zona del espacio de entrada. Para estimar la contribución de cada función base ϕj al error total de la red se usa el parámetro ej:

$$e_j = \sum_{k=1}^{n} \frac{\phi_j(\bar{x}_k)}{\sum_{i=1}^{m} \phi_j(\bar{x}_k)} | y_k - f(\bar{x}_k) | \quad j = 1,...,m \qquad \text{2-102}$$

donde n es el número de muestras de entrenamiento y m es el número de funciones base de la red. Para elegir la función base a la que aplicar el operador, se genera una distribución de probabilidad en la que cada función base ϕj tiene una probabilidad de aplicación proporcional a ej, y se escoge una función aleatoriamente. Una vez escogida la función base ϕj se aplica un algoritmo con sólo dos funciones base a los puntos que cubre la función ϕj, y una vez que la red ha sido creada, se sustituye la función ϕj de la red original, por las dos nuevas funciones base.

3. EOLS (Eliminación de funciones base usando OLS). En este caso se utiliza el método OLS para elegir una función base a eliminar. Para esto se aplica el algoritmo OLS que calculará un vector de coeficientes de reducción de error. Al igual que antes, para elegir la RBF a eliminar, se construye una distribución de probabilidad en la que cada función base tendrá una probabilidad de ser eliminada inversamente proporcional a su coeficiente de reducción de error.

4. ESVD (Eliminación de funciones base usando SVD). Una vez que se aplica la técnica SVD a la matriz de activación de la red se obtiene el vector de valores singurlares asociados a cada una de las funciones base. Partiendo de la base de que cuanto menor sea el valor singular asociado a una función base, menos afectará su eliminación al error de aproximación de la red, la función a eliminar se escogerá con una probabilidad al valor singular asociado.

2.8.5.7 Selección

El proceso de selección/reemplazo sigue un algoritmo elitista, adaptando la implementación a una técnica multiobjetivo. El utilizar un algoritmo selección elitista indica que nunca se perderá la mejor solución obtenida hasta el momento. Para aplicar este algoritmo, se parte evidentemente de una población evaluada o una población en la que a cada individuo se le ha calculado su raíz cuadrada del error cuadrático medio normalizado.

Antes de realizar este cálculo se aplican unas pocas iteraciones del algoritmo Levenberg-Marquardt que ayudan a decidir sobre los mejores individuos. Con estas premisas el algoritmo va guardando los elementos pertenecientes a la frontera del pareto.

2.8.5.8 Ajuste final de las soluciones

Una vez que finaliza el algoritmo evolutivo, este devuelve el conjunto de soluciones encontradas, éstas se mejoran bastante aplicando el algoritmo Levenberg-Marquardt de forma completa. Con esta estrategia se alcanza el objetivo de utilizar un proceso evolutivo para obtener un conjunto de soluciones de las que pueda partir el algoritmo de minimización. Con esto se consigue obtener un conjunto de soluciones muy cercano al óptimo.

2.8.5.9 Criterio de parada

El criterio de parada en ese caso consiste en el cálculo de la pendiente de una recta de regresión de los mejores individuos encontrados en las últimas generaciones, para cada número *i* de funciones base.

De esta manera cuando estas pendientes están debajo de un umbral se detendrá el algoritmo.

2.8.5.8 Evolución multiobjetivo

Para implementar la evolución multiobjetivo se introducen una asignación de pseudo-aptitudes basadas en rangos combinando características del algoritmo MOGA [42] y NSGA [129], para de esta manera distribuir apropiadamente la presión selectiva entre los individuos.

Pero para que esta presión selectiva no sea excesiva se introduce un método para la compartición de pseudo-aptitudes, basado en la distancia fenotípica entre dos redes, que va a dar lugar a la creación de nichos.

2.8.6 Algoritmos coevolutivos cooperativos

En este caso se describirá el trabajo propuesto en [135], que es un claro ejemplo de algoritmo coevolutivo cooperativo para el desarrollo de RBFNs. De hecho es un trabajo, donde el algoritmo que se propone, tiene muchos puntos en común con el que se propone en esta tesis.

Como característica propia de este tipo de algoritmos, un individuo de la población representa una neurona o RBF, en lugar de una RBFN completa. El conjunto de la población va a formar una RBFN completa. Así cuando el algoritmo termine se tendrá un conjunto de RBFs coadaptadas, que trabajan bien juntas y que, por lo tanto, dan lugar a una RBFN eficiente.

El algoritmo evolutivo utilizado es un genético y los elementos de una RBF que van a evolucionar son el centro y el radio. El problema de la aportación de crédito se resuelve teniendo en cuenta la contribución de la RBF a la eficiencia de la red. Para mantener la diversidad, se usan conceptos de la técnica de nichos, de manera que se tiene en cuenta el solapamiento entre los trabajos que las RBFNs realizan.

A continuación se describen los principales elementos que caracterizan este algoritmo:

2.8.6.1 Esquema de selección

Tal y como se ha comentado, se usa un algoritmo genético para guiar la evolución. Este algoritmo genético irá dando lugar a sucesivas generaciones G_k, de modo que cada generación, está constituida por una población de RBFs que forman una RBFN.

Para generar la población de G_{k+1}, a partir de G_k se utiliza un procedimiento de selección basado en el valor de adaptación o fitness. Concretamente, el algoritmo utiliza un esquema de selección proporcional en una población de tamaño fijo. Esto implica dos efectos. El primero es que cada individuo de la población de G_k, tiene una probabilidad de ser seleccionado igual a su valor de adaptación.

El segundo efecto es que este valor de adaptación indicará el número de copias que existirán de un individuo en la siguiente generación G_{k+1}. Una vez que se haya formado esta generación G_{k+1}, se aplicarán los operadores definidos, se evaluarán los nuevos individuos, y así cíclicamente funcionará el algoritmo.

2.8.6.2 Asignación de crédito: Valor de adaptación

Para evaluar la población perteneciente a una determinada población los autores parten de dos premisas principales. La primera es que la evaluación debe dar soporte a un proceso competitivo en el que las RBFs que trabajen mejor deben desplazar a aquellas que trabajen peor.

La segunda premisa es que la evaluación debe también provocar un proceso cooperativo, de modo que el trabajo de las distintas RBFs se sume para, por ejemplo, aproximar el valor de una función.

Para conseguir los objetivos anteriores normalizan las salidas de las RBFs, de modo que una RBF normalizada $\bar{\phi}_i$ se define cómo:

$$\bar{\phi}_i(x) = \frac{\phi_i(x)}{\sqrt{\sum_{j=1}^{p} \phi_i^2(x_j)}} \qquad \text{2-103}$$

donde p es el número de muestras. El valor de adaptación propuesto para una RBF $VA(\phi i)$ es:

$$VA(\phi_i) = \frac{|\, w_i \,|^{\beta}}{E(|\, w_{i'} \,|^{\beta})} \qquad 2\text{-}104$$

donde $E(x_i)$ devuelve la media de los valores x_i y β se establece al valor 3/2. Con este valor de adaptación los autores demuestran que cumplen sus objetivos.

2.8.6.3 Codificación

Los elementos que se codifican en este algoritmo son aquellos sobre los que se realiza la evolución, es decir, el centro y el radio de cada RBF. Para esto se utiliza el esquema de codificación típica utilizado en los algoritmos genéticos, es decir, las cadenas de bits. Una cadena de bits codificará pues, el centro y radio de una RBF.

2.8.6.4 Operadores

Los operadores se van aplicar a la población tras el proceso de selección explicado anteriormente y son tres: un *operador de recombinación* o cruce, un *operador de mutación* y un *operador de mutación suave*. Existe una diferencia entre los dos primeros y el último. Así, los operadores de recombinación y mutación actúan sobre los genotipos o cadenas de bits directamente, sin tener en cuenta los parámetros que estos bits codifican, ni los límites entre los parámetros. Por otro lado, el operador de mutación suave decodifica la cadena de bits, en los correspondientes valores reales, les aplica una perturbación y los vuelve a codificar otra vez.

1. El operador de recombinación el típico operador de cruce de dos puntos, de modo que se delimitan aleatoriamente dos posiciones de los bits, y los bits que existen entre la posición de inicio y final se intercambia. La probabilidad de aplicación de este operador de cruce depende la distancia euclídea entre las dos cadenas, de modo que decrece cuando esta aumenta.

2. El operador de mutación suave se aplica a dos cadenas que no se han recombinado con un operador de cruce, por lo que es complementario a él. Este operador aplica una pequeña perturbación aleatoria al centro y radio, dentro de un rango.

3. El operador de mutación actúa sobre un bit alterándolo con una probabilidad 1/m, donde m es el número de bits que conforman todas las cadenas. Este operador se aplica independientemente de los otros.

2.8.6.5 Entrenamiento de la RBFN

Para calcular los pesos de una RBFN, se utiliza el algoritmo LMS, ya que según los autores es un algoritmo cuyo coste computacional es bajo. Este algoritmo se aplica, pasándole un número *n* de veces pequeño, el conjunto de entrenamiento, con la idea de sólo perfilar los pesos de las distintas RBFs. Este número *n*, va incrementándose a lo largo del tiempo. Al final se aplica el algoritmo SVD para determinar de forma más exacta los pesos.

2.9 Conclusiones

En este capítulo se han presentado las ideas más significativas de lo que se que conoce por Softcomputing, así como una simple reseña de una posible taxonomía.

Se describieron los paradigmas que históricamente han definido el trabajo dentro del campo de la computación evolutiva y la resolución de problemas de optimización, mediante técnicas evolutivas.

También se hizo referencia a las características más importantes de las Redes de Función Base radial, entre las que se destacan su capacidad implícita en la aproximación funcional. Se ha hecho una revisión bibliográfica de los algoritmos que se emplean en el diseño de estas redes, entre los que se destacan los métodos matemáticos, los de clustering, los incrementales o los evolutivos. Algunos de estos métodos se utilizan en mayor o menor medida en el algoritmo de diseño de Redes de Funciones de Base radial que se describe en el siguiente capítulo.

En el siguiente capítulo, haremos una descripción del Proceso objeto de estudio, de manera que se puedan conocer los elementos fundamentales que nos permitan definir el alcance y planteamiento de este trabajo.

Capítulo 3

Planteamiento del Problema

En este capítulo, en su primera parte se realiza una breve descripción del proceso industrial de Punta Gorda, posteriormente se plantea el problema que ha sido objeto de investigación.

Todos los detalles que se recogen en este capítulo han sido aportados por documentos elaborados por la propia industria, así como de la experiencia adquirida por más de 60 años de explotación de yacimientos de níquel [14].

Para realizar el planteamiento del problema, es importante obtener la mayor cantidad posible de información sobre el funcionamiento del sistema objeto de estudio.

El actual nivel de automatización de Punta Gorda, ha permitido crear una base de datos sobre la operación tecnológica de todas sus áreas y en particular del proceso de reducción de mineral. La disponibilidad de estos datos es lo que nos permite un acercamiento al comportamiento de este proceso, y a partir de ahí, aplicar técnicas de Softcomputing para abordar la complejidad del proceso y plantear estrategias que permitan optimizar aspectos importantes que brinden resultados dentro de los parámetros deseados.

El níquel se encuentra en la naturaleza formando silicatos, óxidos, sulfuros, sulfatos, etc. Igual que muchos otros minerales; puede ser explotado a cielo abierto y subterráneo. Es un mineral de gran demanda en la industria, principalmente para la obtención de aceros de gran calidad y en muchísimas aleaciones con Cobre, Cromo, Aluminio, Plomo, Cobalto, Manganeso, Plata y Oro [22].

Es, después del manganeso, el metal más usado en ferroaleaciones, pero también tiene otras numerosas aplicaciones, proporciona a las aleaciones dureza, tenacidad, ligereza, cualidades anticorrosivas, térmicas y eléctricas. Comúnmente se comercializa en forma de lingotes, municiones, pellets y polvo, así como en forma de óxido conteniendo de 75 a 90% de níquel.

El níquel es un elemento bastante abundante, constituye cerca de 0.008% de la corteza terrestre y 0.01% de las rocas ígneas, se presenta además en pequeñas cantidades en plantas, animales, en el agua de mar, el petróleo y en la mayor parte del carbón.

Una de las principales ventajas comparativas específicas de estos yacimientos, es que constituyen minas a "cielo abierto"; y que se encuentran muy cerca de las fábricas. Estas condiciones especiales permiten obtener costos de operación en mina y de procesamiento del mineral relativamente bajos.

El grueso de la producción de níquel en Cuba pertenece al grupo de Clase II, éstos se caracterizan por poseer un contenido de níquel muy amplio de hasta 99.8%, lo que resulta muy importante en la fabricación de aceros inoxidables, y explica porqué este tipo de producto cubre el 60% del mercado internacional de este mineral. A lo anterior se añade la presencia de un rango moderado de elementos residuales, haciendo posible su aplicación en diferentes procesos industriales [97].

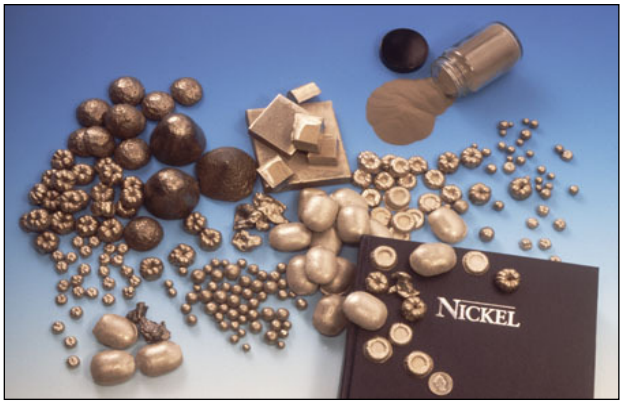


Figura 3-1: Productos obtenidos del Níquel

Desde el punto de vista tecnológico existen en CUBA dos procesos de obtención de Níquel: la Lixiviación Carbonato Amoniacal o Proceso CARON y la Lixiviación Ácida. El esquema tecnológico de la Planta de Níquel de Punta Gorda, al que se hacemos referencia en este trabajo, está basado en el esquema de lixiviación carbonato – amoniacal del mineral reducido o proceso CARON.

Esta tecnología presenta una serie de ventajas:

1. Es un proceso que se realiza en condiciones de presión atmosférica.

2. El equipamiento tecnológico del proceso se distingue por su sencillez y amplia utilización de los aparatos conocidos (hornos de soleras múltiples, espesadores, columnas de destilación, etc.).

3. El esquema amoniacal permite las mezclas de los minerales lateríticos y serpentínicos, mientras que el esquema de lixiviación con ácido sulfúrico permite solamente la elaboración de la fracción laterítica.

Como puntos débiles de este proceso podemos citar:

1. El bajo porciento de extracción: entre 75 – 76 % de Ni y el 25 – 30 % de Co.
2. El alto consumo de recursos energéticos: electricidad y petróleo.

3.1 Resumen del Proceso Industrial:

El mismo comprende las siguientes actividades:

3.1.1 Mina:

Suministra el mineral a la fábrica; está situada a 1–1.5 Km. Al Suroeste del área industrial. El mineral de la capa superior está compuesto de Limonita y el cuerpo de Laterita y Serpentina (blanda). Los componentes fundamentales del mineral son el Níquel, el Cobalto y como acompañante en cantidades considerables el Hierro. La transportación se realiza directamente desde las excavaciones hasta el área de recepción por camiones volquetas.

3.1.2 Planta de Preparación de Mineral:

En esta planta el mineral se somete a un proceso de secado y molienda y se suministra a los silos de almacenaje, de donde se bombea a los hornos de reducción.

3.1.3 Planta de Hornos de Reducción:

En esta planta ocurre el proceso de reducción del Ni contenido en el mineral. Está constituida por 24 hornos de soleras múltiples los cuales descargan el mineral a los enfriadores (12), de donde pasa a la planta de Lixiviación y Lavado.

3.1.4 Planta de Lixiviación y Lavado:

La pulpa de mineral reducido pasa por 3 sistemas paralelos de tres etapas de Lixiviación a contracorriente con el licor carbonato - amoniacal. La Lixiviación se realiza con el licor carbonato – amoniacal en los espesadores por medio de la aereación de la pulpa con aire (en los turboareadores). Luego de la Lixiviación la pulpa se envía al sistema de Lavado (dos en paralelo). El Licor enriquecido en Ni y Co es enviado a la planta de Separación de Cobalto, la pulpa de desecho es enviada a la planta de Recuperación de Amoniaco.

3.1.5 Planta de Separación de Cobalto:

El Licor enriquecido en Ni y Co se somete a una inyección de Hidrosulfuro de Amonio o Sulfhidrato de Sodio para precipitar el Co en forma de sulfuro, este producto se envasa en Big–Bag y se comercializa.

El Licor descobaltizado se envía a la planta de Recuperación de Amoniaco.

3.1.6 Planta de Recuperación de Amoniaco:

El licor carbonato amoniacal enriquecido en Ni recibe un tratamiento con vapor en las torres de destilación obteniéndose el Carbonato Básico de Níquel. La pulpa de desecho de la última etapa de lavado se envía a las torres de destilación de Colas donde recibe tratamiento con vapor para la recuperación del licor amoniacal contenido en esa pulpa.

El producto de desecho(cola) es enviado a la presa de Cola. La pulpa de Carbonato de Ni se envía a la planta de Calcinación y Sinter.

3.1.7 Planta de Calcinación y Sinter:

Luego de filtrado el carbonato básico de Ni es alimentado a los hornos de calcinación para la obtención del óxido de Ni, que es utilizado en el proceso de sinterización en las máquinas destinadas para este fin; obteniéndose el óxido Sinter que constituye el producto final de la planta y de la fábrica. Este producto es envasado en Big – Bag y trasladado al puerto para su comercialización.

3.1.8 Plantas Auxiliares:

Central Termoelectrica, Potabilizadora de Agua y Tratamiento Quimico de Agua.

Dentro de este proceso de recuperación de Níquel, la Reducción del Mineral es uno de los más significativos, pues de acuerdo a los niveles de extracciones que se obtengan en esta etapa, así serán los valores finales de producción que se alcancen.

Por ello el interés mostrado en el estudio de las variables que caracterizan al mismo.

3.2 Descripción del Proceso, Planta de Hornos Reducción.

El objetivo del proceso que se realiza en esta área es reducir el óxido de níquel a níquel metálico, haciéndolo apto para la lixiviación amoniacal. Para ello esta área cuenta con la instalación de 24 hornos, 12 transportadores rotatorios e igual número de enfriadores.

El mineral antes de ser sometido a proceso de reducción se somete a un proceso de secado y molienda en la planta de preparación del mineral. En está planta al mineral se le elimina la humedad hasta un 4 % aproximadamente y se muele hasta una fineza de 0,074 mm. También en transportadores de banda se alimenta alrededor de 2,9 % de petróleo tecnológico, por lo que se logra una homogeneización bastante completa con la desventaja de una pérdida de combustible en el trayecto hasta su alimentación

El mineral, después de pasar por la sección de molienda, es enviado mediante transporte neumático a unos silos como forma de almacenaje, los cuales tienen una capacidad de 1500 ton cada uno, lo que facilita una operación en los hornos de unas 16 horas.

En la sección de los silos se encuentran 9 bombas de una capacidad de 120 ton/h hora, mediante las cuales el mineral es bombeado hasta las tolvas de los hornos que son 12 en total, dispuestas una para cada dos hornos. Estas tolvas permiten realizar una operación de 8 horas a cada horno.

Una vez el mineral en las tolvas, éste pasa a los dosificadores de pesaje automático que son los equipos encargados de garantizar una alimentación uniforme al horno a través del pesaje que este realiza de acuerdo al tonelaje fijado, estos equipos tiene una capacidad hasta 22 ton/h.

Después que el mineral es pesado, se produce la descarga del mismo a un sinfín alimentador el cual transporta el mineral al horno hacia el hogar cero.

El mineral una vez dentro del horno es sometido al proceso de reducción, el cual se logra estableciendo un perfil de temperatura dentro del mismo y una concentración determinada de gases reductores ($CO - H_2$), para ello el horno dispone de cámaras de combustión con quemadores de alta presión para la combustión incompleta del petróleo, el cual permite además de lograr el perfil de temperatura enriquecer la atmósfera reductora.

El proceso de reducción se efectúa en un horno de hogares múltiples de 23,5 m de alto y 6,8 m de diámetro, con 17 hogares ó soleras. Además, el horno cuenta con un eje central al cual se le articulan 68 brazos (4 en cada hogar). Estos brazos tienen dispuestos diente o paletas con la que mediante la rotación del eje central facilitan el traslado de mineral de un hogar a otro.

El movimiento o traslado de mineral de un hogar a otro se realiza en forma de zigzag, ya que los hogares pares tienen su descarga por la periferia y los hogares impares por el centro.

Los gases que salen del horno arrastran consigo partículas muy finas de mineral, las cuales se hacen necesario recuperar para evitar mayores pérdidas en el proceso, para ello, se cuenta con un sistema de limpieza de gases formado por un grupo de 6 ciclones en cada horno, 12 electrofiltros y uno adicional en la sección de los silos para la purificación del aire del transporte neumático y del aire de succión de las tolvas.

De los hornos el mineral recuperado en la batería de ciclones cae por gravedad al horno a través del hogar H0. El mineral que se recupera en los electrofiltros es transportado por unos transportadores de paleta hacia una tolva, debajo de la cual se encuentran unas bombas neumáticas de 120 ton/h de capacidad, las cuales envían el mineral hacia las tolvas de los molinos para más tardes ser bombeado hacia los silos formando así él reciclo.

El proceso de reducción es eminentemente endotérmico. Por este motivo el horno dispone de 10 cámaras de combustión dispuestas en los hogares 15, 12, 10, 8 y 6 con quemadores de petróleo de alta presión, que son los encargados de producir los gases caliente para el calentamiento del mineral, a la vez que enriquece la atmósfera reductora del horno ya que trabaja con combustión incompleta.

La presión del aire utilizado en las cámaras de combustión es de 14 kPa, facilitado por el uso de ventiladores centrífugos capaces de mantener una operación estable en dicha área cada bloque cuenta con tres de ellos, dos en operación y uno de reserva.

Así, tenemos que el mineral en su descenso es sometido a un perfil ascendente de temperatura que permite un calentamiento gradual, a la vez que penetra en un perfil gaseoso que garantiza un aumento en la concentración de reductores. La atmósfera reductora es controlada mediante la relación CO_2/CO en H-10 que debe ser de 1,2 %. Un perfil típico de temperatura es como sigue:

Hogares	H0	H4	H6	H7	H9	H11	H113	H15
T en °C	350	720	690	695	710	7120	740	760

Tabla 3-1: Distribución Temperaturas en el Horno de Reducción

En estas condiciones, es posible el desarrollo de las reacciones de reducción a pesar del corto tiempo de retención del mineral en el horno (90 min.).

Esta operación se complementa con la introducción de aire en los hogares 4 y 6 (2000 m^3/ h) para la postcombustión de los gases con un cierto contenido de CO y H_2 no consumidos en la reacción química, para evitar un incremento de las concentraciones de los mismos en el sistema de limpieza de gases (precipitador electrostático) al mismo tiempo se aprovecha el calor de la reacción exotérmica liberado en la combustión para facilitar el perfil de temperatura ideal en el horno.

$$CO + \tfrac{1}{2}\, O2 \rightarrow CO_2 + 67.6 \text{ M kcal/ kmol}$$

$$H_2 + \tfrac{1}{2}\, O2 \rightarrow H_2O + 57.8 \text{ M kcal/ kmol}$$

Después de reducido el óxido de níquel y los óxidos superiores de hierro a Ni metálico e Fe metálico, el mineral es descargado del horno al transportador rotatorio, a su vez, este entrega el mineral a una temperatura de 650 -–700 °C a un enfriador, el cual tiene el objetivo de enfriar el mineral hasta una temperatura por debajo de los 200 °C. Para ello su interior en forma de unas paletas y los carros raspadores, se encargan de remover el mineral y raspar la superficie interior del enfriador evitando así que el mineral se pegue e interfiera en el proceso de transferencia o intercambio de calor. El enfriador rota sobre una piscina de agua. El agua de la piscina debe salir a una temperatura no menor de 70 °C .

El mineral que sale del enfriador cae a una de las canales de Lixiviación, por donde se introduce una corriente de una solución carbonato amoniacal formando una pulpa que va a hacia unos tanques de contacto, luego es bombeada a la planta de lixiviación y lavado a través de otro grupo de bombas para someterse a la próxima etapa del proceso industrial.

3.3 Particularidades del Proceso de Reducción de Mineral:

El proceso de reducción [31] está encaminado a lograr una selectividad tal que permita máximas extracciones de níquel y mínima de hierro teniendo en cuenta que la fracción magnética es la que más favorece la lixiviación amoniacal.

Este proceso bastante complejo, es heterogéneo y ocurre fundamentalmente entre la línea divisoria entre las fases sólida y gaseosa, y la velocidad de reacción va a estar determinada principalmente por la difusión del gas al interior de las partículas, por la concentración de reductores y la velocidad con que se han desalojado los productos de la reacción.

El mineral que se procesa, denominado laterítico y serpentínico tiene su diferencia en cuanto a su composición química, por tal motivo el tratamiento que se le da durante la operación de reducción también tiene sus diferencias.

En la figura 3-2 se muestra el perfil litológico del mineral utilizado en el Proceso de Reducción de Minera. El mineral laterítico caracterizado por su mayor contenido de hierro está formado por una serie de oxidos de hierro como son: Hematita: Fe203, Goetita: Fe203H20, Magnetita: Fe0Fe203, Limonita: Fe203 3H20, Cromita: Fe0Cr03.

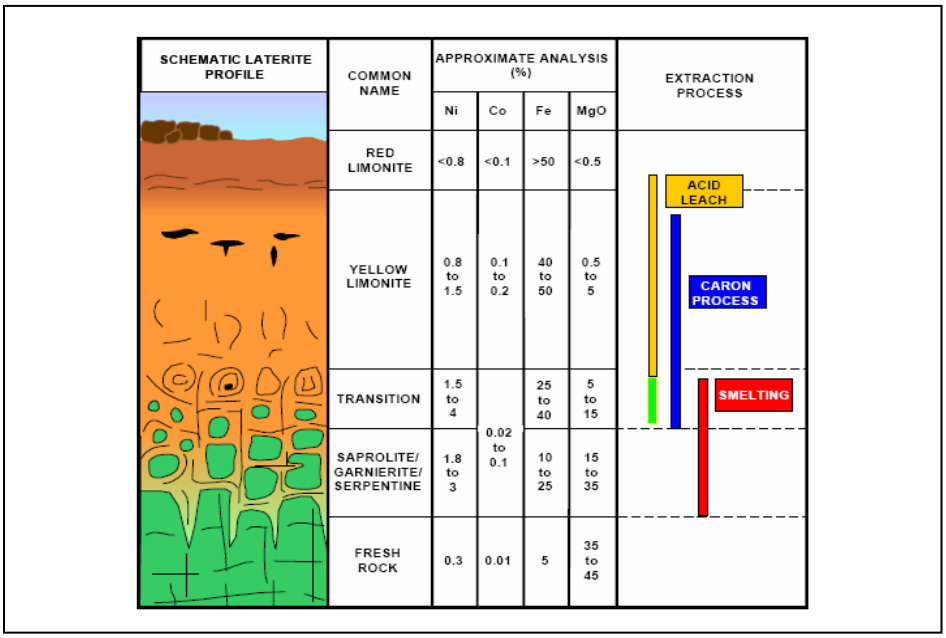


Figura 3-2: Distribución del Níquel en la naturaleza

En estos minerales se encuentra presente el níquel y su reducción es bastante sencilla, una excelente reducción se puede lograr a 480 °C aproximadamente. Además con estos minerales se alcanza una alta fineza, lo cual ayuda positivamente al proceso al proporcionar una mayor superficie activa del mineral.

Sin embargo, el mineral serpentínico está formado principalmente por silicatos hidratados, como son:

1. Gentita: 2NiO2MgO3SiO26H2O,
2. Garnierita: (NiMg)OSiO2H2O,
3. Nepouita: 3(NiMg)O2SiO22H2O

Las investigaciones han demostrado que las extracciones a partir del mineral laterítico son mayores que las que pueden ser obtenidas con las serpentinas, pero la recuperación que pueden brindarnos la mezcla de serpentina y laterita es superior a la obtenida con serpentina sola, siendo la misma de un rango tecnológicamente aceptable.

Se observa que la laterita se reduce hasta un 90%, sin embargo, la serpentina no pasa de un 75%. Se observa también que la mezcla de ambos da una buena extracción, por lo que en la práctica se trabaja con una relación 3:1, o sea tres partes de laterita por una de serpentina. Bajo esta combinación se pueden obtener extracciones de hasta un 84%.

El calentamiento con mineral serpentínico es de suma importancia, ya que se ha verificado una correlación entre la velocidad de calentamiento durante la reducción y la liberación del agua combinada con el mineral, lo cual explica satisfactoriamente la razón del por qué cuando la velocidad del calentamiento es alta la recuperación es pobre.

Las moléculas de los silicatos complejos deben ir perdiendo el agua paulatinamente hasta el momento de su liberación total, que es cuando puede ser reducido el Ni0 presente en ellos. Si en este momento la concentración de reductores no es suficiente, los óxidos componentes se reagrupan nuevamente formando silicatos esta vez deshidratados que requieren condiciones más severas de temperaturas que en nuestro caso no son posibles de efectuar para su reducción.

