UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Departamento de
Ciencias de la Computación e Inteligencia Artificial

Aprendizaje Evolutivo Multiobjetivo de Sistemas Difusos Jerárquicos y su Aplicación en Astrofísica

Tesis Doctoral

Alicia Desirée Benítez Yáñez

Granada, septiembre de 2.015

Editorial: Universidad de Granada. Tesis Doctorales
Autora: Alicia Desirée Benítez Yáñez
ISBN: 978-84-9125-312-9
URI: http://hdl.handle.net/10481/41088

Aprendizaje Evolutivo Multiobjetivo de Sistemas Difusos Jerárquicos y su Aplicación en Astrofísica

MEMORIA QUE PRESENTA

Alicia Desirée Benítez Yáñez

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

SEPTIEMBRE DE 2.015

DIRECTOR

Jorge Casillas Barranquero

DEPARTAMENTO DE
CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

E.T.S. DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN
UNIVERSIDAD DE GRANADA

La memoria titulada **Aprendizaje Evolutivo Multiobjetivo de Sistemas Difusos Jerárquicos y su Aplicación en Astrofísica**, que presenta Dña. Alicia Desirée Benítez Yáñez para optar al grado de doctor, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del doctor D. Jorge Casillas Barranquero.

Granada, septiembre de 2.015

La doctoranda:

Alicia D. Benítez

El director:

Jorge Casillas

La doctoranda Alicia Desirée Benítez Yáñez y el director de la tesis Jorge Casillas Barranquero. Garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por la doctoranda bajo la dirección del director de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, septiembre de 2.015

La doctoranda: _____

Alicia D. Benítez

El director: _____

Jorge Casillas

Agradecimientos

Es difícil imaginar la cantidad de esfuerzo y dedicación que hay detrás de toda memoria de tesis doctoral. En mi caso, horas de programación, pruebas de software, escritura de la memoria, reuniones, radio y litros de té. Además, puede ser realmente exhausto cuando se combina con un trabajo en el que das lo mejor de ti misma. Todo esto se hace más llevadero si las personas que te acompañan durante el viaje te hacen el camino más agradable, de ahí que quiera dedicar unas líneas de agradecimiento a todas ellas, es lo menos que puedo hacer.

Quiero agradecer a mi director de tesis, Jorge Casillas, su tiempo, esfuerzo, paciencia, ayuda y orientación que, durante la realización de este trabajo, han sido tan necesarias para mí.

También desearía dar las gracias a Paco Herrera por despertar en mí el interés hacia el mundo de la algoritmia en sus clases de "Teoría de Algoritmos" cuando estaba en segundo de carrera, por sus consejos y su apoyo incondicional.

A mis padres, Antonio y María del Carmen, por querer siempre lo mejor para sus hijos, por inculcarme los valores de constancia, perseverancia, responsabilidad, el amor por el trabajo bien hecho y su apoyo e interés en cada momento de mi vida. A mi hermano José Antonio por su compañía, su cariño y por estar pendiente de mí día a día.

Mi agradecimiento a todos mis compañeros del Instituto de Astrofísica de Andalucía (IAA-CSIC) por su compañerismo, amabilidad y amistad. A Joserra por su sentido del humor que hacen las horas de trabajo más amenas y aunque me diga que no me aguanta, después de seis años conviviendo en el mismo despacho, sé que no puede vivir sin mí. A María por la dulzura y amabilidad que desprende. A mis compañeros del grupo "Galactic Center", a Rainer, Laly, Teresa, Paco y Hui por sus ánimos, apoyo y confianza. A Isa por su positividad y por sus avisos. A Cris por su simpatía y por esos "phoskitos"que tanta energía me dieron para acabar esta memoria. A Susana por ser afable. A William, Enrique Pérez y Roberto Cid Fernandes por su tiempo y ayuda en la elaboración de parte de este trabajo. A Luisa por su elocuencia, cordialidad y confianza. A mis compañeros del Centro de Cálculo por permitirme usar la infraestructura Grid para los cálculos.

Agradezco a todos mis compañeros del grupo “Soft Computing y Sistemas Inteligentes” de la Universidad de Granada, el estar siempre ahí, en especial a Rafa, Ana María, Rocío y Carlos, por su simpatía y compañerismo.

Tampoco puedo olvidar a mis amistades que siempre me han apoyado en cualquier momento, a pesar de mis incesantes negativas cuando me llamaban para vernos. A Mari Carmen, Wafaá, Puri, María Jesús, Ana y Belén. Gracias a Celia, Migue, Nieves y José Manuel por seguir en contacto a pesar de la distancia. A Salva por esos cafes y sus consejos. A Mercedes que con sus clases de inglés hacía que me evadiera.

A Carlos sus ánimos, consejos, por hacerme ver la vida desde otra perspectiva, estar siempre pendiente y por su infinita paciencia.

A Kenia y Chispa por los momentos de felicidad que me han brindado. A Candy por esperar pacientemente a mi lado cada tarde para salir a pasear mientras escribía esta memoria.

Y por si me dejo a alguien en el camino..., ¡Gracias miles a todos/as!

Índice general

Índice de figuras

Índice de tablas

Introducción

A. Planteamiento

Un espacio multidimensional constituye el resultado de un complejo proceso de abstracción e idealización para generalizar el concepto *espacio*. Su dificultad está íntimamente relacionada con el número de variables del mismo. Un tipo de problema que se adecúa a estas características es la multidimensionalidad en espacios continuos, denominado problema de regresión. El análisis de regresión se realiza para predecir problemas reales, como por ejemplo, el consumo diario de energía hidroeléctrica, eléctrica nuclear, carbón, combustible y gas natural.

Un problema de regresión cuenta con un conjunto de variables de tipo real que podría ser resuelto mediante modelos matemáticos. Estos modelos matemáticos realizan un análisis de las variables para contemplar si existe una relación entre ellas. Este tipo de modelizado puede proporcionar precisión en su solución, pero no existe cabida para la interpretación de los datos. Si lo que pretendemos, además de la resolución del problema de forma precisa, es generar un conocimiento de esa información, debemos plantearnos otra metodología de resolución basadas en la Inteligencia Artificial.

Una forma de modelizado de problemas de regresión que tiene en cuenta tanto la precisión de la solución como su interpretación es el sistema basado en reglas difusas. Este sistema contiene una base de reglas basadas en reglas SI-ENTONCES que establecen relaciones lógicas entre las variables del problema. De esta forma la extracción del conocimiento es rápida y sencilla.

Para determinar la bondad de este tipo de sistemas, hemos de analizar dos aspectos: la precisión, que atiende a la coherencia del conocimiento usado y el comportamiento dinámico del sistema, y la interpretabilidad, que hace referencia a en qué grado es fácil comprender el razonamiento seguido por el sistema. Esta última viene determinada en parte por el tamaño del conocimiento empleado, es decir, por el

número de términos lingüísticos y el número de reglas del sistema.

La literatura muestra un gran número de trabajos que recurren a estos modelos para solventar problemas de regresión. La resolución de estos problemas suele ser satisfactoria cuando el número de variables del problema es pequeño. En este caso, los sistemas basados en reglas difusas son precisos e interpretables debido a que el problema presenta menor complejidad de resolución. Por otro lado, si el problema es complejo, y con esto nos referimos específicamente a que el problema esté formado por un número de variables considerable, se producen dos situaciones que afectan negativa y directamente en la interpretación del sistema:

1. Incremento exponencial del número de reglas. Cuanto mayor sea el número de reglas del sistema, más difícil será su interpretación, tanto a nivel de regla como en su conjunto.

2. Aumento del tiempo de computación, de forma que a mayor número de reglas, mayor tiempo de computación del modelo.

En este caso, debemos hacer uso de otros métodos que permitan erradicar este problema. Los sistemas difusos jerárquicos consiguen resolver problemas de alta dimensionalidad. Estos sistemas se estructuran en capas de subsistemas difusos o módulos enlazados entre sí, donde cada módulo puede recibir como entrada tanto variables del problema como la salida del módulo de la capa anterior. La capacidad de esta metodología, que proporciona una mayor interpretabilidad al sistema, reside en agrupar las reglas en módulos atendiendo a las variables de entrada. Posteriormente, en cada módulo se calcula una solución parcial y cada una de estas soluciones parciales se usan por otros módulos posteriores para calcular la salida final del sistema.

Por otro lado, a pesar de las ventajas que hemos enumerado tanto para sistemas basados en reglas difusas como para jerarquías difusas, ambos modelos presentan un problema: han de establecer un equilibrio entre la precisión y la interpretabilidad del sistema. En estos sistemas cuanto mayor es el número de reglas difusas, mayor es la precisión. Por el contrario, cuanto menor número de reglas tenga el sistema, la interpretabilidad del sistema mejorará, pero perderá precisión. Como se puede apreciar, precisión e interpretabilidad son conceptos inversamente proporcionales.

Modelos computacionales de aprendizaje reforzado, tales como los algoritmos evolutivos, pueden ayudarnos a solventar este problema ya que se puede enfocar como una tarea de optimización. Los algoritmos evolutivos son métodos versátiles, ro-

bustos y no utilizan información adicional distinta a la del propio problema. En nuestro caso, tenemos dos objetivos: minimizar el error del sistema para que la precisión sea mayor y minimizar el número de reglas para que el sistema sea más interpretable. Para realizar una optimización de este tipo, hacemos uso de los algoritmos evolutivos multiobjetivo.

Por tanto, parece interesante el aprendizaje de la estructura jerárquica de sistemas basados en reglas difusas mediante un algoritmo multiobjetivo que garantice una buena interpretabilidad, mejorando o manteniendo la precisión del sistema. Este es el fin del trabajo que perseguimos en esta memoria.

B. Objetivos

El objetivo de este trabajo es el diseño e implementación de un conjunto de métodos de aprendizaje que permitan obtener modelizados jerárquicos difusos que aseguren una buena interpretabilidad del sistema, mejorando o igualando la precisión con respecto a un modelizado difuso convencional. Este objetivo general se descompone en los siguientes objetivos parciales:

1. *Análisis del estado del arte en el campo del Aprendizaje Automático.* Realizaremos un análisis profundo de las propuestas existentes en la actualidad. Debido a que este campo es amplio, nos ceñiremos a las técnicas que usamos en nuestro trabajo: modelizado mediante sistemas basados en reglas difusas, sistemas difusos jerárquicos y aprendizaje evolutivo.

2. *Mejora en la interpretabilidad mediante Sistemas Difusos Jerárquicos.* Propondremos una serie de métodos basados en el aprendizaje de estructuras jerárquicas de distintos tipos (serie, paralelo e híbrido). La principal virtud de cada uno de los métodos que presentamos en este trabajo es la reducción de la complejidad del sistema cuando se resuelven problemas con un número elevado de variables intentando que la precisión de cada uno de los modelos se mantenga o, incluso, mejore. Además, realizaremos un análisis exhaustivo de cada una de las tres propuestas para comprobar su capacidad frente a distintos conjuntos de datos.

3. *Realizar una aplicación a problemas reales.* Nos resulta interesante ver la aplicabilidad en problemas reales. Para ello, realizamos una experimentación con

datos fotométricos astrofísicos referentes a conjuntos de datos reales de espectros de galaxias para determinar algunas de sus propiedades físicas.

C. Resumen

Esta memoria se compone de cinco capítulos, una sección de comentarios finales y cuatro apéndices. El desarrollo de los objetivos propuestos, están estructurados de la siguiente forma en esta memoria:

El capítulo 1 contiene una introducción a este trabajo, de forma que orienta al lector sobre el tipo de problema que pretendemos solventar, en nuestro caso problemas de regresión, haciendo un repaso a las distintas técnicas aplicadas a este tipo de problemas para centrarse posteriormente en los sistemas basados en reglas difusas como alternativa gracias a su legibilidad y robustez. Además, describe el uso generalizado actual de los algoritmos evolutivos multiobjetivo en los sistemas basados en reglas difusas, así como las estrategias para lidiar con problemas de alta dimensionalidad.

El capítulo 2 explica en detalle el modelizado de sistemas mediante Sistemas Difusos Jerárquicos. Presenta cada uno de sus componentes y las distintas topologías que podemos encontrar en la literatura. Este capítulo nos da una referencia general de la importancia de este tipo de modelizado.

El capítulo 3 presenta el diseño de un método de aprendizaje de estructuras jerárquicas difusas en serie. El capítulo realiza una descripción completa de la metodología que implementa el algoritmo: codificación, operadores genéticos, mecanismo de inferencia, aprendizaje de la base de reglas, el enfoque multiobjetivo y las funciones de evaluación del mismo. También incluye un estudio experimental del método, en donde se describen los conjuntos de datos usados y un análisis detallado de su comportamiento. Por otro lado, en este capítulo incluimos un modelizado jerárquico en serie con ajuste en las funciones de pertenencia, junto a un estudio de su potencial.

El capítulo 4 describe dos metodologías de aprendizaje de dos tipos de estructuras jerárquicas: paralela e híbrida. Al igual que en el capítulo 3, este capítulo está dotado con una descripción minuciosa de cada uno de los componentes de las metodologías (codificación, operador de cruce, mutación, etc.) y de una experimentación para comprobar la capacidad de cada una de ellas.

El capítulo 5 aborda la aplicación del modelizado jerárquico en serie con ajuste en las funciones de pertenencia a problemas reales pertenecientes al campo de la Astrofísica, concretamente a cuatro propiedades físicas de las galaxias. En el capítulo se

realiza además una comparativa de esta propuesta con otras metodologías usadas en Astrofísica. Esto nos dará una idea de si nuestra técnica es competitiva con respecto a otras técnicas de regresión que se emplean habitualmente en este campo.

En la sección de comentarios finales, pondremos de manifiesto las conclusiones a las que hemos llegado con la realización del trabajo que proponemos. Esta sección incluye las líneas futuras de investigación como continuación de este trabajo.

Por último, incluiremos cuatro apéndices que contienen una descripción de los métodos con los que comparamos nuestras propuestas y una descripción de un conjunto de técnicas pertenecientes al aprendizaje automático. De esta forma, se pone de manifiesto unas nociones sobre algoritmos evolutivos, en el que se indica en que consiste este tipo de algoritmo y su funcionamiento, para conocer los mecanismos de optimización, una introducción a los sistemas basados en reglas difusas, explicando su composición y su funcionamiento y, finalmente, un conocimiento elemental sobre los sistemas difusos genéticos. Además, nombraremos algunos trabajos pertenecientes a esta temática y que forman parte del estado del arte.

Finalmente, esta memoria concluye con la bibliografía de referencia que ha permitido la implementación de nuestro trabajo.

1. Modelizado de problemas de Regresión

Si un ordenador no puede aprender, no será inteligente.

ROGER SCHANK (1.946–), TEÓRICO DE LA INTELIGENCIA ARTIFICIAL, PSICÓLOGO COGNITIVO Y CIENTÍFICO EN APRENDIZAJE AMERICANO

El diccionario de la Real Academia Española define el término "aprender" como la adquisición de conocimiento por medio del estudio o de la experiencia. Durante el proceso de aprendizaje se identifican objetos, se ordenan y se encasillan en la realidad para poder predecir qué puede suceder. Por ello, "aprender" es un proceso complejo que engloba también el análisis y la comprensión del entorno que rodea a un individuo y que, posteriormente, se aplica a la propia existencia. De esta forma, el aprendizaje hace que se adquieran nuevas conductas para las experiencias presentes y futuras. Tanto humanos como animales poseen capacidad de aprendizaje. Esta capacidad se caracteriza principalmente por ser duradera y es posible gracias a la información externa al individuo.

Muchos investigadores del área de la Inteligencia Artificial han estado, y en la actualidad lo están, interesados en el aprendizaje por parte de agentes inertes, es decir, en máquinas inteligentes. Mediante el análisis de datos, se pueden crear sistemas inteligentes capaces de extraer el conocimiento y dar soluciones factibles a problemas. La clave reside principalmente en la implementación de software inteligente, de forma que realice un razonamiento lógico, rápido y preciso como respuesta a distintos estímulos que se pueden presentar en una determinada situación y que sea capaz de mejorar su rendimiento a través de la experiencia acumulada. De esta forma, ten-

dríamos que codificar las soluciones basándonos en el conocimiento que tenemos del problema para cada situación posible, para así obtener una respuesta aproximada frente a un determinado estímulo. Todo este proceso constituye el aprendizaje automático.

El aprendizaje automático tiene como objetivo el estudio de métodos computacionales mediante la automatización de la adquisición del conocimiento a partir de la experiencia y mejorar su rendimiento. Contiene dos áreas principales: el aprendizaje supervisado y el aprendizaje no supervisado. La principal diferencia entre estas dos áreas es que en el aprendizaje no supervisado no existe un conocimiento a priori. Por el contrario, el aprendizaje supervisado parte de unos datos que permiten realizar una predicción basada en el comportamiento de esos datos. El aprendizaje supervisado juega un papel importante en la actualidad dado que el número de datos, en cualquier ámbito científico-tecnológico, ha crecido considerablemente en los últimos años. La elaboración de este tipo de sistemas es esencial para gestión de tales cantidades de datos, destacando el manejo de los mismos y la búsqueda de su significado. En este sentido, son muchos los investigadores que han implementado métodos de aprendizaje automático basándose en modelos de toma de decisiones, psicológicos (teoría sobre la conducta y comportamiento racional), neurológicos (neuronas y especialización del cerebro), evolutivos (adaptativos e inspirados en la evolución biológica) y lingüísticos (representación del conocimiento, gramática de la lengua y lingüística computacional). Además, estos sistemas pueden resolver problemas tales como clasificación (predicción de categorías), optimización (maximización o minimización de un valor) y regresión (predicción de un número).

El problema de la regresión nos despierta un especial interés debido a que en la literatura existen numerosas técnicas y métodos de aprendizaje automático para solventar problemas de clasificación y optimización. A pesar de que existen técnicas de aprendizaje aplicadas a regresión, lo son en menor medida. La regresión es un problema importante y ampliamente extendido ya que lo podemos encontrar cotidianamente, tanto en el consumo diario de energía hidroeléctrica, eléctrica nuclear, carbón, combustible y gas natural, o científicamente, como el que podría tener un astrofísico para conocer propiedades físicas (edad, metalicidad, masa total, extinción estelar, etc.) de las galaxias.

Por ello, este capítulo realiza una introducción al problema de regresión, a las técnicas de aprendizaje automático, ya que son la base de nuestro trabajo, y las distintas alternativas en aprendizaje automático para obtener una mayor legibilidad, robustez

y eficiencia. En este capítulo no incluiremos una descripción detallada del aprendizaje automático. En los apéndices B, C y D puede consultarse una descripción más detallada de las técnicas en las que se basa nuestro trabajo y su funcionamiento.

1.1. Regresión

A menudo podemos solventar problemas mediante la estimación de una variable en función de otra(s). Estos problemas se denominan problemas de regresión. La estimación de la variable se puede realizar mediante un modelo de regresión o una función. De esta forma, se puede investigar y modelizar la relación entre variables.

Ejemplos de esta clase de problemas son los procesos complejos de los sistemas de ingeniería aeroespacial, bioquímica, sistemas ecológicos, sociales, los correspondientes al área económico-financiera o, como mencionamos anteriormente, problemas astrofísicos para poder explicar propiedades y fenómenos de los cuerpos estelares. Básicamente, lo que se pretende es estudiar las relaciones de los hechos.

En los siguientes apartados analizaremos el problema de la regresión desde dos puntos de vista bien diferenciados: matemático, mediante técnicas estadísticas, y aprendizaje automático, abordado con algoritmos de aprendizaje.

1.1.1. Estadística

La Estadística (Hermoso-Gutiérrez y Hernández-Bastida (1997)) es el área de las matemáticas que se encarga de estudiar los datos, obteniendo las características más relevantes de los mismos. Los problemas de regresión hacen uso de variables numéricas, es decir, variables cuantitativas. Además, toman cualquier valor dentro de un intervalo específico de valores, en pocas palabras, son variables continuas.

También realiza una distinción entre otros tipos de variables que nos podemos encontrar en cuanto al estudio de los datos se refiere, clasificándolas en dos grupos atendiendo a la influencia que ejercen unas variables sobre otras: 1) variables independientes o exógenas, que son las variables que se utilizan para hacer la estimación y su valor no depende de otra(s) variable(s), y 2) variables dependientes o endógenas, que son las variables que se pueden predecir y su valor depende del que tome(n) otra(s) variable(s). Además, en este área podemos encontrar distintos tipos de ajuste, que se ilustran en la figura 1.1:

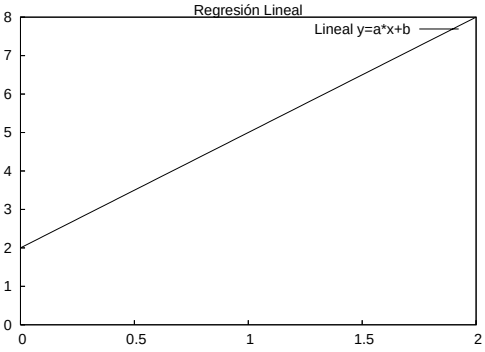
- *Lineal.* Realiza el ajuste a una función recta de un conjunto de valores observados. Es la que mejor ajusta los datos en espacios bidimensionales (Fig. 1.1(a)).
- *No lineal.* A pesar de que la regresión lineal se puede aplicar a muchos problemas, a veces el conjunto de datos sigue una relación distinta a una función lineal, apareciendo relaciones parabólicas, hiperbólicas, potenciales y exponenciales (Fig 1.1(b)).
- *Logística.* Se aplica dependiente del contexto del problema, destacando así la curva logística y la exponencial modificada. La curva logística es creciente, de trayecto de crecimiento rápido que pasa por un punto de inflexión a un crecimiento más lento, de forma que se aproxima a una asíntota horizontal (Fig. 1.1(c)).
- *Segmentada.* En este caso, la variable independiente se particiona en intervalos y el ajuste puede ser una línea o una curva dependiendo de los datos existentes en cada intervalo (Fig. 1.1(d)).

El desarrollo de este tipo de modelos matemáticos para datos reales se puede llevar a cabo mediante simulaciones, análisis de comportamiento de los datos, predicciones, etc. Para implementar modelos de forma matemática, hemos de tener en cuenta dos cuestiones: 1) el rápido crecimiento de la cantidad de datos, y 2) el desarrollo inadecuado de los modelos puede conducir a la obtención de resultados no exitosos y erróneos del modelo en cuestión. A esto se une necesidades tales como, una importante precisión, tratamiento y respuestas en tiempo real y carecimiento formal sobre su funcionamiento. Además puede que los datos posean un comportamiento fuertemente no lineal, con alto grado de incertidumbre y características con variaciones de tiempo.

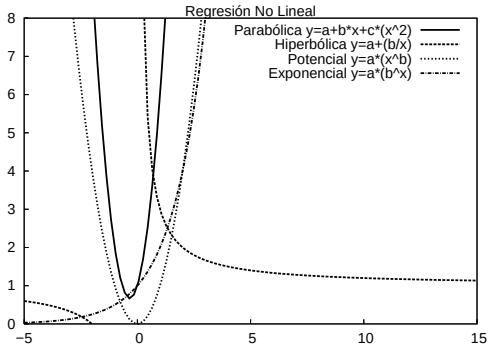
Como se puede apreciar, esta forma incrementa considerablemente la complejidad de la elaboración de los modelos. Para solventar esta complejidad, el aprendizaje automático presenta un conjunto de alternativas que proporcionan una mayor legibilidad de los datos y eficiencia en problemas de regresión.

1.1.2. Algoritmos de Aprendizaje

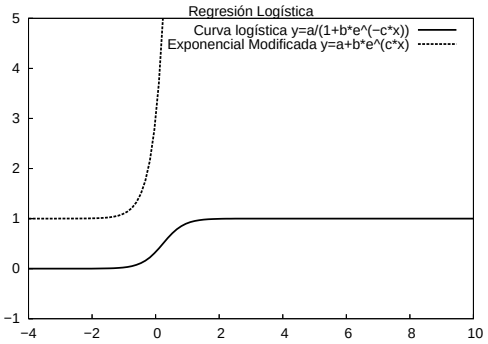
En el aprendizaje automático podemos encontrar una gran diversidad de propuestas que se pueden aplicar a problemas reales (Yao y Liu 2014). Esta sección está



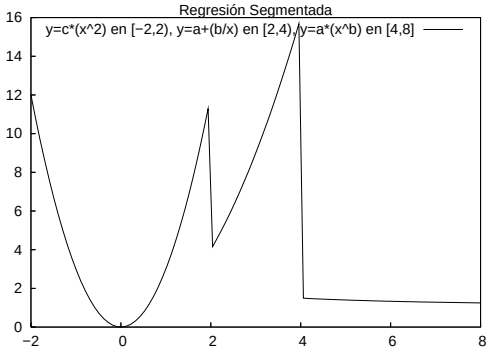
(a) Regresión Lineal



(b) Regresión No Lineal



(c) Regresión Logística



(d) Regresión Segmentada

Figura 1.1.: Tipos de curvas de regresión

encuadrada en el aprendizaje supervisado, es decir, cada uno de los algoritmos que tratamos generan un modelo de predicción a partir de unos datos de entrenamiento, pero no abordaremos los algoritmos de agrupamiento porque se aplican a problemas de clasificación y no a regresión.

1.1.2.1. Redes Neurales

Este tipo de aprendizaje se basa en la simulación del funcionamiento de un cerebro biológico (Ata 2015). Una red neuronal es un sistema que contiene un conjunto de procesos (neuronas) interconectados para desarrollar una tarea. Estos procesos

operan en paralelo y se comunican entre sí mediante sinapsis en varias capas, donde cada conexión entre neuronas tiene un peso asociado. Una red neuronal se compone de una capa de entrada de datos, una capa oculta y una capa de salida de datos. La información se trasmite de capa en capa.

La red neuronal necesita un conjunto de datos de entrenamiento y prueba. Por otro lado, para implementar una red neuronal hemos de atender al número de entradas y salidas del problema que pretendemos resolver y realizar un entrenamiento de la misma mediante un algoritmo de aprendizaje para que la red neuronal aprenda la asociación que existe entre los patrones de entrada y las clases correspondientes.

La figura 1.2 refleja un esquema de red neuronal que, en este caso, está formada por dos capas ocultas. El perceptrón, desarrollado por Rosenblatt en 1962, es una de las técnicas de referencia que incluye una sola capa oculta.

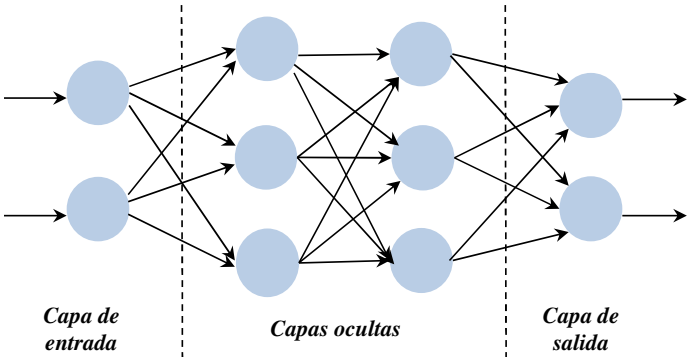


Figura 1.2.: Esquema de una red neuronal

La red neuronal de función de base radial (Montazer y Giveki 2015; Stulp y Sigaud 2015; Y. Zhang y cols. 2015) es una de las más usadas en regresión por su rapidez en la fase de entrenamiento y por proporcionar una buena aproximación gracias a que su arquitectura de red es simple. Su capa oculta consta de un conjunto de unidades de funciones de base radial, donde cada unidad estima la similaridad de los datos de entrada con los pesos de cada conexión.

Tienen aplicación en medicina, ingeniería y reconocimiento de patrones, entre otros. Entre las ventajas de esta metodología destacan su adaptatividad, ya que puede mejorar su capacidad mediante el aprendizaje, y su tolerancia a fallos. Además, no requieren suposiciones, son capaces de realizar predicciones y resolver problemas

no lineales complejos. Destacan por tener la habilidad de extraer una relación entre variables.

Algunas de las desventajas que presenta esta metodología es decidir el algoritmo de aprendizaje que se va a usar, la arquitectura, el número de neuronas por capa y el número de capas. Su rendimiento es muy dependiente de la selección de los datos de entrada, ya que si se tienen en cuenta datos innecesarios, pueden llegar a ser lentos (S. Wang y cols. 2015) desde el punto de vista computacional e incluso converger a óptimos locales. Por otro lado, destacan por ser una *caja negra*, por lo que el significado es difícil de interpretar y, consecuentemente, la extracción del conocimiento se complica.

1.1.2.2. Redes bayesianas

Las redes bayesianas (Huang y cols. (2007); Hüllermeier (2008); Sun (2013); Jarraya y cols. (2014); Yao y Liu (2014)) son herramientas para la representación del conocimiento bajo una incertidumbre y proporcionan una perspectiva del problema mediante un modelo gráfico probabilístico. Se basa en el modelo de Bayes, donde los atributos son independientes de forma condicional generando modelos que permiten representar distribuciones de probabilidad de alta dimensionalidad de forma eficiente mediante distribuciones condicionales de baja dimensionalidad.

Wright (1921) introduce por primera vez su representación que consiste en un grafo dirigido acíclico, donde los vértices son las variables y los enlaces entre los vértices indican la probabilidad de dependencia entre ellas. La figura 1.3 muestra un ejemplo de grafo acíclico dirigido compuesto por siete variables, donde X_1, X_2 y X_3 son variables independientes y el resto de ellas presentan al menos una dependencia de otra variable.

El aprendizaje de una red bayesiana se puede separar en dos tareas (Bouchaala y cols. 2010): 1) aprendizaje de la estructura (Heckerman y cols. 1995; Heckerman 1998), para identificar una topología de red, y 2) aprendizaje de parámetros (Jing y cols. 2011), que consiste en calcular los parámetros numéricos de una topología dada. Encontrar la mejor topología es una tarea ardua porque el espacio de búsqueda se incrementa exponencialmente con el número de variables. Por otro lado, el cálculo de los parámetros de la red bayesiana implica un tiempo computacional alto, sobre todo si se contemplan todos los datos en el cálculo.

El uso de esta técnica posee una serie de ventajas (Vehtari y Lampinen 1999), tales como el control de la complejidad de forma automática, ya que permite la regu-

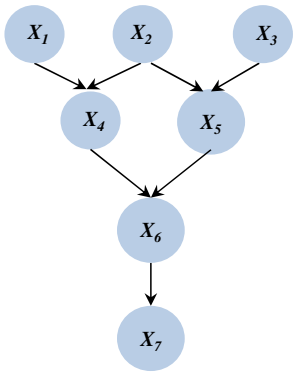


Figura 1.3.: Esquema de representación de un grafo acíclico dirigido

larización de los parámetros mediante los datos de entrenamiento, el uso de modelos jerárquicos con los parámetros de las distribuciones y las distribuciones predictivas para las salidas.

Tienen muchas aplicaciones en problemas reales, como por ejemplo, diagnóstico, visión automática y control. Además, resulta especialmente interesante el uso de esta metodología para representar incertidumbre probabilística. Sin embargo, pueden resultar imprecisas y poco interpretables.

1.1.2.3. Árboles de Decisión

Un árbol de decisión (J. Quinlan (1986); Uysal y Altay-Güvenir (1999); Kotsiantis (2013); Shi (2014)) es una herramienta que permite construir un modelo mediante el particionado del conjunto de datos. Los modelos son comprensibles por los humanos. Además, minimizan el ruido y los datos incompletos.

Un árbol de decisión está formado por un nodo raíz, nodos internos, nodos hojas y arcos. Un nodo interno contiene un test sobre algún valor de una de las variables. Un nodo hoja representa el valor que devuelve el árbol de decisión y las ramas establecen caminos atendiendo a la decisión tomada. El flujo de información recorre el árbol desde el nodo raíz hasta los nodos hoja. La figura 1.4 muestra un árbol de decisión, donde los nodos internos se representan por circunferencias y los nodos hoja por cuadrados.

El concepto de árboles de decisión tiene su origen en los años sesenta. A partir

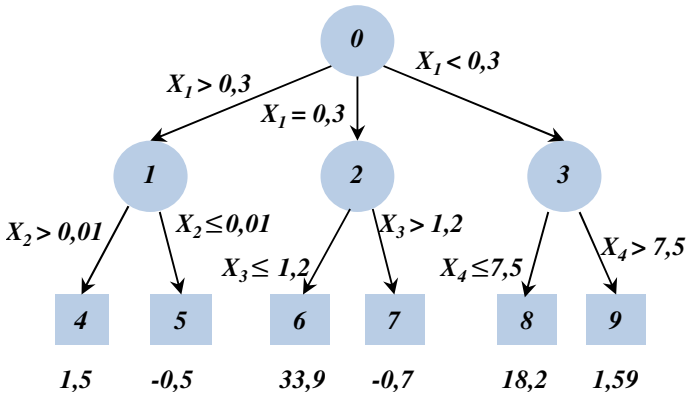


Figura 1.4.: Esquema de representación de un árbol de decisión en un problema de regresión

de aquí, comenzaron a surgir los algoritmos clásicos tales como CART (Breiman y cols. 1984; Stulp y Sigaud 2015) y M5 (R. Quinlan 1992). El algoritmo CART aplica árboles binarios a problemas de regresión, donde la inducción se realiza desde el nodo raíz hasta el nodo hoja y aplicando métodos de división y poda para generar nuevos individuos (Rokach 2016). El valor de la predicción en cada hoja es el valor medio de todos los ejemplos de un conjunto de entrenamiento que alcanzan esa hoja.

Por otro lado, el algoritmo M5 construye árboles donde sus hojas son modelos lineales multivariable. El incremento de la dimensionalidad del conjunto de datos en este algoritmo supone el crecimiento de los árboles modelo, lo que implica la necesidad de recursos computacionales. Existe una extensión de este algoritmo llamado M5Rules (Holmes y cols. 1999) que en cada iteración construye un árbol modelo donde sus nodos hojas son reglas. Mikut y cols. (2005) siguen esta línea de aprendizaje de reglas. Dada la utilidad tanto de M5 como de M5Rules, estos algoritmos se incluyen en WEKA[1], una herramienta implementada en Java que contiene algoritmos de aprendizaje automático y preprocesamiento de datos.

Otros trabajos de mayor actualidad que usan árboles de regresión son los de Fakhari y Moghadam (2013) y Ikonomovska y cols. (2015). El primer trabajo aplica una combinación de árboles de regresión y clasificación para la recuperación de imágenes atendiendo a la semántica. El segundo trabajo, implementa dos métodos de aprendi-

[1]Weka 3: Data Mining Software in Java. http://www.cs.waikato.ac.nz/ml/weka/

zaje de regresión en línea.

Para usar árboles de decisión tenemos que tener en cuenta los siguientes factores L. Chen y Wang (2007): el preprocesamiento de los datos antes de construir el árbol, la relación entre las variables, el intervalo de las variables continuas, la estrategia de poda del árbol, los métodos de extracción del conocimiento del árbol y la escalabilidad del árbol.

Esta herramienta presenta algunos problemas importantes. Uno de ellos es la limitación en el número de atributos por nodo. Si usamos un solo atributo por nodo, podemos obtener árboles de decisión idénticos. Si usamos dos o más atributos por nodo, la generación del árbol puede ser más costosa computacionalmente. El otro problema consiste en su inestabilidad (Breiman 1996): cambios pequeños en el conjunto de datos de entrenamiento producen diferentes elecciones a la hora de construir un nodo, lo cual puede producir cambios importantes, sobre todo si los cambios se realizan en nodos de niveles altos.

1.1.2.4. Sistemas Basados en Reglas Difusas

Existe otro tipo de metodología perteneciente al aprendizaje automático que permite tratar la imprecisión e incertidumbre del conocimiento y los datos, intentando encontrar la representación que más se ajusta a la realidad. Este es el caso de la lógica difusa (Shapiro (2005); Herrera y Lozano (2009)), que hace referencia al uso de valores difusos que capturan el significado de las palabras, el razonamiento humano y la toma de decisiones. Cada variable que interviene en el sistema está asociada a un dominio y se encuentra dividido en intervalos que contienen elementos de forma parcial denominados conjuntos difusos. Cada una de estas particiones tiene asociada un término lingüístico definido como conjunto difuso, recibiendo el nombre de etiqueta lingüística. Para determinar si el valor de la variable pertenece a un determinado conjunto difuso, se define una función de pertenencia que empareja a ese valor con un elemento perteneciente al intervalo [0, 1]. Las funciones de pertenencia pueden tener distintas formas: triangular, trapezoidal, gaussiana, sigmoidal o polinomial. Estas funciones se solapan entre sí (Contreras y cols. 2007), pero para garantizar la interpretabilidad del sistema, no es aconsejable solapar más de dos. Además, si el solapamiento se realiza en 0,5 estamos garantizando que los conjuntos difusos sean distintos. La figura 1.5 muestra las funciones de pertenencia más comunes en la literatura. Cada función de pertenencia se compone de cinco términos lingüísticos en la variable de intervalo [-2, 2] denominados de izquierda a derecha: Muy Pequeño

(MP), Pequeño (P), Mediano (M), Grande (G) y Muy Grande (MG). Normalmente, esta terminología se suele aplicar al aprendizaje de modelos basados en reglas difusas.

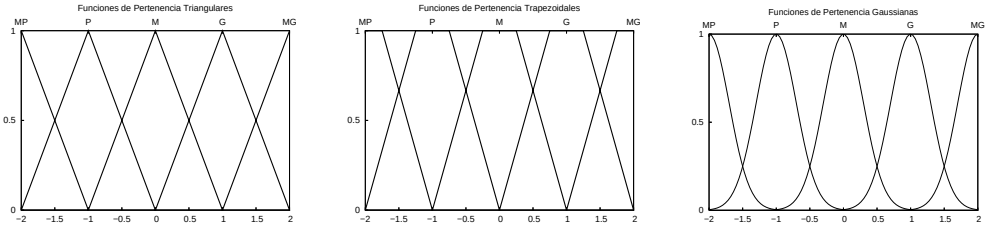


Figura 1.5.: Funciones de pertenencia más comunes en la literatura

Las reglas difusas tienen como misión establecer una relación lógica entre las variables del sistema. Además, su estructura de tipo SI-ENTONCES:

SI *la potencia es alta* **ENTONCES** *la temperatura subirá rápidamente*

Las reglas facilitan la comprensión en el análisis de los modelos y el solapamiento de las funciones de pertenencia asegura la generalización de casos no contemplados en las reglas. Por otro lado, los conjuntos difusos proporcionan una interfaz entre las variables numéricas de entrada/salida y las variables difusas de las reglas (L. A. Zadeh (1973)).

El objetivo principal de este tipo de sistemas es diseñar un modelo comprensible y fiable, porque sino no es útil. En este punto entran en juego dos conceptos muy importantes en el diseño que miden la calidad del sistema (Casillas, Cordón, y cols. (2003a); Gacto y cols. (2011)):

- Interpretabilidad, que se refiere a la capacidad para explicar el comportamiento de un sistema real de forma comprensible.
- Precisión, que nos indica lo fiel que es el modelo a la realidad que representa.

Destacar que en este tipos de sistemas, la precisión y la interpretabilidad son magnitudes inversamente proporcionales, es decir, cuanto mayor es la precisión del modelo, mayor es el número de reglas y, consecuentemente, se produce una pérdida

notable de interpretabilidad. En caso contrario, cuanto menor sea el número de reglas, el sistema ganará interpretabilidad, pero la representación de la realidad será escasa.

Cuando realizamos un modelizado difuso, hemos de tener en cuenta los siguientes aspectos (Hüllermeier (2008)):

- *Gradualidad.* Este concepto se refiere a que la transición entre la pertenencia y no pertenencia debe ser gradual y no abrupta (L. Zadeh (1965)). Las principales ventajas de los conjuntos difusos son la habilidad para expresar de una manera formal y bien definida conceptos graduales y las propiedades difusas, dado que se describen mediante predicados difusos combinados con conectivas lógicas generalizadas. Pero no solo resultan ventajosas para expresar conceptos por sí mismos, sino también para el modelizado de propiedades cualitativas.

- *Interpretabilidad.* Una de las motivaciones del diseño mediante conjuntos difusos es la de proporcionar una interfaz entre una escala numérica y simbólica formada por términos lingüísticos. Por consiguiente, los conjuntos difusos tienen la capacidad de relacionar patrones cuantitativos con estructuras de conocimiento cualitativo expresados en términos de lenguaje natural. Por otro lado, el uso de técnicas de modelizado lingüístico puede producir que los modelos sean subjetivos y dependientes del contexto. A pesar de esta desventaja, la imprecisión del lenguaje natural no es necesariamente perjudicial (L. A. Zadeh (1973)). Otra desventaja que puede presentar este tipo de modelizado se refiere a la transparencia de la complejidad del modelo. Una base de reglas formada por 40 reglas, donde existen siete antecedentes por regla, complica considerablemente la comprensión. De esta forma, los modelos simples pueden ser más interpretables, pero menos precisos, y viceversa. En este caso, existe un conflicto que concierne no solo al tamaño de los modelos sino a las métricas empleadas para modelizar la precisión.

- *Robustez.* Este término hace referencia a las variaciones de los datos. Generalmente, el método de aprendizaje se considera robusto si una pequeña variación en los datos observados apenas altera el modelo inducido o la evaluación de un patrón.

- *Representación de la incertidumbre.* Los datos que manejan los algoritmos de aprendizaje pueden ser imprecisos, incompletos o ruidosos. Aunque los datos

sean perfectos, la generalización de los datos y los procesos de inducción se pueden ver afectados por la incertidumbre.

- *Incorporación del conocimiento de referencia.* El modelizado difuso proporciona herramientas para crear conocimiento experto al que pueden acceder los métodos computacionales y, por lo tanto, incorporar un conocimiento de referencia en el proceso de aprendizaje.

- *Operadores de agregación.* Los operadores de agregación son operadores lógicos y aritméticos que representan las relaciones entre los atributos de los modelos y los patrones. Existe un gran repertorio de operadores lógicos (por ejemplo, las t-normas y las t-conormas) y aritméticos (por ejemplo, Choquet y Sugeno-integral), construyendo así, modelos más flexibles.

Por otro lado, la elaboración de modelos mediante sistemas basados en reglas difusas proporciona una serie de ventajas (Babuška (1998)) frente a otras técnicas inteligentes:

- Fusiona el procesamiento numérico y simbólico en un esquema común. Gracias a esto, se pueden usar tanto con algoritmos de aprendizaje basados en ejemplos (L. X. Wang (1994)) como con técnicas de regresión (Takagi y Sugeno (1985)).

- Permite interpretar los datos numéricos gracias a su estructura basada en reglas.

- Desarrolla diferentes modelos con distintos niveles de precisión, atendiendo al número de valores cualitativos considerados. Esta habilidad se adquiere por el uso de términos difusos en las reglas.

Todo esto nos indica que este enfoque es robusto y legible para elaborar modelos. Esta legibilidad es mayor si el modelizado es de tipo lingüístico, lo cual permitirá una mayor interpretabilidad del sistema (L. A. Zadeh (1975)). El tipo de reglas que se suele usar en este modelizado son las de tipo Mamdani (Alcalá y cols. (2000)).

En la literatura podemos encontrar que numerosos trabajos se han centrado más en encontrar modelos precisos en lugar de modelos interpretables (Casillas, Cordón, y cols. (2003b)). Esta tendencia ha ido cambiado y cada vez son más los investigadores que apuestan por modelos interpretables a la par que precisos (Alonso y Magdalena 2011). A principios del siglo XXI, para obtener un buen equilibrio entre precisión e

interpretabilidad, primero se obtenía un modelo inicial preciso y, posteriormente, se le aplicaba una metodología para mejorar la interpretabilidad sin perder demasiada precisión.

Para implementar un sistema basado en reglas difusas lingüístico *preciso* disponemos de diferentes esquemas (Casillas, Cordón, y cols. (2003a)): 1) Aprendizaje de la Base de Datos, que consiste en la extracción de la Base de Datos a priori por inducción a partir del conjunto de datos (Pedrycz (2001)); 2) Aprendizaje de la Base de Conocimiento, es decir, se incluyen un conjunto de técnicas que permiten el aprendizaje lingüístico difuso del modelo (Rojas y cols. (2000); Xiong y Funk (2006)); 3) Aprendizaje simultáneo, esto es, se realiza un aprendizaje conjunto de la Base de Datos y de la Base de Conocimiento (Xiong y Litz (2000)); y, 4) Ajuste de la Base de Datos, esto significa que dada una definición previa de la Base de Datos, existe un proceso que ajusta la forma de las funciones de pertenencia (Nauk y Kruse (1999)).

Con respecto a la *interpretabilidad*, se pueden hacer mejoras en un modelo preciso, reduciendo la dimensionalidad del problema con las siguientes técnicas (Casillas, Cordón, y cols. (2003b)):

- *Selección de variables.* La selección se puede realizar tanto en el subconjunto de variables de entrada, eliminando las que no son significativas (Chiu (1996); T. Hong y Chen (1999); Jin (2000); Casillas y cols. (2001); X. Hong y Harris (2001); H. Lee y cols. (2001); Xiong y Funk (2006); Tan y cols. (2008); Salimans (2012); Peng y Xu (2013)), o seleccionando las variables de entrada dentro de las reglas lingüísticas, que en este caso si no se considera una variable en una de las reglas no implica que el resto de las reglas no la consideren, es decir, el uso de una variable es local a una determinada regla lingüística (Castro y cols. (1999); González y Pérez (2001); Xiong y Litz (2000)).

- *Selección de reglas lingüísticas.* En este caso, se seleccionan un conjunto de reglas de una Base de Reglas predeterminada (Ishibuchi y cols. (1995); Yen (1999); Taniguchi y cols. (2001); Casillas y cols. (2005); Alcalá y cols. (2006); Alcalá, Alcalá-Fdez, Gacto, y Herrera (2007)) o se puede establecer una fusión de reglas (T. Hong y Lee (1999); Jayaram (2008)).

Estas metodologías pueden ser muy útiles en problemas de alta dimensionalidad, ya que descartan las variables menos relevantes del problema. Por otro lado, si elaboramos modelos de esta forma para conservar un equilibrio aceptable entre precisión

e interpretabilidad, este proceso puede llegar a ser tedioso. Actualmente, la tendencia es implementar este proceso mediante mecanismos de aprendizaje que sean capaces de controlar el equilibrio de estas magnitudes de forma automática, como por ejemplo, el aprendizaje evolutivo.

1.2. Sistemas Difusos Genéticos

El aprendizaje evolutivo es un campo del aprendizaje automático que tiene como objetivo solventar problemas de optimización. Por ello, ha despertado mucho interés a los investigadores, ya que se puede contemplar una extensa bibliografía en la literatura. A pesar de que no son capaces de competir con las técnicas heurísticas tradicionales de optimización, los algoritmos evolutivos son muy útiles ya que obtienen buenos resultados para numerosos tipos de problemas donde las heurísticas fallan con frecuencia.

Este tipo de algoritmos realizan una simulación de la evolución (Yao y Liu (2014)). Su política de evolución parte de una población inicial aleatoria que evoluciona mediante la selección de un conjunto de individuos a los que se les aplica unos operadores genéticos de forma probabilística para obtener nuevas generaciones. Se definen como técnicas de optimización, donde cada individuo representa una solución del espacio de búsqueda. Abarcan varias ramas (Beyer y Schwefel (2002)), como por ejemplo, estrategias de evolución (Bäck y cols. (2013)) , programación evolutiva (D. Fogel y Fogel (1996); G. Fogel (2012)), algoritmos genéticos (Sivanandam y Deepa (2008)) y programación genética (Garg y Tai (2012)).

Estos algoritmos tienen muchas aplicaciones en el mundo real ya que son muchos los dominios en los que es posible maximizar o minimizar determinados objetivos. Además, no requieren conocimiento del problema a resolver y operan en distintos puntos del espacio de búsqueda, pero pueden tardar en converger dependiendo de los parámetros de entrada (tamaño de la población, probabilidad de mutación, probabilidad de cruce, etc.) o pueden converger a un óptimo local. Además, este tipo de aprendizaje no aporta el conocimiento intrínseco de los datos.

Por otro lado, en la computación evolutiva podemos encontrar otros algoritmos denominados *Algoritmos Evolutivos para Optimización Multiobjetivo (AEOM)* (Deb 2001; Patel y Raghuwanshi 2010). Los AEOM han sido tema de interés durante casi 20 años. En la primera década, los investigadores se centraron más en el estudio de la selección de dominancia del Pareto y en las técnicas de nichos. Aquí destacan al-

goritmos tales como NPGA (Horn y cols. 1994), MOGA (Murata y Ishibuchi 1995), NSGA (Srinivas y Deb 1995) y SPEA (Zitzler y Thiele 1998). Sobre el año 2.000, aparecieron nuevas propuestas cuyo rendimiento era mayor, como SPEA-II (Zitzler y cols. 2001) y NSGA-II (Deb y cols. 2002). En esta segunda generación, los algoritmos incorporan técnicas de elitismo. Estos algoritmos son capaces de manejar problemas grandes, complejos y con un espacio de búsqueda de alta dimensionalidad. Sobre 2.003 (Zheng y Lei 2008), aparece una nueva etapa en donde los algoritmos intentan ser más eficientes con un mayor rendimiento. Estas estrategias incluyen métodos híbridos (Ishibuchi y cols. 2003; Arroyo y Armentano 2005), implementaciones paralelas (Jaimes y Coello 2005; Essabri y cols. 2006), co-evolución (Kleeman y Lamont 2006), evolución dinámica (Yang y cols. 2005), entre otros.

La fortaleza de estos algoritmos (R. Wang y cols. 2014; Ishibuchi y cols. 2014) se encuentra en su convergencia rápida (aproximación a la frontera del Pareto) y además cubren bastante bien el espacio de búsqueda para garantizar la diversidad. Este comportamiento viene determinado por equilibrio entre la exploración y la explotación del espacio de búsqueda. Además manejan estructuras de datos fáciles de representar y flexibles. Otra de las ventajas de estos algoritmos es que son capaces de optimizar más de un objetivo en competición simultánea.

Como comentamos en la sección anterior, los sistemas basados en reglas difusas proporcionan modelos ajustados a problemas del mundo real de forma precisa e interpretable por el ser humano a partir de datos numéricos, pero, como ya sabemos, el incremento de la precisión en este tipo de sistemas conlleva a un aumento del número de reglas del sistema, lo cual repercute negativamente en su interpretabilidad. Por el contrario, la interpretabilidad del sistema se mejora cuanto menor sea el número de reglas, pero esto hace que la precisión empeore.

Esto nos lleva a considerar esta situación como un problema de optimización, ya que hemos de maximizar la precisión y minimizar el número de reglas del sistema. En este caso, podemos utilizar técnicas de aprendizaje evolutivo, pero no podemos considerar cada magnitud por separado, porque nos llevaría a obtener sistemas muy precisos o sistemas muy interpretables. Está claro que el equilibrio entre ambas magnitudes ha de estar latente y por ello la mejor metodología aplicable en este caso son los AEOM. Esto se lleva aplicando desde principios de los años 90, ya que se empezaron a hibridar los sistemas difusos con la computación evolutiva (Alcalá y Nojima 2009; Casillas y Carse 2009; Fernández y cols. 2015). En un principio, el objetivo era el de usar la capacidad de la optimización de los algoritmos evolutivos para mejorar

la precisión de los sistemas difusos. De esta forma, se ajustaban automáticamente algunos elementos de la base de datos, de la base de reglas o del sistema de inferencia. El uso de algoritmos evolutivos en sistemas difusos proporcionan una codificación genérica independiente del problema con flexibilidad y capacidad de incorporar conocimiento.

Dado que la precisión y la interpretabilidad son magnitudes inversamente proporcionales, actualmente se ha realizado una extensión de los AEOM considerando estas como objetivos. Muchos autores aplican algoritmos evolutivos multiobjetivo para conseguir este equilibrio. Esta hibridación recibe el nombre de *sistemas difusos genéticos* (Herrera y Magdalena (1997); Cordón y cols. (2004); Herrera (2008); Casillas y Carse (2009)).

En la literatura (Casillas y cols. 2007) podemos encontrar el aprendizaje de sistemas difusos combinados con metaheurísticas pertenecientes al aprendizaje evolutivo (Herrera y Lozano (2009); Shukla y Tripathi (2012); Fernández y cols. (2015)) para optimizar distintos elementos del sistema. A este nivel, se introduce un nuevo concepto, el *metaaprendizaje*, que es una propiedad que adquiere el algoritmo de autoevaluar su propio comportamiento. Según Herrera en su trabajo de 2008, este metaaprendizaje se puede clasificar en dos grupos bien definidos:

- *Ajuste genético:* Si existe una base de conocimiento, se ajustan los parámetros del sistema basado en reglas difusas para mejorar su rendimiento, manteniendo la misma base de reglas. Tenemos tres posibilidades de adaptación:
  - *Funciones de pertenencia:* Se puede ajustar su forma (Alcalá, Alcalá-Fdez, Gacto, y Herrera 2007; Gacto y cols. 2010; Alcalá, Gacto, y Herrera 2011), el número de términos lingüísticos (Casillas y cols. 2005) y los parámetros de las funciones de pertenencia (Gacto y cols. 2009; Alcalá, Nojima, y cols. 2011). La figura 1.6 muestra una representación gráfica de este esquema de aprendizaje.
  - *Sistema de inferencia:* En el sistema de inferencia se usan expresiones parametrizadas con el objeto de conseguir una mayor cooperación entre las reglas difusas y modelos más precisos sin perder interpretabilidad (Alcalá-Fdez y cols. 2007).
  - *Métodos de defuzzificación:* Consiste en aplicar la función de defuzzificación a todo conjunto difuso de reglas inferido y computarlo mediante un operador de media ponderada (Kim y cols. 2002; Márquez y cols. 2010).

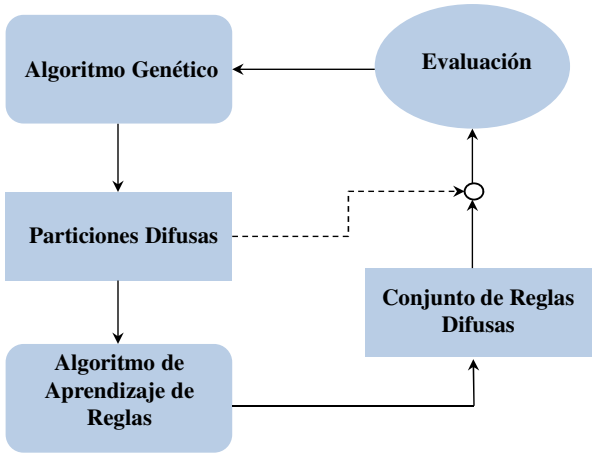


Figura 1.6.: Esquema de metaaprendizaje de las funciones de pertenencia

- *Aprendizaje genético:* Se aprenden los componentes de la base de conocimiento, además de los componentes del sistema basado en reglas difusas:
  - *Aprendizaje de reglas genético:* Se predefine la base de datos (número y distribución de las funciones de pertenencia en una variable del universo de discurso) y se realiza un aprendizaje de la base de reglas (Thrift 1991). Las reglas se pueden codificar de dos formas en un algoritmo genético: 1) Estrategia Pittsburg (Smith 1980), esto es, un conjunto de reglas se codifica como un cromosoma (Jong y cols. 1993); y 2) Un cromosoma constituye una regla, donde encontramos tres estrategias:
    - *Michigan.* Son sistemas basados en reglas o de paso de mensaje que usan aprendizaje por refuerzo, donde un algoritmo genético guía su comportamiento (Kovacs 2004).
    - *IRL (Iterative Rule Learning).* En este caso, los cromosomas compiten por cada ejecución de un algoritmo genético, eligiendo la mejor regla por ejecución. La solución final está formada por las mejores reglas que se han obtenido en las ejecuciones (Venturini 1993).
    - *GCCL (Genetic Cooperative-Competitive Learning).* En este caso, los cromosomas compiten y cooperan simultáneamente (Greene y Smith 1993; Giordana y Neri 1995).

- *Selección de reglas genética:* Si el número de reglas en la base de reglas es muy grande, actúa en detrimento de la interpretabilidad, por lo que hemos de eliminar las reglas irrelevantes, redundantes, erróneas y conflictivas (Yen 1999; Alcalá y cols. 2007; Casillas y cols. 2009; Gacto y cols. 2010).
- *Aprendizaje de la base de datos:* Implica el aprendizaje de la forma de las funciones de pertenencia, funciones de escala o la granularidad de las particiones difusas (Botta y cols. 2009). Existe otra opción que es embeber el aprendizaje de la base de reglas en el aprendizaje de la base de datos, de forma que cuando se aprende una base de datos en una iteración del algoritmo, la base de reglas se usa para derivar reglas y se utiliza una métrica que permita validad el error de la base de conocimiento obtenido (Cordón, Herrera, y Villar 2001).
- *Aprendizaje simultáneo de los componentes de la base de conocimiento:* En este caso, se aprende simultáneamente la base de datos y la base de reglas. El problema principal que presenta este tipo de aprendizaje es que se incrementa considerablemente el espacio de búsqueda por lo que el proceso de aprendizaje es difícil y lento (Hoimafar y Mccormic 1995).

Por otro lado, podemos aplicar otras técnicas de aprendizaje distintas a los AEOM, como por ejemplo las redes neuronales, denominándose esta hibridación sistemas neuro-difusos (Herrera y Lozano 2009; Fernández y cols. 2015). En este caso los conjuntos difusos y las reglas difusas se ajustan mediante una red neuronal. El ajuste abarca solo a los parámetros internos de una prefijada estructura, por lo que existen una serie de deficiencias en el aprendizaje: 1) la complejidad del aprendizaje aumenta con el incremento del número de variables; y 2) tiene dificultades en el aprendizaje de la estructura de las reglas, ya que solo se pueden aprender las funciones de pertenencia y los coeficientes de los consecuentes. Frente a esto, la aplicación de AEOM a sistemas basados en reglas difusas permite codificar y evolucionar los operadores de agregación de los antecedentes de la regla, diferentes semánticas de reglas, los operadores de agregación de la base de reglas y métodos de defuzzificación, además de tener en cuenta varios objetivos a optimizar simultáneamente, generando un conjunto de sistemas difusos no dominados, en donde existe un equilibrio entre precisión e interpretación.

A pesar de las ventajas que proporcionan los AEOM, hemos de tener en cuenta

que el incremento del número de objetivos degrada su potencial. Además, existen algunos problemas con un número excesivo de variables, reglas difusas o conjuntos difusos donde el modelo que implementan los algoritmos multiobjetivo no es todo lo interpretable que se pretende. En este caso, podemos recurrir a otros métodos, como los Sistemas Difusos Jerárquicos.

1.3. Sumario

Este capítulo describe un conjunto de técnicas pertenecientes al aprendizaje automático, de forma que ofrece una visión general de los conceptos indispensables y básicos usados en trabajo, con el objetivo de facilitar la comprensión al lector, encaminándolo desde conceptos más generales a los conceptos más específicos, todo ello, basado en un estudio del estado del arte del área.

Además, el capítulo describe algunos trabajos relevantes que podemos encontrar en la literatura. De esta forma, hemos comprobado el potencial de resolución de problemas que posee cada una de las técnicas.

2. Modelizado de Sistemas mediante Sistemas Difusos Jerárquicos

El razonar riguroso y preciso es el único remedio universal válido para todas las personas y disposiciones.

DAVID HUME (1.711–1.776), FILÓSOFO, ECONOMISTA, SOCIÓLOGO E HISTORIADOR ESCOCÉS

Si se aplica un Sistema Difuso (SD) convencional a problemas de alta dimensionalidad (es decir, aquellos con un alto número de variables de entrada), el número de reglas crece con respecto al número de entradas que recibe (Raju y cols. 1991; Joo y Lee 1999). De hecho, si tenemos n variables y k términos lingüísticos por variable, se requiere un total de k^n reglas para construir una base de reglas de tipo Mamdani completa para un SD. Si estudiamos esta expresión atentamente, de forma que cada una de estas variables tomen un valor numérico, podemos observar que si la variable n crece, el número de reglas del sistema crecerá exponencialmente. La ventaja de este suceso es que el sistema incrementará su precisión ya que al contener un mayor número de reglas será capaz de afinar en su inferencia. Por otro lado, pone de manifiesto una serie de consecuencias negativas:

1. La *transparencia e interpretabilidad* se reduce ya que ningún humano sería capaz de comprender cientos o miles de reglas difusas y parámetros. A esto se suma la complejidad que tendría la regla, ya que a mayor número de variables de entrada, mayor número de antecedentes en la regla (Mencar 2013).

2. El *sobreajuste* ocurre a menudo. Debido a que el número de datos disponibles en un problema es limitado, una gran cantidad de reglas y parámetros puede

producir un sobreajuste que reduce la generalidad del SD, es decir, el aprendizaje puede llegar a ser complicado o imposible.

3. Los *requerimientos de computación y memoria* pueden llegar a ser excesivos.

El desequilibrio que se produce entre precisión e interpretabilidad del SD se conoce en la literatura como "la maldición de la dimensionalidad" y ha sido estudiado por muchos investigadores. Para resolver este problema, se han propuesto varios enfoques, tales como la selección de variables (Chiu 1996; T. Hong y Chen 1999; Jin 2000; González y Pérez 2001; X. Hong y Harris 2001; H. Lee y cols. 2001; Xiong y Funk 2006; Tan y cols. 2008) o la reducción del conjunto de reglas (Ishibuchi y cols. 1995; Taniguchi y cols. 2001; Casillas y cols. 2005; Alcalá y cols. 2006; Alcalá, Alcalá-Fdez, Gacto, y Herrera 2007; Gacto y cols. 2009; Alcalá, Nojima, y cols. 2011).