En el proceso de Reducción de Mineral deben tenerse en cuenta las siguientes reacciones químicas:

1. $NiO(s) + H_2 (g) \sim Ni(s) + H_2O(g)$
2. $NiO(s) + CO (g) \sim Ni(s) + CO_2 (g)$
3. $3Fe_20_3 (s) + H_2 \sim 2Fe_30_4 + H_20 (g)$
4. $3Fe_20_3 (s) + CO (g) \sim 2Fe_30_4 + CO_2(g)$
5. $3Fe_20_3 (s) + H_2 (g) \sim 2Fe0 (s) + H_20 (g)$
6. $3Fe_20_3 (s) + CO (g) \sim 2Fe0 (s) + CO_2 (g)$
7. $Fe0 (s) + H_2 (g) \sim Fe (s) + H_20 (g)$
8. $Fe0 (s) + CO (g) \sim Fe (s) + CO_2 (g)$

Se ha confirmado experimentalmente que las reacciones indicadas arriba son reversibles, el curso de la reacción bien hacia la derecha o hacia a la izquierda, es gobernado por las concentraciones relativas de las sustancias reaccionantes.

Las últimas reacciones no son convenientes por las pérdidas que ocasionan en el proceso de Lixiviación. El compuesto Fe0 se descompone a bajas temperaturas. De ellas encontramos líneas de equilibrio para la reducción del óxido de níquel y de hierro con H_2 y C0 en función de la temperatura.

Se ha podido determinar que el Ni0 no puede ser reducido a metal a una T=540°C, si $K=H_20/H_2$ excede de un valor numérico de 23. Esto se cumple sin tener en cuenta el volumen de gas que se pone en contacto con el óxido de Níquel. El proceso de la reacción química no es proporcional a las cantidades reales de reactivo puestos en contacto, sino a las cantidades presentes en la unidad de volumen.

La reacción ocurrirá con suficiente velocidad cuando las relaciones $C0_2/C0$ y H_20/H_2 no sean muy grandes. Sí, por ejemplo se tiene que:

Para una temperatura de 700ºC

1. $K = H_20/H_2$ para Ní 0= Ní para hornos industriales = 0,84
2. $K = H_20/H_2$ para $Fe_30_4 = Fe0$ para hornos industriales = 0,84.

La reacción de óxido de níquel a níquel metálico ocurrirá a una mayor velocidad que la reducción del Fe_3O_4 a óxido de hierro bajo las condiciones de temperatura y concentración de gases prevalecientes en los hogares inferiores. Bajo estas condiciones, las pruebas en planta piloto y la operación de la planta metalúrgica de Nicaro así lo han demostrado, ya que las extracciones de Ni son superiores al 80% mientras el Fe sólo se extrae en un 2,3%.

3.4 Factores que Influyen en la Operación del Horno.

1. Temperatura.
2. Granulometría.
3. Composición química de la materia prima
4. Concentración de gases.
5. Número de hogares.
6. Estabilidad en la alimentación.
7. Tiempo de retención.

3.4.1 Influencia de la Temperatura.

Este es un parámetro fundamental en todo el proceso pirometalúrgico, ya que la temperatura facilita el cambio de estado o el debilitamiento de la estructura cristalina.

El perfil de temperatura se mantiene mediante la utilización de quemadores de petróleo que se encuentran en las cámaras de combustión. En éstas se trata de mantener una relación aire-petróleo que garantice la combustión incompleta, a la vez ayuda a enriquecer la atmósfera reductora dentro del horno, además en los hogares 4 y 6 se introduce aire secundario proveniente de las camisas de enfriamiento en los hogares 4 y 6 para quemar el C0 residual.

Debido a esta reacción exotérmica se produce una cantidad de calor adicional que contribuye al calentamiento del mineral y a mantener el perfil térmico del horno.

Durante la operación se debe mantener un perfil de temperatura aumentando de arriba hacia abajo que garantice un calentamiento gradual para que las pérdidas de agua de los silicatos no sean bruscas y no se afecte la extracción del mismo.

3.4.2 Influencia de la Granulometría

Por este ser un proceso heterogéneo la granulometría influye determinantemente en los buenos resultados de reducción. Las reacciones ocurren fundamentalmente en la línea divisoria de las fases y la velocidad de la reacción está determinada por la penetración del gas al interior de la partícula. Si estas son pequeñas, aumentara la superficie activa del mineral, será mayor el contacto entre las fases, lo que incide directamente en la conversión de la reacción.

En la práctica se trabaja con un 83 – 85% de fracción 0,074mm, con lo que se puede lograr extracciones aceptables para este proceso sin que el arrastre de polvo sea incrementado sustancialmente.

3.4.3 Composición de la Materia Prima.

La composición química de la matera prima influye directamente en los resultados de la reducción. Atendiendo a esta se fijan las temperaturas en el horno y el régimen de calentamiento de mineral así por ejemplo, cuando el mineral es serpentinoso o sea que su contenido de hierro es bajo, el régimen de temperatura debe ser más alto, así como la concentración de reductores debe ser elevada para reducir el níquel en el momento del debilitamiento de los cristales por la expulsión de agua cristalina.

Para el mineral laterítico. Las condiciones de reducción pueden ser menos severas ya que estos minerales se reducen a bajas temperaturas.

3.4.4 Concentración de Gases

La reducción se lleva a cabo mediante el contacto de los gases reductores con el mineral. Ambas fases se ponen en contacto en la cama de mineral y las caídas de un hogar a otro.

Al encontrarse el petróleo aditivo en hogares con temperaturas superiores a 350°C, comienza la descomposición del mismo, formándose el CO y H2 una vez reaccionado el carbono activo en la primera etapa de la cadena de reacción química, además la atmósfera reductora es enriquecida por el gas producto de la combustión incompleta en los quemadores de petróleo.

La concentración de los gases influye directamente en la conversión del níquel.

3.4.5 Número de Hogares

Este ha sido un factor de importancia en el desarrollo de los hornos modernos, ya que fue detectado que el número de caídas de un hogar a otro influía directamente en los resultados de la reducción debido a que en el momento de las caídas ocurre mayor contacto entre las fases, y las partículas son bañadas completamente por el gas, calentándolas a las temperaturas indicadas y reduciendo el níquel.

La reducción en la cama de mineral lleva el peso de la reducción alrededor de un 70% de la misma, gracias al petróleo aditivo.

3.4.6 Estabilidad en la Alimentación

La inestabilidad en la alimentación al horno afecta seriamente el perfil de temperaturas, y por ende la operación del mismo, ya que todas las condiciones son fijadas para el tonelaje a procesar, y si en este tiempo al horno se les suministra más mineral que el fijado, la temperatura comenzarán a bajar y si ocurre lo contrario la temperatura aumentan y ambas situaciones no son conveniente para el proceso ni para una buena estabilidad en la operación.

3.4.7 Tiempo de Retención

En este equipo éste es un factor que de acuerdo a los principios teóricos de la tecnología incide en los resultados de la reducción por el grado de terminación de la reacción. Valor fijado a 90 minutos, como lo establece la tecnología.

Se ha intentado modificar este tiempo en varias ocasiones, pero los resultados arrojados no son significativos, y por tanto es un parámatreo que no interviene en la propia operación del proceso.

3.5 Variables medidas, Proceso de Reducción de Mineral:

Básicamente se cuenta con dos tipos de variables para el control de la operación del proceso de reducción de mineral: variables de entrada y variables de salida. De cada una de ellas se lleva el registro histórico, y se miden en varios intervalos de tiempo, fundamentado en su comportamiento e inercia de modificación.

3.5.1 Variables de Entrada:

Ni	%	1.262
Co	%	0.104
Fe	%	39.307
SiO_2	%	9.955
MgO	%	5.91
H_2O	%	3.326
100	%	7.395
200	%	5.593
-200	%	87.012

Tabla 3-2: Valores de Muestras de Entrada

Sus valores se obtienen tomando muestras del proceso cada 4 Horas y realizándoles posteriormente los correspondientes análisis químicos.

3.5.2 Variables de Salida:

Ni Extractable	%	79.179
Co Extractable	%	56.486
Indice Consumo Petróleo	***Kg/ton***	***53.00***

Tabla 3-3 Valores de Muestras de Salida

Sus valores se obtienen tomando muestras del proceso cada 4 Horas y realizándoles posteriormente los correspondientes análisis químicos.

Se excluye la variable indice de consumo de petróleo que se muestrea en tiempo real.

3.5.3 Variables de Operación

Parámetros	Valores	Parámetros	Valores	Unidades
Temp. Hogar H0	350	Temp. Camara 6N	350	°C
Temp. Hogar H4	720	Temp. Camara 8N	720	°C
Temp. Hogar H6	690	Temp. Camara 10N	690	°C
Temp. Hogar H7	695	Temp. Camara 12N	695	°C
Temp. Hogar H9	710	Temp. Camara 15N	710	°C
Temp. Hogar H11	720	Temp. Camara 6S	720	°C
Temp. Hogar H13	740	Temp. Camara 8S	740	°C
Temp. Hogar H5	760	Temp. Camara 10S	760	°C

Tabla 3-4: Valores de Muestras de Operación

3.6 Acceso a datos del proceso de Reducción de Mineral.

El acceso a la información es factor esencial a la hora de garantizar la seguridad y productividad en este tipo de proceso de producción.

Los datos que sirven de base a este trabajo, se obtienen desde dos sistemas de producción que posee la industria para la supervisión y el control de su operación: *Plant2Business-Citect y CheNet*.

3.6.1 Plant2Business-Citect

El software Citect garantiza este acceso en dos posibles niveles:

1. CitectScada, nivel de operaciones *on-line*
2. Plant2Business, nivel de supervisión *on-line y off-line*

Ella es la plataforma servidora para la Gestión de la Información Industrial (Industrial Information Management, IIM).

Esta aplicación recoge, almacena y suministra en tiempo real, datos relevantes de planta para toda la empresa como se muestra en la figura 3-3. Se conecta de forma nativa a los sistemas de control y programas SCADA-Citect y bases de datos de gestión más comunes del mercado.

Tanto el establecimiento de la conexión, como el almacenamiento, la publicación y transferencia de datos entre sistemas, se realizan mediante simples clics de ratón, lo que

permite que incluso personal no técnico pueda fácilmente interconectar sistemas de control de marcas diversas con bases de datos de gestión.

La conexión directa a bases de datos tales como SQLServer u Oracle, significa que los datos de planta pasan directamente a los informes de calidad ya existentes, y se mejora la certeza del cumplimiento con normas internas y legislación (procedimientos operacionales).

La concentración de todos esos los datos de planta en una única base de datos simplifica los sistemas de manejo de informaciones a los cuales se hallan conectados. La capacidad de emitir reportes en tiempo real son utilizados para darle seguimiento a las desviaciones de las operaciones pre-definidas.

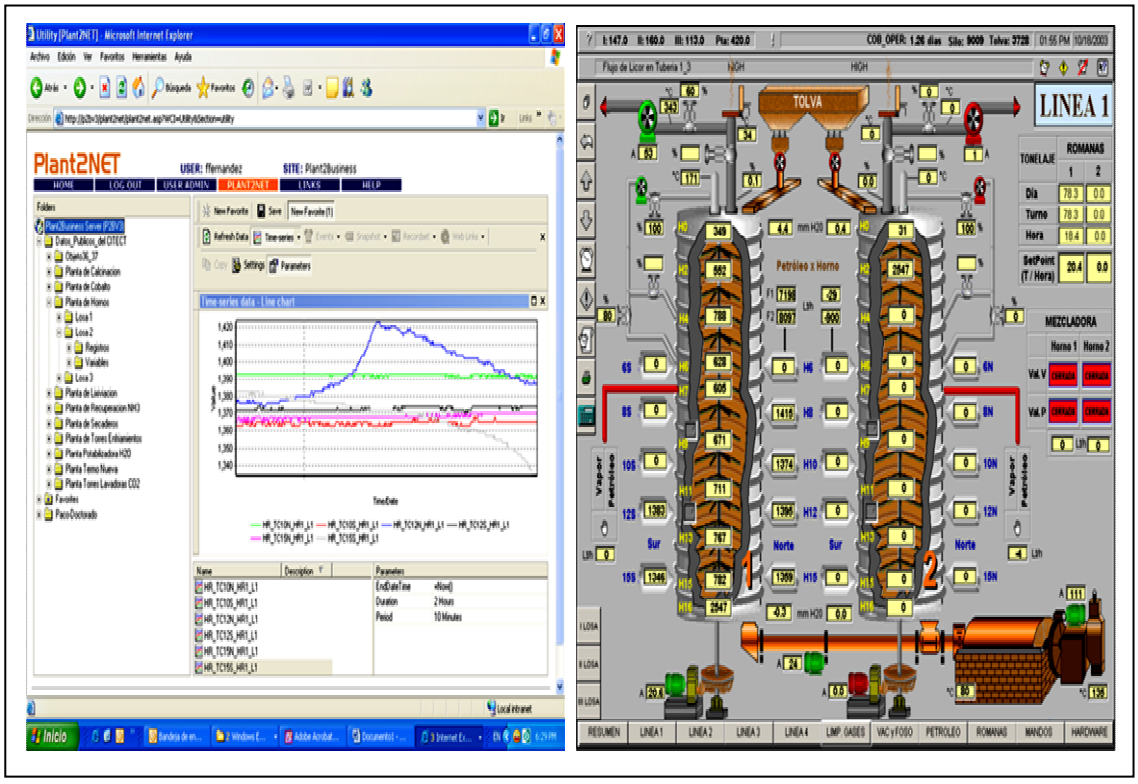


Figura 3-3: Vista de un típico registro de variables medidas en tiempo real

Plant2Business ofrece informaciones como desviaciones estándar, valores máximos y mínimos, que son utilizadas para realizar pequeños análisis del comprtamiento de las operaciones. La totalidad de los datos de los sistemas de control son únicos y se acceden desde cualquier lugar.

Con un módulo de Historial, la integridad y continuidad de los datos está virtualmente garantizada, incluso aunque fallen las redes o los computadores. El Historial posee una resolución de almacenamiento menor a un segundo, y almacena junto a los datos, variables de calidad, status, y estadísticas. Permite almacenar años de información en línea sin comprometer la precisión de los datos ni cargar los computadores.

De acuerdo a nuestros intereses en este trabajo, y como se aprecia en la figura 3-4, se configuraron diferentes grupos de variables de acuerdo a su tipo (temperaturas en cámaras, hogares, tonelaje, etc), ello facilita su exportación a un formato texto de manera que puedan ser tratadas adecuadamente por la herramienta de procesamiento (MatLab 7.0).

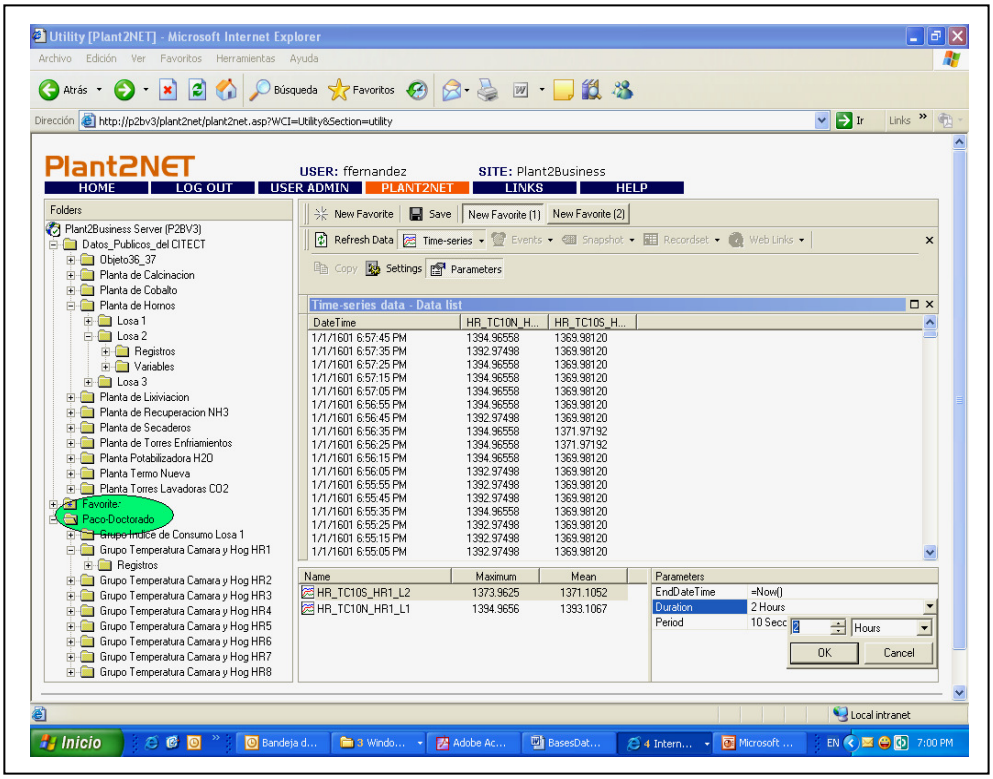


Figura 3-4: Configuración preparada para este trabajo

Otro aspecto que se tiene en cuenta tiene que ver con el tiempo de muestreo; en todos los casos se ajustó a 10 minutos para llegar a una media horaria y luego a una media en el período de duración de un turno de trabajo (8 Horas). Esto tiene que ver con el período de mediciones de los parámetros que se tratan off-line, que es cuando se obtiene el resultado de la variable objeto de valoración.

3.6.2 CheNet

Es la otra plataforma servidora de la Gestión Productiva 8H para la Consolidación de la Información de Producción. Esta aplicación recoge, almacena y suministra en tiempo no real, datos relevantes de todos los análisis químicos off-line a nivel de empresa.

De igual manera se conecta a los sistemas de control y programas SCADA-Citect y bases de datos de gestión más comunes del mercado.

La figura 3-5 muestra un reporte típico de datos que son resultados de esos análisis.

Estas son las variables que determinan la calidad del proceso realizado, y por tanto son tan importantes como las de operación que se registran en tiempo real.

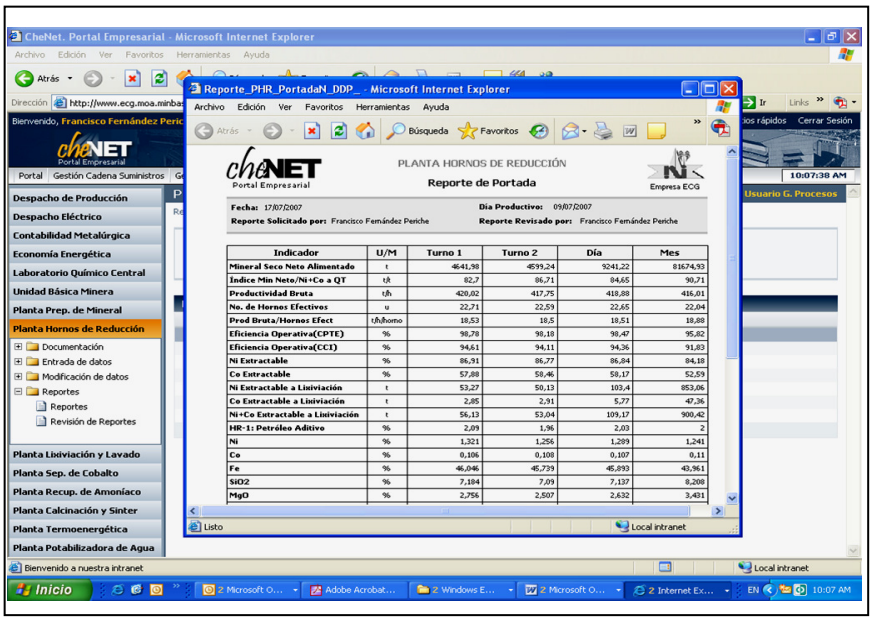


Figura 3-5: Reporte de resultados analíticos del sistema

Para estos casos, el tiempo de muestreo es de 2horas, teniendo que ver en específico con la calidad del mineral que suministra la Mina al proceso industrial (su materia prima fundamental).

3.6.3 Integración de los datos utilizados

Estas dos plataformas están insertadas al sistema de Gestión Empresarial a través de un Portal WEB, que tiene por misión aunar toda la información y conocimientos que aporta la operación diaria de la industria, como lo muestra la figura 3-6.

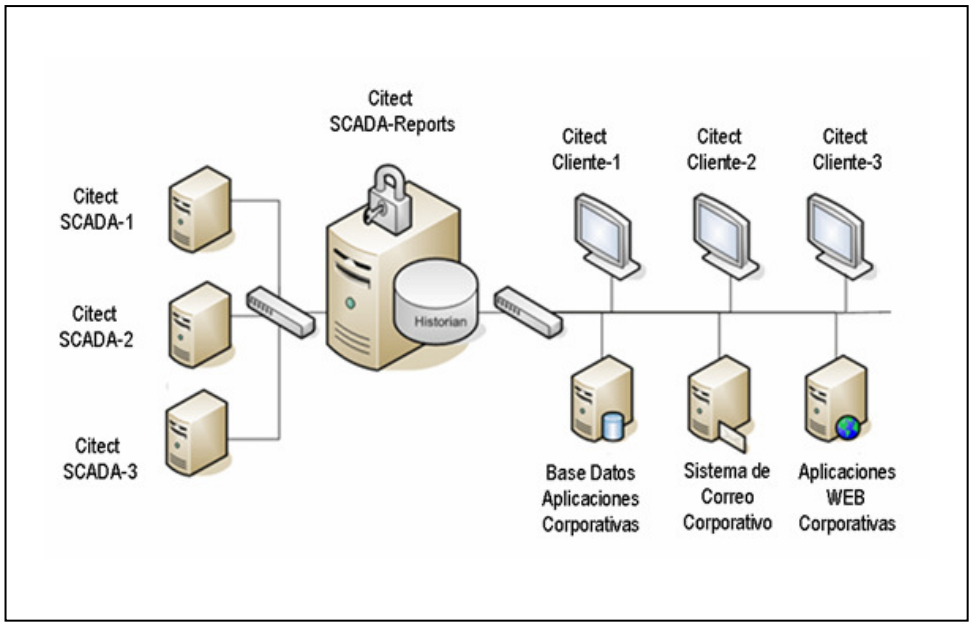


Figura 3-6: Integración de las Plataformas descritas a la Gestión Empresarial actual

Por medio de las mismas, es posible agregar y publicar información en Bases de Datos particulares o distribuidas, implementando para ello servicios de transformación de datos (DTS) encargados de colocar los mismos en las tablas de las cuales provienen las informaciones mostradas en los reportes empresariales que permiten a los ejecutivos fundamentar criterios operativos basados en un sistema de tiempo real.

Los paquetes de Servicios de transformación de datos incluyen una o más tareas DTS. Cada tarea define un elemento de trabajo que se ejecutará como parte del proceso de transferencia y transformación de los datos.

Se tiene acceso a estas tareas desde el Diseñador DTS-Transformar datos.

Se emplea la tarea Transformar datos para copiar datos, asignar una gran variedad de transformaciones a los datos y personalizar las transformaciones con una secuencia de comandos Microsoft ActiveX® desde la base de datos *ChenetCitec* a la Base de datos *ChenetDB* ambas ubicadas en el mismo servidor, dependiendo de la frecuencia de captura de los mismos están destinados a distintas tablas.

La figura 3-7 muestra algunos de los DTS que están implemetados y permiten la disponibilidad en tiempo de los diferentes grupos de datos.

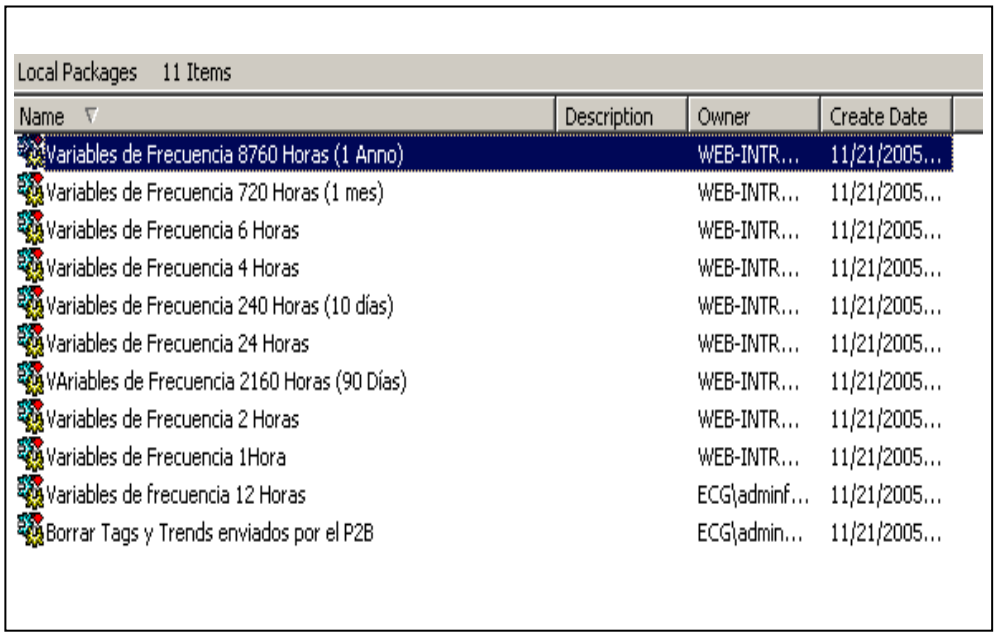


Figura 3-7: DTS implementados

Según la frecuencia de entrada se harán activos los DTS correspondientes, garantizando la incorporación de la información a las tablas de la base de datos empresarial, con lo que se logra un adecuado manejo de la información a la hora de presentar reportes a partir de las mismas o realizar cálculos para la obtención de otros a partir de los indicadores primarios.

3.7 Petróleo en las Cámaras de los Hornos de Reducción.

Uno de los portadores energéticos a que se hace referencia en la parte inicial del documento; y cuya incidencia es significativa en la consecución eficiente del proceso de Reducción de Mineral es *el petróleo tecnológico o fuel oil.*

Este se utiliza con 3 objetivos específicos:

1. Mezcla con el mineral de entrada como agente aditivo.
2. Generar gases reductores mediante combustión incompleta.
3. Garantizar el perfil térmico de los hornos por medio de quemadores de alta presión.

Loa objetivos 2 y 3 se logran por medio de la combustión incompleta que se produce en las diferentes cámaras que poseen los Hornos.

Figura 3-8-: Vista de una Batería de Cámaras de Combustión de los Hornos de Reducción

El suministro de petróleo a las mismas se realiza desde una estación de bombeo. Para garantizar esta alimentación se hallan instaladas un grupo de 8 bombas, de las cuales 6 están destinadas para realizar el bombeo a los hornos y las otras dos para la adición de petróleo en los transportadores de bandas en la Planta de Preparación de Mineral. Además se disponen de 5 calentadores de 49 m^2 de superficie de calentamiento.

El petróleo llega a un colector de succión de las bombas por gravedad, procedente del depósito de petróleo por medio de dos líneas, junto a estas se hallan otras para suministrar vapor con el objetivo de mantener la temperatura del petróleo de acuerdo a los requisitos establecidos.

Para cada línea que suministra combustible, se encuentran instaladas las correspondientes bombas. El petróleo es bombeado a una presión de 10 atm. A la salida de los Intercambiadores de calor la presión de petróleo será de 9.5 atm. La temperatura de salida de 100 a 110 °C.

El combustible que no se consume retorna por otra tubería, la cual llega a la unidad calentadora por la línea de retorno donde se encuentra un platillo orificio que regula flujo y un cheque.

La línea de petróleo sube hasta el piso de las cámaras de H-6, de la cual salen dos tuberías para distribuir el petróleo a los hornos correspondientes.

3.7.1 Indice de Consumo de Petróleo en Cámaras:

Consumo de Petróleo Total, 1 día	*ton*	726
Consumo de Petróleo Cámaras, 1 día	*ton*	468
Consumo de Petróleo Aditivo, 1 día	*ton*	258
Indice de Consumo de Petróleo en Cámara	***Kg/ton***	***53.00***

Tabla 3-5: Valores de Muestra de la variable Indice Consumo en Cámaras

3.7.2 Características del Petróleo utilizado en Cámaras:

Densidad Trabajo	*g/cm3*	0.9605
Densidad a 15°C	*g/cm3*	0.9943
Valor Calórico	*Kcal/kg*	9863.6
Azufre	%	2.45
Carbon	%	12.18
Asfaltenos	%	6.54

Tabla 3-6: Valores de Muestras de Entrada

3.7.3 Estructura del Consumo de petróleo en las Cámaras

La estructura del consumo de petróleo está definida por la organización que tiene la Planta de Hornos de Redución. Cuenta con 24 Hornos, agrupados desde el punto de operacional en grupos de 8, denominado LOSA, a su vez cada grupo de estos está formado por parejas, denominadas LINEA. Cada horno cuenta con 10 cámaras situadas en lugares determinados por la propia tecnología CARON: 5 para la zona norte y 5 para la zona sur, formando un espejo.

Losa 1	Losa 2	Losa 3
156 ton al día	156 ton al día	156 ton al día
56 940 ton al año	56 940 ton al año	56 940 ton al año
53 kg/ton	53 kg/ton	53 kg/ton
	total anual	

	170 820 ton	

Tabla 3-7: Estructura de Consumo de Petróleo en Cámaras

3.8 Planteamiento del Problema

Como se ha expresado en el resumen de este trabajo, el conocimiento a priori o la predicción de cómo evolucionará una variable en el futuro, además de constituir una información muy valiosa, permite disponer de tiempo suficiente para adecuar una respuesta de forma óptima en la detección de oportunidades, prevención de problemas y optimización de recursos entre otros aspectos.

La predicción como una actividad intelectual humana, puede definirse como un juicio sobre el futuro, el cual está basado en las experiencias personales pasadas, *la información histórica disponible* y los mecanismos de razonamiento que utiliza la mente humana para emitir sus juicios [24]. Ya sea realizada a través del juicio informado o mediante el uso de técnicas estadísticas, la predicción implica el uso de un modelo mental o matemático que representa el conocimiento que el modelador/pronosticador ha adquirido sobre el funcionamiento de un sistema real (que puede representar un mercado o una industria) a través de su experiencia, para determinar cuál es la evolución más probable en un determinado momento.

No obstante, dicho comportamiento es dictaminado por diversos factores que se relacionan de forma compleja [24] haciendo que el modelo usado para la predicción sólo pueda ser construido a través de un proceso de agregaciones, simplificaciones y conjeturas sobre la dinámica que gobierna el sistema real.

A partir de los datos históricos almacenados y utilizando técnicas de Softcomputing, es posible elaborar modelos con una capacidad razonable de estimación, con los cuales los administradores tendrían manera de predecir los valores de eficiencia de ciertos procesos industriales, impidiéndoles conducir a falsos optimismos y a inevitables retrasos en los planes de producción.

La definición formal del problema a resolver es: dada una serie de observaciones;

$$\left\{\left(\overrightarrow{x_k ; y_k}\right); k = 1,...n \right\} \qquad 3\text{-}1$$

con su correspondiente salida

$$y_k = F\left(\vec{x}\right) \in \mathbb{R} \text{ y } \vec{x}_k \in \mathbb{R}^d \qquad 3\text{-}2$$

se desea obtener una función f tal que:

$$y_k \simeq f\left(\vec{x}_k\right) \qquad 3\text{-}3$$

En la definición de predicción, existe una mención explícita a la dimensión temporal, sin embargo, debido a la abstracción de la definición matemática, no se identifica al tiempo como una variable en concreto.

En realidad se podría seguir considerando la palabra predecir puesto que para el modelo que proporciona las salidas, una vez creado, cada entrada nueva es temporalmente posterior a su creación y, consecuentemente, futura en el tiempo.

Ante estos elementos, se utilizará el término *aproximación funcional* para hacer referencia al problema que se desea resolver: dada una hipotética función (que asigna una salida a un vector de entrada) se desea tener un modelo que sea capaz de aproximar a la función hipotética, es decir, que para el mismo conjunto de entradas con sus salidas conocidas, sea capaz de generar aproximadamente las mismas salidas.

En este caso, las relaciones que se pretenden descubrir entre las diferentes variables de entrada/salida que se almacenan constantemente en los diferentes soportes informáticos de la industria, se pueden asociar al problema de aproximación de una función desconocida.

Así pues, se tendría un espacio de búsqueda donde cada uno de sus puntos representa una solución (ó RBF) al problema de aproximación.

El principio de valorar la evolución futura de una variable de este tipo, que a su vez depende del comportamiento de otras (composición del mineral oxidado de entrada, perfiles térmicos y gaseosos) que generan cambios en las entradas de la planta, y aportan un cierto grado de incertidumbre al problema, puede basarse en el principio de que es necesario alcanzar un equilibrio entre el beneficio inmediato y el comportamiento óptimo del sistema a lo largo del tiempo [24].

Con todo lo revisado hasta aquí, y en relación a los principios de funcionamiento del proceso de Reducción de Mineral y sus variables más importantes, el mismo queda caracterizado a partir de las siguientes funciones:

$$\%\ \text{Extracciones} = f1\ (\text{Mineral, Perfil Temperatura Hogares, Agentes Reductores}) \qquad 3\text{-}4$$

$$\text{Perfil Temperatura Hogares} = f2\ (\text{Mineral, Temperatura Cámaras}) \qquad 3\text{-}5$$

$$\text{Agentes Reductores} = f3\ (\text{Mineral, Petróleo Aditivo, Petróleo en Cámaras}) \qquad 3\text{-}6$$

$$\textbf{Indice Consumo Petróleo} = \textbf{f4 (Mineral, Temperatura Hogares y Cámaras, Agentes Reductores)} \qquad 3\text{-}7$$

La expresión 3-7, encierra en sí misma el planteamiento del problema que este trabajo se ha propuesto resolver: *predecir el consumo de petróleo en cámara*, y que como ha quedado reflejado, puede considerarse como un problema de aproximación funcional en el que se obtiene una descripción de la relación entre las entradas y salidas a partir de las parejas de entrada/salida conocidas mediante un proceso de ajuste o aprendizaje.

Una vez obtenida dicha función, es posible utilizarla para determinar relaciones de entrada/salida en zonas del espacio de entrada a partir de entradas que no se encuentren en el conjunto de entrenamiento.

3.9 Conclusiones

En este capítulo se ha expuesto un resumen de las características más importantes del proceso de Reducción de Mineral de Punta Gorda, donde los elementos a los cuales hemos hecho referencia, han dado motivos para considerar la optimización de portadores energéticos como un de los temas de mayor relevancia en este tipo de proceso industrial.

De igual manera ha quedado definido el planteamiento del problema que nos hemos propuesto resolver en el presente trabajo.

En el próximo capítulo de expone el procedimiento utilizado como solución al problema que ha quedado planteado.

Capítulo 4

Procedimiento para la caracterización del proceso de reducción de mineral

Este capítulo presenta los componentes de la propuesta de solución que se ha concebido para dar respuesta al problema que quedó planteado en el Capítulo 3.

Básicamente se refiere a la aplicación de varios modelos de computación bio-inspirados, los cuales se basan en emplear analogías con sistemas naturales o sociales para diseñar métodos heurísticos no determinísticos de búsqueda, aprendizaje, de imitación de comportamiento, etc.

El procedimiento que hemos desarrollado implica la hibridación e integración de diversas técnicas y paradigmas de programación tales como, algoritmos de búsqueda local, algoritmos evolutivos multiobjetivo y redes neuronales de funciones de base radial.

La razón para integrar las diversas técnicas mencionadas se deriva de la posibilidad de poder combinar las ventajas que posee cada una, supliendo sus desventajas con las aportaciones de las demás. Por ejemplo, dado que la búsqueda local puede dejar sin explorar zonas del espacio de soluciones debido a su carácter local, los algoritmos evolutivos permitirán realizar esa tarea, supliendo la posible falta de precisión en el ajuste de las soluciones de éstos mediante la búsqueda local.

La evolución multiobjetivo permite al algoritmo mantener un compromiso entre la complejidad de las redes y la calidad de las soluciones a costa de un coste computacional mayor.

4.1 Introducción

Mediante el procedimiento que hemos desarrollado, pretendemos explotar la capacidad de las redes de funciones de base radial como instrumento de *modelización y predicción*, poniendo de relieve la superioridad mostrada por las mismas en el modelado de los fenómenos no-lineales. Para ello confeccionaremos un modelo, implementado sobre Matlab, que nos permitirá predecir la evolución de la variable *índice de consumo de petróleo*, descrita en el capítulo anterior.

Debido al tamaño de las bases de datos, a la presencia de ruido, datos inconsistentes, redundantes, etc., se hace necesario aplicar técnicas de selección de variables sobre ellos, de manera que se generen valores representativos y mejoren las prestaciones de los algoritmos implementados. Los módulos de *Selección de Variables* y *Aproximación Funcional con redes de funciones de base radial* de la figura 4-1 permiten validar el planteamiento del problema definido.

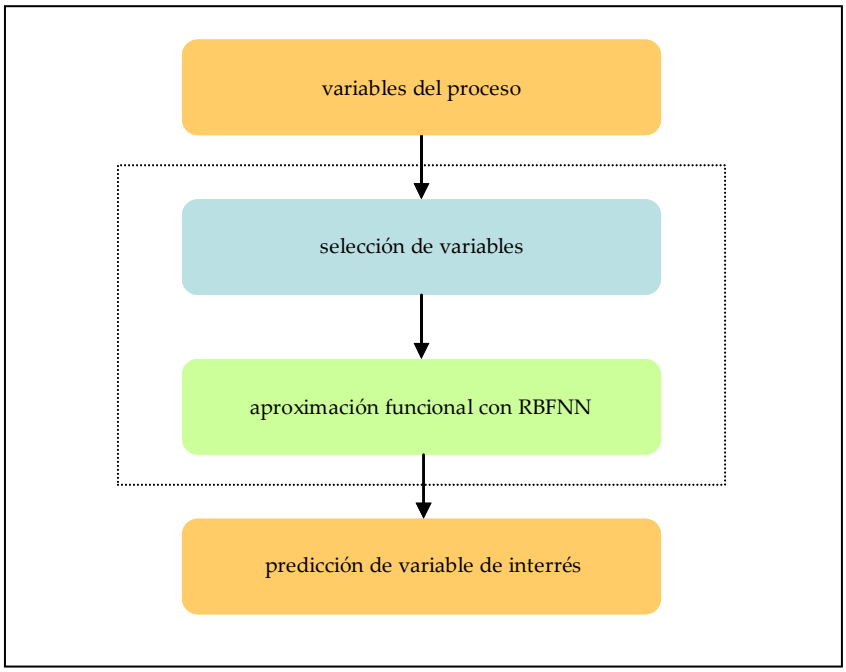


Figura 4-1: Etapas del procedimiento propuesto.

4.2 Proceso de selección de variables:

Cuando se utiliza una gran cantidad de variables en un proceso de aprendizaje, se corre el riesgo de que muchas de ellas no aporten la suficiente información (con respecto a las demás) para poder generar un modelo suficientemente preciso de un espacio de entrada tan grande. En estos casos la mayoría de los algoritmos de aprendizaje adolecen de pérdida de rendimiento y dificultad de comprensión en los resultados, además de altos tiempos de ejecución.

4.2.1 Reducción de la dimensión:

Existen varias aproximaciones para realizar una reducción de dimensión, entre las más uitilizadas de encuentran: la extracción de variables *(feature extraction, FE)* y la selección de variables *(feature subset selection, FSS)*. La extracción de variables consiste en encontrar una nueva descripción del dominio de variables, de otra parte, la selección de variables trata de seleccionar un subconjunto de variables sin hacer ninguna transformación. [81]. En general, en estos procedimientos de selección se distinguen cuatro etapas esenciales:

1. *Selección:* en esta etapa se determina el posible subconjuto de características para realizar la representación del problema
2. *Evaluación:* en esta etapa se evalúa el subconjunto de características escogidas en el punto anterior.
3. *Detención:* se chequea si el subconjunto seleccionado satisface el criterio de detención de la búsqueda.
4. *Validación:* esta etapa se utiliza para verificar la calidad del subconjunto de características que se determinaron.

La figura 4-2 muestra cómo es la relación temporal entre estas etapas del proceso. [25]:

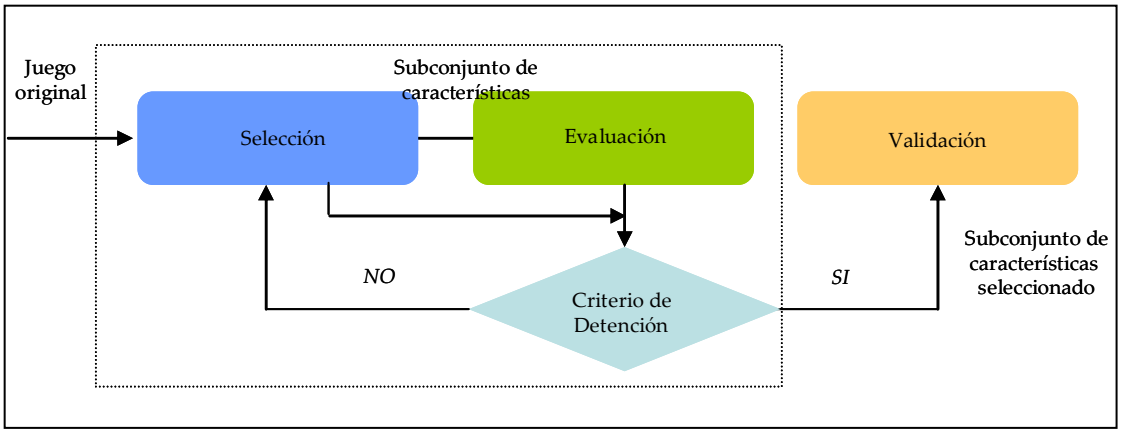


Figura 4-2: Procedimiento general de selección de características.

En otro sentido, los métodos de selección de características se clasifican desde el punto de vista de la manera en que se determina el nuevo subconjunto a evaluar, lo que conduce a tres clases de métodos [25].

1. *Métodos Completos:* Estos métodos examinan todas las posibles combinaciones de características. Son muy costosos computacionalmente dado que el espacio de búsqueda es del orden O(2^n) para n características; pero encuentran de forma segura el subconjunto óptimo de características.

2. *Métodos Heurísticos:* Utilizan una metodología de búsqueda de forma tal que hace innecesario evaluar todos los subconjuntos de características. Ello conlleva a una mayor velocidad en la determinación del conjunto de características, ya que el espacio de búsqueda es menor que en los métodos anteriores. No obstante estos métodos no aseguran la obtención del mejor sub-conjunto.

3. *Métodos Aleatorios:* Son aquellos métodos que no tienen una forma especifica de definir el subconjunto de características a analizar, sino que utilizan metodologías aleatorias. Dan lugar a una búsqueda probabilística en el espacio de características, cuyo resultado dependerá del número de intentos. Por lo tanto tampoco aseguran la obtención del conjunto de características óptimo.

Desde el punto de vista de la función de evaluación, los procedimientos de selección de características se pueden clasificar en dos categorías [74].

1. *Métodos de filtraje:* Estos son métodos donde el procedimiento de selección es realizado en forma independiente a la función de evaluación (clasificación). Se pueden distinguir cuatro diferentes medidas: distancia, información, dependencia y consistencia.

2. *Métodos dependientes (wrapped):* En estos métodos el algoritmo de selección utiliza como medida la tasa de error del clasificador. Se obtienen generalmente mejores resultados que en el caso anterior, pero trae consigo un costo computacional mucho mayor.

4.2.2 Variables obtenidas del proceso de Reducción de Mineral:

En la figura 4-3 se muestra el conjunto de variables que se miden en el actual proceso de Reducción de Mineral; las mismas se hallan agrupadas como variables de entrada *(27)* y de salida *(1)* respectivamente.

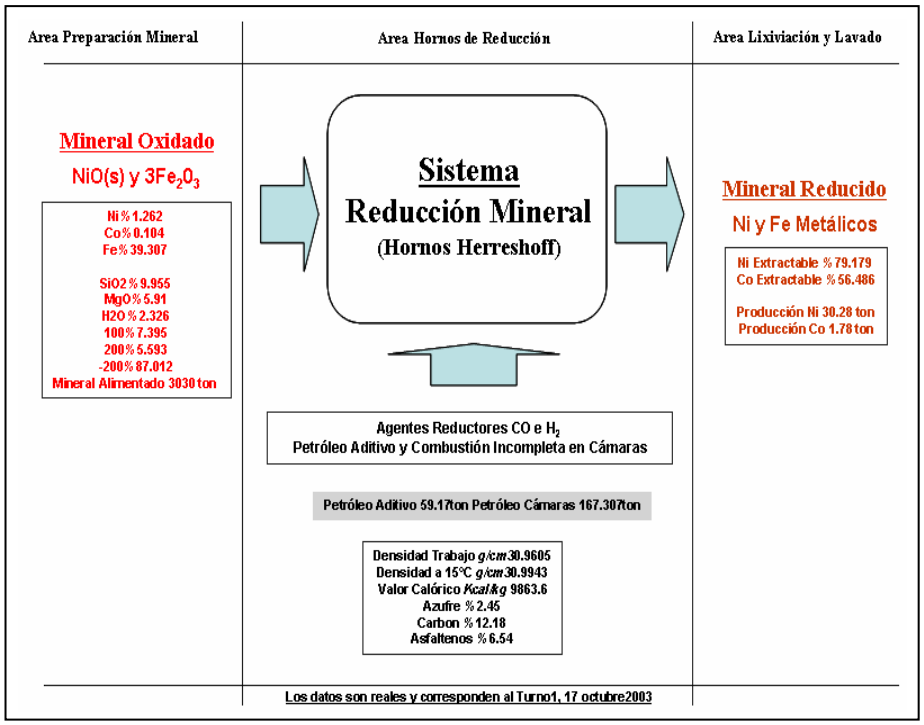


Figura 4-3: Distribución de variables en el proceso de Reducción de Mineral.

En nuestro modelo se han considerado como variables iniciales la totalidad de las que se tienen en cuenta para la operación de esta parte del proceso industrial. Son todas aquellas que se monitorean constantemente por los operadores e ingenieros de proceso, de acuerdo a lo que establecen los principios de diseño de la tecnología CARON.

La tabla 4-1 hace referencia a la leyenda que le hemos dado a las variables utilizadas.

1	**2**	**3**	**4**	**5**	**6**	**7**
Ni	Co	Fe	Si	Mg	CO/CO_2	Ton
8	**9**	**10**	**11**	**12**	**13**	**14**
A_Petr	TH0	TH2	TH4	TH6	TH7	TH9
15	**16**	**17**	**18**	**19**	**20**	**21**
TH11	TH13	TH14	TC6N	TC6S	TC8N	TC8S
22	**23**	**24**	**25**	**26**	**27**	**28**
TC10N	TC10S	TC12N	TC12S	TC15N	TC15S	***I_Petr***

Tabla 4-1: Leyenda de las variables utilizadas

Dentro de ese conjunto de variables, las figuras 4-4 y 4-5 muestran el comportamiento de un grupo de las que están consideradas como de gran importancia de acuerdo a la tecnología de producción de níquel.

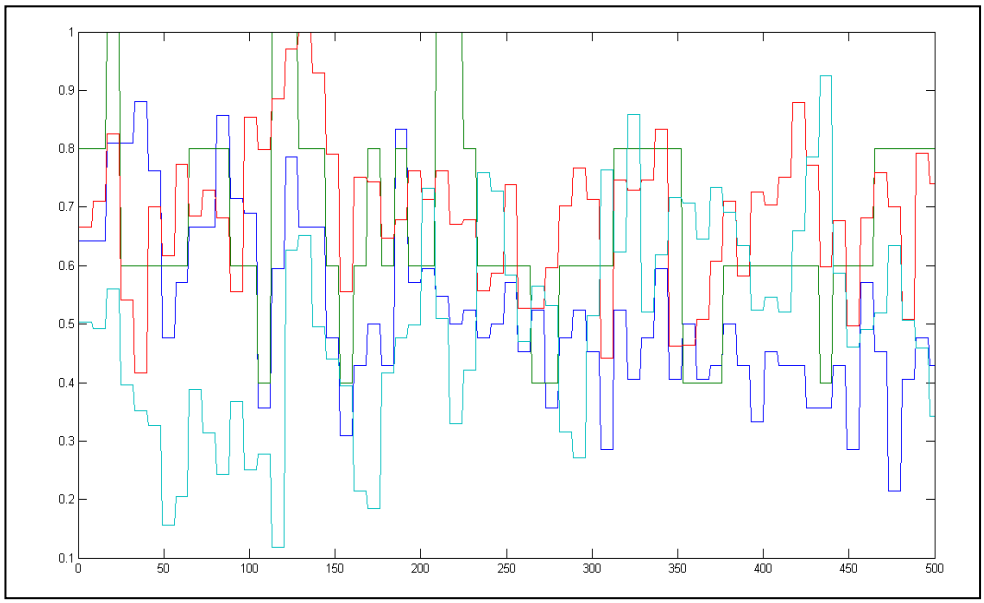


Figura 4-4: Comportamiento de las variables Ni , Co, Fe, Indice Consumo Petróleo.

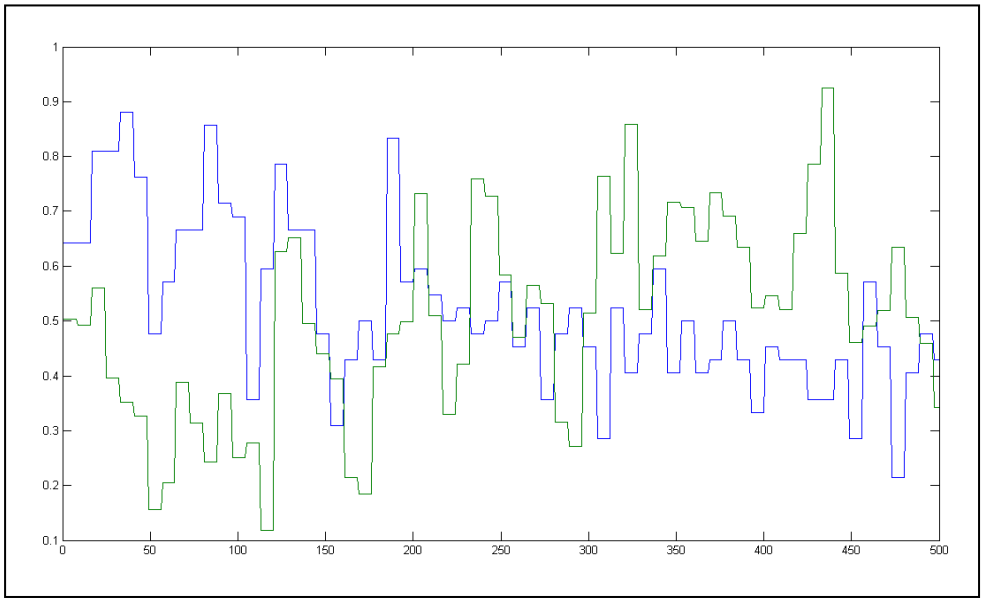


Figura 4-5: Comportamiento de las variables Ni (verde), Indice Consumo Petróleo (azul).

4.2.3 Test Delta para la selección de variables:

El problema de la estimación de la varianza residual [34] es un tema muy conocido en inteligencia artificial y estadística en diferentes ámbitos de aplicación. La estimación de la varianza residual se puede contemplar como la estimación de la varianza de la parte de la salida que no puede ser modelada con el juego de las variables de entrada. Este tipo de información se puede evaluar y proporciona métodos adecuados de selección.

El Test Delta es una de las formas más simples para resolver el problema de la estimación de la varianza residual. Su principal ventaja es su robustez e intuitividad, lo que hacen de el una herramiento ideal para ser aplicada en problemas de pocas dimensiones.

La idea del Test Delta es que entradas similares que pertenezcan a un espacio de entrada dado, tienden a producir salidas similares, estando causada la diferencia por flutuaciones estáticas en la salida. En términos matemáticos se expresa como:

$$\gamma_{M,k} = \frac{1}{2M}\sum_{i=1}^{M}(Y_i - Y_{N[i,k]})^2. \qquad 4\text{-}1$$

Tras la aplicación de este algoritmo a nuestro problema, se escogieron varios escenarios para evalauar la selección de variables: ***27, 8 y 7,*** serán las dimensiones sujetas a experimentación. La figura 4-6 ilustra el comportamiento de la totalidad de ellas una vez aplicado el algoritmo, donde las que se hallan entre los límites 0-10 manifiestan la mayor relevancia.

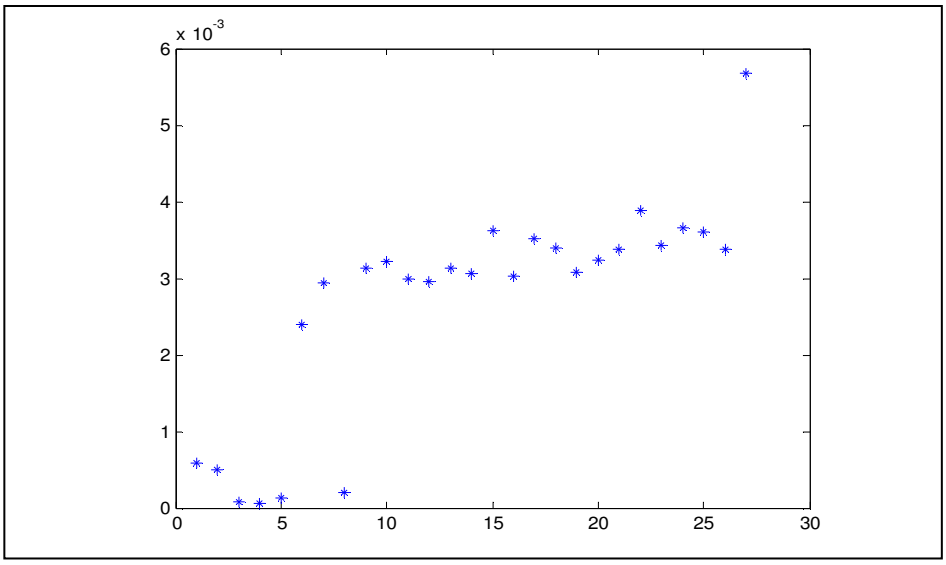


Figura 4-6: Resultados del Test Delta.

4.3 Aproximación Funcional con Algoritmos Multiobjetivo

En lo adelante, se va a abordar un problema de aproximación funcional desde el punto de vista de la optimización multiobjetivo.

4.3.1 Introducción

En muchas de las búsquedas de soluciones, suele ser necesario satisfacer de manera simultánea varios objetivos, los cuales pueden ser contradictorios. A veces existe la posibilidad de combinar todos los objetivos en uno solo, pero lo usual es que no se conozca de qué manera unificar los diferentes objetivos, o bien no es adecuado o imposible. En estas circunstancias, nos enfrentamos a un Problema de Optimización Multiobjetivo.

La optimización multiobjetivo (denominada también optimización multicriterio, optimización multiprestaciones u optimización vectorial) aborda el problema de encontrar un vector de variables de decisión que satisfagan unas restricciones y optimicen una función vectorial cuyos elementos representan a la función objetivo. Estas funciones proporcionan una descripción matemática de criterios de evaluación que generalmente están en conflicto unos con otros. Por tanto, el término "optimizar" implica encontrar el conjunto de valores de todas las funciones objetivo que son aceptables para el diseñador.

Cualquier algoritmo de optimización multiobjetivo debe hallar todos los vectores de decisión para los cuales los correspondientes vectores objetivos no se puedan mejorar en una dimensión sin degradar otra. A ello se denomina el conjunto de Pareto Optimo (frente de Pareto).

Según se indica en [21], aunque la capacidad de los Algoritmos Evolutivos (AE) para resolver problemas con múltiples objetivos fue sugerida por Rosenberg en la década de 1960, hasta mediados de la década de 1980 no se presentó la implementación de un algoritmo evolutivo para optimización multiobjetivo [125]. A partir de la década de 1990 aparecieron una gran cantidad de propuestas de MOAE, organizándose una comunidad de investigadores en el área que trabaja activamente en la actualidad.

Los AE tienen la potencialidad de tratar problemas con objetivos múltiples, y son capaces de hallar en cada ejecución un conjunto de soluciones aproximadas al frente de Pareto. Esto representa una importante ventaja respecto a los algoritmos tradicionales, que solamente generan una solución por ejecución.

Complementariamente, los AE tienen otras ventajas respecto a los algoritmos tradicionales, como ser menos sensibles a la forma o a la continuidad del frente de Pareto o permitir abordar problemas con espacio de soluciones de gran dimensión.

Un MOAE debe diseñarse para lograr dos propósitos en forma simultánea: lograr buenas aproximaciones al frente de Pareto y mantener la diversidad de las soluciones, muestreando adecuadamente el espacio de soluciones sin converger a una solución única o a una sección acotada del frente. La figura 4-7 presenta gráficamente los propósitos de un MOEA, donde la región celeste corresponde al espacio de búsqueda de funciones objetivo de un problema hipotético, mientras que la línea roja gruesa representa al frente de Pareto. El mecanismo evolutivo de los AE permite alcanzar el primer propósito, mientras que para preservar la diversidad los MOAE utilizan las técnicas de mantenimiento de la diversidad como las técnicas de *nichos*, *sharing*, *crowding* o similares, utilizadas tradicionalmente por los AE en la optimización de funciones multimodales.

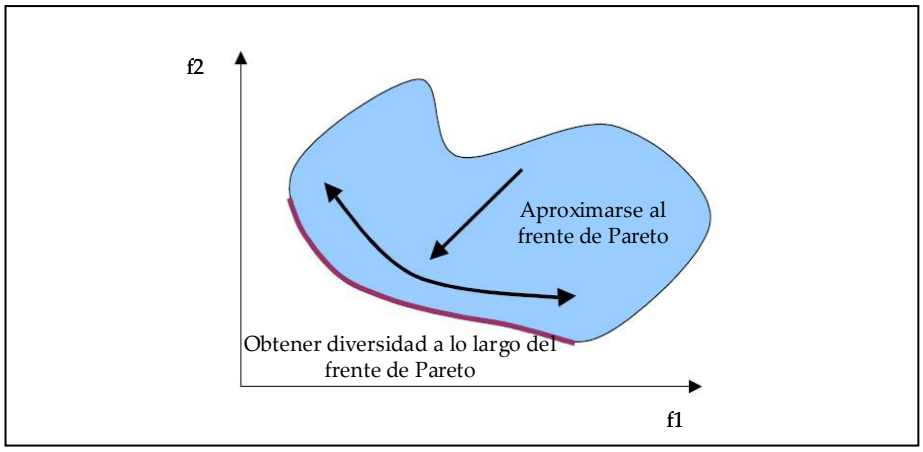


Figura 4-7: Criterios de evaluación de un Algoritmo Evolutivo para Optimización Multiobjetivo.

4.3.2 Algoritmo NSGA-II

Uno de los aspectos que ha influido notablemente en el desarrollo de los algoritmos genéticos es la aparición de problemas multiobjetivo y su posible solución mediante modificaciones de los algoritmos genéticos clásicos. También se puede plantear que un problema multiobjetivo es aquel en el que existen soluciones que no se pueden considerar ni mejores ni peores que otras ya que dichas soluciones encuentran compromisos distintos pero igualmente válidos entre los objetivos a satisfacer.