La selección de variables consiste básicamente en que los métodos que se diseñan sean capaces de distinguir las variables más importantes e influyentes en la parte del antecedente de las reglas pertenecientes al SD. Con una selección adecuada de variables se reduce la complejidad del sistema tanto a nivel de regla como el número de reglas del sistema, pero es una tarea ardua ya que al reducir la dimensionalidad del problema, se está perdiendo información que puede ser importante en el proceso de inferencia del SD.

Los métodos de reducción del conjunto de reglas se centran, como su propio nombre indica, en reducir el número de reglas que contiene el SD. Para que el diseño de la metodología sea adecuado, la reducción de la base de reglas ha de poner de manifiesto las siguientes propiedades (Casillas y cols. 2009):

- *Consistencia:* No pueden existir dos reglas con igual antecedente y distinto consecuente.
- *Compleción:* La base de reglas debe ser capaz de inferir una salida para cualquier estado de las variables de entrada.
- *Compactibilidad:* Se debe obtener el menor número de reglas que representen la relación entre las variables de entrada y salida, sin que exista redundancia de reglas.
- *Sobregeneralización de reglas:* Una regla debe ser lo suficientemente general como para representar de una manera compacta la relación entre las variables

de entrada y salida, pero lo suficientemente específica como para evitar que cubra áreas de entrada sin datos.

Sin embargo, cuando el número de variables se incrementa considerablemente, esta clase de propuestas pueden no ser suficientes para resolver esta problemática. En la literatura existe una propuesta diferente para abordarla: los Sistemas Difusos Jerárquicos (SDJ). Esta metodología alternativa consiste en agrupar las reglas en módulos de forma que cada módulo calcula una solución parcial y cada una de estas soluciones parciales se usan por otros módulos posteriores para calcular la salida final del sistema. Con este enfoque se reduce el número de reglas del sistema considerablemente gracias al particionamiento jerárquico del dominio, ya que la base de reglas del sistema se descompone en varias bases de reglas pequeñas. En las siguientes secciones de este capítulo se describen las propiedades de un SDJ que permiten contestar a las preguntas:

- ¿Cómo determinamos la estructura jerárquica de un sistema difuso?
- ¿Cómo manejamos las variables intermedias?
- ¿Cómo obtenemos un sistema jerárquico difuso de forma eficiente?
- ¿Cómo enfocar el problema multidimensional a este modelizado?

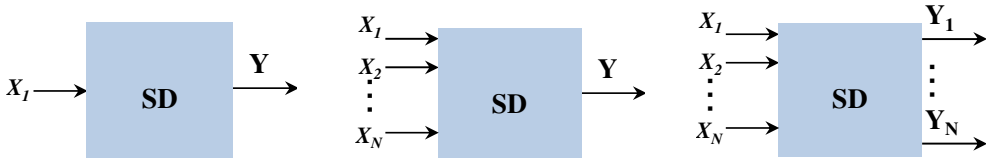
2.1. Diseño del Módulo

El diseño de un módulo difuso forma parte de la base para construir un buen diseño de un SDJ. Una *caja gris* (Babuška 1998) es la técnica de modelizado de SD que, por un lado, utiliza el conocimiento físico para modelizar las partes conocidas del sistema y, por otro, estima los parámetros del modelo en base a los datos de entrada disponibles para aproximar las partes más inciertas. Para ello, se utilizan metodologías inspiradas en sistemas biológicos e inteligencia humana, tales como lenguaje natural, reglas, redes semánticas o modelos cuantitativos. Gracias a este tipo de modelizado se obtienen soluciones con una mayor aproximación y bastante interpretables.

En términos generales, un módulo es una *caja gris* que recibe un conjunto de variables de entrada e infiere una o varias variables de salida en función del conocimiento que aprendió el mismo.

Un SDJ está compuesto por un conjunto de módulos difusos (también llamados subsistemas) enlazados entre sí, de forma que la salida de un módulo es la entrada de otro, y así sucesivamente. Podemos encontrar tres tipos distintos de módulos (figura 2.1) atendiendo al número de variables de entrada y salida que se tengan en cuenta:

- Módulo con una entrada y una salida (figura 2.1(a)).
- Módulo con varias entradas y una sola salida (figura 2.1(b)). D. Wang y cols. (2006), Y. Chen y cols. (2007), Y. Chen y Abraham (2010) y Zajaczkowski y Verma (2012) usan este tipo de módulos en sus trabajos. Por otro lado, podemos encontrar un módulo al que la literatura normalmente denomina Unidad Lógica Difusa (ULD) y que se puede clasificar en este punto, puesto que recibe dos variables como entrada e infiere una de salida. Las propuestas de Joo y Lee (1999), Shimojima y cols. (1995), L. Wang (1998), Joo y Lee (2002), M. Lee y cols. (2003), Gaweda y Scherer (2004), Jelleli y Alimi (2005), X. Zhang y Zhang (2006), Cheong (2007), Aja-Fernández y Alberola-López (2008), Jelleli y Alimi (2010), Masmoudi y cols. (2010) y Cala y Moreno-Velo (2010) hacen uso de este tipo de módulo.
- Módulo con varias entradas y varias salidas. Salgado (2005; 2008) trabaja con este tipo de módulos que se ilustran en la figura 2.1(c).



(a) Módulo con una entrada y una salida

(b) Módulo con varias entradas y una salida

(c) Módulo con varias entradas y varias salidas

Figura 2.1.: Tipos de módulos

Como se puede apreciar, dependiendo del tipo de problema que se intente resolver tenemos distintas alternativas a escoger en lo que al modelizado de módulo se refiere. Además destacar que cuanto mayor sea el número de variables, la complejidad modular será mayor dado que el modelizado difuso lingüístico y la estructura de la base de reglas es más compleja, lo que repercute seriamente en el tiempo de

computación del proceso de inferencia. Otros aspectos importantes a evaluar en el diseño y que condicionan la complejidad del módulo son el tipo de regla (Mamdani, TSK, etc.) y el método de aprendizaje que se use, pero no son el objetivo de estudio en este trabajo.

2.2. Modelizado de la Jerarquía

Aparte de distinguir distintos tipos de módulos, además es posible encontrar en la literatura diferentes tipos de estructuras jerárquicas atendiendo a distintos criterios. Un criterio evaluable a la hora de establecer una jerarquía modular difusa es la forma en la que la información se va a procesar. Así lo describe Torra en su trabajo del año 2002:

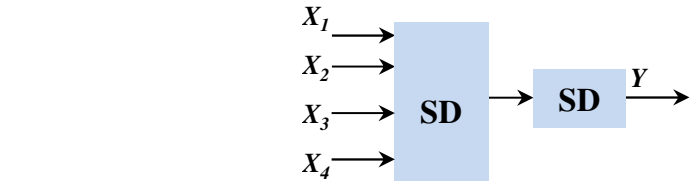
- *Estructura jerárquica de prioridades.* En este caso, los módulos con información específica se aplican al principio y los módulos menos específicos, en lo que a información respecta, sólo se aplican cuando el proceso de inferencia de los módulos más específicos falla. Normalmente se suele usar para modelos particularmente específicos. La ventaja que presenta este tipo de estructura jerárquica basada en prioridades es que la salida de los módulos con menor grado de especificidad se propaga sin que se vean afectados por ello los módulos más específicos. La figura 2.2(a) pone de manifiesto una estructura jerárquica con dos módulos ordenados por prioridad, en donde las variables de entrada inciden en el módulo con más prioridad.

- *Reglas encadenadas.* En este enfoque, la salida de un módulo se usa como variable de entrada de otro módulo, en donde la variable de salida del módulo anterior tomada como entrada se transforma para calcular la salida de ese módulo. Estos sistemas son los conocidos como sistemas multi-etapa. Las figuras 2.2(b) y 2.2(c) son dos ejemplos de estructuras jerárquicas que se clasifican en esta categoría.

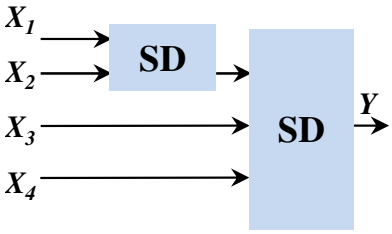
- *Uso de meta-conocimiento.* Al igual que en el punto anterior, existe un encadenamiento de módulos mediante variables de salida, pero en este caso la variable de salida de un módulo que es la entrada de otro se usa para modificar las reglas del módulo en donde incide. El procedimiento de modificación de las reglas del módulo consiste en el uso de la variable de entrada para fijar algunos

parámetros de la base de reglas. Joo y Lee (1999) ponen de manifiesto esta metodología y aconsejan que sea usada sólo en una jerarquía de dos niveles. La figura 2.2(d) muestra un ejemplo de este tipo de modelizado jerárquico.

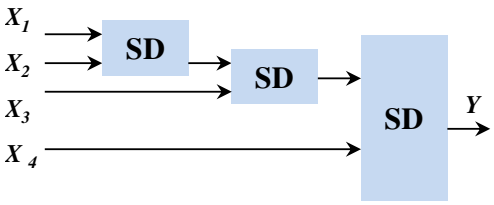
- *Combinación de salidas.* En esta metodología, las reglas de los módulos de nivel superior dictan cómo se combinan los resultados de los niveles más bajos (figura 2.2(e)). Para combinar las salidas de los módulos de bajo nivel, los módulos de alto nivel se pueden definir mediante procedimientos distintos a las reglas difusas (Hagras y cols. 1999), como por ejemplo, métodos de fusión de información.



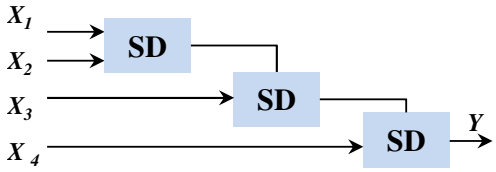
(a) Estructura jerárquica de prioridades



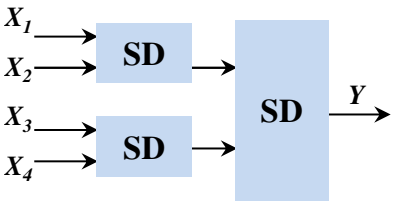
(b) Reglas encadenadas



(c) Reglas encadenadas



(d) Uso de meta-conocimiento



(e) Combinación de salidas

Figura 2.2.: Algunas estructuras jerárquicas difusas para cuatro variables según Torra (2002)

Por otro lado, Duan y Chung (2002) y Gaweda y Scherer (2004) coinciden en considerar la existencia de tres estructuras básicas multi-capa que a continuación se describen (figura 2.3, donde el subíndice de cada variable indica número de variable y el superíndice representa la capa a la que pertenece la variable de entrada a un módulo de la siguiente capa):

- *SDJ Incremental:* En esta estructura, las variables de entrada se encuentran distribuidas entre la primera capa, las capas intermedias y la última capa. El proceso de inferencia de un módulo está basado en las salidas de los módulos de las capas anteriores y en las variables de entrada asignadas a ese módulo. La estructura realiza un razonamiento incremental realizando una aproximación en las primeras capas y completando la misma con más factores (más variables de entrada) a medida que la información avanza por las capas de la estructura jerárquica hasta que realiza la inferencia final. La figura 2.3(a) ilustra este tipo de jerarquía.

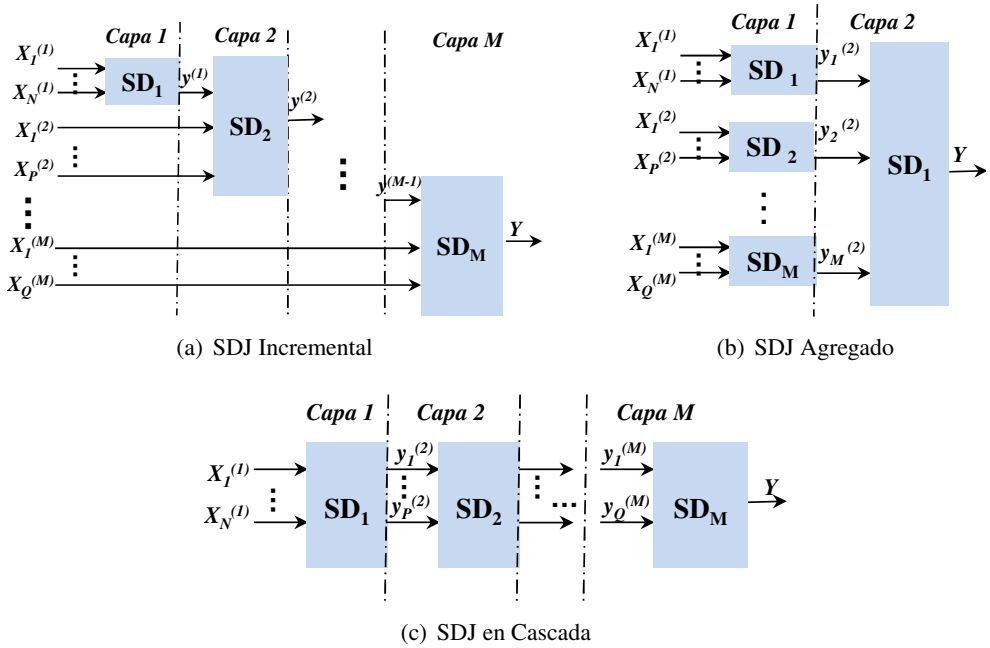
- *SDJ Agregado:* Las variables de entrada solo pueden acceder a una primera capa de razonamiento compuesta por un conjunto de módulos. Las variables de salida de la primera capa son las variables de entrada a la siguiente capa y así sucesivamente. Esta estructura se puede observar en la figura 2.3(b).

- *SDJ en Cascada:* Este tipo de estructura es el que más se parece al razonamiento humano. Todas las variables de entrada son recibidas por el primer módulo de la primera capa. El resultado del razonamiento alimentará al modulo de la siguiente capa y así sucesivamente capa por capa como refleja la figura 2.3(c).

Duan y Chung (2002) aconsejan una hibridación de estas jerarquías si se necesita un sistema más sofisticado para resolver el problema.

Finalmente, en la literatura encontramos la taxonomía de Aja-Fernández y Alberola-López (2008) que es más general, ya que todas las estructuras anteriormente descritas pueden ser clasificadas atendiendo a la forma de su jerarquía en:

- *Sistema Difuso Jerárquico en Serie (SDJS):* La entrada a un módulo es la variable de salida del módulo previo, además de tener como entrada un conjunto de variables externas (M. Lee y cols. 2003; Cheong 2007; L. Wang 1999; Aja-Fernández y Alberola-López 2008; Zeng y cols. 2008; Benítez y Casillas 2009; Zajaczkowski y Verma 2012) (figura 2.4(a)).



(a) SDJ Incremental

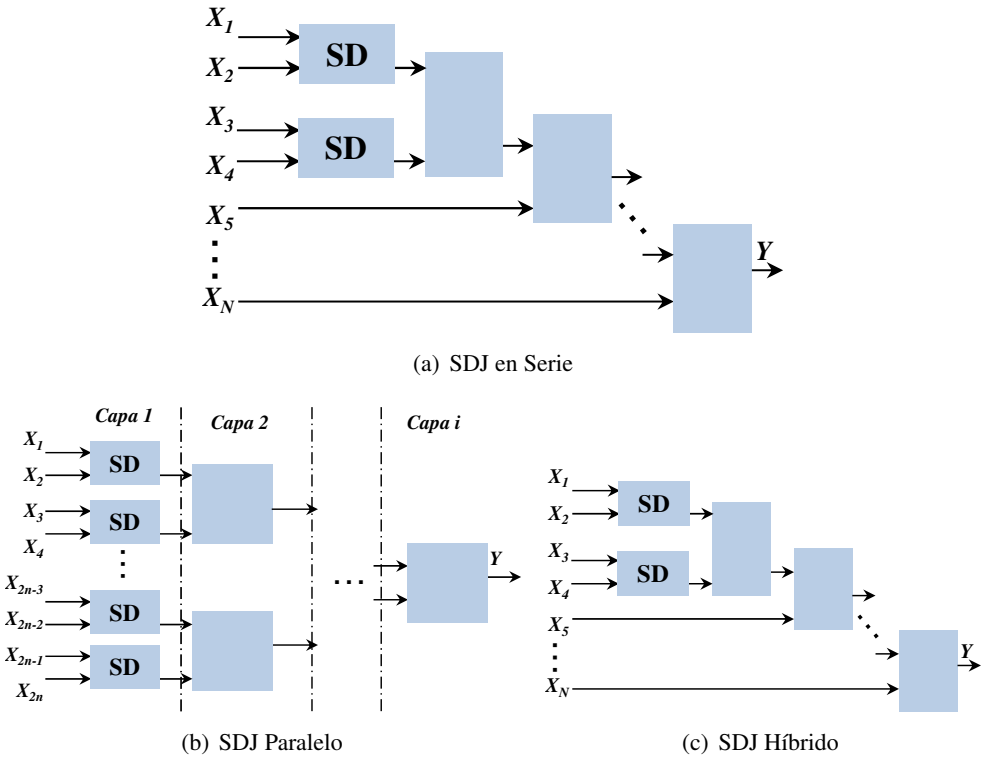
(b) SDJ Agregado

(c) SDJ en Cascada

Figura 2.3.: Tipos de estructuras jerárquicas según Duan y Chung (2002) y Gaweda y Scherer (2004)

- *Sistema Difuso Jerárquico Paralelo (SDJP):* Este sistema se organiza en capas (figura 2.4(b)). La primera capa está formada por un conjunto de módulos que reciben variables de entrada. Cada variable se usa como una sola entrada a un módulo. Las salidas de los módulos de la primera capa son las entradas de los módulos que constituyen la siguiente capa, y así sucesivamente. Destacar que podría existir una operación de agregación para combinar las salidas de una capa (Holve 1998; Joo y Lee 1999; Salgado 2005; Jelleli y Alimi 2005; M. Lee y cols. 2003; Salgado 2008; X. Zhang y Zhang 2006; Aja-Fernández y Alberola-López 2008; Joo y Sudkamp 2009; Jelleli y Alimi 2010).

- *Sistema Difuso Jerárquico Híbrido (SDJH):* Este tipo de SDJ es una mezcla de los dos anteriores (Joo y Lee 2002; D. Wang y cols. 2006; Y. Chen y cols. 2004, 2007; Aja-Fernández y Alberola-López 2008; Y. Chen y Abraham 2010; Cala y Moreno-Velo 2010) (figura 2.4(c)).



(a) SDJ en Serie

(b) SDJ Paralelo

(c) SDJ Híbrido

Figura 2.4.: Tipos de estructuras jerárquicas según Aja-Fernández y Alberola-López (2008)

La figura 2.4 muestra esta clasificación de estructuras jerárquicas. Así mismo, en la propia estructura jerárquica se pueden distinguir dos tipos de variables:

- *Variable exógena:* Es una variable externa al propio sistema (variable del problema) y no es inferenciada por ningún módulo.
- *Variable endógena:* Se define como aquella variable que tiene su origen en la inferencia realizada por un módulo y hace de enlace con otro, es decir, es la variable que enlaza dos módulos.

A pesar de la reducción de la complejidad en los problemas, hemos de ser conscientes de algunas desventajas que conllevan la utilización de este tipo de herramientas:

1. *La agrupación de variables por módulo.* Este es otro aspecto importante a considerar, ya que dependiendo de las variables que se usen en el SDJ y su distribución en el mismo, influyen en la precisión e interpretabilidad del propio sistema. Como se ha venido comentando, cuantas más variables de entrada tenga un módulo, habrá más precisión en la inferencia, pero la interpretabilidad es menor. En este aspecto, además puede influir bastante el nivel de dificultad del problema, es decir, si la correlación entre las variables de entrada es mayor, es más fácil establecer una dependencia entre variables, en otras palabras, inferir unas variables en función de otras.

2. *Pérdida de información global.* Al establecer la estructura jerárquica y asignar un módulo determinado a un conjunto de variables, dicho módulo opera sólo con las variables asignadas. De esta forma, cada módulo trata una parte de la totalidad del problema y pierde precisión, ya que no está considerando el resto de interdependencias que también proporcionan información. Destacar en este punto, que la pérdida de precisión depende además del método de aprendizaje de la base de reglas de cada módulo. Si el aprendizaje del conocimiento es bueno, la pérdida de información global será menor.

3. *La propagación del error a través del SDJ.* Este problema está intrínsecamente relacionado con el de la pérdida de información global. Surge como consecuencia de la inferencia de las variables de salida en los distintos módulos. Dado que la variable de salida de un módulo puede ser precisa aunque no exacta ya que son metodologías de aproximación, la salida inferenciada presentará un pequeño error en las primeras capas, que se puede ir incrementando a medida que las variables inferenciadas avanzan por el SDJ hasta llegar al módulo de salida del sistema. Esto es lo que se denomina el acarreo del error. Hemos de tener en cuenta que cuanto mayor sea el número de módulos y de capas en el SDJ, el error acarreado se puede incrementar con independencia del tipo de jerarquía que se esté considerando. En este sentido, se debe ser cauto en el diseño de la estructura jerárquica para que el error que se propaga sea el menor posible. En la literatura, trabajos como el de Maeda (1996) y D. Wang y cols. (2006) indican esta carencia de estas herramientas. Es más, el trabajo de Torra (2002) aconseja que en los SDJ basados en meta-conocimiento su estructura no esté formada por más de dos capas ya que las variables inferenciadas en una capa influyen activamente en la creación de la base de reglas de la capa

posterior.

A pesar de esto, estos problemas no han de suponer un obstáculo en el uso de este tipo de sistemas si se tienen en cuenta a la hora de diseñar un SDJ, ya que podemos obtener una estructura bastante aceptable en términos de precisión mejorando la interpretabilidad. Hemos de usar este tipo de sistemas siempre que queramos una reducción de la complejidad de nuestro problema porque este tipo de estructuras nos ayuda a entender mejor el conocimiento intrínseco de los datos que manejamos.

La tabla 2.1 es una tabla resumen de las ventajas e inconvenientes que conllevan el uso de estructuras jerárquicas en el modelizado de un SD con respecto a la interpretabilidad y la precisión. A modo de resumen, dado que en este tipo de estructuras las reglas son más simples, debido a que el número de variables de entrada en cada módulo de la jerarquía es menor que en un SD, y que el número de reglas es menor, porque no presenta una tendencia exponencial a la hora de generar las reglas, la repercusión de estos factores en la interpretabilidad del sistema es muy positiva ya que el sistema es más transparente. La precisión se mantiene o incluso mejora gracias a las variables sintéticas, lo cual es comparable al método de análisis de componentes principales (Mackiewicz y Ratajczak 1993): ambos reducen la dimensionalidad del conjunto de datos, ordenándolas por importancia. Por otro lado, si las variables endógenas del sistema jerárquico no pertenecen al problema, aumenta la complejidad del problema reduciendo su transparencia. Además, cuanto mayor sea el número de módulos del sistema jerárquico, mayor será el acarreo y, consecuentemente, empeora la precisión. La precisión también se ve afectada por la correlación de las variables, de forma que cuanto menos correladas sean, no se podrá establecer una relación entre ellas y la precisión empeorará.

Tabla 2.1.: Resumen de las principales ventajas e inconvenientes del uso de estructuras jerárquicas

	Ventajas	*Inconvenientes*
Interpretabilidad	- Reglas más simples - Menor número de reglas - Transparencia	- Si las variables endógenas no son propias del problema, existe dificultad de interpretar con varias capas
Precisión	- Se mantiene o mejora - Más precisión con variables endógenas sintéticas	- Acarreo del error - Correlación de las variables de entrada

2.3. Modelizado de la Base de Reglas

Como se ha descrito en la sección anterior, cada módulo de un SDJ aborda un subconjunto de variables, en otras palabras, no trata el problema por completo. Gracias a la descomposición de un SD en una jerarquía de sistemas difusos, la complejidad de cada módulo se ve significativamente reducida debido al hecho de que las reglas que pertenecen a cada módulo son más simples que las reglas de un SD con un sólo módulo, ya que el número de variables de cada módulo es bajo. Además, hemos visto que se pueden distinguir distintos tipos de variables dentro de un SDJ. En este sentido, en varias propuestas se ha estudiado la mejor forma para tratar las variables que enlazan los módulos, es decir, las variables endógenas. La forma más extendida entre los SDJ es la creación de nuevas variables sin significado lingüístico (M. Lee y cols. 2003; Salgado 2005; Cheong 2007; Salgado 2008; L. Wang 1999; Zeng y cols. 2008; Holve 1998; Joo y Lee 1999; Jelleli y Alimi 2005; X. Zhang y Zhang 2006; Joo y Sudkamp 2009; Joo y Lee 2002; D. Wang y cols. 2006; Y. Chen y cols. 2004; Aja-Fernández y Alberola-López 2008; Y. Chen y cols. 2007; Y. Chen y Abraham 2010; Jelleli y Alimi 2010; Zajaczkowski y Verma 2012), esto es, un tipo de variable que es difícil de comprender desde el punto de vista humano, ya que el valor de la variable es de tipo cuantitativo. Este tipo de variables se suelen obtener del resultado del cálculo con operaciones matriciales, mediante el uso de técnicas de optimización de parámetros de la base de reglas o funciones de aproximación numérica gracias a las reglas Takagi-Sugeno (TS).

El trabajo de Gaweda y Scherer (2004) genera las variables de salida de cada módulo mediante números difusos. En este trabajo, el término "subconjuntos compartidos" aparece como una métrica de similaridad basada en el grado de inclusión de A' en A y, al mismo tiempo, en el grado de inclusión de A en A', siendo A un conjunto difuso y A' un conjunto de números difusos.

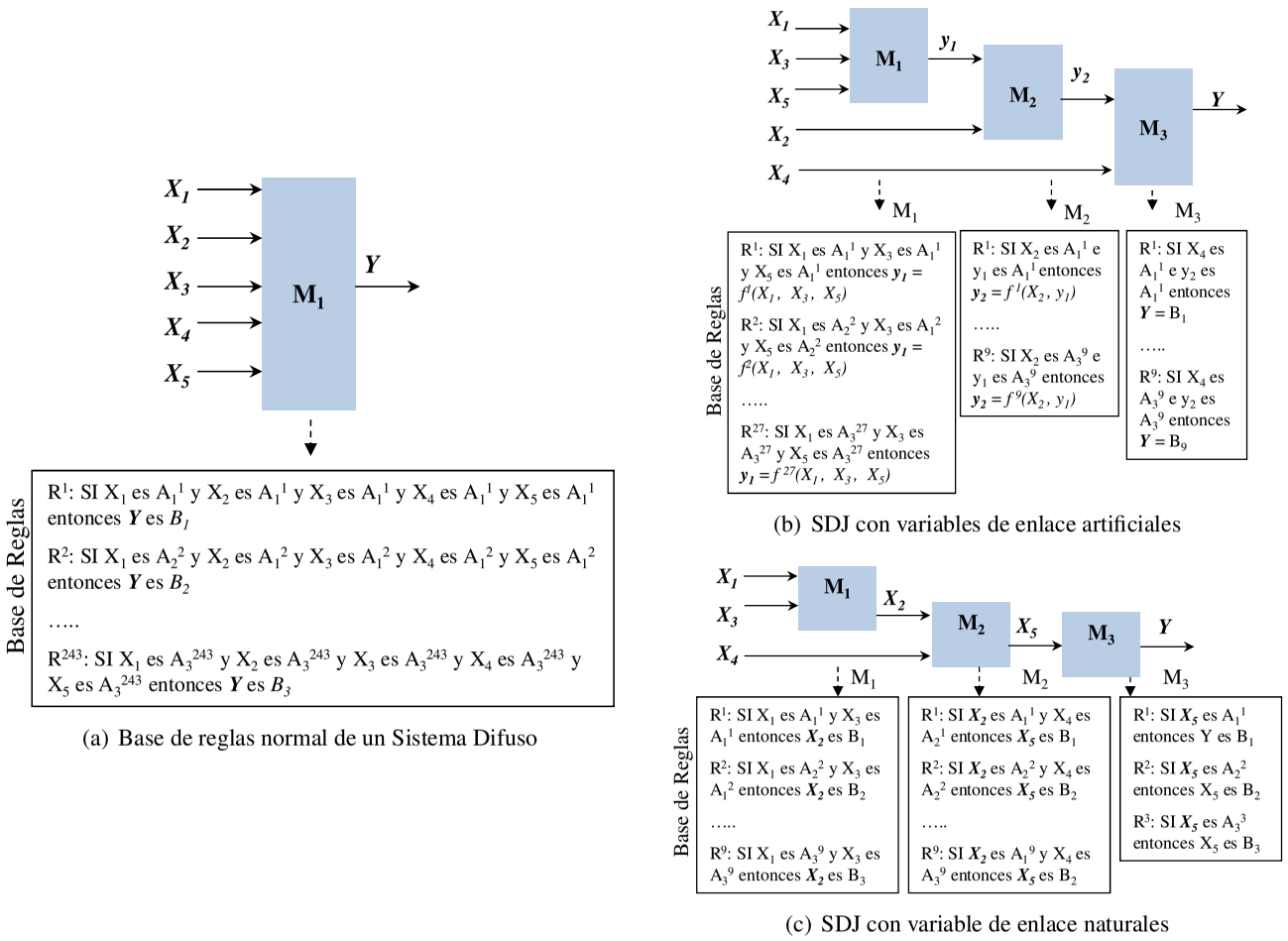
Jelleli y Alimi (2005) proponen un algoritmo iterativo que asocia linealmente las entradas por parejas $(x_{L,i}, x_{L,i+1})$, construyendo una primera capa con un conjunto de ULD, siendo L la L-ésima capa e i el i-ésimo ULD. Cada ULD genera una variable de salida, $y_{L,i}$, que se asocian en parejas $(y_{L,i}, y_{L,i+1})$. La variable de salida no dominada, $d_{L,i}$, es la que se selecciona de cada pareja. A continuación, estas variables son las entradas de la siguiente capa. Este método es un bucle que recorre todos los niveles de la jerarquía.

Por lo tanto, atendiendo a las distintas formas de generación de variables de salida

de los módulos, se pueden distinguir tres tipos de bases de reglas:

1. Una base de reglas convencional de un SD (figura 2.5(a)).
2. Un SDJ con variables artificiales para enlazar módulos. Este tipo de variables son nuevas variables de salida creadas por la inferencia de un módulo mediante funciones matemáticas. La figura 2.5(b) muestra que las variables y_1 e y_2 son variables artificiales.
3. Un SDJ con variables naturales para el enlace de módulos. Las variables naturales se corresponden con variables pertenecientes al propio problema. En la figura 2.5(c), las variables X_2 y X_5 son variables naturales.

Una vez que se conocen los tipos de variables de salida que podemos utilizar para nuestro modelizado de la base de reglas, es interesante comprobar la complejidad que pueden proporcionar cada una de ellas. Por ello, calcularemos el número de reglas generado por cada uno de los sistemas de la figura 2.5. Si tenemos cinco variables y tres términos lingüísticos por variable, el SD de la figura 2.5(a) require hasta un máximo de $3^5 = 243$ reglas para generar un SD completo; el SDJ de la figura 2.5(b) tiene 45 reglas (3^3 reglas en M_1, 3^2 reglas en M_2, y 3^2 reglas en M_3), y el SDJ de la figura 2.5(c) genera 21 reglas (3^2 reglas en M_1, 3^2 reglas en M_2, y 3^1 reglas M_3). En términos generales, se puede apreciar que la reducción del conjunto de reglas de un SD con respecto a un SDJ es una función lineal del número de variables. Como se puede observar, la base de reglas del SDJ de la figura 2.5(c) es la que obtiene el menor número total de reglas. Si se compara el SDJ de la figura 2.5(c) con el SD de la figura 2.5(a), se puede comprobar que la distribución de variables en módulos de una jerarquía de SD hace que un SDJ disminuya el número total de reglas y, además, reduce la complejidad de cada regla, ya que como se puede observar, el número de variables por regla es menor en un sistema que implementa una base de reglas con variables de salida naturales. Si comparamos el SDJ de la figura 2.5(c) con el SDJ de la figura 2.5(b), el número de reglas de la figura 2.5(c) es menor que el de la figura 2.5(b) porque no genera variables artificiales adicionales para enlazar los módulos. Como se puede apreciar en estos ejemplos, un SDJ que implementa una base de reglas con variables naturales es capaz de disminuir el número de reglas con respecto a un SD hasta en un 91 % y hasta en un 53 % en relación al SDJ con variables artificiales. Estas cifras se traducen en una disminución de la complejidad y una mejora en la interpretabilidad del sistema, por lo que merece la pena apostar por este tipo de arquitectura.



(a) Base de reglas normal de un Sistema Difuso

(b) SDJ con variables de enlace artificiales

(c) SDJ con variable de enlace naturales

Figura 2.5.: Tipos de bases de reglas en un Sistema Difuso

2.4. Antecedentes en Sistemas Difusos Jerárquicos

En la literatura encontramos varias publicaciones sobre SDJ y topologías. Algunas de estas propuestas tratan de resolver problemas mediante SDJ, pero no tienen en cuenta el aprendizaje de la estructura jerárquica (L. Wang 1998; Joo y Lee 1999; L. Wang 1999; Rattasiri y Halgamuge 2000; Joo y Lee 2002; Gaweda y Scherer 2004; M. Lee y cols. 2003; X. Zhang y Zhang 2006; Cheong 2007; Zajaczkowski y Verma 2012). Claramente, no nos interesa analizar cada uno de estos casos. De hecho, un buen diseño de una estructura jerárquica es tan importante como la precisión del sistema, ya que como venimos comentando, si la jerarquía está formada por un número elevado de módulos, el error producido puede incrementarse considerablemente como consecuencia del acarreo en el proceso de inferencia a través de las capas de un SDJ.

Por ello, esta sección realiza una revisión de los principales enfoques de aprendizaje de estructuras jerárquicas para así estudiar qué tipo de modelizado realizan en cada uno de los problemas que manejan. La descripción de cada método se realiza mediante un orden cronológico sin mencionar los enfoques que no implementan el aprendizaje de la estructura jerárquica.

La tabla 2.2 muestra un resumen de los enfoques que realizan un aprendizaje de estructura jerárquica. La columna *Año* en la tabla hace referencia al año de publicación del trabajo, *Autor* indica el autor o autores que han elaborado el trabajo, *Funciones pertenencia* especifica el tipo de funciones de pertenencia usadas en el trabajo, *Tipo reglas* se refiere al tipo de reglas que usa el trabajo en el proceso de inferencia (Mandani o TSK), *SDJ* es el tipo de topología que aprende, es decir, serie (S), paralela (P) o híbrida (H), *Variable enlace* muestra el tipo de variable de conexión entre módulos, que puede ser artificial (A) o natural (N), y *Aprendizaje* hace alusión a la forma del aprendizaje de la estructura jerárquica.

El algoritmo de Shimojima y cols. (1995) aprende a generar una estructura jerárquica híbrida mediante un AG y el método de retropropagación. Usa funciones de pertenencia de base radial exponencial que se ajustan mediante aprendizaje supervisado con el método de descenso del gradiente para que se converja a la solución de forma rápida y precisa. Las reglas de la estructura jerárquica híbrida son Takagi-Sugeno-Kang (TSK).

Y. Chen y cols. (2004) diseñaron un algoritmo con la capacidad de aprender estructuras jerárquicas híbridas mediante Ant Programming y realiza un ajuste de los

Tabla 2.2.: Resumen de los principales enfoques de aprendizaje de estructuras jerárquicas

Año	*Autor*	*Funciones pertenencia*	*Tipo reglas*	*SDJ*	*Variable enlace*	*Aprendizaje*
1995	Shimojima et al.	Base Radial	TSK	H	A	Algoritmo Genético, Retropropagación
2004	Chen et al.	Exponencial	TSK	H	A	Ant Programming
2006	Wang et al.	Triangular	Mamdani	H	A	Gradiente Descendente
2007	Chen et al.	Exponencial	TSK	H, S	A	Programación Evolutiva Incremental Probabilística
2008	Aja-Fernández et al.	Triangular	Mamdani	P, H	A	Inferencia Rápida usando Matrices de Transición
2009	Joo y Sudkamp	Triangular	TSK	P	A	Voraz
2010	Jelleli y Alimi	Gaussiana	TSK	P	A	Toma de Decisiones
		Gaussiana,		S		
2010	Cala y Moreno-Velo	Triangular,	n/d	P	A	Gradiente Descendente
		B-Spline		H		
2012	Zajaczkowski y Verma	Triangular	Mamdani	H	A	Algoritmo Evolutivo
2014	Zangh et al.	Triangular	Mamdani	S	A	Algoritmo Genético
2015				S		
2015	Nuestras propuestas	Triangular	Mamdani	P	N	Algoritmo Genético Multiobjetivo
2015				H		

parámetros de las reglas mediante el algoritmo de enjambre de optimización de Partículas. En la codificación de los SDJ utiliza una estructura de árbol, en donde cada hormiga construirá y modificará los árboles atendiendo a la cantidad de feromona que se haya depositado en cada nodo del árbol; cada nodo posee una tabla en donde memoriza el ratio de feromona. Con respecto a la base de reglas, usa reglas de tipo TSK con funciones de pertenencia exponencial.

D. Wang y cols. (2006) usaban el método del Gradiente Descendente para aprender la estructura jerárquica. La funciones de pertenencia son triangulares y hace uso de reglas de tipo Mamdani.

Y. Chen y cols. (2007) mejoraron su versión de 2004, aprendiendo estructuras jerárquicas híbridas por medio de Programación Evolutiva Incremental Probabilística y realizando un ajuste de los parámetros de las reglas mediante programación evolutiva alternativamente y conservando la codificación en árbol de las soluciones. La forma de las funciones de pertenencia se aprenden con técnicas adaptativas y usa reglas TSK.

Aja-Fernández y Alberola-López (2008) desarrollaron una nueva forma de aprendizaje de la base de reglas por medio un procedimiento con matrices llamado Inferencia Rápida usando Matrices de Transición, que se propone como una metodología para conseguir inferencias con el modelo aditivo estándar. La principal ventaja de este método reside en poder descomponer un SD en un SDJ mediante una matriz de descomposición o factorización, aunque no garantiza que el SDJ final posea un número menor de reglas que el SD del que se parte. Hace uso de reglas de tipo Mamdani y el aprendizaje de la base de reglas puede usar en cualquier estructura jerárquica, ya sea serie, paralela o híbrida.

Joo y Sudkamp (2009) diseñaron un sistema jerárquico difuso en paralelo de dos capas. El modelizado del SDJ se implementa de forma que la primera capa está formada por un conjunto de módulos que contienen bases de reglas linealmente independientes. Las combinaciones lineales de las bases de reglas se utilizan en la segunda capa. Usa reglas Mamdani en la primera capa, reglas TSK en la segunda capa y aplica una reducción de reglas estableciendo una relación de dependencia entre los SD de la primera capa. Las funciones de pertenencia son triangulares. Este enfoque se describirá detalladamente en el apéndice A.3.

Jelleli y Alimi (2010) publicaron un algoritmo de aprendizaje de estructuras jerárquicas paralelas. Las variables se agrupan en módulos mediante aprendizaje del comportamiento difuso de los datos. Después, se eligen los módulos de la siguiente

capa mediante una Toma de Decisiones ponderada. Las funciones de pertenencia son gaussianas y las reglas de tipo TSK.

Cala y Moreno-Velo (2010) proponen un algoritmo recursivo que consiste en realizar una búsqueda entre todas las posibles estructuras jerárquicas dadas n variables de entrada, en donde cada módulo es una ULD. De todas las estructuras jerárquicas posibles con n variables, la metodología selecciona la que mejor se adapta al conjunto de datos de entrenamiento. Para evaluar la capacidad de aproximación de cada estructura usa un algoritmo de optimización paramétrica basado en el método de Gradiente Descendente. Además, permite usar distintas funciones de pertenencia (Gaussianas, Triangulares y B-Splines) sin importar el tipo de estructura jerárquica. Por otro lado, los autores indican que este diseño no es factible si el número de variables de entrada es mayor de siete u ocho ya que el espacio de búsqueda es demasiado grande y una búsqueda exhaustiva no es viable.

La propuesta de Zajaczkowski y Verma (2012) realiza un estudio del impacto de las distintas topologías usando un Algoritmo Evolutivo (AE) para el aprendizaje de la estructura jerárquica de dos, tres y cuatro capas. Para crear la estructura jerárquica establece la importancia de las variables de entrada y sus interacciones. Las variables deben ser agrupadas atendiendo al grado de influencia en la salida del sistema y sus inter-relaciones. Usa funciones de pertenencia triangulares y reglas de tipo Mamdani aplicadas a la resolución del sistema de péndulo invertido.

El trabajo de X. Zhang y cols. (2014) es uno de los últimos que se pueden encontrar en el área de los SDJ. Los autores implementan un AG para el aprendizaje de la estructura jerárquica en serie, utilizando funciones de pertenencia triangulares, reglas de tipo Mamdani y realizando selección de variables.

Todos estos trabajos coinciden en la creación de variables artificiales para enlazar los distintos módulos del SDJ que modelan. Además destacan como conclusión la efectividad de los SDJ frente a un SD, ya que los distintos niveles del SDJ hacen que el número de reglas y operaciones difusas durante el proceso de modelizado puedan reducirse significativamente con respecto a su homólogo de una sola capa.

2.5. Sumario

El objetivo de este capítulo es el de introducir el modelizado de sistemas mediante Sistemas Difusos Jerárquicos. Se ha presentado cada uno de los componentes que se han de tener en cuenta a la hora de realizar el modelizado, tanto a nivel de

módulo (tipos de sistemas difusos y base de reglas) como a nivel de jerarquía (serie, paralelo e híbrido), su aprendizaje estructural con distintos métodos (algoritmos genéticos, evolutivos, etc) y los tipos de variables que establecen una relación entre los módulos. También se ha realizado un estudio del estado del arte en el área, pudiendo apreciar que es un tema de interés a explotar desde el punto de vista del aprendizaje de estructuras jerárquicas difusas ya que, como se comentó en la sección 2.4, existen modelos jerárquicos para la resolución de problemas de regresión pero no se centran en el aprendizaje de la estructura.

Con todo lo que se ha puesto de manifiesto en este capítulo, a continuación enumeramos las principales propiedades de este modelizado que nos hacen verlo como prometedor:

1. *Precisión:* El uso de módulos con un número bajo de variables de entrada hace que cada uno de los sistemas difusos que componen la jerarquía tenga un conocimiento más específico de la zona del espacio de búsqueda que está tratando.

2. *Simplicidad, interpretabilidad y transparencia:* Un número bajo de variables de entrada en un módulo hace que disminuya la complejidad a nivel de regla.

3. *Complejidad:* El encadenamiento de módulos en una estructura jerárquica hace que el número de reglas global del sistema disminuya, transformándose la función exponencial que genera la base de reglas completa del problema a una función lineal.

2.6. Nuevas Líneas de Trabajo Propuestas en esta Memoria

Después de haber analizado las características y el modelizado de los Sistemas Difusos Jerárquicos, vamos a finalizar este capítulo estableciendo las principales propiedades que poseen los algoritmos que se han diseñado e implementado en este trabajo. Estas propiedades serán desarrolladas en profundidad en los siguientes capítulos.

Este trabajo propone el uso de variables del problema para la comunicación entre módulos. Una versión previa a la propuesta de jerarquía en serie fue publicada por Benítez y Casillas en 2009. Cada módulo se compone de una base de reglas de

tipo Mamdani que infieren las variables que enlazan los módulos. El SD que se genera mediante la jerarquía es más interpretable porque usa reglas de tipo Mamdani. Además, usamos metodologías robustas mediante un proceso de aprendizaje basado en computación evolutiva, que recibe el nombre de Sistemas Difusos Genéticos. Los Sistemas Difusos Genéticos se caracterizan por la combinación de una búsqueda basada en poblaciones con una representación lingüística interpretable (Casillas y Carse 2009; Nojima y cols. 2011).

El trabajo propone el diseño e implementación de tres Algoritmos Genéticos Multiobjetivo (AG Multiobjetivo), lo cual supone una novedad en el área y garantiza sistemas con una gran legibilidad. Cada uno implementa una estructura jerárquica distinta (serie, paralela e híbrida) y cumplen las siguientes características:

1. Permiten manejar problemas de complejidad considerable gracias a su estructura jerárquica y la inclusión de la selección de variables.

2. Garantizan una buena interpretabilidad debido a que el algoritmo usa variables del problema para enlazar módulos. La estructura jerárquica merma el número de reglas y cada regla es más simple, ya que el número de variables por regla es bajo. Además, hacen uso de reglas tipo Mamdani, lo cual implica que la regla sea más comprensible.

3. Cada algoritmo obtiene distintos equilibrios entre interpretabilidad y precisión gracias al uso en el aprendizaje de un algoritmo multiobjetivo, por lo que genera distintas soluciones con distintos equilibrios.

Destacar que la mayoría de las propuestas en el estado del arte están centradas en la mejora de la precisión. Sin embargo, el objetivo de este trabajo es mejorar la interpretabilidad mediante un conjunto reglas precisas y compactas, mientras que la precisión se mantiene o se mejora.

3. Modelizado de Sistemas Mediante Sistemas Difusos Jerárquicos en Serie

Todo hombre, por naturaleza, desea saber.

ARISTÓTELES (384 A. C. – 322 A. C.), FILÓSOFO, LÓGICO Y CIENTÍFICO DE LA ANTIGUA GRECIA

En el capítulo anterior se han descrito tanto los distintos tipos de jerarquías que podemos encontrar (serie, paralela e híbrida) como su modelizado (tipos de módulos y reglas). Además, se da una idea general del estado del arte correspondiente al área. Todos estos trabajos nos motivan a presentar el modelizado de uno de los Sistemas Difusos Jerárquicos: la modalidad en serie. Este capítulo describe el diseño de un algoritmo genético multiobjetivo que aprende una población de estructuras jerárquicas en serie a lo largo de un ciclo de vida medido en evaluaciones, en donde la selección natural de individuos se basa en el criterio de minimizar tanto el error cuadrático medio como el número total de reglas de cada jerarquía. Para este propósito, se han diseñado tres operadores genéticos específicos: un operador de cruce y dos operadores de mutación. El esquema básico del algoritmo se especifica en el algoritmo 1. El algoritmo obtiene un conjunto Pareto de soluciones con distintos equilibrios entre precisión e interpretabilidad.

Para introducir este modelizado, comenzaremos el capítulo con la definición del esquema de codificación de las estructuras jerárquicas en serie del algoritmo de aprendizaje. A continuación, explicaremos los operadores genéticos de cruce y mutación, en donde se hará una descripción detallada de la funcionalidad de cada uno de ellos. Seguidamente, se indicarán los problemas seleccionados y la configuración experimental. Finalmente, se mostrarán los resultados obtenidos en la experimentación junto al análisis y a un estudio minucioso de su comportamiento.

Algoritmo 1 Algoritmo SDJSG

Entrada: Tamaño de la población, Probabilidad de cruce y mutación. Conjunto de datos: $D = \{(x,y)|x \in \mathbb{R}^n, y \in \mathbb{R}^m\}$. Definición de las funciones de pertenencia.

Salida: Conjunto de soluciones no dominadas, cada una con un equilibrio distinto entre precisión/número máximo de reglas.

```
Inicialización(P)
Evaluación(P)
Mientras (no se cumpla la condición de parada) hacer
    P1 ← Selección_Multiobjetivo(P)
    P2 ← Cruce(P1)
    P3 ← Mutación_Intercambio(P2)
    P4 ← Mutación_Inserción(P3)
    Evaluación(P4)
    P ← Reemplazamiento_Multiobjetivo(P4)
Fin Mientras
```

3.1. Descripción del Algoritmo

3.1.1. Esquema de Codificación

El esquema de codificación es uno de los aspectos más importantes en el diseño del algoritmo ya que es la base de donde partimos para poder definir el resto de componentes. Por ello, la codificación ha de ser sencilla y lo más representativa posible del problema que estamos tratando. La codificación de la jerarquía en serie de nuestro algoritmo considera los dos tipos de variables de entrada que comentamos en el capítulo anterior: variables endógenas y exógenas.

La idea de la que partimos es que el algoritmo codifique cada SDJS como un individuo de la población, de forma que cada individuo sea una concatenación de genes (en otras palabras, una representación vectorial de genes) donde cada gen contenga dos campos: la variable y el tipo de variable. El campo de tipo de variable toma valores binarios, de forma que si toma el valor '0' indica que la variable es exógena y si toma el valor '1' representa una variable endógena.

Una variable que en su tipo contenga el valor '1', además de ser una variable endógena, indica que es la variable de salida de un módulo y variable de entrada del siguiente módulo. Queremos destacar un aspecto importante y que hace que nuestra propuesta sea distinta a las que normalmente se encuentra en la literatura: todas las variables que se usan en la codificación de cada individuo son variables del problema y es el propio algoritmo el que decide si una variable del problema será endógena o exógena, pero no crea nuevas variables artificiales. Por otro lado, en una codificación

de este tipo de jerarquía, el mayor nivel jerárquico ocurre cuando se usan todas las variables del problema en una estructura en donde todos sus módulos tienen una variable de entrada y una variable de salida. En este caso, todas las variables del problema serán endógenas excepto la primera y el número de capas en la jerarquía es igual al número de variables de entrada. La figura 3.1 muestra el máximo nivel de jerarquía para un sistema con cuatro variables de entrada.

En la figura 3.2 se puede apreciar un ejemplo de codificación. Este ejemplo pone de manifiesto un sistema jerárquico en serie con 10 variables de entrada, en donde se observa claramente el tipo de variable según su campo tipo asociado: cuatro de ellas son variables exógenas (las variables X_{10}, X_3, X_5 y X_6, ya que el campo tipo de variable está inicializado a '0'), dos de ellas son endógenas (las variables X_8 y X_7, porque el campo tipo de variable está inicializado a '1') y cuatro variables (X_1, X_2, X_4 y X_9) que el algoritmo no considera en la codificación de este sistema jerárquico.

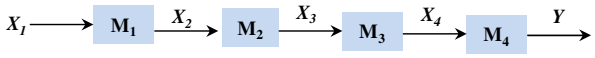


Figura 3.1.: Máximo nivel jerárquico para un sistema con cuatro variables de entrada en un SDJS

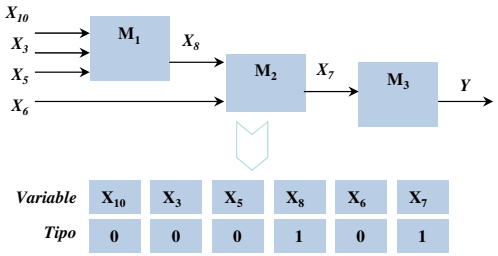


Figura 3.2.: Ejemplo de esquema de codificación de un SDJS

3.1.2. Inicialización

Como en cualquier algoritmo basado en poblaciones, se establece una población inicial de partida. El algoritmo que implementa nuestra propuesta, genera la pobla-

ción inicial de forma aleatoria porque queremos que haya diversidad estructural en las soluciones y así tener una representación de la mayor parte de la población posible o al menos evitar una convergencia prematura.

En nuestro algoritmo, la variable y el tipo (ya sea endógena o exógena) se escogen aleatoriamente para cada gen teniendo como única restricción que la variable no se puede repetir y que el tipo de la variable del primer gen no puede inicializarse al valor '1', ya que la variable de este gen obligatoriamente ha de ser exógena. Un aspecto importante es que el algoritmo limita el número de módulos por individuo debido a dos razones:

1. Si el número de módulos es alto, la propagación del error se incrementará considerablemente como venimos comentando en los capítulos anteriores.

2. Queremos reducir el espacio de búsqueda, ya que de esta forma el algoritmo puede converger a la solución más rápidamente y desestimar del espacio de búsqueda soluciones que sabemos que no son prometedoras.

3.1.3. Operador de Cruce

El operador de cruce es una funcionalidad importante que ha de implementar cualquier AG para la recombinación de individuos, ya que explota la información disponible en poblaciones anteriores para influir en futuras búsquedas de soluciones, dando lugar a la diversidad en la población y evitando la convergencia prematura. El operador de cruce se aplica según una probabilidad entre pares de cromosomas padres.

Cuando el operador de cruce se aplica a dos padres, P_1 y P_2, se pueden distinguir diferentes casos:

1. Ambos padres, P_1 y P_2, tienen varios módulos.

2. El padre P_1 tiene varios módulos y el padre P_2 tiene un solo módulo (o viceversa).

3. Ambos padres, P_1 y P_2, tienen un módulo.

El operador de cruce implementa un enfoque centrado en el padre, donde los descendientes principalmente heredan la información de uno de los padres y el resto se hereda del padre secundario para añadir diversidad. La figura 3.3 muestra todos

los casos posibles del operador de cruce mediante árboles de decisión. El operador de cruce controla el número máximo de módulos con el objetivo de reducir el espacio de búsqueda, creando una nueva solución si el cruce entre dos padres genera un número de módulos menor o igual al número máximo de módulos. En otro caso, el individuo generado se descarta y se genera uno nuevo.

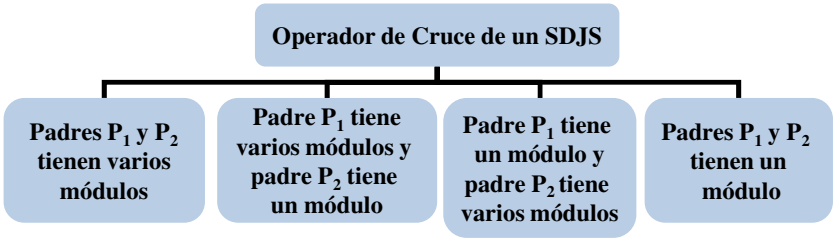
El operador de cruce se aplica a los individuos en función de los tipos de variables que los dos padres tengan en común. Cada caso tiene una lista de prioridades ordenadas (de mayor a menor prioridad), por lo tanto el cruce se aplica a la primera coincidencia del operador con su puesto en la lista. La figura 3.3 ilustra la prioridad de cada tipo de cruce. En los apartados siguientes se describe la funcionalidad de cada tipo de cruce.

3.1.3.1. Ambos padres, P_1 y P_2, tienen varios módulos

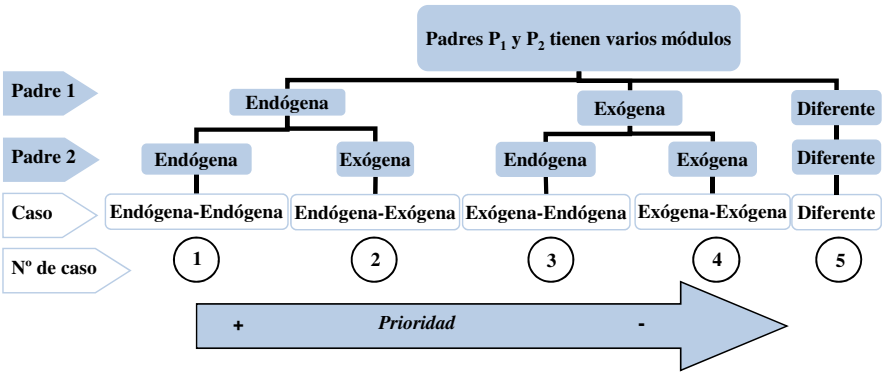
- *Prioridad 1. Caso Endógena-Endógena:* Si P_1 tiene variables endógenas comunes con P_2 (figura 3.3(b), número de caso 1), se selecciona aleatoriamente una variable endógena común. Esta variable es el punto de cruce. El descendiente D_1 se genera centrado en P_1. Así, el descendiente D_1 hereda de P_2 las variables y módulos desde el comienzo del cromosoma hasta el punto de cruce, pero excluyéndolo. La parte restante se toma de P_1 y las variables repetidas se eliminan de la parte heredada de P_2. El descendiente D_2 se genera de la misma forma pero centrado en P_2. La figura 3.4(a) ilustra un ejemplo de este caso.

- *Prioridad 2. Caso Endógena-Exógena.* Si P_1 tiene variables endógenas que son exógenas en P_2 (figura 3.3(b), número de caso 2), se selecciona una variable común de forma aleatoria como punto de cruce. El descendiente D_1 se genera centrado en P_1. La variable común endógena heredada de P_1 se convierte en exógena y el resto de los módulos previos a esa variable se eliminan. Esta clase de alteración produce un cambio con baja agresividad porque los descendientes se parecen a sus padres.

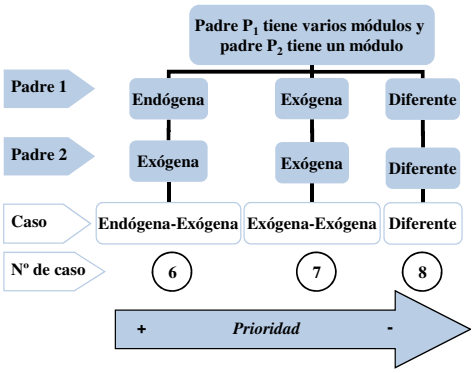
- *Prioridad 3. Caso Exógena-Endógena.* Si P_1 tiene variables exógenas que son endógenas en P_2 (figura 3.3(b), número de caso 3), se selecciona una de esas variables aleatoriamente como punto de cruce. El descendiente D_1 se crea centrado en P_1, pero excluyendo el punto de cruce. La primera parte de D_1 se hereda de la primera parte de P_2 (desde el primer gen hasta el punto de cruce). Las variables repetidas se eliminan en D_1 de la parte heredada de P_2.



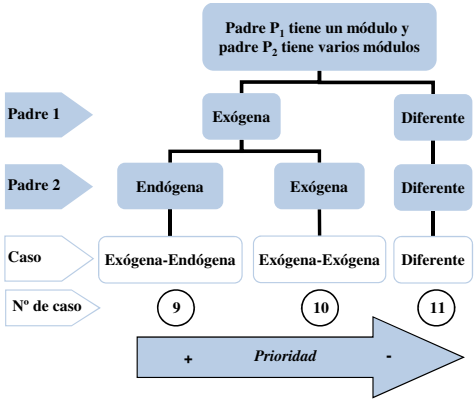
(a) Árbol de decisión general del operador de cruce



(b) Padres P_1 y P_2 tienen varios módulos



(c) Padre P_1 tiene varios módulos y padre P_2 tiene un módulo



(d) Padre P_1 tiene un módulo y padre P_2 tiene varios módulos

Figura 3.3.: Árboles de decisión del operador de cruce de un SDJS

- *Prioridad 4. Caso Exógena-Exógena.* Si P_1 tiene un conjunto de variables exógenas en común con variables exógenas de P_2 (figura 3.3(b), número de caso 4), el descendiente D_1 se genera como una copia de P_1 pero con un cambio: se elige aleatoriamente una variable exógena común entre P_1 y P_2. Esta variable en P_2, junto con otras variables, son la entrada a un módulo que genera una variable endógena. Esta variable endógena en P_2 se selecciona para reemplazar la variable exógena aleatoria seleccionada en D_1, que es común a P_1 y P_2. A continuación, las variables repetidas se eliminan de D_1 en la parte heredada de P_1. Si este caso también ocurre para P_2, el descendiente D_2 se generará mediante el mismo procedimiento, pero centrado en P_2. La figura 3.4(b) muestra un ejemplo de este caso. El descendiente tiene cuatro módulos. Destacar que las variables exógenas comunes de P_1 se buscan en P_2 excluyendo las variables exógenas del módulo con la salida Y en P_2, porque si se incluye este módulo, cuando una variable exógena se selecciona, no hay ninguna variable endógena como salida ya que coincide con la salida del SDJS.

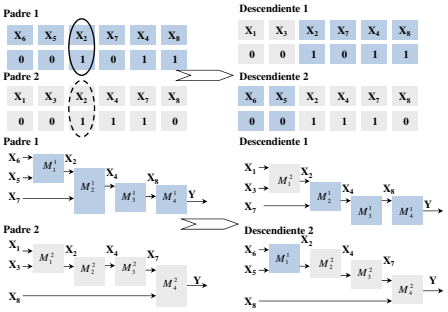
- *Prioridad 5. Caso diferentes variables.* Si las variables de P_1 y P_2 son diferentes (figura 3.3(b), número de caso 5), se elige una variable endógena de P_1 y P_2 aleatoriamente como punto de cruce. Los descendientes se generan de la misma forma que en la prioridad 1.

3.1.3.2. El padre P_1 tiene varios módulos y el padre P_2 tiene un módulo

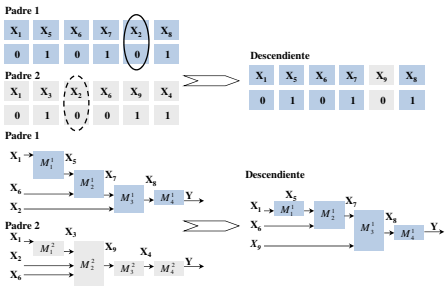
- *Prioridad 1. Caso Endógena-Exógena.* Si P_1 tiene variables endógenas comunes con variables exógenas de P_2 (figura 3.3(c), número de caso 6), el descendiente D_1 se genera como en el caso Endógena-Exógena cuando P_1 y P_2 tienen varios módulos. La figura 3.4(c) representa este caso.