Formalmente, la solución a un problema multiobjetivo no consiste en un único valor sino en un vector $f : D \to \mathbb{R}^{N_{obj}}$ dentro del espacio de soluciones *D* con N_{obj} funciones a optimizar.

Dentro de los algoritmos evolutivos multiobjetivo, en [118] se propuso el algoritmo denominado *Non-dominated Sorting Genetic Algorithm II (NSGAII)*, que es considerado como uno de los mejores algoritmos de este tipo. Debido a esa circunstancia, este ha sido el algoritmo tomado como base y a partir del cual se añadirán los distintos elementos que componen el nuevo algoritmo propuesto para diseñar RBFNN.

Otros factores que han contribuido a tomar esta decisión son:

1. Su política de reemplazo de individuos considera el elitismo de un modo que se puede adaptar fácilmente a la hora de integrar individuos que migran desde otras islas.

2. Su eficiciencia a la hora de generar la siguiente población

3. Su sencillez y claridad a nivel de diseño.

A continuación se mostrarán los aspectos esenciales del funcionamiento del algoritmo NSGAII y luego se describirán los elementos que constituyen el algoritmo propuesto para el diseño de RBFNN, que proponemos para resolver el problema que nos planteamos en el ámbito del proceso de producción de níquel.

Este algoritmo es una versión adaptada a nuestros propósitos del algoritmo *Non-dominated Sorting Genetic Algorithm (NSGA)* propuesto en [25]. El aspecto que mejora la segunda versión del NSGA es el procedimiento mediante el cual se van obteniendo poblaciones de soluciones no dominadas, haciéndolo más eficiente. Para poder comprender esto, es necesario interpretar adecuadamente los conceptos de dominancia y frente de Pareto.

En los problemas de optimización multiobjetivo existen soluciones $t_j \in D / f\left(t_j\right) = \left(f_1\left(t_j\right), f_2\left(t_j\right), \ldots f_{N_{obj}}\left(t_j\right)\right)$ que no pueden ser ordenadas utilizando las relaciones clásicas que se establecen en los problemas con un único objetivo f, donde se tiene que:

$$f(t_1) \leq f(t_2) \quad \text{ó} \quad f(t_2) \leq f(t_1) \qquad 4\text{-}2$$

Al ser cada solución un vector, las anteriores comparaciones se establecen así:

$$\begin{aligned} &f(t_1) = f(t_2) \Leftrightarrow f_i(t_1) = f_i(t_2) \quad \forall i \in 1,2,...N_{obj} \\ &f(t_1) \leq f(t_2) \Leftrightarrow f(t_1) \leq f(t_2) \quad \forall i \in 1,2,...N_{obj} \\ &f(t_1) < f(t_2) \Leftrightarrow \left(f(t_1) \leq f(t_2)\right) \wedge \left(f(t_1) \neq f(t_2)\right) \end{aligned} \qquad 4\text{-}3$$

siendo imposible establecer un orden estricto entre las soluciones ya que, por ejemplo, puede darse el caso de que $f_1(t2) < f_1(t3)$ y $f_2(t2) < f_2(t3)$. Debido a esto, se obtienen conjuntos de soluciones parcialmente ordenados [74] donde ninguna solución es mejor que otra. Por tanto, podemos definir la relación de dominancia entre soluciones como:

$$\begin{aligned} &t_1 \prec t_2 \quad (t_1 \text{ domina a } t_2) && \Leftrightarrow f(t_1) < f(t_2) \\ &t_1 \preceq t_2 \quad (t_1 \text{ domina débilmente a } t_2) && \Leftrightarrow f(t_1) < f(t_2) \\ &t_1 \sim t_2 \quad (t_1 \text{ es indiferente a } t_2) && \Leftrightarrow f(t_1) \not\leq f(t_2) \wedge f(t_2) \not\leq f(t_1) \end{aligned} \qquad 4\text{-}4$$

Utilizando esta relación de dominancia se puede, dado un conjunto de soluciones, obtener un subconjunto de éstas donde todas las soluciones no sean dominadas por ninguna otra. Este subconjunto se denomina conjunto de Pareto y el conjunto de Pareto óptimo es el conjunto Pareto de todas las soluciones posibles del espacio solución, es decir, el conjunto de soluciones que no pueden ser dominadas por ninguna otra [21]; [81]; [125].

Así podemos afirmar que la solución t_j es Pareto óptima (pertenece al conjunto Pareto óptimo) si y sólo si:

$$\Delta t_i \in D : t_i \prec t_j \qquad 4\text{-}5$$

donde D es el conjunto de todas las soluciones posibles y Á es el criterio que determina si una solución domina a otra.

El funcionamiento a grandes rasgos del NSGAII se describe a continucación:

Dada la población del algoritmo evolutivo, se generan los descendientes y se aplica una estrategia de reemplazo basándose en la obtención de conjuntos de Pareto no dominados.

Más concretamente, tras generar el conjunto de descendientes se unen las poblaciones de ancestros y descendientes. A partir de este conjunto de individuos, se van obteniendo conjuntos de Pareto que pasarán a formar la población para la siguiente generación, procediendo de este modo hasta que el tamaño de la población se haya completado. Este proceso muestra gráficamente en la figura 4-8.

Otra característica de este algoritmo es la asignación un un ranking para cada individuo de modo que, de entre el último conjunto de Pareto que completa la población, sólo se seleccionan los que están más dispersos en la frontera del Pareto.

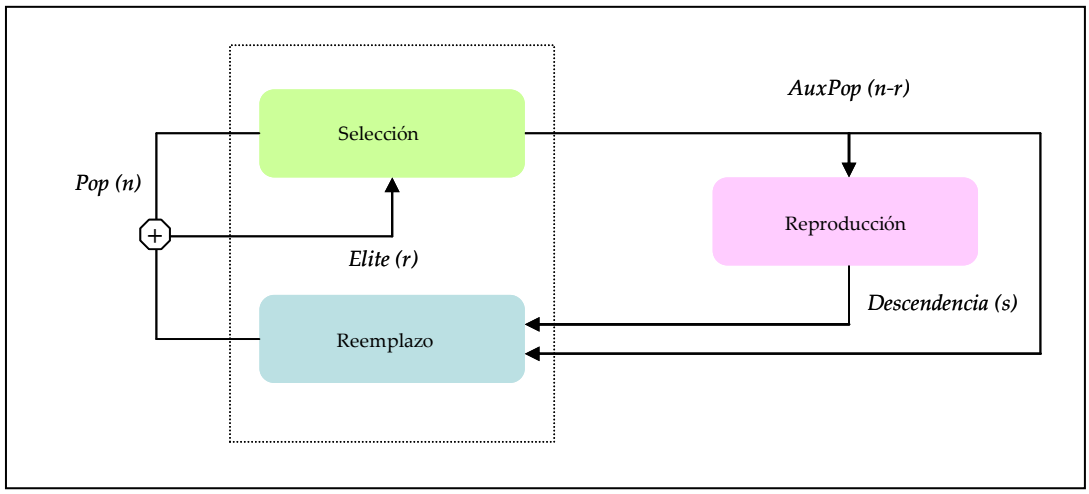


Figura 4-8: Esquema de ejecución de un algoritmo elitista como el NGSA-II

4.4 Algoritmo MOFA

En esta sección se describe el *Multiobjetive Algorithm for Funtion Approximation (MOFA)* utilizado para el diseño de las redes de funciones de base radial. Este algoritmo se basa en el algoritmo genético con clasificación no dominada en su segunda versión (NSGA-II) [76].

El algoritmo MOFA se ha aplicado aquí con dos objetivos. Concretamente se trata de obtener la red con el menor error posible y con la menor cantidad de RBF. Cuanto más grandes llegan a ser las redes, más costosa es su manipulación dentro del algoritmo genético, haciéndola funcionar muy lentamente y para este caso la red debe obtener nuevas habilidades cada ***4 horas***, al considerar que es el mínimo tiempo de muestreo de las variables predictoras.

En la figura 4-9 se muestra el funcionamiento de dicho algoritmo.

```
Inicializar (P(0))
generacion = 0
Evaluar (P(0))
mientras (no CriterioParada) hacer
    R = Padres ∪ Hijos
    Frentes = Sorting No Dominado(R)
    NuevaPop = Ø
    i=1
    mientras |NuevaPop| + |Frentes(i)| ≤ sizepop
        Calcular Distancia de Crowding (Frentes(i))
        NuevaPop = NuevaPop ∪ Frentes(i)
        i++
    Sorting por Distancia (Frentes(i))
    NuevaPop = NuevaPop ∪ Frentes(i)[1:(sizepop - |NuevaPop|)
    Hijos = Seleccion y Reproduccion(NuevaPop)
    generacion ++
    P(generacion) = NuevaPop
retornar Mejor Solucion Hallada
```

Figura 4-9: Pseudocódigo del algoritmo MOFA

A continuación describimos los elementos que garantizan el funcionamiento del algoritmo MOFA.

4.4.1 Representación de RBFNN en los individuos

Como ya fue expuesto en el capítulo 2, para diseñar una RBFNN es necesario especificar:

1. el número de RBFs
2. la posición de los centros del RBFs
3. la longitud de los radios
4. los pesos para la capa de la salida

Los individuos en la población del algoritmo MOFA contendrán los primeros tres elementos en un vector de números reales. En vez de incluir los pesos, el error de la aproximación se almacena para ahorrar esfuerzo de cómputo en el momento en que comparen a los individuos.

El error local se define como la suma de los errores entre la salida real y la salida generada por la RBF pero, en vez de considerar todos los vectores de la entrada, sólo serán seleccionados los que activan cada RBF. Para conocer si un vector de entrada activa una RBF, su función de activación se calcula para cada vector de entrada y si es mayor que un determinado umbral, entonces el vector de entrada activa la neurona.

4.4.2 Población Inicial

La población inicial se genera utilizando diferentes algoritmos de clustering, para obtener buenos individuos que permitan hallar de manera más fácil y rápida buenas soluciones. Estos algoritmos de clustering se describen a continucación.

4.4.2.1 Algoritmo FCM, Fuzzy C-Media, [62]:

En general, las técnicas de agrupamiento difusa se basan en encontrar el mínimo de una función objetivo que determina los prototipos de los grupos buscados. La función objetivo base de una gran familia de algoritmos de agrupamiento borroso es la siguiente:

$$J_h(U, C; X) = \sum_{k=1}^{n} \sum_{i=1}^{m} u_{ik}^{h} \|x_k - c_i\|^2 \qquad 4\text{-}6$$

Donde *X* es el vector de entrada y *C* el vector de los centros.

Este algoritmo realiza una partición borrosa de los datos de entrada donde el mismo vector de entrada puede pertenecer a varios clusters, y al mismo tiempo con un grado de calidad de miembro.

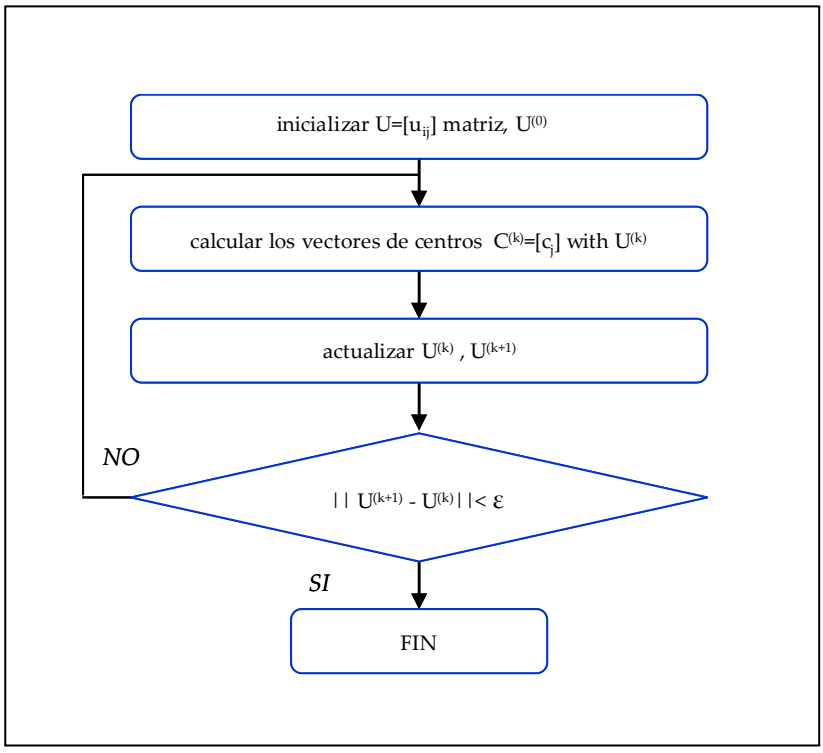


Figura 4-10. Diagrama del Algoritmo Fuzzy C-Media

4.4.2.2 Algoritmo IPCFA, Cluster Probabilístico Mejorado para Aproximación Funcional, [62]:

Este algoritmo corresponde a una modificación del algoritmo PCM propuesta en [140] donde se emplea una combinación de una partición posibilística y una difusa, con lo cual se pueden obtener diferentes clusters. Para llevar a cabo su propósito, los autores reemplazan el criterio de similitud en la función de distorsión del FCM (la distancia Euclídea) por la función de distorsión definida en el PCM.

El algoritmo IPCM, al igual que el FPCM, presentan un enfoque híbrido entre una partición difusa y una posibilística. Cuando este algoritmo se ejecuta, reparte los centros de manera uniforme por el espacio de entrada poniéndose de manifiesto que no padece los problemas que presenta el PCM. Al provenir directamente del enfoque posibilístico, es necesario dar valor a los mismos parámetros que requería el PCM además de al exponente para la función de pertenencia que requiere el enfoque difuso.

La figura 4-11 indica los principales pasos de este algoritmo.

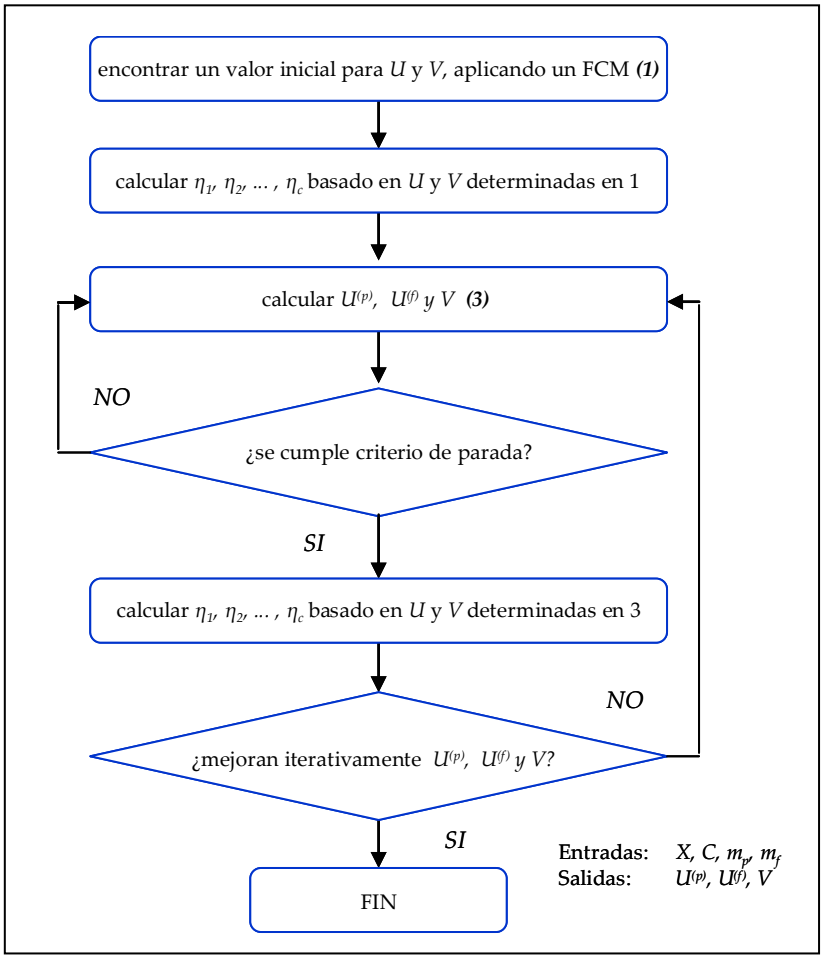


Figura 4-11. Diagrama del Algoritmo IPCFA

La adaptación del algoritmo ICFA a este tipo de enfoque mixto tiene un comportamiento bastante bueno, situando los centros donde la función es más variable pero manteniendo un compromiso entre las distancias que separan a los centros.

4.4.2.3 Algoritmo PCI, Inicializador Probabilístico de Centros, [62]:

Este algoritmo utiliza una partición posibilística y una partición borrosa, combinando ambas como en [140].

Esta función se obtiene substituyendo la distancia medida por el algoritmo FCM por la función objetivo del algoritmo PCM, con lo cual se logra un acercamiento mixto. El parámetro escala determina la relación entre el primer término de la función objetivo con relación al primero. Este segundo término implica que la calidad de miembro sea tan grande como sea posible, eligiendo este valor para mantener un equilibrio entre las calidades de miembro borrosas y posibilísticas.

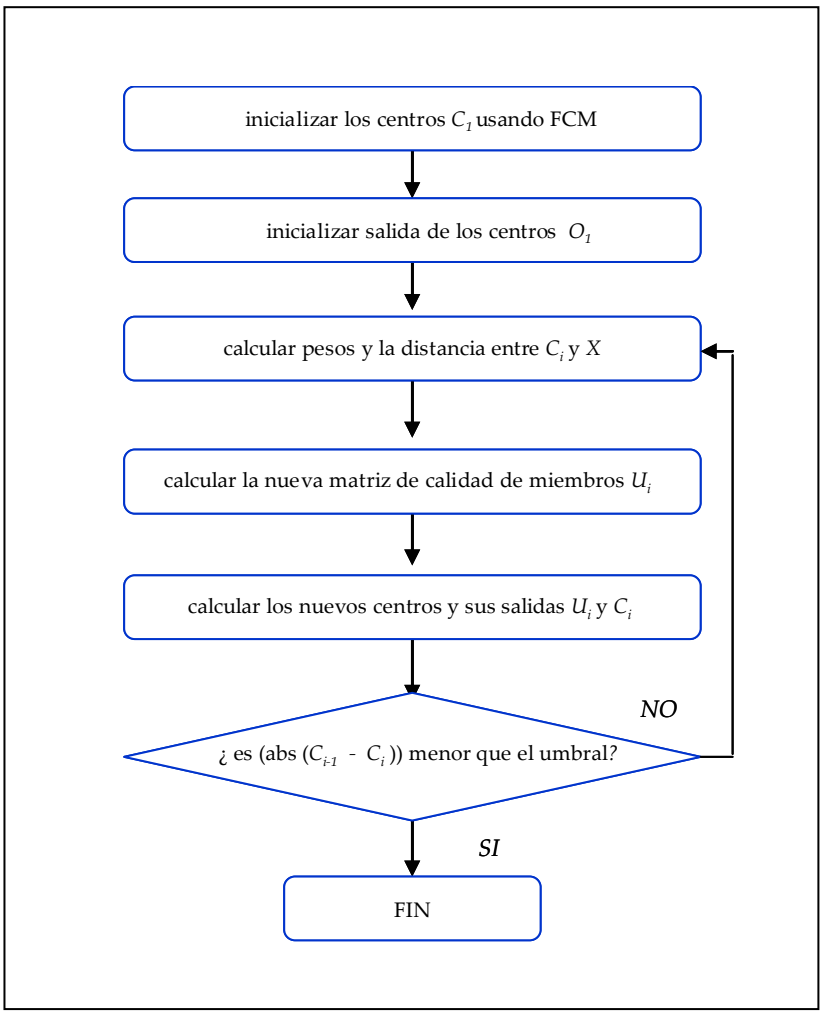


Figura 4-12. Diagrama del Algoritmo PCI

4.4.2.4 Algoritmo ICFA, Cluster Mejorado para Aproximación Funcional, [58]:

En el algoritmo ICFA, se define una partición borrosa que es substituida por las que son usadas en [71] y [98] respectivamente. Este algoritmo [61] utiliza clustering supervisado para identificar las áreas donde la función es más variable. Para ello, se define el concepto de salida estimada de un centro para asignar un valor para el centro en el eje de salida. Existen también individuos incluidos aleatoriamente para no perder diversidad en la población.

Desde el punto de vista de una RBFNN, el hecho de que un vector de entrada active una única neurona no es del todo adecuado puesto que éste está conectado con todas las RBF de la red. Es más, si un vector de entrada no pudiera activar varias neuronas simultáneamente se perdería la capacidad de generalización e interpolación de la red. Por tanto, parece más adecuado emplear una partición difusa durante la inicialización de los centros de la red para poder ubicarlos teniendo en cuenta que un vector de entrada puede influir en la posición de varios centros. En [58], [59], [60] se realizó un estudio previo de lo beneficioso que puede ser el uso de clustering difuso en problemas de aproximación funcional.

En el algoritmo ICFA, el punto de partida no consiste en una inicialización aleatoria de los centros o de la matriz con los valores de pertenencia, sino que se fija un valor concreto para las posiciones de los centros. Éstos se distribuyen uniformemente por todo el espacio de entrada y todas sus salidas estimadas son inicializadas a 1. Dado que se realiza un descenso en gradiente, se alcanza un mínimo local y, aunque la migración ayude a mejorar la convergencia a un mínimo local mejor, no lo garantiza. La consecuencia es una falta de robustez del algoritmo, tal y como ocurre con otros algoritmos.

Por tanto se ha decidido dar una inicialización fija de modo que para la misma entrada, el algoritmo devuelve la misma salida siempre. La posición inicial de los centros permite que un vector de entrada, no importa dónde se encuentre, tendrá un centro cercano desde el punto de partida del algoritmo.

La razón de fijar las salidas estimadas al mismo valor tiene como motivo ser equitativo a la hora de calcular por primera vez las distancias entre los centros y los vectores de entrada.

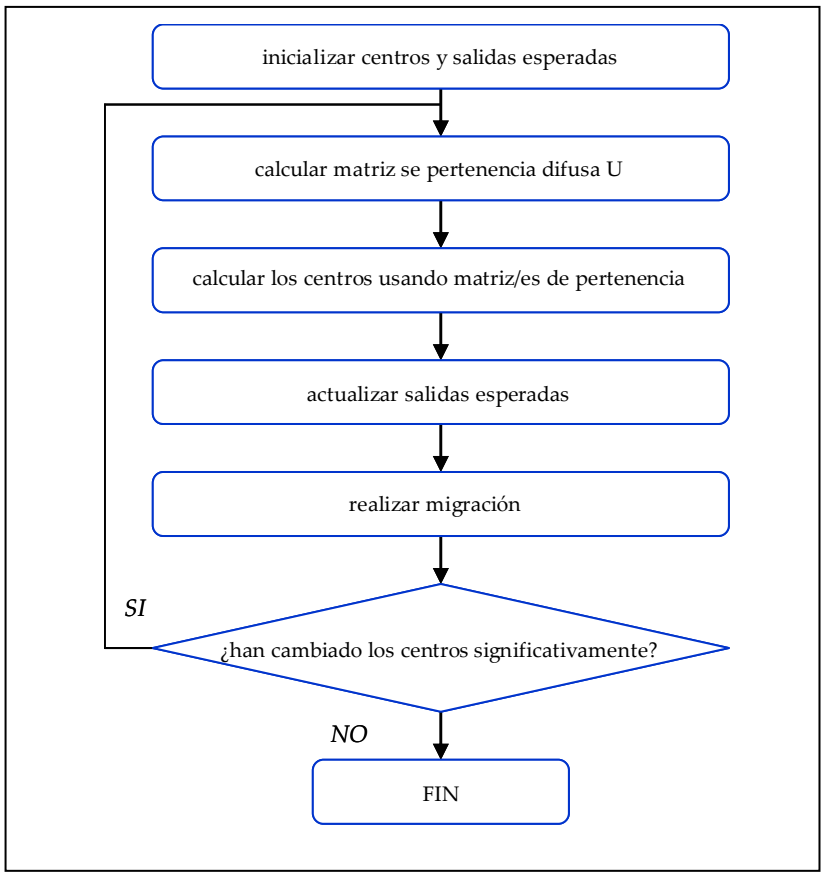


Figura 4-13. Diagrama del Algoritmo ICFA

Después de esta inicialización, muy pocas iteraciones de un algoritmo local de búsqueda (Levenberg-Marquardt) [23] se ejecutan y los resultados se concatenan a la población.

Esto se hace con la finalidad de mejorar la calidad de los resultados, puesto que la población se hace más diversa, y que los algoritmos de clustering son absolutamente robustos, se podrían generar individuos bastantes similares.

El tamaño de la red RBF que pertenece a la primera generación debe ser pequeño por dos razones:

1. Hacer la inicialización tan rápida como sea posible
2. Permitir que el algoritmo genético determine los tamaños del RBFNNs desde el punto de vista incremental, ahorrando el esfuerzo de cómputo que supondría ocuparse de redes grandes de las primeras.

Los operadores de cruce tendrán la ocasión de incrementar el número de RBF y junto con los operadores de mutación permitirán eliminar las RBFs inútiles.

4.4.3 Operadores de Cruce

El operador original de cruce sobre un cromosoma codificado en binario o con números reales no puede aplicarse con los individuos de este algoritmo porque cada uno los grupos de genes tienen significados diferentes.

Dos operadores de cruce se han diseñado para estos individuos, y se ha determinado experimentalmente que el uso de ambos operadores con la misma probabilidad, proporciona mejores resultados que aplicando solamente uno de ellos.

4.4.3.1 Operador de Cruce 1: Intercambio de Neuronas

Este operador de cruce, es conceptualmente más similar al cruce original. Puesto que los individuos representan una red RBF con varias neuronas, el cruce de dos individuos será el resultado de intercambiar una neurona. Este intercambio se representa en la figura 4-14.

Las ventajas de este operador de cruce son que explota el material genético de cada individuo sin modificar la estructura de la red por un lado y su simplicidad y eficiencia por el otro.

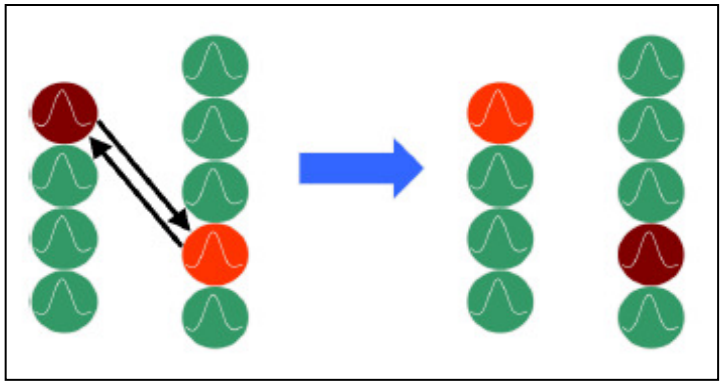


Figura 4-14. Operadores de Cruce 1

4.4.3.2 Operador de Cruce 2: Adición de la neurona con el error más pequeño

Este operador consiste en añadir la neurona con el error local más pequeño de las que pertenecen al otro individuo. Se representa en la figura 4-15.

Si la neurona con el error local más pequeño es muy similar a otra en la otra red, se elige la neurona con el segundo error más pequeño.

Este operador proporciona la oportunidad de aumentar el número de redes RBFs en un individuo, permitiendo que el algoritmo explore más topologías.

Un paso adicional se muestra en la figura 4-15. Este paso de refinamiento consiste en la eliminación de la red de las RBF que no tienen influencia en la salida de la red. Para hacer esto se calculan todos los pesos que conectan las unidades de proceso con la capa de la salida y las neuronas que no tienen un peso significativo se eliminan.

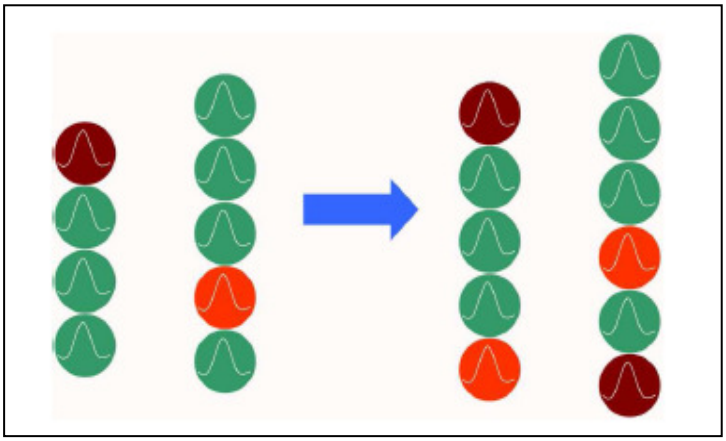


Figura 4-15: Operadores de Cruce 2

4.4.4 Operadores de Mutación

Los operadores de mutación propuestos para este algoritmo se pueden clasificar en dos categorías:

1. Mutaciones sin conocimiento incorporado
2. Mutaciones que incorporan el conocimiento experto

El primer caso se refiere a aquellas mutaciones que se realizan de manera aleatoria. Esos cambios pueden afectar la estructura y los parámetros de la red RBF.

El objetivo de estos operadores es incorprar efectos aleatorios en el proceso de búsqueda para evitar la convergencia a mínimos locales.

Los operadores de mutación que incorporan conocimiento experto, son mutaciones que afectan también la estructura y los parámetros de la red RBF, pero aprovechan cierta información para que los cambios no sean totalmente al azar.