- *Prioridad 2. Caso Exógena-Exógena.* Si P_1 y P_2 tienen en común variables exógenas (figura 3.3(c), número de caso 7), el descendiente D_1 centrado en P_1 se crea de la siguiente forma. Primero D_1 se genera como una copia de P_1. Después, una variable exógena común entre P_1 y P_2 se elige aleatoriamente y se mueve al módulo con salida Y. Esto se muestra en la figura 3.4(d).

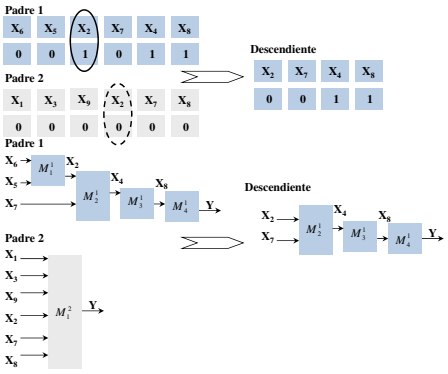
- *Prioridad 3. Caso diferentes variables.* En este caso (figura 3.3(c), número de caso 8), D_1 es una copia de P_1 y, a continuación, una variable exógena de P_2 se



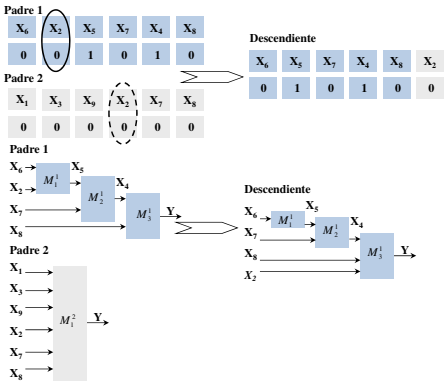
(a) Padres P_1 y P_2 tienen varios módulos: Caso Endógena-Endógena



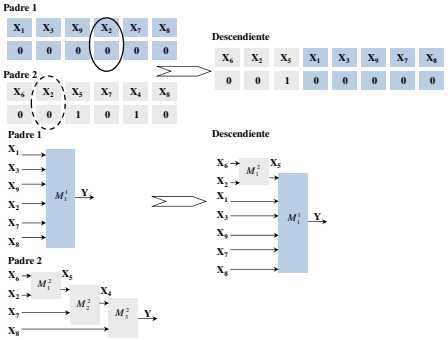
(b) Padres P_1 y P_2 tienen varios módulos: Caso Exógena-Exógena



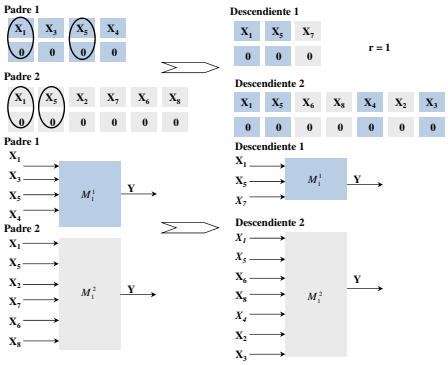
(c) Padre P_1 tiene varios módulos y padre P_2 tiene un módulo: Caso Endógena-Exógena



(d) Padre P_1 tiene varios módulos y padre P_2 tiene un módulo: Caso Exógena-Exógena



(e) Padre P_1 tiene un módulo y padre P_2 tiene varios módulos: Caso Exógena-Exógena



(f) Padres P_1 y P_2 tienen un módulo

Figura 3.4.: Operador de cruce de un SDJS

escoge aleatoriamente y se inserta en D_1 en el módulo que tiene la variable Y como salida.

3.1.3.3. El padre P_1 tiene un módulo y el padre P_2 tiene varios módulos

- *Prioridad 1. Caso Exógena-Endógena.* Si P_1 tiene variables exógenas que son endógenas en P_2 (figura , número de caso 9), el descendiente D_1 se genera como en el caso Exógena-Endógena cuando P_1 y P_2 tienen varios módulos.
- *Prioridad 2. Caso Exógena-Exógena.* Si P_1 tiene un conjunto de variables exógenas en común con variables exógenas en P_2 (figura , número de caso 10), el descendiente D_1 se genera de la siguiente forma: 1) se hace una copia exacta de P_1 en D_1; 2) se selecciona aleatoriamente una variable exógena común a ambos padres; 3) la variable común en P_2 junto con otras variables son la entrada a un módulo que genera una variable endógena. Esa variable endógena se establece como punto de cruce y se selecciona la subestructura desde el principio de P_2 hasta ese punto de cruce. La subestructura reemplaza la variable exógena seleccionada de P_1 en D_1; 4) las variables repetidas se eliminan de D_1 de la parte heredada de P_1. La figura 3.4(e) ilustra este caso.
- *Prioridad 3. Caso diferentes variables.* Si las variables de P_1 y P_2 son distintas (figura , número de caso 11), se elige una variable exógena de P_1 y una variable endógena de P_2 aleatoriamente como punto de cruce. Los descendientes se generan de la misma forma que en la prioridad 1.

3.1.3.4. Ambos padres, P_1 y P_2, tienen un módulo

En este caso, los descendientes D_1 y D_2 se generan de la siguiente forma (la figura 3.4(f) muestra este caso):

1. Las variables exógenas comunes de P_1 y P_2 se insertan en ambos descendientes.
2. El resto de variables (las variables no comunes entre padres) se insertan en un conjunto V de tamaño $n = |V|$,
3. Se genera un número aleatorio r: si no existen variables comunes entre P_1 y P_2, entonces $r \in \{1,\ldots,n-1\}$; en otro caso, $r \in \{0,\ldots,n\}$.
4. Se escogen r variables aleatoriamente y se insertan en D_1.

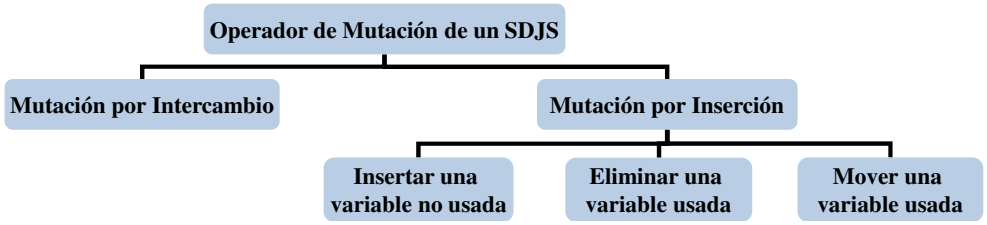


Figura 3.5.: Árbol de decisión del operador de mutación de un SDJS

5. El resto de variables se insertan en D_2.

3.1.4. Operador de Mutación

El operador de mutación realiza cambios aleatorios locales en el cromosoma, lo que permite explorar el espacio de búsqueda. Al igual que el operador de cruce, este operador controla el número máximo de módulos. Si se genera un individuo con un número de módulos mayor al establecido, este individuo se descarta y se genera uno nuevo. Se han diseñado dos tipos de mutación (la figura 3.5 muestra el árbol de decisión de este operador):

3.1.4.1. Mutación por Intercambio

Este operador de mutación realiza un intercambio de variables de un módulo seleccionado de forma aleatoria previamente. Para ello, elige una variable exógena aleatoriamente de un módulo y la intercambia por la variable endógena del módulo escogido. La figura 3.6 muestra un ejemplo de este operador, donde el módulo seleccionado por el operador es M_2.

3.1.4.2. Mutación por Inserción

Este operador realiza una distinción entre variables usadas y no usadas en el individuo. Atendiendo a esto y por medio de una decisión aleatoria (el algoritmo 2 muestra un esquema), el operador de mutación elige entre tres posibilidades: insertar una variable no usada en un módulo, eliminar una variable usada o mover una variable usada a otro módulo.

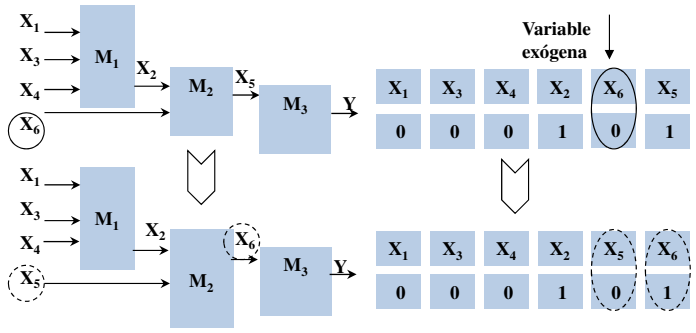
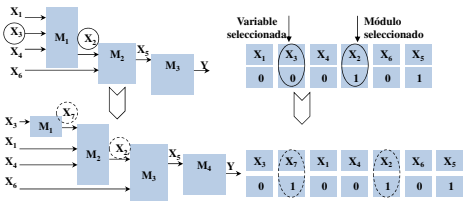


Figura 3.6.: Ejemplo de mutación por intercambio en un SDJS

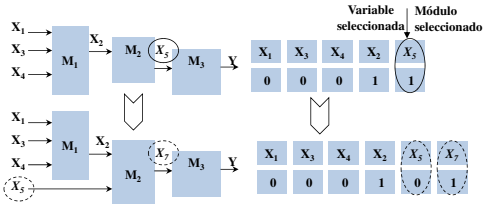
Algoritmo 2 Mutación por Inserción de un SDJS

```
r = U[0,1]; {Probabilidad uniforme}
Si (r < 0,5 y hay variables no usadas) entonces
    v = Elegir_aleatoriamente_una_variable_no_usada
    Insertar_variable_no_usada(v); {Algoritmo 3}
Sino
    t = U[0,1]
    Si (t < 0,5 y el SDJS no tiene sólo un módulo con una entrada y una salida) entonces
        Eliminar_variable_usada {Algoritmo 4}
    Sino
        Mover_variable_usada
    Fin Si
Fin Si
```

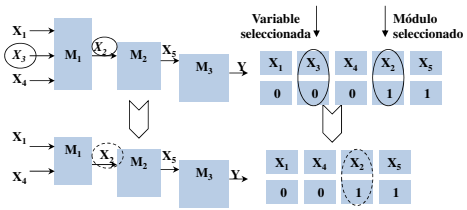
- La inserción de una variable no usada se implementa de la siguiente forma: Dada una variable no usada v, se elige aleatoriamente un módulo m. Seguidamente, el operador decide si v se inserta en m con una probabilidad de 0,5. Si es así, v se inserta como variable exógena. En otro caso, si m tiene al menos una variable exógena como entrada, una de ellas, e, se selecciona aleatoriamente. El operador de mutación crea un módulo previo a m con una entrada y una salida, donde la entrada es e y la salida es v. La figura 3.7(a) muestra un ejemplo. En este caso, v será endógena y la entrada del módulo m. Si m no tiene variables exógenas (figura 3.7(b)), la salida de m se convierte en una nueva variable exógena de m y v será la nueva salida de m. La mutación se aplica si la solución tiene un número de módulos menor que el máximo número de módulos.



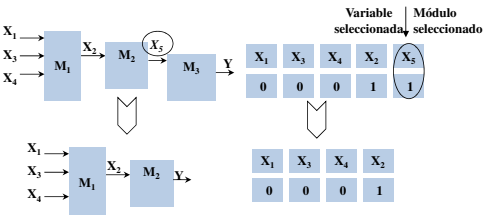
(a) Inserción de una variable no usada. Caso: Si m tiene al menos una variable exógena como entrada



(b) Inserción de una variable no usada. Caso: Si m no tiene variables exógenas



(c) Eliminar una variable usada. Caso: Si m tiene variables exógenas



(d) Eliminar una variable usada. Caso: Si m no tiene variables exógenas

Figura 3.7.: Operador de mutación por inserción de un SDJS

- Para eliminar una variable usada, primero se elige aleatoriamente un módulo m del SDJS. Si m tiene variables exógenas, entonces una de ellas se escoge aleatoriamente y se elimina. La figura 3.7(c) representa este caso. En otro caso, se elimina el módulo m, convirtiendo la salida del módulo anterior a m en la entrada del módulo posterior a m. La figura 3.7(d) ilustra un ejemplo.

- Para mover una variable usada de un módulo a otro, se eligen aleatoriamente del cromosoma dos módulos aleatorios: un módulo origen m_1 de donde se elimina la variable y un módulo destino m_2, en donde se inserta la variable. Se elimina una variable exógena de m_1 y se inserta como exógena en m_2. Si no hay variables exógenas en m_1, y consecuentemente m_1 tiene una entrada y una salida, se elige la variable endógena. En este caso, m_1 desaparece y la variable elegida se inserta como una variable exógena en m_2. Si $m_1 = m_2$, se elige una variable exógena v aleatoriamente y se crea un nuevo módulo después de m_1/m_2. La entrada de este nuevo módulo (y por lo tanto la salida del módulo

de m_1/m_2) es v, mientras que la salida del nuevo módulo será la antigua salida de m_1/m_2. En este caso, la mutación se aplica si la solución tiene un número de módulos menor que el máximo número de módulos.

Los algoritmos 3, 4 y 5 muestran el comportamiento del operador de mutación.

Algoritmo 3 Inserción de una variable no usada en un SDJS

Entrada: Una variable no usada: v
1: m = Elegir_un_módulo_aleatoriamente
2: r = U[0,1] {Probabilidad uniforme}
3: **Si** ($r < 0,5$) **entonces**
4: Insertar_variable_exógena(v, m)
5: **Sino**
6: t = U[0,1]
7: **Si** ($t < 0,5$ *y m no es un módulo de una entrada y una salida*) **entonces**
8: e = Elegir_una_variable_exógena_aleatoriamente(m)
9: Crear_módulo_previo_de_una_entrada_una_salida(e, v, m)
10: **Sino**
11: Convertir_variable_salida_en_exógena(m)
12: Convertir_variable_a_salida(v, m)
13: **Fin Si**
14: **Fin Si**

Algoritmo 4 Borrar una variable usada de un SDJS

1: m = Elegir_un_módulo_aleatoriamente
2: **Si** (*m no es un módulo de una entrada y una salida*) **entonces**
3: e = Elegir_una_variable_exógena_aleatoriamente(m)
4: Borrar_variable_exógena(e, m)
5: **Sino**
6: Borrar_módulo(m)
7: Enlazar_módulos($m-1$, $m+1$)
8: **Fin Si**

3.1.5. Mecanismo de Inferencia

La función principal del mecanismo de inferencia es la de aplicar los conjuntos difusos a cada una de las reglas de la base de reglas y así dar una salida dentro del Universo de Discurso de las variables lingüísticas. En la implementación del algoritmo se considera el esquema de inferencia Max–Min (es decir, la T-conorma del máximo como agregación y la T-norma del mínimo como operador relacional), la T-norma del mínimo como conjunción y el centro de gravedad como defuzzificación.

Algoritmo 5 Mover una variable usada en un SDJS

```
1: m1 = Elegir_un_módulo_aleatoriamente
2: m2 = Elegir_un_módulo_aleatoriamente
3: Si (m1 ≠ m2) entonces
4:    Si (m1 no es un módulo de una entrada y una salida) entonces
5:       e = Elegir_variable_exógena_aleatoriamente(m1)
6:       Borrar_variable(e, m1)
7:       Insertar_variable(e, m2)
8:    Sino
9:       v = Elegir_variable_endógena(m1)
10:      Borrar_módulo(m1);
11:      Insertar_variable_exógena(v, m2)
12:   Fin Si
13: Sino
14:   v = Elegir_variable_exógena(m1)
15:   w = Elegir_variable_endógena(m1)
16:   m_nuevo=Crear_nuevo_módulo_posterior(m1)
17:   Insertar_variable_entrada_en_módulo(v, m_nuevo)
18:   Insertar_variable_salida_en_módulo(w, m_nuevo)
19:   Enlazar_módulos(m1, m_nuevo)
20: Fin Si
```

La inferencia en las estructuras jerárquicas se produce desde los módulos situados a la izquierda a los módulos situados en la derecha. Como se ha comentado anteriormente, cuando un módulo infiere la salida, produce un error que se acarrea a través de la estructura jerárquica independientemente del tipo de jerarquía. Por esta razón, no es adecuado tener un número elevado de módulos en un sistema difuso jerárquico ya que el error que se acarrea puede incrementarse considerablemente. Hay que ser cauto a la hora de diseñar el sistema con el objetivo de conseguir que el error tenga poca propagación porque cuantos más módulos se consideren, obtendremos mayor incertidumbre (Maeda 1996).

3.1.6. Aprendizaje de la Base de Reglas

Mediante el aprendizaje de la base de reglas se obtiene el conocimiento para determinar la salida del sistema. En el algoritmo, el aprendizaje del conjunto de reglas difuso se realiza de la siguiente forma: cada módulo aprende su conjunto de reglas difusas de tipo Mamdani mediante el método de L. Wang y Mendel (WM) de 1992 (ver Apéndice A.1). Todas las variables de un sistema difuso jerárquico son propias del problema, es decir, el sistema no crea variables artificiales. Las variables exógenas se

extraen del conjunto de datos y las variables endógenas se infieren por su respectivo módulo de acuerdo a las variables de entrada de cada módulo. La figura 3.8 muestra un ejemplo gráfico del proceso de inferencia de un SDJ en serie: el primer módulo recibe tres variables exógenas del conjunto de datos e infiere una variable natural endógena; el valor de la variable endógena inferida, que no se extrae del conjunto de datos, es la entrada del siguiente módulo junto con otras variables extraídas del conjunto de datos, y así sucesivamente. El sistema obtenido se evalúa mediante ejemplos de datos de entrenamiento y eligiendo los valores correspondientes de variables exógenas en ese módulo. El objetivo es construir un sistema cerrado para asegurar una buena interpretabilidad y reducir el error.

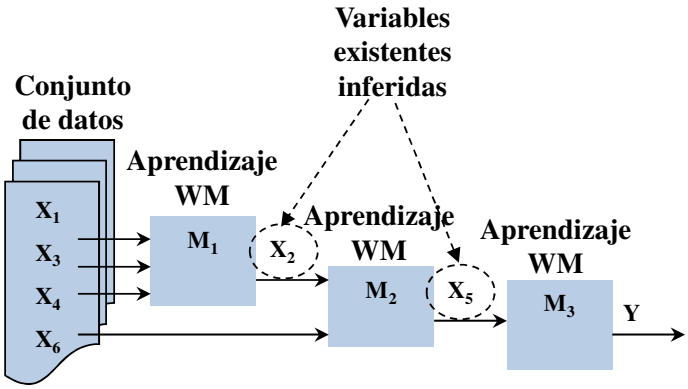


Figura 3.8.: Ejemplo del proceso de inferencia en un SDJ en serie

3.1.7. Enfoque Multiobjetivo

El algoritmo hace uso del enfoque generacional con la estrategia de reemplazamiento multiobjetivo del algoritmo NSGA-II (Deb y cols. 2002). Considera la distancia de crowding en el espacio de la función objetivo y se aplica una selección por torneo binario que se basa en una clasificación de no dominancia (o en que la distancia de crowding cuando ambas soluciones pertenecen a la misma frontera). La distancia de crowding se normaliza por cada objetivo de acuerdo a los valores extremos de las soluciones contenidas en la frontera que se analiza. El aprendizaje de la estructura jerárquica en serie se hace posible gracias al uso de este enfoque que

minimizará dos funciones objetivo: el error cuadrático medio (precisión) y el número de reglas (interpretabilidad). Las funciones objetivo se describen en el apartado siguiente.

La decisión del uso del algoritmo NSGA-II se debe a su rendimiento eficiente y efectividad. No es objetivo de esta memoria el análisis detallado sobre el mejor enfoque multiobjetivo.

3.1.8. Funciones Objetivo

Consideramos dos funciones objetivo a minimizar, que nos permitirán evaluar la calidad de la solución obtenida por el algoritmo:

- *Error Cuadrático Medio (ECM):* Si el ECM es bajo, nos indica que la precisión mejora. Se calcula como:

$$F_1(S) = \frac{1}{N}\sum_{i=1}^{N}(S(x^i) - y^i)^2 \tag{3.1}$$

  Siendo S el SD a evaluar, N el tamaño del conjunto de datos y (x^i, y^i) la i-ésima pareja entrada-salida del conjunto de datos. El objetivo, en nuestro caso, es minimizar esta función para mejorar la precisión.

- *Máximo número de reglas:* El sistema será más interpretable si el número máximo de reglas es bajo. Se calcula de la siguiente forma:

$$F_2(S) = \sum_{i=1}^{M}(k^n) \tag{3.2}$$

  Siendo M el número de módulos de SD, y k y n el número de etiquetas y variables por módulo, respectivamente.

  Este objetivo tiene la principal ventaja (en comparación con el número final de reglas) de que depende exclusivamente de la estructura "estática" de la jerarquía, que es el número de módulos, el número de variables por módulo y el número de términos lingüísticos usados en cada variable. Así, el objetivo no está influenciado por el aprendizaje del algoritmo usado para generar el conjunto de reglas difusas en cada módulo. Esto es especialmente relevante cuando se refinan los parámetros de las funciones de pertenencia, en donde se pueden generar soluciones con distinto número de reglas y la misma estructura jerárquica.

Además, aunque el número de módulos o el número de variables por módulo se pueden considerar para evaluar cómo de compacta es una estructura jerárquica, el número máximo de reglas es un criterio más intuitivo para validar la interpretabilidad porque combina ambos criterios en un solo objetivo. Si se considera el número máximo de reglas, el número de variables y módulos se regularán automáticamente: a medida que el número máximo de reglas decrece, el número de módulos se incrementa y el número de variables por módulo decrece.

3.2. Experimentos

3.2.1. Problemas

Hemos considerado 14 problemas de regresión en donde poder aplicar el algoritmo con un moderado y alto número de variables reales:

- *Mantenimiento Eléctrico (Ele2):* Este problema hace referencia a la estimación del coste de mantenimiento de una red eléctrica de una línea de voltaje medio basado en la suma de las longitudes de todas las calles de la ciudad, el área total de la ciudad, el área ocupada por edificios y el suministro de energía en la ciudad (Cordón y cols. (1999)). La información se obtiene muestreando un modelo óptimo de red eléctrica para una ciudad.

- *Laser:* Usa un conjunto de datos de laser de predicción de series temporales y competición de análisis del Instituto de Santa Fe (ISF) (Weigend y Gershenfeld (1993)). El conjunto de datos original de ISF consistían en 1.000 observaciones de las fluctuaciones de un laser infrarrojo. Esta serie temporal de datos ha sido adaptada para regresión considerando los cuatro últimos valores como entrada y el siguiente valor como salida.

- *Daily Electric Energy (DEE):* Consiste en la predicción del precio medio diario de la energía eléctrica TkWhe en España. El conjunto de datos está formado por una serie de datos reales de 2003 sobre el consumo diario en España de energía hidroeléctrica, eléctrica nuclear, carbón, combustible y gas natural.

- *Concrete:* El hormigón (concrete, en inglés), es el material más importante en la Ingeniería Civil. Se trata de predecir la fuerza de compresión del hormigón

a partir de la edad y otros siete componentes de la mezcla. Es un problema con valores reales y de ocho variables.

- *Census-House:* Este problema trata de la predicción de los precios medios de las casas a partir de los datos del censo estadounidense de 1990. Consta de dos problemas de ocho y 16 variables divididos en correlación entre las variables alta (H) y baja (L).

- *Weather-Ankara (WeAnk):* Contiene la información sobre el tiempo realizado en Ankara (Turquía) desde 01/01/1994 al 28/05/1998. El problema consta de nueve variables continuas (temperatura, presión, precipitación, velocidad del viento, etc.) donde el objetivo es predecir la temperatura media.

- *Mortgage:* Este conjunto de datos contiene la información de los datos económicos de E.E.U.U. desde 01/04/1980 hasta 02/04/2000 de forma semanal. De las quince características que posee, el objetivo es predecir la tasa hipotecaria convencional a 30 años.

- *Treasury:* Este conjunto de datos recoge la información de los datos económicos de E.E.U.U. desde 01/04/1980 hasta 02/04/2000 de forma semanal. De las características dadas, un total de quince, el objetivo es predecir el cambio mensual económico.

- *Elevators:* El conjunto de datos se obtiene del control de un avión F16. El objetivo es predecir la acción que se ha de realizar sobre el elevador del avión. Consta de 18 variables de entrada.

- *Computer Activity (CompAct):* Este problema está formado por un conjunto de métricas sobre la actividad de un ordenador Sun Spacstation 20/712 con dos CPUs y 128 MBytes de memoria RAM en un departamento multi-usuario de una universidad. Los datos se recogieron de forma continua en dos ocasiones cada cinco segundos. La meta es predecir la porción de tiempo que las CPUs ejecutan en modo usuario de un conjunto de 21 variables continuas.

- *Ailerons:* Es un problema de control de direcciones de un avión F16, aunque las variables están en un dominio distinto a Elevators. El estado del avión consta de 40 variables de entrada. El objetivo es predecir la acción de control en función del alerón del avión.

La tabla 3.1 recoge las principales características de cada problema, donde *#VarEntrada* es sinónimo del número de variables de entrada, *#Ejem* identifica el número total de instancias, *#TermLing* indica el número de términos lingüísticos triangulares y *Disponible* referencia el sitio web en donde el conjunto de datos está disponible.

Además, las particiones triangulares son fuertes. Una partición difusa fuerte se define de la siguiente forma: una partición difusa X_i, es fuerte si se cumple que $\forall x \in X_i, \sum_j \mu_{A_j^i}(x) = 1$. La figura 3.9 ilustra un ejemplo de funciones de pertenencia triangulares fuertes. En este caso, los valores de pertenencia correspondientes a A y B cumplen este requisito. Su principal ventaja es que proporcionan interpretabilidad en el modelizado (Pedrycz 1994).

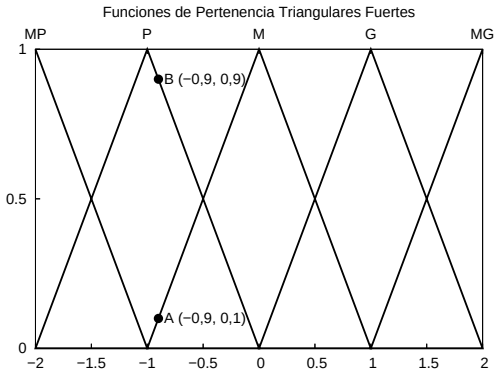


Figura 3.9.: Ejemplo de funciones de pertenencia triangulares fuertes

En la experimentación, establecemos tres etiquetas por variable para problemas complejos porque si el número de entradas y etiquetas es alto, el conjunto de reglas se incrementará considerablemente y el SD tendrá poca interpretabilidad. Por esta razón, se usa un número alto de etiquetas en problemas con un número de variables bajo.

[2]KEEL: Knowledge extraction based on evolutionary learning.http://www.keel.es

[3]UCI Machine Learning Repository. Colección de conjunto de datos de regresión. http://archive.ics.uci.edu/ml/datasets.html

[4]Delve. Conjunto de Datos de Regresión. http://www.cs.utoronto.ca/∼delve/data/datasets.html

[5]L. Torgo, Colección de conjunto de datos de regresión. http://www.liacc.up.pt/∼ltorgo/Regression/DataSets.html

Tabla 3.1.: Conjunto de datos considerados en el análisis experimental

Conjunto de datos	*#VarEntrada*	*#Ejem*	*#TermLing*	*Disponible*
Ele2	4	1.056	5	KEEL[2]
Laser	4	993	5	KEEL[2]
DEE	6	365	5	KEEL[2]
Concrete	8	1.030	5	UCI[3]
Census-House 8L (Census 8L)	8	22.784	5	Delve[4]
Census-House 8H (Census 8H)	8	22.784	5	Delve[4]
Weather-Ankara (WeAnk)	9	1.609	5	KEEL[2]
Mortgage	15	1.049	5	KEEL[2]
Treasury	15	1.049	5	KEEL[2]
Census-House 16L (Census 16L)	16	22.784	3	Delve[4]
Census-House 16H (Census 16H)	16	22.784	3	Delve[4]
Elevators	18	16.559	3	web site[5]
Computer Activity (CompAct)	21	8.192	3	web site[5]
Ailerons	40	13.750	3	web site[5]

El algoritmo tiene como objetivo la resolución de problemas de regresión. El conjunto de datos con mayor número de entradas que hemos encontrado en los sitios web considerados (tabla 3.1, columna *Disponible*) ha sido *Ailerons* con 40 variables reales. El conjunto de datos llamado TIC (85 variables) se encuentra en el sitio web de KEEL, pero no lo usamos porque la mayoría de las variables son nominales sin orden y la variable de salida es binaria, por lo tanto, este conjunto de datos no es problema de regresión y sí de clasificación.

3.2.2. Cálculo del Pareto Medio

Como nuestro algoritmo realiza una optimización multiobjetivo, en cada ejecución se obtienen varias soluciones. La media de las fronteras del Pareto (30 ejecuciones por problema) se calcula atendiendo a un proceso basado en los percentiles. Sea R el número de ejecuciones (30 en nuestra experimentación) y SP_i el conjunto Pareto obtenido en la i-ésima ejecución y ordenado por el primer objetivo. Por lo tanto, teniendo en cuenta el tamaño medio de los conjuntos de Pareto:

$$\bar{p} = redondeo\left(\frac{\sum_{i=1}^{R}|SP_i|}{R}\right), \tag{3.3}$$

la media del conjunto Pareto

$$\overline{SP} = \{\boldsymbol{p}_k \in \mathbb{R}^C \mid k \in \{1,\ldots,\bar{p}\}\} \tag{3.4}$$

se calcula como:

$$\boldsymbol{p}_k = (p_{k,1}, \ldots, p_{k,C}), \;\; p_{k,i} = \frac{\sum_{j=1}^{R} t_{j,i}^k}{R}, \tag{3.5}$$

siendo C el número de objetivos y

$$\boldsymbol{t}_j^k = (t_{j,1}^k, \ldots, t_{j,C}^k) = clasificar\left(1 + \frac{(|SP_i|-1)\cdot(k-1)}{\bar{p}-1}, SP_i\right) \tag{3.6}$$

siendo $clasificar(r,P)$ el r-ésimo elemento del conjunto ordenado P. Si r es un número entero, el elemento de esta posición se toma directamente. Si no, se calcula una interpolación lineal. Por ejemplo, si consideramos dos objetivos ($C = 2$) y $SP_1 = \{(1,10),(2,7),(5,6),(7,5),(9,3)\}$:

$$\begin{aligned}
&clasificar(1,SP_1) = (1,10), \quad clasificar(2,SP_1) = (2,7),\\
&clasificar(3,SP_1) = (5,6), \quad clasificar(4,SP_1) = (7,5),\\
&clasificar(5,SP_1) = (9,3), \quad clasificar(3{,}33,SP_1) =\\
&(5,6)\cdot 0{,}33 + (7,5)\cdot 0{,}77 = (6{,}33,5{,}33)
\end{aligned}$$

Por lo tanto, en un experimento con tres ejecuciones ($R = 3$) donde tenemos lo anterior PS_1, $PS_2 = \{(2,9),(3,6),(7,4)\}$ y $PS_3 = \{(2,10),(4,8),(6,5),(8,3)\}$, la media del conjunto Pareto es:

$$\overline{PS} = \{(1{,}67,9{,}67),(3{,}22,7{,}22),(5{,}56,5{,}22),(8,3{,}33)\}$$

La figura 3.10 muestra una representación gráfica de cada uno de los Paretos obtenidos en las tres ejecuciones y la media del conjunto Pareto.

3.2.3. Configuración Experimental

Un aspecto crucial a considerar después del diseño e implementación de un algoritmo es la evaluación de su rendimiento, esto es, cómo de bueno es el modelo de aprendizaje en el dominio de los datos. Hemos de ser cuidadosos con la forma en que el modelo reproduce los datos de aprendizaje. Debemos evitar que el modelo esté optimizado sólo para un conjunto de datos específico ya que puede no trabajar tan bien en otros datos del mismo dominio que no se habían visto antes, es decir, que se produzca un sobreajuste. Para prevenirlo usamos el mecanismo de *validación cruzada*.

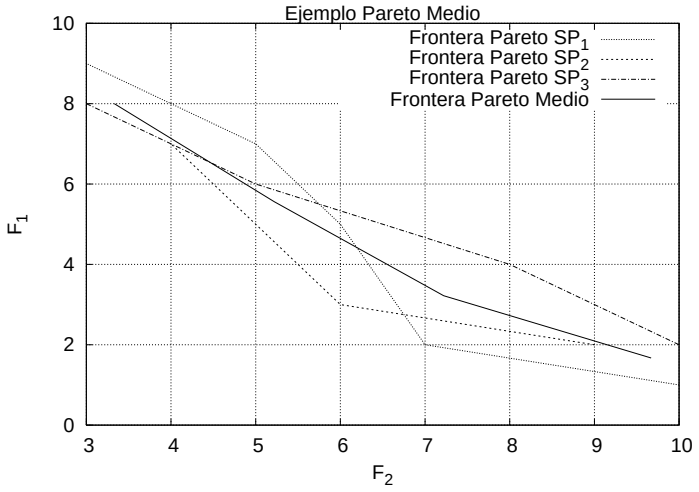


Figura 3.10.: Ejemplo del cálculo del Pareto medio

Este método divide los datos en *varias sub-muestras* de igual tamaño (aproximadamente). Para cada sub-muestra *i*, el algoritmo se entrena con cada una del resto de las sub-muestras, y la prueba se realiza con la sub-muestra *i*. El porcentaje de error estimado es el error promedio de las distintas sub-muestras. De esta forma, todos los datos se utilizan para validar. Nuestra experimentación utiliza este método con cinco particiones de datos.

Por otro lado, se aplican seis ejecuciones por partición del conjunto de datos mediante seis semillas distintas. Resumiendo, las cinco particiones de datos con validación cruzada, junto con las seis semillas hacen un total de 30 ejecuciones por problema.

Finalmente, el algoritmo se ha ejecutado con los siguientes parámetros de configuración: 50.000 evaluaciones, un tamaño de población de 60 individuos, una probabilidad de cruce de 0,7 y una probabilidad de mutación de 0,2 por cromosoma. No hemos realizado ningún análisis anterior para fijar estos valores, por lo que se pueden obtener mejores resultados regulándolos, aunque hemos notado informalmente que nuestro algoritmo no es especialmente sensible a cualquier parámetro. Además, hemos usado particiones difusas triangulares uniformemente distribuidas en la semántica de cada variable.

3.3. Estudio Experimental

3.3.1. Descripción de los algoritmos de comparación

A continuación, se describen otros algoritmos implementados para el estudio del algoritmo que proponemos:

- *Método WM* (L. Wang y Mendel 1992). Aunque WM es un algoritmo simple y los resultados que obtiene no son buenos, lo incluimos en este análisis porque se ha usado para aprender la base de reglas, y por lo tanto, podemos observar computacionalmente los beneficios proporcionados por la estructura jerárquica.

- *JS (algoritmo de Joo y Sudkamp)*. Este método implementa un algoritmo que transforma un SD en un SDJ de dos capas a partir de una base de reglas completa. En el apéndice A.3 se realiza una descripción más detallada.

- *SVAG (Selección de Variables mediante un Algoritmo Genético)*: Es un algoritmo de selección de variables basado en un proceso genético. El apéndice A.2 explica este algoritmo detalladamente. La razón por la que se ha implementado es para realizar una comparación justa con nuestro algoritmo, pudiendo aislar en el análisis el efecto de la selección de variables.

- *SDJSG s/SV (SDJSG sin Selección de Variables)*. Este algoritmo es el algoritmo SDJSG pero no realiza una selección de variables. El operador de cruce hace intercambio de módulos entre SDJS: se selecciona un punto de cruce aleatoriamente en cada padre; el primer descendiente hereda del primer padre desde el comienzo del cromosoma hasta el punto de cruce; el resto de variables se heredan del segundo padre pero en el orden contemplado en el segundo padre. El segundo descendiente hereda del segundo padre desde el comienzo del cromosoma hasta el punto de cruce; el resto de variables se heredan del primer padre pero en el orden relativo del primer padre. Con respecto al operador de mutación, realiza intercambio, inserción y mueve variables en un SDJS. El algoritmo crea SDJS con n número de módulos, siendo n el número de variables de entrada del problema.

- *SDJSG4M s/SV (SDJSG sin Selección de Variables creando 4 Módulos)*: Es el algoritmo SDJSG s/SV pero el número máximo de módulos está limitado a

cuatro módulos.

3.3.2. Resultados obtenidos

Las tablas 3.2, 3.3 y 3.4 recogen los resultados obtenidos por el algoritmo SDJSG, donde ECM_{entr} y ECM_{prue} son los valores del error de aproximación (ecuación 3.1) en entrenamiento y prueba respectivamente para un conjunto de datos; *#M*, *#R* y *#V* hacen referencia al número de módulos, número de reglas difusas y número de variables de entrada respectivamente; $\sigma_{\#M}$ es la desviación típica del número de módulos y $\bar{x}_{\#R}$ es el número de variables por regla, que se calcula de la siguiente forma:

$$\bar{x}_{\#R} = \frac{\sum_{i=1}^{\#M}(V_i \times R_i)}{\sum_{i=1}^{\#M} R_i} \tag{3.7}$$

siendo R_i el número de reglas del i-ésimo módulo y V_i el número de variables del i-ésimo módulo. Esta medida indica la complejidad media de las reglas.

Establecemos el número máximo de módulos a cuatro porque un SDJS con mayor número de módulos produce una mayor propagación del error y, al mismo tiempo, estamos realizando una reducción del espacio de búsqueda.

En dichas tablas se muestran cinco soluciones representativas de la media del Pareto ordenadas de menor a mayor ECM_{entr}: la primera fila de cada problema es la solución más precisa, la segunda fila es el primer cuartil, la tercera fila es la mediana, la cuarta fila es el tercer cuartil y la quinta fila es la solución menos precisa.

Las tablas 3.2, 3.3, 3.4 y 3.5 además contienen los resultados obtenidos por los algoritmos con los que comparamos. Las filas SDJSG s/SV, SDJSG4M s/SV y SVAG muestran el promedio de la última solución más exacta del Pareto obtenido por ejecución que mejora la solución dada por WM. La fila JS contiene el promedio de los resultados obtenidos por el algoritmo.

Un análisis más profundo del algoritmo SDJSG se realizará en la subsección 3.3.4. Con este estudio observaremos si los beneficios de nuestro algoritmo se deben a la estructura jerárquica, a la selección de variables o a ambos. Previamente, analizaremos los resultados obtenidos por SDJSG y los compararemos con los resultados obtenidos mediante el algoritmo JS.

Tabla 3.2.: Resultados obtenidos por SDJSG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
LASER							ELE2						
WM	265,21	278,58	1,0±0,0	58,4	4,0	4,0	WM	112.271	112.719	1,0±0,0	65,0	4,0	4,0
JS	299,46	323,24	6,0±0,0	55,6	2,1	4,0	JS	117.172	127.343	6,0±0,0	36,6	1,3	3,0
SVAG	265,21	278,58	1,0±0,0	58,4	4,0	4,0	SVAG	90.956	100.100	1,0±0,0	36,6	3,0	3,0
SDJSG s/SV	263,74	293,06	1,2±0,4	55,0	3,7	4,0	SDJSG s/SV	98.205	102.994	2,0±0,0	41,6	2,8	4,0
SDJSG4M s/SV	263,74	293,06	1,2±0,4	55,0	3,7	4,0	SDJSG4M s/SV	98.205	102.994	2,0±0,0	41,6	2,8	4,0
	263,74	293,06	1,2±0,4	55,0	3,8	4,0		72.231	75.695	1,4±0,5	15,1	1,9	2,4
	309,01	327,13	1,3±0,3	38,9	2,9	3,3		183.426	185.363	1,3±0,2	12,4	1,7	2,1
SDJSG	342,74	364,76	1,1±0,2	25,3	2,3	2,5	SDJSG	349.446	347.696	1,6±0,4	10,2	1,3	1,9
	694,76	728,33	1,1±0,1	15,0	1,7	1,8		487.700	473.206	1,5±0,3	7,7	1,1	1,5
	1.469,79	1.513,25	1,0±0,0	5,0	1,0	1,0		591.403	565.462	1,0±0,0	5,0	1,0	1,0
DEE							CONCRETE						
WM	0,14117	0,22888	1,0±0,0	178,4	6,0	6,0	WM	73,080	97,006	1,0±0,0	310,4	8,0	8,0
JS	0,23050	0,33550	9,8±4,7	179,6	3,0	6,0	JS	95,540	113,840	13,6±2,3	300,2	3,5	8,0
SVAG	0,13900	0,22390	1,0±0,0	155,2	5,4	5,4	SVAG	72,477	88,984	1,0±0,0	270,0	7,0	7,0
SDJSG s/SV	0,13961	0,22728	1,4±0,5	162,5	5,6	6,0	SDJSG s/SV	71,725	88,160	1,8±1,0	263,4	6,8	8,0
SDJSG4M s/SV	0,13948	0,22713	1,4±0,5	161,3	5,5	6,0	SDJSG4M s/SV	70,355	88,666	1,7±0,9	267,7	6,8	8,0
	0,13728	0,23338	1,1±0,2	155,1	5,6	5,6		50,859	73,521	1,1±0,2	312,5	7,0	7,1
	0,15308	0,22436	1,2±0,3	87,5	4,0	4,2		55,786	71,124	1,2±0,5	231,5	5,4	5,8
SDJSG	0,18253	0,21145	1,5±0,5	38,8	2,7	3,5	SDJSG	70,310	79,036	1,2±0,4	133,7	4,1	4,4
	0,23023	0,23913	1,5±0,6	14,7	1,9	2,4		112,184	117,851	1,2±0,4	48,1	2,6	2,9
	0,34664	0,35492	1,0±0,0	4,5	1,0	1,0		240,195	248,009	1,0±0,0	3,9	1,0	1,0

Tabla 3.3.: Resultados obtenidos por SDJSG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 8L* y *Census 8H*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
CENSUS 8L							CENSUS 8H						
WM	15.234	16.122	1,0±0,0	762,8	8,0	8,0	WM	23.350	24.504	1,0±0,0	713,4	8,0	8,0
JS	14.503	15.188	6,3±0,7	86,7	2,9	4,0	JS	22.038	23.708	11,2±7,0	291,7	3,1	5,0
SVAG	13.468	14.023	1,0±0,0	140,1	5,0	5,0	SVAG	20.903	21.907	1,0±0,0	397,4	5,4	5,4
SDJSG s/SV	12.750	13.090	3,3±0,6	145,0	4,2	8,0	SDJSG s/SV	20.734	21.846	2,7±0,6	419,5	5,6	8,0
SDJSG4M s/SV	12.546	12.974	3,1±0,6	166,4	4,6	8,0	SDJSG4M s/SV	20.648	21.717	2,4±0,6	444,1	6,0	8,0
	12.739	13.211	1,6±0,6	169,4	5,6	6,7		20.284	21.238	1,4±0,6	437,5	6,4	6,8
	13.431	13.876	1,6±0,8	98,3	4,3	5,2		21.195	22.072	1,3±0,4	273,7	4,7	5,2
SDJSG	16.045	16.516	1,6±0,8	39,9	2,9	3,8	SDJSG	22.714	23.242	1,6±0,8	119,3	3,3	4,4
	20.584	20.950	1,8±0,8	19,8	1,8	2,8		25.140	25.270	1,5±0,7	28,6	2,3	2,8
	30.996	31.408	1,0±0,0	5,0	1,0	1,0		30.391	30.345	1,0±0,0	5,0	1,0	1,0
WEANK							MORTGAGE						
WM	9,75605	12,25650	1,0±0,0	456,8	9,0	9,0	WM	0,25600	0,26806	1,0±0,0	197,2	15,0	15,0
JS	13,13610	16,56026	6,9±2,4	100,8	3,0	5,0	JS	0,90399	0,90393	7,1±2,4	80,6	2,7	5,0
SVAG	7,58657	8,84168	1,0±0,0	140,2	5,4	5,4	SVAG	0,11871	0,12598	1,0±0,0	61,5	5,4	5,4
SDJSG s/SV	7,10852	7,35495	3,3±0,6	148,9	4,8	9,0	SDJSG s/SV	0,11376	0,12191	8,1±1,2	101,7	2,9	15,0
SDJSG4M s/SV	7,10044	7,45815	3,0±0,4	170,0	4,2	9,0	SDJSG4M s/SV	0,10971	0,11165	4,0±0,0	131,9	4,2	15,0
	6,63906	6,84172	1,9±0,8	64,8	3,8	5,7		0,10097	0,10848	2,1±1,0	40,7	4,6	7,2
	7,46367	7,49844	2,0±0,8	40,1	3,0	4,9		0,10856	0,11200	2,0±0,9	33,6	3,7	5,5
SDJSG	8,63689	8,62772	1,8±0,6	25,7	2,4	3,7	SDJSG	0,13511	0,13602	1,8±0,9	27,6	2,9	4,3
	10,75790	11,00243	1,5±0,5	13,8	1,9	2,6		0,19417	0,20913	1,5±0,6	16,5	2,0	2,7
	13,66512	13,62635	1,0±0,0	2,1	1,0	1,0		0,32981	0,34130	1,0±0,0	5,0	1,0	1,0

Tabla 3.4.: Resultados obtenidos por SDJSG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 16L* y *Census 16H*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
TREASURY							CENSUS 16L						
WM	0,80168	0,80909	1,0±0,0	195,2	15,0	15,0	WM	32.505	32.887	1,0±0,0	682,0	16,0	16,0
JS	0,19888	0,20635	2,8±1,6	18,8	1,8	2,0	JS	23.815	24.108	6,4±2,3	62,6	3,1	7,0
SVAG	0,16362	0,17451	1,0±0,0	16,8	2,2	2,2	SVAG	21.507	21.804	1,0±0,0	79,1	6,4	6,4
SDJSG s/SV	0,11325	0,11924	10,3±1,1	80,0	1,9	15,0	SDJSG s/SV	19.772	19.972	8,3±1,7	62,0	3,6	16,0
SDJSG4M s/SV	0,13011	0,13808	4,0±0,0	130,2	4,4	15,0	SDJSG4M s/SV	19.797	20.069	3,9±0,1	114,7	5,1	16,0
	0,14692	0,15025	1,5±0,8	17,5	3,0	4,1		17.884	18.163	2,6±1,0	75,1	5,7	10,5
	0,16304	0,17243	1,5±0,7	16,4	2,5	3,4		18.529	18.823	2,5±1,0	50,2	4,5	8,2
SDJSG	0,18550	0,19955	1,6±0,9	14,8	2,0	2,8	SDJSG	20.052	20.273	2,6±0,9	30,9	3,3	6,6
	0,23274	0,24625	1,3±0,5	10,7	1,6	2,0		23.701	23.878	2,2±1,0	16,1	2,4	4,3
	0,31949	0,33830	1,0±0,0	5,0	1,0	1,0		41.698	41.841	1,0±0,0	3,0	1,0	1,0
CENSUS 16H							ELEVATORS						
WM	33.218	33.896	1,0±0,0	754,6	16,0	16,0	WM	0,33258	0,33485	1,0±0,0	511,0	18,0	18,0
JS	28.557	29.022	7,9±2,0	88,8	3,2	7,0	JS	0,28862	0,29267	4,7±1,0	35,5	3,0	4,0
SVAG	24.906	25.319	1,0±0,0	104,8	6,7	6,7	SVAG	0,24786	0,25033	1,0±0,0	44,6	6,0	6,0
SDJSG s/SV	25.757	26.002	8,7±2,1	72,0	3,9	16,0	SDJSG s/SV	0,23387	0,23466	10,6±1,9	58,8	3,4	18,0
SDJSG4M s/SV	24.395	24.758	4,0±0,6	133,0	5,5	16,0	SDJSG4M s/SV	0,23721	0,23774	4,0±0,0	105,1	5,2	18,0
	22.533	22.844	1,8±0,9	125,7	7,4	9,6		0,22158	0,22221	1,7±0,7	21,9	4,3	5,6
	23.182	23.458	1,8±0,9	98,0	5,4	7,6		0,23308	0,23294	2,1±0,9	16,0	3,0	4,9
SDJSG	24.925	25.198	2,3±1,1	46,7	3,6	6,2	SDJSG	0,24437	0,24395	2,0±0,7	11,9	2,4	3,9
	27.946	28.183	1,9±0,9	18,5	2,5	4,0		0,26514	0,26509	1,6±0,5	7,1	1,8	2,5
	50.624	50.712	1,0±0,0	2,7	1,0	1,0		0,31726	0,31712	1,0±0,0	2,9	1,0	1,0

Tabla 3.5.: Resultados obtenidos por SDJSG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; $\#M$, $\#R$ y $\#V$ son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	$\#R$	$\bar{x}_{\#R}$	$\#V$
COMPACT						
WM	81,7173	82,4732	1,0±0,0	425,6	21,0	21,0
JS	44,3714	44,0328	4,3±0,7	21,6	2,5	5,0
SVAG	46,7455	46,5157	1,0±0,0	20,2	5,3	5,3
SDJSG s/SV	28,0853	28,3043	14,2±1,8	56,0	4,0	21,0
SDJSG4M s/SV	35,1213	35,8445	3,9±0,1	124,4	6,2	21,0
SDJSG	12,6598	12,7747	1,4±0,6	45,5	8,4	9,8
	13,1206	13,2273	1,4±0,5	38,9	6,4	7,5
	14,7702	14,9839	1,4± 0,5	27,7	4,6	5,8
	21,2532	21,3664	1,3±0,4	16,8	3,0	3,6
	143,9023	144,5856	1,0± 0,0	3,0	1,0	1,0

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	$\#R$	$\bar{x}_{\#R}$	$\#V$
AILERONS						
WM	0,06452	0,06534	1,0±0,0	1080,6	40,0	40,0
JS	0,05896	0,05931	6,8±1,6	74,3	3,9	7,0
SVAG	0,04887	0,04912	1,0±0,0	92,0	8,1	8,1
SDJSG s/SV	0,06007	0,06049	31,5±3,0	91,3	3,0	40,0
SDJSG4M s/SV	0,04635	0,04695	4,0±0,0	416,1	10,3	40,0
SDJSG	0,04244	0,04263	2,7±0,9	82,2	6,9	10,6
	0,04777	0,04794	2,5±0,9	60,6	4,9	8,3
	0,05728	0,05760	2,5±0,9	30,3	3,5	6,1
	0,06656	0,06735	2,5±0,8	11,2	2,2	4,4
	0,10148	0,10228	1,0±0,0	2,0	1,0	1,0

3.3.3. Análisis

Nuestro objetivo principal es mejorar la interpretabilidad mediante un conjunto de reglas compacto y preciso, mientras que la precisión se mantiene o mejora. Cuando el número de reglas de un SD es bajo, el sistema mejora la interpretabilidad, pero la precisión empeora y viceversa. Esta situación se puede observar claramente en los resultados obtenidos en problemas tanto con un número bajo de variables como con un número considerable de variables. Destacar que el conjunto de reglas difusas se aprende individualmente para cada módulo.

Como se puede apreciar, en los problemas con un número bajo de variables (de cuatro a seis variables) el sistema jerárquico en serie obtenido por nuestro algoritmo tiene mejor precisión que la obtenida por el método de WM. Por otro lado, si observamos los problemas con un número considerable de variables (más de ocho), nuestro algoritmo sigue teniendo mejor rendimiento. En estos problemas, los resultados obtenidos por nuestro algoritmo en precisión y complejidad son mejores, incluyendo el tercer cuartil, que los resultados obtenidos de WM.

En el problema *DEE* podemos observar que la precisión es mejor cuando el número de reglas es alto y, consecuentemente, la interpretabilidad del SD será peor. Comparando con WM, la solución más precisa es mejor en precisión, teniendo un número de reglas y número de variables bajo. En el primer cuartil, la precisión se incrementa un poco con respecto a la solución obtenida por WM en el ECM_{entr}, pero en el ECM_{pru} es más bajo y la interpretabilidad se reduce casi a la mitad.

A continuación abordaremos el problema *Elevators* que tiene 18 variables. La solución más precisa obtenida por nuestro algoritmo es mejor que la solución de WM. Tanto en el ECM_{entr} como en el ECM_{pru}, la interpretabilidad es mejor porque se consiguen SDJS con un número bajo de variables con casi dos módulos y el número de reglas se ha reducido en un 95 %. El ECM_{entr} del tercer cuartil mejora en precisión a WM, el ECM_{pru} es también más preciso y el número de reglas que obtiene nuestro algoritmo se reduce en un 98 %.

En el problema *Ailerons* (40 variables), podemos ver que nuestra solución más precisa es mejor que la solución obtenida por WM. Queremos destacar que el número de reglas se reduce en un 92 % y el número de variables en un 75 %. El ECM_{entr} y el ECM_{pru} de la mediana es bajo y la interpretabilidad es mejor que la solución dada por WM, debido a la reducción del número de variables y su distribución en módulos.

La figura 3.11 muestra el promedio de la frontera del Pareto obtenido por SDJSG en los siguientes problemas de complejos: *Concrete*, *Census 8L*, *Census 16H*, *Eleva-*

tors, *Computer Activity* y *Ailerons*. Los SDJS consiguen una buena precisión cuando el número de módulos tiende a dos. Si el SDJS tiene un número de módulos elevado, el acarreo del error a través de los módulos se incrementará considerablemente. En cuanto a la complejidad, la configuración de variables de entrada en un SDJS por módulo es baja. Por ello, el número total de reglas de un SDJS es bajo y permite una mayor interpretabilidad del sistema.

En las figuras 3.12 y 3.13 podemos observar el ranking de los resultados obtenidos por SDJSG con respecto a los algoritmos JS, SVAG y WM. Del algoritmo SDJSG sólo consideramos la solución más precisa (*SDJSG_M*) y el primer cuartil (*SDJSG_C1*). El ranking se ha configurado de forma que se ha realizado una ordenación independiente ascendente de los algoritmos por ECM de entrenamiento y por número de reglas. A cada algoritmo se le asigna un valor atendiendo a la posición que ocupa según el criterio de ordenación. El eje de ordenadas representa el ranking según el ECM de entrenamiento y el eje de abscisas el ranking atendiendo al número de reglas del sistema. Así, cada punto de cada gráfica representa la posición de un algoritmo en cada uno de los criterios considerados. En cada gráfica hemos representado un cuadrado cuyos vértices son la solución más precisa y el primer cuartil de nuestra propuesta. Esto nos permitirá conocer el rango de ranking existente entre ambos resultados. De esta forma, pretendemos representar gráficamente el potencial de cada uno de los algoritmos.

Como se puede apreciar en las figuras, nuestra propuesta de jerarquía en serie suele ocupar las primeras posiciones del ranking, seguido por SVAG, JS y WM. Solo SVAG en *Census 8H* obtiene una posición intermedia entre la solución más precisa y el primer cuartil obtenido por nuestra propuesta.

Las tablas 3.6 y 3.7 contienen las posiciones numéricas representadas en las figuras 3.12 y 3.13 donde cada columna representa un método, cada fila el problema y la casilla de la tabla definida por ambas contiene la posición de un método con respecto a un problema. La última fila contiene información sobre el ranking medio que ocupa cada algoritmo. En estas tablas podemos comprobar que nuestra propuesta de jerarquía en serie ocupa las primeras posiciones en el ranking. De esta forma, confirmamos el buen comportamiento del algoritmo SDJSG.

La figuras 3.14 y 3.15 ilustran algunos ejemplos de estructuras jerárquicas obtenidas por nuestro algoritmo. Son ejemplos de soluciones no dominadas obtenidas por SDJSG en una partición de un conjunto de datos de los problemas *Census 16H*, *Elevators*, *Computer Activity* y *Ailerons*. El objetivo de esto es observar el efecto

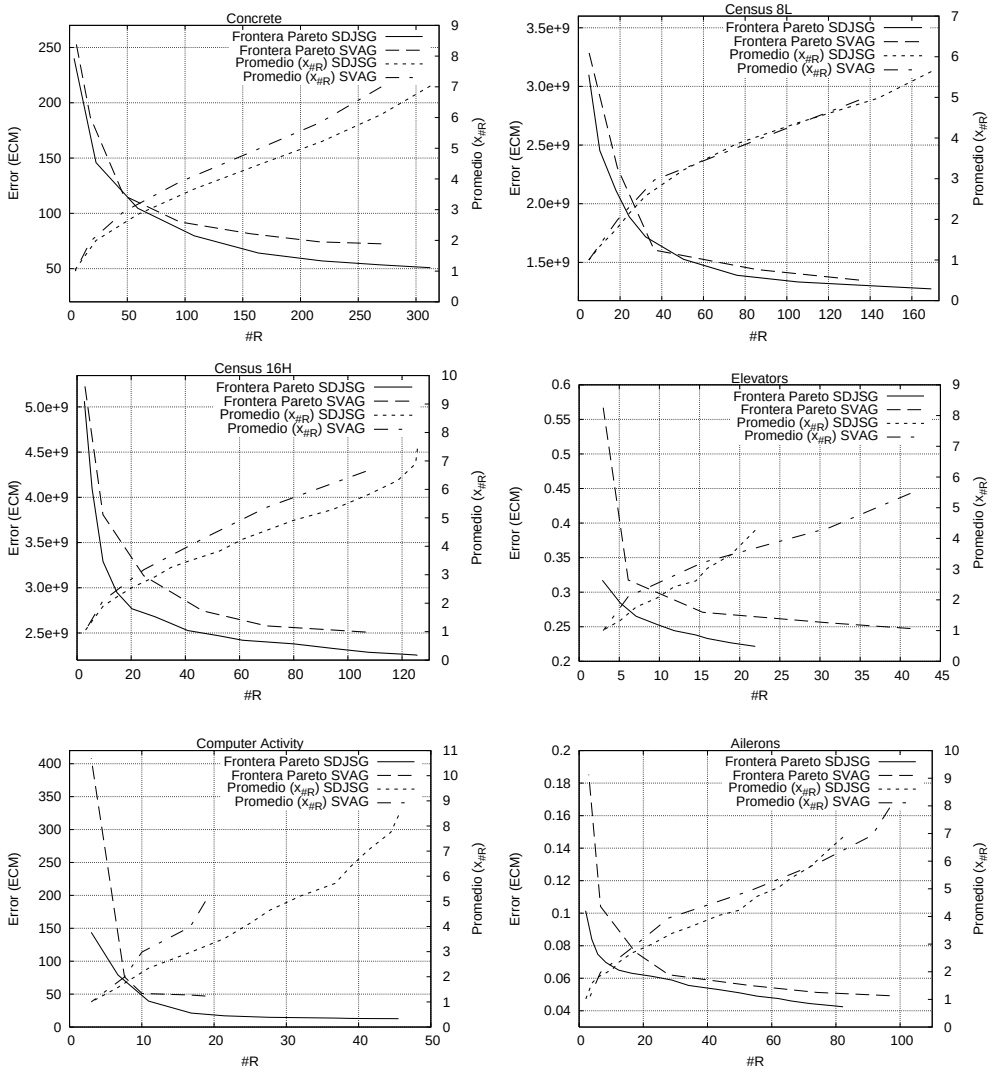


Figura 3.11.: Promedio de la frontera Pareto en varios problemas para el algoritmo SDJSG

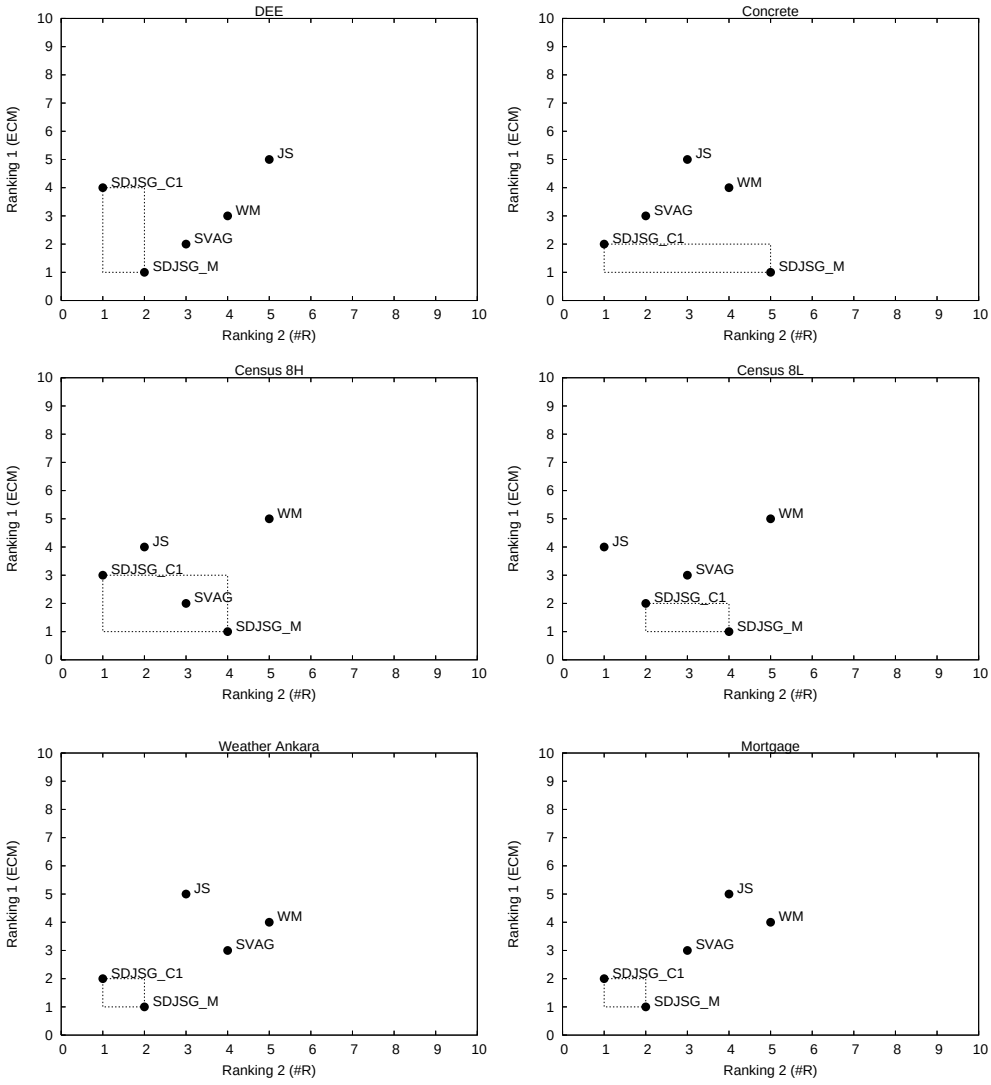


Figura 3.12.: Ranking de algoritmos en varios problemas para el algoritmo SDJSG

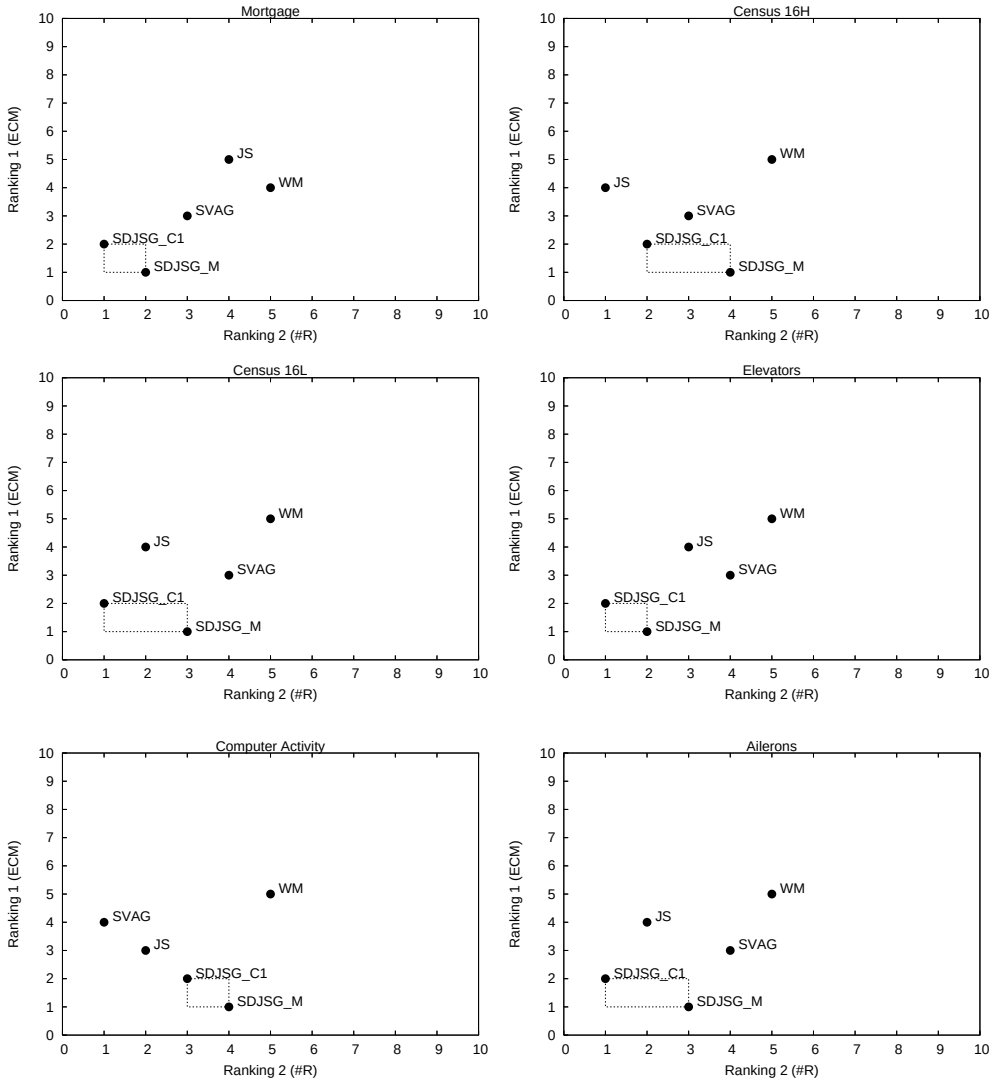


Figura 3.13.: Ranking de algoritmos en varios problemas para el algoritmo SDJSG

Tabla 3.6.: Ranking de algoritmos considerando como criterio ECM_{tra}

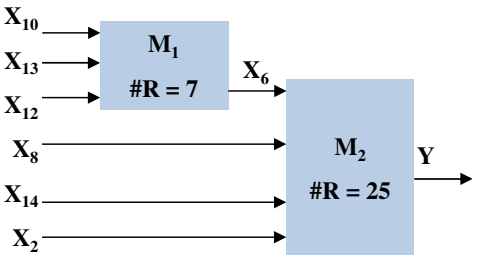
Método / Problema	*WM*	*SVAG*	*JS*	*SDJSG_M*	*SDJSG_C1*
DEE	3	2	5	1	4
Concrete	4	3	5	1	2
Census 8H	5	2	4	1	3
Census 8L	5	3	4	1	2
Weather Ankara	4	3	5	1	2
Mortgage	4	3	5	1	2
Treasury	5	3	4	1	2
Elevators	5	3	4	1	2
Computer Activity	5	4	3	1	2
Ailerons	5	3	4	1	2
Ranking medio	4	3	4	1	2

Tabla 3.7.: Ranking de algoritmos considerando como criterio el número de reglas

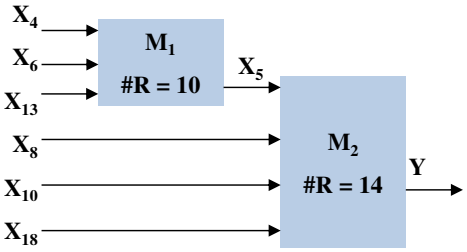
Método / Problema	*WM*	*SVAG*	*JS*	*SDJSG_M*	*SDJSG_C1*
DEE	4	3	5	2	1
Concrete	4	2	3	5	1
Census 8H	5	3	2	4	1
Census 8L	5	3	1	4	2
Weather Ankara	5	4	3	2	1
Mortgage	5	3	4	2	1
Treasury	5	2	4	3	1
Elevators	5	4	3	2	1
Computer Activity	5	1	2	4	3
Ailerons	5	4	2	3	1
Ranking medio	5	3	3	3	1

que produce el módulo de salida de una estructura jerárquica en serie. Por lo tanto, comparamos estas soluciones obtenidas por SDJSG y por WM para el conjunto de variables de entrada del último módulo obtenido a través de SDJSG. Como se puede observar, los valores de ECM_{entr} y ECM_{pru} obtenidos por nuestro algoritmo son menores que los obtenidos por WM sin inferir una variable de entrada endógena en el último módulo, reduciendo el número de reglas en un 6 % en *Census 16H*, 14 % en *Elevators*, 26 % en *Computer Activity* y 16 % en *Ailerons*. El promedio del número de variables por regla es bajo en las soluciones obtenidas por SDJSG y permite una mayor interpretabilidad del sistema porque las reglas son más simples.

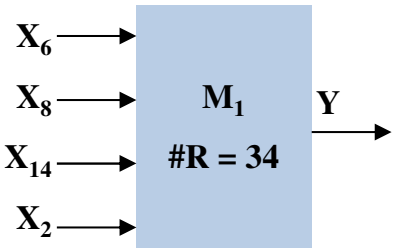
Las soluciones obtenidas por SDJSG son más prometedoras en precisión al igual que en complejidad debido a que busca en áreas locales del espacio de búsqueda y las reglas son más simples. El sistema obtenido es más compacto: tiene un número



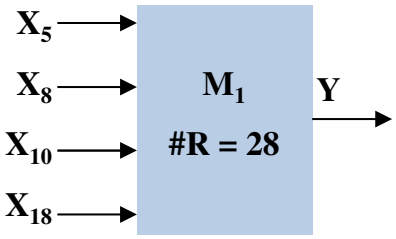
(a) SDJSG en *Census 16H*: ECM_{entr} = 24.345; ECM_{pru} = 25.007; #M = 2; #R = 32; #V = 7; $\bar{x}_{\#R}$ = 3,8



(b) SDJSG en *Elevators*: ECM_{entr} = 0,216918; ECM_{pru} = 0,207847; #M = 2; #R = 24; #V = 7; $\bar{x}_{\#R}$ = 3,6



(c) WM en *Census 16H*: ECM_{entr} = 63.465; ECM_{pru} = 64.826; #M = 1; #R = 34; #V = 4; $\bar{x}_{\#R}$ = 4,0



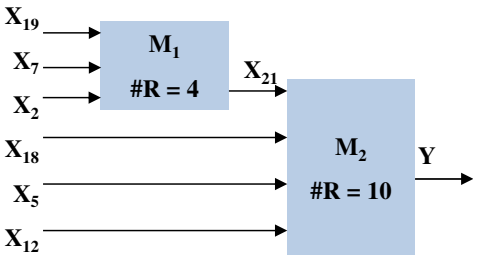
(d) WM en *Elevators*: ECM_{entr} = 0,278183; ECM_{pru} = 0,270400; #M = 1; #R = 28; #V = 4; $\bar{x}_{\#R}$ = 4,0

Figura 3.14.: Ejemplos de soluciones obtenidas por SDJSG (Fig. 3.14(a), 3.14(b)) y las correspondientes soluciones obtenidas por WM sobre el conjunto de variables de entrada del último módulo obtenido por SDJSG (Fig. 3.14(c), 3.14(d)). Los resultados en esta figura (ECM_{entr}/ECM_{pru}) se deben multiplicar por 10^5 en el caso de *Census 16H*

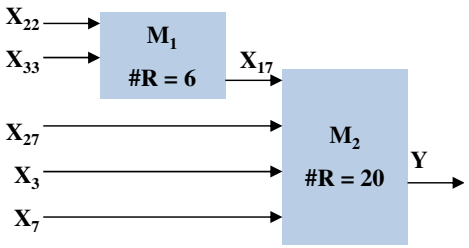
bajo de variables que se distribuye entre distintos módulos y, de esta forma, es más fácil interpretar las reglas aprendidas porque son simples y no hay muchos módulos (el número de módulos está en torno a dos).

La figura 3.16 ilustra un ejemplo real de una base de reglas jerárquica. Esta base de reglas se corresponde con el SDJS de la figura 3.14(b). El problema está formado por 18 variables de entrada. Como se puede apreciar, el número de variables de entrada se ha reducido a siete, distribuidas jerárquicamente en dos módulos.

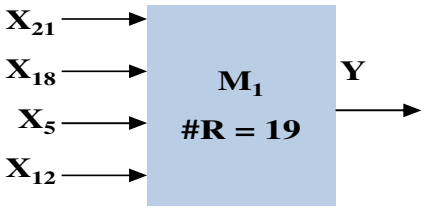
La figura 3.17 es un gráfico de barras que representa la tasa de dependencia de



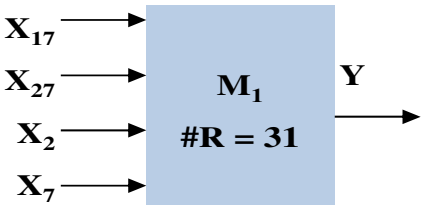
(a) SDJSG en *CompAct*: ECM_{entr} = 14,156858; ECM_{pru} = 13,488079; #M = 2; #R = 14; #V = 7; $\bar{x}_{\#R}$ = 3,7



(b) SDJSG en *Ailerons*: ECM_{entr} = 0,047635; ECM_{pru} = 0,053565; #M = 2; #R = 26; #V = 6; $\bar{x}_{\#R}$ = 3,5



(c) WM en *CompAct*: ECM_{entr} = 115,990035; ECM_{pru} = 119,246400; #M = 1; #R = 19; #V = 4; $\bar{x}_{\#R}$ = 4,0



(d) WM en *Ailerons*: ECM_{entr} = 0,057472; ECM_{pru} = 0,062500; #M = 1; #R = 31; #V = 4; $\bar{x}_{\#R}$ = 4,0

Figura 3.15.: Ejemplos de soluciones obtenidas por SDJSG (Fig. 3.15(a), 3.15(b)) y las correspondientes soluciones obtenidas por WM sobre el conjunto de variables de entrada del último módulo obtenido por SDJSG (Fig. 3.15(c),3.15(d))

variables desde la solución más precisa hasta las soluciones del primer cuartil para todos los problemas. La tasa de dependencia de variables se define como el número de variables diferentes que dependen de otra(s) variable(s) al menos en una de las soluciones más precisas no dominadas (las contenidas en el primer cuartil). Consideramos que una variable X es una variable dependiente si es la salida de uno de los módulos de un SDJS. La tasa ha sido normalizada al porcentaje.

El gráfico de barras muestra que hay algunos problemas complejos con un alto porcentaje de variables dependientes (*Mortgage*, *Census 16L* y *Ailerons*) pero otros no (*Treasury* y *Computer Activity*). Este comportamiento también ocurre en problemas con un número bajo de variables. Por ejemplo, *Ele2* y *Laser* tienen un porcentaje de dependencia mayor que los problemas *DEE* y *Concrete*. También podemos observar que algunos problemas simples (*Laser* y *Ele2*) tienen un porcentaje mayor de

R_1 : **SI** q *es* P **y** absRoll *es* M **y** SaTime4 *es* P **ENTONCES** curRoll *es* G
R_2 : **SI** q *es* P **y** absRoll *es* G **y** SaTime4 *es* P **ENTONCES** curRoll *es* G
R_3 : **SI** q *es* M **y** absRoll *es* P **y** SaTime4 *es* P **ENTONCES** curRoll *es* M
R_4 : **SI** q *es* M **y** absRoll *es* M **y** SaTime4 *es* P **ENTONCES** curRoll *es* G
R_5 : **SI** q *es* M **y** absRoll *es* M **y** SaTime4 *es* M **ENTONCES** curRoll *es* M
R_6 : **SI** q *es* M **y** absRoll *es* G **y** SaTime4 *es* P **ENTONCES** curRoll *es* G
R_7 : **SI** q *es* G **y** absRoll *es* P **y** SaTime4 *es* P **ENTONCES** curRoll *es* G
R_8 : **SI** q *es* G **y** absRoll *es* M **y** SaTime4 *es* P **ENTONCES** curRoll *es* G
R_9 : **SI** q *es* G **y** absRoll *es* M **y** SaTime4 *es* M **ENTONCES** curRoll *es* G
R_{10}: **SI** q *es* G **y** absRoll *es* G **y** SaTime4 *es* P **ENTONCES** curRoll *es* G

(a) Base de Reglas del módulo M_1

R_1 : **SI** curRoll *es* M **y** diffRollRate *es* M **y** SaTime1 *es* P **y** Sa *es* G **ENTONCES** Goal *es* M
R_2 : **SI** curRoll *es* M **y** diffRollRate *es* M **y** SaTime1 *es* M **y** Sa *es* G **ENTONCES** Goal *es* P
R_3 : **SI** curRoll *es* M **y** diffRollRate *es* M **y** SaTime1 *es* G **y** Sa *es* G **ENTONCES** Goal *es* P
R_4 : **SI** curRoll *es* M **y** diffRollRate *es* G **y** SaTime1 *es* M **y** Sa *es* G **ENTONCES** Goal *es* M
R_5 : **SI** curRoll *es* M **y** diffRollRate *es* G **y** SaTime1 *es* G **y** Sa *es* G **ENTONCES** Goal *es* P
R_6 : **SI** curRoll *es* G **y** diffRollRate *es* P **y** SaTime1 *es* P **y** Sa *es* M **ENTONCES** Goal *es* M
R_7 : **SI** curRoll *es* G **y** diffRollRate *es* P **y** SaTime1 *es* P **y** Sa *es* G **ENTONCES** Goal *es* M
R_8 : **SI** curRoll *es* G **y** diffRollRate *es* P **y** SaTime1 *es* M **y** Sa *es* G **ENTONCES** Goal *es* P
R_9 : **SI** curRoll *es* G **y** diffRollRate *es* M **y** SaTime1 *es* P **y** Sa *es* M **ENTONCES** Goal *es* M
R_{10}: **SI** curRoll *es* G **y** diffRollRate *es* M **y** SaTime1 *es* P **y** Sa *es* G **ENTONCES** Goal *es* M
R_{11}: **SI** curRoll *es* G **y** diffRollRate *es* M **y** SaTime1 *es* M **y** Sa *es* G **ENTONCES** Goal *es* M
R_{12}: **SI** curRoll *es* G **y** diffRollRate *es* M **y** SaTime1 *es* G **y** Sa *es* G **ENTONCES** Goal *es* P
R_{13}: **SI** curRoll *es* G **y** diffRollRate *es* G **y** SaTime1 *es* M **y** Sa *es* G **ENTONCES** Goal *es* M
R_{14}: **SI** curRoll *es* G **y** diffRollRate *es* G **y** SaTime1 *es* G **y** Sa *es* G **ENTONCES** Goal *es* P

(b) Base de Reglas del módulo M_2

Figura 3.16.: Base de Reglas obtenida por SDJSG para el problema *Elevators* correspondiente a la figura 3.14(b)

dependencia entre variables que otros problemas complejos (*Treasury* y *Computer Activity*). La dependencia entre variables no está relacionada con el número de variables del problema sino que está ligada a las características del problema. Esto se puede ver claramente en los problemas Census. Las letras (L o H) denotan el nivel de dificultad del problema. Para la dificultad Low (baja), se escogieron las variables de entrada más correladas según un ajuste suave univariante. En la dificultad High (alta), las variables de entrada se seleccionaron de forma que tuvieran una correlación baja. Los problemas *Census 16L* y *Census 16H* tienen el mismo número de variables e instancias, pero podemos observar en el gráfico de barras que la tasa de dependencia de variables en *Census 16L* es más alta que en *Census 16H*. Esto significa que las variables de *Census 16L* están más correladas y el aprendizaje de dependencias entre variables es más fácil. De esta forma, un SDJS puede funcionar bastante bien en problemas tanto simples como complejos, adaptándose a sus características.

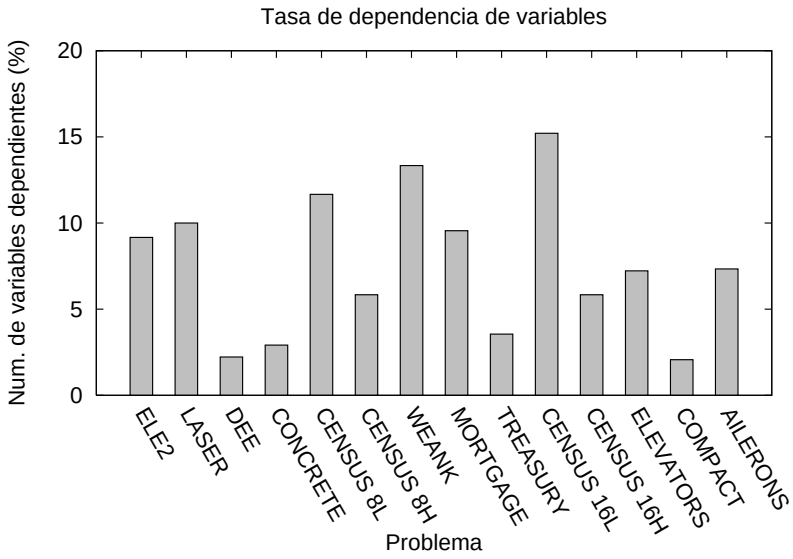


Figura 3.17.: Gráfico de barras de la tasa de dependencia de variables desde la solución más precisa a la solución del primer cuartil

Este comportamiento tan interesante merece un análisis más profundo que el que hemos diseñado. Para ello, sería útil realizar un análisis de la complejidad de los datos de forma similar a como se hizo en clasificación (Ho y Basu 2002; Ho y cols. 2006). Sin embargo, estas medidas de la complejidad de los datos no son fáciles de definir en regresión, así que herramientas como la que proponemos pueden ayudar a proporcionar una visión para comprender la complejidad del problema, más allá del número de variables e instancias.

Las figuras 3.18 y 3.19 ilustran la correlación entre variables mediante un grafo dirigido en varios problemas. Cada nodo se ha etiquetado con el nombre real de la variable y los arcos representan el grado de dependencia entre parejas de variables, donde la cabeza es la entrada de un módulo con una cola como salida que muestra el porcentaje de casos. Por ejemplo, en la figura 3.18(a), la variable *H5.6* depende de *P19.4* en un 13 % (es decir, en un 13 % de los casos el algoritmo obtiene una solución precisa con un módulo donde *H5.6* es la salida y *P19.4* es una de sus entradas). Este grafo puede ayudar a entender la correlación entre las variables de entrada descubiertas por SDJSG. Como se ha indicado anteriormente en la figura 3.17, podemos

observar que las dependencias pueden diferir considerablemente en problemas con el mismo número de variables e instancias como se muestra en las figuras 3.18(a) (*Census 16L*) y 3.18(b) (*Census 16H*).

El algoritmo de JS (Joo y Sudkamp 2009) es una de las propuestas más conocidas en este área. Por esta razón, pensamos que podría ser interesante implementar esta propuesta para compararla con nuestro algoritmo. Las tablas 3.2, 3.3 y 3.4 muestran los resultados obtenidos por JS para los problemas que se consideran. En los problemas con cuatro variables, la solución más precisa obtenida por SDJSG es mejor que la solución obtenida por JS en ECM_{entr} y ECM_{pru} con un número bajo de reglas y módulos. En los problemas de seis a nueve variables de entrada, el primer cuartil o la mediana de SDJSG obtiene mejores resultados en ECM_{entr}, ECM_{pru}, número de módulos, reglas y variables que la propuesta de JS. En problemas complejos, el tercer cuartil de SDJSG obtiene mejor precisión y un número de reglas más bajo que el algoritmo JS.

En *DEE*, la mediana es considerablemente más precisa que la solución obtenida por JS. Con respecto al número de módulos, SDJSG lo reduce en un 85 % y el número de reglas en un 78 %. En *Census 16L*, la solución más precisa obtenida por SDJSG mejora en precisión en un 25 % y decrementa el número de módulos en un 59 %. El número de reglas es un poco más alto. *Ailerons* es el problema con el mayor número de variables que se ha tratado. La mediana obtenida por SJDSG mejora la solución de JS en un 3 % en ECM_{entr} y ECM_{pru}, el número de reglas y módulos se decrementa en un 59 % y un 63 %, respectivamente.

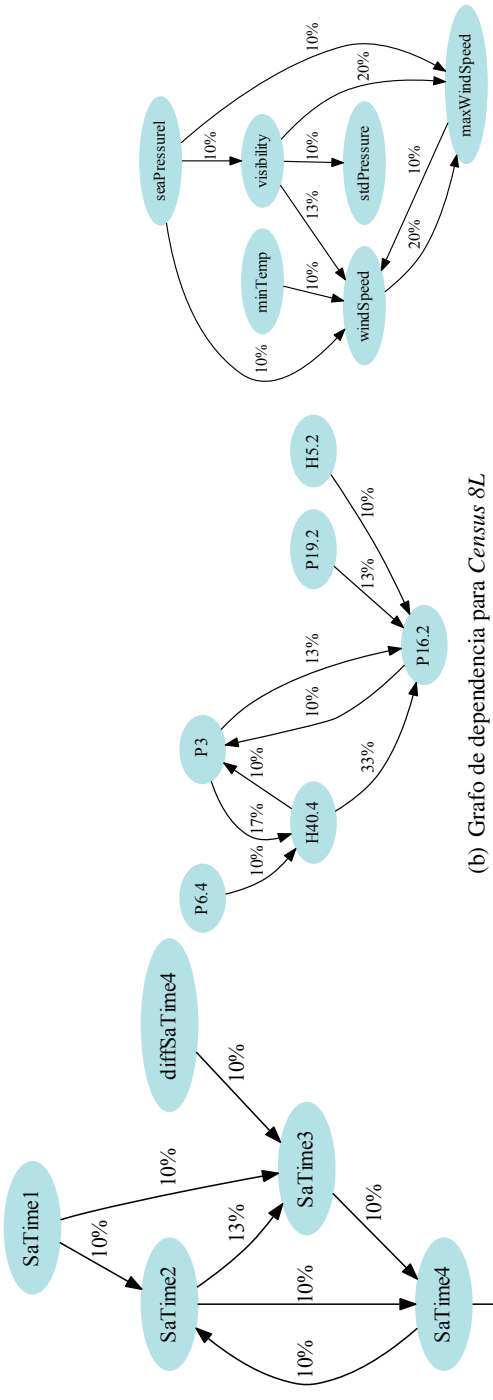
SDJSG mejora los resultados obtenidos por JS y mantiene un equilibrio mejor entre precisión e interpretabilidad. Resumiendo, la propuesta de JS obtiene una solución con un número alto de módulos. Anteriormente, se ha mencionado que el error acarreado aumenta si el número de módulos se incrementa. Por esta razón, la precisión del sistema difuso jerárquico es peor. El algoritmo de JS no elimina variables, por lo que el número de reglas es alto. El SD generado es menos interpretable debido al uso de reglas TSK y necesita una base de reglas completa para crear el sistema difuso jerárquico. Esto origina un problema serio cuando el algoritmo aborda un número de variables alto, como por ejemplo, el problema *Ailerons*.

Sin embargo, SDJSG solventa esos problemas: controla el número de módulos y variables mediante operadores genéticos y no genera nuevas variables para enlazar módulos. De esta forma, el número de reglas es bajo y como las reglas de nuestra propuesta son de tipo Mamdani, son más interpretables. Por estas razones, es más

(a) Grafo de dependencia para Census 16L

(b) Grafo de dependencia para *Census 16H*

Figura 3.18.: Algunos ejemplos de grafos de dependencias encontradas (quedan omitidas las dependencias por debajo del 10 %)



(a) Grafo de dependencia para *Elevators*

(b) Grafo de dependencia para *Census 8L*

(c) Grafo de dependencia para *Weank*

Figura 3.19.: Algunos ejemplos de grafos de dependencias encontradas (quedan omitidas las dependencias por debajo del 10 %)

apropiada en problemas complejos.

3.3.4. Estudio del comportamiento del algoritmo SDJSG

En esta sección, pretendemos analizar las razones de por qué SDJSG obtiene buenos resultados. Para ello, compararemos nuestro algoritmo con SVAG y con SDJSG generando uno, dos, tres y cuatro módulos. El objetivo de este estudio es aislar los beneficios de la selección de variables y comprobar qué mecanismo influye más: la selección de variables o la estructura jerárquica.

La solución más precisa obtenida por SVAG para cada problema se muestra en las tablas 3.2, 3.3 y 3.4. Si comparamos los resultados obtenidos por SVAG y SDJSG en problemas con un número de variables entre cuatro y nueve, podemos observar que no existe una diferencia importante en la precisión de los resultados cuando los problemas tienen cuatro variables (*Laser* y *Ele2*). Si nuestro algoritmo considera problemas complejos y problemas con un número de variables entre seis y nueve, el comportamiento del algoritmo mejora. Podemos observar que los resultados obtenidos por SDJSG son mejores que los resultados de SVAG, tanto en precisión como en simplicidad.

$\bar{x}_{\#R}$ (tablas 3.2, 3.3 y 3.4) es una métrica que distribuye el número de variables, y así, si el número de variables es el mismo, el algoritmo con menor $\bar{x}_{\#R}$ será mejor. De esta forma, podemos observar la simplicidad de SDJSG: el número de variables por regla en nuestro algoritmo es menor que el de SVAG. Esto se debe a que el espacio de búsqueda se trata de manera modular mediante SDJSG.

La figura 3.11 representa el promedio de la frontera del Pareto de SDJSG y SVAG y el valor de $\bar{x}_{\#R}$ en los problemas *Concrete*, *Census 8L*, *Census 16H*, *Computer Activity* y *Ailerons*. Esta figura muestra que el algoritmo SDJSG es mejor en precisión y simplicidad (porque el número de variables por regla en cada módulo es bajo). Si consideramos la métrica $\bar{x}_{\#R}$, el valor de ésta en SDJSG es menor que en SVAG. Podemos observar este comportamiento en el resto de problemas, lo cual indica que las reglas generadas por SDJSG son simples y, consecuentemente, más fáciles de interpretar. En *Census 16H*, podemos apreciar claramente cómo SDJSG es un poco peor en $\bar{x}_{\#R}$ que SVAG cuando el número de reglas está entre 60 y 100 reglas más o menos.

Para obtener más información sobre el comportamiento de SDJSG, hemos experimentado con otra versión de nuestro algoritmo. Cambiamos el criterio de dominancia para obtener soluciones con uno, dos, tres y cuatro módulos. De esta forma,

podemos observar los beneficios que proporciona una estructura jerárquica. La figura 3.20 representa el promedio de las fronteras del Pareto con uno, dos, tres y cuatro módulos obtenidas en los problemas *DEE*, *Census 8L*, *Census 16H*, *Elevators*, *Computer Activity* y *Ailerons* por nuestro algoritmo. En la leyenda de la gráfica se muestra el promedio del tamaño de cada Pareto. Estas figuras ilustran que la combinación de estos cuatro tipos de SDJS consiguen una mejor precisión a medida que el número de reglas es alto, ya que las variables del problema se seleccionan y se distribuyen en el SDJS.

Ahora comparemos el algoritmo SDJSG con SDJSG s/SV y SDJSG4M s/SV. Podemos observar que SDJSG4M s/SV es más preciso que SDJSG s/SV en algunos problemas, aunque el número de reglas se incrementa ligeramente. En problemas complejos, la mejora es más significativa que en problemas con menor número de variables. En *Concrete*, el ECM_{entr} se reduce en un 2 % y el número de reglas se incrementa en un 1,6 % aproximadamente. Sin embargo, en *Mortgage* se mejora la precisión en un 3,6 %, a pesar de que el número de reglas se incrementa en un 23 %. Queremos destacar que SDJSG4M s/SV reduce el ECM_{entr} y el número de reglas obtenidas por WM en un 57 % y un 33 %, respectivamente. De esta forma, podemos verificar que no es bueno tener un SDJS con un número elevado de módulos porque la precisión es peor.

Si comparamos SDJSG con SDJSG s/SV y SDJSG4M s/SV, observaremos que la eliminación de variables mediante operadores genéticos produce mejoras en todos los problemas, particularmente en los problemas complejos. En *Concrete* la mediana mejora la solución obtenida por SDJSG4M s/SV en ECM_{entr} y el número de reglas en un 0,1 % y un 50 % respectivamente.

Si prestamos atención a *CompAct*, la mejora de la solución más precisa obtenida por SDJSG es bastante considerable: el ECM_{entr} se reduce en un 64 % y el número de reglas en un 63 %. Incluso en el tercer cuartil el ECM_{entr} se reduce en un 39 %.

3.3.5. Análisis del tiempo de ejecución

Para completar el análisis, queremos mostrar el tiempo de ejecución de SDJSG. La tabla 3.8 recoge el tiempo de ejecución del algoritmo SDJSG, donde *#VarEntrada* es sinónimo del número de variables de entrada, *#Ejem* identifica el número total de instancias, *#VarWM* indica el promedio del número de variables por módulo (y, por lo tanto, de llamadas al método WM) durante la ejecución completa y *#Tiempo* es el promedio y la desviación típica del tiempo de ejecución. El algoritmo está implemen-

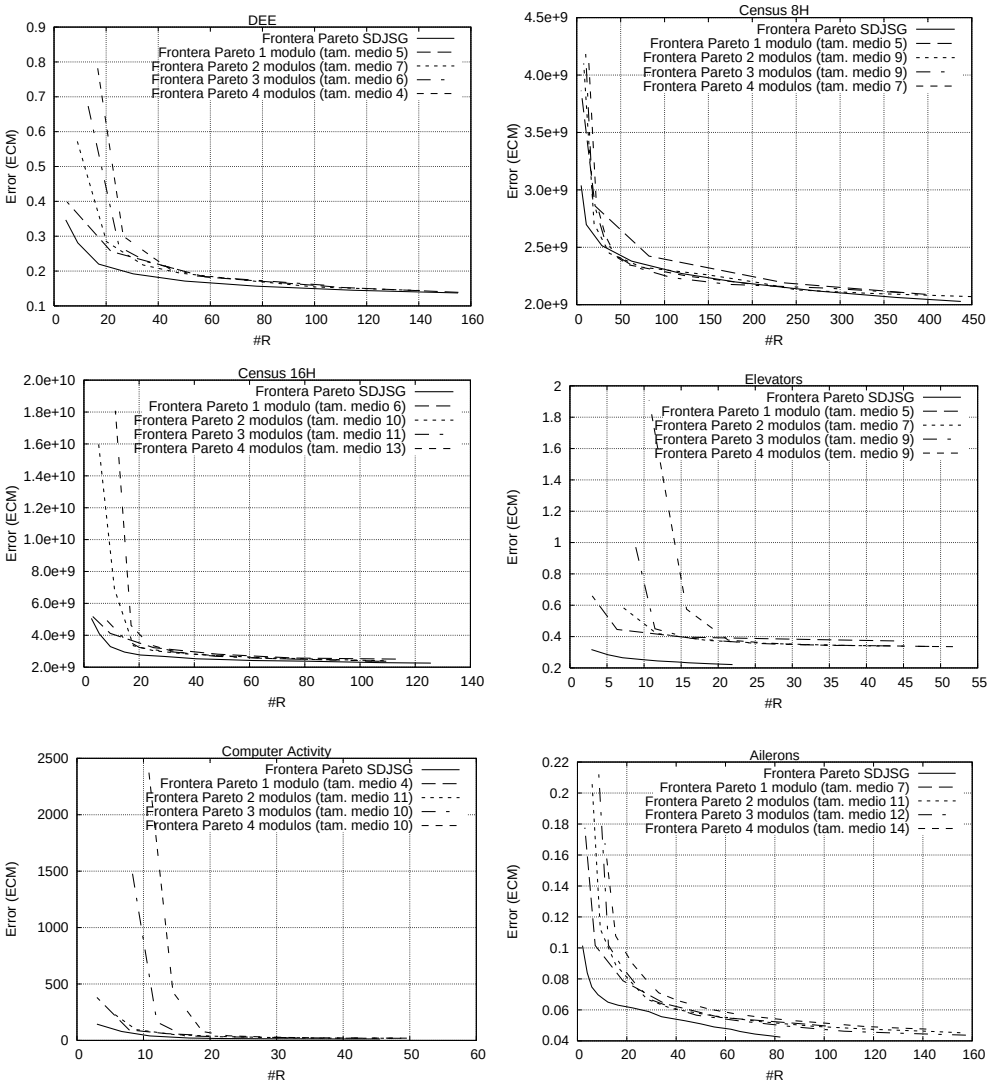


Figura 3.20.: Promedio de la frontera del Pareto con uno, dos, tres y cuatro módulos en varios problemas para el algoritmo SDJSG

tado en lenguaje C y todos los experimentos se ejecutaron en un Intel Core 4 Quad, 2.5-GHz CPU, 8 GB de RAM, y un sistema operativo Linux Red Hat 5.2.

La ejecución del algoritmo dura hasta 12 horas en el peor de los casos. Puede ser

Tabla 3.8.: Tiempo de ejecución del algoritmo SDJSG

Conjunto de datos	*#VarEntrada*	*#Ejem*	*#VarWM*	*Tiempo (hh:mm:ss)*
Ele2	4	1.056	1,2	00:01:27 ± 00:00:12
Laser	4	993	1,6	00:02:03 ± 00:00:27
DEE	6	365	2,3	00:01:24 ± 00:00:17
Concrete	8	1.030	3,1	00:08:37 ± 00:02:41
Census-House 8L (Census 8L)	8	22.784	2,4	10:06:29 ± 01:43:53
Census-House 8H (Census 8H)	8	22.784	2,7	10:43:44 ± 01:14:00
Weather-Ankara (WeAnk)	9	1.609	2,1	00:06:19 ± 00:02:11
Mortgage	15	1.049	2,2	00:03:16 ± 00:00:56
Treasury	15	1.049	1,6	00:01:55 ± 00:00:50
Census-House 16L (Census 16L)	16	22.784	2,7	12:32:40 ± 02:50:47
Census-House 16H (Census 16H)	16	22.784	3,1	11:25:17 ± 02:24:37
Elevators	18	16.559	2,1	05:11:55 ± 01:03:27
Computer Activity (CompAct)	21	8.192	3,8	01:26:59 ± 00:27:10
Ailerons	40	13.750	3,6	08:38:05 ± 02:12:17

un tiempo de ejecución razonable considerando que estamos tratando con algunos conjuntos de datos de regresión complejos (lo cual no es tan frecuente en Sistemas Difusos Genéticos) y hay que tener en cuenta que el algoritmo propuesto está concebido para el aprendizaje fuera de línea.

Podemos encontrar aspectos interesantes si observamos la tabla. De hecho, el tiempo de ejecución depende principalmente del número de instancias del conjunto de datos (ver tabla 3.1). Esto es de esperar ya que el método de WM genera tantas reglas candidatas como instancias de entrenamiento existan, así que esto es una clara dependencia de este parámetro. De esta forma, problemas como *Census*, *Elevators* o *Ailerons* tienen un mayor tiempo de ejecución ya que el número de ejemplos de entrenamiento varía entre 11.000 y 18.228. Sin embargo, es interesante destacar que el número de variables de entrada del problema no es relevante a la hora de determinar el tiempo de ejecución. Por lo tanto, un problema con ocho variables como *Concrete* dura más (sobre ocho minutos) que un problema con 15 variables como es *Treasury* (sobre dos minutos). La explicación de esto se debe encontrar en el número real de variables usadas en cada llamada al método de WM, es decir, el número de variables de los módulos de las estructuras jerárquicas. Como se muestra en la tabla 3.8, este parámetro (el número de variables usado en los módulos durante la ejecución del algoritmo) además influye en el tiempo de ejecución y al final podría representar la dificultad de resolver los problemas más allá del número de variables. El parámetro estaría relacionado con características más complejas tales como la dispersión de los ejemplos o la correlación entre las variables.

3.4. Propuesta de Ajuste sobre SDJSG

Para evaluar la precisión y la interpretabilidad de nuestra propuesta en problemas con un número elevado de variables, vamos a compararla con algunos algoritmos del estado del arte en el área de los Sistemas Difusos Genéticos (SDG). Actualmente, la mayoría de los algoritmos SDG (y particularmente la optimización multiobjetivo) diseñan un ajuste de los parámetros de las funciones de pertenencia para conseguir una precisión adicional. Por ello, para hacer una comparación justa, hemos optado por dotar a nuestra propuesta con la capacidad de ajustar las funciones de pertenencia. Sin embargo, el objetivo principal de este trabajo no es proporcionar un algoritmo que genere modelos difusos con una excelente precisión, sino obtener modelos difusos jerárquicos que organizan las variables de entrada de una forma más comprensible sin sacrificar demasiado la precisión.

Esta sección se organiza de la siguiente forma: la subsección 3.4.1 explica la propuesta de ajuste de las funciones de pertenencia para el algoritmo SDJSG; la subsección 3.5.1 describe los parámetros con los que se ha realizado la experimentación; la subsección 3.5.3 muestra los resultados que se han obtenido y la subsección 3.5.4 analiza y compara nuestro algoritmo con otros conocidos SDG.

3.4.1. Codificación

La propuesta de ajuste de las funciones de pertenencia está formada por dos operadores genéticos que mueven el centro de las funciones de pertenencia para adaptar las particiones difusas, con el objetivo de obtener una mejora adicional en la precisión. Las particiones difusas son triangulares fuertes. El esquema de codificación emplea un vector de números reales, en donde cada posición del vector representa un centro de cada etiqueta de las funciones de pertenencia. La figura 3.21 ilustra un ejemplo de la codificación.

El intervalo de las funciones de pertenencia se calcula mediante el parámetro ε (ecuación 3.8):

$$\varepsilon = \frac{max - min}{k - 1} \tag{3.8}$$

siendo *max* y *min* el máximo y mínimo del intervalo de la variable respectivamente y k el número de etiquetas por variable. De esta forma, el centro de la primera etiqueta es el *mínimo*, el centro de la segunda etiqueta es *mínimo*$+\varepsilon$, el centro de la tercera etiqueta es *mínimo*$+2\varepsilon$, y así sucesivamente.

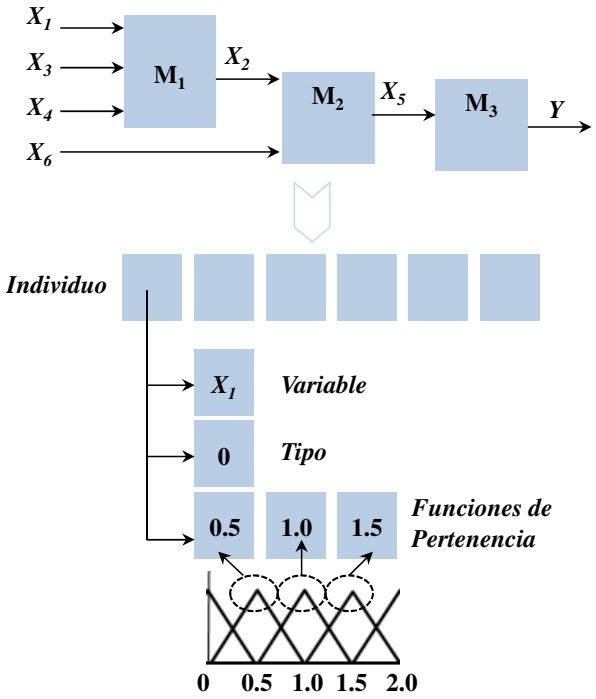


Figura 3.21.: Ejemplo de esquema de codificación de partición difusa

3.4.2. Inicialización

El punto de partida para el ajuste se realiza inicializando la primera mitad de la población con funciones de pertenencia predefinidas y la segunda mitad restante aleatoriamente mediante el algoritmo 6.

3.4.3. Operador de cruce

Este operador genético se aplica por variable a un descendiente obtenido mediante el operador de cruce del algoritmo SDJSG, en otras palabras, está incrustado en el operador de cruce del algoritmo SDJSG. Se definen tres tipos de cruces para el ajuste que dependen de la variable común seleccionada en el descendiente:

1. Si la variable heredada por el descendiente proviene de los dos padres, se aplica un operador de cruce BLX-α por etiqueta. La figura 3.22 ilustra este tipo cruce.

Algoritmo 6 Inicialización aleatoria de las funciones de pertenencia

```
d = 1/k {Siendo k el número de etiquetas por variable}
Para cada individuo hacer
  Para var = 1 hasta N hacer
    {Siendo N el número de variables}
    Para l = 1 hasta etiquetas − 2 hacer
      r=Elegir_aleatoriamente(var, l) {Número aleatorio en el intervalo de la etiqueta}
      Generar_nuevo_centro(var,l,r)
    Fin Para
  Fin Para
  Para var = 1 hasta N hacer
    {Siendo N el número de variables}
    Para l = 1 hasta etiquetas − 2 hacer
      p=Obtener_centro(var, l)
      p_derecho=Obtener_centro(var, l+1)
      distancia=p_derecho-p
      Si ( distancia < (d/2)) entonces
        {Las etiquetas están cerca}
        p_derecho=p_derecho+(d/2)
        p=p-(d/2)
        Generar_nuevo_centro(var,l,p)
      Fin Si
    Fin Para
  Fin Para
Fin Para
```

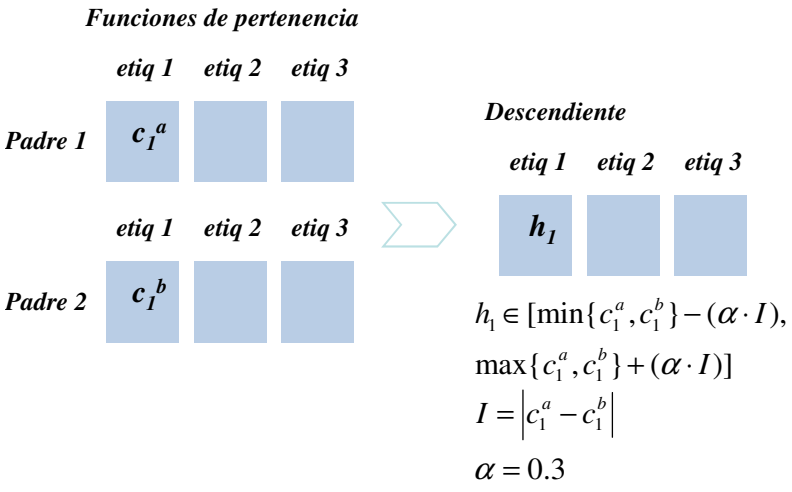


Figura 3.22.: Operador de cruce de las funciones de pertenencia

2. Si la variable heredada por el descendiente proviene de uno de los dos padres, los valores de las funciones de pertenencia del padre se copian en la variable heredada por el hijo.

Finalmente, si los centros de las etiquetas de las funciones de pertenencia se solapan, el vector con los centros de las etiquetas se ordenará de menor a mayor valor.

3.4.4. Operador de mutación

Este operador genético realiza una mutación a nivel de etiqueta de la siguiente forma:

1. Se selecciona un individuo *i* aleatoriamente.

2. Se elige aleatoriamente una variable *x* de *i*.

3. Se elige aleatoriamente una etiqueta *l* de *x*.

4. Se selecciona un número aleatorio *r* que pertenezca al intervalo de la etiqueta *l*.

5. Comprobar si el centro r de la etiqueta se solapa con los centros de ambos lados. En caso afirmativo, el nuevo centro de la etiqueta será la media aritmética de los centros de las etiquetas localizadas a ambos lados de r.

3.4.5. Uso de la metodología de ajuste

Para realizar un ajuste efectivo de las funciones de pertenencia, realizamos varios estudios en donde consideramos dos posibilidades: un aprendizaje simultáneo y un aprendizaje secuencial de la estructura jerárquica y de la base de reglas. Tras un estudio previo experimental, concluimos que un aprendizaje temprano de las funciones de pertenencia y simultáneo con la estructura jerárquica de la solución, producía una convergencia prematura a óptimos locales del algoritmo, cuyas soluciones no presentaban una jerarquía bien formada. Por ello, nos decantamos por un modelizado secuencial, en donde aplicamos los operadores genéticos de ajuste de las funciones de pertenencia después del aprendizaje de la estructura jerárquica transcurriendo un 25 %, 50 % y 75 % del total de evaluaciones asignadas al algoritmo. Posteriormente, pudimos comprobar que para obtener soluciones prometedoras, debíamos posponer la aplicación del ajuste de las funciones de pertenencia, de forma que cuando la jerarquía presente estabilidad, será el momento de aprenderlas. Finalmente, esta estabilidad estructural de la solución se cumple en promedio cuando en el algoritmo han transcurrido el 75 % del total de evaluaciones.

3.5. Estudio experimental del algoritmo de ajuste

3.5.1. Configuración experimental

Los experimentos que se muestran en esta sección se han realizado mediante validación cruzada con cinco particiones del conjunto de datos y seis ejecuciones por partición (es decir, un total de 30 ejecuciones por problema). El algoritmo se ha ejecutado con la siguiente configuración paramétrica: 100.000 y 300.000 evaluaciones, un tamaño de población de 60 individuos, una probabilidad de cruce de 0,7, una probabilidad de mutación de 0,2 por cromosoma y el parámetro α del cruce BLX se ha inicializado a 0,3. Con respecto a las funciones de pertenencia, las particiones difusas son triangulares distribuidas uniformemente y constan de cinco conjuntos difusos por cada variable lingüística. Por otro lado, el número máximo de módulos que genera el algoritmo por estructura jerárquica es de cuatro.

Para adaptar los experimentos del algoritmo al resto de algoritmos con los que comparamos, se ha considerado un total de 100.000 evaluaciones para los conjuntos de datos *Ele2*, *Weather Ankara*, *Mortgage*, *Treasury* y *Computer Activity*; sin embargo, para los conjuntos de datos *Ele2*, *Weather Ankara*, *Mortgage* y *Computer Activity* se ha establecido un ciclo de vida de 300.000 evaluaciones para el algoritmo.

3.5.2. Descripción de los algoritmos de comparación

Para la comparación, se han incluido los siguientes SDG:

- Tres métodos basados en un solo objetivo: GR-MF (Cordón, Herrera, y Villar 2001) aprende la granularidad para cada partición difusa y los parámetros de las funciones de pertenencia. GA-WM (Cordón, Herrera, Magdalena, y Villar 2001) aprende la granularidad, factores de escala y los dominios para cada variable del sistema. GLD-WM (Alcalá, Alcalá-Fdez, Herrera, y Otero 2007) aprende la base de conocimiento mediante la obtención de la granularidad y los desplazamientos laterales individuales de las funciones de pertenencia. Los resultados de estos tres algoritmos se han extraído de (Alcalá, Gacto, y Herrera 2011).
- Tres versiones del algoritmo de dos objetivos (2+2)M-PAES: PAES-RB aprende únicamente las reglas. PAES-SF y PAES-SFC aprenden concurrentemente la base de reglas y los parámetros de las funciones de pertenencia usando una transformación lineal por tramos. Estos algoritmos se han tomado del trabajo realizado por Antonelli y cols. en 2011.
- Un algoritmo evolutivo de tres objetivos: PAES-SF3 (Antonelli y cols. 2011) busca diferentes equilibrios entre complejidad, precisión e integridad en la partición mediante un aprendizaje concurrente de la base de reglas y los parámentros de las variables lingüísticas.

3.5.3. Resultados

Los resultados de la experimentación se recogen en las tablas 3.9 y 3.10, en donde ECM_{entr} y ECM_{pru} son los valores de error de aproximación (ecuación 3.1) de entrenamiento y prueba respectivamente; *#M*, *#R* y *#V* hacen referencia al número de módulos, el número de reglas difusas y el número de variables de entrada respectivamente; $\sigma_{\#M}$ es la desviación típica del número de módulos y $\bar{x}_{\#R}$ es el número medio

de variables por regla. Además, podemos observar los resultados de las soluciones más precisas obtenidas por nuestra propuesta, llamada SDJSG-Ajuste, GR-MF, GA-WM y GLD-WM. Estos algoritmos se han ejecutado durante 100.000 evaluaciones y con cinco conjuntos difusos por cada variable lingüística. Estas tablas además muestran el *primer* punto obtenido por PAES-RB, PAES-SF, PAES-SFC y PAES-SF3, que se han ejecutado durante 300.000 evaluaciones y con cinco conjuntos difusos por cada variable lingüística. La mayoría de los conjuntos de datos de (Antonelli y cols. 2011) tienen un tamaño pequeño. El algoritmo SDJSG-Ajuste se ha ejecutado durante 100.000 y 300.000 evaluaciones para compararlo con estos algoritmos. Las tablas muestran la solución más precisa *Min* y una del primer cuartil *C1* ordenadas ascendentemente por error.

3.5.4. Análisis

En esta subsección, analizaremos la capacidad de las estructuras jerárquicas frente a otros algoritmos SDG que no generan estructuras jerárquicas.

Podemos observar que en los problemas de cuatro a 15 variables, tanto la precisión como el número de reglas obtenido por SDJSG-Ajuste ocupan puestos intermedios con respecto al resto de algoritmos. Sin embargo, en problemas con un número considerable de variables (de 18 a 40 variables), la propuesta es más precisa, pero presenta un número alto de reglas. No obstante, destacamos que el número de variables por regla es menor en el algoritmo SDJSG-Ajuste que en los distintos algoritmos evolutivos multiobjetivo implementados por Antonelli y cols. (2011). Esto nos indica que aunque el número de reglas obtenido es alto, cada regla a nivel individual es más simple.

Además, podemos observar que con 100.000 evaluaciones, los errores de SDJSG-Ajuste son normalmente más altos que los errores de GLD-WM, pero el número de reglas es significativamente menor, por ejemplo, en un 70 % en el problema *Ele2*.

Si comparamos con los algoritmos basados en PAES ejecutados durante 300.000 evaluaciones, la precisión de SDJSG-Ajuste sólo mejora al algoritmo PAES-RB en el problema *Ele2*, pero la simplicidad es alta con respecto al número de reglas como en el número de variables por regla, disminuyendo en un 65 % y 50 %, respectivamente.

Si echamos un vistazo al problema *Computer Activity*, podemos observar que SDJSG-Ajuste obtiene resultados más precisos. La precisión se mejora en un 25 % con respecto a los resultados obtenidos por PAES-SF. Por otro lado, SDJSG-Ajuste incrementa el número de reglas en un 97 %, pero queremos destacar que el número

Tabla 3.9.: Resultados obtenidos por SDJSG-Ajuste con 100.000 y 300.000 evaluaciones y cinco conjuntos difusos por cada variable lingüística

#Eval	*Método*		ECM_{entr}	ECM_{pru}	*#M*	*#R*	$\bar{x}_{\#R}$	*#V*
	ELE2							
100.000	WM		112.270	112.718	1,0	65,0	4,0	4,0
	GR-MF		33.290	37.274	1,0	97,0	4,0	4,0
	GA-WM		34.460	37.954	1,0	47,0	4,0	4,0
	GLD-WM		22.966	26.768	1,0	33,0	4,0	4,0
	SDJSG-Ajuste	Min	30.933	31.682	1,1	9,9	2,0	2,1
		C1	130.626	130.553	1,2	8,5	1,7	1,8
300.000	PAES-RB		30.908	34.270	1,0	30,0	4,0	4,0
	PAES-SF		27.197	31.990	1,0	26,2	4,0	4,0
	PAES-SFC		24.464	28.521	1,0	29,1	4,0	4,0
	PAES-SF3		24.543	28.759	1,0	27,0	4,0	4,0
	SDJSG-Ajuste	Min	30.504	31.036	1,0	9,4	1,9	2,0
		C1	133.398	133.028	1,1	8,1	1,7	1,8
	WEANK							
100.000	WM		9,756	12,610	1,0	457,0	9,0	9,0
	GR-MF		2,812	14,762	1,0	397,0	9,0	9,0
	GA-WM		3,044	5,640	1,0	279,0	9,0	9,0
	GLD-WM		2,222	4,150	1,0	133,0	9,0	9,0
	SDJSG-Ajuste	Min	4,567	5,073	1,4	24,5	2,4	3,3
		C1	6,084	6,424	1,3	16,0	2,1	2,3
300.000	PAES-RB		5,400	7,000	1,0	28,3	9,0	9,0
	PAES-SF		3,160	3,980	1,0	15,9	9,0	9,0
	PAES-SFC		2,880	3,620	1,0	20,9	9,0	9,0
	PAES-SF3		3,200	3,900	1,0	18,1	9,0	9,0
	SDJSG-Ajuste	Min	4,390	4,730	1,7	18,9	2,2	3,3
		C1	6,060	6,330	1,4	14,5	2,0	2,6
	MORTGAGE							
100.000	WM		0,256	0,268	1,0	199,0	15,0	15,0
	GR-MF		0,060	0,352	1,0	209,0	15,0	15,0
	GA-WM		0,040	0,186	1,0	160,0	15,0	15,0
	GLD-WM		0,032	0,044	1,0	78,0	15,0	15,0
	SDJSG-Ajuste	Min	0,065	0,073	1,5	19,8	2,5	3,3
		C1	0,094	0,105	1,3	15,5	2,1	2,6
300.000	PAES-RB		0,120	0,140	1,0	12,9	15,0	15,0
	PAES-SF		0,060	0,100	1,0	9,0	15,0	15,0
	PAES-SFC		0,080	0,160	1,0	10,1	15,0	15,0
	PAES-SF3		0,060	0,080	1,0	9,0	15,0	15,0
	SDJSG-Ajuste	Min	0,060	0,060	1,4	20,8	2,7	3,3
		C1	0,080	0,090	1,3	16,6	2,2	2,8

de variables por regla es más bajo, y por lo tanto se consideran reglas más simples. Además, señalar que la precisión es mejorada por las soluciones del primer cuartil y que el número de reglas se reduce a más de la mitad.

Resumiendo, los resultados obtenidos por SDJSG-Ajuste son más precisos cuando los problemas tienen una complejidad considerable, pero el número de reglas

Tabla 3.10.: Resultados obtenidos por SDJSG-Ajuste con 100.000 y 300.000 evaluaciones y cinco conjuntos difusos por cada variable lingüística. GR-MF, GA-WM y GLD-WM no se encuentran disponibles para el problema *CompAct*. PAES-RB, PAES-SF, PAES-SFC y PAES-SF3 no están disponibles para el problema *Treasury*

#Eval	*Método*		ECM_{entr}	ECM_{pru}	*#M*	*#R*	$\bar{x}_{\#R}$	*#V*
	TREASURY							
100.000	WM		0,802	0,810	1,0	196,0	15,0	15,0
	GR-MF		0,132	0,288	1,0	189,0	15,0	15,0
	GA-WM		0,090	0,128	1,0	136,0	15,0	15,0
	GLD-WM		0,066	0,090	1,0	70,0	15,0	15,0
	SDJSG-Ajuste	Min	0,096	0,105	1,3	15,2	2,2	2,8
		C1	0,106	0,119	1,3	13,4	1,9	2,4
300.000	PAES-RB		–	–	–	–	–	–
	PAES-SF		–	–	–	–	–	–
	PAES-SFC		–	–	–	–	–	–
	PAES-SF3		–	–	–	–	–	–
	SDJSG-Ajuste	Min	–	–	–	–	–	–
		C1	–	–	–	–	–	–
	COMPACT							
100.000	WM		16,898	24,880	1,0	1539,0	21,0	21,0
	GR-MF		–	–	–	–	–	–
	GA-WM		–	–	–	–	–	–
	GLD-WM		–	–	–	–	–	–
	SDJSG-Ajuste	Min	9,891	10,727	1,7	247,2	7,4	9,1
		C1	10,641	11,131	1,6	153,2	5,2	6,5
300.000	PAES-RB		35,040	37,820	1,0	7,0	21,0	21,0
	PAES-SF		12,980	14,320	1,0	6,3	21,0	21,0
	PAES-SFC		19,080	20,000	1,0	8,4	21,0	21,0
	PAES-SF3		13,140	13,340	1,0	11,6	21,0	21,0
	SDJSG-Ajuste	Min	9,730	10,610	2,0	251,3	7,0	9,4
		C1	10,310	10,840	1,8	122,6	4,7	6,5

es más alto. Además, comparamos el algoritmo SDJSG-Ajuste con algoritmos que aprenden el número de etiquetas, reduciendo considerablemente el número de reglas.

Las figuras 3.23 y 3.24 muestran el ajuste de las funciones de pertenencia realizado por el algoritmo en una solución obtenida en una ejecución para los problemas *Computer Activity* y *Mortgage* durante 100.000 y 300.000 evaluaciones, respectivamente.