Mientras esto ocurre con los operadores de cruce, si dividimos estos subconjuntos de operadores de mutación y se realizan diferentes ejecuciones, los resultados obtenidos serán peores que si se ejecuta el algoritmo utilizando ambas clase de operadores de mutación.

A continucación se describen más detenidamente los diferentes tipos de mutaciones.

4.4.4.1 Mutaciones sin ningún conocimiento:

Existen cuatro operadores que son totalmente aleatorios:

1. Eliminación de una red RBF en una posición elegida al azar sobre el espacio de los vectores de la entrada que fija su radio también con un valor al azar. Todos los valores elegidos al azar están en el intervalo [0 a 1], pues los vectores de entrada y sus salidas se normalizan.
2. El segundo operador es el contrario el anterior, suprimiendo una red RBF existente. Esta mutación es forzada y no se aplica cuando el individuo tiene menos de dos neuronas
3. Agregar a todas las coordenadas de un centro una distancia elegida al azar, en el intervalo [- 0.5 a 0.5].
4. Este operador tiene exactamente el mismo comportamiento que el anterior pero cambia el valor del radio de la red RFB seleccionada.

Los dos primeros operadores modifican la estructura de la red mientras que el tercero y el cuarto modifican los parámetros de la red. Los operadores terceros y cuarto se refieren a los algoritmos genéticos como se presenta en [37]

4.4.4.2 Mutaciones con el conocimiento experto

Estos operadores de mutación utilizan la información proporcionada por la salida de la función que se aproxima. Igual que los operadores anteriores, éstos afectan la estructura de la red, agregando y quitando redes RBF, y también modificarán el valor de los parámetros de las redes RBF.

Los operadores son:

1. Insertar una red RBF en la posición del vector de entrada con el error más alto. Para seleccionar esta posición, se calcula la salida de la red de RBF y se compara

con la salida de la función destino. El centro se sitúa en la posición del punto donde existe la mayor diferencia entre la salida verdadera y la salida generada

2. Quitar una RBF de la red de RBF. La RBF que se quita es la que tiene menor error local. Esto podría parecer ilógico al principio, pero permite mantener una mayor diversidad de la población, puesto que uno de los operadores de cruce agrega la neurona del individuo con menor error local. Así al combinar estos elementos, los genes permanecerán en la población para evitar elementos redundantes y permitir la búsqueda en nuevas áreas del espacio de entrada
3. Aplicar un algoritmo de búsqueda local, por ejemplo el (Levenberg-Mardquardt) para refinar las posiciones de los centros y de sus radios, pero teniendo cuidado de que con estos movimientos el error disminuye. Este operador debe utilizarse cuidadosamente durante pocas iteraciones. De lo contrario la población convergerá demasiado rápido a mínimos locales.

4.5 Conclusiones:

En este capítulo se ha descrito el procedimiento que presentamos para ser empleado en la caracterización del comportamiento de algunos procesos que se presentan en la reducción de mineral de la tecnología CARON.

Uno de los módulos que construyen el procedimiento se encarga de predecir lo que puede ocurrir a corto plazo utilizando para ello datos previamente extraídos del proceso industrial. El modelo consiste en una red neuronal de RBF. Mediante un análisis y selección previa de las variables predictoras pueden aportar mejoras en el rendimiento del diseño de la propia estructura de la red. Asi mismo se utiliza un algoritmo genético multiobjetivo que entrena a la red dentro del marco de tiempo requerido por el sistema.

Tras describir el proceso CARON en el capítulo 3, y el procedimiento que utilizamos para modelar y caracterizar el mismo en este capítulo 4, dedicaremos el próximo capítulo a mostrar los resultados experimentales obtenidos y extraer las correspondientes conclusiones.

Capítulo 5

Experimentos y resultados

En este capítulo se presentan los distintos experimentos realizados con el conjunto de datos seleccionados en 4.1 utilizando el algoritmo MOFA descrito en el capítulo anterior. Estos resultados ponen de manifiesto la posibilidad de mejorar el rendimiento del proceso industrial que consideramos, a partir de un modelo del mismo en base a redes RBF.

En una primera parte se describe la forma en qué se han elegido los parámetros del algoritmo genético que intervienen en su ejecución. Seguidamente se muestran los resultados obtenidos a partir de los diferentes experimentos. Finalmente se describe de manera muy sencilla el sistema que permitirá llevara a la práctica el procedimiento diseñado para tales objetivos.

5.1 Selección de parámetros para los experimentos:

De acuerdo a lo planteado en el capítulo 2, las Redes RBF se han formado teniendo en cuenta la formulación general del proceso de Reducción de Mineral de Punta Gorda, que se describe a través de la expresión 3.7 del capítulo 3.

Las expresiones 3.4 a la 3.7 muestran las variables más importantes en la operación de dicho proceso, y de las cuales también se cuenta con datos históricos. Dichos datos se resumen cada ***4 horas*** en las condiciones actuales.

El ajuste que se necesitaría obtener para brindar la predicción del valor siguiente debe estar por debajo de este tiempo; *4 horas es un valor aceptable*, teniendo en cuenta que hay que cumplir con dos requisitos: tiempo de ejecución del algoritmo y tiempo de muestreo de las calidades del mineral de entrada al proceso.

Para la realización de los diferentes experimentos, los parámetros tamaño de población *(P)*, probabilidad de mutación *(Pm)* y de cruce *(Pc)*, necesitan ser configurados adecuadamente con el fin de lograr una optimización del algoritmo genético.

El problema de la asignación óptima de valores a los parámetros de un AE ha sido estudiado arduamente a lo largo de los años [50], incluso se han hecho clasificaciones de los AEs tomando como punto de referencia la forma en que se manejan los valores de los parámetros dentro de estos.

A continuación describimos la forma en que se han seleccionado los valores de los parámetros para la aplicación que estamos considerando. Cada experimento se ha denominado con una combinación de número y letra para su diferenciación, por ejemplo, el ***1A*** corresponde al primer resultado del experimento 1.

5.1.1 Tamaño de la Población

El tamaño de la población es un parámetro que influye determinantemente en los resultados obtenidos por un algoritmo evolutivo [70]. Un tamaño pequeño produce que la ejecución del algoritmo sea rápida, ya que se tendrán que procesar pocos individuos en cada generación, pero suele implicar una rápida pérdida de la diversidad de la población, o al menos supone una limitación en la misma. Por el contrario, un tamaño elevado hace la ejecución del algoritmo muy lenta, aunque ayuda a mantener una población diversa.

Para determinar el tamaño de la población, en [56] se lleva a cabo un estudio empírico siguiendo las sugerencias de [28] y [105].

Como se muestra en la tabla 5.1, el error de aproximación de las soluciones se comienza afectar cuando aumenta el tamaño de la población a partir de un número determinado.

Estos resultados sugieren la elección de un tamaño de población medio (70 a 120), de forma que el algoritmo sea lo más rápido posible, pero sin sacrificar a cambio la calidad de la solución final obtenida.

En la tabla 5-1 se tienen los resultados obtenidos. El mejor resultado corresponde a la ejecución ***1A***.

Parámetros	Ejecución 1A	Ejecución 1B	Ejecución 1C
Prob.Cruce	0,800	0,800	0,800
Prob.Mutación	0,150	0,150	0,150
Tamaño Pobl.	***50***	***100***	***150***
No. Generaciones	50	50	50
No. RBF	6	6	6
Vector Entrada	2400	2400	2400
Dimensión	8	8	8
Vector Entrenamiento	1902	1902	1902
Vector Test	498	498	498
NRMSE Error test	0,919	0,924	0,936
NRMSE Error train	0,879	0,918	0,918
Tiemp. Ejecuc. seg	591,710	1156,000	1547,000
Tiemp. Ejecuc. min	9,862	19,267	25,783

Tabla 5-1: Influencia del parámetro tamaño de la población

5.1.2 Probabilidad de cruce y de mutación

Para que la población evolucione, es necesario generar nuevos individuos con el fin de que algunos de estos sean mejores que los que constituyen la población actual. Los individuos se crean a partir de transformaciones que involucran a uno o más individuos ya existentes en la población de partida, a los que se les llama padres. El escoger a los mejores individuos como padres, es de esperar que aumente la posibilidad de generar buenos individuos. La mutación sólo necesita un padre para generar un nuevo individuo, cambiando una parte del genotipo del padre. El cruce necesita de dos o más padres para generar un individuo. Para ello se combinan partes de su genotipo. En las tablas 5-2 y 5-3 se muestran los resultados obtenidos. El mejor resultado corresponde a la ejecución ***2D***.

Parámetros	Ejecución 2A	Ejecución 2B	Ejecución 2C
Prob.Cruce	***0,800***	***0,800***	***0,800***
Prob.Mutación	***0,150***	***0,150***	***0,150***
Tamaño Pobl.	50	100	150
No. Generaciones	50	50	50
No. RBF	6	6	6
Vector Entrada	2400	2400	2400
Dimensión	8	8	8
Vector Entrenamiento	1902	1902	1902
Vector Test	498	498	498
NRMSE Error test	0,919	0,924	0,936
NRMSE Error train	0,879	0,918	0,918
Tiemp. Ejecuc. seg	591,710	1156,000	1547,000
Tiemp. Ejecuc. min	9,862	19,267	25,783

Tabla 5-2: Comportamiento de los Parámetros probabilidad de cruce y mutación, 1

Parámetros	Ejecución 2D	Ejecución 2E	Ejecución 2F
Prob.Cruce	***0,800***	***0,800***	***0,800***
Prob.Mutación	***0,035***	***0,035***	***0,035***
Tamaño Pobl.	50	100	150
No. Generaciones	50	50	50
No. RBF	6	6	6
Vector Entrada	2400	2400	2400
Dimensión	8	8	8
Vector Entrenamiento	1902	1902	1902
Vector Test	498	498	498
NRMSE Error test	0,896	0,909	0,924
NRMSE Error train	0,876	0,890	0,918
Tiemp. Ejecuc. seg	475,56	1047,01	1426,8
Tiemp. Ejecuc. min	7,92	17,45	23,78

Tabla 5-3: Comportamiento de los Parámetros probabilidad de cruce y mutación, 2

5.1.3 Tamaño de las RBF

Aunque existen propuestas en las que la población que se maneja está constituida por un conjunto de RBF [117], en la mayoría de las propuestas evolutivas de diseño de redes RBF, un individuo representa una RBF completa [86]. De esa manera los operadores evolutivos actúan sobre los individuos añadiendo RBF, eliminándolas y modificándolas.

El diseño de reglas para este fin, tiene en cuenta el hecho de que una RBF es peor si su contibución (a_i) es baja, su error es alto (e_i) es alto y su solapamiento (o_i) también es alto. Por el contrario, una red de RBF es mejor cuando su contibución es alta y su error y solapamiento son bajos [86].

Así se tiene que la probabilidad de eliminar una RBF se incrementa cuando ésta tiene un mal comportamiento; la probabilidad de mutar va aumentando a medida que el comportamiento es mejor. Sin embargo, y según el trabajo de [109], la computación evolutiva tradicional presenta ciertos problemas relacionados con la evaluación de subcomponentes independientes. La coevolución cooperativa logra un entorno de diseño en el que los individuos de la población no sólo compiten por sobrevivir, sino que deben cooperar para lograr una solución.

En la tabla 5-4 se tienen los resultados obtenidos. El mejor resultado corresponde a la ejecución ***3B***.

Parámetros	Ejecución 3A	Ejecución 3B	Ejecución 3C	Ejecución 3D
Prob.Cruce	0,800	0,800	0,800	0,800
Prob.Mutación	0,035	0,035	0,035	0,035
Tamaño Pobl.	**50**	**50**	**50**	**50**
No. Generaciones	80	80	80	80
No. RBF	***4***	***6***	***8***	***10***
Vector Entrada	2400	2400	2400	2400
Dimensión	8	8	8	8
Vector Entrenamiento	1902	1902	1902	1902
Vector Test	498	498	498	498
NRMSE Error test	0,92528	0,92091	0,9433	0,93296
NRMSE Error train	0,92653	0,86627	0,8958	0,92004
Tiemp. Ejecuc. seg	562.77	793,19	2470,00	3091,6
Tiemp. Ejecuc. min	9,37	13,21	41,16	51,51

Tabla 5-4: Parámetro tamaño de las RBF

5.2 Experimentos realizados:

A continuación se describen los experimentos que se han realizado con los mejores valores obtenidos para los parámetros tamaño de la población, probabilidades de cruce y mutación y número de RBF, tal y como se ha se ilustrado en la sección anterior.

En todos experimentos se utilizaron los siguientes parámetros:

Prob. de Cruce	Prob. de Mutación	Población	Generaciones
0.8	0.035	30-120	50-220

Tabla 5-5: Parámetros utilizados en el Algoritmo Genético

Los experimentos realizados se han agrupado como se indica a continuación.

En un primer grupo (Grupo 1) se consideran la totalidad de las variables como se muestra en la tabla 5-6. El segundo grupo (Grupo 2) incluye los experimentos con un espacio de 8 variables como lo muestra la tabla 5-7. Finalmente el grupo tercero (Grupo 3) considera una dimensión de 7 variables, tabla 5-8.

1	2	3	4	5	6	7
Ni	*Co*	*Fe*	*Si*	*Mg*	*CO/CO₂*	*Ton*
8	9	10	11	12	13	14
A_Petr	*TH0*	*TH2*	*TH4*	*TH6*	*TH7*	*TH9*
15	16	17	18	19	20	21
TH11	*TH13*	*TH14*	*TC6N*	*TC6S*	*TC8N*	*TC8S*
22	23	24	25	26	27	
TC10N	*TC10S*	*TC12N*	*TC12S*	*TC15N*	*TC15S*	

Tabla 5-6: Variables utilizadas, dimensión 27

1	2	3	4	5	6	7
Ni	*Co*	*Fe*	*Si*	*Mg*	*CO/CO₂*	*Ton*
8	9	10	11	12	13	14
A_Petr	-	-	-	-	-	-
15	16	17	18	19	20	21
-	-	-	-	-	-	-
22	23	24	25	26	27	
-	-	-	-	-	-	

Tabla 5-7: Variables utilizadas, dimensión 8

1	2	3	4	5	6	7
Ni	*Co*	*Fe*	*Si*	*Mg*	*CO/CO₂*	*A_Petr*
8	9	10	11	12	13	14
-	-	-	-	-	-	-
15	16	17	18	19	20	21
-	-	-	-	-	-	-
22	23	24	25	26	27	
-	-	-	-	-	-	

Tabla 5-8: Variables utilizadas, dimensión 7

En el análisis de los resultados que se realizarán en la próxima sección, se muestran en forma de tablas los valores de los diferentes parámetros que hemos considerado importante para llegar a conclusiones.

Por otra parte, la representación gráfica tendrá el mismo formato e interpretación para todos los experimentos considerados mejores. La misma hace referencia a la tendencia de la variable *I_Petr*. a lo largo de los diferentes puntos de muestreo (eje X), tanto para los valores reales (color azul) como para los valores aproximados (color magenta). De la misma manera se muestran los valores del error (color amarillo) que representa la diferencia entre el valor real y el estimado.

5.2.1 Resultados para la dimensión de 27 variables:

El conjunto de ejecuciones denominado ***4-A,B,C,D*** correspondiente a la tabla 5-9 muestra los resultados obtenidos con un vector de entrada de 27 variables. El mejor resultado se ha obtenido en el experimento ***4C***. En la medida que crecen el tamaño de la población y las generaciones, se emplea más tiempo en obtener un resultado, sin embargo el error comienza a incrementarse a partir del momento en que se utiliza una población de más de 90 individuos.

Parámetros	Ejecución 4A	Ejecución 4B	Ejecución 4C	Ejecución 4D
Prob.Cruce	0,800	0,800	0,800	0,800
Prob.Mutación	0,035	0,035	0,035	0,035
Tamaño Pobl.	30	70	90	120
No. Generaciones	50	90	120	220
No. RBF	6	6	6	6
Vector Entrada	2416	2416	2416	2416
Dimensión	***27***	***27***	***27***	***27***
Vector Entrenamiento	1930	1930	1930	1930
Vector Test	486	486	486	486
NRMSE Error test	0,938	0,927	**0,899**	0,927
NRMSE Error train	0,972	0,950	**0,930**	0,963
Tiemp. Ejecuc. seg	450,99	1287,20	2420,60	4254,70
Tiemp. Ejecuc. min	7,52	21,45	40,34	70,91

Tabla 5-9: Resultados utilizando 27 variables

La figura 5-1 muestra el comportamiento de los datos de salida una vez obtenida la aproximación funcional para la dimensión de 27 variables. El mayor error está asociado a los valores máximos y mínimos del conjunto de datos de la variable de salida (63,51 y 39,81).

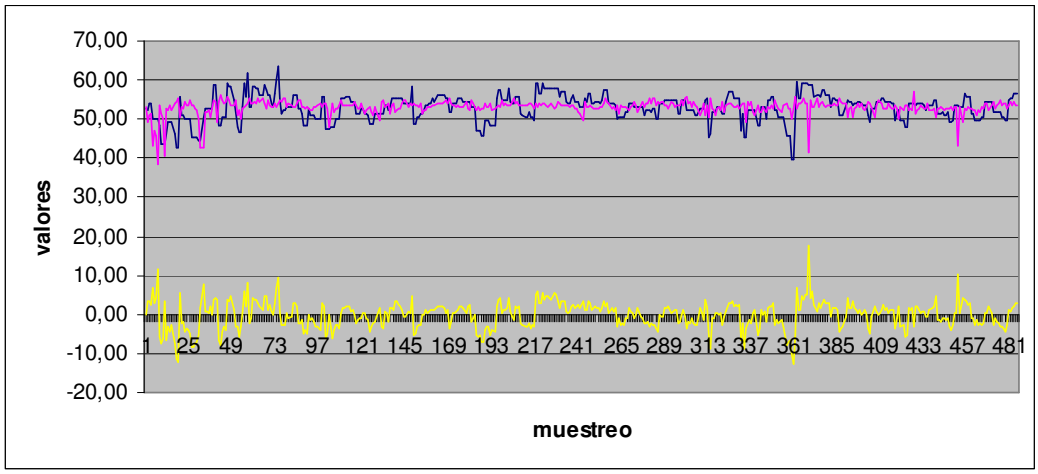


Figura 5-1: Aproximación con 27 variables:
valor real (azul), valor estimado (magenta), diferencia (amarillo)

5.2.2 Resultados para la dimensión de 8 variables:

El conjunto de ejecuciones denominado ***5-A,B,C,D*** correspondiente a la tabla 5-10 muestra los resultados obtenidos con un vector de entrada de 8 variables. El mejor resultado se ha obtenido en el experimento ***5B***. Con relación a los experimentos anteriores, se tiene una disminución del tiempo de ejecución del algoritmo, asimismo el error también comienza a incrementarse pero a partir del momento en que se utiliza una población de más de 70 individuos.

Parámetros	Ejecución 5A	Ejecución 5B	Ejecución 5C	Ejecución 5D
Prob.Cruce	0,800	0,800	0,800	0,800
Prob.Mutación	0,035	0,035	0,035	0,035
Tamaño Pobl.	30	70	90	120
No. Generaciones	50	90	120	220
No. RBF	6	6	6	6
Vector Entrada	2416	2416	2416	2416
Dimensión	***8***	***8***	***8***	***8***
Vector Entrenamiento	1930	1930	1930	1930
Vector Test	486	486	486	486
NRMSE Error test	0,923	0,911	0,919	0,932
NRMSE Error train	0,937	0,908	0,932	0,958
Tiemp. Ejecuc. seg	255,71	817,87	2024,50	2160,40
Tiemp. Ejecuc. min	4,26	23,63	33,74	46,01

Tabla 5-10: Resultados utilizando 8 variables

La figura 5-2 muestra el comportamiento muy similar al análisis con 27 variables. En este caso el mayor error también está asociado a los valores máximos y mínimos del conjunto de datos de la variable de salida (63,51 y 39,81).

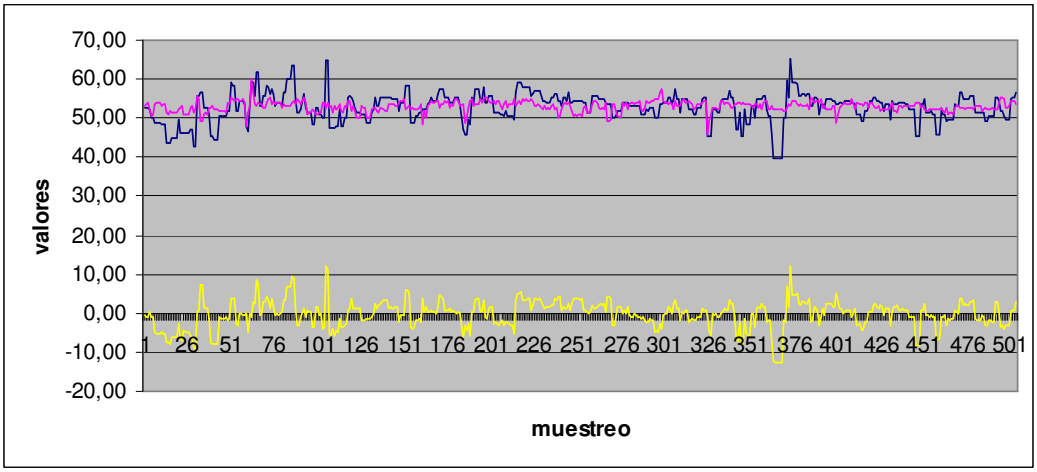


Figura 5-2: Aproximación con 8 variables:
valor real (azul), valor estimado (magenta), diferencia (amarillo)

5.2.3 Resultados para la dimensión de 7 variables:

El conjunto de ejecuciones denominado ***6-A,B,C,D*** correspondiente a la tabla 5-11 muestra los resultados obtenidos con la totalidad de las variables. El mejor resultado se ha obtenido en el experimento ***6D***. En este caso se aprecia otra disminución del tiempo de ejecución del algoritmo, se aprecia por su parte que el menor error se obtiene a cuando se utiliza una población de más de 120 individuos.

Parámetros	Ejecución 6A	Ejecución 6B	Ejecución 6C	Ejecución 6D
Prob.Cruce	0,800	0,800	0,800	0,800
Prob.Mutación	0,035	0,035	0,035	0,035
Tamaño Pobl.	30	70	90	120
No. Generaciones	50	90	120	220
No. RBF	6	6	6	6
Vector Entrada	2416	2416	2416	2416
Dimensión	7	7	7	7
Vector Entrenamiento	1930	1930	1930	1930
Vector Test	486	486	486	486
NRMSE Error test	0,932	0,901	0,906	0,882
NRMSE Error train	0,945	0,936	0,948	0,923
Tiemp. Ejecuc. seg	208,57	944,67	1018,80	2626,20
Tiemp. Ejecuc. min	3,48	15,74	16,98	43,77

Tabla 5-11: Resultados utilizando 7 variables

Para este caso, la figura 5-3 muestra el mismo comportamiento al análisis realizado con 27 y 8 variables, el mayor error también está asociado a los valores máximos y mínimos del conjunto de datos de la variable de salida (63,51 y 39,81).

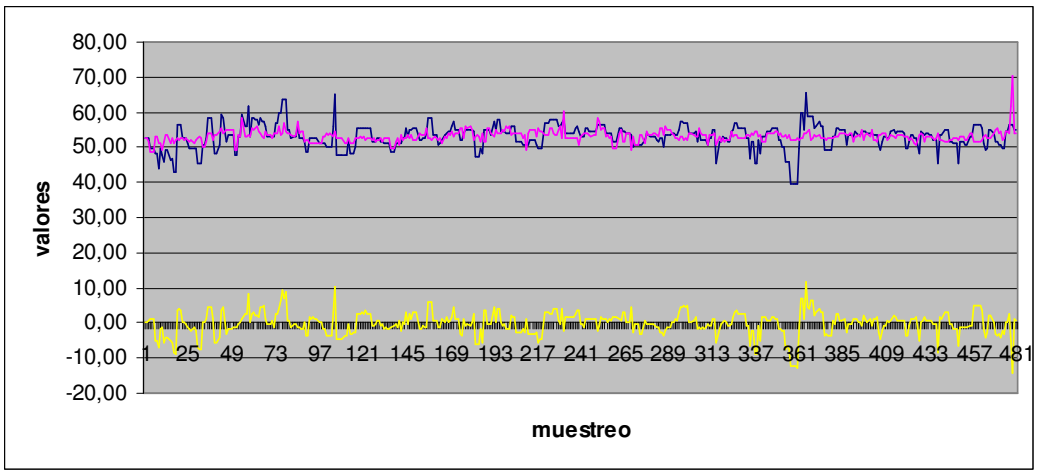


Figura 5-3: Aproximación con 8 variables:
valor real (azul), valor estimado (magenta), diferencia (amarillo)

De acuerdo a los resultados mostrados en el conjunto de tablas y gráficos anteriores, se puede afirmar que los mejores resultados se han obtenido con el espacio de entrada que corresponde a una selección de variables de dimensión siete (7). Con ello conseguimos validar el proceso de selección de variables utilizado.

Para mejorar el error obtenido al aproximar la variable *I_Petro,* se puede realizar una selección de todos aquellos datos que presenten una *desviación estándar menor que 1.* Estos datos coinciden con los valores extremos a que hemos hecho referencia anteriormente. De esta forma, es posible garantizar que más del 90% de los valores aproximados se encuentren, a su vez, dentro de un margen de error del 2.1%.

La operación actual de la industria tiene establecido un entorno de error entre 2,5% y 3.0% debido en lo fundamental a los errores de los instrumentos de medición que utiliza, por lo tanto ese nuevo error obtenido en la ejecución ***7B*** puede ser aceptado.

La tabla 5-12 y la figura 5-4 muestran la nueva aproximación realizada.

Parámetros	Ejecución 7A	Ejecución 7B	Ejecución 7C	Ejecución 7D
Prob.Cruce	0,800	0,800	0,800	0,800
Prob.Mutación	0,035	0,035	0,035	0,035
Tamaño Pobl.	30	70	90	120
No. Generaciones	50	90	120	220
No. RBF	6	6	6	6
Vector Entrada	2120	2120	2120	2120
Dimensión	***7***	***7***	***7***	***7***
Vector Entrenamiento	1679	1679	1679	1679
Vector Test	441	441	441	441
NRMSE Error test	0,904	0,870	0,906	0,920
NRMSE Error train	0,955	0,916	0,948	0,959
Tiemp. Ejecuc. seg	475,21	627,21	1018,80	1654,80
Tiemp. Ejecuc. min	7,92	10,45	16,98	27,58

Tabla 5-12: Resultados utilizando 7 variables con ***desviación std<1***

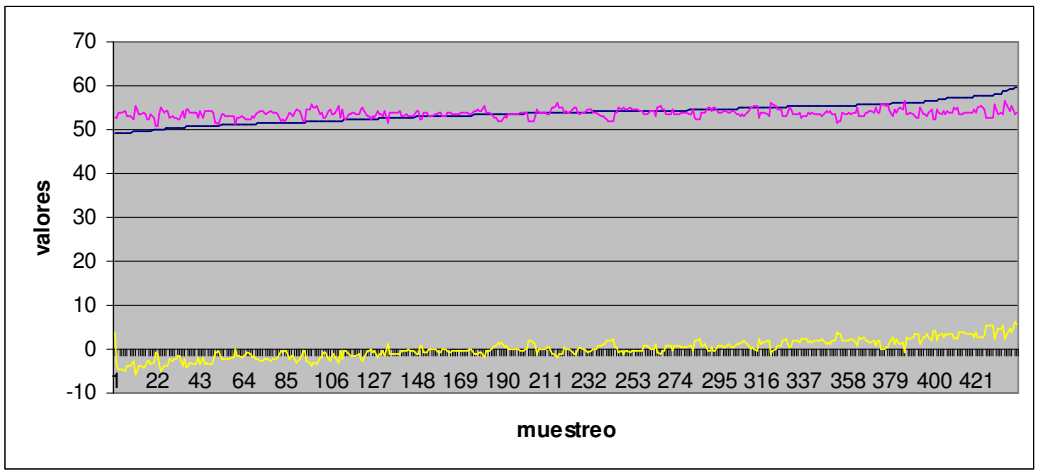


Figura 5-4: Aproximación con 7 variables:
valor real (azul), valor estimado (magenta), diferencia (amarillo)

5.2 Propuesta de aplicación práctica

Conjuntamente con el desarrollo de la investigación realizada, se fue planteando y organizando una estrategia de implementación práctica de los resultados que se debían obtener. La figura 5-5 muestra la manera en que se ha concebido integrar a la gestión empresarial las nuevas aplicaciones que se han definido como *Aplicaciones Softcomputing*.