3.6. Sumario

Hemos dedicado este capítulo a la propuesta de una nueva metodología para el aprendizaje de estructuras jerárquicas en serie con las siguientes características:

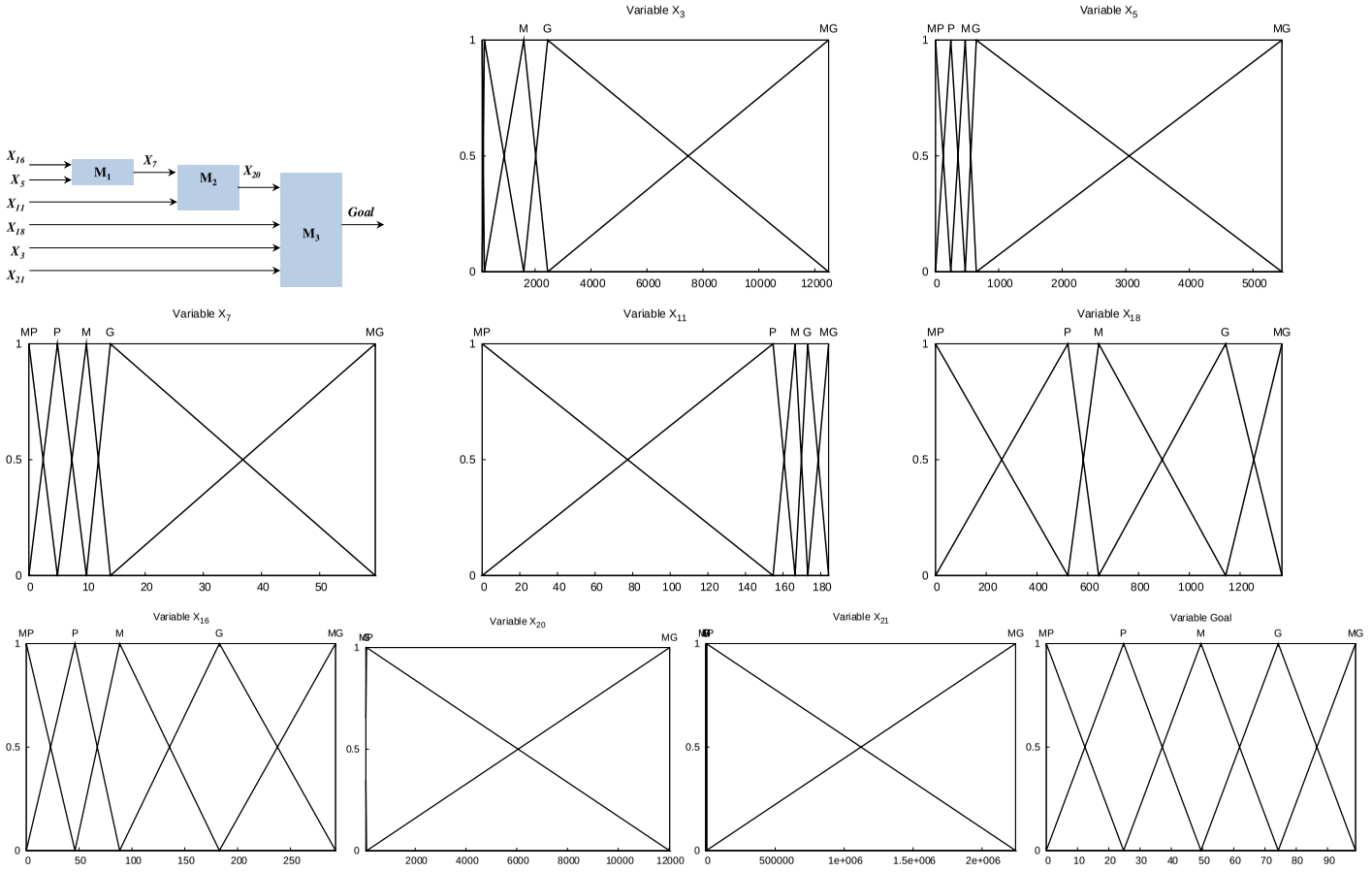


Figura 3.23.: Funciones de pertenencia generadas por SDJSG en el problema Computer Activity. #M=3. #R antes del ajuste = 148. #R después del ajuste = 78. $ECM_{entr/prue}$ antes del ajuste = 42,608/43,798. $ECM_{entr/prue}$ después del ajuste= 10,095/10,915

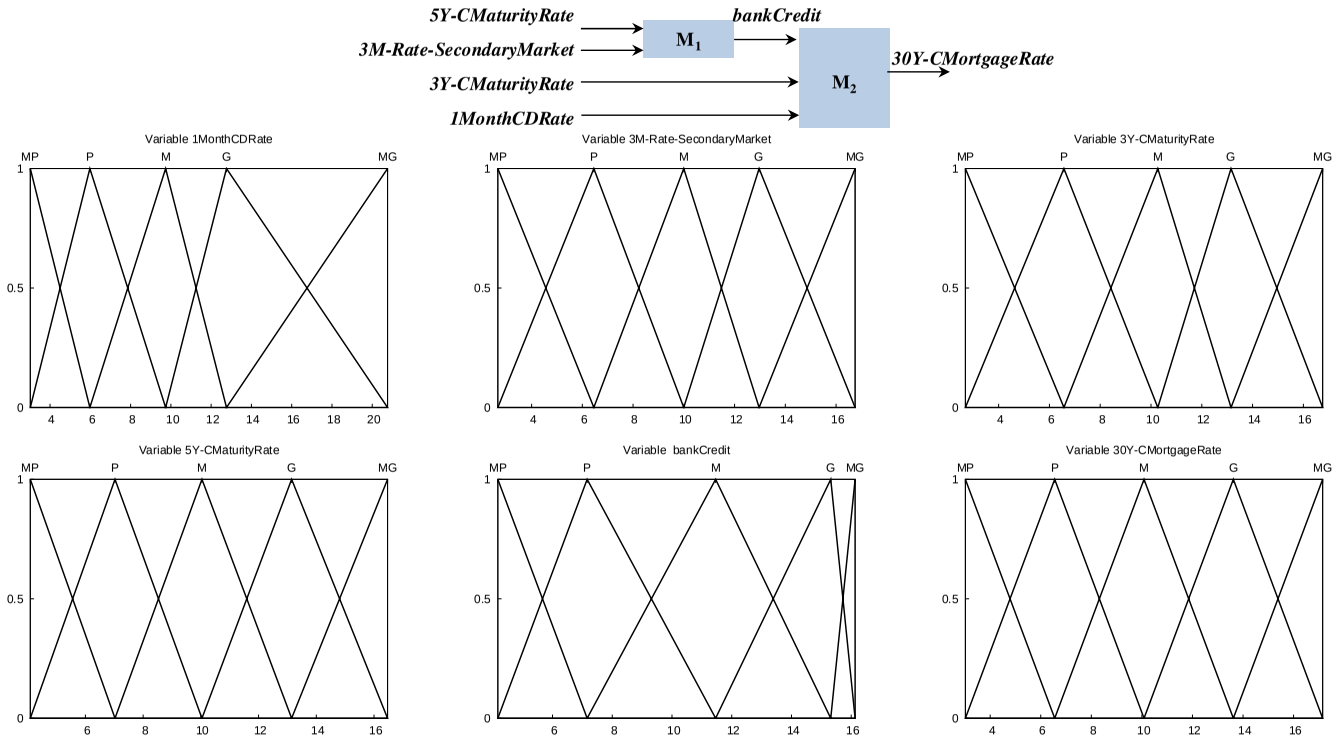


Figura 3.24.: Funciones de pertenencia generadas por SDJSG en el problema *Mortgage*. #M = 2. #R antes del ajuste = 39. #R después del ajuste = 33. $ECM_{entr/prue}$ antes del ajuste =$0,347/0,368$. $ECM_{entr/prue}$ después del ajuste= 0,066/0,069

- Aprende la estructura jerárquica en serie mediante un algoritmo multiobjetivo para paliar el problema de la alta dimensionalidad en los sistemas difusos.
- Obtiene modelos en donde la precisión se mantiene o se mejora.
- Los modelos lingüísticos generados presentan una buena interpretabilidad en dos aspectos: 1) a nivel de regla, porque el número de variables por regla es bajo; y 2) existe una disminución en el número de reglas.
- El aprendizaje de la estructura es flexible, lo que permite la adaptación de la estructura a las necesidades del problema.
- La limitación del espacio de búsqueda en el número de módulos en la jerarquía permite obtener buenas soluciones de forma rápida.
- La precisión de estos sistemas se puede mejorar con un pequeño ajuste a nivel de etiqueta.
- Los modelos obtenidos por nuestra propuesta con respecto a los métodos con los que comparamos son más compactos y transparentes, manteniendo o mejorando la precisión de los modelos obtenidos por otras técnicas de modelizado.

4. Modelizado de Sistemas Mediante Sistemas Difusos Jerárquicos Paralelos e Híbridos

Cada proceso enseña algo que necesitamos aprender.

CHARLES DICKENS (1.812 – 1.870), NOVELISTA INGLÉS

En el capítulo anterior pudimos comprobar que el hecho de usar una estructura jerárquica en serie para modelizar un sistema difuso convencional convierte una base de reglas, cuyo crecimiento en el número de reglas atiende a una función de tipo exponencial, en una base de reglas en donde el crecimiento del número de reglas sigue una función lineal. Simultáneamente, la estructura jerárquica disminuye la complejidad lingüística a nivel de regla. Todo esto se traduce en una mejora en la interpretabilidad, en donde además se mantiene o se mejora la precisión del sistema.

Estos resultados nos motivan para seguir con el estudio de otros dos modelos jerárquicos: paralelo e híbrido. Como veremos a lo largo de este capítulo, se aprenderán modelos jerárquicos específicos con un cierto grado de libertad con respecto a un conjunto de criterios predefinidos con el objetivo de no perder precisión ni legibilidad.

Este capítulo se estructura en dos secciones definidas: una sección está destinada a la descripción del modelizado de la jerarquía paralela y otra al modelizado jerárquico híbrido. El interior de cada sección incluye una descripción detallada del diseño de cada algoritmo de aprendizaje de cada estructura jerárquica, en donde se puede distinguir la codificación del problema, los operadores genéticos aplicados que evolucionan las jerarquías, la configuración paramétrica con la que se ha realizado la experimentación, los resultados obtenidos tras la ejecución del algoritmo y un análisis

detallado de su comportamiento, en donde comprobaremos su eficacia. El esquema básico del algoritmo para el modelizado de ambas estructuras jerárquicas, al igual que el modelizado jerárquico en serie, se especifica al principio del capítulo 3.

4.1. Modelizado de Sistemas Difusos Jerárquicos en Paralelo

En esta sección se realiza tanto una descripción detallada de nuestra propuesta jerárquica paralela como la presentación y análisis de los resultados obtenidos, además de una comparativa frente a otros métodos. Esto se irá realizando a lo largo de los siguientes apartados.

4.1.1. Descripción del algoritmo

4.1.1.1. Codificación

Para abordar este tipo de estructuras, creemos que es más adecuado cambiar el tipo de representación con respecto a la usada en el capítulo 3. De esta forma consideramos cada estructura jerárquica paralela como un árbol n-ario, donde cada árbol puede tener n hijos. Y. Chen y cols. (2004), X. Zhang y Zhang (2006), Y. Chen y cols. (2007) apuestan por este tipo de representación de soluciones, ya que es una estructura de datos que permite flexibilidad a la hora de modelizar la jerarquía y una clara identificación de los elementos de la jerarquía del sistema difuso: el nodo raíz del árbol se corresponde con el módulo de salida del sistema jerárquico paralelo del que cuelgan el resto de módulos formando las sucesivas capas. La figura 4.1 muestra la similitud entre el sistema difuso jerárquico paralelo (figura 4.1(a)), su correspondencia en árbol (figura 4.1(b)) y su codificación (figura 4.1(c)). En el árbol, las variables contenidas en los nodos intermedios (X_7, X_8, X_{11}, X_{15}, X_4 y X_{13}) dependen de las variables de los nodos hoja (X_{10}, X_3, X_5, X_6, X_9, X_{14}, X_1 y X_2), y la variable de salida (Y), situado en el nodo raíz, se genera a partir de los nodos intermedios. En el sistema difuso jerárquico paralelo existe la misma dependencia entre variables, por lo que como se aprecia, la elección de una estructura de datos en árbol es muy representativa y adecuada. Este sistema difuso jerárquico en paralelo considera un total de 15 variables, de las cuales ocho son variables exógenas, seis son variables endógenas (la variable X_{12} no se considera en esta estructura) distribuidas en tres capas bien definidas. Como se puede comprobar, no todas las variables del problema se usan en la

estructura jerárquica, aunque también cabe la posibilidad de que todas las variables del problema se usen.

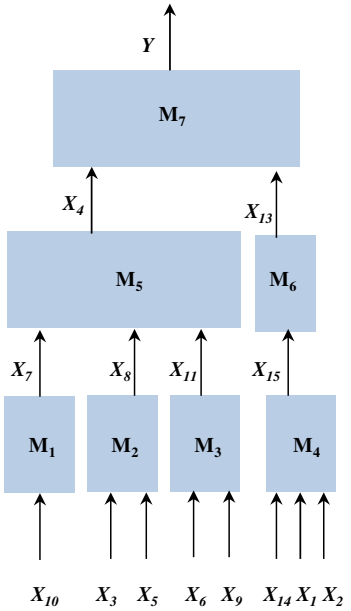
Remitiéndonos a la codificación de las soluciones del espacio de búsqueda, en un sistema difuso jerárquico en paralelo, cada módulo (o nodo) está formado por un vector de tipo entero que contiene las variables de entrada al módulo, un campo de tipo entero que almacena la variable de salida y un vector de referencias a los módulos de los que depende. El flujo de datos a través de la jerarquía transita desde los nodos hoja hacia el nodo raíz (módulo de salida del sistema jerárquico), por lo que será el orden natural para referirnos a las capas del sistema difuso jerárquico en paralelo, es decir, la capa con salida Y es la última capa y la capa con variables exógenas es la primera capa. Un aspecto a destacar es que dado que los módulos de la primera capa no dependen de ningún otro, el vector de referencia a los módulos de los que depende, situado en un nodo del árbol, está inicializado a un valor nulo, de ahí que en la figura las posiciones correspondientes a este vector aparezcan señalados con un 'aspa'. Finalmente, cada árbol constituye un individuo de la población.

Como se puede apreciar, la estructura de datos que codifica el problema se caracteriza por su simplicidad debido a que el número de variables y nodos no es extremadamente complejo. Por ello, nos decantamos por esta representación y no por su codificación mediante programación genética. La programación genética se suele usar para problemas de semántica más compleja, en donde se implementan unos operadores específicos para árboles.

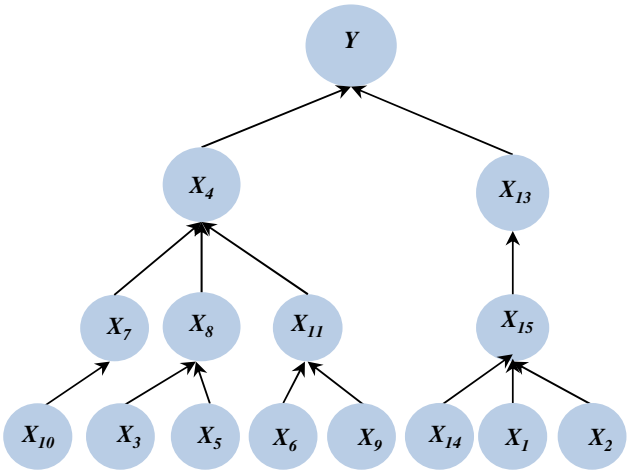
4.1.1.2. Inicialización

En el algoritmo, la inicialización de la población se realiza de forma aleatoria. Esto significa que partiremos de una población de estructuras jerárquicas paralelas en donde el número de capas, el número de variables distribuidas en la jerarquía y el número de módulos por capa va a ser totalmente aleatorio.

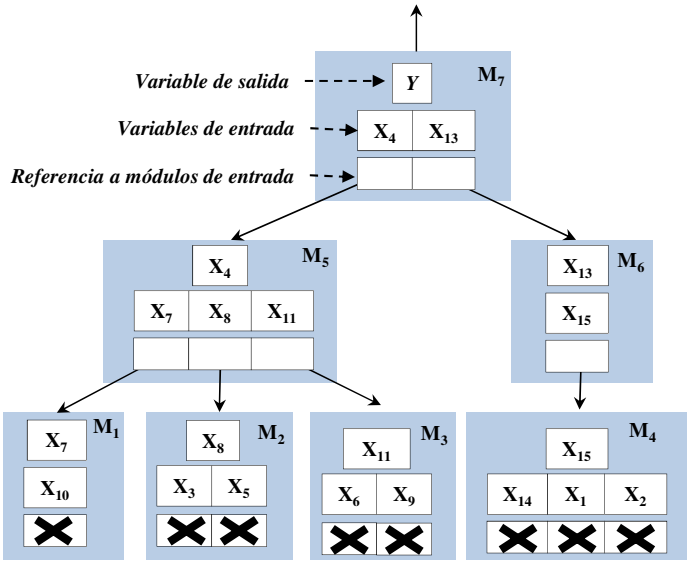
Destacar que estas estructuras jerárquicas en paralelo, al igual que las estructuras jerárquicas en serie, se encuentran limitadas en la creación del número de capas: establecemos un máximo de tres capas, dado que, como venimos comentando, existe el problema del acarreo del error generado por los módulos a través de la estructura jerárquica, pero además queremos limitar el espacio de búsqueda para no tener en cuenta soluciones claramente no prometedoras. Si el individuo generado está formado por más de tres capas, se descarta y se genera uno nuevo. También se considera que ninguno de los módulos de las capas intermedias de la jerarquía puede hacer uso



(a) Sistema difuso jerárquico paralelo



(b) Estructura de datos en árbol



(c) Esquema de codificación

Figura 4.1.: Correspondencia entre un sistema difuso jerárquico en paralelo, una estructura de datos en árbol y su esquema de codificación

de variables exógenas.

Con respecto al número de módulos no existe ningún tipo de limitación y las variables contenidas como elementos de la estructura jerárquica, tanto endógenas como exógenas, son variables pertenecientes al problema abordado.

4.1.1.3. Operador de Cruce

El operador de cruce, al igual que en la estructura jerárquica en serie, se aplica según una probabilidad entre pares de cromosomas padres, atendiendo al número de capas que tengan los padres seleccionados. Cuando el operador de cruce se aplica a dos padres, P_1 y P_2, se pueden distinguir diferentes casos:

1. Ambos padres, P_1 y P_2, tienen varias capas.
2. El padre P_1 tiene varias capas y el padre P_2 tiene una sola capa (o viceversa).
3. Ambos padres, P_1 y P_2, tienen una sola capa (o lo que es lo mismo en este caso, ambos tienen un sólo módulo).

El operador de cruce implementa un enfoque centrado en el padre: los descendientes principalmente heredan la información de uno de los padres y el resto se hereda del padre secundario para añadir diversidad. El operador de cruce controla el número máximo de capas con el objetivo de reducir el espacio de búsqueda, creando una nueva solución si el cruce entre dos padres genera un número de capas menor o igual a tres. En otro caso, se descarta y se genera otro individuo. Por otro lado, también tiene en cuenta la restricción de que ningún módulo de capas intermedias puede contener variables exógenas. La figura 4.2 muestra un árbol de decisión con todos los casos que considera el operador de cruce.

4.1.1.3.1. Ambos padres, P_1 y P_2, tienen varias capas

Dados dos padres, P_1 y P_2, y ambos tienen varias capas, el proceso de generación del descendiente, D_1, es como sigue:

1. *Generación de la capa de salida:* El descendiente D_1 se genera centrado en P_1, es decir, D_1 hereda de P_1 el módulo de salida.

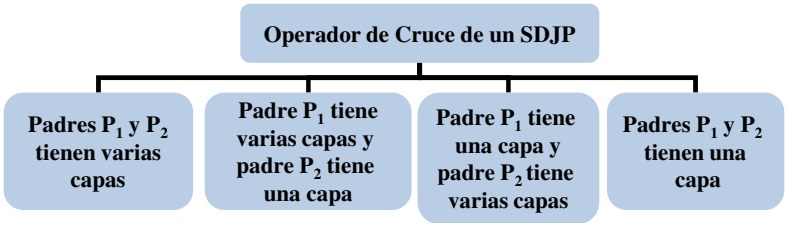


Figura 4.2.: Árbol de decisión general del operador de cruce de un SDJP

2. *Generación de la capa intermedia:* Para cada variable exógena v de D_1, buscar un módulo con salida v en P_2, insertarlo en D_1 y borrar las variables exógenas repetidas últimas insertadas de D_1. Si v no existe en P_2, buscar en P_1 un módulo con salida v. A continuación, borrar las variables exógenas repetidas últimas insertadas en D_1. Si al terminar el proceso de generación de la segunda capa existen módulos sin variables de entrada, para cada variable exógena e del módulo con salida v en D_1, buscar sus dependencias en la capa previa en P_2. Seguidamente, seleccionar una variable aleatoria y añadirla como entrada al módulo de salida v. Si no se encuentran dependencias, realizar la búsqueda en P_1.

3. *Testeo de la capa intermedia:* Si existe algún módulo en la capa intermedia que no tenga variables exógenas:

   a) Seleccionar una variable no usada en P_2 e insertarla en el módulo sin variables exógenas como variable exógena.

   b) Si no existen variables usadas en P_2, seleccionar una variable no usada en P_1 e insertarla como exógena en el módulo sin variables de entrada.

   c) Si no existen variables no usadas en P_1, seleccionar una variable no usada en D_1 e insertarla como variable exógena en el módulo sin entradas.

   d) Si no existen variables no usadas en D_1, realizar una reagrupación de variables en el módulo de salida de la siguiente forma:

      1) Extraer las variables exógenas que se quedaron sin encontrar su endógena en los padres y emparejarlas aleatoriamente entre ellas mediante un módulo. A continuación, insertar los módulos obtenidos en el módulo de salida de la jerarquía.

2) Si alguna variable queda sin emparejar, debido a que el número de variables exógenas que no encontraron su endógena es impar, extraer esa variable del sistema e insertarla aleatoriamente en un módulo de la nueva capa creada. La figura 4.3 muestra un ejemplo de este caso particular. La figura consiste en un sistema jerárquico paralelo aplicado a un problema de ocho variables en donde se puede apreciar la transformación modular realizada a las variables exógenas del sistema. Cabe destacar que después de realizar el emparejamiento de dos variables, la variable restante en el nuevo módulo creado podría haber sido insertada en el módulo M_1, en lugar del módulo M_2, pero, en este caso, el algoritmo eligió aleatoriamente el módulo M_2.

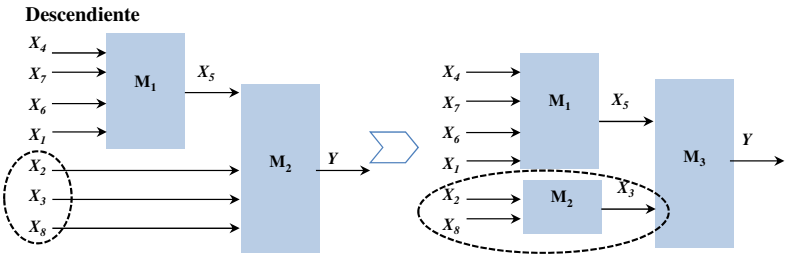


Figura 4.3.: Caso donde ambos padres tienen varias capas. Testeo de la capa intermedia del descendiente en el operador de cruce: Emparejamiento de variables exógenas

4. *Creación de la primera capa:* Para la construcción de la primera capa, se decide aleatoriamente si crear esta nueva capa o no. En caso positivo, repetir el procedimiento de generación de la capa intermedia para su creación.

La creación del descendiente D_2 seguiría este mismo procedimiento, pero estaría centrado en P_2. Por otro lado, si los padres P_1 y P_2 no presentan en su estructura variables comunes, el descendiente D_1 será una copia de P_1 y el descendiente D_2 será una copia de P_2.

4.1.1.3.2. El padre P_1 tiene varias capas y el padre P_2 tiene una sola capa (o viceversa)

Dados dos padres, P_1, con varias capas, y P_2, con una sola capa, el proceso de generación de los descendientes, D_1 y D_2, es el siguiente:

1. El descendiente D_1 es una copia de P_1.
2. El descendiente D_2 se genera centrado en P_2 y atendiendo al tipo de variables que tiene en común con P_1:

   a) Si P_2 tiene variables exógenas comunes con variables endógenas de P_1, D_2 se genera de forma similar a cuando P_1 y P_2 tienen varias capas.

   b) En otro caso, se escogen n variables endógenas aleatoriamente de la primera capa de P_1 y se añaden en D_2 como variables exógenas, donde $n \in [1, \#varEndogenas]$ y siendo $\#varEndogenas$ el número total de variables endógenas de la primera capa de P_1. La figura 4.4 muestra un ejemplo de este caso.

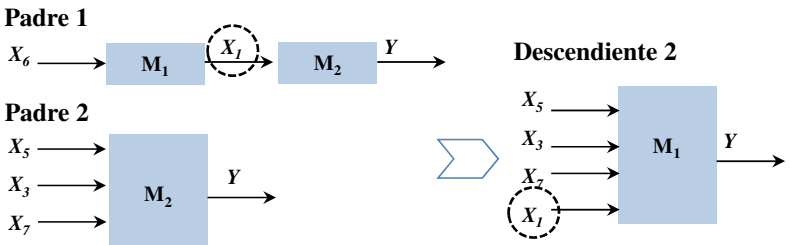


Figura 4.4.: Caso padre P_1 tiene varias capas y el padre P_2 tiene una sola capa. Generación de D_2

4.1.1.3.3. Ambos padres, P_1 y P_2, tienen una sola capa

El cruce de dos padres, P_1 y P_2, con una sola capa para generar dos descendientes, D_1 y D_2, sigue el mismo procedimiento que el operador de cruce de los sistemas difusos jerárquicos en serie en este mismo caso (ver subsección 3.1.3.4).

4.1.1.4. Operador de Mutación

En este tipo de jerarquía, se puede realizar una mutación a tres elementos distintos: capas, módulos y variables. El operador de mutación del algoritmo decide

aleatoriamente qué tipo de mutación realizar sobre la estructura jerárquica paralela:

- Insertar, eliminar, intercambiar o mover variables.
- Añadir o eliminar módulos.
- Añadir o eliminar capas.

La tabla 4.1 muestra un esquema resumen de la funcionalidad de este operador, el algoritmo 7 describe su pseudocódigo y la figura 4.5 muestra un árbol de decisión con todos los casos que considera este operador.

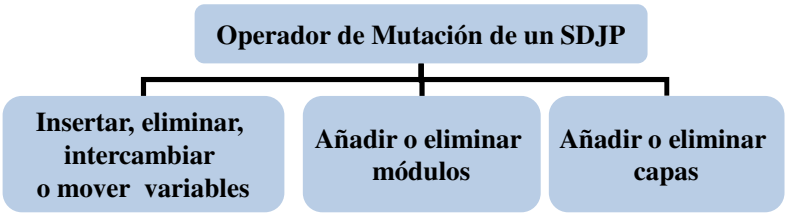


Figura 4.5.: Árbol de decisión general del operador de mutación de un SDJP

Tabla 4.1.: Funcionalidad del operador de mutación de un SDJP

Funcionalidad / Elemento	Añadir	Eliminar	Intercambiar	Mover
Variables	X	X	X	X
Módulos	X	X	-	-
Capas	X	X	-	-

4.1.1.4.1. Mutación de variables

Este tipo de operador de mutación maneja las variables, tanto endógenas como exógenas, eligiendo aleatoriamente una de las siguientes operaciones:

- *Inserción de una variable no usada.* Para insertar una variable en una jerarquía paralela, el algoritmo selecciona una variable no usada y un módulo de la primera capa de forma aleatoria. A continuación, inserta la variable no usada

Algoritmo 7 Operador de mutación de un SDJP

Entrada: Un SDJP de la población: *indiv*
1: r = U[0,1] {Probabilidad uniforme}
2: **Si** (($r < 0,33$) y (*número_de_capas(indiv)* > 1) y (*no tiene una sola rama de una entrada y una salida*)) **entonces**
3: Mutación_de_módulos(*indiv*)
4: **Sino**
5: **Si** (($r >= 0,33$) y ($r <= 0,66$) y (*número_de_capas(indiv)*> 1)) **entonces**
6: Mutación_de_capas(*indiv*)
7: **Sino**
8: Mutación_de_variables(*indiv*)
9: **Fin Si**
10: **Fin Si**

seleccionada como variable exógena en el módulo seleccionado (figura 4.6(a) y algoritmo 8).

Algoritmo 8 Inserción de una variable no usada en un SDJP

Entrada: Un SDJP: *indiv*
1: v = Elegir_una_variable_no_usada_aleatoriamente(*indiv*)
2: m = Elegir_un_módulo_de_la_última_capa_aleatoriamente(*indiv*)
3: Insertar_variable_exógena(v, m, *indiv*)

- *Eliminación de variables exógenas.* Este operador selecciona un módulo aleatoriamente de la primera capa, donde todas sus variables son exógenas. Seguidamente, se selecciona en ese módulo una variable aleatoriamente y se elimina. Si el módulo seleccionado solo tiene esa variable exógena, el módulo desaparece junto con su variable de salida (figura 4.6(b) y algoritmo 9). Este operador sólo se puede aplicar si la estructura jerárquica está formada por más de una capa.

Algoritmo 9 Eliminación de una variable exógena en un SDJP

Entrada: Un SDJP: *indiv*
1: **Si** (*(numero_de_capas*(*indiv*) > 1) **entonces**
2: m = Elegir_un_módulo_de_la_primera_capa_aleatoriamente(*indiv*)
3: v = Elegir_una_variable_aleatoriamente(m, *indiv*)
4: Eliminar_variable_exógena(v, m, *indiv*)
5: **Fin Si**

- *Intercambio de variables.* En este caso (figura 4.6(c) y algoritmo 10), el operador selecciona un módulo aleatoriamente, excluyendo el módulo de salida del

sistema jerárquico. Posteriormente, selecciona una de las variables de entrada al módulo de forma aleatoria y la intercambia con la variable endógena del módulo.

Algoritmo 10 Intercambio de variables en un SDJP

Entrada: Un SDJP: *indiv*
1: m = Elegir_un_módulo(*indiv*)
2: v = Elegir_una_variable_de_entrada_aleatoriamente(m, *indiv*)
3: Intercambiar_variable(v, m, *indiv*)

- *Mover una variable exógena.* Para ello, se selecciona un módulo de forma aleatoria que tenga variables exógenas. Del módulo se extrae una variable y se selecciona otro módulo aleatoriamente de la primera capa en donde, finalmente, se inserta la variable extraída (figura 4.6(d) y algoritmo 11). Este operador se aplica cuando la estructura jerárquica tiene más de una capa y cuando la capa que tiene las variables exógenas tiene más de un módulo.

Algoritmo 11 Mover una variable exógena en un SDJP

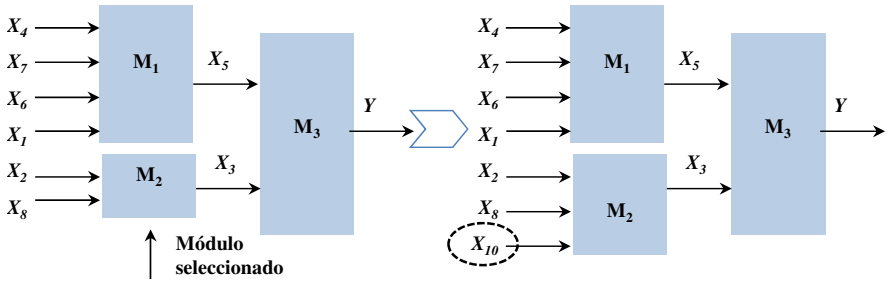
Entrada: Un SDJP: *indiv*
1: m_1 = Elegir_un_módulo_con_variables_exógenas_aleatoriamente(*indiv*)
2: m_2 = Elegir_un_módulo_con_variables_exógenas_aleatoriamente(*indiv*)
3: **Si** (m_1 *no es un módulo de una entrada y una salida*) **entonces**
4: e = Elegir_variable_exógena_aleatoriamente(m_1, *indiv*)
5: Borrar_variable(e, m_1, *indiv*)
6: Insertar_variable(e, m_2, *indiv*)
7: **Fin Si**

La figura 4.6 ilustra ejemplos de los cuatro casos de este operador de mutación.

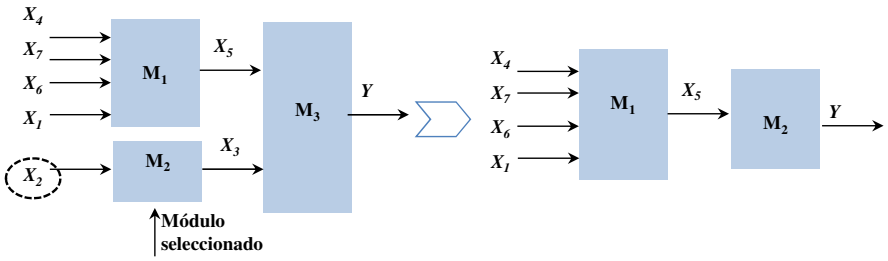
4.1.1.4.2. Mutación de módulos

La mutación de módulos implementa dos operaciones en donde la elección de la acción se toma de forma aleatoria (figura 4.7):

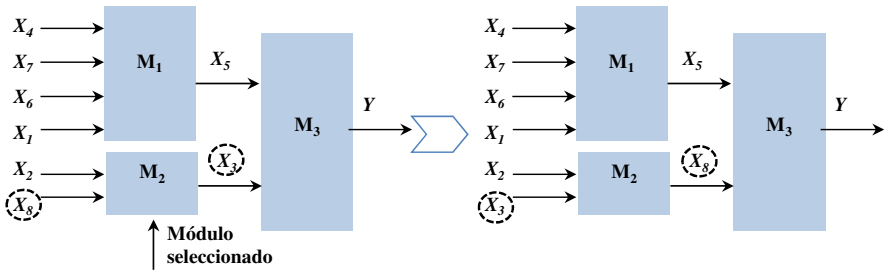
- *Añadir módulos.* Esta opción contempla la inserción de un nuevo módulo en la capa intermedia o en la capa que contiene variables exógenas. La elección de una opción u otra se realiza de forma aleatoria.



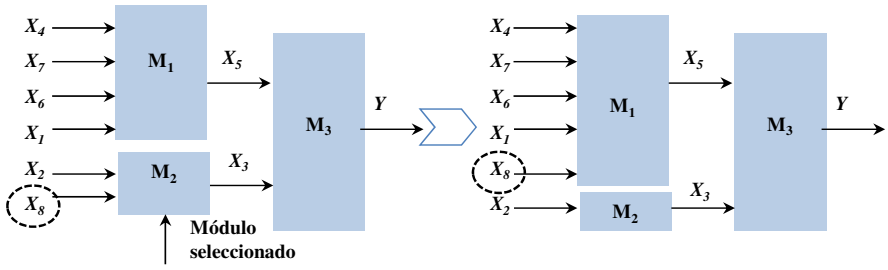
(a) Inserción de variables



(b) Eliminación de variables



(c) Intercambio de variables



(d) Mover una variable

Figura 4.6.: Operador de mutación de variables en un SDJP

- *Añadir módulos en la capa que contiene variables exógenas.* Para ello, se selecciona aleatoriamente un módulo m_1 que contenga más de dos variables exógenas (condición indispensable para poder aplicar el operador) y se divide en dos módulos, de forma que el módulo del que se parte, m_1, mantiene su variable endógena y conserva como mínimo una variable exógena. El nuevo módulo, m_2, se crea seleccionando n variables exógenas de m_1, con $n \in [2, totalVarExogenas(m_1) - 1]$ y donde la variable de salida de m_2 se escoge aleatoriamente entre las n variables. La figura 4.7(a) muestra un ejemplo de este caso. El sistema difuso jerárquico paralelo está formado por tres capas, donde la primera capa es la que contiene las variables exógenas. De los módulos contenidos en dicha capa, se selecciona M_1 y se procede a la mutación. Finalmente, con $n = 3$, se escogieron aleatoriamente las variables X_1, X_6 y X_7 de M_1, donde se escogió aleatoriamente a X_6 como variable de salida del nuevo X_1. El anterior M_1 queda como M_2 en la capa.

- *Añadir módulos a la capa intermedia.* Este operador promociona un módulo de la primera capa a la capa intermedia para añadir un módulo. Por ello, se selecciona aleatoriamente un módulo m_1 de la primera capa que cumpla la siguiente restricción: el módulo a promocionar ha de tener como entrada como mínimo dos variables exógenas. Una vez seleccionado el modulo aleatoriamente, se escoge una variable exógena v de m_1, se creará un nuevo módulo m_2 en donde las variables exógenas serán las variables exógenas de m_1 y la variable endógena sera v. Finalmente, v será la variable de entrada de m_1. Si todos los módulos en la estructura jerárquica paralela de la primera capa tienen una variable exógena y una endógena, este operador no se aplica. La figura 4.7(b) ilustra la promoción del módulo M_1 hacia la capa intermedia.

- *Eliminar módulos.* Se escogen los módulos que cuelgan del mismo módulo para realizar una fusión de los mismos. Se selecciona de forma aleatoria uno de los módulos de los que cuelgan otros módulos. *Si los módulos que cuelgan pertenecen a la capa intermedia*, de los módulos que cuelgan, se selecciona aleatoriamente un módulo origen m_{orig} que se extrae para la fusión y un módulo destino aleatorio m_{dest} con el que se realiza la fusión. La variable endógena del módulo m_{orig} se convierte en una variable endógena de entrada del módu-

lo m_{dest} y las variables tanto endógenas como exógenas del módulo m_{orig} se transforman en variables exógenas del nuevo módulo creado. La figura 4.7(c) es un ejemplo de este tipo de eliminación de módulos.

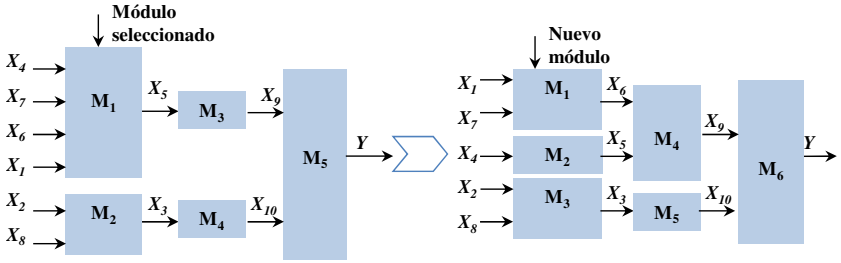
En otro caso, si se escoge la opción en donde los *módulos con variables exógenas cuelgan del mismo módulo*, se eligen dos módulos aleatoriamente, origen y destino, donde todas las variables del módulo origen se convierten en exógenas del módulo destino. La figura 4.7(d) ilustra este tipo de eliminación de módulos de la estructura jerárquica paralela.

4.1.1.4.3. Mutación de capas

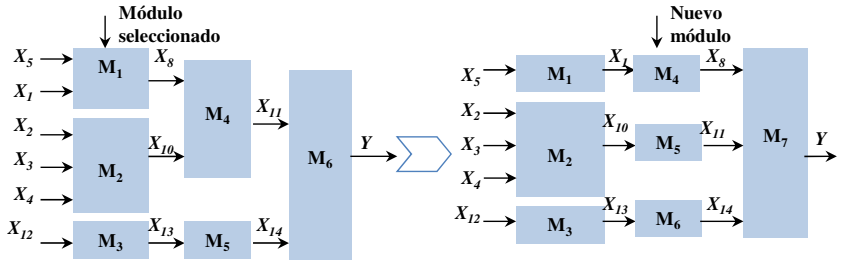
La mutación de capas implica dos operaciones sobre una estructura jerárquica paralela y la elección de cada una de ellas se decide aleatoriamente (figura 4.8):

- *Añadir capas.* Para añadir una nueva capa a la jerarquía, por cada módulo de la actual primera capa, m_i^{act} donde i es el i-ésimo módulo y act es la primera capa actual, se analiza el número de variables exógenas. Si hay dos o más variables exógenas por cada módulo m_i^{act}, se selecciona aleatoriamente una variable exógena por módulo $v_{i,m}^{act}$ que será la variable de salida del nuevo módulo que va a formar la nueva capa m_i^{nueva}, donde *nueva* es la nueva capa que se crea. El resto de variables exógenas se convierten en las variables exógenas de este nuevo módulo y su variable endógena será la entrada de m_i^{act}. Si durante el análisis de las variables exógenas para la creación de una nueva capa, se detecta un módulo con una sola variable exógena, esa variable exógena será la variable de salida del nuevo módulo que se crea en la nueva capa y la variable de entrada al mismo será una variable no usada en el sistema jerárquico paralelo. La mutación implementa un mecanismo para controlar el número de capas a tres. La figura 4.8(a) muestra la creación de una primera nueva capa en un sistema difuso jerárquico paralelo con dos capas.

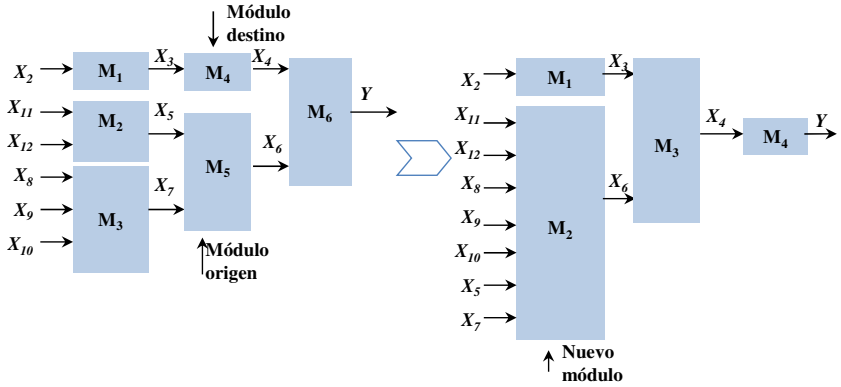
- *Eliminar capas.* En este caso, el operador de mutación siempre elimina la primera capa del sistema difuso jerárquico paralelo. La figura 4.8(b) ilustra el borrado de la última capa.



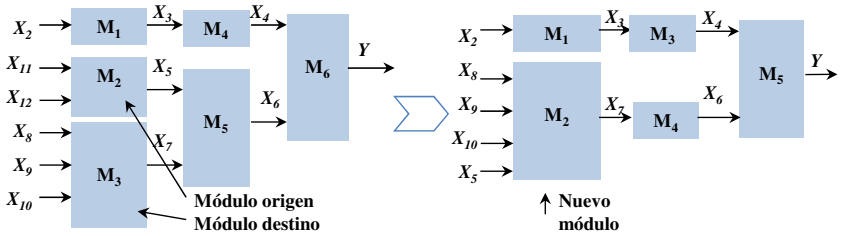
(a) Añadir módulos en la capa que contiene variables exógenas



(b) Añadir módulos a la capa intermedia

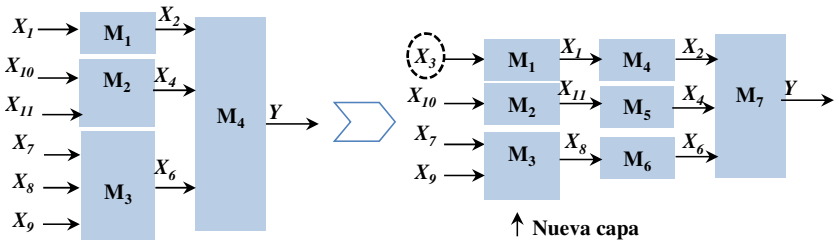


(c) Eliminar módulos de la capa intermedia

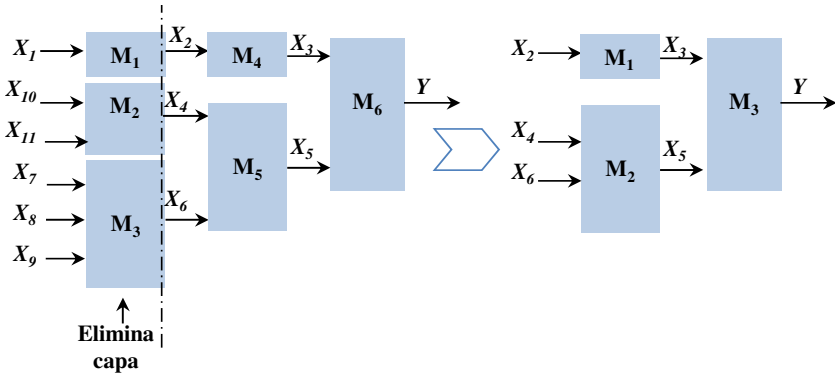


(d) Eliminar módulos de la capa que contiene variables exógenas

Figura 4.7.: Operador de mutación de módulos en un SDJP



(a) Añadir capa



(b) Eliminar capa

Figura 4.8.: Operador de mutación de capas de un SDJP

4.1.1.5. Mecanismo de Inferencia

Al igual que en el algoritmo SDJSG, el mecanismo de inferencia que implementa este algoritmo es el Max–Min, la T-norma del mínimo como conjunción y el centro de gravedad como defuzzificación.

La inferencia de las estructuras jerárquicas paralelas se realiza de izquierda a derecha, es decir, desde los módulos que contienen variables exógenas hasta el módulo de salida. La inferencia se realiza capa a capa ya que las variables endógenas de la capa intermedia necesitan la inferencia de los módulos situados en la capa anterior para obtener su valor. Como se viene comentando, cuando el módulo infiere una salida, se produce un error que se va acarreando a través de las capas, por lo que no es conveniente que el número de módulos de la estructura jerárquica sea elevado.

4.1.1.6. Aprendizaje de la Base de Reglas

Para obtener el conocimiento, el aprendizaje de la base de reglas se realiza módulo a módulo, atendiendo a sus correspondientes variables de entrada y salida, y cuyo sentido parte de los módulos situados en la primera capa hacia el módulo de salida. A nivel local, cada módulo aprende un conjunto de reglas de tipo Mamdani mediante el método de WM. Todas las variables existentes en la estructura jerárquica, tanto endógenas como exógenas, son variables naturales, por lo que no se generan nuevas variables. Las variables exógenas se extraen del conjunto de datos y las variables endógenas se infieren según las variables de entrada al módulo. La figura 4.9 muestra un ejemplo gráfico para un SDJ Paralelo. Este sistema difuso jerárquico paralelo está formado por cuatro módulos, de los cuales sólo los módulos de la última capa (módulos M_1 y M_2) reciben como entrada variables exógenas e infieren variables naturales endógenas (X_2 y X_7). El valor de cada variable endógena inferida es la entrada a los módulos de la siguiente capa (la capa intermedia) y así sucesivamente. Como se puede apreciar en la figura, la capa intermedia y la capa de salida no manejan variables exógenas, sino que sólo se utilizan en la estructura en la primera capa del sistema jerárquico. El sistema obtenido se evalúa mediante ejemplos de datos de entrenamiento y eligiendo los valores correspondientes de variables exógenas en ese módulo.

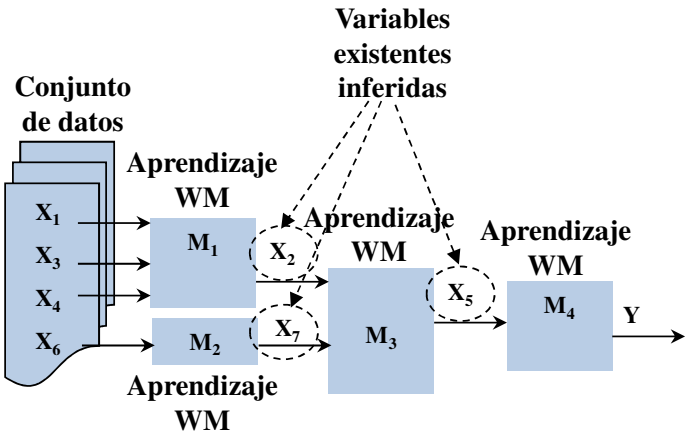


Figura 4.9.: Ejemplo del proceso de inferencia en un SDJ Paralelo

4.1.1.7. Enfoque Multiobjetivo

El algoritmo de aprendizaje de estructuras jerárquicas paralelas hace uso del algoritmo NSGA-II, al igual que el algoritmo SDJSG, que minimiza igualmente las dos funciones objetivo descritas en el capítulo 3(ver subsección 5.5.2).

4.1.2. Experimentos

En este algoritmo, al igual que el algoritmo SDJSG, considera los 14 problemas de regresión detallados en la subsección 3.2.1 y usa la misma configuración experimental a nivel paramétrico (subsección 3.2.3). Por otro lado, utilizaremos los mismos métodos de comparación que en el capítulo anterior, considerando además la propuesta del algoritmo en serie.

4.1.3. Estudio Experimental

4.1.3.1. Resultados obtenidos

Las tablas 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 y 4.8 recogen los resultados obtenidos por la propuesta en paralelo a la que se ha llamado Sistema Difuso Jerárquico en Paralelo Genético (SDJPG), donde ECM_{entr} y ECM_{prue} son los valores del error de aproximación (ecuación 3.1) en entrenamiento y prueba respectivamente para un conjunto de datos; *#M*, *#R* y *#V* hacen referencia al número de módulos, el número de reglas difusas y el número de variables de entrada respectivamente; $\sigma_{\#M}$ es la desviación típica del número de módulos y $\bar{x}_{\#R}$ es el número de variables por regla.

En las tablas se muestran cinco soluciones representativas de la media del Pareto ordenadas de menor a mayor ECM_{entr}: la primera fila de cada problema es la solución más precisa, la segunda fila es el primer cuartil, la tercera fila es la mediana, la cuarta fila es el tercer cuartil y la quinta fila es la solución menos precisa.

4.1.3.2. Análisis

Como venimos comentando, nuestro objetivo principal es el de mejorar la interpretabilidad del sistema difuso conservando o mejorando la precisión del mismo. El modelizado de problemas mediante un sistema difuso jerárquico paralelo presenta un comportamiento diferente al modelizado en serie.

Tabla 4.2.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
LASER							ELE2						
WM	265,21	278,58	1,0±0,0	58,4	4,0	4,0	WM	112.271	112.719	1,0±0,0	65,0	4,0	4,0
JS	299,46	323,24	6,0±0,0	55,6	2,1	4,0	JS	117.172	127.343	6,0±0,0	36,6	1,3	3,0
SVAG	265,21	278,58	1,0±0,0	58,4	4,0	4,0	SVAG	90.956	100.100	1,0±0,0	36,6	3,0	3,0
SDJSG s/SV	263,74	293,06	1,2±0,4	55,0	3,7	4,0	SDJSG s/SV	98.205	102.994	2,0±0,0	41,6	2,8	4,0
SDJSG4M s/SV	263,74	293,06	1,2±0,4	55,0	3,7	4,0	SDJSG4M s/SV	98.205	102.994	2,0±0,0	41,6	2,8	4,0
	263,74	293,06	1,2±0,4	55,0	3,8	4,0		72.231	75.695	1,4±0,5	15,1	1,9	2,4
	309,01	327,13	1,3±0,3	38,9	2,9	3,3		183.426	185.363	1,3±0,2	12,4	1,7	2,1
SDJSG	342,74	364,76	1,1±0,2	25,3	2,3	2,5	SDJSG	349.446	347.696	1,6±0,4	10,2	1,3	1,9
	694,76	728,33	1,1±0,1	15,0	1,7	1,8		487.700	473.206	1,5±0,3	7,7	1,1	1,5
	1.469,79	1.513,25	1,0±0,0	5,0	1,0	1,0		591.403	565.462	1,0±0,0	5,0	1,0	1,0
	265,21	278,58	1,0±0,0	58,4	4,0	4,0		90.956	100.100	1,0±0,0	36,6	3,0	3,0
	332,67	347,05	1,0±0,0	32,4	2,8	2,8		101.363	98.542	1,0±0,0	23,4	2,3	2,3
SDJPG	602,40	621,70	1,0±0,0	16,7	2,0	2,0	SDJPG	311.068	327.170	1,0±0,0	15,0	2,0	2,0
	1.278,19	1.335,25	1,0±0,0	11,5	1,9	1,9		492.073	501.914	1,0±0,0	10,3	1,9	1,9
	1.469,79	1.513,25	1,0±0,0	5,0	1,0	1,0		631.654	631.655	1,0±0,0	5,0	1,0	1,0

Tabla 4.3.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
DEE							CONCRETE						
WM	0,14117	0,22888	1,0±0,0	178,4	6,0	6,0	WM	73,080	97,006	1,0±0,0	310,4	8,0	8,0
JS	0,23050	0,33550	9,8±4,7	179,6	3,0	6,0	JS	95,540	113,840	13,6±2,3	300,2	3,5	8,0
SVAG	0,13900	0,22390	1,0±0,0	155,2	5,4	5,4	SVAG	72,477	88,984	1,0±0,0	270,0	7,0	7,0
SDJSG s/SV	0,13961	0,22728	1,4±0,5	162,5	5,6	6,0	SDJSG s/SV	71,725	88,160	1,8±1,0	263,4	6,8	8,0
SDJSG4M s/SV	0,13948	0,22713	1,4±0,5	161,3	5,5	6,0	SDJSG4M s/SV	70,355	88,666	1,7±0,9	267,7	6,8	8,0
	0,13728	0,23338	1,1±0,2	155,1	5,6	5,6		50,859	73,521	1,1±0,2	312,5	7,0	7,1
	0,15308	0,22436	1,2±0,3	87,5	4,0	4,2		55,786	71,124	1,2±0,5	231,5	5,4	5,8
SDJSG	0,18253	0,21145	1,5±0,5	38,8	2,7	3,5	SDJSG	70,310	79,036	1,2±0,4	133,7	4,1	4,4
	0,23023	0,23913	1,5±0,6	14,7	1,9	2,4		112,184	117,851	1,2±0,4	48,1	2,6	2,9
	0,34664	0,35492	1,0±0,0	4,5	1,0	1,0		240,195	248,009	1,0±0,0	3,9	1,0	1,0
	0,13900	0,22390	1,0±0,0	155,2	5,4	5,4		72,510	88,810	1,0± 0,0	267,4	6,9	6,9
	0,15769	0,22729	1,0±0,0	105,7	4,4	4,4		85,994	97,443	1,0± 0,0	141,0	4,7	4,7
SDJPG	0,18913	0,22881	1,0±0,0	60,9	3,4	3,4	SDJPG	104,736	111,313	1,2± 0,6	70,6	3,6	3,8
	0,23843	0,26082	1,1±0,4	28,7	2,5	2,6		174,188	176,147	1,1± 0,5	29,5	2,5	2,7
	0,39815	0,41665	1,0±0,0	5,0	1,0	1,0		257,855	251,284	1,0± 0,0	5,0	1,0	1,0

Tabla 4.4.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los errores de aproximación de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los ECM_{entr} y ECM_{pru} se deben multiplicar por 10^5

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
CENSUS 8L							CENSUS 8H						
WM	15.234	16.122	1,0±0,0	762,8	8,0	8,0	WM	23.350	24.504	1,0±0,0	713,4	8,0	8,0
JS	14.503	15.188	6,3±0,7	86,7	2,9	4,0	JS	22.038	23.708	11,2±7,0	291,7	3,1	5,0
SVAG	13.468	14.023	1,0±0,0	140,1	5,0	5,0	SVAG	20.903	21.907	1,0±0,0	397,4	5,4	5,4
SDJSG s/SV	12.750	13.090	3,3±0,6	145,0	4,2	8,0	SDJSG s/SV	20.734	21.846	2,7±0,6	419,5	5,6	8,0
SDJSG4M s/SV	12.546	12.974	3,1±0,6	166,4	4,6	8,0	SDJSG4M s/SV	20.648	21.717	2,4±0,6	444,1	6,0	8,0
	12.739	13.211	1,6±0,6	169,4	5,6	6,7		20.284	21.238	1,4±0,6	437,5	6,4	6,8
	13.431	13.876	1,6±0,8	98,3	4,3	5,2		21.195	22.072	1,3±0,4	273,7	4,7	5,2
SDJSG	16.045	16.516	1,6±0,8	39,9	2,9	3,8	SDJSG	22.714	23.242	1,6±0,8	119,3	3,3	4,4
	20.584	20.950	1,8±0,8	19,8	1,8	2,8		25.140	25.270	1,5±0,7	28,6	2,3	2,8
	30.996	31.408	1,0±0,0	5,0	1,0	1,0		30.391	30.345	1,0±0,0	5,0	1,0	1,0
	13.517	14.067	1,0±0,0	129,0	4,9	4,9		20.903	21.907	1,0±0,0	397,4	5,4	5,4
	14.899	15.366	1,0±0,0	44,3	3,8	3,8		22.911	23.646	1,0±0,0	180,9	4,4	4,4
SDJPG	23.149	23.509	1,0±0,0	24,3	2,9	2,9	SDJPG	24.791	24.992	1,1±0,3	76,9	3,7	3,8
	28.739	29.228	1,0±0,0	12,8	2,1	2,1		31.525	31.526	1,0±0,0	14,8	2,1	2,1
	52.840	52.850	1,0±0,0	3,6	1,0	1,0		52.837	52.848	1,0±0,0	3,8	1,0	1,0

Tabla 4.5.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los errores de aproximación de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
		WEANK							MORTGAGE				
WM	9,75605	12,25650	1,0±0,0	456,8	9,0	9,0	WM	0,25600	0,26806	1,0±0,0	197,2	15,0	15,0
JS	13,13610	16,56026	6,9±2,4	100,8	3,0	5,0	JS	0,90399	0,90393	7,1±2,4	80,6	2,7	5,0
SVAG	7,58657	8,84168	1,0±0,0	140,2	5,4	5,4	SVAG	0,11871	0,12598	1,0±0,0	61,5	5,4	5,4
SDJSG s/SV	7,10852	7,35495	3,3±0,6	148,9	4,8	9,0	SDJSG s/SV	0,11376	0,12191	8,1±1,2	101,7	2,9	15,0
SDJSG4M s/SV	7,10044	7,45815	3,0±0,4	170,0	4,2	9,0	SDJSG4M s/SV	0,10971	0,11165	4,0±0,0	131,9	4,2	15,0
	6,63906	6,84172	1,9±0,8	64,8	3,8	5,7		0,10097	0,10848	2,1±1,0	40,7	4,6	7,2
	7,46367	7,49844	2,0±0,8	40,1	3,0	4,9		0,10856	0,11200	2,0±0,9	33,6	3,7	5,5
SDJSG	8,63689	8,62772	1,8±0,6	25,7	2,4	3,7	SDJSG	0,13511	0,13602	1,8±0,9	27,6	2,9	4,3
	10,75790	11,00243	1,5±0,5	13,8	1,9	2,6		0,19417	0,20913	1,5±0,6	16,5	2,0	2,7
	13,66512	13,62635	1,0±0,0	2,1	1,0	1,0		0,32981	0,34130	1,0±0,0	5,0	1,0	1,0
	7,58927	8,85345	1,0±0,0	137,4	5,3	5,3		0,11956	0,12515	1,0±0,0	61,1	5,3	5,3
	8,95733	9,25060	1,0±0,0	83,3	4,2	4,2		0,13853	0,14424	1,0±0,0	45,6	4,6	4,6
SDJPG	10,93894	11,27933	1,0±0,0	28,3	3,0	3,0	SDJPG	0,16755	0,17893	1,0±0,0	27,1	3,2	3,2
	25,91153	26,12875	1,0±0,0	12,0	2,0	2,0		0,25081	0,25673	1,0±0,0	15,9	2,3	2,3
	1298,61069	1362,51850	1,0±0,0	3,6	1,0	1,0		0,34735	0,37721	1,0±0,0	5,0	1,0	1,0

Tabla 4.6.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 16L*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
	TREASURY							CENSUS 16L					
WM	0,80168	0,80909	1,0±0,0	195,2	15,0	15,0	WM	32.505	32.887	1,0±0,0	682,0	16,0	16,0
JS	0,19888	0,20635	2,8±1,6	18,8	1,8	2,0	JS	23.815	24.108	6,4±2,3	62,6	3,1	7,0
SVAG	0,16362	0,17451	1,0±0,0	16,8	2,2	2,2	SVAG	21.507	21.804	1,0±0,0	79,1	6,4	6,4
SDJSG s/SV	0,11325	0,11924	10,3±1,1	80,0	1,9	15,0	SDJSG s/SV	19.772	19.972	8,3±1,7	62,0	3,6	16,0
SDJSG4M s/SV	0,13011	0,13808	4,0±0,0	130,2	4,4	15,0	SDJSG4M s/SV	19.797	20.069	3,9±0,1	114,7	5,1	16,0
	0,14692	0,15025	1,5±0,8	17,5	3,0	4,1		17.884	18.163	2,6±1,0	75,1	5,7	10,5
	0,16304	0,17243	1,5±0,7	16,4	2,5	3,4		18.529	18.823	2,5±1,0	50,2	4,5	8,2
SDJSG	0,18550	0,19955	1,6±0,9	14,8	2,0	2,8	SDJSG	20.052	20.273	2,6±0,9	30,9	3,3	6,6
	0,23274	0,24625	1,3±0,5	10,7	1,6	2,0		23.701	23.878	2,2±1,0	16,1	2,4	4,3
	0,31949	0,33830	1,0±0,0	5,0	1,0	1,0		41.698	41.841	1,0±0,0	3,0	1,0	1,0
	0,16295	0,17131	1,1±0,5	17,8	2,2	2,4		21.531	21.788	1,1±0,4	64,2	5,9	6,1
	0,18937	0,20525	1,1±0,3	15,9	2,1	2,1		22.896	23.008	1,1±0,4	33,2	4,6	4,7
SDJPG	0,22394	0,21765	1,0±0,0	12,7	1,9	1,9	SDJPG	25.811	25.947	1,1±0,4	15,8	3,5	3,7
	0,26099	0,28414	1,0±0,0	8,6	1,4	1,4		35.802	35.895	1,1±0,4	7,9	2,5	2,6
	0,31710	0,33580	1,0±0,0	5,0	1,0	1,0		48.319	48.495	1,0±0,0	2,6	1,0	1,0

Tabla 4.7.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 16H*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
CENSUS 16H							ELEVATORS						
WM	33.218	33.896	1,0±0,0	754,6	16,0	16,0	WM	0,33258	0,33485	1,0±0,0	511,0	18,0	18,0
JS	28.557	29.022	7,9±2,0	88,8	3,2	7,0	JS	0,28862	0,29267	4,7±1,0	35,5	3,0	4,0
SVAG	24.906	25.319	1,0±0,0	104,8	6,7	6,7	SVAG	0,24786	0,25033	1,0±0,0	44,6	6,0	6,0
SDJSG s/SV	25.757	26.002	8,7±2,1	72,0	3,9	16,0	SDJSG s/SV	0,23387	0,23466	10,6±1,9	58,8	3,4	18,0
SDJSG4M s/SV	24.395	24.758	4,0±0,6	133,0	5,5	16,0	SDJSG4M s/SV	0,23721	0,23774	4,0±0,0	105,1	5,2	18,0
	22.533	22.844	1,8±0,9	125,7	7,4	9,6		0,22158	0,22221	1,7±0,7	21,9	4,3	5,6
	23.182	23.458	1,8±0,9	98,0	5,4	7,6		0,23308	0,23294	2,1±0,9	16,0	3,0	4,9
SDJSG	24.925	25.198	2,3±1,1	46,7	3,6	6,2	SDJSG	0,24437	0,24395	2,0±0,7	11,9	2,4	3,9
	27.946	28.183	1,9±0,9	18,5	2,5	4,0		0,26514	0,26509	1,6±0,5	7,1	1,8	2,5
	50.624	50.712	1,0±0,0	2,7	1,0	1,0		0,31726	0,31712	1,0±0,0	2,9	1,0	1,0
	25.080	25.440	1,0±0,0	98,0	6,6	6,6		0,24957	0,25110	1,0±0,0	42,2	5,4	5,4
	27.115	27.248	1,1±0,5	39,5	4,6	4,8		0,25645	0,25697	1,0±0,0	29,6	4,7	4,7
SDJPG	30.848	31.065	1,2±0,7	24,9	3,9	4,3	SDJPG	0,26862	0,26888	1,0±0,0	16,7	3,8	3,8
	36.671	36.400	1,1±0,5	11,4	2,6	2,8		0,32372	0,32237	1,0±0,0	8,7	2,7	2,7
	52.513	52.531	1,0±0,0	2,7	1,0	1,0		3,06487	3,05410	1,0±0,0	2,5	1,0	1,0

Tabla 4.8.: Resultados obtenidos por SDJPG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
COMPACT							AILERONS						
WM	81,7173	82,4732	1,0±0,0	425,6	21,0	21,0	WM	0,06452	0,06534	1,0±0,0	1080,6	40,0	40,0
JS	44,3714	44,0328	4,3±0,7	21,6	2,5	5,0	JS	0,05896	0,05931	6,8±1,6	74,3	3,9	7,0
SVAG	46,7455	46,5157	1,0±0,0	20,2	5,3	5,3	SVAG	0,04887	0,04912	1,0±0,0	92,0	8,1	8,1
SDJSG s/SV	28,0853	28,3043	14,2±1,8	56,0	4,0	21,0	SDJSG s/SV	0,06007	0,06049	31,5±3,0	91,3	3,0	40,0
SDJSG4M s/SV	35,1213	35,8445	3,9±0,1	124,4	6,2	21,0	SDJSG4M s/SV	0,04635	0,04695	4,0±0,0	416,1	10,3	40,0
	12,6598	12,7747	1,4±0,6	45,5	8,4	9,8		0,04244	0,04263	2,7±0,9	82,2	6,9	10,6
	13,1206	13,2273	1,4±0,5	38,9	6,4	7,5		0,04777	0,04794	2,5±0,9	60,6	4,9	8,3
SDJSG	14,7702	14,9839	1,4± 0,5	27,7	4,6	5,8	SDJSG	0,05728	0,05760	2,5±0,9	30,3	3,5	6,1
	21,2532	21,3664	1,3±0,4	16,8	3,0	3,6		0,06656	0,06735	2,5±0,8	11,2	2,2	4,4
	143,9023	144,5856	1,0± 0,0	3,0	1,0	1,0		0,10148	0,10228	1,0±0,0	2,0	1,0	1,0
	41,2532	41,4038	1,5±1,1	22,5	4,7	6,2		0,05064	0,05096	1,4±1,1	76,3	7,3	8,2
	44,4700	44,4004	1,5±1,0	15,9	3,7	4,8		0,05603	0,05661	1,4±1,0	43,1	5,5	6,2
SDJPG	53,4364	53,5629	1,4±0,9	9,9	2,4	3,2	SDJPG	0,06782	0,06926	1,3±0,9	23,3	4,4	4,9
	114,0604	116,9202	1,1±0,7	6,2	2,2	2,1		0,09145	0,09121	1,0±0,0	10,9	3,4	3,4
	1.587,7189	1.576,2604	1,0±0,0	2,7	1,0	1,0		0,68423	0,69138	1,0±0,0	2,4	1,0	1,0

En las tablas 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 y 4.8 se puede apreciar cómo disminuye la tendencia modular que se adquiría con el modelizado de jerarquía en serie. A partir de seis variables, comienzan a hacer acto de presencia algunas soluciones con un modelo paralelo. En los problemas con mayor complejidad, el comportamiento se va consolidando, aunque no es tan pronunciado como en las soluciones obtenidas en el modelizado en serie.