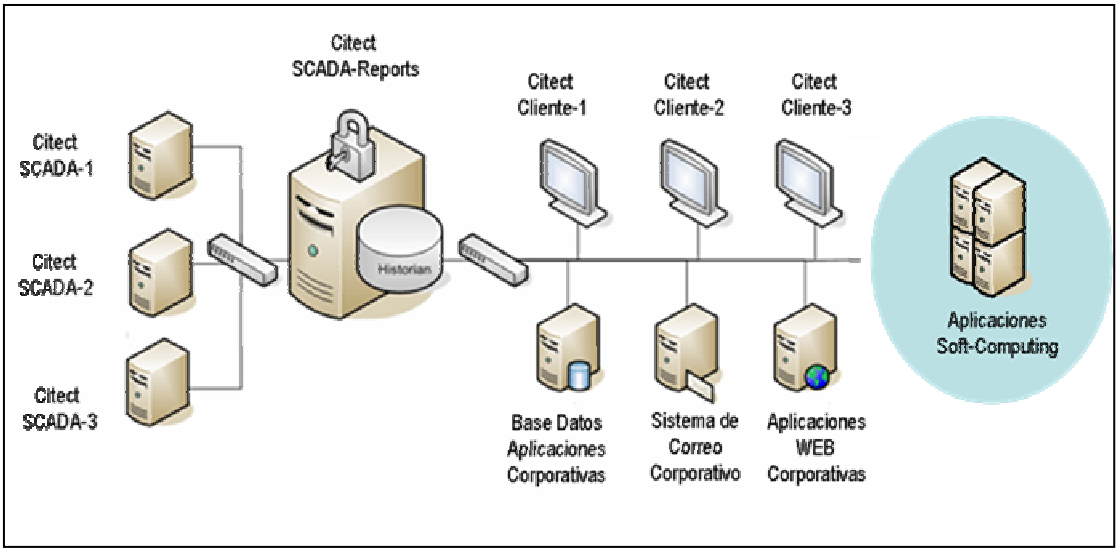


Figura 5-5 Plataforma para aplicaciones Softcomputing

Por lo general, este tipo de aplicaciones se desarrollan sobre una de las tecnologías más importantes y con más perspectivas de desarrollo en el ámbito del supercómputo: *el cluster de computadoras*.

El surgimientos de plataformas computacionales de comunicación y procesamiento estándares de bajo costo, ha brindado la oportunidad de crear herramientas computacionales del dominio público o de costo razonable. Estas realidades permiten la implantación de códigos paralelizados sobre este tipo de plataformas obteniendo un rendimiento competitivo en relación a equipos paralelos especializados cuyos costos de operación y mantenimiento son elevados.

Hoy día los supercomputadores son equipos excesivamente costosos que están fuera del alcance de empresas o instituciones pequeñas. Un cluster, formado por una combinación de equipos microcomputadores, ofrece un rendimiento muy cercano a un supercomputador en cuanto a poder de cómputo.

Actualmente, los clusters están presentes en varias áreas de la ciencia y ya se proyectan en la iniciativa empresarial. Su demanda está asociada básicamente a programas o aplicaciones que requieren enormes recursos computacionales (altas velocidades de procesamiento, gran capacidad de memoria RAM y disco duro, rapidez en la transmisión de información, etc).

Debido a la gran popularidad alcanzada en los últimos años, se han convertido en excelentes herramientas de apoyo para actividades que requieren cómputo paralelo de alto rendimiento.

5.3 Conclusiones

El algoritmo utilizado aproxima con cierta rapidez y adecuada exactitud el *índice de consumo de petróleo en cámaras*, que constituía el objetivo fundamental de nuestro trabajo. Con nuestro procedimiento se puede tener una referencia importante en el momento de predecir la cantidad que será utilizada en el proceso de reducción de mineral a partir de los datos empíricos obtenidos previamente.

De esta forma se pone de manifiesto que el algoritmo desarrollado puede aplicarse prácticamente sin cambios a la solución del problema que nos hemos planteado y donde la selección, agrupamiento y limpieza de los datos utilizados ha sido decisiva a la hora de llegar a esos resultados.

El mejor resultado se ha obtenido luego de haber aplicado un procedimiento de selección de variables, en el experimento ***7B.*** Luego de una valoración de compromiso entre los valores de aproximación obtenidos y el tiempo de ejecución empleado por el algoritmo, se puede considerar adecuado el resultado que se muestra en el apéndice E.

Capítulo 6

Conclusiones y principales aportaciones

Durante la realización de esta investigación, hemos podido inciar un trabajo muy importante para la Empresa Ernesto Che Guevara, Planta de Níquel de Punta Gorda en el ámbito de la aplicación de técnicas de aproximación funcional basadas en softcomputing, con vistas a mejorar su rendimiento.

Para la industria, ha supuesto una gran oportunidad el hecho de poder iniciar la aplicación de extracción de conocimientos a los datos que los sistemas instalados proporcionan diariamente. A continuación se emuneran las aportaciones principales del trabajo de investigación realizado.

1. El capítulo 2 ha presentado las ideas más significativas de lo que se que conoce por Softcomputing, así como una simple reseña de una posible taxonomía. Se describieron los paradigmas que históricamente han definido el trabajo dentro del campo de la computación evolutiva y la resolución de problemas de optimización, mediante técnicas evolutivas. También se hizo referencia a las características más importantes de las redes de Funciones de Base radial (RBF), entre las que se destacan su capacidad implícita en la aproximación funcional. Se ha hecho una revisión bibliográfica de los algoritmos que se emplean en el diseño de estas redes, entre los que se sobresalen los métodos de clustering, los incrementales o los evolutivos. Algunos de estos métodos se utilizan en mayor o menor medida en el algoritmo de diseño de redes de Funciones de Base Radial que se ha utilizado en esta memoria para solucionar el problema que nos hemos planteado.

2. En el capítulo 3 se ha expuesto un resumen de las características más importantes del proceso de reducción de mineral de Punta Gorda. Tras una detallada descripción de las etapas del proceso y de los factores que afectan el rendimiento del mismo, se ha formulado el problema a abordar con el objetivo de mejorar los niveles de consumo de portadores energéticos que genera el proceso de reducción de mineral en particular. Concretamente se ha considerado la necesidad de obtener un modelo del comportamiento de los hornos en función de las variables que intervienen en el proceso de reducción de mienral, y por lo tanto se ha planteado como un problema de aproximación funcional para la predicción a corto plazo con un número elevado de variables y un conjunto seleccionado de datos que, por otra parte resulta relativamente costoso de obtener.

3. En el Capítulo 4 se ha descrito el procedimiento que proponemos para la caracterización del comportamiento de algunos de los procesos relevantes que tienen lugar en la reducción de mineral de la tecnología CARON. Uno de los módulos que construyen el procedimiento se encarga de predecir lo que puede ocurrir a corto plazo utilizando para ello datos previamente extraídos del proceso industrial. El modelo de predicción se basa en una red neuronal de RBF, para cuya construcción se han aplicado diversos procedimientos que resuelven problemas relativos tanto a la selección de variables de entrada del modelo, como a la determinación de la estructura de la red RBF; asimismo se aborda como un problema de aproximación multiobjetivos en el que se busca obtener tanto un error de aproximación bajo como una complejidad reducida del mismo. Para la selección de variables se ha aplicado un procedimiento que se basa en el análisis de la varianza residual, y para la optimización multiobjetivo se aplica un algoritmo que tiene sus bases en algoritmos elitistas de de clasificación no dominada

4. En el Capítulo 5 se proporcionan y analizan los resultados experimentales obtenidos. El algoritmo utilizado aproxima con cierta rapidez y adecuada exactitud el *índice de consumo de petróleo en las cámaras*, de los hornos de reducción, que precisamente constituía el objetivo fundamental de nuestro trabajo. Con este procedimiento se puede tener una referencia significativa en la predicción de la cantidad de combustible que será utilizada en el proceso de reducción de mineral a partir de los datos empíricos medidos previamente. De esta forma se pone de manifiesto que el algoritmo desarrollado puede aplicarse casi sin cambios a la solución del problema que nos hemos planteado y donde la selección, el agrupamiento y limpieza de los datos utilizados ha sido decisiva a la hora de

llegar a esos resultados. El mejor resultado se ha obtenido tras haber aplicado un procedimiento de selección de variables, que permite reducir el espacio de búsqueda al reducir su dimensión. Luego de una valoración de compromiso entre los valores de aproximación obtenidos y el tiempo de ejecución empleado por el algoritmo, se ha llegado a un resultado final satisfactorio.

Por tanto y resumiendo, los principales aportes de esta tesis son los siguientes:

1. El desarrollo de una metodología que utiliza por primera vez técnicas de Softcomputing para solucionar problemas de predicción realistas planteados en la industria del Níquel en Cuba.

2. El análisis de las diferentes variables que se miden en los sistemas de supervisión y control de la planta de hornos de reducción con la finalidad de formular el problema de la *predicción del índice de consumo de petróleo en cámaras*.

3. El desarrollo, la aplicación y el análisis de resultados de un procedimiento de aproximación funcional híbrido, cuyos parámetros se han determinado mediante el uso de varias de las técnicas asociadas al SoftComputing, entre las que destacan los algoritmos evolutivos para aproximación multiobjetivo y la selección de variables.

4. Los resultados obtenidos han sido altamente satisfactorios, corroborando que la formulación del problema planteado ha sido enfocada de manera adecuada. Se logró obtener una aproximación funcional con un margen de error de un 2.1% aceptable para la operación de la industria.

5. Relacionado con el trabajo presentado se pueden citar las publicaciones descritas en el anexo A.

Entre la multitud de aplicaciones y problemas más directamente relacionados con el trabajo que se muestra en esta memoria que habría que abordarse en el futuro están:

1. La implementación práctica del procedimiento propuesto para el proceso de reducción de mineral de Punta Gorda, como parte de un modelo de predicción del comportamiento de la variable correspondiente al *índice de consumo de petróleo en cámaras*.

2. Incorporación al algoritmo desarrollado, de un procedimiento automático que utilice operadores genéticos para la selección de variables de entrada, discriminando aquellas que no aporten la suficiente información.

3. La paralelización del algoritmo propuesto, para hacerlo viable en el caso de que el elevado número de datos a procesar requieran tiempos de ejecución inaceptables dado los límites de tiempo real planteados.

4. La integración de los resultados prácticos que se obtengan dentro de un modelo de optimización dinámica de los niveles de consumo de combustible.

Este trabajo supone un punto de partida en la aplicación de técnicas de Softcomputing a los procesos industriales de la Planta de Níquel de Punta Gorda, ahora que se dispone de la tecnología de adquisición y almacenamiento necesarios para mantener las bases de datos de las variables que definen el funcionamiento de diferentes procesos.

Bibliografía

1. Alander, J. 2000. "Genetic algorithms and other natural optimizations methods to solve hard problems A tutorial review." Technical Report 96-1, Department of Information Technology and Production Economics, University of Vaasa, Finland.

2. Andersen, H, Tsoi, C, and (1993) . "A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm" . Complex Systems, vol. 7 no. 4 pp. 249-268.

3. Angeline, P. 1995. "Adaptive and self-adaptive evolutionary computation". In Palaniswami M., Attikiouzel Y. Marks R. J. Fogel D. and Fukuda T. editors Computational Intelligence A Dynamic System Perspective pp. 152-161. IEEE press.

4. Balakrishnan, K., Honavar, V., and (1995) . "Evolutionary design of neural architectures". Technical Report CS: TR: 95-01. Departament of computer science. Iowa State University, Ames, IA50011.

5. Beasly D, Bull D, Martin R (1993): "A sequential niche technique for multimodal function optimization."; in AnonymousEvolutionary Computation. pp 101-105.

6. Belew, R. and McInerney, J. and Schaudolph N. 1991 . "Evolving networks: using genetic algorithm with connectionist learning". Technical Report #CS90-171 (Revised) Computer Science & Engr. Dept. (C-014), Univ. of California at San Diego La Jolla CA 92093 USA February.

7. Bezdek JC1: Pattern recognition with fuzzy objective function algorithms. Plenum, New York.; in Anonymous

8. Broomhead DaLD1: "Multivariable functional interpolation and adaptive networks. Complex System. 2"; in Anonymouspp 321-355.

9. Bäck, T., Hammel, U., Schwefel, H., and (1997). Evolutionary computation: comments on the history and current state. IEEE Transactions on Evolutionary Computation . vol. 1 n. 1 April. Pp. 3-17.

10. Bäck, T. 1993. "Optimal mutation rates in genetic search" in Proc. 5th Int. Conf. on Genetic Algorithms. S. Forrest, Ed. San Mateo CA Morgan Kaufmann pp. 2-8.

11. Bäck, T. 1994. "Selective pressure in evolutionary algorithms: a characterization of selection mechanism" in Proc. 1st IEEE Conf. on Evolutionary Computation. Piscataway. NJ: IEEE Press, 1994 pp. 57-62.

12. Bäck, T. 1995. "Generalized convergence models for tournament and (,)-selection. In Eshelman, L. editor, Proceedings of the Sixth International Conference on Genetic Algorithms. , pp. 2-8 San Francisco CA. Morgan Kaufmann.

13. Bäck, T. 1996. Evolutionary Algorithms in Theory and Practice. Oxford University Press; New York, 1996.

14. Calvache Antonio 1: Historia y Desarrollo de la Minería en Cuba. Mariano Jimenez II and Mariano G. Jiménez (Abstract)

15. Castro Ruz Fidel (1996): Lineamientos Económicos III Congreso PCC. Informe Central del III Congreso del PCC (Abstract)

16. Cavalier T2: Página de enlaces sobre Optimization: http://www.personal.psu.edu/faculty/t/m/tmc7/tmclinks.html ; in Anonymous

17. Chen S, Chung E, Alkadhimi K, (1996) : "Regularized orthogonal least squares algorithm for constructing radial basis function networks".; in Anonymous Int. J. Contr., 64(5) . pp 829-837.

18. Chen S, Cowan C, Grant P, (1991) : "Orthogonal least squares learning algorithm for radial basis function networks"; in AnonymousIEEE Trans. Neural Networks, 2:302-309.

19. Chen S, Woo Y, Luk B, (1999) : "Combined genetic algorithms optimization and regularized orthogonal least squares learning for radial basis function networks."; in Anonymous. IEEE Trans. Neural Netwoks, 10(5):1239-1243.

20. Chien, C 1990. "Fuzzy logic in control system: fuzzy logic controller" . " IEEE transactions on system, man and cybernetics vol. 20 n. 2 pp 404-435 March-April .

21. Coello C., Van Veldhuizen D. Lamont G. 2002. Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publisher.

22. Cutler Peter 1: Nickel, Nickel Everywhere. Nickel Institute, Technical Information Centre (Abstract)

23. D. W. Marquardt. An algorithm for least-squares estimation of nonlinear inequalities. SIAM J. Appl. Math., 11:431{441, 1963.

24. Damas, M., Salmerón, M., Díaz, A. F., Ortega, J., and Prieto, A. 2000. "Genetic algorithms and Neuro-dynamic programming: application on water supply networks". Congress on Evolutionary Computation (CEC'2000). 7-14. San Diego (USA).

25. Dash M. and Liu H. "Feature selection for classification" Intelligent Data Analysis, Vol. 1,1997, pp. 131-156.

26. Davis, L. 1991. Handbook of genetic algorithms. New York: Van Nostrand Reinhold.

27. De Jong, K 1975. "An analysis of the behaviour of a class of genetic adaptive system" Ph. D. dissertation, Univ. of Michigan, Ann Arbor, Diss. Abstr. Int. 36(10), 5140B, University Microfilms n. 76-9381.

28. De Jong, K. A. and Spears W. M. 1991. An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms. In Schwefel, H. P. and Männer, R., editors, Proceedings of the First International Conference on Parallel Problem Solving from Nature, PPSN I, volume 496 of Lecture Notes in Computer Science, pages 3847, University of Dortmund. Springer-Verlag.

29. De la Herrán Gascón, Manuel 2003. Notas sobre Computación Evolutiva.

30. Dennis, J. E. and Schnabel, R. B. 1983. Numerical Methods for unconstrained optimization and nonlinear equations. Prentice Hall, Englewood Cliffs, N.J.

31. Dirección Técnica Empresa ECG 1: Operaciones en la Planta de Hornos de Reducción. Manual de Operaciones de Plantas (Abstract)

32. Duda, R. O, Hart, P. E, and (1973) . Pattern classification and scene análisis, Wiley, New York.

33. Eiben, A. , Hinderting, R., and Michalewicz, Z. 2000. "Parameter control in evolutionary computation". " IEEE Trans. on Evolutionary Computation, vol. 3 n. 2 pp. 124-141.

34. Elia Liiti¨ainen, Amaury Lendasse and Francesco Corona. 2003. Non-parametric Residual Variance Estimation in Supervised Learning. Helsinki University of Technology - Lab. of Computer and Information Science .

35. Eshelman, L., Caruna, R., Schaffer J., and (1989) . "Biases in the crossover

landscape" in Proc. 3rd Int. conf. on Genetic Algorithms. San Mateo, CA Morgan Kaufmann pp. 10-19.

36. Eshelman, L., Schaffer, J., and (1993) . Real-coded genetic algorithms and interval-schemata" in Foundations of Genetic Algorithms 2. San Mateo, CA: Morgan Kaufmann, pp. 187-202.

37. F. Herrera, M. Lozano and J. L. Verdegay. . Tackling real-coded genetic algorithms: operators and tools for the behavioural analysis . Arti¯cial Intelligence Reviews, 12(4):265{319, 1998.

38. Fahlman, S. and (1988) . "Faster-learning variations on back-propagation: an empirical study". In Touretzky, D. et al. editors Proc. of the 1988. Connectionist models summer school pp. 38-51. Morgan Kaufmann San Mateo CA. 1988.

39. Fogarty, T. 1989. "Varying the probability of mutation in the genetic algorithm". In J.D. Schaffer, editor, Proceedings of the 3rd . International Conference on Genetic Algorithms, pp 104-109. Morgan Kaufmann.

40. Fogel, D. 1993. in Proc. 2nd Annu. Conf. on Evolutionary Programming Society. 23-29.

41. Fogel, L. 1962. "Autonomous automata" Ind. Res. vol. 4 . 14-19.

42. Fonseca, C. , Fleming, P., and (1993) . "Genetic algorithms for multiobjetive optimization. In Forrest ed. Proceedings of the fifth international conference on genetic algorithms. Morgan Kaufmann.

43. Frean, M. and (1990) . "The upstart algorithm: a method for constructing and training feedforward neural networks," Neural Computation, vol. 2, pp. 198 - 209.

44. García, N. Hervás C. Muñoz J. 2002. "Multi-objetive cooperative coevolution of artificial neural networks". Neural networks, 15. 1259-1278.

45. Geman, S. Bienenstock E. and Doursat R. 1992. "Neural Networks and the Bias/Variance Dilemma", Neural Computation, 4, pp 1-58.

46. Gersho, A. and (1979) . "Asymtotically optimal block quatization" . IEEE Trans. Inf. Theory, 25(4). 373-380.

47. Ghost, J., Deuser, L., Beck, S., and (1992) . "A neural network based hybrid system for detection, characterization and classification of short-duration oceanic signals, IEEE Jl. Of Ocean Enginering, 17(4). 351-363.

48. Giordana, A. Neri F. 1996. "Search-intensive concept induction". Evolutionary Computation, 3(4):375-416.

49. Giordana, A Saitta L. and Zini F. 1994. "Learning disjunctive concepts by means of genetic algorithms". In Cohen, W et al., editors, Proceedings of the 11th International Conference on Machine Learning, pp. 96-104, Morgan Kaufmann, San Francisco, California.

50. Giovanni A. Cantor Monroy. Asignación de Parámetros en los Algoritmos Genéticos. Estado del Arte. Seminario de Investigación. 2007.

51. Goldberg, D., Richardson J., and (1987). "Genetic algorithms with sharing for multimodal function optimization. In Grefenstette (ed.), Proceedings of the Second International Conference on Genetic Algorithms . 41-49.

52. Goldberg, D. E. 1978. Genetic Algorithms. Addison Wesley, Reading, M.A.

53. Golub, G., Van Loan, C., and (1996) . Matriz computations. Johns Hopkins University Press, Baltimore, 3rd ed.

54. González, J and (2001) . Identificación y optimización de redes de funciones de base radial para aproximación funcional. Tesis doctoral. Dpto. Arquitectura y Tecnología de Computadores. Universidad de Granada.

55. González, J., Rojas, I., Pomares, H., Ortega, J., Prieto, A., and (2002) . "A new clustering technique for function approximation" IEEE Trans. Neural Networks. Vol. 13. n.1 Januray. 132-142.

56. González, J. Rojas I. Ortega J. Pomares H. Fernández F. and Díaz A. 2003. 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.

57. Grönroos, M. and (1998) . Evolutionary design of neural networks. Master of Science Thesis. University of Turku.

58. Guillén, A. González J. Rojas I. Pomares H. Herrera L. Valenzuela O. and Prieto A. 2007a. Improving Clustering Technique for Functional Approximation Problem Using Fuzzy Logic: ICFA algorithm. Neurocomputing, DOI:10.1016/j.neucom.2006.06.017.

59. Guillén, A. González J. Rojas I. Pomares H. Herrera L. Valenzuela O. and Rojas F. 2007b. Output Value-Based Initialization For Radial Basis Function Neural Networks. Neural Processing Letters, 25(3): 209-225, DOI:10.1007/s11063-007-9039-8.

60. Guillén, A. González J. Rojas I. Pomares H. Herrera L. Valenzuela O. and Rojas F. 2007c. Studying Possibility in a Clustering for Function Approximation Algorithm. Neural Computing & Applications.

61. Guillen, I. Rojas J. Gonzalez H. Pomares L. J. Herrera O. Valenzuela and A. Prieto. Improving Clustering Technique for Functional Approximation

Problem Using Fuzzy Logic: ICFA algorithm. Lecture Notes in Computer Science, 3512:272{280, June 2005.

62. Guillen, I. Rojas J. Gonzalez H. Pomares L. J. Herrera O. Valenzuela and A. Prieto. A possibilistic approach to rbfn centers initialization. Lecture Notes in Computer Science, 3642:174{183, 2005.

63. Harp, S, Samad, T, Guha, A, and (1989) . "Towards the genetic synthesis of neural etworks". in Proc. of the Third Int'l Conf. on Genetic Algorithms and Their Applications (J. D. Schaffer, ed. pp. 360--369 Morgan Kaufmann San Mateo CA.

64. Hartigan, J. A. and (1975) . Clustering Algorithms. Willey, New York, 1975.

65. Haykin, S. and (1999) . Neural Networks. A comprehensive foundation, 2nd. Upper Saddle River, NJ: Prentice-Hall.

66. Herrera Francisco. Introducción a los Algoritmos Metaheurísticos. Grupo de Investigación "Soft Computing and Intelligent Information Systems" Dpto. Ciencias de la Computación e I.A. Universidad de Granada 18071 – ESPAÑA herrera@decsai.ugr.es http://sci2s.ugr.es .

67. Hindertin, R., Michalewicz Z., Peachey, T., and (1996) . Self-adaptive genetic algorithm for numeric functions. In Voight et al. Proceedings of fourth Conference on Parallel Problem Solving from Nature. , number 1141 in LNCS. Springer Berlin.

68. Holland, J. 1975. Adaptation in natural and artificial system. Ann Arbor, MI: Univ. of Michigan Press.

69. Holland, J. 1978. Cognitive system based on adaptive algorithms. . In Waterman, D. et al. editors Pattern-Directed Inferenced System. Academic Press New York.

70. Horn, J. 1997. Multicriteria Decision Making. In (Bäck et al., 1997a).

71. J. Zhang and Y. Leung. Improved possibilistic C{means clustering algorithms. IEEE Transactions on Fuzzy Systems, 12:209{217, 2004).

72. Jacobs, D. A. and (1977) . The state of the art in numerical analysis, chapter III. Academic Press, London.

73. Janikow, C. and Michalewicz, Z 1991. "An experimental comparison of binary and floating point representation in genetic algorithms" . in Proc. 4th Int. Conf. on Genetic Algorithms. San Mateo, CA Morgan Kaufman pp. 31-36.

74. John G. H., Kohavi R. and Pfleger P. "Irrelevant features and the subset selection problem". Proceedings of the Eleventh International Conference

on Machine Learning. New Brunswick, Morgan Kaufmann, 1994, pp. 121-129.

75. Kadirkamanathan, V. and (1993) . A function estimation approach to sequential learning with neural networks. Neural computation, 5, 954-975.

76. Kalyanmoy Deb, Samir Agrawal Amrit Pratap and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolutionary Computation, 6(2):182{197, 2002.

77. Kanjilal, P., Banerjee, D., and (1995) . "On the application of orthogonal transformation for the design and analysis of feedforward networks. IEEE Trans. neural networks, 6(5): 1061-1070.

78. Karayiannis, N. B and Mi, G. W. 1997. "Growing radial basis networks: merging supervised and unsupervised learning with networks growth techniches. IEEE Trans. Neural Networks, 8(6):1492-1506, Nov.

79. Kowalski, J., Hartman, E., and Keeler, J. 1990. "Layered neural networks with Gaussian hidden units as universal approximators" Neural Computation, 2: 210-215.

80. Kursawe, F. and (1995) . "Toward self-adapting evolution strategies" in Proc. 2nd IEEE Conf. Evolutionary Computation, Perth, Australia. Piscataway, NJ IEEE Press pp. 283-288.

81. L.C. Molinay, Ll. Belanchey A. Neboty 2001. Caracterización de Algoritmos de Selección de Variables. Proyecto CICyT TAP. Universidad de Cataluña.

82. Langdon W.B. PR2: An introduction to Genetic Algorithms. MIT Press. Foundations of Genetic Programming. Springer Mitchell M. (1998) (Abstract)

83. Lee, S. and and Rhee, M. 1988. "Multilayer feedforward potential function network" In Proceedings of the Second International Conference on Neural Networks. 161-171.

84. Light, W. 1992. "Some aspects of radial basis function approximation. In S.P. Singh, editor, Approximation Theory, Spline Functions and Applications. 163-190.

85. Lotfi A. Zadeh. Nacimiento y Evolución de la Lógica Borrosa, el Softcomputing y la Computación sin palabras: Un punto de vista personal. * Texto del discurso presentado para la recepción del Doctorado Honoris Causa por la Facultad de Ciencias de la Universidad de Oviedo, España 1-XII-1995. Psicothema, 1996. Vol. 8, nº 2, pp. 421-429 ISSN 0214 - 9915 CODEN PSOTEG 421 Universidad de California .

86. M. Dolores Perez Godoy, Antonio J. Rivera M. Jose del Jesús, Ignacio Rojas.

Optimización de CoEvRBF para aumentar su eficiencia en tareas de clasificación.

87. Mandani, E. and Assilian, S. 1975. "An experiment in linguistic synthesis with a fuzzy controller" Int. J. Mach Studies vol. 7 n. 1 . 1-13.

88. Marquetti Nodarse H2: Cuba: proceso de reestructuración y recuperación de la Industria del Níquel. Oficina Nacional de Estadísticas (2001): Anuario Estadístico de Cuba 2000 (Abstract).

89. McCulloch, W. and Pitts W. 1943. "A logical calculus of the ideas immanent in nervous activity". Bulletin of Mathematical Biophysics, 7. pp115-133.

90. Megdassy, P. 1961. Decomposition of superposition of distributed functions. Hungarian Academy of Sciences, Budapest.

91. Micchelli, C. 1986. "Interpolation of scattered data: distance matrices and conditionally positive definite functions" Constructive Approximation, 2: 11-22.

92. Michalewicz, Z., Krawczyk, J., Kazemi, M., and Janikow, C 1990. "Genetic algorithms and optimal control problems". In proceedings of the 29th IEEE Conference on Decision an Control, pp 1664-1666. IEEE Computer Society Press.

93. Moller, M. and (1993) . "A scaled conjugate gradient algorithm for fast supervised learning". Neural Networks, 6 pp. 525-533.

94. Moody, J and Darken, C. 1989. "Fast learning networks of locally-tuned processing units," Neural Computation, vol. 3 n. 4 . 579-588.

95. Moriarty, D. Miikkulainen R. 1997. Forming neural networks through efficient and adaptive coevolution. Evolutionary computation, 5(4): 373-399.

96. Musavi, M., Ahmed, W., Chan, K., Faris, K., Hummels, D., and (1992) . "On the training of radial basis functions classifiers" Neural Networks, 5(4). 595-603.

97. N.M. Borjas 2: Ore Variability Influence on the Metallurgical Process Punta Gorda Nickel Plant. Laterite Nickel 2004 (Abstract)

98. N. R. Pal, K. Pal and J. C. Bezdek. A Mixed C{Means Clustering Model. In Proceedings of the 6th IEEE International Conference on Fuzzy Systems (FUZZIEEE'97), volume 1, pages 11{21, Barcelona, July 1997.

99. Nix, A., Vose, M., and (1992) . "Modelling genetic algorithms with Markov chains" . Ann. Math. Artif. Intell. vol. 5, pp. 79-88.

100. Orr, M. and (1993) . "Regularized center recruitment in radial basis function

networks". Technical report 59. Center for cognitive science. University of Edinburgh.

101. Orr, M. and (1996) . "Introduction to radial basis function networks". Technical report. Center for cognitive science. University of Edinburgh.

102. Park, J., Sandberg, I., and (1991) . "Universal approximation using radial basis function networks". Neural Computation, 3(2):246-257.

103. Park, J., Sandberg, I., and (1993) . "Universal approximation and radial basis function networks". Neural Computation, 5(2):305-316.

104. Pedrycz, W. and (1998) . "Conditional fuzzy clustering in the design of radial basis function neural networks". IEEE Trans. Neural Networks. 9(1):601-612.

105. Pelikan, M. and Lobo F. 1999. Parameter-less Genetic Algorithm: A Worst-case Time and Space Complexity Analysis. Technical Report IlliGAL 99014, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

106. Platt, J. and (1991) . "A resource-allocating network for function interpolation", Neural Computation 3, 213-225.

107. Poggio, T. and Girosi. F. (1990) . "Networks for approximation and learning. Proc. IEEE, 78(9). 1481-1497.

108. Polak, E. and (1971) . Computational methods in optimization. Academic Press. New York.

109. Potter, M. De Jong K 2000. "Cooperative Coevolution: an architecture for evolving coadapted subcomponents. Evolutionary Computation. 8(1), 1-29.

110. Powell, M. 1985. "Radial basis functions for multivariable interpolation: A review". In IMA. Conf. on Algorithms for the approximation of functions and data, pp. 143-167.

111. Prados, D. 1992. Training multilayered neural networks by replacing the least fit hidden neurons. In Proc. of IEEE SOUTHEASTCON'92, vol. 2 pp. 634-637. IEEE Press New York NY.

112. Press, W., Vettering, W., Teukolsky, S., Flannery, B., and (1994) . Numerical recipes in C. Cambridge University Press, 2nd edition.

113. Procyk, R. and Mandani E.(1979) . "A linguistic self-organizing process controller" Automat. vol 15 n. 1, pp 15-30.

114. Rechenberg, I. 1973. Evolutionsstrategies: Optimierung technischer systeme nach prinzipien der biologischen evolution. Sttutgart, Germany.

:Frommann-Holzboog.

115. Renders, J., Flasse, S., and (1996) . "Hybrid methods using genetics algorithms for global optimization" IEEE Trans. Syst., Man, Cybern. B, vol. 26 n. 2, pp. 243-258.

116. Rivas, V., Castillo, P., Merelo, J., and (2001) . "Evolving RBF Networks" In Mira, J. and Prieto, A. (eds): IWANN 2001, LNCS 2084. 506-513.

117. Rivera, AJ. 2003. Diseño y Optimización de Redes RBF mediante técnicas Bioinspiradas: Tésis Doctoral, ETS Ingeniería Informática, UGR.

118. Rivera Rivas, A 2003. Identificación y optimización de redes de funciones de base radial para aproximación funcional. Tesis doctoral. Dpto. Arquitectura y Tecnología de Computadores. Universidad de Granada.

119. Rojas, I., Pomares, H., Ortega, J., Prieto, A., and (2000a) . "Self-Organized fuzzy system generation from training examples". IEEE Trans. On Fuzzy System, vol.8. n. 1. 23-36.

120. Rojas, I., Valenzuela, O., and Prieto, A. 1997. "Statistical Analysis of the Main Parameters in the Definition of Radial Basic Function Networks", Lecture Notes in Computer Science, Vol.1240. 882-891.

121. Rosipal, R., Koska, M., Farkaš, I., and (1998). "Prediction of chaotic time series with a resource allocating rbf networks". Neural Processin Letters, 7:185-197.

122. Runkler, T. A. and Bezdek, J. C. 1999. "Alternating cluster estimation: A new tool for clustering and function approximation. IEEE Trans. Fuzzy System, 7(4):377-393, Aug.

123. Russo, M., Patané, G., and (1999) . "Improving the LBG alorithm", volume 1606 of Lecture Notes on Computer Science. pp. 621-630. ISSN: 0302-9743, Springer-Verlag 1999.

124. Salmerón, M., Ortega, J., Puntonet, C., Prieto, A., and (2001) . "Improved RAN sequential prediction using orthogonal techniques. Neurocomputing. In press. Paper code Neucom .

125. Schaffer D. (1984). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. PhD thesis, Vanderbilt University.

126. Schwefel, H 1975. Evolutionsstrategie und numerische optimierung dissertation. Technische Universität Berlin, Germany. May .

127. Smith, J., Fogarty T., and (1996) . "Adaptively parameterised evolutionary system: Self adaptive recombination and mutation in a genetic algorithm.

128. Spears, W. and (1995) . "Adapting crossover in evolutionary algorithms. In McDonnell et al. Proceedings of the Fourth Annual Conference on Evolutionary Programing pp. 367-384.

129. Srinivas, N., Dev, K., and (1995) . Multiobjetive optimization using nondominated sorting in genetic algorithms. Evolutionary computation, 2(3):221-248.

130. Sugeno, M. and Kang, G. 1988. "Structure identification of fuzzy model" Fuzzy Sets and System, vol. 28 pp. 15-33.

131. Syswerda, G. 1989. "Uniform crossover in genetic algorithms" in Proc. 3rd Int. Conf. on Genetic Algorithms. San Mateo, CA: Morgan Kaufman. pp. 2-9.

132. Tsukamoto, Y. 1979. "An approach to fuzzy reasoning method" . in Advances in Fuzzy Set Theory and Applications , Madan M. Gupta Rammohan K. Ragade and Ronald R. Yager Eds. Amsterdam North-Holland pp. 137-149.

133. Uhr, L. and (1973) . Pattern Recognition, Learning, and Thought. Englewood Cliffs, NJ: Prentice Hall.

134. Valenzuela Gaete Ricardo. Aplicaciones de Soft Computing al análisis de ficheros los de sitios web. Editorial Universidad de Granada ISBN 84-338-4166-1.

135. Whitehead, B, Choate, T., and (1996) . IEEE Trans. on Neural Networks, Vol.7, No.4. 869-880.

136. Wolpert, D. H. ed. 1995b. The Mathematics of Generalization: The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning, Santa Fe Institute Studies in the Sciences of Complexity, Volume XX, Reading, MA: Addison-Wesley.

137. Yao, X 1999. "Evolutionary artificial neural networks" Proceedings of the IEEE, 87 (9): 1423-1447 September.

138. Yingwei, L., Sundarajan, N., Saratchandran, P., and (1997) . "A sequential learning scheme for function approximation using minimal radial basis function neural networks" Neural Computation, 9:361-478.

139. Zadeh, L. 1994. "Fuzzy logic and soft computing: issues, contentions and perspectives". In Iizuka 94: 3rd International conference on fuzzy logic, neural nets and soft computing pp 1-2 Iizuka Japan.

140. Zhang, J. and Leung Y. 2004. Improved possibilistic Cmeans clustering algorithms. IEEE Transactions on Fuzzy Systems, 12:209217.

Anexos

A. Publicaciones

B. Flujo tecnológico de la planta de Reduccion de Mineral

C. Modelo del horno usado en el planteamiento del problema

D. Resultados de los Experimentos

E. Datos del Autor

A

Publicaciones

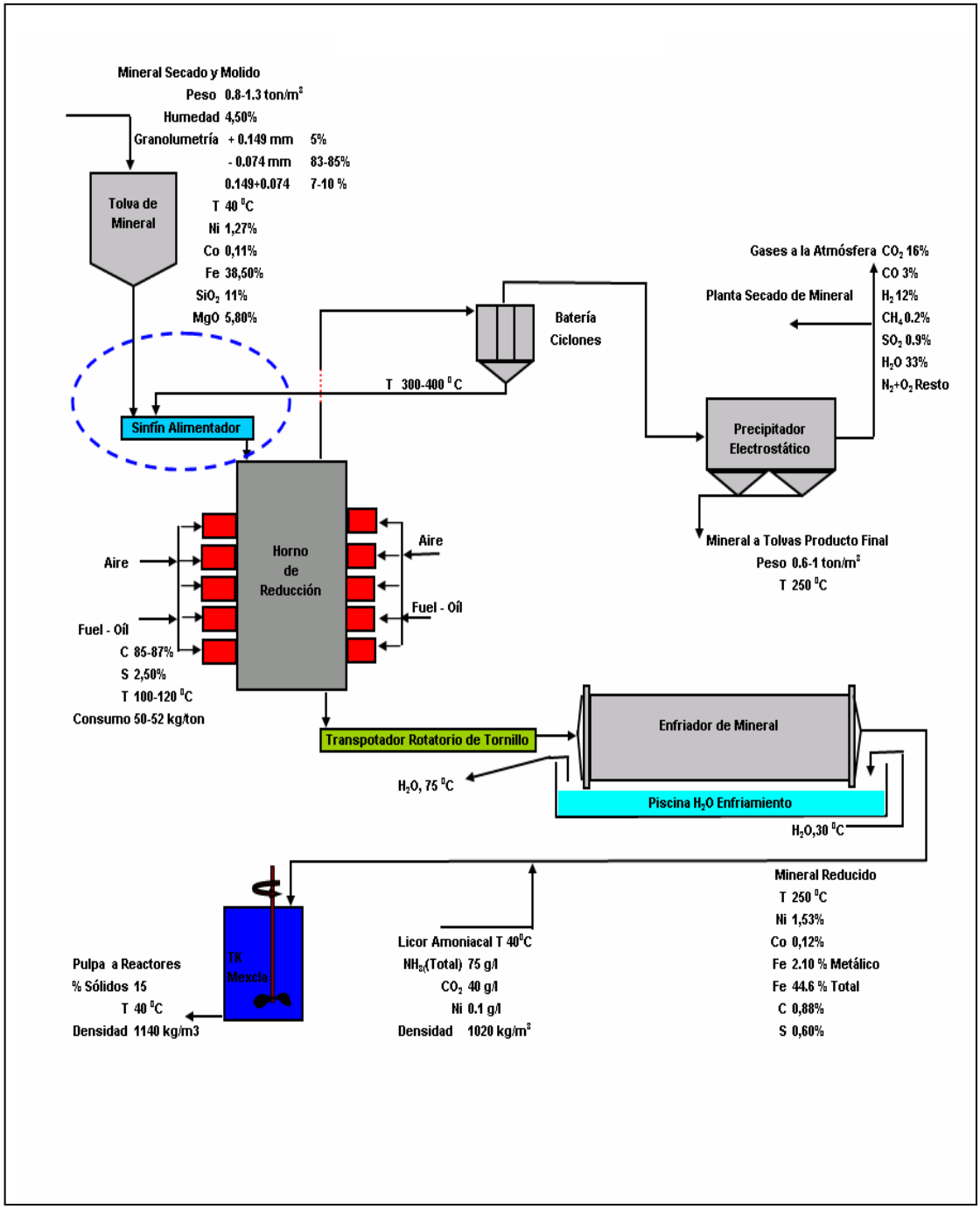
1. Multiobjective RBFNNs Designer for Function Approximation: An Application for Mineral Reduction. *Second International Conference on Natural Computation (ICNC'06) and the Third International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'06).*

2. Multiobjective RBFNNs Designer for Function Approximation: An Application for Mineral Reduction . *Lecture Notes in Computer Science, 4221: 511-520, 2006.*

3. A New Multiobjective RBFNNs Designer and Feature Selector for a Mineral Reduction Application Reduction . *2007 IEEE International Conference on Fuzzy Systems.*

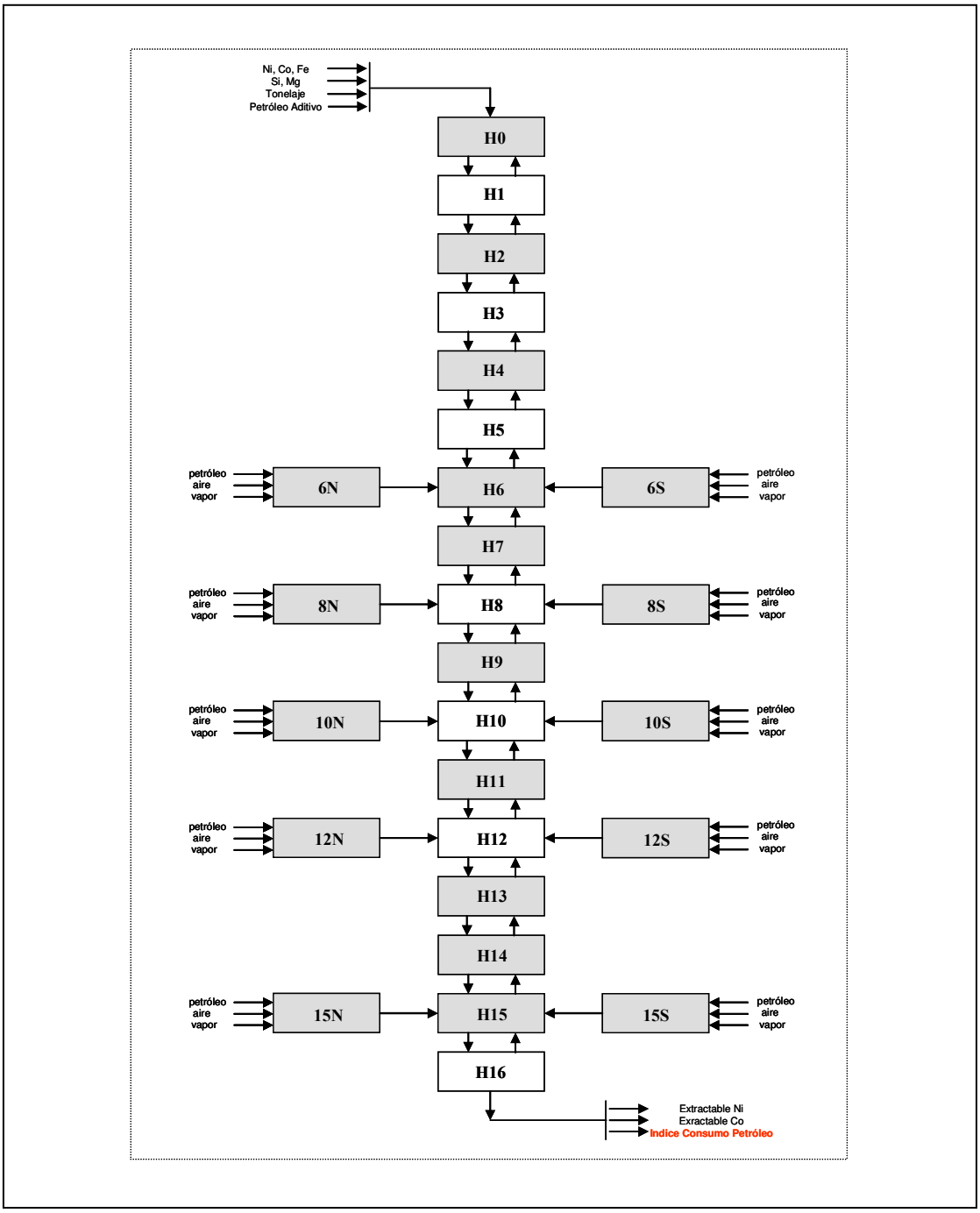
B

Flujo Tecnológico, Reducción Mineral



C

Modelo del Horno de Reducción



D

Resultados Experimentales

Como se ha comentado en el capítulo 5, la mejor aproximación se corresponde con el experimento ***7B***. El resumen que se muestra corresponde a los resultados que son considerados como los más aceptables para la industria.

Parámetros		Exp-7A	Exp-7B	Exp-7C	Exp-7D
cruce/mut.		0,8/0,035	0,8/0,035	0,8/0,035	0,8/0,035
pob/gen		30_50	70_90	90_120	120_220
error test		0,905	0,890	0,906	0,921
erro train		0,956	0,937	0,948	0,959
seg		475,21	627,21	1018,80	1654,80
min		7,92	10,45	16,98	27,58
	real	aprox.	aprox.	aprox.	aprox.
media	53,75	53,69	53,76	53,72	53,69
relación		100,12	99,98	100,07	100,12
máximo	59,58	57,96	57,43	56,70	57,25
mínimo	49,06	51,53	48,64	50,85	51,18
desvest	2,20	0,95	1,03	0,92	0,91

Las tablas que se relacionan a continuación muestran los valores obtenidos por la aproximación realizada; correspondiendo con el juego de datos que se han utilizado para el test. Los parámetros indicados en la tabla han sido:

1. Valores real, aproximado y diferencia de las variables.
2. Relación entre los valores real y aproximado de las variables.
3. Máximo y mínimo de los valores real y aproximado de las variables.
4. Desviación estándar real y aproximada de las variables.

	Exp-7A		Exp-7B		Exp-7C		Exp-7D	
real	aprox.	dif.	aprox.	dif.	aprox.	dif.	aprox.	dif.
53,75	53,69		53,76		53,72		53,69	
100,00	100,12		99,98		100,07		100,12	
59,58	57,96		57,43		56,70		57,25	
49,06	51,53		48,64		50,85		51,18	
2,20	0,95		1,03		0,92		0,91	
49,06	52,49	-3,43	52,49	-3,43	52,78	3,72	52,65	3,59
49,06	52,43	-3,37	52,44	-3,38	52,74	-3,68	52,59	-3,53
49,16	53,56	-4,40	53,75	-4,59	53,81	-4,65	53,65	-4,49
49,16	53,50	-4,34	53,69	-4,53	53,77	-4,61	53,59	-4,43
49,16	53,50	-4,34	53,69	-4,53	53,77	-4,61	53,59	-4,43
49,22	54,11	-4,89	53,72	-4,50	54,04	-4,82	54,43	-5,21
49,38	52,75	-3,37	53,29	-3,91	53,27	-3,89	52,85	-3,47
49,38	52,82	-3,44	51,04	-1,66	53,06	-3,68	52,79	-3,41
49,38	52,93	-3,55	51,07	-1,69	53,16	-3,78	52,87	-3,49
49,49	51,94	-2,45	52,59	-3,10	52,34	-2,85	52,04	-2,55
49,66	54,74	-5,08	54,29	-4,63	55,35	-5,69	55,39	-5,73
49,66	54,51	-4,85	54,16	-4,50	54,76	-5,10	55,14	-5,48
49,68	53,04	-3,36	53,97	-4,29	53,52	-3,84	53,06	-3,38
49,68	53,07	-3,39	53,99	-4,31	53,59	-3,91	53,09	-3,41
49,68	53,11	-3,43	54,04	-4,36	53,74	-4,06	53,13	-3,45
49,68	53,00	-3,32	53,95	-4,27	53,45	-3,77	53,03	-3,35
49,70	51,87	-2,17	48,76	0,94	52,43	-2,73	52,06	-2,36
49,70	51,68	-1,98	48,64	1,06	52,20	-2,50	51,87	-2,17
49,74	53,75	-4,01	51,82	-2,08	53,28	-3,54	53,74	-4,00
49,88	52,06	-2,18	52,13	-2,25	52,77	-2,89	52,49	-2,61
49,94	53,39	-3,45	50,42	-0,48	50,85	-0,91	53,38	-3,44
49,94	53,14	-3,20	50,08	-0,14	50,93	-0,99	53,13	-3,19
49,97	53,47	-3,50	53,50	-3,53	53,47	-3,50	53,44	-3,47
50,05	54,11	-4,06	54,49	-4,44	55,04	-4,99	54,47	-4,42
50,09	54,04	-3,95	53,80	-3,71	53,89	-3,80	54,20	-4,11
50,30	54,42	-4,12	54,31	-4,01	54,05	-3,75	54,54	-4,24
50,30	54,35	-4,05	54,25	-3,95	54,08	-3,78	54,46	-4,16
50,34	52,46	-2,12	52,66	-2,32	52,82	-2,48	52,71	-2,37
50,34	52,47	-2,13	52,69	-2,35	52,93	-2,59	52,74	-2,40
50,34	52,39	-2,05	52,61	-2,27	52,65	-2,31	52,65	-2,31
50,43	52,26	-1,83	51,97	-1,54	52,77	-2,34	52,88	-2,45
50,49	52,42	-1,93	53,14	-2,65	52,19	-1,70	52,41	-1,92
50,49	52,39	-1,90	53,11	-2,62	52,21	-1,72	52,37	-1,88
50,51	53,48	-2,97	53,61	-3,10	54,36	-3,85	54,07	-3,56
50,56	53,19	-2,63	53,34	-2,78	52,83	-2,27	53,18	-2,62
50,60	54,54	-3,94	55,03	-4,43	54,67	-4,07	54,90	-4,30
50,60	54,31	-3,71	54,84	-4,24	54,50	-3,90	54,65	-4,05
50,64	53,25	-2,61	53,11	-2,47	53,90	-3,26	53,23	-2,59
50,64	53,23	-2,59	53,08	-2,44	53,93	-3,29	53,21	-2,57
50,64	53,04	-2,40	52,85	-2,21	54,01	-3,37	53,01	-2,37
50,64	53,32	-2,68	53,17	-2,53	53,71	-3,07	53,27	-2,63

50,70	52,77	-2,07	53,23	-2,53	52,53	-1,83	52,70	-2,00
50,76	53,85	-3,09	53,86	-3,10	54,06	-3,30	53,90	-3,14
50,77	52,53	-1,76	53,04	-2,27	52,74	-1,97	52,53	-1,76
50,80	53,80	-3,00	53,73	-2,93	54,20	-3,40	53,60	-2,80
50,80	53,96	-3,16	53,85	-3,05	54,05	-3,25	53,75	-2,95
50,89	53,52	-2,63	51,89	-1,00	54,17	-3,28	53,34	-2,45
50,89	53,31	-2,42	51,69	-0,80	54,38	-3,49	53,14	-2,25
50,89	53,43	-2,54	52,75	-1,86	53,74	-2,85	53,60	-2,71
50,90	52,33	-1,43	52,81	-1,91	51,50	-0,60	51,34	-0,44
50,90	52,42	-1,52	52,87	-1,97	51,64	-0,74	51,36	-0,46
50,90	52,12	-1,22	52,67	-1,77	51,26	-0,36	51,18	-0,28
51,07	53,41	-2,34	53,38	-2,31	53,36	-2,29	53,36	-2,29
51,07	53,46	-2,39	53,42	-2,35	53,32	-2,25	53,41	-2,34
51,07	53,40	-2,33	53,36	-2,29	53,25	-2,18	53,23	-2,16
51,07	53,46	-2,39	53,41	-2,34	53,21	-2,14	53,29	-2,22
51,07	53,46	-2,39	53,41	-2,34	53,21	-2,14	53,29	-2,22
51,13	53,80	-2,67	54,02	-2,89	53,04	-1,91	53,18	-2,05
51,13	54,00	-2,87	53,94	-2,81	53,01	-1,88	53,19	-2,06
51,15	52,91	-1,76	53,30	-2,15	51,30	-0,15	52,72	-1,57
51,17	52,86	-1,69	49,69	1,48	52,88	-1,71	52,92	-1,75
51,26	52,38	-1,12	53,53	-2,27	52,66	-1,40	52,62	-1,36
51,26	52,56	-1,30	53,71	-2,45	53,03	-1,77	52,84	-1,58
51,26	52,49	-1,23	53,62	-2,36	52,82	-1,56	52,75	-1,49
51,26	52,34	-1,08	52,02	-0,76	52,28	-1,02	52,43	-1,17
51,26	52,26	-1,00	51,95	-0,69	52,16	-0,90	52,35	-1,09
51,26	52,35	-1,09	52,03	-0,77	52,31	-1,05	52,45	-1,19
51,31	52,75	-1,44	52,04	-0,73	53,10	-1,79	52,71	-1,40
51,31	52,65	-1,34	51,95	-0,64	52,74	-1,43	52,58	-1,27
51,33	53,45	-2,12	53,92	-2,59	53,63	-2,30	52,84	-1,51
51,42	53,70	-2,28	53,61	-2,19	53,79	-2,37	54,03	-2,61
51,43	53,86	-2,43	54,21	-2,78	54,20	-2,77	53,89	-2,46
51,43	54,06	-2,63	54,36	-2,93	54,14	-2,71	54,09	-2,66
51,44	52,82	-1,38	53,12	-1,68	53,94	-2,50	52,79	-1,35
51,44	53,06	-1,62	53,30	-1,86	53,59	-2,15	53,02	-1,58
51,44	52,88	-1,44	53,17	-1,73	53,85	-2,41	52,85	-1,41
51,47	54,39	-2,92	53,79	-2,32	54,19	-2,72	54,34	-2,87
51,54	52,76	-1,22	52,26	-0,72	53,85	-2,31	52,64	-1,10
51,54	52,84	-1,30	52,31	-0,77	53,68	-2,14	52,72	-1,18
51,54	54,06	-2,52	53,02	-1,48	54,02	-2,48	53,64	-2,10
51,54	53,85	-2,31	52,87	-1,33	53,52	-1,98	53,41	-1,87
51,57	53,10	-1,53	52,87	-1,30	51,98	-0,41	52,89	-1,32
51,57	52,85	-1,28	52,69	-1,12	52,20	-0,63	52,63	-1,06
51,57	53,08	-1,51	52,86	-1,29	52,03	-0,46	52,87	-1,30
51,57	53,00	-1,43	52,80	-1,23	52,13	-0,56	52,78	-1,21
51,60	53,77	-2,17	54,60	-3,00	53,79	-2,19	54,28	-2,68
51,62	53,37	-1,75	52,17	-0,55	53,30	-1,68	53,44	-1,82
51,62	53,27	-1,65	52,07	-0,45	53,19	-1,57	53,33	-1,71
51,65	53,72	-2,07	53,94	-2,29	53,92	-2,27	54,40	-2,75
51,65	54,21	-2,56	54,10	-2,45	54,56	-2,91	54,86	-3,21
51,65	53,91	-2,26	54,00	-2,35	54,13	-2,48	54,58	-2,93

51,65	53,26	-1,61	53,47	-1,82	53,09	-1,44	53,00	-1,35
51,70	52,09	-0,39	52,69	-0,99	51,92	-0,22	52,14	-0,44
51,70	52,13	-0,43	52,71	-1,01	52,02	-0,32	52,17	-0,47
51,82	54,09	-2,27	53,70	-1,88	54,69	-2,87	54,57	-2,75
51,82	54,17	-2,35	53,74	-1,92	54,80	-2,98	54,65	-2,83
51,86	55,64	-3,78	55,41	-3,55	55,58	-3,72	55,24	-3,38
51,86	55,43	-3,57	55,21	-3,35	54,98	-3,12	54,99	-3,13
51,86	55,51	-3,65	55,24	-3,38	55,06	-3,20	55,04	-3,18
51,86	55,66	-3,80	55,33	-3,47	55,32	-3,46	55,17	-3,31
51,87	54,01	-2,14	53,71	-1,84	53,63	-1,76	53,68	-1,81
51,87	54,06	-2,19	53,76	-1,89	53,71	-1,84	53,76	-1,89
51,87	52,29	-0,42	52,63	-0,76	52,56	-0,69	52,40	-0,53
51,87	52,30	-0,43	52,64	-0,77	52,60	-0,73	52,42	-0,55
51,92	54,03	-2,11	54,60	-2,68	54,26	-2,34	54,09	-2,17
51,93	54,19	-2,26	54,57	-2,64	54,63	-2,70	54,31	-2,38
52,00	52,78	-0,78	52,78	-0,78	53,40	-1,40	52,53	-0,53
52,00	52,85	-0,85	52,84	-0,84	53,28	-1,28	52,59	-0,59
52,01	54,27	-2,26	54,57	-2,56	54,01	-2,00	54,57	-2,56
52,03	54,55	-2,52	53,97	-1,94	55,22	-3,19	55,33	-3,30
52,07	52,00	0,07	53,02	-0,95	52,60	-0,53	52,27	-0,20
52,11	54,11	-2,00	54,68	-2,57	54,51	-2,40	54,20	-2,09
52,16	53,36	-1,20	53,22	-1,06	52,59	-0,43	53,19	-1,03
52,16	53,45	-1,29	53,28	-1,12	52,47	-0,31	53,28	-1,12
52,18	53,34	-1,16	53,68	-1,50	53,32	-1,14	53,35	-1,17
52,18	53,41	-1,23	53,76	-1,58	53,63	-1,45	53,39	-1,21
52,18	53,77	-1,59	53,80	-1,62	53,74	-1,56	53,74	-1,56
52,18	53,79	-1,61	53,84	-1,66	53,80	-1,62	53,78	-1,60
52,18	53,67	-1,49	53,70	-1,52	53,61	-1,43	53,62	-1,44
52,18	53,00	-0,82	53,57	-1,39	53,33	-1,15	53,18	-1,00
52,22	54,35	-2,13	54,37	-2,15	54,48	-2,26	54,80	-2,58
52,27	54,74	-2,47	54,48	-2,21	54,90	-2,63	54,34	-2,07
52,27	54,14	-1,87	54,20	-1,93	54,09	-1,82	53,93	-1,66
52,37	52,87	-0,50	53,10	-0,73	52,90	-0,53	52,72	-0,35
52,37	52,90	-0,53	53,13	-0,76	52,89	-0,52	52,76	-0,39
52,41	52,15	0,26	53,70	-1,29	52,50	-0,09	53,03	-0,62
52,41	52,34	0,07	53,87	-1,46	53,01	-0,60	53,15	-0,74
52,41	52,23	0,18	53,76	-1,35	52,67	-0,26	53,12	-0,71
52,47	54,40	-1,93	53,86	-1,39	54,24	-1,77	54,34	-1,87
52,51	53,69	-1,18	52,90	-0,39	53,59	-1,08	53,32	-0,81
52,51	53,69	-1,18	52,90	-0,39	53,59	-1,08	53,32	-0,81
52,58	53,53	-0,95	53,32	-0,74	53,05	-0,47	53,55	-0,97
52,59	53,90	-1,31	53,95	-1,36	54,12	-1,53	53,93	-1,34
52,71	53,07	-0,36	52,07	0,64	51,44	1,27	52,80	-0,09
52,72	53,81	-1,09	54,24	-1,52	53,95	-1,23	53,85	-1,13
52,72	53,58	-0,86	54,00	-1,28	53,85	-1,13	53,61	-0,89
52,72	53,73	-1,01	54,15	-1,43	53,91	-1,19	53,76	-1,04
52,72	53,66	-0,94	54,08	-1,36	53,88	-1,16	53,69	-0,97
52,72	53,73	-1,01	54,15	-1,43	53,91	-1,19	53,76	-1,04
52,77	53,77	-1,00	53,64	-0,87	54,10	-1,33	53,92	-1,15
52,85	53,50	-0,65	53,63	-0,78	53,16	-0,31	53,49	-0,64