En los problemas con menor complejidad, el algoritmo SDJPG puede llegar a tener un comportamiento de selección de variables. Las soluciones generadas con un modelo paralelo no llegan a mejorar ya que es bastante difícil conseguir un buen modelo paralelo con la existencia tan limitada de posibilidades de generarlas debido a la restricción del número de variables del problema. En cambio, un problema más complejo, o lo que es lo mismo, con un mayor número de variables a considerar, da más libertad a nuestra propuesta para crear soluciones con estructuras jerárquicas paralelas diferentes.

Con respecto a la interpretabilidad de los modelos paralelos se puede observar que disminuye en problemas menos complejos con respecto a un modelizado con jerarquía en serie debido a la disminución del comportamiento modular que presenta. Por otro lado, el algoritmo SDJPG aplicado a problemas más complejos es capaz de crear soluciones en donde la interpretabilidad es mejor que la proporcionada por el modelizado en serie, creando, de esta forma, modelos con menor número de reglas. Este es el caso de las soluciones obtenidas en los problemas *Computer Activity* y *Ailerons* con una reducción en el número total de reglas del 51 % y 7 % respectivamente en la solución más precisa.

Una mejora en la interpretabilidad repercute directamente en la calidad de la precisión de la solución. Si atendemos a la precisión obtenida del modelizado paralelo frente al modelizado en serie, podemos apreciar que aunque se obtienen valores de ECM_{entr} y ECM_{pru} en el mismo rango, el modelizado paralelo los empeora ligeramente. Esto se debe al problema que presentan estas estructuras jerárquicas: el acarreo del error que se propaga por los módulos. Las jerarquías paralelas parten de la peculiaridad de usar variables exógenas sólo en la primera capa. A diferencia de esto, una jerarquía en serie puede usar variables exógenas en cualquier capa de la jerarquía. Estas variables exógenas suavizan el error en el proceso de inferencia, cosa que no ocurre en el modelo paralelo, donde todas las variables de las capas intermedias son endógenas.

De esta forma, es más justo comparar el modelizado paralelo con el método *JS*.

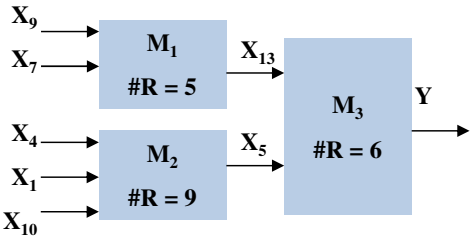
En la mayoría de los problemas, la tendencia del modelizado paralelo propuesto con respecto a JS se corresponde con una mejora en precisión que se aprecia tanto en la solución más precisa como las soluciones pertenecientes al primer cuartil. Esto es posible gracias a que el número de módulos es menor en el modelo que proponemos que el número de módulos obtenido mediante JS, teniendo un número de reglas y variables similar en el caso de la solución más precisa y disminuyendo ambos en las soluciones que se sitúan en el primer cuartil. De esta forma destacan los problemas *Census 16H*, *Census 16L* y *Ailerons* con una mejora en la precisión con respecto a *JS* de un 5 %, un 4 % y un 5 %, respectivamente. En estos problemas, la reducción en el número total de reglas es de un 56 %, un 47 % y un 42 % respectivamente. En el caso del problema *Computer Activity*, la mejora en precisión es de un 4 % atendiendo a la solución más precisa, obteniendo un número de reglas similar a *JS*.

Las figuras 4.10 y 4.11 muestran, para una determinada partición de datos, las soluciones obtenidas por SDJPG y JS que parte de un conjunto reducido de variables dado por el algoritmo SVAG para los problemas *Census 16L*, *Computer Activity* y *Ailerons*. Destacar que en las soluciones obtenidas con SDJPG no se han creado nuevas variables. SDJPG sólo usa las variables del problema a diferencia de JS que genera nuevas variables y se utilizan en el módulo de salida. En las figuras 4.10(c), 4.10(d) y 4.11(b) las nuevas variables se denotan por y_{ij}, donde el subíndice i hace referencia a la capa a la que pertenece la variable y el subíndice j denota el módulo en donde se infieren.

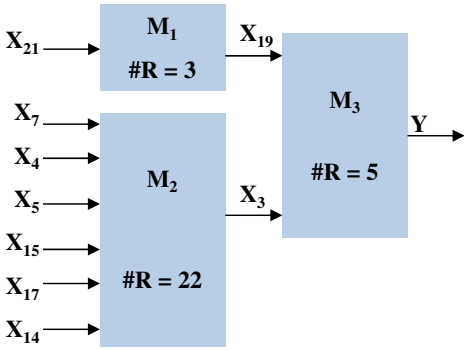
Como se puede apreciar, el modelo paralelo que proponemos proporciona mayor precisión que JS, minimizando el error de acarreo entre capas, dado que el número de módulos es inferior, y sin obviar la interpretabilidad del sistema ya que se conserva una estructura modular. Además, el número total de reglas de cada una de las soluciones obtenidas para cada problema por SDJPG es menor que el número de reglas obtenido por JS, manteniendo un número de variables por regla bastante similar.

La figura 4.12 ilustra un ejemplo real de una base de reglas jerárquica. Esta base de reglas se corresponde con el SDJP de la figura 4.10(a). El problema está formado por 16 variables de entrada. Como se puede apreciar, el número de variables de entrada se ha reducido a siete, distribuidas jerárquicamente en tres módulos.

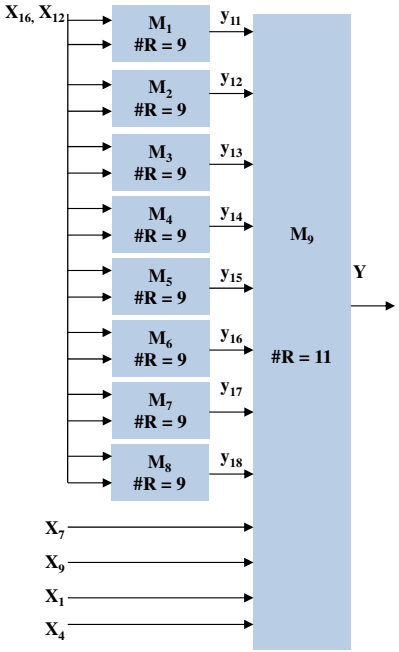
Atendiendo al método WM, algoritmo usado para el aprendizaje de la base de reglas en cada módulo del modelo en paralelo, podemos observar que se mejora tanto en precisión como en interpretabilidad por las soluciones situadas en la mediana y que son obtenidas por el algoritmo SDJPG. En este caso, observamos problemas



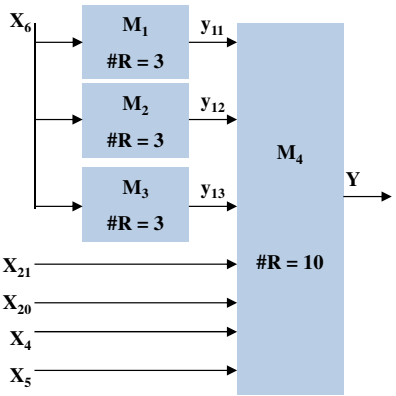
(a) SDJPG en *Census 16L*: ECM_{entr} = 22.361; ECM_{pru} = 20.361; #M = 3; #R = 20; #V = 7; $\bar{x}_{\#R}$ = 2,4



(b) SDJPG en *Computer Activity*: ECM_{entr} = 15,4507; ECM_{pru} = 16,2699; #M = 3; #R = 30; #V = 9; $\bar{x}_{\#R}$ = 4,8

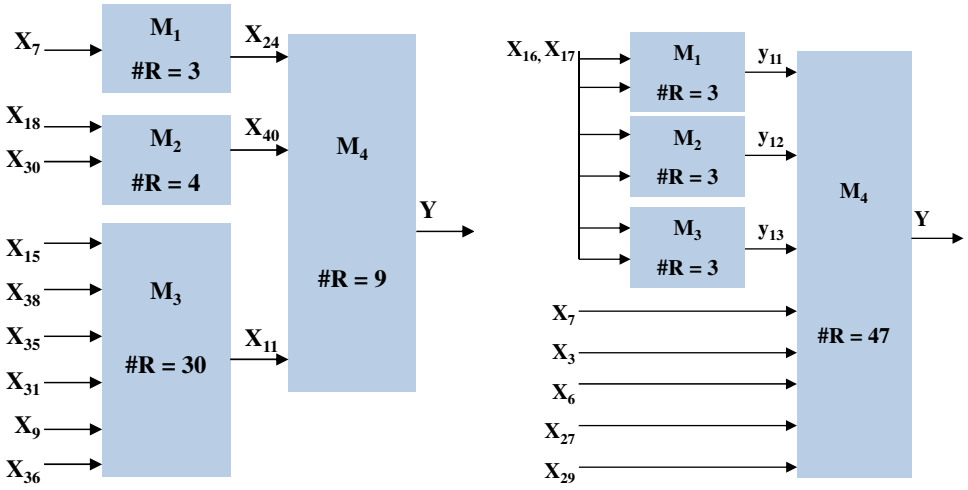


(c) JS en *Census 16L*: ECM_{entr} = 23.358; ECM_{pru} = 21.652; #M = 9; #R = 83; #V = 5; $\bar{x}_{\#R}$ = 2,3



(d) JS en *Computer Activity*: ECM_{entr} = 43,9187; ECM_{pru} = 43,3872; #M = 4; #R = 19; #V = 5; $\bar{x}_{\#R}$ = 2,3

Figura 4.10.: Ejemplos de soluciones obtenidas por SDJPG (Fig. 4.10(a), 4.10(b)) y las correspondientes soluciones obtenidas por JS partiendo de un conjunto reducido de variables dado por SVAG. Los resultados en esta figura (ECM_{entr}/ECM_{pru}) se deben multiplicar por 10^5 en el caso de *Census 16L*



(a) SDJPG en *Ailerons*: ECM_{entr} = 0,06342; ECM_{pru} = 0,06370; #M = 4;#R = 46; #V = 9; $\bar{x}_{\#R}$ = 4,7

(b) JS en *Ailerons*: ECM_{entr} = 0,10969; ECM_{pru} = 0,11055; #M = 4; #R = 56; #V = 7; $\bar{x}_{\#R}$ = 4,5

Figura 4.11.: Ejemplo de solución obtenida por SDJPG (Fig. 4.11(a)) y la correspondiente solución obtenidas por JS (Fig. 4.11(b)) partiendo de un conjunto reducido de variables dado por SVAG

como *Census 16H*, *Census 16L* y *Computer Activity* con una mejora en la precisión de un 7 %, 21 % y 35 % respectivamente. Atendiendo al número de reglas total del sistema jerárquico paralelo, éste se reduce en un 96 %, 98 % y 98 % respectivamente. Haciendo referencia a la solución obtenida en el primer cuartil en el problema *Ailerons*, tanto la precisión como la interpretabilidad mejora a la obtenida por WM en un 13 % y un 96 % respectivamente. Por otro lado, SDJPG reduce la complejidad considerablemente ya que WM usa todas las variables del problema en cada regla de la base de reglas y nuestra propuesta sólo usa las que considera oportunas según el conocimiento aprendido.

La figura 4.13 muestra el promedio del conjunto Pareto en los problemas *Census 16H*, *Census 16L*, *Computer Activity* y *Ailerons*. En la figura se puede apreciar que existe una mejora de la precisión del algoritmo SDJPG con respecto al algoritmo de selección de variables SVAG a medida que el número de reglas se incrementa. Atendiendo a la métrica $\bar{x}_{\#R}$, el valor de esta métrica en SDJPG es menor que en SVAG en

R$_1$: **SI** X_9 *es* P **y** X_7 *es* P **ENTONCES** X_{13} *es* G
R$_2$: **SI** X_9 *es* M **y** X_7 *es* P **ENTONCES** X_{13} *es* M
R$_3$: **SI** X_9 *es* P **y** X_7 *es* M **ENTONCES** X_{13} *es* P
R$_4$: **SI** X_9 *es* P **y** X_7 *es* G **ENTONCES** X_{13} *es* G
R$_5$: **SI** X_9 *es* G **y** X_7 *es* P **ENTONCES** X_{13} *es* P

(a) Base de Reglas del módulo M_1

R$_1$: **SI** X_4 *es* M **y** X_1 *es* P **y** X_{10} *es* P **ENTONCES** X_5 *es* P
R$_2$: **SI** X_4 *es* G **y** X_1 *es* P **y** X_{10} *es* P **ENTONCES** X_5 *es* P
R$_3$: **SI** X_4 *es* M **y** X_1 *es* P **y** X_{10} *es* M **ENTONCES** X_5 *es* P
R$_4$: **SI** X_4 *es* P **y** X_1 *es* P **y** X_{10} *es* M **ENTONCES** X_5 *es* P
R$_5$: **SI** X_4 *es* P **y** X_1 *es* P **y** X_{10} *es* P **ENTONCES** X_5 *es* G
R$_6$: **SI** X_4 *es* M **y** X_1 *es* M **y** X_{10} *es* P **ENTONCES** X_5 *es* P
R$_7$: **SI** X_4 *es* P **y** X_1 *es* P **y** X_{10} *es* G **ENTONCES** X_5 *es* P
R$_8$: **SI** X_4 *es* M **y** X_1 *es* G **y** X_{10} *es* P **ENTONCES** X_5 *es* P
R$_9$: **SI** X_4 *es* G **y** X_1 *es* P **y** X_{10} *es* M **ENTONCES** X_5 *es* P

(b) Base de Reglas del módulo M_2

R$_1$: **SI** X_{13} *es* G **y** X_5 *es* P **ENTONCES** Y *es* P
R$_2$: **SI** X_{13} *es* G **y** X_5 *es* M **ENTONCES** Y *es* P
R$_3$: **SI** X_{13} *es* M **y** X_5 *es* P **ENTONCES** Y *es* M
R$_4$: **SI** X_{13} *es* M **y** X_5 *es* M **ENTONCES** Y *es* P
R$_5$: **SI** X_{13} *es* G **y** X_5 *es* G **ENTONCES** Y *es* P
R$_6$: **SI** X_{13} *es* P **y** X_5 *es* P **ENTONCES** Y *es* P

(c) Base de Reglas del módulo M_3

Figura 4.12.: Base de Reglas obtenida por SDJPG en el problema *Census 16L* correspondiente a la figura 4.10(a)

los problemas *Census 16L* y *Computer Activity*. Esto indica que las reglas generadas por SDJPG son simples y de una mejor interpretabilidad que las obtenidas con el algoritmo de selección de variables. Por otro lado, encontramos los problemas *Census 16H* y *Ailerons* que a pesar de que el número de variables por regla se incremente en el algoritmo SDJPG, es capaz de encontrar dependencias entre variables que hacen que mejore a la precisión obtenida por SVAG.

También hemos elaborado unas gráficas de ranking en donde aparecen los algoritmos considerados en el capítulo anterior y nuestra propuesta paralela denominada por *SDJPG_M*, en el caso de la solución más precisa, y por *SDJPG_C1*, para representar el primer cuartil. Esto se pone de manifiesto en las figuras 4.14 y 4.15.

En estas gráficas podemos comprobar que nuestra propuesta en serie es seguida de cerca por nuestra propuesta en paralelo. Destacar que nuestra propuesta en paralelo obtiene soluciones menos precisas pero más interpretables que nuestra propuesta en serie, pero la propuesta en serie es capaz de establecer un mayor equilibrio entre

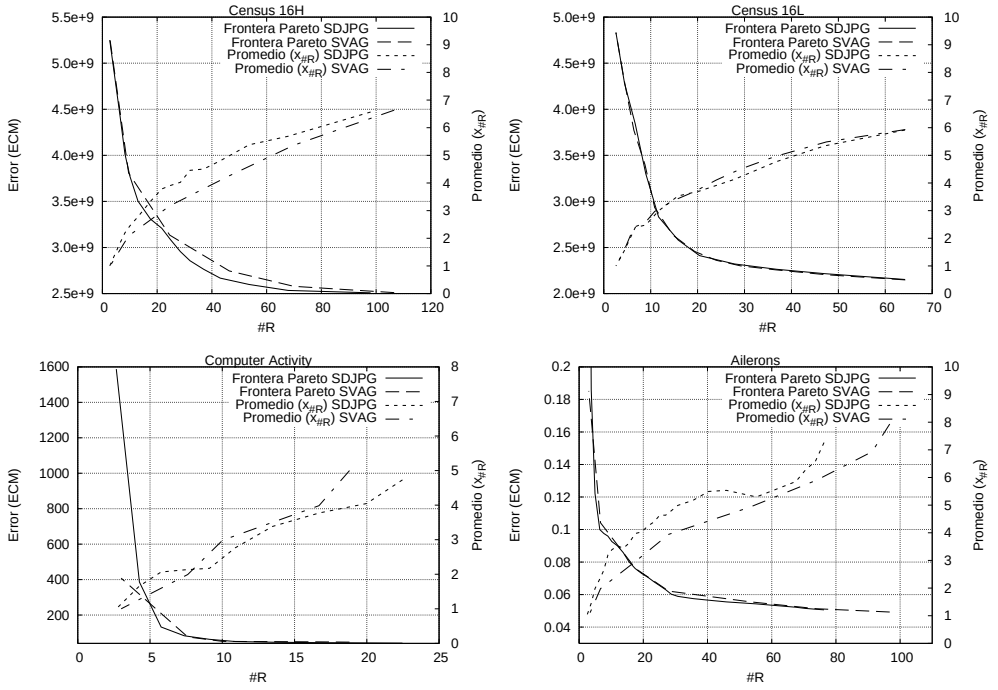


Figura 4.13.: Promedio de la frontera Pareto en varios problemas para el algoritmo SDJPG

precisión e interpretabilidad. Con respecto a WM, podemos decir que continua siendo el último del ranking, JS sigue muy de cerca a nuestra propuesta en paralelo en ambos criterios y SVAG es más preciso.

Las tablas 4.9 y 4.10 muestran las posiciones numéricas representadas en las figuras 4.14 y 4.15 de un método frente a un determinado problema. Contiene además una fila con la información sobre el ranking medio que ocupa cada algoritmo. En estas tablas podemos comprobar que nuestra propuesta de jerarquía en paralelo ocupa las posiciones intermedias en el ranking por debajo de nuestra propuesta de jerarquía en serie si consideramos la precisión como criterio de ordenación. En el caso del número de reglas, ambas propuestas ocupan posiciones intermedias indistintamente.

Al igual que con nuestra propuesta para generar jerarquías en serie, también hemos experimentado con otra versión de nuestra propuesta de modelo paralelo, modificando el criterio de dominancia, para así obtener soluciones con una, dos y tres

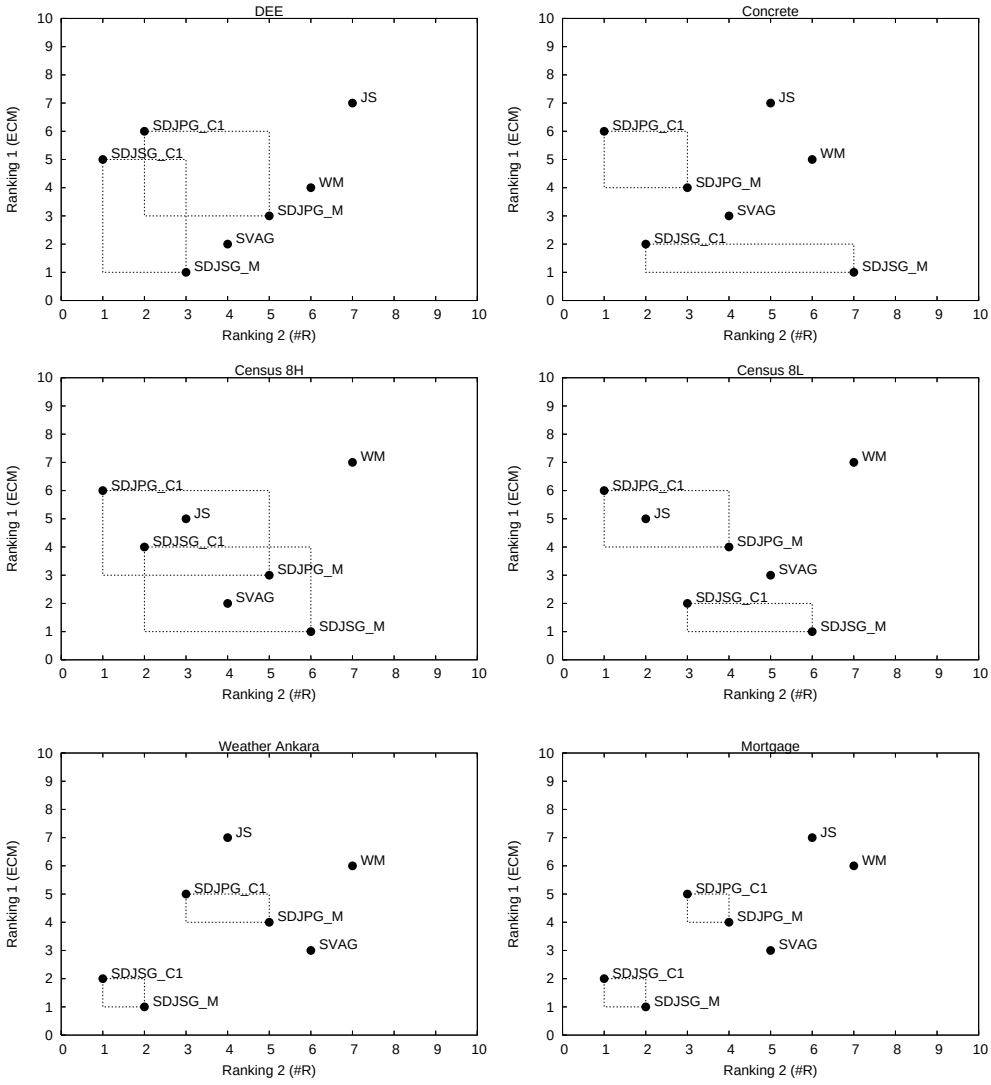


Figura 4.14.: Ranking de algoritmos en varios problemas para el algoritmo SDJPG

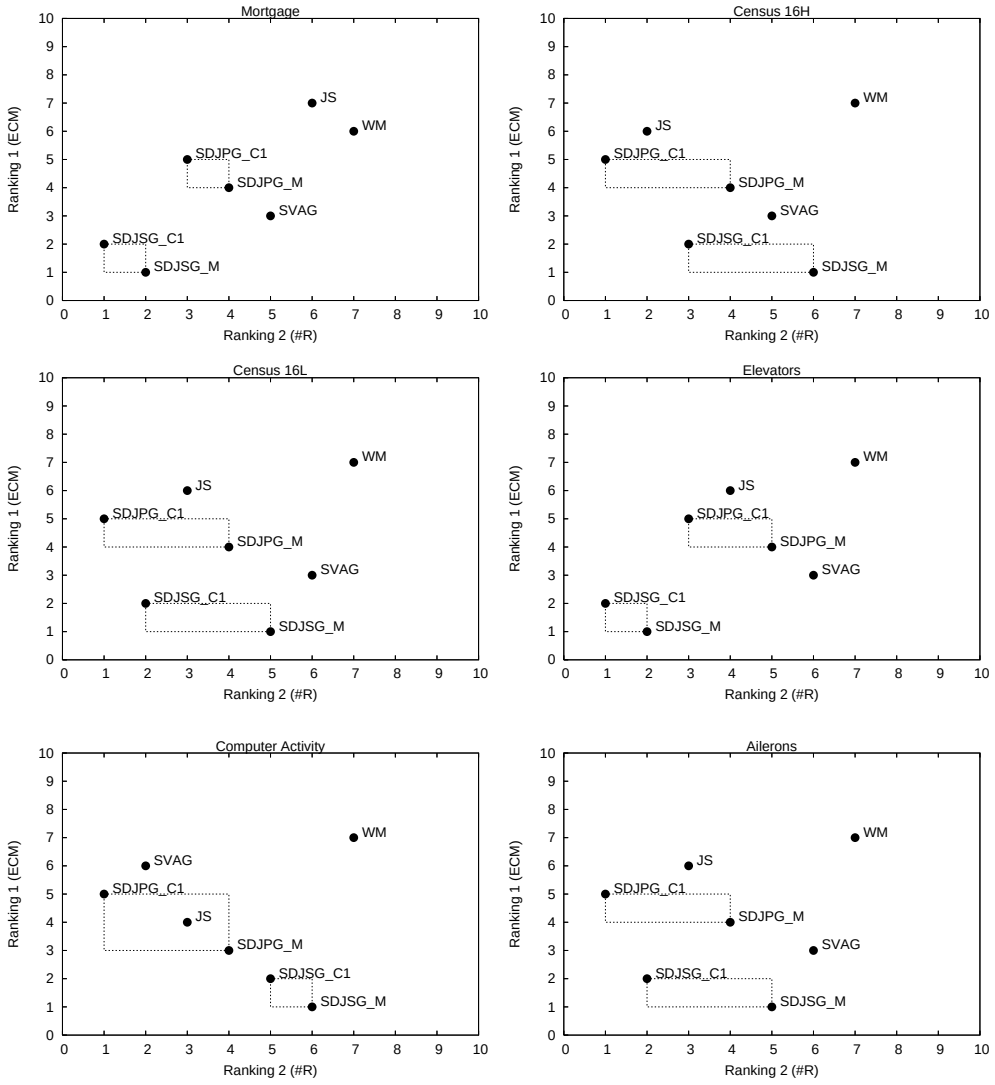


Figura 4.15.: Ranking de algoritmos en varios problemas para el algoritmo SDJPG

Tabla 4.9.: Ranking de algoritmos considerando como criterio ECM_{tra}

Método / Problema	*WM*	*SVAG*	*JS*	*SDJSG_M*	*SDJSG_C1*	*SDJPG_M*	*SDJPG_C1*
DEE	4	2	7	1	5	3	6
Concrete	5	3	7	1	2	4	6
Census 8H	7	2	5	1	4	3	6
Census 8L	7	3	5	1	2	4	6
Weather Ankara	6	3	7	1	2	4	5
Mortgage	6	3	7	1	2	4	5
Treasury	7	4	6	1	3	2	5
Elevators	7	3	6	1	2	4	5
Computer Activity	7	6	4	1	2	3	5
Ailerons	7	3	6	1	2	4	5
Ranking medio	6	3	6	1	3	3	5

Tabla 4.10.: Ranking de algoritmos considerando como criterio el número de reglas

Método / Problema	*WM*	*SVAG*	*JS*	*SDJSG_M*	*SDJSG_C1*	*SDJPG_M*	*SDJPG_C1*
DEE	6	4	7	3	1	5	2
Concrete	6	4	5	7	2	3	1
Census 8H	7	4	3	6	2	5	1
Census 8L	7	5	2	6	3	4	1
Weather Ankara	7	6	4	2	1	5	3
Mortgage	7	5	6	2	1	4	3
Treasury	7	3	6	4	2	5	1
Elevators	7	6	4	2	1	5	3
Computer Activity	7	2	3	6	5	4	1
Ailerons	7	6	3	5	2	4	1
Ranking medio	7	4	4	4	2	4	2

capas. De esta forma, podemos observar el comportamiento y beneficios dados una estructura jerárquica en paralelo. La figura 4.16 representa el promedio de las fronteras del Pareto con una, dos y tres capas obtenidas en los problemas *Census 16H*, *Census 16L*, *Computer Activity* y *Ailerons* por nuestro algoritmo. En la leyenda de la gráfica se muestra el promedio del tamaño de cada Pareto. Como se observa en la figura, el modelizado de tres capas es el que aporta soluciones menos precisas y es el modelo que obtiene menor número de soluciones en el Pareto. Por otro lado, el modelizado con una y dos capas son los que proporcionan soluciones más precisas, en donde el tamaño medio del conjunto Pareto es similar. De esta forma, el comportamiento de los tres modelos confirma la premisa que veníamos comentando en esta sección: la propagación del error a través de las capas de la jerarquía paralela produce una menor precisión en las soluciones. Destacar que el algoritmo SDJPG, a pesar de

la similaridad en las soluciones con el modelizado de una capa, obtiene soluciones más precisas en ciertas regiones del espacio de búsqueda.

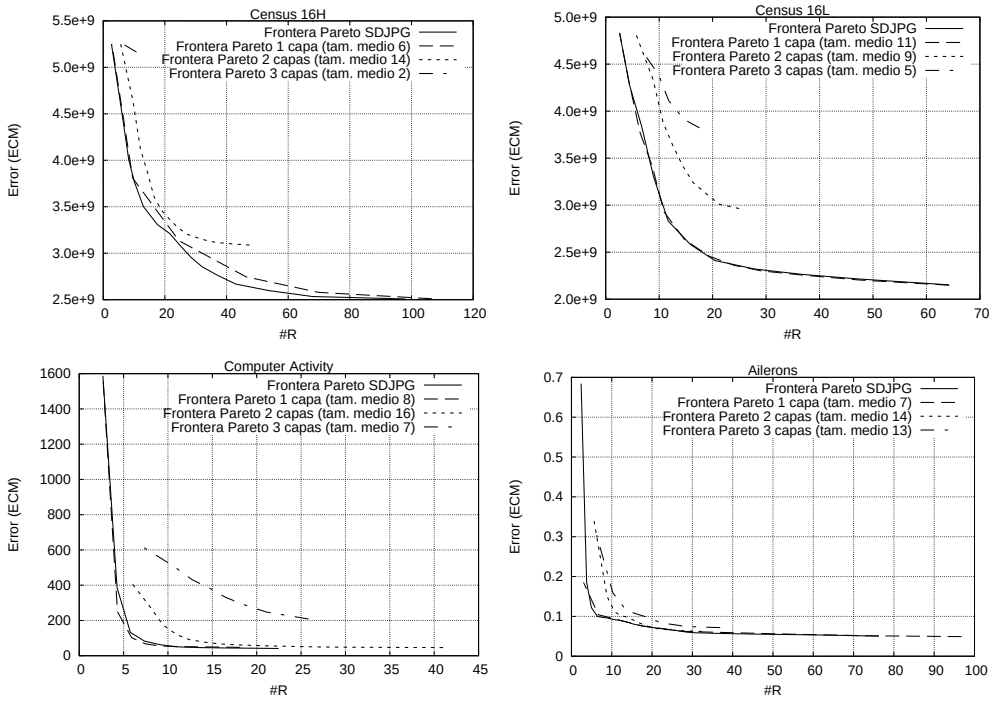


Figura 4.16.: Promedio de la frontera del Pareto con una, dos y tres capas en varios problemas para el algoritmo SDJPG

Como hemos podido observar en este análisis, la jerarquía paralela no incluye variables exógenas en las capas intermedias de la estructura. Las variables exógenas aportan datos sin ruido. En la siguiente sección, comprobaremos el comportamiento de una jerarquía que si las incluye, la jerarquía híbrida.

4.2. Modelizado de Sistemas Difusos Jerárquicos Híbridos

Esta sección describe en los siguientes apartados nuestra propuesta jerárquica híbrida, los resultados obtenidos y su análisis. También incluye un apartado de comparativa frente a otros métodos.

4.2.1. Descripción del algoritmo

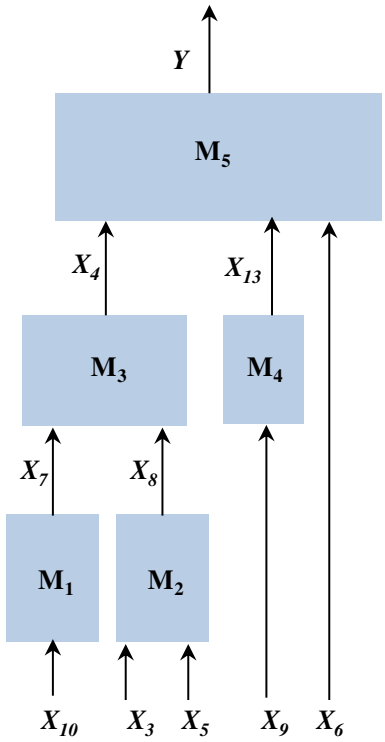
4.2.1.1. Codificación

Para codificar un modelo jerárquico híbrido, hemos hecho uso de un modelo basado en árboles n-arios pero ampliando el grado de libertad estructural. En este caso, no existe ningún tipo de restricción referente al tipo de variable en los módulos de la jerarquía: un módulo puede tener como variable de entrada variables endógenas, exógenas o ambas a cualquier nivel. Este tipo de modelizado hace que la codificación de la solución sea más representativa, orientativa y flexible a la hora de aplicar los operadores genéticos.

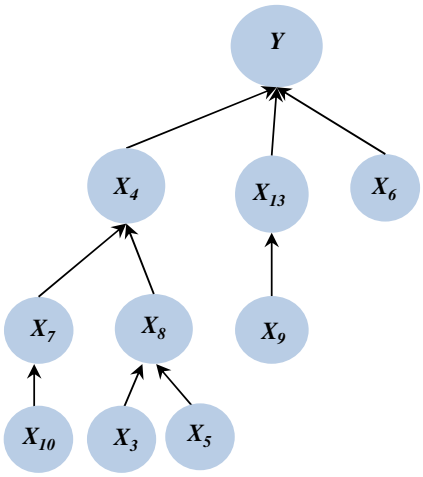
El modelizado se compone de un nodo raíz del árbol, que se corresponde con el módulo de salida del sistema jerárquico híbrido y del que cuelgan el resto de módulos formando las sucesivas capas. La figura 4.17 muestra la correspondencia entre un sistema difuso jerárquico híbrido (figura 4.17(a)), un árbol (figura 4.17(b)) y su codificación(figura 4.17(c)). En el árbol, las variables contenidas en los nodos intermedios (X_7, X_8, X_4 y X_{13}) dependen de las variables de los nodos hoja (X_{10}, X_3, X_5 y X_9), y la variable de salida (Y), situado en el nodo raíz, se genera a partir de los nodos intermedios y una variable exógena (X_6). Este sistema difuso jerárquico híbrido considera un total de 15 variables, de las cuales cinco son variables exógenas y cuatro son variables endógenas (las variables X_1, X_2, X_{11}, X_{12}, X_{14} y X_{15} no se consideran en esta estructura) distribuidas en tres capas bien definidas. Como se puede comprobar, no todas las variables del problema tienen porqué usarse en la estructura jerárquica, aunque también cabe la posibilidad de que todas las variables del problema se usen.

Atendiendo a la codificación de las soluciones del espacio de búsqueda, en un sistema difuso jerárquico híbrido, cada módulo (o nodo) está formado por un vector de tipo entero que contiene las variables de entrada al módulo, un campo de tipo entero que almacena la variable de salida y un vector de referencias a los módulos de los que depende.

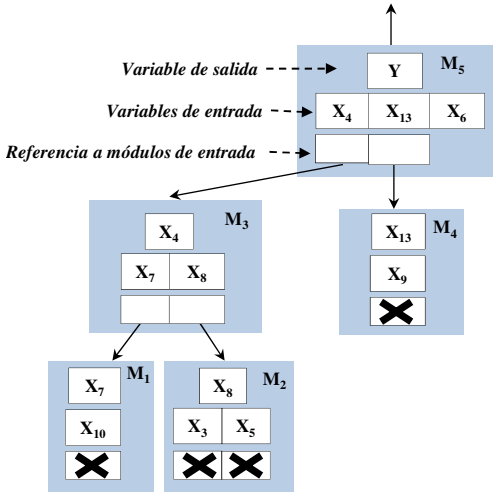
A pesar de que el flujo de datos a través de la jerarquía transita desde los nodos hoja hacia el nodo raíz (módulo de salida del sistema jerárquico), al igual que en el modelizado jerárquico paralelo, el orden natural para referirnos a las capas del sistema difuso jerárquico híbrido es desde los nodos hoja al nodo raíz. Un aspecto a destacar es que dado que los módulos que sólo tienen como entrada variables exógenas no dependen de ningún otro, el vector de referencia a los módulos de los que depende está inicializado a un valor nulo, de ahí que en la figura las posiciones



(a) Sistema difuso jerárquico híbrido



(b) Estructura de datos en árbol



(c) Esquema de codificación

Figura 4.17.: Correspondencia entre un sistema difuso jerárquico híbrido, una estructura de datos en árbol y su esquema de codificación

correspondientes a este vector aparezcan señalados con un 'aspa'. Finalmente, cada árbol constituye un individuo de la población.

4.2.1.2. Inicialización

El algoritmo parte de una población aleatoria, donde cada individuo es un sistema jerárquico difuso híbrido. Las estructuras jerárquicas híbridas generadas en la población de partida se caracterizan por tener un número de capas, un número de variables distribuidas en la jerarquía y un número de módulos por capa totalmente aleatorio.

Por otro lado, el algoritmo controla la generación del número de capas de la estructura jerárquica por individuo. Establecemos un máximo de tres capas debido al acarreo del error generado por los módulos a través de la estructura jerárquica, limitando simultáneamente el espacio de búsqueda de soluciones para evitar soluciones claramente no prometedoras. Si el individuo generado está formado por más de tres capas, se descarta y se genera uno nuevo. Dado que una estructura jerárquica híbrida puede tener tanto variables endógenas como exógenas en sus capas intermedias, el algoritmo hace eco de ello y lo implementa en la generación de soluciones aleatorias.

Con respecto al número de módulos no existe ningún tipo de limitación y las variables contenidas como elementos de la estructura jerárquica, tanto endógenas como exógenas, son variables pertenecientes al problema que se considera.

4.2.1.3. Operador de Cruce

El operador de cruce, al igual que en las estructuras jerárquicas anteriores, se aplica según una probabilidad entre pares de cromosomas padres atendiendo al tipo de variable (exógena o endógena) que tienen en común los padres y controlando el número de capas en cada hijo generado con el objetivo de reducir el espacio de búsqueda, creando una nueva solución si el cruce entre dos padres genera un número de capas menor o igual a tres. Si el individuo generado está formado por más de tres capas, se descarta y se genera uno nuevo. El operador de cruce implementa un enfoque centrado en el padre, de forma que los descendientes principalmente heredan la información de uno de los padres y el resto se hereda del padre secundario para añadir diversidad.

Por otro lado, no limita el número de módulos en la jerarquía de las capas intermedias y pueden contener tanto variables exógenas como endógenas. La figura

4.18 muestra un árbol de decisión con todos los casos que considera este operador de cruce.

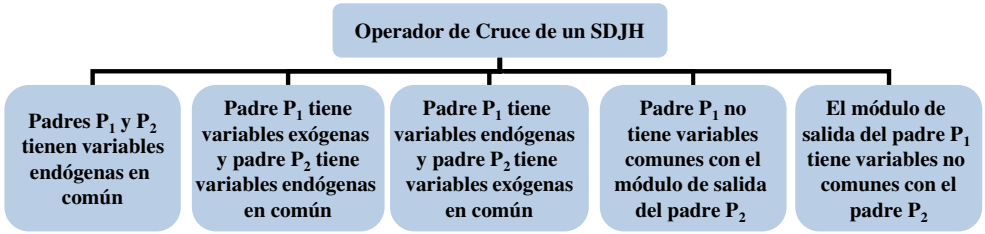


Figura 4.18.: Árbol de decisión general del operador de cruce de un SDJH

Dados dos padres, P_1 y P_2, el operador de cruce de un sistema difuso jerárquico híbrido se aplica como sigue:

4.2.1.3.1. Ambos padres, P_1 y P_2, tienen variables endógenas en común

Dados dos padres, P_1 y P_2, y ambos tienen variables endógenas en común, el proceso de generación del descendiente, D_1, es como sigue:

1. El descendiente D_1 es una copia de P_1.
2. Escoger aleatoriamente una variable endógena común en P_1 y P_2.
3. Podar la rama en D_1 que genera la variable común seleccionada.
4. Copiar la rama de P_2 que infiere la variable común en D_1.
5. Borrar las variables repetidas en D_1 de la rama heredada de P_2.

El descendiente D_2 se genera de la misma forma pero centrado en P_2. La figura 4.19(a) ilustra un ejemplo de este caso.

4.2.1.3.2. El padre P_1 tiene variables exógenas y el padre P_2 tiene variables endógenas en común

Dados dos padres, P_1, con variables exógenas, y P_2, con variables endógenas en común, el proceso de generación de los descendientes, D_1 y D_2, es el siguiente:

1. El descendiente D_1 es una copia de P_1.
2. Escoger aleatoriamente una variable exógena de D_1 que sea endógena en P_2.
3. Copiar la rama de P_2 que infiere la variable común en D_1.
4. Borrar las variables repetidas en D_1 de la rama heredada de P_2.

La figura 4.19(b) muestra un ejemplo de este caso.

4.2.1.3.3. El padre P_1 tiene variables endógenas y el padre P_2 tiene variables exógenas en común

El cruce de dos padres, P_1 y P_2, con variables endógenas y exógenas comunes respectivamente, se realiza de la siguiente forma:

1. El descendiente D_1 es una copia de P_1.
2. Escoger aleatoriamente una variable endógena de P_1 que sea exógena en P_2.
3. Podar la rama que genera la variable común seleccionada en D_1.

La figura 4.19(c) representa este caso.

4.2.1.3.4. El padre P_1 no tiene variables comunes con el módulo de salida del padre P_2

Si se cruza un padre P_1 que no tiene variables en común con las variables de entrada del módulo de salida del padre P_2, el descendiente D_1 se crea de la siguiente manera:

1. El descendiente D_1 es una copia de P_1.
2. Escoger una variable de entrada del módulo de salida de P_2 aleatoriamente y que no sea común con alguna variable del módulo de salida de D_1.
3. Copiar la variable seleccionada del módulo de salida de P_2 en el módulo de salida de D_1.

La figura 4.19(d) representa este caso.

4.2.1.3.5. El módulo de salida del padre P_1 no tiene variables comunes con el padre P_2

Si el módulo de salida del padre P_1 no presenta variables comunes con las variables del padre P_2, el operador de cruce genera al descendiente D_1 como sigue:

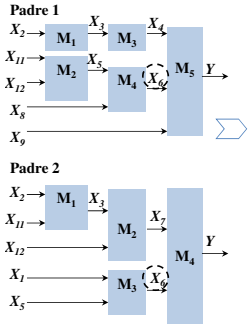
1. El descendiente D_1 es una copia de P_1.
2. Escoger aleatoriamente una variable de entrada del módulo de salida de D_1 que no se encuentre en el padre P_2.
3. Eliminar la variable seleccionada del módulo de salida de D_1.

La figura 4.19(e) muestra un ejemplo de este caso.

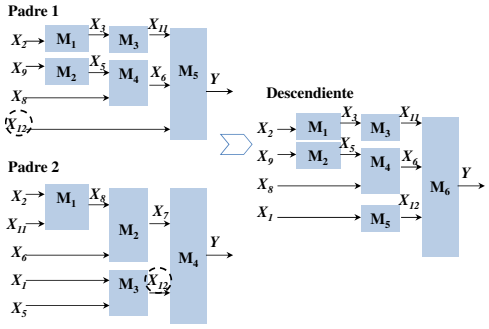
4.2.1.4. Operador de Mutación

El operador de mutación controla que no se cree una estructura jerárquica híbrida con más de tres capas. La mutación se aplica a nivel de variable y la selección de cada operador se realiza de forma aleatoria (figura 4.20). El algoritmo 12 muestra el esquema general del funcionamiento de este operador de mutación. La mutación distingue entre:

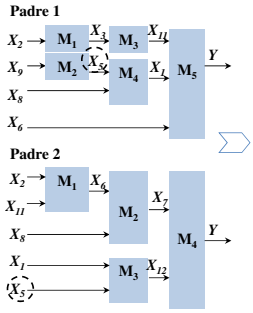
- *Insertar una variable.* Esta mutación afecta sólo a las variables no usadas. Para la inserción, se selecciona un módulo m de la jerarquía híbrida y una variable no usada v, ambos aleatoriamente. A continuación, se inserta la variable no usada v como exógena en el módulo m.
- *Eliminar una variable.* Se puede definir como un operador de poda del árbol que representa la jerarquía híbrida. Consiste en seleccionar una variable v aleatoriamente entre todas las existentes en el SDJH. Si v es:
  - *Exógena*, v se elimina del módulo. Si v es la única existente en el módulo, se elimina y, además, la variable endógena generada por ese módulo se convierte en exógena. Esto implica la desaparición del módulo de la estructura jerárquica híbrida. La figura 4.21(a) ilustra este caso.
  - *Endógena*, v se convierte en exógena y se elimina los módulos que cuelga que cuelgan de ella. La figura 4.21(b) muestra este tipo de mutación.



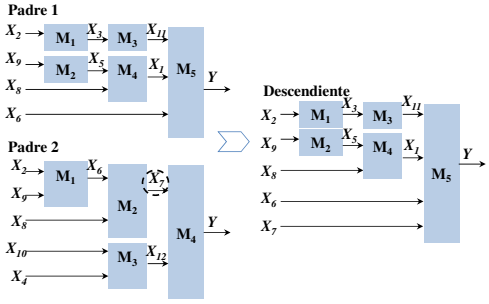
(a) Padres P_1 y P_2 tienen variables en común: Caso Endógena-Endógena



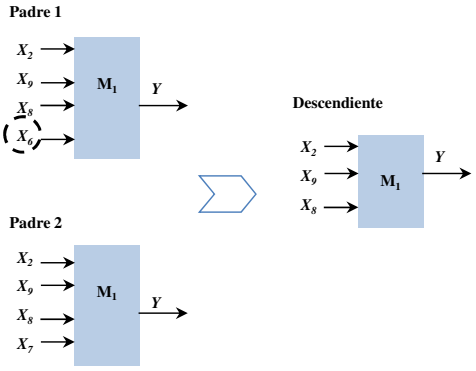
(b) Padres P_1 y P_2 tienen variables en común: Caso Exógena-Endógena



(c) Padres P_1 y P_2 tienen variables en común: Caso Endógena-Exógena



(d) Padre P_1 no tiene variables comunes con el módulo de salida del padre P_2



(e) El módulo de salida del padre P_1 tiene variables no comunes con el padre P_2

Figura 4.19.: Operador de cruce en un SDJH

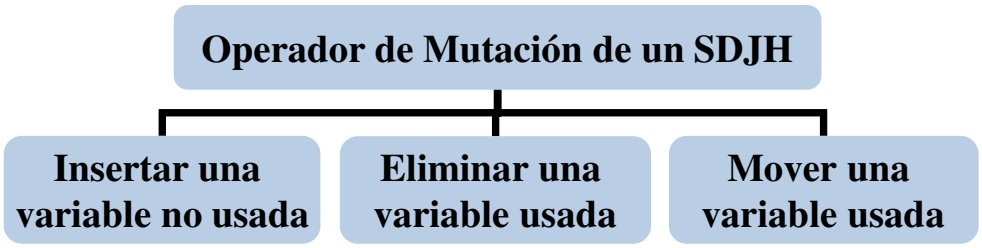


Figura 4.20.: Árbol de decisión del operador de mutación de un SDJH

- *Mover una variable.* Para ello, se selecciona aleatoriamente un módulo m. Seguidamente, se escoge una variable v de forma aleatoria del SDJH. Si v es:

  - *Exógena*, eliminar v del módulo en que se encuentra e insertarla en el módulo m.
  - *Endógena*, cortar la rama desde v hasta los nodos hoja e insertarla en m.

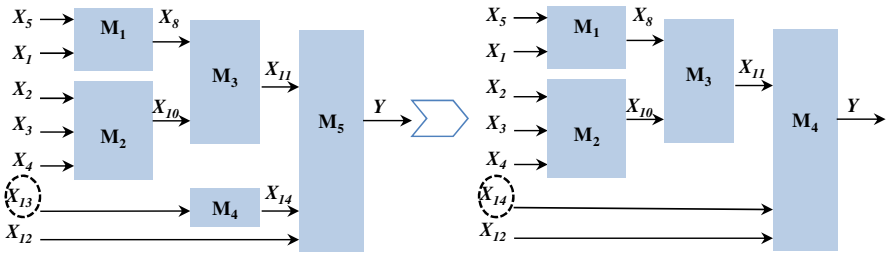
  En ambos casos, si v es la única del módulo, la variable de salida del módulo del que cuelga la variable v se convierte en exógena y el módulo que generaba v, desaparece (figuras 4.21(c) y 4.21(d)).

Destacar que no existe ningún tipo de restricción a la hora de operar sobre la estructura jerárquica híbrida porque, como hemos comentado anteriormente, esta estructura se caracteriza por tener más libertad que las estructuras jerárquicas en serie y la paralela. Lo que sí se ha limitado es el número de capas a tres para evitar el acarreo del error a través de los módulos. Si se genera un individuo con más de tres capas, este se obvia y se genera uno nuevo.

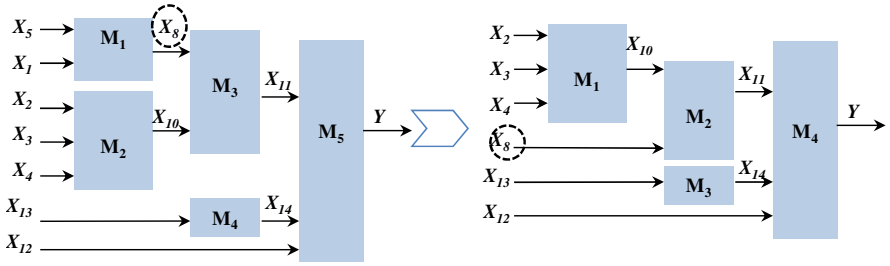
4.2.1.5. Mecanismo de Inferencia

El algoritmo SDJHG implementa un mecanismo de inferencia similar a los algoritmos SDJSG y SDJPG donde hace uso de una inferencia Max–Min, la T-norma del mínimo como conjunción y el centro de gravedad como defuzzificación.

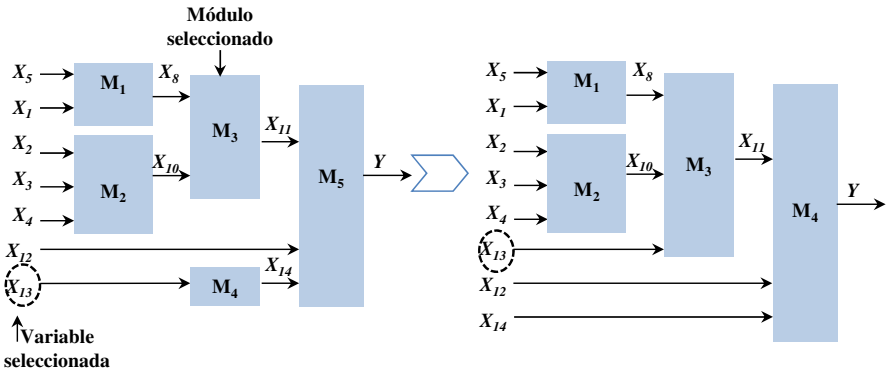
Al igual que en el algoritmo SDJPG, las estructuras jerárquicas híbridas realizan la inferencia de izquierda a derecha, es decir, desde los módulos que contienen variables exógenas hasta el módulo de salida. Dado que las variables endógenas de la



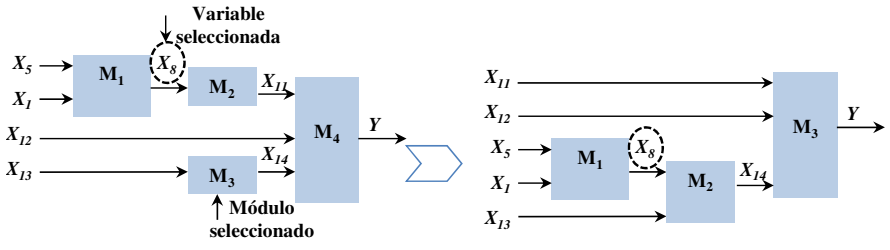
(a) Eliminar una variable exógena



(b) Eliminar una variable endógena



(c) Mover una variable exógena



(d) Mover una variable endógena

Figura 4.21.: Operador de mutación de un SDJH

Algoritmo 12 Operador de mutación de un SDJH

Entrada: Un sistema difuso jerárquico híbrido: h

```
 1: r = U[0,1] {Probabilidad uniforme}
 2: Si (r < 0,5 o #M >1) entonces
 3:     s = U[0,1] {Probabilidad uniforme}
 4:     Si (s < 1/3) entonces
 5:         m = Elegir_un_módulo_aleatoriamente(h)
 6:         v = Elegir_una_variable_aleatoriamente(h)
 7:         Mover_variable(h,v,m)
 8:     Sino
 9:         Si (1/3 <= s < 2/3) entonces
10:             v = Elegir_una_variable_aleatoriamente(h)
11:             Eliminar_variable(h, v)
12:         Sino
13:             m = Elegir_un_módulo_aleatoriamente(h)
14:             v = Elegir_una_variable_no_usada_aleatoriamente(h)
15:             Insertar_variable(h, v, m)
16:         Fin Si
17:     Fin Si
18: Sino
19:     m = Elegir_un_módulo_con_variables_exógenas_aleatoriamente(h)
20:     v = Elegir_una_variable_exógena_aleatoriamente(h, m)
21:     e = Convertir_variable_en_endógena(h, m, v)
22:     t = U[0,1]; {Probabilidad uniforme}
23:     Si (#Variables_exógenas(m) >= 1 y t <0,5) entonces
24:         w = Elegir_una_variable_exógena_aleatoriamente(h, m)
25:         Insertar_variable(h, w, e)
26:     Sino
27:         Si (#Variables_no_usadas(h)>0) entonces
28:             x = Elegir_una_variable_no_usada_aleatoriamente(h)
29:             Insertar_variable(h, x, e)
30:         Fin Si
31:     Fin Si
32: Fin Si
```

capa intermedia necesitan la inferencia de los módulos situados en la capa anterior para obtener su valor, la inferencia se realiza capa a capa.

Por otro lado, como se ha mencionado anteriormente, la inferencia de una variable por un módulo produce un error que se acarrea a través de las capas. Por ello, es aconsejable que el número de módulos de la jerarquía híbrida no sea demasiado alto.

4.2.1.6. Aprendizaje de la Base de Reglas

El algoritmo de modelizado de SDJH realiza un aprendizaje de la base de reglas del sistema jerárquico a nivel de módulo atendiendo a las variables de entrada y salida del mismo. Cada módulo aprende un conjunto de reglas tipo Mamdani mediante el método de WM, en donde las variables exógenas se extraen del conjunto de datos y las variables endógenas se infieren dependiendo de las variables de entrada del módulo correspondiente. Destacar que el algoritmo no está diseñado para la creación de nuevas variables en el proceso de inferencia de las mismas, es decir, el algoritmo solo tiene en cuenta variables pertenecientes al problema que se considera. Dicha inferencia se implementa siguiendo el sentido desde la primera capa hasta el módulo de salida de la estructura jerárquica híbrida.

La figura 4.22 muestra un SDJ Híbrido formado por cuatro módulos: los módulos M_1 y M_2 al formar parte de las hojas del árbol jerárquico, solo reciben como entrada variables exógenas, sin embargo, los módulos de las capas intermedias (M_3 y M_4) reciben tanto variables endógenas como exógenas dado que la naturaleza de la estructura es híbrida. Los módulos de la estructura M_1, M_2 y M_3 infieren variables naturales (X_2, X_7 y X_5) que a su vez constituyen la entrada a módulos en la siguiente capa. El módulo M_4 infiere la salida del sistema jerárquico híbrido. Los módulos de las capas intermedias M_3 y M_4, como se puede apreciar, reciben cada uno una variable exógena (X_9 y X_2, respectivamente). El sistema obtenido se evalúa mediante ejemplos de datos de entrenamiento y eligiendo los valores correspondientes de variables exógenas en ese módulo del conjunto de datos.

4.2.1.7. Enfoque Multiobjetivo

El algoritmo de modelizado de SDJH utiliza el algoritmo NSGA-II para minimizar las dos funciones objetivo descritas en el capítulo 3(ver subsección 5.5.2).

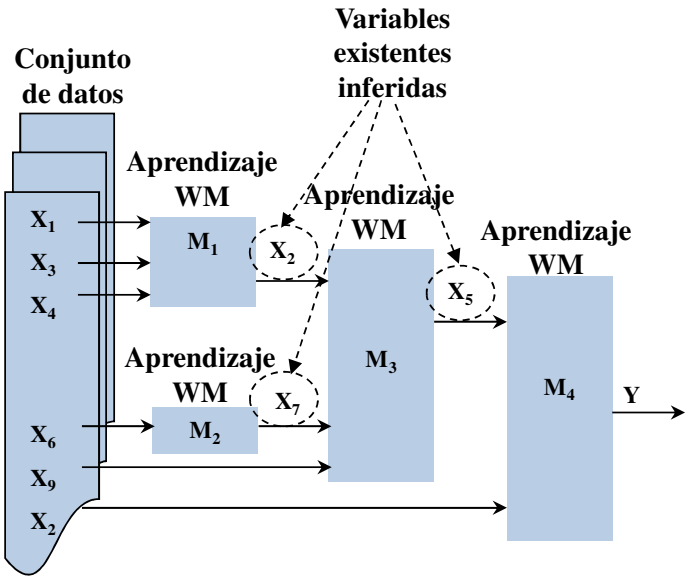


Figura 4.22.: Ejemplo del proceso de inferencia en un SDJ Híbrido

4.2.2. Experimentos

En este algoritmo, al igual que en los algoritmos anteriormente propuestos, considera los 14 problemas de regresión detallados en la subsección 3.2.1, usando la misma configuración experimental a nivel paramétrico (subsección 3.2.3). Además, incluiremos los métodos de comparación de los capítulos anteriores incluyendo los algoritmos en serie y paralelo propuestos.