52,85	53,39	-0,54	53,55	-0,70	53,25	-0,40	53,37	-0,52
52,85	53,36	-0,51	53,52	-0,67	53,27	-0,42	53,34	-0,49
52,85	53,61	-0,76	53,76	-0,91	53,00	-0,15	53,64	-0,79
52,87	52,48	0,39	52,02	0,85	53,08	-0,21	52,37	0,50
52,87	52,42	0,45	51,87	1,00	53,11	-0,24	52,31	0,56
52,87	52,46	0,41	51,98	0,89	53,09	-0,22	52,35	0,52
53,00	53,15	-0,15	54,97	-1,97	54,17	-1,17	53,64	-0,64
53,00	53,11	-0,11	54,94	-1,94	54,10	-1,10	53,61	-0,61
53,00	52,53	0,47	53,20	-0,20	52,29	0,71	52,40	0,60
53,00	52,58	0,42	53,25	-0,25	52,34	0,66	52,46	0,54
53,06	54,03	-0,97	55,56	-2,50	53,78	-0,72	54,13	-1,07
53,06	53,88	-0,82	55,34	-2,28	53,81	-0,75	53,97	-0,91
53,09	53,21	-0,12	52,68	0,41	53,26	-0,17	52,93	0,16
53,09	53,41	-0,32	52,84	0,25	53,12	-0,03	53,13	-0,04
53,09	53,46	-0,37	52,87	0,22	53,07	0,02	53,17	-0,08
53,09	53,17	-0,08	52,65	0,44	53,28	-0,19	52,89	0,20
53,09	53,56	-0,47	53,53	-0,44	53,31	-0,22	53,26	-0,17
53,13	53,67	-0,54	53,86	-0,73	53,91	-0,78	53,79	-0,66
53,13	53,88	-0,75	54,03	-0,90	53,83	-0,70	54,02	-0,89
53,14	53,47	-0,33	52,69	0,45	52,97	0,17	53,37	-0,23
53,14	53,50	-0,36	52,72	0,42	52,96	0,18	53,39	-0,25
53,15	53,71	-0,56	52,98	0,17	53,68	-0,53	53,53	-0,38
53,16	54,09	-0,93	53,52	-0,36	53,76	-0,60	53,93	-0,77
53,16	54,06	-0,90	53,46	-0,30	53,75	-0,59	53,84	-0,68
53,22	53,72	-0,50	53,56	-0,34	53,51	-0,29	53,36	-0,14
53,22	53,59	-0,37	53,45	-0,23	53,55	-0,33	53,22	0,00
53,22	53,75	-0,53	53,62	-0,40	53,46	-0,24	53,41	-0,19
53,22	53,74	-0,52	53,60	-0,38	53,48	-0,26	53,40	-0,18
53,23	53,84	-0,61	53,18	0,05	53,74	-0,51	53,57	-0,34
53,23	53,79	-0,56	53,16	0,07	53,67	-0,44	53,56	-0,33
53,23	53,83	-0,60	53,16	0,07	53,75	-0,52	53,55	-0,32
53,23	53,70	-0,47	53,05	0,18	53,79	-0,56	53,40	-0,17
53,25	52,95	0,30	53,66	-0,41	53,31	-0,06	52,85	0,40
53,25	52,95	0,30	53,66	-0,41	53,31	-0,06	52,85	0,40
53,27	53,54	-0,27	54,13	-0,86	54,36	-1,09	54,05	-0,78
53,27	53,56	-0,29	54,18	-0,91	54,40	-1,13	54,07	-0,80
53,27	53,53	-0,26	54,18	-0,91	54,45	-1,18	54,04	-0,77
53,34	54,22	-0,88	54,17	-0,83	54,15	-0,81	54,43	-1,09
53,39	54,71	-1,32	54,80	-1,41	54,18	-0,79	54,73	-1,34
53,39	55,30	-1,91	55,11	-1,72	55,39	-2,00	55,23	-1,84
53,41	53,68	-0,27	53,18	0,23	53,73	-0,32	53,57	-0,16
53,42	54,29	-0,87	54,22	-0,80	53,80	-0,38	53,72	-0,30
53,42	54,24	-0,82	54,20	-0,78	53,75	-0,33	53,69	-0,27
53,42	53,33	0,09	53,95	-0,53	53,32	0,10	53,25	0,17
53,42	52,46	0,96	53,95	-0,53	52,72	0,70	52,54	0,88
53,42	52,57	0,85	54,34	-0,92	52,77	0,65	52,66	0,76
53,43	51,97	1,46	52,96	0,47	51,89	1,54	51,67	1,76
53,43	52,08	1,35	53,05	0,38	52,10	1,33	51,77	1,66
53,43	52,00	1,43	52,98	0,45	51,94	1,49	51,70	1,73
53,45	53,37	0,08	53,41	0,04	53,16	0,29	53,16	0,29

53,48	53,47	0,01	54,36	-0,88	52,66	0,82	53,34	0,14
53,53	53,03	0,50	55,32	-1,79	53,37	0,16	53,34	0,19
53,53	53,02	0,51	55,29	-1,76	53,35	0,18	53,32	0,21
53,53	53,05	0,48	55,36	-1,83	53,39	0,14	53,36	0,17
53,53	53,05	0,48	55,36	-1,83	53,39	0,14	53,36	0,17
53,57	52,85	0,72	53,25	0,32	53,40	0,17	53,06	0,51
53,57	52,86	0,71	53,22	0,35	53,52	0,05	53,11	0,46
53,61	54,16	-0,55	53,83	-0,22	53,96	-0,35	53,68	-0,07
53,61	54,16	-0,55	53,83	-0,22	53,96	-0,35	53,68	-0,07
53,70	54,41	-0,71	53,70	0,00	53,89	-0,19	53,94	-0,24
53,70	54,43	-0,73	53,64	0,06	53,80	-0,10	53,89	-0,19
53,74	51,55	2,19	51,62	2,12	51,88	1,86	51,74	2,00
53,74	51,53	2,21	51,61	2,13	51,86	1,88	51,70	2,04
53,74	51,68	2,06	51,69	2,05	52,02	1,72	51,94	1,80
53,75	52,24	1,51	52,24	1,51	53,05	0,70	52,40	1,35
53,77	54,19	-0,42	56,81	-3,04	54,79	-1,02	54,65	-0,88
53,80	53,92	-0,12	53,70	0,10	53,69	0,11	53,82	-0,02
53,80	53,91	-0,11	53,68	0,12	53,71	0,09	53,80	0,00
53,80	53,89	-0,09	53,66	0,14	53,72	0,08	53,78	0,02
53,87	54,45	-0,58	53,98	-0,11	54,00	-0,13	54,20	-0,33
53,87	53,67	0,20	52,52	1,35	53,56	0,31	53,75	0,12
53,87	53,58	0,29	52,45	1,42	53,43	0,44	53,65	0,22
53,88	53,86	0,02	53,70	0,18	55,04	-1,16	54,44	-0,56
53,88	53,78	0,10	53,68	0,20	54,91	-1,03	54,38	-0,50
53,88	54,08	-0,20	53,80	0,08	55,58	-1,70	54,63	-0,75
53,88	54,14	-0,26	53,86	0,02	55,99	-2,11	54,72	-0,84
53,90	54,66	-0,76	54,81	-0,91	55,11	-1,21	55,31	-1,41
53,90	54,55	-0,65	54,61	-0,71	54,90	-1,00	55,13	-1,23
53,94	53,38	0,56	52,96	0,98	53,69	0,25	53,10	0,84
53,95	53,95	0,00	54,23	-0,28	54,11	-0,16	53,89	0,06
53,95	53,91	0,04	54,18	-0,23	54,00	-0,05	53,83	0,12
53,95	53,82	0,13	54,10	-0,15	53,85	0,10	53,71	0,24
53,97	54,59	-0,62	55,00	-1,03	54,70	-0,73	54,95	-0,98
53,97	55,11	-1,14	55,32	-1,35	55,06	-1,09	55,42	-1,45
53,97	55,08	-1,11	55,29	-1,32	55,02	-1,05	55,38	-1,41
53,97	53,84	0,13	53,55	0,42	53,82	0,15	54,08	-0,11
53,97	53,51	0,46	53,29	0,68	53,64	0,33	53,55	0,42
53,97	54,04	-0,07	53,53	0,44	53,90	0,07	54,40	-0,43
53,97	54,06	-0,09	53,58	0,39	53,91	0,06	54,43	-0,46
54,02	53,93	0,09	54,69	-0,67	54,37	-0,35	54,33	-0,31
54,02	54,17	-0,15	54,92	-0,90	54,72	-0,70	54,61	-0,59
54,02	54,10	-0,08	54,83	-0,81	54,56	-0,54	54,51	-0,49
54,04	53,59	0,45	54,48	-0,44	54,02	0,02	53,73	0,31
54,04	53,60	0,44	54,49	-0,45	54,01	0,03	53,74	0,30
54,04	53,57	0,47	54,46	-0,42	54,02	0,02	53,72	0,32
54,05	53,27	0,78	53,99	0,06	53,32	0,73	53,49	0,56
54,05	53,23	0,82	53,97	0,08	53,26	0,79	53,45	0,60
54,05	52,92	1,13	53,84	0,21	52,98	1,07	53,23	0,82
54,05	52,57	1,48	53,73	0,32	52,77	1,28	52,97	1,08
54,12	53,20	0,92	53,21	0,91	52,19	1,93	53,11	1,01

54,12	53,15	0,97	53,18	0,94	52,11	2,01	53,08	1,04
54,12	53,10	1,02	53,15	0,97	52,03	2,09	53,03	1,09
54,12	53,07	1,05	53,13	0,99	51,98	2,14	52,99	1,13
54,13	53,49	0,64	54,67	-0,54	53,84	0,29	53,78	0,35
54,13	53,54	0,59	54,61	-0,48	53,79	0,34	53,89	0,24
54,15	54,22	-0,07	54,09	0,06	54,92	-0,77	54,02	0,13
54,15	54,37	-0,22	54,24	-0,09	54,66	-0,51	54,16	-0,01
54,15	54,09	0,06	53,98	0,17	55,14	-0,99	53,90	0,25
54,15	54,40	-0,25	54,28	-0,13	54,60	-0,45	54,19	-0,04
54,18	54,65	-0,47	54,80	-0,62	54,64	-0,46	55,02	-0,84
54,18	54,94	-0,76	54,98	-0,80	54,93	-0,75	55,28	-1,10
54,18	54,65	-0,47	54,80	-0,62	54,64	-0,46	55,02	-0,84
54,19	54,97	-0,78	55,62	-1,43	54,77	-0,58	54,88	-0,69
54,19	54,75	-0,56	55,43	-1,24	54,48	-0,29	54,67	-0,48
54,19	54,81	-0,62	55,47	-1,28	54,52	-0,33	54,71	-0,52
54,19	54,59	-0,40	55,36	-1,17	54,39	-0,20	54,58	-0,39
54,19	54,59	-0,40	55,36	-1,17	54,39	-0,20	54,58	-0,39
54,22	53,58	0,64	52,86	1,36	53,44	0,78	53,41	0,81
54,22	53,65	0,57	52,91	1,31	53,40	0,82	53,47	0,75
54,22	53,58	0,64	52,86	1,36	53,44	0,78	53,41	0,81
54,23	53,81	0,42	53,93	0,30	54,19	0,04	54,22	0,01
54,23	53,81	0,42	53,93	0,30	54,19	0,04	54,22	0,01
54,23	53,88	0,35	53,98	0,25	54,25	-0,02	54,30	-0,07
54,26	52,63	1,63	53,52	0,74	53,25	1,01	52,77	1,49
54,27	55,02	-0,75	55,35	-1,08	54,98	-0,71	54,72	-0,45
54,27	54,92	-0,65	55,51	-1,24	55,41	-1,14	54,83	-0,56
54,28	54,57	-0,29	54,72	-0,44	54,61	-0,33	54,93	-0,65
54,28	54,39	-0,11	54,60	-0,32	54,50	-0,22	54,74	-0,46
54,31	53,67	0,64	53,79	0,52	53,63	0,68	53,66	0,65
54,32	55,00	-0,68	53,94	0,38	53,52	0,80	54,83	-0,51
54,32	54,79	-0,47	53,94	0,38	53,51	0,81	54,58	-0,26
54,38	54,22	0,16	54,15	0,23	54,77	-0,39	54,53	-0,15
54,40	53,79	0,61	53,47	0,93	53,48	0,92	53,71	0,69
54,40	53,78	0,62	53,46	0,94	53,49	0,91	53,69	0,71
54,40	53,72	0,68	53,41	0,99	53,52	0,88	53,63	0,77
54,42	53,90	0,52	53,87	0,55	53,96	0,46	54,00	0,42
54,42	54,10	0,32	54,02	0,40	53,94	0,48	54,21	0,21
54,42	53,94	0,48	53,90	0,52	53,95	0,47	54,04	0,38
54,42	52,47	1,95	54,18	0,24	53,93	0,49	52,79	1,63
54,46	53,33	1,13	53,81	0,65	53,55	0,91	53,72	0,74
54,46	53,37	1,09	53,85	0,61	53,58	0,88	53,77	0,69
54,47	54,66	-0,19	54,80	-0,33	54,66	-0,19	54,97	-0,50
54,57	52,98	1,59	52,18	2,39	52,79	1,78	52,75	1,82
54,57	53,18	1,39	52,29	2,28	52,48	2,09	52,94	1,63
54,57	53,20	1,38	52,29	2,28	52,45	2,12	52,95	1,62
54,62	53,81	0,81	53,51	1,11	53,75	0,87	53,78	0,84
54,62	53,86	0,76	53,57	1,05	53,82	0,80	53,85	0,77
54,62	53,67	0,95	53,39	1,23	53,66	0,96	53,62	1,00
54,63	53,36	1,27	54,73	-0,10	54,84	-0,21	53,97	0,66
54,63	53,36	1,27	54,73	-0,10	54,84	-0,21	53,97	0,66

54,63	53,25	1,38	54,63	0,00	54,41	0,22	53,84	0,79
54,64	56,65	-2,01	55,39	-0,75	54,99	-0,35	54,90	-0,26
54,64	56,92	-2,28	55,46	-0,82	55,14	-0,50	54,99	-0,35
54,65	54,03	0,62	54,18	0,47	53,95	0,70	54,35	0,30
54,65	53,74	0,91	53,97	0,68	53,98	0,67	54,05	0,60
54,70	54,04	0,66	54,28	0,42	53,97	0,73	54,13	0,57
54,70	54,62	0,08	54,55	0,15	54,39	0,31	54,55	0,15
54,70	54,60	0,10	54,53	0,17	54,40	0,30	54,53	0,17
54,73	53,44	1,29	53,90	0,83	53,68	1,05	53,50	1,23
54,73	53,65	1,08	54,03	0,70	53,94	0,79	53,71	1,02
54,75	54,37	0,38	55,00	-0,25	54,20	0,55	54,41	0,34
54,75	54,20	0,55	54,82	-0,07	54,27	0,48	54,23	0,52
54,76	53,31	1,45	52,88	1,88	53,47	1,29	53,25	1,51
54,82	53,59	1,23	53,64	1,18	53,64	1,18	53,45	1,37
54,88	53,49	1,39	53,51	1,37	53,27	1,61	53,34	1,54
54,91	53,05	1,86	53,24	1,67	53,51	1,40	53,20	1,71
54,91	52,88	2,03	53,14	1,77	53,18	1,73	53,01	1,90
55,00	53,76	1,24	54,13	0,87	53,98	1,02	53,91	1,09
55,00	53,90	1,10	54,26	0,74	54,10	0,90	54,07	0,93
55,05	54,52	0,53	54,17	0,88	54,92	0,13	54,38	0,67
55,06	55,20	-0,14	54,44	0,62	54,87	0,19	54,73	0,33
55,06	55,36	-0,30	54,56	0,50	55,23	-0,17	54,90	0,16
55,06	55,35	-0,29	54,58	0,48	55,30	-0,24	54,92	0,14
55,08	52,56	2,52	51,17	3,91	52,64	2,44	52,52	2,56
55,10	53,54	1,56	54,56	0,54	53,77	1,33	53,90	1,20
55,10	53,51	1,59	54,53	0,57	53,75	1,35	53,87	1,23
55,10	53,46	1,64	54,48	0,62	53,71	1,39	53,81	1,29
55,11	53,36	1,75	53,81	1,30	53,28	1,83	53,26	1,85
55,11	53,29	1,82	53,74	1,37	53,26	1,85	53,17	1,94
55,12	53,63	1,49	54,72	0,40	56,28	-1,16	54,10	1,02
55,12	54,75	0,37	54,63	0,49	55,60	-0,48	54,51	0,61
55,13	54,77	0,36	54,75	0,38	55,23	-0,10	54,92	0,21
55,15	56,08	-0,93	54,63	0,52	54,47	0,68	54,46	0,69
55,17	54,75	0,42	55,36	-0,19	54,44	0,73	55,04	0,13
55,17	54,51	0,66	55,15	0,02	54,43	0,74	54,80	0,37
55,17	53,52	1,65	53,63	1,54	53,16	2,01	52,94	2,23
55,17	53,46	1,71	53,53	1,64	52,93	2,24	52,86	2,31
55,21	53,46	1,75	54,38	0,83	53,93	1,28	53,58	1,63
55,23	55,75	-0,52	54,68	0,55	54,98	0,25	56,28	-1,05
55,23	55,18	0,05	54,59	0,64	54,64	0,59	55,97	-0,74
55,28	54,16	1,12	53,73	1,55	53,27	2,01	53,33	1,95
55,28	54,27	1,01	53,82	1,46	53,32	1,96	53,44	1,84
55,28	54,25	1,03	53,94	1,34	53,38	1,90	53,46	1,82
55,29	53,78	1,51	53,23	2,06	53,91	1,38	53,74	1,55
55,29	52,68	2,61	53,64	1,65	52,86	2,43	52,76	2,53
55,29	52,74	2,55	53,67	1,62	52,90	2,39	52,79	2,50
55,29	52,80	2,49	53,69	1,60	52,93	2,36	52,83	2,46
55,31	53,71	1,60	54,08	1,23	53,69	1,62	53,76	1,55
55,31	53,94	1,37	54,29	1,02	53,55	1,76	53,99	1,32
55,31	53,63	1,68	54,01	1,30	53,74	1,57	53,68	1,63

55,38	54,02	1,36	55,14	0,24	53,59	1,79	54,02	1,36
55,38	54,00	1,38	55,10	0,28	53,60	1,78	54,00	1,39
55,38	53,87	1,51	54,91	0,47	53,65	1,73	53,87	1,51
55,38	53,97	1,41	55,06	0,32	53,61	1,77	53,97	1,41
55,40	53,34	2,06	53,12	2,28	53,23	2,17	53,26	2,14
55,40	54,31	1,09	53,89	1,51	53,97	1,43	54,35	1,05
55,40	54,07	1,33	53,73	1,67	54,10	1,30	54,11	1,29
55,40	54,25	1,15	53,85	1,55	54,00	1,40	54,30	1,10
55,40	54,07	1,33	53,73	1,67	54,10	1,30	54,11	1,29
55,41	53,84	1,57	53,62	1,79	53,36	2,05	53,81	1,60
55,41	53,71	1,70	53,51	1,90	53,43	1,98	53,67	1,74
55,51	51,86	3,65	52,82	2,69	51,65	3,86	51,74	3,77
55,51	52,04	3,47	52,97	2,54	51,91	3,60	51,93	3,58
55,51	52,04	3,47	52,97	2,54	51,91	3,60	51,93	3,58
55,54	53,73	1,81	53,89	1,65	53,73	1,81	53,23	2,31
55,54	53,48	2,06	53,76	1,78	53,45	2,09	53,08	2,46
55,54	53,48	2,06	53,76	1,78	53,45	2,09	53,08	2,46
55,54	53,74	1,80	53,92	1,62	53,78	1,76	53,26	2,29
55,54	53,21	2,33	53,68	1,86	53,30	2,24	52,96	2,58
55,57	53,83	1,74	53,22	2,35	53,76	1,81	53,76	1,81
55,57	54,10	1,47	53,33	2,24	53,93	1,64	53,94	1,63
55,58	54,08	1,50	54,18	1,40	54,81	0,77	54,58	1,00
55,64	53,03	2,61	53,04	2,60	53,12	2,52	52,74	2,90
55,64	53,13	2,51	53,11	2,53	53,27	2,37	52,82	2,82
55,64	52,76	2,88	52,96	2,68	52,93	2,71	52,61	3,03
55,66	53,95	1,71	53,80	1,86	53,82	1,84	54,25	1,41
55,66	53,88	1,78	53,70	1,96	53,73	1,93	54,16	1,50
55,70	53,67	2,03	54,56	1,14	54,27	1,44	54,01	1,69
55,70	54,08	1,62	54,90	0,80	54,07	1,63	54,44	1,26
55,74	54,26	1,48	54,50	1,24	53,93	1,81	54,20	1,54
55,75	56,09	-0,34	55,20	0,55	54,90	0,85	54,75	1,00
55,77	53,71	2,06	53,06	2,71	53,74	2,03	53,56	2,21
55,77	53,78	1,99	53,11	2,66	53,68	2,09	53,63	2,14
55,85	56,06	-0,21	56,07	-0,22	55,74	0,11	56,18	-0,33
55,85	56,06	-0,21	56,07	-0,22	55,74	0,11	56,18	-0,33
55,85	56,15	-0,30	56,16	-0,31	55,55	0,30	56,27	-0,42
55,85	55,03	0,82	54,72	1,13	55,63	0,22	56,45	-0,60
55,85	54,67	1,18	54,48	1,37	54,41	1,44	56,04	-0,19
55,98	52,77	3,21	53,32	2,66	53,18	2,80	52,92	3,06
55,99	54,27	1,72	54,27	1,72	54,66	1,33	54,56	1,43
56,04	53,89	2,15	53,43	2,61	53,74	2,30	53,74	2,30
56,06	54,20	1,86	54,83	1,23	54,48	1,58	54,38	1,68
56,06	54,35	1,71	55,05	1,01	54,99	1,07	54,63	1,43
56,06	54,28	1,78	54,88	1,18	54,59	1,47	54,47	1,59
56,07	57,21	-1,14	55,08	0,99	56,70	-0,63	57,25	-1,18
56,09	53,08	3,01	54,36	1,73	53,44	2,65	53,13	2,96
56,09	53,26	2,83	54,56	1,53	53,63	2,46	53,28	2,81
56,09	53,28	2,81	54,59	1,50	53,68	2,41	53,30	2,79
56,09	52,92	3,17	54,23	1,86	53,34	2,75	53,02	3,07
56,33	52,52	3,81	52,32	4,01	52,99	3,34	52,91	3,42

56,33	52,43	3,90	52,26	4,07	52,82	3,51	52,81	3,52
56,33	52,36	3,97	52,22	4,11	52,74	3,59	52,73	3,60
56,34	55,09	1,25	54,94	1,40	55,15	1,19	55,48	0,86
56,34	53,23	3,11	54,06	2,28	53,60	2,74	53,41	2,93
56,41	54,02	2,39	53,71	2,70	53,64	2,77	53,67	2,74
56,48	52,96	3,52	53,12	3,36	53,07	3,41	52,84	3,64
56,50	53,05	3,45	52,46	4,04	53,96	2,54	52,83	3,67
56,50	52,89	3,61	52,35	4,15	54,47	2,03	52,68	3,82
56,67	52,60	4,07	53,24	3,43	52,47	4,20	52,78	3,89
56,67	52,62	4,05	53,26	3,41	52,49	4,18	52,80	3,87
56,71	54,98	1,73	54,79	1,92	54,61	2,10	55,01	1,70
56,74	54,06	2,68	54,12	2,62	53,80	2,94	53,73	3,01
56,91	53,91	3,00	53,92	2,99	53,43	3,48	53,94	2,97
56,95	54,55	2,40	54,57	2,38	54,40	2,55	54,86	2,09
57,10	53,38	3,72	53,79	3,31	53,52	3,58	53,59	3,51
57,32	54,41	2,91	55,06	2,26	54,15	3,17	53,95	3,37
57,32	54,15	3,17	54,90	2,42	53,93	3,39	53,79	3,53
57,32	54,15	3,17	54,90	2,42	53,93	3,39	53,79	3,53
57,33	54,86	2,47	54,92	2,41	55,14	2,19	54,89	2,44
57,33	54,37	2,96	54,69	2,64	54,58	2,75	54,53	2,80
57,33	54,79	2,54	54,87	2,46	54,99	2,34	54,82	2,51
57,49	53,78	3,71	53,22	4,27	53,63	3,86	53,65	3,85
57,49	53,82	3,67	53,25	4,24	53,61	3,88	53,68	3,81
57,49	53,48	4,01	54,09	3,40	53,82	3,67	53,90	3,59
57,49	53,51	3,98	54,12	3,37	53,84	3,65	53,94	3,55
57,49	53,55	3,94	54,14	3,35	53,86	3,63	53,97	3,52
57,49	53,58	3,91	54,17	3,32	53,89	3,60	54,01	3,48
57,49	53,51	3,98	54,12	3,37	53,84	3,65	53,94	3,55
57,52	54,63	2,89	54,79	2,73	54,65	2,87	54,71	2,81
57,77	53,73	4,04	54,27	3,50	53,84	3,93	53,84	3,93
57,81	56,90	0,91	54,33	3,48	55,04	2,77	54,86	2,95
57,81	57,32	0,49	54,39	3,42	55,27	2,54	54,98	2,83
57,82	54,32	3,50	53,99	3,83	55,06	2,76	54,57	3,25
57,82	54,12	3,70	53,88	3,94	54,63	3,19	54,38	3,44
57,85	52,61	5,24	53,25	4,60	52,65	5,20	52,46	5,39
57,85	52,64	5,21	53,09	4,76	52,63	5,22	52,44	5,41
57,85	52,55	5,30	53,01	4,84	52,63	5,22	52,34	5,51
57,85	52,68	5,17	53,14	4,71	52,63	5,22	52,49	5,36
57,92	56,85	1,07	55,67	2,25	55,77	2,15	55,73	2,19
58,16	54,03	4,13	55,53	2,63	54,61	3,55	54,93	3,23
58,24	53,16	5,08	53,70	4,54	53,60	4,64	53,63	4,61
58,24	53,33	4,91	53,85	4,39	53,74	4,50	53,82	4,42
58,90	54,10	4,80	57,43	1,47	53,59	5,31	54,06	4,84
59,03	57,35	1,68	54,77	4,26	56,69	2,34	56,38	2,65
59,03	57,96	1,07	54,65	4,38	55,95	3,08	56,29	2,74
59,05	54,57	4,48	55,22	3,83	54,41	4,64	54,47	4,58
59,26	56,10	3,16	55,61	3,65	55,41	3,85	55,37	3,89
59,30	54,62	4,68	54,95	4,35	54,59	4,71	54,90	4,40
59,58	53,49	6,09	54,19	5,39	53,50	6,08	53,16	6,42
59,58	53,76	5,82	54,32	5,26	53,85	5,73	53,32	6,26

E

Datos del Autor

1. Jefe de Informática y Comunicaciones de la Empresa Cmdte Ernesto Che Guevara.
2. Miembro del consejo IAC del Ministerio de la Industria Básica de Cuba.
3. Máster en Matemática Aplicada e Informática para la Administración por la Universidad de Holguín.
4. Instructor Adjunto de la Universidad de Holguín.

Es graduado en Ingeniería Electrónica en la Universidad Central de las Villas, Cuba en el año 1985, con resultados académicos sobresalientes.

Formó parte del Contingente Estudiantil Universitario que realizaron sus proyectos de tesis en las instalaciones del entonces Proyecto 304, proceso inversionista que formó la actual Empresa Cmdte Ernesto Che Guevara. Planta de Níquel de Punta Gorda.

En 1986 participó en los trabajos de ajuste y puesta en marcha de la automatización de la Planta de Preparación de Mineral como parte del proceso de inico de sus operaciones. Posteriormente, de conjunto con la asesoría extranjera se le asigna la responsabilidad de diagnosticar los sistemas de cómputo suministrados.

Durante el período 1986 a 1990 trabaja en la búsqueda de soluciones a los problemas que se frecuentan en dichos sistemas, hasta lograr dirigir la introducción de la técnica de los Autómatas Programables en sustitución de todo la tecnología suministrada, que trabajaba con muchas dificultades.

En 1996 es nombrado al frente del Grupo de Automática de la Empresa, que comienza con la ejecución de un fuerte proceso inversionista en la automatización de las Plantas de Hornos de Reducción y Calcinación y Sínter. Muchas son las tareas que se materializan como parte de la modernización Empresarial.

En el año 2000, se le responsabiliza con la dirección de la Actividad de Informática y las Comunicaciones de la Empresa, con el objetivo de desarrollar la informatización de los procesos de negocios. Luego de desarrollarse un importante proceso inversionista, se introducen aplicaciones informáticas de uso multiusuario.

Es iniciador de los trabajos de colaboración entre la Universidad de Holguín y la Empresa Cmdte Ernesto che Guevara, hasta materializar la creación de una Unidad Docente. Manifiesta un interés marcado en mantener la superación técnica para todo el colectivo que dirige.

Miembro del Consejo de Dirección de la Empresa, goza de un gran prestigio técnico, es respetado y admirado por sus compañeros. Sus relaciones humanas son excelentes y por varios años le ha sido reconocido su trabajo en aras de la informatización empresarial.

Actualmente investiga, de conjunto con el Departamento de Arquitectura de Computadores de la Universidad de Granada en España, sobre la aproximación funcional con redes RBF de procesos asociados al consumo de portadores energéticos en la Planta de Hornos de Reducción, como parte de un Doctorado cooperado con la Universidad de Holguín.

El tema de la Gestión de la Cadena de Suministros, es otra línea de investigación en la cual participa, y que de una forma muy especial se ha materializado con jóvenes estudiantes y profesores de la Facultad de Informática de la Universidad de Holguín.