4.2.3. Estudio Experimental

4.2.3.1. Resultados obtenidos

Las tablas 4.11, 4.12, 4.13, 4.14, 4.15, 4.16 y 4.17 contienen los resultados obtenidos por la propuesta híbrida a la que se ha llamado Sistema Difuso Jerárquico Híbrido Genético (SDJHG), donde ECM_{entr} y ECM_{prue} son los valores del error de aproximación (ecuación 3.1) en entrenamiento y prueba respectivamente para un conjunto de datos; *#M*, *#R* y *#V* hacen referencia al número de módulos, al número de reglas difusas y al número de variables de entrada respectivamente; $\sigma_{\#M}$ es la desviación

típica del número de módulos y $\bar{x}_{\#R}$ es el número de variables por regla.

4.2.3.2. Análisis

Atendiendo a los resultados dados por el algoritmo de modelizado de jerarquía híbrida, se puede apreciar cómo en todos los problemas se obtienen soluciones con varios módulos. Concretamente, en los problemas con un número bajo de variables (de cuatro a ocho) el algoritmo obtiene soluciones en donde la jerarquía está constituida por un número de módulos en torno a dos. Sin embargo, el número de módulos se incrementa (en torno a tres) cuando la complejidad del problema es mayor.

Con respecto a la precisión de los sistemas jerárquicos híbridos hay que destacar que mejora considerablemente en comparación al algoritmo de WM y es más significativa en los problemas con mayor complejidad. En rasgos generales, en los problemas de cuatro a seis variables, la solución más precisa mejora a la solución dada por WM; los problemas de ocho y nueve variables destacan por su mejora en precisión en las soluciones situadas en el primer cuartil; por último, encontramos los problemas complejos, en donde las soluciones situadas en la mediana mejoran a la solución obtenida por WM.

En el problema *DEE*, de seis variables, la precisión en el primer cuartil se incrementa un poco con respecto a la solución obtenida por WM en el ECM_{entr}, pero en el ECM_{pru} es más bajo y la interpretabilidad se reduce casi a la mitad, concretamente el número de reglas se reduce en un 42 %, reduciéndose además el número de variables que utiliza el modelo.

Los resultados obtenidos en el problema *Elevators*, de 18 variables, muestran que la solución más precisa obtenida por nuestro algoritmo es mejor que la solución de WM. Tanto en el ECM_{entr} como en el ECM_{pru}, la interpretabilidad es mejor porque se consiguen SDJH con un número bajo de variables con casi tres módulos y el número de reglas se ha reducido en un 90 %. El ECM_{entr} del tercer cuartil mejora en precisión a WM, el ECM_{pru} es también más preciso y el número de reglas que obtiene nuestro algoritmo se reduce en un 97 %.

En el problema *Ailerons* (40 variables), podemos comprobar que nuestra solución más precisa es mejor que la solución obtenida por WM. Además el número de reglas se reduce en un 90 % y el número de variables en un 69 %. El ECM_{entr} y el ECM_{pru} de la mediana es bajo y la interpretabilidad es mejor que la solución dada por WM, debido a la reducción del número de variables y su distribución en módulos.

La figura 4.23 ilustra un ejemplo de estructura jerárquica híbrida obtenida por

Tabla 4.11.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
		LASER							ELE2				
WM	265,21	278,58	1,0±0,0	58,4	4,0	4,0	WM	112.271	112.719	1,0±0,0	65,0	4,0	4,0
JS	299,46	323,24	6,0±0,0	55,6	2,1	4,0	JS	117.172	127.343	6,0±0,0	36,6	1,3	3,0
SVAG	265,21	278,58	1,0±0,0	58,4	4,0	4,0	SVAG	90.956	100.100	1,0±0,0	36,6	3,0	3,0
SDJSG s/SV	263,74	293,06	1,2±0,4	55,0	3,7	4,0	SDJSG s/SV	98.205	102.994	2,0±0,0	41,6	2,8	4,0
SDJSG4M s/SV	263,74	293,06	1,2±0,4	55,0	3,7	4,0	SDJSG4M s/SV	98.205	102.994	2,0±0,0	41,6	2,8	4,0
	263,74	293,06	1,2±0,4	55,0	3,8	4,0		72.231	75.695	1,4±0,5	15,1	1,9	2,4
	309,01	327,13	1,3±0,3	38,9	2,9	3,3		183.426	185.363	1,3±0,2	12,4	1,7	2,1
SDJSG	342,74	364,76	1,1±0,2	25,3	2,3	2,5	SDJSG	349.446	347.696	1,6±0,4	10,2	1,3	1,9
	694,76	728,33	1,1±0,1	15,0	1,7	1,8		487.700	473.206	1,5±0,3	7,7	1,1	1,5
	1.469,79	1.513,25	1,0±0,0	5,0	1,0	1,0		591.403	565.462	1,0±0,0	5,0	1,0	1,0
	265,21	278,58	1,0±0,0	58,4	4,0	4,0		90.956	100.100	1,0±0,0	36,6	3,0	3,0
	332,67	347,05	1,0±0,0	32,4	2,8	2,8		101.363	98.542	1,0±0,0	23,4	2,3	2,3
SDJPG	602,40	621,70	1,0±0,0	16,7	2,0	2,0	SDJPG	311.068	327.170	1,0±0,0	15,0	2,0	2,0
	1.278,19	1.335,25	1,0±0,0	11,5	1,9	1,9		492.073	501.914	1,0±0,0	10,3	1,9	1,9
	1.469,79	1.513,25	1,0±0,0	5,0	1,0	1,0		631.654	631.655	1,0±0,0	5,0	1,0	1,0
	263,74	293,06	1,2±0,4	55,0	3,7	4,0		90.956	100.100	1,0±0,0	36,6	3,0	3,0
	309,01	327,13	1,2±0,3	38,8	2,9	3,2		100.369	102.897	1,0±0,0	23,5	2,3	2,3
SDJHG	342,74	364,76	1,1±0,2	25,3	2,3	2,5	SDJHG	315.381	314.876	1,4±0,2	14,8	1,6	2,0
	694,76	728,33	1,1±0,1	15,0	1,7	1,8		579.027	578.520	1,6±0,3	9,3	1,7	1,1
	1.469,79	1.513,25	1,0±0,0	5,0	1,0	1,0		631.654	631.655	1,0±0,0	5,0	1,0	1,0

Tabla 4.12.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
DEE							CONCRETE						
WM	0,14117	0,22888	1,0±0,0	178,4	6,0	6,0	WM	73,080	97,006	1,0±0,0	310,4	8,0	8,0
JS	0,23050	0,33550	9,8±4,7	179,6	3,0	6,0	JS	95,540	113,840	13,6±2,3	300,2	3,5	8,0
SVAG	0,13900	0,22390	1,0±0,0	155,2	5,4	5,4	SVAG	72,477	88,984	1,0±0,0	270,0	7,0	7,0
SDJSG s/SV	0,13961	0,22728	1,4±0,5	162,5	5,6	6,0	SDJSG s/SV	71,725	88,160	1,8±1,0	263,4	6,8	8,0
SDJSG4M s/SV	0,13948	0,22713	1,4±0,5	161,3	5,5	6,0	SDJSG4M s/SV	70,355	88,666	1,7±0,9	267,7	6,8	8,0
	0,13728	0,23338	1,1±0,2	155,1	5,6	5,6		50,859	73,521	1,1±0,2	312,5	7,0	7,1
	0,15308	0,22436	1,2±0,3	87,5	4,0	4,2		55,786	71,124	1,2±0,5	231,5	5,4	5,8
SDJSG	0,18253	0,21145	1,5±0,5	38,8	2,7	3,5	SDJSG	70,310	79,036	1,2±0,4	133,7	4,1	4,4
	0,23023	0,23913	1,5±0,6	14,7	1,9	2,4		112,184	117,851	1,2±0,4	48,1	2,6	2,9
	0,34664	0,35492	1,0±0,0	4,5	1,0	1,0		240,195	248,009	1,0±0,0	3,9	1,0	1,0
	0,13900	0,22390	1,0±0,0	155,2	5,4	5,4		72,510	88,810	1,0± 0,0	267,4	6,9	6,9
	0,15769	0,22729	1,0±0,0	105,7	4,4	4,4		85,994	97,443	1,0± 0,0	141,0	4,7	4,7
SDJPG	0,18913	0,22881	1,0±0,0	60,9	3,4	3,4	SDJPG	104,736	111,313	1,2± 0,6	70,6	3,6	3,8
	0,23843	0,26082	1,1±0,4	28,7	2,5	2,6		174,188	176,147	1,1± 0,5	29,5	2,5	2,7
	0,39815	0,41665	1,0±0,0	5,0	1,0	1,0		257,855	251,284	1,0± 0,0	5,0	1,0	1,0
	0,13869	0,22610	1,2±0,4	156,4	5,4	5,6		67,949	81,196	1,9±0,8	248,3	6,2	7,1
	0,15328	0,22801	1,6±0,5	104,1	4,1	4,7		78,291	82,861	1,8±0,7	140,9	4,4	5,3
SDJHG	0,18532	0,22159	1,4±0,5	54,0	3,0	3,7	SDJHG	107,197	108,259	1,6±0,4	56,8	3,1	3,8
	0,23893	0,26291	1,8±0,7	25,7	1,9	3,2		162,378	162,989	1,5±0,4	24,5	2,1	2,7
	0,39815	0,41665	1,0±0,0	5,0	1,0	1,0		257,855	251,284	1,0±0,0	5,0	1,0	1,0

Tabla 4.13.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 8L* y *Census 8H*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
CENSUS 8L							CENSUS 8H						
WM	15.234	16.122	1,0±0,0	762,8	8,0	8,0	WM	23.350	24.504	1,0±0,0	713,4	8,0	8,0
JS	14.503	15.188	6,3±0,7	86,7	2,9	4,0	JS	22.038	23.708	11,2±7,0	291,7	3,1	5,0
SVAG	13.468	14.023	1,0±0,0	140,1	5,0	5,0	SVAG	20.903	21.907	1,0±0,0	397,4	5,4	5,4
SDJSG s/SV	12.750	13.090	3,3±0,6	145,0	4,2	8,0	SDJSG s/SV	20.734	21.846	2,7±0,6	419,5	5,6	8,0
SDJSG4M s/SV	12.546	12.974	3,1±0,6	166,4	4,6	8,0	SDJSG4M s/SV	20.648	21.717	2,4±0,6	444,1	6,0	8,0
	12.739	13.211	1,6±0,6	169,4	5,6	6,7		20.284	21.238	1,4±0,6	437,5	6,4	6,8
	13.431	13.876	1,6±0,8	98,3	4,3	5,2		21.195	22.072	1,3±0,4	273,7	4,7	5,2
SDJSG	16.045	16.516	1,6±0,8	39,9	2,9	3,8	SDJSG	22.714	23.242	1,6±0,8	119,3	3,3	4,4
	20.584	20.950	1,8±0,8	19,8	1,8	2,8		25.140	25.270	1,5±0,7	28,6	2,3	2,8
	30.996	31.408	1,0±0,0	5,0	1,0	1,0		30.391	30.345	1,0±0,0	5,0	1,0	1,0
	13.517	14.067	1,0±0,0	129,0	4,9	4,9		20.903	21.907	1,0±0,0	397,4	5,4	5,4
	14.899	15.366	1,0±0,0	44,3	3,8	3,8		22.911	23.646	1,0±0,0	180,9	4,4	4,4
SDJPG	23.149	23.509	1,0±0,0	24,3	2,9	2,9	SDJPG	24.791	24.992	1,1±0,3	76,9	3,7	3,8
	28.739	29.228	1,0±0,0	12,8	2,1	2,1		31.525	31.526	1,0±0,0	14,8	2,1	2,1
	52.840	52.850	1,0±0,0	3,6	1,0	1,0		52.837	52.848	1,0±0,0	3,8	1,0	1,0
	12.956	13.347	1,9±0,7	131,8	5,0	6,3		20.773	21.882	1,2±0,4	446,8	5,6	5,8
	13.420	13.735	1,6±0,5	71,3	3,6	4,5		22.286	22.916	1,7±0,6	166,4	3,8	5,1
SDJHG	16.508	16.947	1,6±0,6	38,1	2,7	3,9	SDJHG	24.075	24.237	1,7±0,4	82,7	3,1	4,1
	20.829	21.205	1,4±0,3	23,1	2,1	2,7		26.361	26.095	1,7±0,3	26,6	2,1	3,3
	32.971	33.309	1,0±0,0	5,0	1,0	1,0		38.646	38.473	1,0±0,0	5,0	1,0	1,0

Tabla 4.14.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
		WEANK							MORTGAGE				
WM	9,75605	12,25650	1,0±0,0	456,8	9,0	9,0	WM	0,25600	0,26806	1,0±0,0	197,2	15,0	15,0
JS	13,13610	16,56026	6,9±2,4	100,8	3,0	5,0	JS	0,90399	0,90393	7,1±2,4	80,6	2,7	5,0
SVAG	7,58657	8,84168	1,0±0,0	140,2	5,4	5,4	SVAG	0,11871	0,12598	1,0±0,0	61,5	5,4	5,4
SDJSG s/SV	7,10852	7,35495	3,3±0,6	148,9	4,8	9,0	SDJSG s/SV	0,11376	0,12191	8,1±1,2	101,7	2,9	15,0
SDJSG4M s/SV	7,10044	7,45815	3,0±0,4	170,0	4,2	9,0	SDJSG4M s/SV	0,10971	0,11165	4,0±0,0	131,9	4,2	15,0
	6,63906	6,84172	1,9±0,8	64,8	3,8	5,7		0,10097	0,10848	2,1±1,0	40,7	4,6	7,2
	7,46367	7,49844	2,0±0,8	40,1	3,0	4,9		0,10856	0,11200	2,0±0,9	33,6	3,7	5,5
SDJSG	8,63689	8,62772	1,8±0,6	25,7	2,4	3,7	SDJSG	0,13511	0,13602	1,8±0,9	27,6	2,9	4,3
	10,75790	11,00243	1,5±0,5	13,8	1,9	2,6		0,19417	0,20913	1,5±0,6	16,5	2,0	2,7
	13,66512	13,62635	1,0±0,0	2,1	1,0	1,0		0,32981	0,34130	1,0±0,0	5,0	1,0	1,0
	7,58927	8,85345	1,0±0,0	137,4	5,3	5,3		0,11956	0,12515	1,0±0,0	61,1	5,3	5,3
	8,95733	9,25060	1,0±0,0	83,3	4,2	4,2		0,13853	0,14424	1,0±0,0	45,6	4,6	4,6
SDJPG	10,93894	11,27933	1,0±0,0	28,3	3,0	3,0	SDJPG	0,16755	0,17893	1,0±0,0	27,1	3,2	3,2
	25,91153	26,12875	1,0±0,0	12,0	2,0	2,0		0,25081	0,25673	1,0±0,0	15,9	2,3	2,3
	1298,61069	1362,51850	1,0±0,0	3,6	1,0	1,0		0,34735	0,37721	1,0±0,0	5,0	1,0	1,0
	7,45402	8,46947	1,5±0,5	144,9	5,4	6,1		0,10652	0,11764	2,8±1,2	77,8	4,6	8,2
	8,22755	8,76478	1,5±0,4	99,3	4,0	5,0		0,12443	0,12823	2,4±1,1	56,6	3,7	6,6
SDJHG	9,50709	9,47301	1,5±0,4	55,1	3,1	4,0	SDJHG	0,15164	0,15368	2,4±1,1	40,5	2,8	5,5
	13,08096	13,09503	1,2±0,4	26,4	2,4	2,6		0,20716	0,21191	1,9±0,7	22,4	2,2	3,4
	39,14571	39,14548	1,0±0,0	5,0	1,0	1,0		0,34836	0,37810	1,0±0,0	5,0	1,0	1,0

Tabla 4.15.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 16L*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
TREASURY							CENSUS 16L						
WM	0,80168	0,80909	1,0±0,0	195,2	15,0	15,0	WM	32.505	32.887	1,0±0,0	682,0	16,0	16,0
JS	0,19888	0,20635	2,8±1,6	18,8	1,8	2,0	JS	23.815	24.108	6,4±2,3	62,6	3,1	7,0
SVAG	0,16362	0,17451	1,0±0,0	16,8	2,2	2,2	SVAG	21.507	21.804	1,0±0,0	79,1	6,4	6,4
SDJSG s/SV	0,11325	0,11924	10,3±1,1	80,0	1,9	15,0	SDJSG s/SV	19.772	19.972	8,3±1,7	62,0	3,6	16,0
SDJSG4M s/SV	0,13011	0,13808	4,0±0,0	130,2	4,4	15,0	SDJSG4M s/SV	19.797	20.069	3,9±0,1	114,7	5,1	16,0
	0,14692	0,15025	1,5±0,8	17,5	3,0	4,1		17.884	18.163	2,6±1,0	75,1	5,7	10,5
	0,16304	0,17243	1,5±0,7	16,4	2,5	3,4		18.529	18.823	2,5±1,0	50,2	4,5	8,2
SDJSG	0,18550	0,19955	1,6±0,9	14,8	2,0	2,8	SDJSG	20.052	20.273	2,6±0,9	30,9	3,3	6,6
	0,23274	0,24625	1,3±0,5	10,7	1,6	2,0		23.701	23.878	2,2±1,0	16,1	2,4	4,3
	0,31949	0,33830	1,0±0,0	5,0	1,0	1,0		41.698	41.841	1,0±0,0	3,0	1,0	1,0
	0,16295	0,17131	1,1±0,5	17,8	2,2	2,4		21.531	21.788	1,1±0,4	64,2	5,9	6,1
	0,18937	0,20525	1,1±0,3	15,9	2,1	2,1		22.896	23.008	1,1±0,4	33,2	4,6	4,7
SDJPG	0,22394	0,21765	1,0±0,0	12,7	1,9	1,9	SDJPG	25.811	25.947	1,1±0,4	15,8	3,5	3,7
	0,26099	0,28414	1,0±0,0	8,6	1,4	1,4		35.802	35.895	1,1±0,4	7,9	2,5	2,6
	0,31710	0,33580	1,0±0,0	5,0	1,0	1,0		48.319	48.495	1,0±0,0	2,6	1,0	1,0
	0,15254	0,16402	1,6±1,0	22,2	2,1	3,3		19.816	19.986	2,7±1,0	57,2	4,6	9,5
	0,18280	0,19488	1,5±0,9	18,6	1,9	2,8		21.076	21.145	2,8±1,0	40,3	3,6	7,7
SDJHG	0,21428	0,22697	1,4±0,8	15,6	1,7	2,4	SDJHG	23.183	23.281	2,5±1,1	23,6	2,8	5,9
	0,25140	0,27253	1,2±0,4	10,9	1,4	1,8		27.010	27.223	2,1±0,6	13,4	2,1	4,0
	0,31710	0,33580	1,0±0,0	5,0	1,0	1,0		45.508	45.508	1,0±0,0	3,0	1,0	1,0

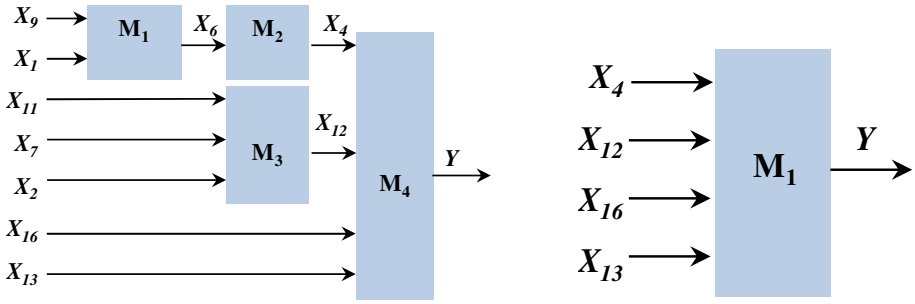
Tabla 4.16.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla. Los resultados de esta tabla (ECM_{entr}/ECM_{pru}) deben ser multiplicados por 10^5 en el caso de *Census 16H*

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
CENSUS 16H							ELEVATORS						
WM	33.218	33.896	1,0±0,0	754,6	16,0	16,0	WM	0,33258	0,33485	1,0±0,0	511,0	18,0	18,0
JS	28.557	29.022	7,9±2,0	88,8	3,2	7,0	JS	0,28862	0,29267	4,7±1,0	35,5	3,0	4,0
SVAG	24.906	25.319	1,0±0,0	104,8	6,7	6,7	SVAG	0,24786	0,25033	1,0±0,0	44,6	6,0	6,0
SDJSG s/SV	25.757	26.002	8,7±2,1	72,0	3,9	16,0	SDJSG s/SV	0,23387	0,23466	10,6±1,9	58,8	3,4	18,0
SDJSG4M s/SV	24.395	24.758	4,0±0,6	133,0	5,5	16,0	SDJSG4M s/SV	0,23721	0,23774	4,0±0,0	105,1	5,2	18,0
	22.533	22.844	1,8±0,9	125,7	7,4	9,6		0,22158	0,22221	1,7±0,7	21,9	4,3	5,6
	23.182	23.458	1,8±0,9	98,0	5,4	7,6		0,23308	0,23294	2,1±0,9	16,0	3,0	4,9
SDJSG	24.925	25.198	2,3±1,1	46,7	3,6	6,2	SDJSG	0,24437	0,24395	2,0±0,7	11,9	2,4	3,9
	27.946	28.183	1,9±0,9	18,5	2,5	4,0		0,26514	0,26509	1,6±0,5	7,1	1,8	2,5
	50.624	50.712	1,0±0,0	2,7	1,0	1,0		0,31726	0,31712	1,0±0,0	2,9	1,0	1,0
	25.080	25.440	1,0±0,0	98,0	6,6	6,6		0,24957	0,25110	1,0±0,0	42,2	5,4	5,4
	27.115	27.248	1,1±0,5	39,5	4,6	4,8		0,25645	0,25697	1,0±0,0	29,6	4,7	4,7
SDJPG	30.848	31.065	1,2±0,7	24,9	3,9	4,3	SDJPG	0,26862	0,26888	1,0±0,0	16,7	3,8	3,8
	36.671	36.400	1,1±0,5	11,4	2,6	2,8		0,32372	0,32237	1,0±0,0	8,7	2,7	2,7
	52.513	52.531	1,0±0,0	2,7	1,0	1,0		3,06487	3,05410	1,0±0,0	2,5	1,0	1,0
	24.771	25.121	2,2±1,1	93,9	5,3	8,5		0,23703	0,23874	2,6±1,3	51,4	4,7	8,5
	26.081	26.428	2,4±0,9	53,3	3,6	6,8		0,24418	0,24537	2,5±0,9	36,3	3,5	7,1
SDJHG	28.502	28.677	2,5±0,8	30,7	2,8	5,7	SDJHG	0,25725	0,25753	2,4±0,9	23,7	2,8	5,9
	35.173	35.108	1,9±0,6	12,9	2,0	3,5		0,28829	0,29040	2,1±1,0	14,1	2,3	4,1
	52.607	52.631	1,0±0,0	3,0	1,0	1,0		0,56691	0,56691	1,0±0,0	3,0	1,0	1,0

Tabla 4.17.: Resultados obtenidos por SDJHG, siendo ECM_{entr} y ECM_{pru} los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*	*Método*	ECM_{entr}	ECM_{pru}	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
COMPACT							AILERONS						
WM	81,7173	82,4732	1,0±0,0	425,6	21,0	21,0	WM	0,06452	0,06534	1,0±0,0	1080,6	40,0	40,0
JS	44,3714	44,0328	4,3±0,7	21,6	2,5	5,0	JS	0,05896	0,05931	6,8±1,6	74,3	3,9	7,0
SVAG	46,7455	46,5157	1,0±0,0	20,2	5,3	5,3	SVAG	0,04887	0,04912	1,0±0,0	92,0	8,1	8,1
SDJSG s/SV	28,0853	28,3043	14,2±1,8	56,0	4,0	21,0	SDJSG s/SV	0,06007	0,06049	31,5±3,0	91,3	3,0	40,0
SDJSG4M s/SV	35,1213	35,8445	3,9±0,1	124,4	6,2	21,0	SDJSG4M s/SV	0,04635	0,04695	4,0±0,0	416,1	10,3	40,0
	12,6598	12,7747	1,4±0,6	45,5	8,4	9,8		0,04244	0,04263	2,7±0,9	82,2	6,9	10,6
	13,1206	13,2273	1,4±0,5	38,9	6,4	7,5		0,04777	0,04794	2,5±0,9	60,6	4,9	8,3
SDJSG	14,7702	14,9839	1,4± 0,5	27,7	4,6	5,8	SDJSG	0,05728	0,05760	2,5±0,9	30,3	3,5	6,1
	21,2532	21,3664	1,3±0,4	16,8	3,0	3,6		0,06656	0,06735	2,5±0,8	11,2	2,2	4,4
	143,9023	144,5856	1,0± 0,0	3,0	1,0	1,0		0,10148	0,10228	1,0±0,0	2,0	1,0	1,0
	41,2532	41,4038	1,5±1,1	22,5	4,7	6,2		0,05064	0,05096	1,4±1,1	76,3	7,3	8,2
	44,4700	44,4004	1,5±1,0	15,9	3,7	4,8		0,05603	0,05661	1,4±1,0	43,1	5,5	6,2
SDJPG	53,4364	53,5629	1,4±0,9	9,9	2,4	3,2	SDJPG	0,06782	0,06926	1,3±0,9	23,3	4,4	4,9
	114,0604	116,9202	1,1±0,7	6,2	2,2	2,1		0,09145	0,09121	1,0±0,0	10,9	3,4	3,4
	1.587,7189	1.576,2604	1,0±0,0	2,7	1,0	1,0		0,68423	0,69138	1,0±0,0	2,4	1,0	1,0
	24,4144	24,4760	3,0±1,1	42,2	4,5	10,1		0,04659	0,04686	3,7±1,3	102,9	6,0	12,5
	25,5950	25,6866	2,9±1,2	32,2	3,6	8,3		0,04949	0,04942	3,3±1,0	77,0	4,8	9,8
SDJHG	28,7035	29,0452	2,8±1,1	28,3	3,0	7,4	SDJHG	0,05378	0,05408	2,9±1,0	45,2	3,8	7,6
	39,3777	39,4531	2,4±1,0	18,0	2,4	5,0		0,07023	0,07097	2,5±0,8	21,3	2,5	5,0
	396,7598	405,1998	1,0±0,0	3,0	1,0	1,0		0,19484	0,19666	1,0±0,0	3,0	1,0	1,0

nuestro algoritmo (figura 4.23(a)) y su correspondencia con la solución obtenida por WM (figura). La solución jerárquica híbrida es una solución no dominada de una partición de un conjunto de datos del problema *Census16L*.



(a) SDJHG en *Census 16L*: ECM_{entr} = 20.187; ECM_{pru} = 21.567; #M = 4; #R = 40; #V = 10; $\bar{x}_{\#R}$ = 3,4

(b) WM en *Census 16L*: ECM_{entr} = 34.284; ECM_{pru} = 31.865; #M = 1; #R = 55; #V = 4; $\bar{x}_{\#R}$ = 4,0

Figura 4.23.: Ejemplo de solución obtenida por SDJHG y su correspondiente solución obtenida por WM sobre el conjunto de variables de entrada del último módulo obtenido por SDJHG. Los resultados en esta figura (ECM_{entr}/ECM_{pru}) se deben multiplicar por 10^5

La figura 4.24 ilustra un ejemplo real de una base de reglas de una estructura jerárquica híbrida en el problema *Census 16L*. Esta base de reglas se corresponde con la generada en el SDJH de la figura 4.23(a). El problema está formado por 16 variables de entrada. Como se puede apreciar, el número de variables de entrada se ha reducido a 10, distribuidas jerárquicamente en cuatro módulos.

Si atendemos al método WM, en la tabla de resultados podemos observar que el modelo jerárquico híbrido que proponemos lo mejora tanto en precisión como en interpretabilidad en las soluciones situadas en el primer cuartil en los problemas de hasta ocho variables. En problemas a partir de nueve variables la mejora en estos dos objetivos es considerable, llegando a mejorar a las soluciones obtenidas por WM las soluciones obtenidas por nuestra propuesta situadas en el tercer cuartil. En este caso, observamos problemas como *Mortgage*, *Census 16L* y *Elevators* con una mejora en la precisión de un 19 %, 17 % y 13 % respectivamente. Además, el número de reglas total del sistema jerárquico híbrido se ve reducido en un 89 %, 98 % y 97 % respectivamente. Por otro lado, la reducción de la complejidad se hace latente en el momento

R$_1$: **SI** X_9 *es* P **y** X_1 *es* P **ENTONCES** X_6 *es* P
R$_2$: **SI** X_9 *es* M **y** X_1 *es* P **ENTONCES** X_6 *es* M
R$_3$: **SI** X_9 *es* P **y** X_1 *es* M **ENTONCES** X_6 *es* M
R$_4$: **SI** X_9 *es* P **y** X_1 *es* G **ENTONCES** X_6 *es* M

(a) Base de Reglas del módulo M_1

R$_1$: **SI** X_6 *es* P **ENTONCES** X_4 *es* M
R$_2$: **SI** X_6 *es* M **ENTONCES** X_4 *es* G

(b) Base de Reglas del módulo M_2

R$_1$: **SI** X_{11} *es* G **y** X_7 *es* P **y** X_2 *es* P **ENTONCES** X_{12} *es* G
R$_2$: **SI** X_{11} *es* M **y** X_7 *es* P **y** X_2 *es* P **ENTONCES** X_{12} *es* G
R$_3$: **SI** X_{11} *es* P **y** X_7 *es* P **y** X_2 *es* P **ENTONCES** X_{12} *es* G
R$_4$: **SI** X_{11} *es* M **y** X_7 *es* M **y** X_2 *es* P **ENTONCES** X_{12} *es* M
R$_5$: **SI** X_{11} *es* G **y** X_7 *es* M **y** X_2 *es* P **ENTONCES** X_{12} *es* M
R$_6$: **SI** X_{11} *es* G **y** X_7 *es* G **y** X_2 *es* P **ENTONCES** X_{12} *es* P
R$_7$: **SI** X_{11} *es* G **y** X_7 *es* P **y** X_2 *es* M **ENTONCES** X_{12} *es* M
R$_8$: **SI** X_{11} *es* G **y** X_7 *es* P **y** X_2 *es* G **ENTONCES** X_{12} *es* M

(c) Base de Reglas del módulo M_3

R$_1$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* M **y** X_{13} *es* P **ENTONCES** Y *es* P
R$_2$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* P **y** X_{13} *es* P **ENTONCES** Y *es* P
R$_3$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* M **y** X_{13} *es* M **ENTONCES** Y *es* P
R$_4$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* P **y** X_{13} *es* G **ENTONCES** Y *es* P
R$_5$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* P **y** X_{13} *es* M **ENTONCES** Y *es* P
R$_6$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* G **y** X_{13} *es* P **ENTONCES** Y *es* P
R$_7$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* M **y** X_{13} *es* P **ENTONCES** Y *es* P
R$_8$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* M **y** X_{13} *es* G **ENTONCES** Y *es* P
R$_9$: **SI** X_4 *es* M **y** X_{12} *es* M **y** X_{16} *es* M **y** X_{13} *es* P **ENTONCES** Y *es* M
R$_{10}$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* G **y** X_{13} *es* M **ENTONCES** Y *es* P
R$_{11}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* M **y** X_{13} *es* M **ENTONCES** Y *es* P
R$_{12}$: **SI** X_4 *es* M **y** X_{12} *es* G **y** X_{16} *es* G **y** X_{13} *es* G **ENTONCES** Y *es* P
R$_{13}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* P **y** X_{13} *es* P **ENTONCES** Y *es* G
R$_{14}$: **SI** X_4 *es* M **y** X_{12} *es* M **y** X_{16} *es* M **y** X_{13} *es* M **ENTONCES** Y *es* M
R$_{15}$: **SI** X_4 *es* M **y** X_{12} *es* P **y** X_{16} *es* M **y** X_{13} *es* M **ENTONCES** Y *es* P
R$_{16}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* G **y** X_{13} *es* G **ENTONCES** Y *es* P
R$_{17}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* G **y** X_{13} *es* P **ENTONCES** Y *es* P
R$_{18}$: **SI** X_4 *es* G **y** X_{12} *es* M **y** X_{16} *es* M **y** X_{13} *es* P **ENTONCES** Y *es* M
R$_{19}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* G **y** X_{13} *es* M **ENTONCES** Y *es* P
R$_{20}$: **SI** X_4 *es* M **y** X_{12} *es* M **y** X_{16} *es* P **y** X_{13} *es* P **ENTONCES** Y *es* M
R$_{21}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* P **y** X_{13} *es* M **ENTONCES** Y *es* M
R$_{22}$: **SI** X_4 *es* G **y** X_{12} *es* G **y** X_{16} *es* M **y** X_{13} *es* G **ENTONCES** Y *es* P
R$_{23}$: **SI** X_4 *es* M **y** X_{12} *es* P **y** X_{16} *es* M **y** X_{13} *es* P **ENTONCES** Y *es* P
R$_{24}$: **SI** X_4 *es* M **y** X_{12} *es* M **y** X_{16} *es* P **y** X_{13} *es* M **ENTONCES** Y *es* M
R$_{25}$: **SI** X_4 *es* M **y** X_{12} *es* M **y** X_{16} *es* G **y** X_{13} *es* M **ENTONCES** Y *es* M
R$_{26}$: **SI** X_4 *es* M **y** X_{12} *es* P **y** X_{16} *es* M **y** X_{13} *es* G **ENTONCES** Y *es* P

(d) Base de Reglas del módulo M_4

Figura 4.24.: Base de Reglas obtenida por SDJHG en el problema *Census 16L* correspondiente a la figura 4.23(a)

en que el número de variables del problema por regla es menor. En el modelizado híbrido que proponemos, se aprecia que se reduce considerablemente el número de variables de entrada en cada módulo con respecto al número de variables usadas por

WM.

Por otro lado, si comparamos el modelizado de sistemas jerárquicos híbridos con el modelizado de sistemas en serie podemos comprobar que generalmente el número de módulos obtenidos suele ser mayor en el modelizado híbrido que en el modelizado en serie. Esto repercute directamente en el número de reglas del sistema, siendo menor en el modelizado híbrido y mejorando así la interpretabilidad del sistema. De esto se benefician problemas como *Concrete*, *Census 8L* o *Census 16H* que experimentan una mejora en la interpretabilidad de sus reglas de un 20 %, 22 % y 25 % respectivamente. Con respecto al número de variables, éste suele ser menor en el modelizado híbrido, lo que permite una mejor interpretabilidad por regla en cada módulo del sistema. Además, observamos que en algunos problemas, tales como *Laser* o *DEE*, la precisión es bastante similar. Esto se debe a que el espacio de búsqueda es más reducido y menos complejo, por lo que el algoritmo SDJHG llega a encontrar soluciones con igual precisión e interpretabilidad que el algoritmo SDJSG. En otros problemas, la precisión de las soluciones obtenidas por el modelizado híbrido disminuye, pero no significativamente. En este caso podemos encontrar problemas como *Treasury* y *Elevators* en donde la precisión empeora en un 4 % y un 6 % respectivamente. También es digno de destacar que el número de módulos es más elevado en el modelizado híbrido, ya que el modelizado en serie establece la limitación de que en cada capa exista solo un módulo. Esto indica claramente que se acarrea menor error por capa en el modelizado en serie, obteniendo soluciones más precisas pero menos interpretables.

El modelizado paralelo con respecto al modelizado híbrido que proponemos, obtiene menor número de módulos, más reglas y en sus soluciones se usa un mayor número de variables del problema. Esto conlleva una peor interpretabilidad a nivel de regla, ya que cada regla se compone de un mayor número de variables. El número de módulos obtenido por el modelizado paralelo es menor que el número de módulos obtenido por el modelizado híbrido. Este dato nos llevaría a pensar que existiría un menor acarreo de errores a través de las capas. A pesar de que esta afirmación es verdadera, en el modelizado híbrido se cuenta con un modelizado libre de jerarquía, en donde cada capa puede estar formada por variables exógenas y endógenas. En el caso del modelizado paralelo, se establece la restricción de que no pueden existir variables exógenas en las capas intermedias del sistema paralelo. A la vista de los resultados, nos queda claro que la existencia de variables exógenas en las capas intermedias es bastante beneficioso para el sistema jerárquico híbrido, ya que la variable exógena

aporta información sin ruido que hace que disminuya el error producido en la variable de salida inferida por el módulo, mejorando así la precisión del sistema híbrido con respecto al paralelo.

Dado que el modelizado jerárquico híbrido combina dos modelizados (serie y paralelo), el espacio de búsqueda que tiene que cubrir el algoritmo es mucho mayor, por lo que le es más difícil encontrar soluciones prometedoras.

La figura 4.25 muestra el promedio del conjunto Pareto en los problemas *Census 16H*, *Census 16L*, *Computer Activity* y *Ailerons*. En la figura podemos observar que la precisión obtenida por el algoritmo SDJHG con respecto al algoritmo de selección de variables SVAG es mejor a medida que el número de reglas se incrementa. También se puede apreciar cómo el número de variables por regla (métrica $\overline{x}_{\#R}$) obtenido por SDJHG es menor que en SVAG, indicando así que las reglas generadas por nuestra propuesta son más simples y, consecuentemente, más interpretables que las generadas por el algoritmo de selección de variables.

Al igual que en las propuestas en serie y paralelo, hemos elaborado unas gráficas de ranking en donde añadimos nuestra propuesta híbrida denominada *SDJHG_M*, en el caso de la solución más precisa, y *SDJHG_C1*, para representar el primer cuartil, quedando latente en las figuras 4.26 y 4.27.

Como podemos apreciar, nuestra propuesta en serie es más precisa que la híbrida, así se manifiesta en todos los problemas, y en más del 50 % de los problemas también proporciona mejor interpretabilidad. Si comparamos nuestra propuesta híbrida con la paralela, podemos observar que la paralela presenta una mayor interpretabilidad, pero la híbrida es más precisa. Queda claro que SDJHG ocupa el segundo lugar si consideramos un ranking con estos tres algoritmos. Podemos esperar este comportamiento ya que el algoritmo SDJHG evoluciona estructuras en serie y en paralelo simultáneamente. WM sigue ocupando las últimas posiciones con respecto a ambos criterios. JS proporciona soluciones más interpretables pero menos precisas con respecto a nuestra propuesta híbrida. Si atendemos a los resultados de SVAG, podemos comprobar que en la mayoría de los casos se encuentra entre la solución más precisa y el primer cuartil proporcionado por SDJHG.

Las tablas 4.18 y 4.19 contienen las posiciones numéricas de cada método representadas en las figuras 4.26 y 4.27 atendiendo a un determinado problema y su correspondiente ranking medio. Con respecto al criterio de ordenación ascendente del ECM_{tra}, podemos comprobar que la jerarquía híbrida se encuentra situada entre nuestra propuesta de jerarquía en serie y paralela. El resto de algoritmos con los que

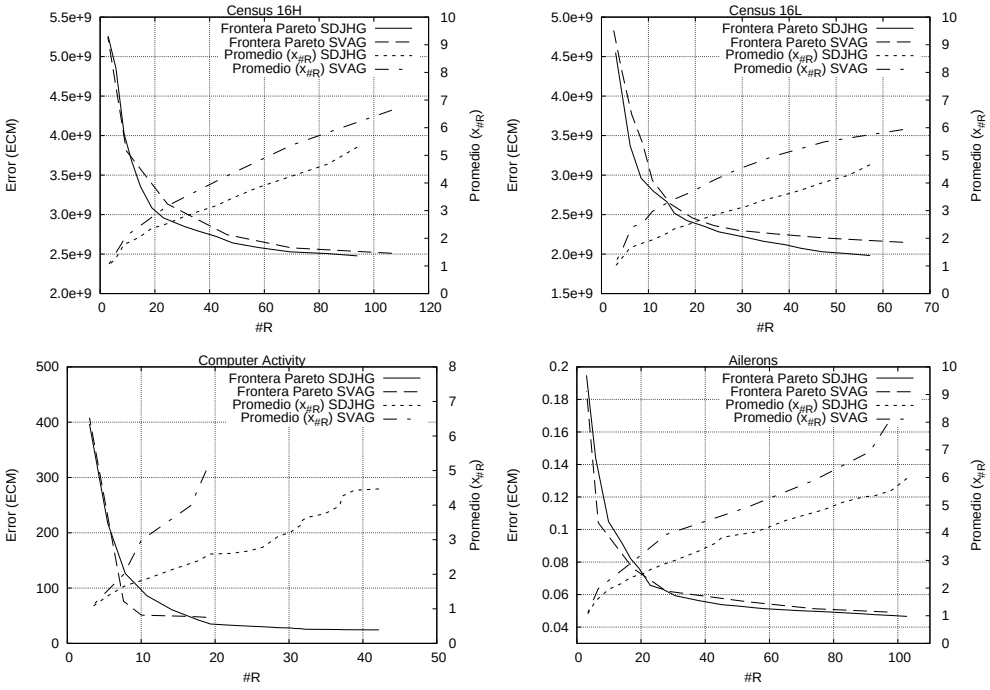


Figura 4.25.: Promedio de la frontera Pareto en varios problemas para el algoritmo SDJHG

comparamos están a la cola del ranking según el ranking medio. Atendiendo al número de reglas, el primer cuartil de nuestras tres propuestas se encuentra entre las primeras posiciones, pero, evidentemente, pierde precisión.

En esta propuesta de modelo jerárquico también hemos realizado una experimentación modificando el criterio de dominancia, para así obtener soluciones con una, dos y tres capas y observar el comportamiento que tiene una estructura jerárquica híbrida. La figura 4.28 muestra el promedio de las fronteras del Pareto con uno, dos y tres capas obtenidas en los problemas *Census 16H*, *Census 16L*, *Computer Activity* y *Ailerons* por el modelizado híbrido. La leyenda de la gráfica podemos apreciar el promedio del tamaño de cada Pareto. En la figura se puede observar que el modelo de dos y tres capas suele aportar mayor número de soluciones precisas que el modelo de una sola capa. Además esto se puede apreciar en el tamaño medio de conjunto Pareto.

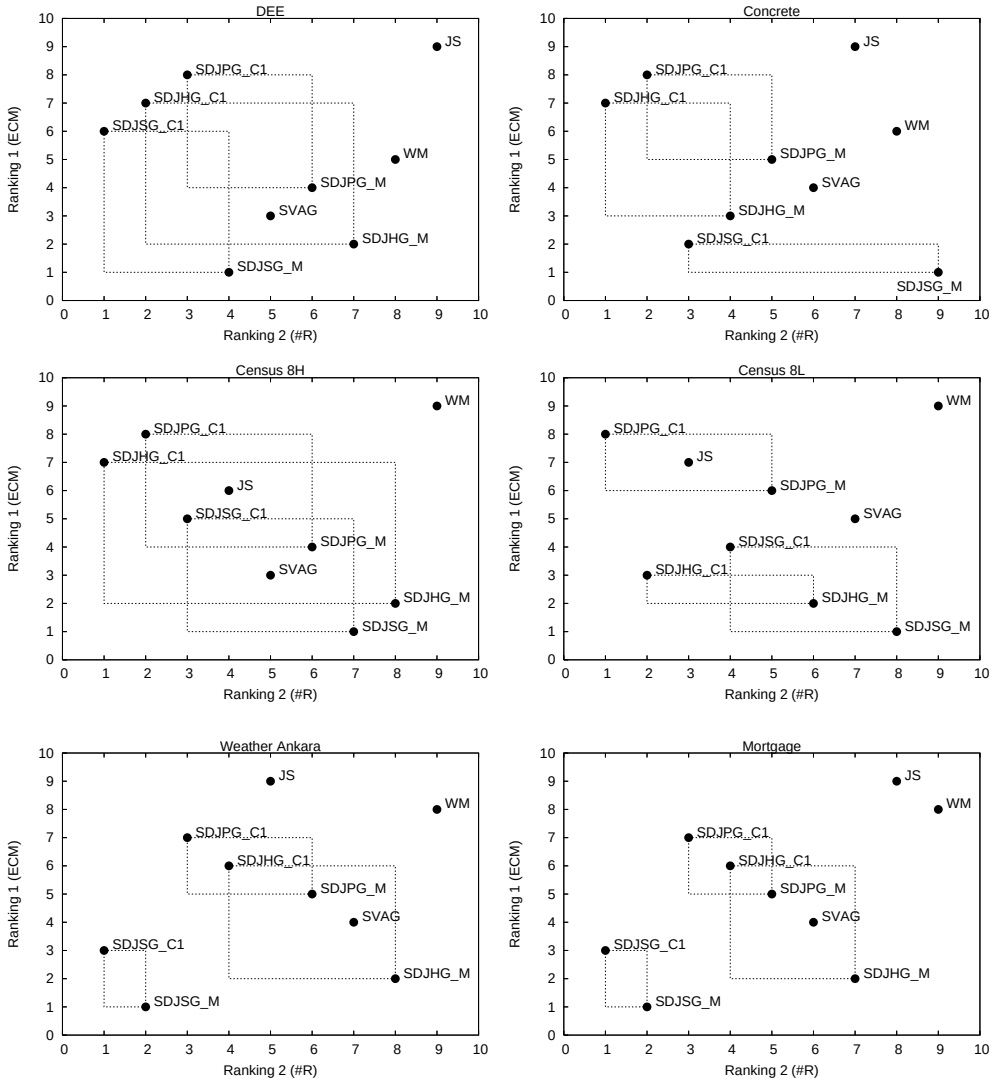


Figura 4.26.: Ranking de algoritmos en varios problemas para el algoritmo SDJHG

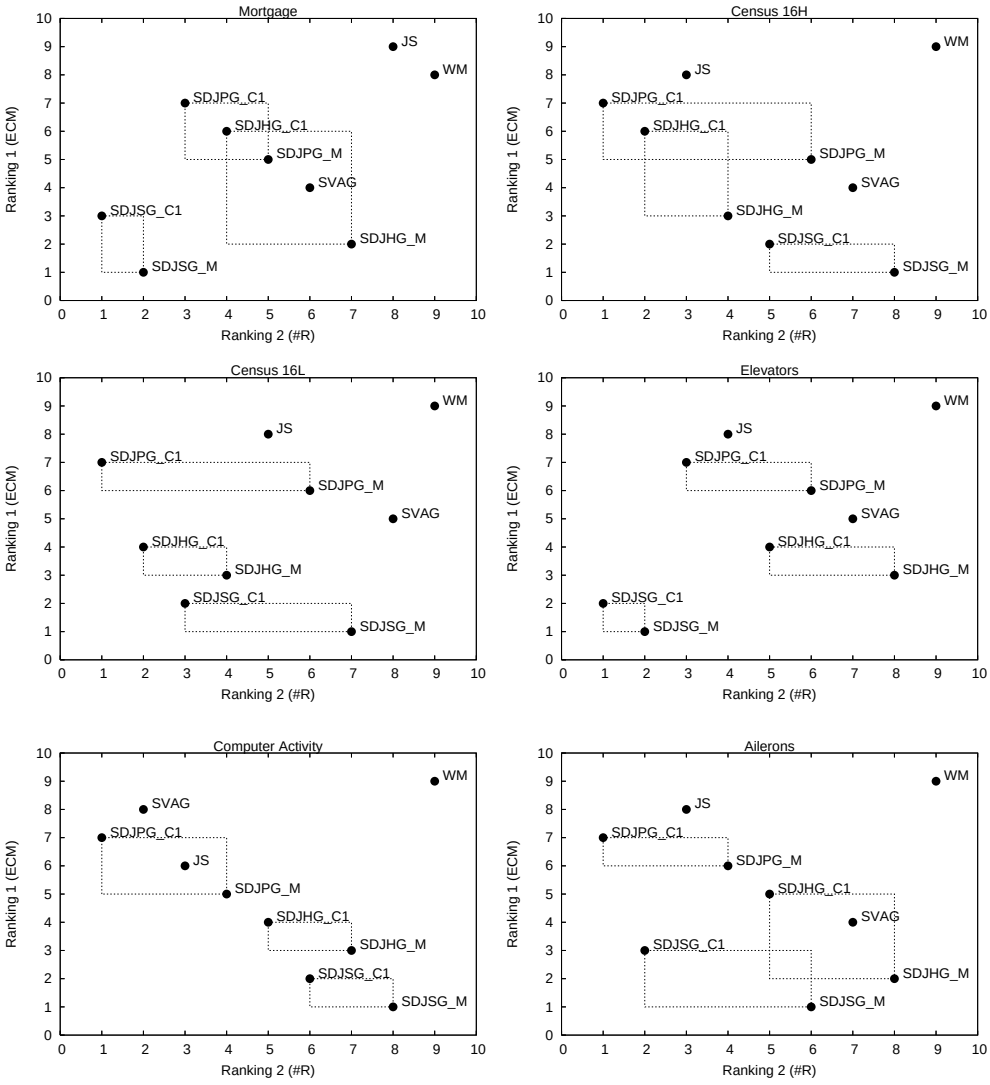


Figura 4.27.: Ranking de algoritmos en varios problemas para el algoritmo SDJHG

Tabla 4.18.: Ranking de algoritmos considerando como criterio ECM_{tra}

Método / Problema	*WM*	*SVAG*	*JS*	*SDJSG_M*	*SDJSG_C1*	*SDJPG_M*	*SDJPG_C1*	*SDJHG_M*	*SDJHG_C1*
DEE	5	3	9	1	6	4	8	2	7
Concrete	6	4	9	1	2	5	8	3	7
Census 8H	9	3	6	1	5	4	8	2	7
Census 8L	9	5	7	1	4	6	8	2	3
Weather Ankara	8	4	9	1	3	5	7	2	6
Mortgage	8	4	9	1	3	5	7	2	6
Treasury	9	5	8	1	4	3	7	2	6
Elevators	9	5	8	1	2	6	7	3	4
Computer Activity	9	8	6	1	2	5	7	3	4
Ailerons	9	4	8	1	3	6	7	2	5
Ranking medio	8	4	8	1	3	5	7	2	5

Tabla 4.19.: Ranking de algoritmos considerando como criterio el número de reglas

Método / Problema	*WM*	*SVAG*	*JS*	*SDJSG_M*	*SDJSG_C1*	*SDJPG_M*	*SDJPG_C1*	*SDJHG_M*	*SDJHG_C1*
DEE	8	5	9	4	1	6	3	7	2
Concrete	8	6	7	9	3	5	2	4	1
Census 8H	9	5	4	7	3	6	2	8	1
Census 8L	9	7	3	8	4	5	1	6	2
Weather Ankara	9	7	5	2	1	6	3	8	4
Mortgage	9	6	8	2	1	5	3	7	4
Treasury	9	3	7	4	2	5	1	8	6
Elevators	9	7	4	2	1	6	3	8	5
Computer Activity	9	2	3	8	6	4	1	7	5
Ailerons	9	7	3	6	2	4	1	8	5
Ranking medio	9	5	5	5	2	5	2	7	3

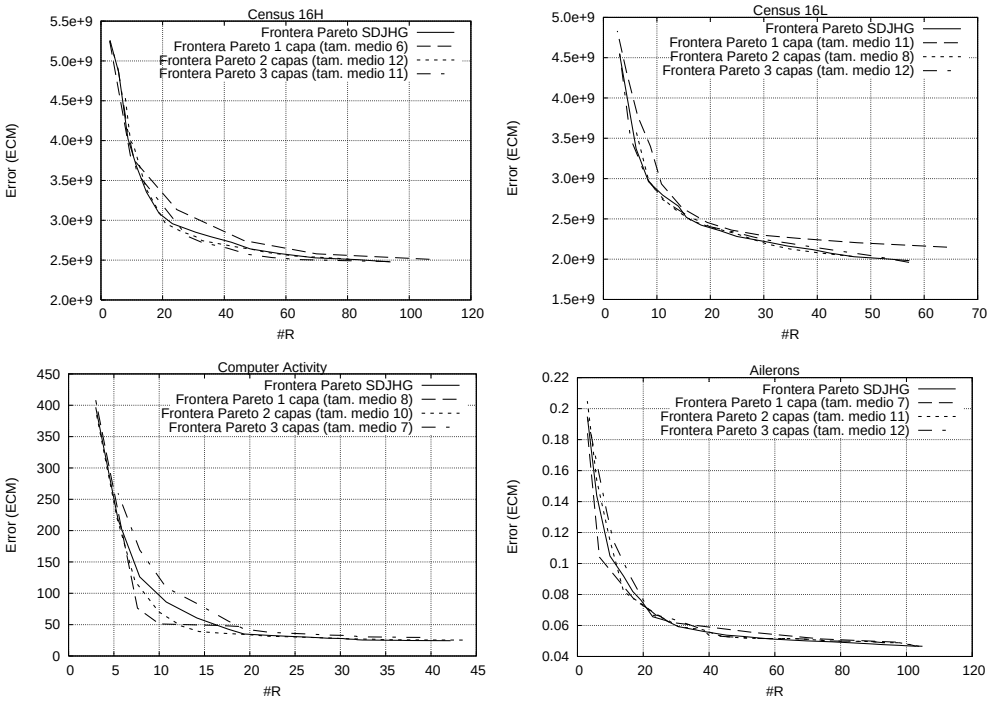


Figura 4.28.: Promedio de la frontera del Pareto con una, dos y tres capas en varios problemas para el algoritmo SDJHG

4.3. Sumario

En este capítulo hemos presentado y descrito dos métodos de aprendizaje de sistemas jerárquicos con el objetivo de obtener soluciones equilibradas en precisión e interpretabilidad. Para ello, los métodos constan de una serie de operadores genéticos que exploran y explotan el espacio de búsqueda. Atendiendo a los resultados, hemos obtenido las siguientes conclusiones:

- Un sistema jerárquico paralelo es un sistema cerrado: sólo recibe variables exógenas del problema en los nodos hoja del árbol que representa la jerarquía. Esto promueve la generación de errores y la propagación de ellos a través de las capas.

- La búsqueda de soluciones es compleja para el modelizado jerárquico híbrido

debido a que el espacio de búsqueda es mayor por combinar el modelizado de estructuras jerárquicas en serie y en paralelo.

- Las variables exógenas en capas intermedias de un sistema jerárquico aportan información sin ruido al sistema que disminuye el error acarreado a través de las capas, lo que hace que el modelizado de sistemas jerárquicos en serie sea más preciso.

5. Aplicación del algoritmo SDJSG-Ajuste en Astrofísica

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica

ARISTÓTELES (384 A. C.–322 A. C.), FILÓSOFO, LÓGICO Y CIENTÍFICO DE LA ANTIGUA GRECIA

En los dos capítulos anteriores hemos descrito tres algoritmos que aprenden distintos tipos de estructuras jerárquicas difusas para ganar interpretabilidad intentando mantener o mejorar la precisión del sistema. El aprendizaje de la estructura jerárquica es posible gracias a dos componentes en su diseño:

- La multi-objetividad, que permite el equilibrio entre la precisión y la interpretabilidad.
- Los operadores genéticos, por incorporar mecanismos que incluyen un determinado grado de libertad en la elaboración de las estructuras jerárquicas difusas en cada algoritmo.

Cabe señalar que, las estructuras jerárquicas paralelas e híbridas presentan algunas deficiencias debido a su morfología, que se refieren principalmente al número de módulos y el acarreo del error en la inferencia a lo largo de las capas de la estructura jerárquica. Por otro lado, el modelizado de estructuras jerárquicas en serie presenta un comportamiento más adecuado en la resolución de problemas por la limitación que establecen en el número de módulos por capa, en este caso uno, y la incorporación de variables exógenas en capas intermedias de la estructura. Esto lo convierte en

un sistema abierto en donde se minimiza el error de inferencia del módulo ya que las variables exógenas aportan información sin ruido.

En este sentido, los resultados del modelizado en serie nos animan a aplicarlo a otros problemas reales distintos al conjunto de datos empleados como bancos de pruebas en los capítulos anteriores. De esta forma, podremos comprobar la capacidad y las prestaciones que puede ofrecer el algoritmo. Para ello, realizamos una experimentación con datos fotométricos astrofísicos referentes a conjuntos de datos reales de espectros de galaxias. Se trata de una aplicación original que no se ha abordado anteriormente con aprendizaje automático y mucho menos con sistemas difusos.

En las siguientes secciones, describiremos el conjunto de datos astrofísicos en donde vamos a aplicar el modelo en serie y cómo se obtuvieron. A continuación, explicaremos un método bayesiano de ajuste espectral con el objetivo de comparar los resultados obtenidos por el modelo en serie. Finalizaremos el capítulo con un estudio experimental y algunas conclusiones.

5.1. Introducción a la Astronomía Extragaláctica

La Astrofísica es una ciencia que tiene como objetivo estudiar el origen, la formación y la evolución de los planetas, estrellas y galaxias, así como explicar los fenómenos del Universo, basándose en otras ciencias como la Física o las Matemáticas. Civilizaciones tales como griegos, egipcios, chinos o japoneses estuvieron interesadas por la observación del cielo y los fenómenos celestes, por lo que se considera una de las ciencias más antiguas.

Gracias al estudio de esta ciencia se han desarrollado nuevas tecnologías, tales como telescopios, satélites de observación, super-computadores, GPS, etc., que tienen aplicación directa en nuestras vidas y que además ayudan a resolver las cuestiones sobre el universo.

La Astronomía Extragaláctica es la disciplina astronómica que estudia los objetos que se encuentran fuera de la Vía Láctea, por ejemplo, galaxias, estrellas o cuásares. Principalmente analiza los movimientos de las galaxias y sus agrupaciones, las interacciones entre galaxias, el medio intergaláctico, la evolución de las galaxias y la estructura del universo.

En esta disciplina, se suelen mapear regiones del cielo usando detectores, sensibles a diferentes anchos de banda. Esta tarea recibe el nombre de *survey*. El objetivo de un *survey* es el de clasificar las galaxias en función de su morfología, descubrir

sus estructuras internas y su evolución con el tiempo. Los *surveys* se recogen en un catálogo astronómico. En el campo de los *surveys* actuales, destaca *Sloan Digital Sky Survey* (SDSS) (York y cols. (2000)) como el más productivo en cantidad de datos y publicaciones de los últimos diez años. El survey SDSS [6] consta de un conjunto de mapas tridimensionales detallados del Universo, con imágenes en múltiples colores de un tercio del cielo y los espectros de más de tres millones de objetos astronómicos destacando sus más de 930.000 galaxias y 120.000 cuásares, sin contar estrellas, cometas, aglomerados y otras observaciones indirectas.

Los datos de este survey se han obtenido gracias al telescopio SDSS (York y cols. (2000); Gunn y cols. (2006)) situado en el Observatorio de Apache Point, Nuevo Méjico (EE.UU.). La figura 5.1 muestra una imagen del observatorio. A la izquierda de la imagen se puede apreciar el telescopio.

Figura 5.1.: Observatorio de Apache Point (Nuevo Méjico, EE.UU.). Telescopio SDSS 2.5m a la izquierda [7]

[6] SDSS, http://www.sdss.org/

[7] Observatorio de Apache Point (Nuevo Mexico, EE.UU.), http://www.sdss3.org/instruments/

5.2. Starlight

Los espectros de las galaxias codifican la información de las estrellas que las forman, nos informan sobre cuándo se formaron las estrellas y nos especifica la química de las galaxias. Por ello, la información que se recaba durante las observaciones es esencial para comprender la formación y evolución de las galaxias. El trabajo observacional y teórico se ha traducido en la producción de gran cantidad de conjuntos de datos de espectros de estrellas de alta calidad.

Como describimos en la sección anterior, el survey SDSS proporciona una base de datos homogénea de cientos de miles de espectros de galaxias. Esta enorme cantidad de datos de alta precisión será la que nos permita entender la constitución, la formación y la evolución de las galaxias.

Para analizar conjuntos de datos, tales como el survey SDSS, se debe establecer una metodología de traducción de los espectros observados a las propiedades físicas de las galaxias.

Starlight (Cid-Fernandes y cols. (2005), Cid-Fernandes y cols. (2010)) es un software que computa la síntesis espectral de grandes cantidades de datos obtenidos mediante espectroscopía óptica, es decir, de los datos obtenidos del flujo de energía recibido de los objetos celestes respecto a la longitud de onda. En pocas palabras, Starlight procesa los datos observados (espectros) en un espacio de propiedades físicas, como por ejemplo, masas estelares, edad, metalicidades medias, extinción estelar, dispersión de velocidades estelares y un histórico de la formación estelar. El software está disponible online[8].

5.3. J-PAS

El proyecto J-PAS (Javalambre Physics of the Accelerating Universe Astrophysical Survey[9]) es una colaboración hispano-brasileña que tiene como objetivo el cartografiado del Universo visible. Para ello, consta de un telescopio dedicado, con un espejo de 2,5m de diámetro y un gran campo de visión y una cámara de 1,2 Gigapíxeles formada por 14 CCDs. Este instrumento, situado en Teruel, producirá imágenes de calidad y un espectro de baja resolución sobre todo el área del cartografiado para así proyectar el Cosmos. El proyecto comienza en 2.015, en donde se observarán más

[8] Starlight, http://www.starlight.ufsc.br/

de 8.500 grados cuadrados (aproximadamente una quinta parte de todo el cielo) durante cinco o seis años. Con esta información, además, se construirá el primer mapa en 3D del Universo con el cual se podrán buscar evidencias de la acción de la materia y energía oscuras a gran escala.

5.4. Método de Ajuste J-Espectral

En los datos obtenidos por el proyecto J-PAS también podemos encontrar información para estudiar las propiedades físicas de las galaxias, tanto de sus estrellas como de su gas. Una de las formas de tratar los datos sería con el software Starlight, es decir, mediante el uso de modelos de síntesis evolutiva de poblaciones estelares para inferir la Historia de la Formación Estelar a partir de ajustes espectrales y medir las líneas de emisión. Hay que destacar que si se realizan este tipo de ajustes, tenemos que hacer frente a una serie de limitaciones en la resolución espectral de J-PAS. Si se usa Starlight para procesar los datos obtenidos por el proyecto J-PAS, se producirá una pérdida de información que se traducirá en una falta de precisión en la estimación de las propiedades de las galaxias.

Como alternativa, Schoenell y cols. en 2013 proponen el método Ajuste J-Espectral que proporciona la misma cantidad de información que un método de modelos similar. Las galaxias componen un conjunto de características que determinan una propiedad física. Estas características se denominan J-Espectros. En este trabajo, para la comparativa asumen que si dos galaxias tienen igual J-Espectro, estas poseen las mismas propiedades físicas. Para obtener las propiedades físicas, el método calcula para 100.000 objetos y dos muestras de comparación sus funciones de probabilidad atendiendo a un factor de escala y un criterio de convergencia hasta que se alcanza 10^{-5}. El método ajusta las funciones de probabilidad de cada propiedad física al ancho de una función de distribución de probabilidad para minimizar errores. Para la comparación de los resultados, como aún no se han obtenido datos del proyecto J-PAS, estos datos se simulan extrayendo la información de los espectros de SDSS. La figura 5.2 muestra un esquema gráfico de este método.

[9] Proyecto J-PAS, http://j-pas.org

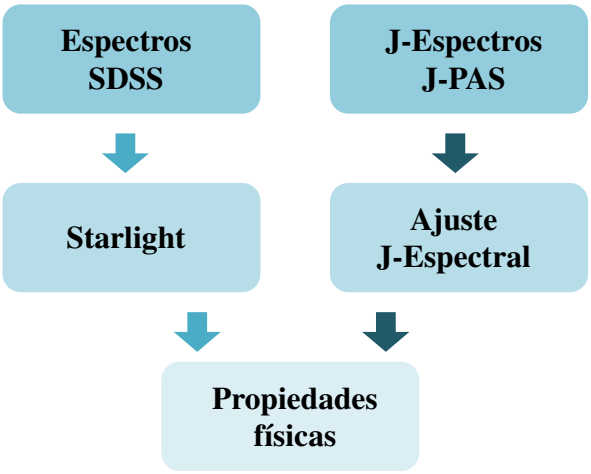


Figura 5.2.: Esquema gráfico del método Ajuste J-Espectral

5.5. Experimentos

5.5.1. Problemas

La experimentación se centra en obtener un conjunto de propiedades físicas (variables de salida del conjunto de datos) de interés en el campo de la Astrofísica a partir de un conjunto de J-Espectros (variables de entrada del conjunto de datos). De la gran cantidad de información que se puede extraer del conjunto de datos, consideraremos cuatro propiedades físicas:

- *Masa Total:* Esta propiedad hace referencia a la masa actual de la población de estrellas de la(s) galaxia(s).
- *Metalicidad:* Identifica la proporción de la materia que compone los elementos químicos en estrellas u otros tipos de objetos astronómicos, excluyendo su hidrógeno y helio. La metalicidad en las estrellas y otros objetos astronómicos permite estimar de forma aproximada las abundancias químicas que cambian con el tiempo mediante los mecanismos de la evolución estelar, por lo que puede proporcionar una indicación de la edad.
- *Extinción Estelar:* Representa un espectro de la radiación electromagnética de

una fuente de radiación cuando las características emitidas son diferentes a las originales. Este cambio se produce debido a la dispersión de polvo de luz y otras materias en el medio interestelar.

- *Edad:* Esta propiedad, como su propio nombre indica, hace referencia a la edad de las poblaciones estelares de la galaxia.

Los datos provienen del SSDS y han sido previamente procesados por Starlight. El conjunto de datos para todas las propiedades se compone de datos de entrenamiento y prueba. La tabla 5.1 recoge las principales características de cada propiedad, donde *#VarEntrada* es sinónimo del número de variables de entrada, *#Ejem* identifica el número total de instancias, *#TermLing (E/S)* indica el número de términos lingüísticos triangulares fuertes de entrada/salida considerados y *Disponible* referencia el sitio web en donde el conjunto de datos está disponible.

En la experimentación, establecemos cinco etiquetas por variable de entrada y siete etiquetas para la variable de salida con el objeto de obtener mayor precisión en el sistema obtenido.

Tabla 5.1.: Conjunto de datos astrofísicos considerados en el análisis experimental

Conjunto de datos	*#Ejem*	*#VarEntrada*	*#TermLing (E/S)*	*Disponible*
Masa Total Metalicidad Extinción Estelar Edad	4.000	45	5/7	STARLIGHT[10]

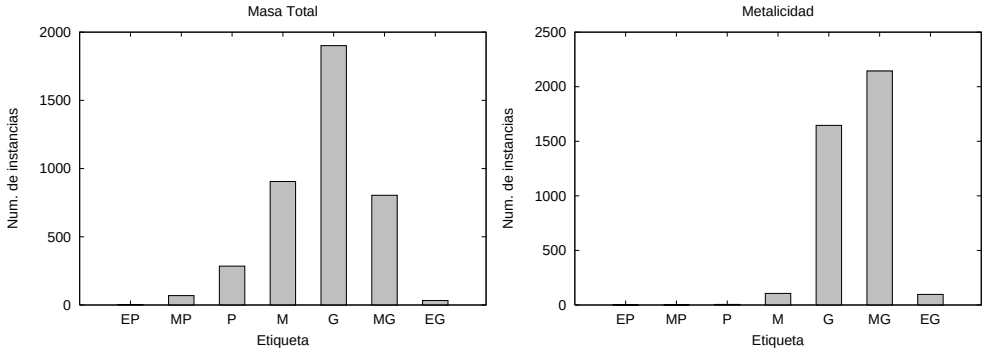
Las figuras 5.3 y 5.4 muestran la distribución de las instancias de estas propiedades físicas en cada una de las etiquetas. Estas figuras permiten hacernos una idea de la distribución de los datos para comprobar dónde se concentra la mayoría de las instancias de las variables de salida.

5.5.2. Funciones Objetivo

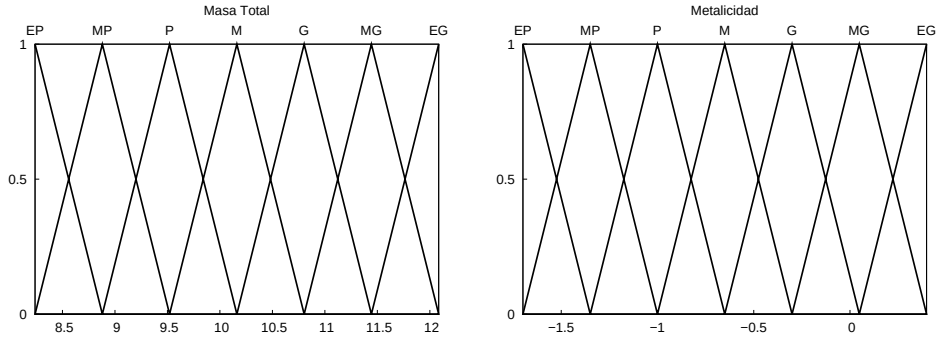
En esta experimentación vamos a considerar dos funciones objetivo a minimizar:

- *Raíz del Error Cuadrático Medio (RECM):* Si el RECM es bajo, la precisión

[10]STARLIGHT: Spectral Synthesis Code. http://astro.ufsc.br/starlight/node/3



(a) Distribución de instancias de la Masa Total (b) Distribución de instancias de la Metalicidad



(c) Funciones de pertenencia de la Masa Total (d) Funciones de pertenencia de la Metalicidad

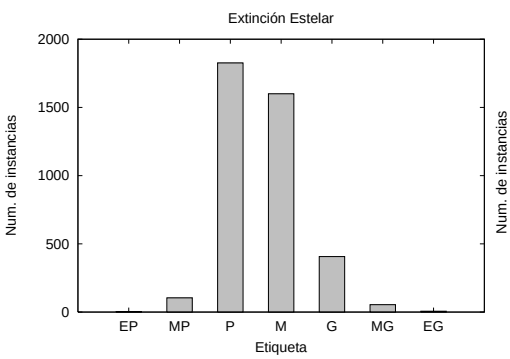
Figura 5.3.: Distribución del valor de las instancias de las propiedades físicas por etiqueta

es buena. Se calcula como:

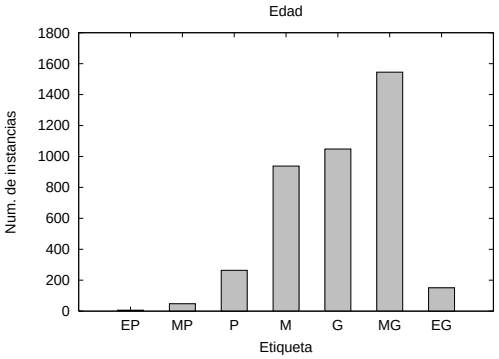
$$F_1(S) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S(x^i) - y^i)^2} \tag{5.1}$$

Siendo S el SD a evaluar, N el tamaño del conjunto de datos y (x^i, y^i) la i-ésima pareja entrada-salida del conjunto de datos. El objetivo, en nuestro caso, es minimizar esta función para mejorar la precisión.

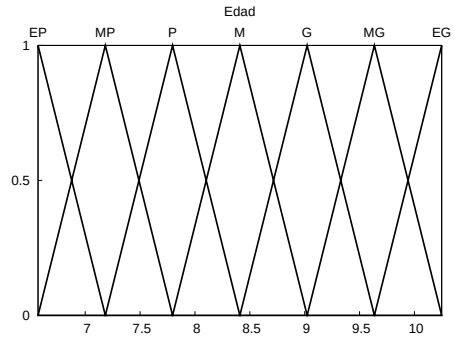
- *Máximo número de reglas:* El sistema será más interpretable con un número máximo de reglas bajo (en la subsección).



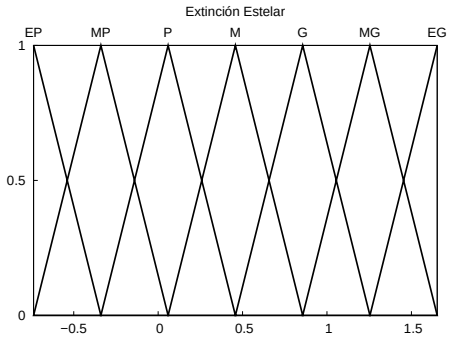
(a) Distribución de instancias de la Extinción Estelar



(b) Distribución de instancias de la Edad



(c) Funciones de pertenencia de la Edad



(d) Funciones de pertenencia de la Extinción Estelar

Figura 5.4.: Distribución del valor de las instancias de las propiedades físicas por etiqueta

5.5.3. Configuración Experimental

Para obtener las propiedades que mencionamos en la Subsección 5.5.1, hemos hecho uso del algoritmo SDJSG con ajuste (SDJSG-Ajuste) de las funciones de pertenencia. Este algoritmo, además de proporcionar una buena interpretabilidad, que nos permitirá conocer los J-Espectros relacionados con cada propiedad, obtendrá una mayor precisión gracias al ajuste de las funciones de pertenencia. El algoritmo SDJSG-Ajuste se ha ejecutado con los siguientes parámetros de configuración: 50.000 evaluaciones, un tamaño de población de 60 individuos, una probabilidad de

cruce de 0,7, una probabilidad de mutación de 0,2 por cromosoma y el parámetro α del cruce BLX se ha inicializado a 0,3.

Como se comentó en la sección 5.5.1, la experimentación se compone de un conjuntos de datos de entrenamiento y prueba. Hemos usado validación cruzada con cinco particiones del conjunto de datos y seis ejecuciones por partición. Esto hace un total de 30 ejecuciones por propiedad física.

5.6. Estudio Experimental

5.6.1. Resultados obtenidos

La tabla 5.2 recoge los resultados obtenidos por el algoritmo SDJSG-Ajuste, donde $RECM_{entr}$ y $RECM_{prue}$ son los valores de la raíz del ECM de entrenamiento y prueba respectivamente para un conjunto de datos; *#M*, *#R* y *#V* hacen referencia al número de módulos, el número de reglas difusas y el número de variables de entrada respectivamente; $\sigma_{\#M}$ es la desviación típica del número de módulos y $\bar{x}_{\#R}$ es el número de variables por regla. En la tabla se muestran cinco soluciones representativas de la media del Pareto ordenadas de menor a mayor $RECM_{entr}$: la primera fila de cada propiedad física es la solución más precisa, la segunda fila es el primer cuartil, la tercera fila es la mediana, la cuarta fila es el tercer cuartil y la quinta fila es la solución menos precisa.

Establecemos el número máximo de módulos a cuatro para que la propagación del error sea menor y reducir el espacio de búsqueda. Por otro lado, consideramos un espacio de búsqueda mayor a lo habitual debido al incremento del número de etiquetas en las variables de entrada y salida.

La tabla anteriormente mencionada, contiene una fila denominada *Ajuste J-Esp* que hace referencia al resultado obtenido por el algoritmo Ajuste J-Espectral. Debido a que esta propuesta es un método bayesiano (método numérico), no dispone de datos de entrenamiento. En este caso, solo podremos realizar comparaciones entre los datos de prueba.

Un análisis del algoritmo SDJSG-Ajuste aplicado a esta casuística se realizará en profundidad en la subsección 5.6.2 para comprobar el comportamiento y beneficios de nuestro algoritmo en problemas astrofísicos reales. Previamente, analizaremos los resultados obtenidos por SDJSG-Ajuste y los compararemos con los resultados obtenidos con otra metodología usada en astrofísica, el algoritmo *Ajuste J-Esp*.

Tabla 5.2.: Resultados obtenidos por SDJSG-Ajuste en distintas propiedades físicas, siendo $RECM_{entr}$ y $RECM_{pru}$ la raíz de los valores de error de aproximación en los datos de entrenamiento y prueba; *#M*, *#R* y *#V* son el n° de módulos, reglas difusas y variables de entrada; $\sigma_{\#M}$ es la desviación típica del n° de módulos y $\bar{x}_{\#R}$ es el promedio del n° de variables por regla

Método	$RECM_{entr}$	$RECM_{pru}$	$\#M \pm \sigma_{\#M}$	*#R*	$\bar{x}_{\#R}$	*#V*
MASA TOTAL						
Ajuste J-Esp		0,301000				45,0
SDJSG-Ajuste	0,332392	0,337660	2,5±0,8	68,8	4,2	6,7
	0,340594	0,345836	2,1±0,7	43,7	3,2	5,0
	0,351167	0,355827	1,5±0,6	24,8	2,6	3,4
	0,367165	0,370638	1,2±0,3	14,4	2,0	2,3
	0,401045	0,404027	1,0±0,0	4,8	1,0	1,0
EXTINCIÓN ESTELAR						
Ajuste J-Esp		0,106000				45,0
SDJSG-Ajuste	0,136841	0,139551	2,3±0,9	43,4	3,0	5,3
	0,145147	0,147178	1,8±0,7	27,0	2,3	3,7
	0,163975	0,165311	1,4±0,5	15,7	1,9	2,3
	0,208664	0,208435	1,4±0,5	9,8	1,4	1,8
	0,245338	0,246577	1,0±0,0	4,8	1,0	1,0
METALICIDAD						
Ajuste J-Esp		0,144000				45,0
SDJSG-Ajuste	0,126427	0,133453	2,8±0,9	100,6	5,1	7,7
	0,129158	0,133856	2,5±0,7	50,4	3,2	5,3
	0,132925	0,136253	2,3±0,5	29,6	2,2	4,1
	0,137244	0,138752	2,5±0,8	15,0	1,4	3,1
	0,142128	0,142996	1,0±0,0	4,8	1,0	1,0
EDAD						
Ajuste J-Esp		0,199000				45,0
SDJSG-Ajuste	0,200072	0,204985	1,8±1,0	25,6	2,4	3,5
	0,209782	0,214554	1,6±0,7	16,7	1,9	2,7
	0,220307	0,223034	1,4±0,5	10,0	1,6	2,0
	0,234810	0,236899	1,3±0,4	7,4	1,2	1,5
	0,248780	0,249957	1,0±0,0	4,8	1,0	1,0

5.6.2. Análisis

Como hemos mencionado anteriormente, el algoritmo Ajuste J-Espectral usa una metodología numérica, por lo que la política de manejo de datos no está basada en el aprendizaje de un conjunto de datos y sus posteriores pruebas, que es el comportamiento que tiene nuestra propuesta. Por ello, compararemos el $RECM_{pru}$ de los modelos obtenidos por nuestra propuesta con los resultados obtenidos por el algoritmo Ajuste J-Espectral.

Si analizamos los resultados obtenidos, el Ajuste J-Espectral mejora en términos de precisión a nuestra propuesta en dos propiedades físicas analizadas (*Masa Total* y *Extinción Estelar*), mientras que nuestra propuesta mejora al Ajuste J-Espectral en un 7 %. En *Edad* y *Metalicidad*, ambas propuestas obtienen tasas de precisión similares.

Por otro lado, podemos comprobar en la tabla de resultados que el número de variables que usa nuestra propuesta es considerablemente menor, concretamente SDJSG-Ajuste reduce un 85 %, 88 %, 83 % y 92 % para la *Masa Total*, *Extinción Estelar*, *Metalicidad* y *Edad* respectivamente con respecto al total de variables del problema. Claramente, nuestra propuesta reduce la complejidad del problema sin que la pérdida de precisión se vea afectada significativamente.

Como hemos comentado en anteriores capítulos, nuestro objetivo no se centra en mejorar la precisión, sino en implementar un sistema más interpretable, de forma que el modelo que obtenga nuestro algoritmo proporcione el conocimiento contenido en el conjunto de datos. Desde este punto de vista, el Ajuste J-Espectral es un método opaco, es decir, no indica la relación existente entre las variables que analiza y consecuentemente, es muy difícil encontrarles un significado. Sin embargo, SDJSG-Ajuste genera modelos de estructuras difusas jerárquicas que relaciona variables y donde cada módulo contiene un conjunto de reglas que permiten interpretar los datos.

Los modelos difusos jerárquicos obtenidos por nuestra propuesta se caracterizan por obtener modelos cuya estructura jerárquica ronda las dos capas, lo que conlleva a que el error acarreado a lo largo de las capas no sea elevado y no afecte demasiado a la precisión. También se puede apreciar que en las soluciones obtenidas el número de reglas es bastante aceptable, lo cual, junto con un número reducido de variables por regla, ofrece modelos con un nivel de interpretabilidad adecuado.

La figura 5.5 muestra el promedio de la frontera del Pareto obtenido por SDJSG-Ajuste en las cuatro propiedades físicas que estamos considerando. Como se aprecia en las representaciones gráficas, un menor número de reglas implica mayor error en la precisión y viceversa.

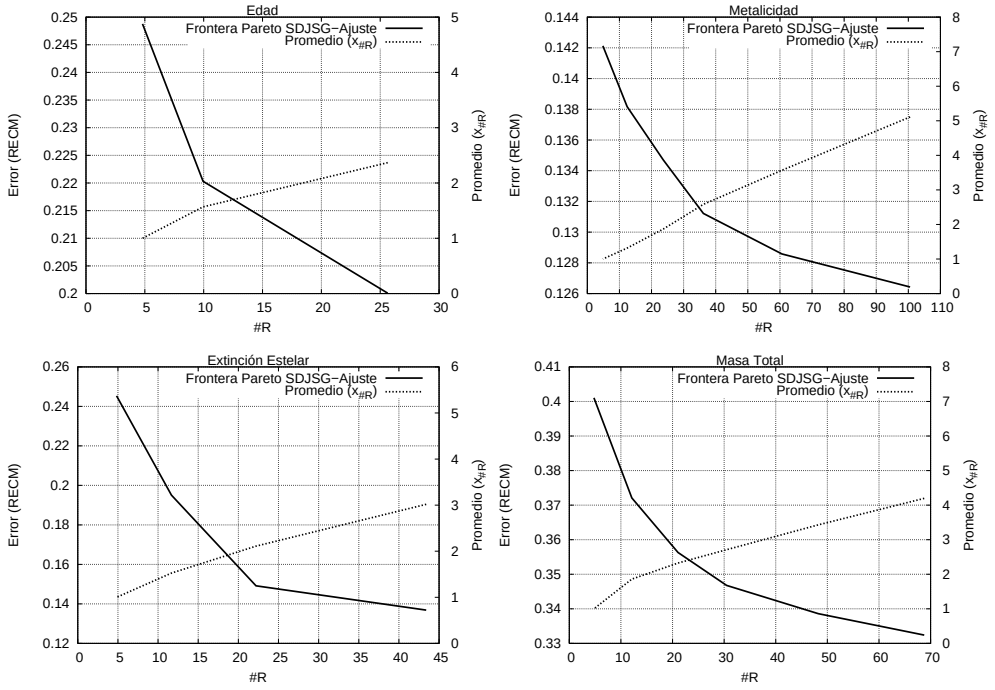


Figura 5.5.: Promedio de la Frontera Pareto de las propiedades físicas

La figura 5.6 muestra un gráfico de barras que representa la tasa de dependencia de variables desde la solución más precisa hasta las soluciones del primer cuartil para los problemas astrofísicos que estamos considerando. Como ya comentamos anteriormente, la tasa de dependencia de variables viene dada por el número de variables diferentes que depende de otra(s) variable(s) al menos en una de las soluciones más precisas no dominadas (las contenidas en el primer cuartil). La tasa ha sido normalizada al porcentaje. En el gráfico de barras podemos observar que las dos propiedades físicas con mayor tasa de dependencia son *Metalicidad* y *Masa Total*. A pesar de ser las propiedades físicas con mayor dependencia, no superan una tasa del 6 % con respecto al total de variables de las que se dispone en cada problema. Esto nos indica que las variables están menos correladas, lo que entraña una dificultad alta en el aprendizaje de las variables. En este caso, cada una de las variables que intervienen en cada una de las propiedades físicas deberían ser analizadas en profundidad en

términos de regresión. Nuestra propuesta puede ayudar a entender la complejidad de cada problema.

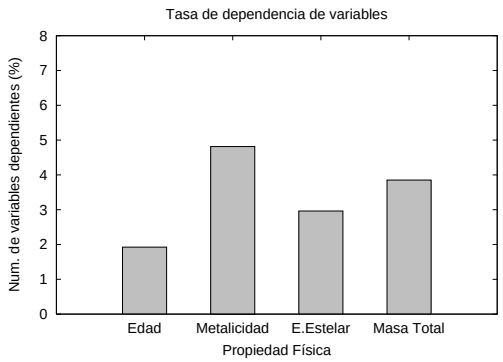
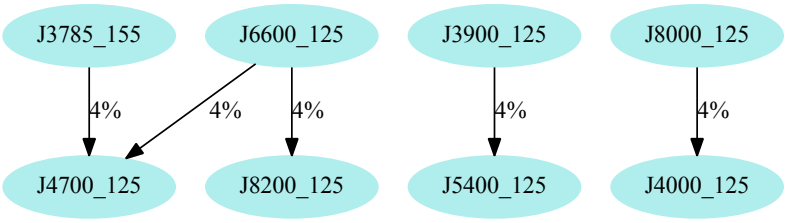


Figura 5.6.: Gráfico de barras de la tasa de dependencia de variables desde la solución más precisa a la solución del primer cuartil en problemas astrofísicos

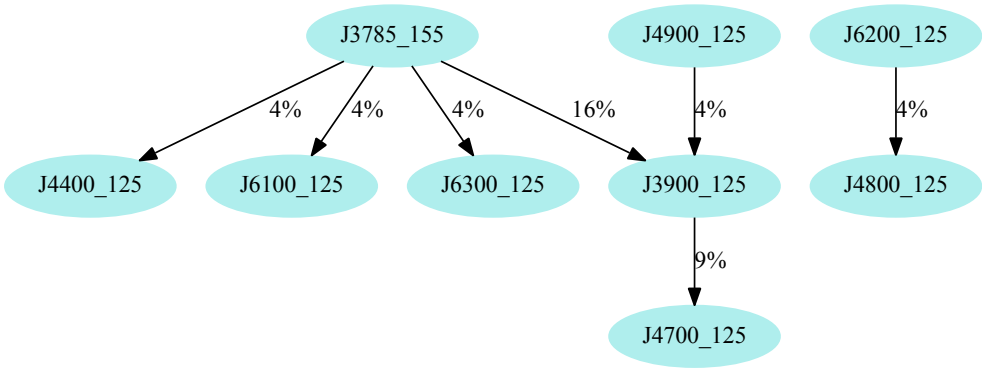
La figura 5.7 ilustra la correlación entre variables mediante un grafo dirigido en cada uno de los problemas. Cada nodo se ha etiquetado con el nombre de la variable y los arcos representan el grado de dependencia entre parejas de variables, donde la cabeza es la entrada de un módulo con una cola como salida que muestra el porcentaje de casos. Este grafo nos puede dar una idea para comprender la correlación existente entre las variables de entrada descubiertas por SDJSG-Ajuste. En las figuras se han omitido las relaciones de dependencia menores al 4 %.

La figura 5.8 presenta algunos modelos propuestos de estructuras jerárquicas obtenidas por SDJSG-Ajuste para las propiedades físicas que estamos analizando. Son soluciones no dominadas en una partición de un conjunto de datos. Cada variable de cada modelo es la longitud de onda de la luz emitida por los objetos de la galaxia que queda reflejada en un espectro con una anchura de 100 ångströms. Los modelos presentados en las figuras permiten identificar cuáles son las longitudes de onda esenciales (variables exógenas) para poder determinar una propiedad física. También podemos comprobar que algunas longitudes de onda se pueden obtener a partir de otras (variables dependientes).

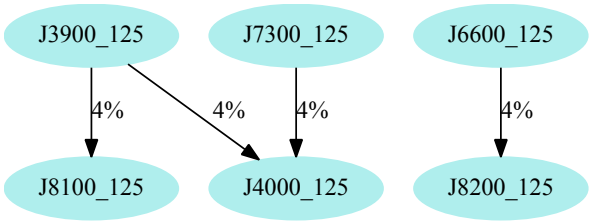
Además, si comparamos todos los modelos podemos ser capaces de extraer longitudes de onda comunes entre modelos. Si nos fijamos detenidamente en los modelos, podemos comprobar que la variable $J3785\_155$ define tanto la metalicidad como la



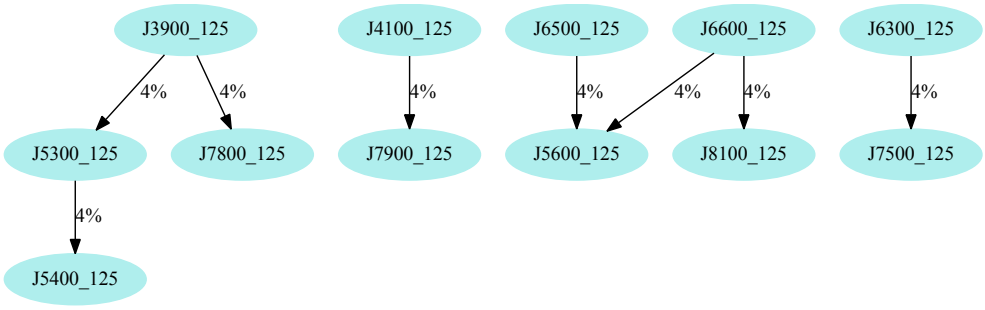
(a) Grafo de dependencia para la Masa Total



(b) Grafo de dependencia para la Metalicidad



(c) Grafo de dependencia para la Extinción Estelar



(d) Grafo de dependencia para la Edad

Figura 5.7.: Grafos de dependencias encontradas en las propiedades físicas (quedan omitidas las relaciones de dependencia menores al 4 %)

masa total y es independiente a otras variables. Sin embargo, en la edad se encuentra como una variable dependiente.

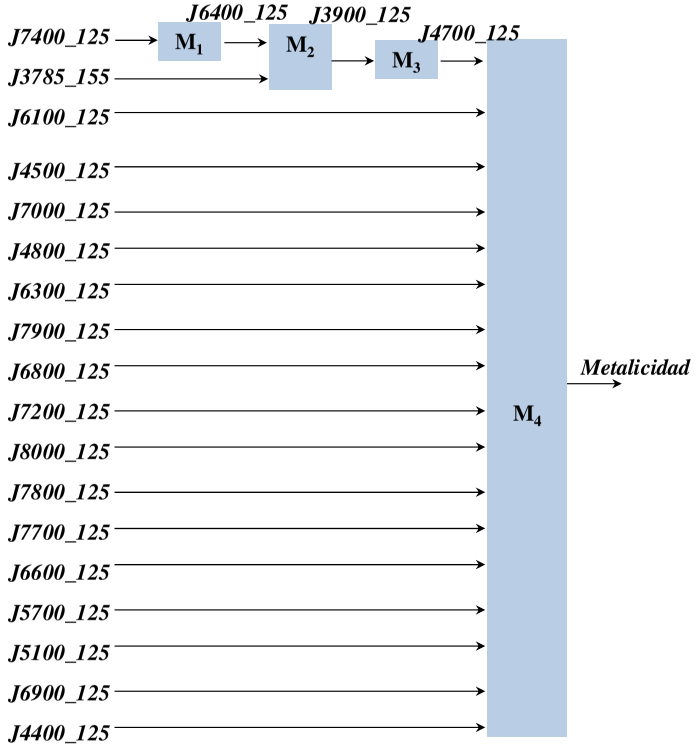
Por otro lado, la variable *J*4700_125 se encuentra como endógena en la metalicidad y en la masa total y como exógena en la edad. Esto nos indica que esta longitud de onda es prescindible en la metalicidad y en la masa total, e imprescindible en la edad. Esto nos puede ayudar a comprender qué longitudes de onda son más influyentes en las propiedades físicas y cuáles podríamos descartar, controlando de esta forma la maldición de la dimensionalidad.

Las figuras 5.9 y 5.10 ilustran dos bases de reglas jerárquicas correspondientes a los modelos 5.8(c) y 5.8(d) respectivamente. En estos modelos se ha reducido el número de variables en un 84 % en el caso de la *Extinción Estelar* y en un 89 % en la *Edad*. Además se han distribuido jerárquicamente en la estructura, lo que permite una mayor interpretabilidad de los problemas debido a la disminución de la complejidad del sistema difuso.

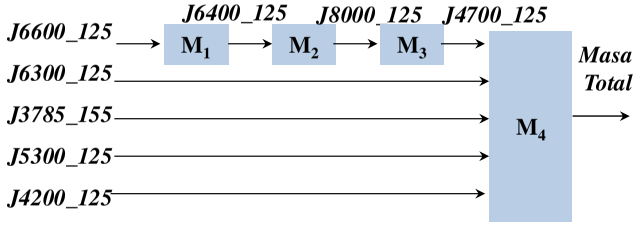
Las figuras 5.11, 5.12 y 5.13 muestran las funciones de pertenencia de cada variable obtenidas por SDJSG-Ajuste en la *Masa Total*, la *Extición Estelar* y en la *Edad* respectivamente correspondientes a los modelos de las figuras 5.8(b), 5.8(c) y 5.8(d), respectivamente.

Una vez que hemos analizado los resultados obtenidos por nuestra propuesta, sería interesante analizar el resultado obtenido por los modelos dados por SDJSG-Ajuste frente a Starlight, que es la metodología más usada para el estudio de la síntesis espectral. La figura 5.14 hace referencia a un conjunto de representaciones gráficas de la dispersión de los datos inferidos por SDJSG-Ajuste frente a la salida dada por Starlight de cada instancia del conjunto de datos de cada propiedad física de los modelos de la figura 5.8. Cada gráfica nos indica si existe algún tipo de correlación entre los resultados obtenidos por Starlight y los resultados obtenidos por SDJSG-Ajuste. Se puede apreciar que la correlación existente en todas las propiedades físicas es positiva ya que los datos concluyen en una línea de ajuste diagonal con pendiente positiva. Esto nos indica que el uso de un modelo jerárquico en serie con ajuste de las funciones de pertenencia es una buena apuesta para la resolución de problemas complejos de regresión.

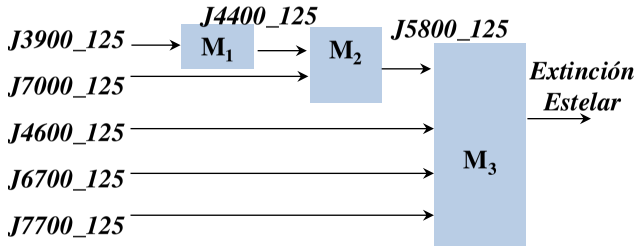
La figura 5.15 ilustra los histogramas correspondientes a la diferencia entre la salida inferida por SDJSG-Ajuste en los SDJS de la figura 5.8 y los resultados de Starlight por instancia del conjunto de datos de cada propiedad física. Los datos representados en el histograma se basan en la diferencia de la inferencia obtenida por



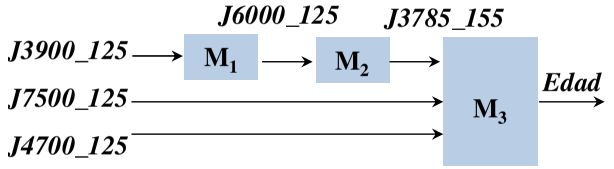
(a) Metalicidad: $RECM_{entr}$ = 0,121899; $RECM_{pru}$ = 0,141944; #M = 4; #R = 457; #V = 21; $\bar{x}_{\#R}$ =16,3



(b) Masa Total: $RECM_{entr}$ = 0,320209; $RECM_{pru}$ = 0,317522; #M = 4; #R = 100; #V = 8; $\bar{x}_{\#R}$ = 4,5



(c) Extinción Estelar: $RECM_{entr}$ = 0,123927; $RECM_{pru}$ = 0,129796; #M = 3; #R = 52; #V = 7; $\bar{x}_{\#R}$ =3,3



(d) Edad: $RECM_{entr}$ = 0,194765; $RECM_{pru}$ = 0,209115; #M = 3; #R = 46; #V = 5; $\bar{x}_{\#R}$ =2,7

Figura 5.8.: Modelos propuestos para cada propiedad física

R_1: **SI** J3900_125 *es* M **ENTONCES** J4400_125 *es* MP
R_2: **SI** J3900_125 *es* P **ENTONCES** J4400_125 *es* M
R_3: **SI** J3900_125 *es* G **ENTONCES** J4400_125 *es* MP
R_4: **SI** J3900_125 *es* MG **ENTONCES** J4400_125 *es* M

(a) Base de Reglas del módulo M_1

R_1 : **SI** J4400_125 *es* M **y** J7000_125 *es* P **ENTONCES** J5800_125 *es* MP
R_2 : **SI** J4400_125 *es* P **y** J7000_125 *es* P **ENTONCES** J5800_125 *es* MP
R_3 : **SI** J4400_125 *es* G **y** J7000_125 *es* M **ENTONCES** J5800_125 *es* EP
R_4 : **SI** J4400_125 *es* M **y** J7000_125 *es* MP **ENTONCES** J5800_125 *es* MP
R_5 : **SI** J4400_125 *es* M **y** J7000_125 *es* M **ENTONCES** J5800_125 *es* P
R_6 : **SI** J4400_125 *es* P **y** J7000_125 *es* MP **ENTONCES** J5800_125 *es* MP
R_7 : **SI** J4400_125 *es* M **y** J7000_125 *es* G **ENTONCES** J5800_125 *es* M
R_8 : **SI** J4400_125 *es* G **y** J7000_125 *es* P **ENTONCES** J5800_125 *es* MP
R_9 : **SI** J4400_125 *es* G **y** J7000_125 *es* MP **ENTONCES** J5800_125 *es* P
R_{10}: **SI** J4400_125 *es* G **y** J7000_125 *es* G **ENTONCES** J5800_125 *es* M
R_{11}: **SI** J4400_125 *es* G **y** J7000_125 *es* MG **ENTONCES** J5800_125 *es* G

(b) Base de Reglas del módulo M_2

R_1 : **SI** J5800_125 *es* P **y** J4600_125 *es* P **y** J6700_125 *es* P **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* M
R_2 : **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* P
R_3 : **SI** J5800_125 *es* P **y** J4600_125 *es* G **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* P
R_4 : **SI** J5800_125 *es* P **y** J4600_125 *es* P **y** J6700_125 *es* P **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* P
R_5 : **SI** J5800_125 *es* P **y** J4600_125 *es* P **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* MG
R_6 : **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* G
R_7 : **SI** J5800_125 *es* P **y** J4600_125 *es* P **y** J6700_125 *es* M **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* MG
R_8 : **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* P **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* EP
R_9 : **SI** J5800_125 *es* P **y** J4600_125 *es* MP **y** J6700_125 *es* P **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* M
R_{10}: **SI** J5800_125 *es* M **y** J4600_125 *es* G **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* P
R_{11}: **SI** J5800_125 *es* P **y** J4600_125 *es* G **y** J6700_125 *es* G **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* P
R_{12}: **SI** J5800_125 *es* MP **y** J4600_125 *es* G **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* MP
R_{13}: **SI** J5800_125 *es* P **y** J4600_125 *es* MP **y** J6700_125 *es* P **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* P
R_{14}: **SI** J5800_125 *es* M **y** J4600_125 *es* M **y** J6700_125 *es* G **y** J7700_125 *es* G **ENTONCES** Extinción Estelar *es* EG
R_{15}: **SI** J5800_125 *es* M **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* MG
R_{16}: **SI** J5800_125 *es* M **y** J4600_125 *es* G **y** J6700_125 *es* G **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* G
R_{17}: **SI** J5800_125 *es* MP **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* P
R_{18}: **SI** J5800_125 *es* M **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* P
R_{19}: **SI** J5800_125 *es* P **y** J4600_125 *es* P **y** J6700_125 *es* MP **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* M
R_{20}: **SI** J5800_125 *es* P **y** J4600_125 *es* G **y** J6700_125 *es* M **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* P
R_{21}: **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* MP **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* P
R_{22}: **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* M
R_{23}: **SI** J5800_125 *es* MP **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* MP
R_{24}: **SI** J5800_125 *es* MP **y** J4600_125 *es* G **y** J6700_125 *es* M **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* MP
R_{25}: **SI** J5800_125 *es* M **y** J4600_125 *es* G **y** J6700_125 *es* G **y** J7700_125 *es* G **ENTONCES** Extinción Estelar *es* G
R_{26}: **SI** J5800_125 *es* P **y** J4600_125 *es* MP **y** J6700_125 *es* MP **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* P
R_{27}: **SI** J5800_125 *es* M **y** J4600_125 *es* M **y** J6700_125 *es* G **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* MG
R_{28}: **SI** J5800_125 *es* P **y** J4600_125 *es* G **y** J6700_125 *es* M **y** J7700_125 *es* P **ENTONCES** Extinción Estelar *es* EP
R_{29}: **SI** J5800_125 *es* G **y** J4600_125 *es* G **y** J6700_125 *es* G **y** J7700_125 *es* G **ENTONCES** Extinción Estelar *es* MG
R_{30}: **SI** J5800_125 *es* M **y** J4600_125 *es* M **y** J6700_125 *es* M **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* MG
R_{31}: **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* P **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* P
R_{32}: **SI** J5800_125 *es* M **y** J4600_125 *es* P **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* EG
R_{33}: **SI** J5800_125 *es* M **y** J4600_125 *es* MG **y** J6700_125 *es* G **y** J7700_125 *es* G **ENTONCES** Extinción Estelar *es* M
R_{34}: **SI** J5800_125 *es* P **y** J4600_125 *es* MG **y** J6700_125 *es* M **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* P
R_{35}: **SI** J5800_125 *es* MG **y** J4600_125 *es* MG **y** J6700_125 *es* MG **y** J7700_125 *es* MG **ENTONCES** Extinción Estelar *es* MG
R_{36}: **SI** J5800_125 *es* M **y** J4600_125 *es* MP **y** J6700_125 *es* P **y** J7700_125 *es* MP **ENTONCES** Extinción Estelar *es* G
R_{37}: **SI** J5800_125 *es* P **y** J4600_125 *es* M **y** J6700_125 *es* MP **y** J7700_125 *es* M **ENTONCES** Extinción Estelar *es* G

(c) Base de Reglas del módulo M_3

Figura 5.9.: Base de Reglas obtenida por SDJSG-Ajuste para la *Extinción Estelar* correspondiente a la figura 5.8(c)

R_1: **SI** J3900_125 *es* M **ENTONCES** J6000_125 *es* P
R_2: **SI** J3900_125 *es* P **ENTONCES** J6000_125 *es* M
R_3: **SI** J3900_125 *es* G **ENTONCES** J6000_125 *es* P
R_4: **SI** J3900_125 *es* MG **ENTONCES** J6000_125 *es* P

(a) Base de Reglas del módulo M_1

R_1: **SI** J6000_125 *es* P **ENTONCES** J3785_155 *es* MG
R_2: **SI** J6000_125 *es* M **ENTONCES** J3785_155 *es* MP

(b) Base de Reglas del módulo M_2

R_1 : **SI** J3785_155 *es* M **y** J7500_125 *es* P **y** J4700_125 *es* P **ENTONCES** Edad *es* M
R_2 : **SI** J3785_155 *es* G **y** J7500_125 *es* P **y** J4700_125 *es* M **ENTONCES** Edad *es* M
R_3 : **SI** J3785_155 *es* P **y** J7500_125 *es* P **y** J4700_125 *es* M **ENTONCES** Edad *es* MG
R_4 : **SI** J3785_155 *es* MP **y** J7500_125 *es* M **y** J4700_125 *es* G **ENTONCES** Edad *es* EG
R_5 : **SI** J3785_155 *es* M **y** J7500_125 *es* MP **y** J4700_125 *es* P **ENTONCES** Edad *es* M
R_6 : **SI** J3785_155 *es* G **y** J7500_125 *es* MP **y** J4700_125 *es* P **ENTONCES** Edad *es* M
R_7 : **SI** J3785_155 *es* M **y** J7500_125 *es* P **y** J4700_125 *es* M **ENTONCES** Edad *es* G
R_8 : **SI** J3785_155 *es* P **y** J7500_125 *es* M **y** J4700_125 *es* M **ENTONCES** Edad *es* G
R_9 : **SI** J3785_155 *es* G **y** J7500_125 *es* MP **y** J4700_125 *es* MP **ENTONCES** Edad *es* P
R_{10}: **SI** J3785_155 *es* MP **y** J7500_125 *es* M **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{11}: **SI** J3785_155 *es* P **y** J7500_125 *es* M **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{12}: **SI** J3785_155 *es* MG **y** J7500_125 *es* P **y** J4700_125 *es* M **ENTONCES** Edad *es* M
R_{13}: **SI** J3785_155 *es* G **y** J7500_125 *es* P **y** J4700_125 *es* M **ENTONCES** Edad *es* G
R_{14}: **SI** J3785_155 *es* MG **y** J7500_125 *es* MP **y** J4700_125 *es* MP **ENTONCES** Edad *es* MP
R_{15}: **SI** J3785_155 *es* P **y** J7500_125 *es* G **y** J4700_125 *es* G **ENTONCES** Edad *es* M
R_{16}: **SI** J3785_155 *es* MG **y** J7500_125 *es* P **y** J4700_125 *es* P **ENTONCES** Edad *es* P
R_{17}: **SI** J3785_155 *es* P **y** J7500_125 *es* P **y** J4700_125 *es* P **ENTONCES** Edad *es* M
R_{18}: **SI** J3785_155 *es* MG **y** J7500_125 *es* P **y** J4700_125 *es* P **ENTONCES** Edad *es* M
R_{19}: **SI** J3785_155 *es* M **y** J7500_125 *es* M **y** J4700_125 *es* M **ENTONCES** Edad *es* G
R_{20}: **SI** J3785_155 *es* MP **y** J7500_125 *es* P **y** J4700_125 *es* M **ENTONCES** Edad *es* MG
R_{21}: **SI** J3785_155 *es* MP **y** J7500_125 *es* MP **y** J4700_125 *es* M **ENTONCES** Edad *es* MG
R_{22}: **SI** J3785_155 *es* M **y** J7500_125 *es* M **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{23}: **SI** J3785_155 *es* MP **y** J7500_125 *es* MP **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{24}: **SI** J3785_155 *es* MP **y** J7500_125 *es* G **y** J4700_125 *es* G **ENTONCES** Edad *es* EG
R_{25}: **SI** J3785_155 *es* M **y** J7500_125 *es* G **y** J4700_125 *es* M **ENTONCES** Edad *es* M
R_{26}: **SI** J3785_155 *es* MG **y** J7500_125 *es* P **y** J4700_125 *es* MP **ENTONCES** Edad *es* P
R_{27}: **SI** J3785_155 *es* G **y** J7500_125 *es* M **y** J4700_125 *es* M **ENTONCES** Edad *es* M
R_{28}: **SI** J3785_155 *es* P **y** J7500_125 *es* P **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{29}: **SI** J3785_155 *es* G **y** J7500_125 *es* G **y** J4700_125 *es* M **ENTONCES** Edad *es* MG
R_{30}: **SI** J3785_155 *es* P **y** J7500_125 *es* MP **y** J4700_125 *es* M **ENTONCES** Edad *es* MG
R_{31}: **SI** J3785_155 *es* MP **y** J7500_125 *es* P **y** J4700_125 *es* G **ENTONCES** Edad *es* EG
R_{32}: **SI** J3785_155 *es* M **y** J7500_125 *es* M **y** J4700_125 *es* P **ENTONCES** Edad *es* G
R_{33}: **SI** J3785_155 *es* P **y** J7500_125 *es* MP **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{34}: **SI** J3785_155 *es* G **y** J7500_125 *es* G **y** J4700_125 *es* G **ENTONCES** Edad *es* M
R_{35}: **SI** J3785_155 *es* MP **y** J7500_125 *es* G **y** J4700_125 *es* MG **ENTONCES** Edad *es* EG
R_{36}: **SI** J3785_155 *es* G **y** J7500_125 *es* P **y** J4700_125 *es* MP **ENTONCES** Edad *es* M
R_{37}: **SI** J3785_155 *es* M **y** J7500_125 *es* G **y** J4700_125 *es* G **ENTONCES** Edad *es* MG
R_{38}: **SI** J3785_155 *es* MP **y** J7500_125 *es* MG **y** J4700_125 *es* MG **ENTONCES** Edad *es* EG
R_{39}: **SI** J3785_155 *es* P **y** J7500_125 *es* G **y** J4700_125 *es* G **ENTONCES** Edad *es* G
R_{40}: **SI** J3785_155 *es* MP **y** J7500_125 *es* M **y** J4700_125 *es* MG **ENTONCES** Edad *es* EG

(c) Base de Reglas del módulo M_3

Figura 5.10.: Base de Reglas obtenida por SDJSG-Ajuste para la *Edad* correspondiente a la figura 5.8(d)

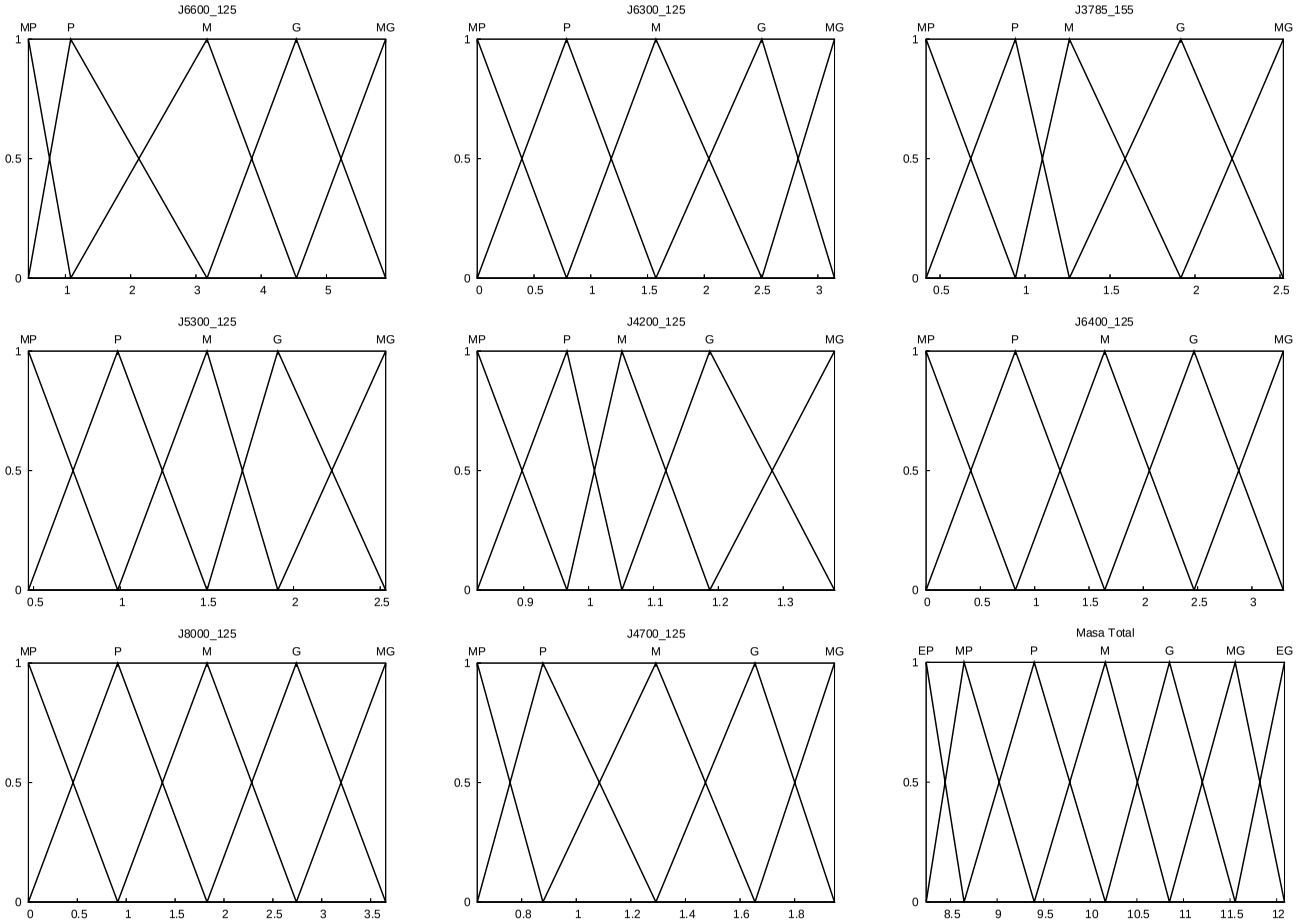


Figura 5.11.: Funciones de pertenencia obtenidas por SDJSG-Ajuste en la *Masa Total* correspondiente al modelo de la figura 5.8(b)

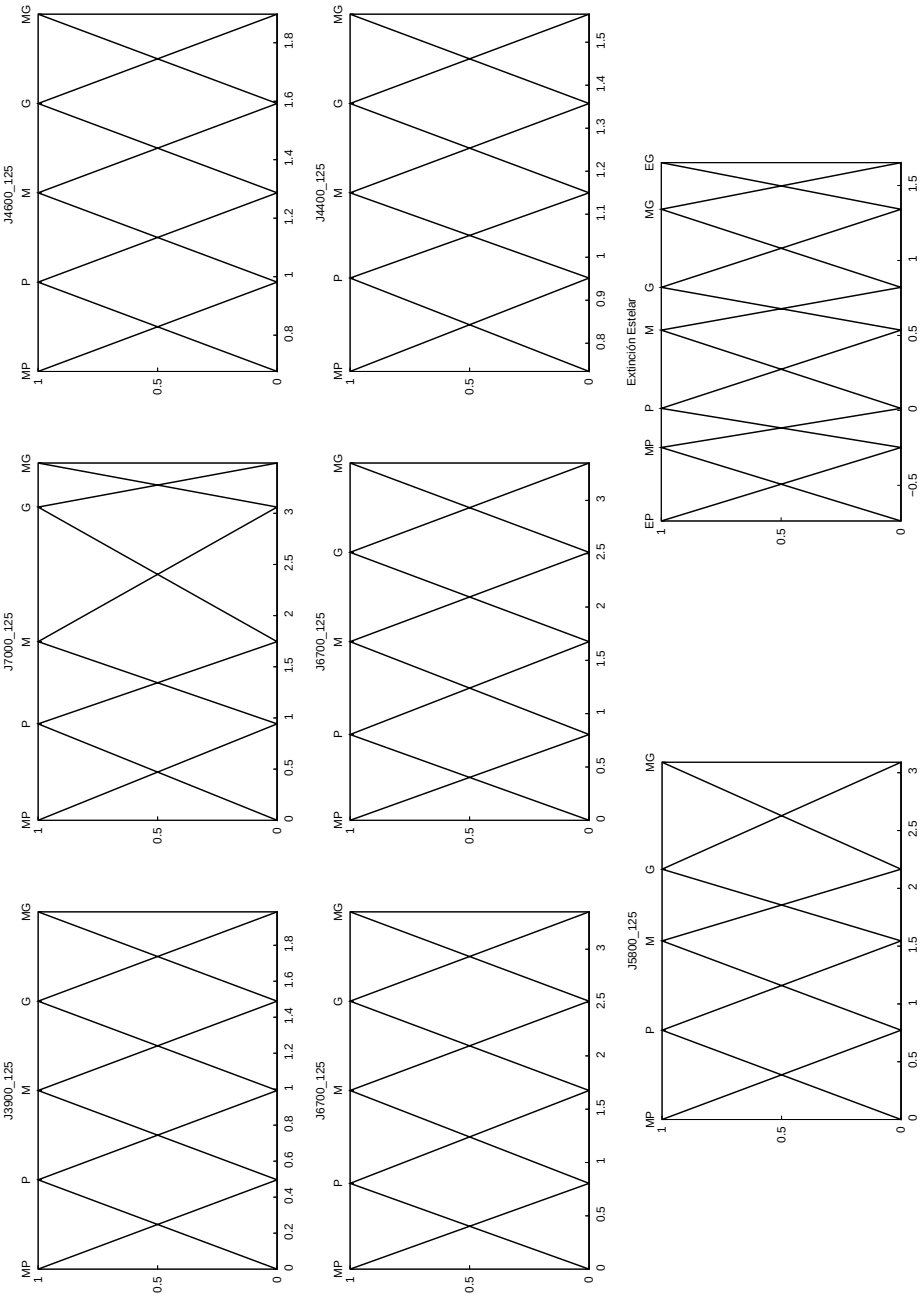


Figura 5.12.: Funciones de pertenencia obtenidas por SDJSG-Ajuste en la *Extinción Estelar* correspondiente al modelo de la figura 5.8(c)

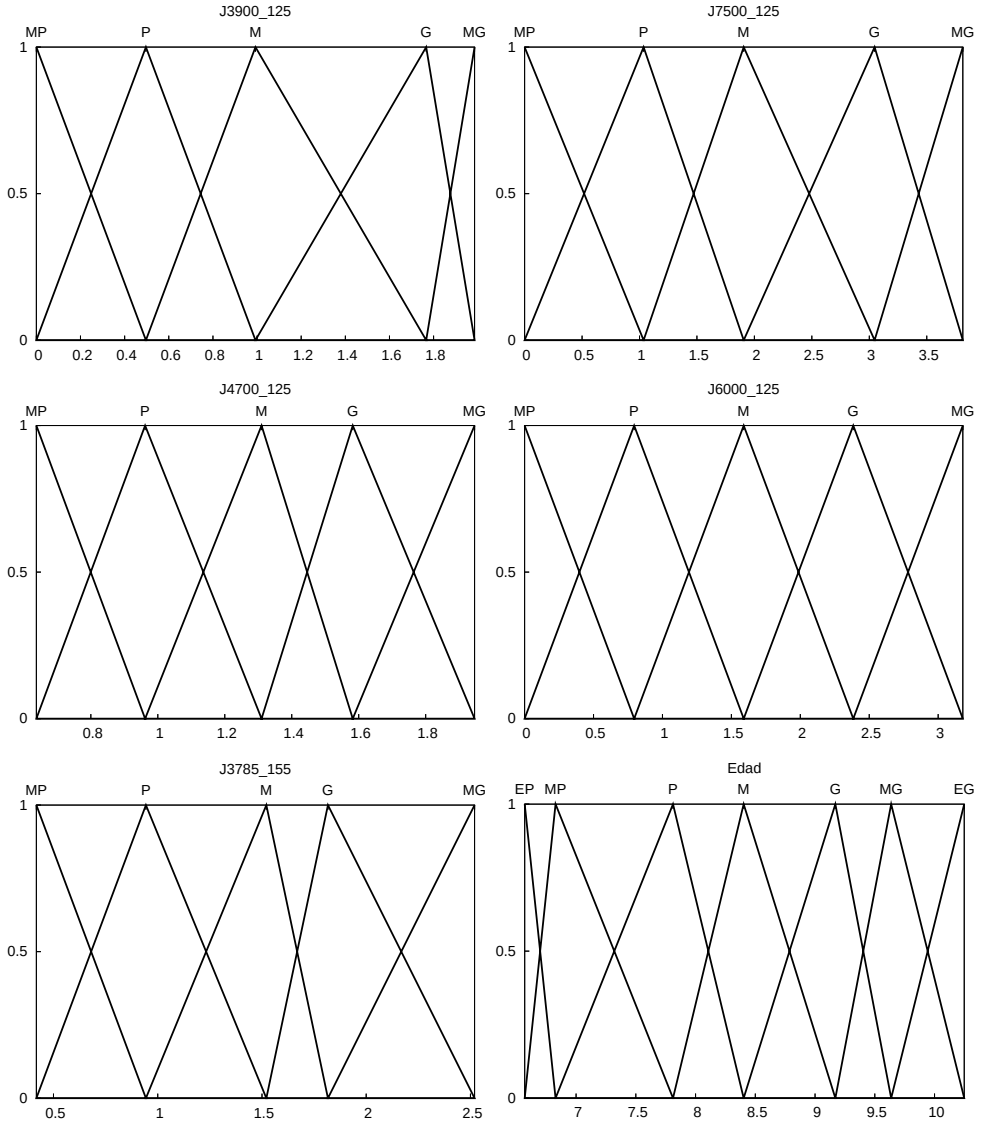


Figura 5.13.: Funciones de pertenencia obtenidas por SDJSG-Ajuste en la *Edad* correspondiente al modelo de la figura 5.8(d)

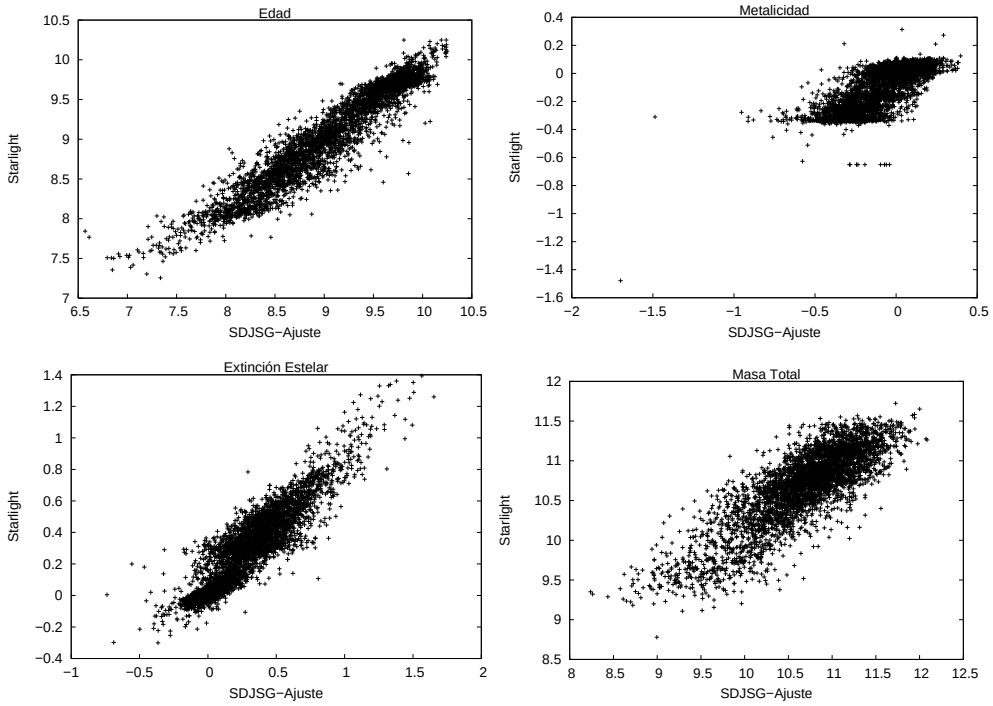


Figura 5.14.: Dispersión de datos de las propiedades físicas correspondientes a los modelos de la figura 5.8

instancia del conjunto de datos entre SDJSG-Ajuste y Starlight. Estos histogramas nos dan una vista general de nuestra propuesta frente a Starlight. En cada histograma de cada propiedad física encontramos una distribución normal de los datos de la muestra. Los datos se concentran alrededor del valor 0 del eje de abcisas y son más escasos a medida que nos aproximamos a los extremos. Esto es una prueba concluyente del comportamiento de nuestra propuesta ya que en la mayoría de las inferencias realizadas, el resultado es similar al método Starlight.

5.7. Sumario

Este capítulo aborda la aplicación de los SDJ a un problema real original, en el campo de la Astrofísica. Para ello, hemos considerado un modelo jerárquico en serie

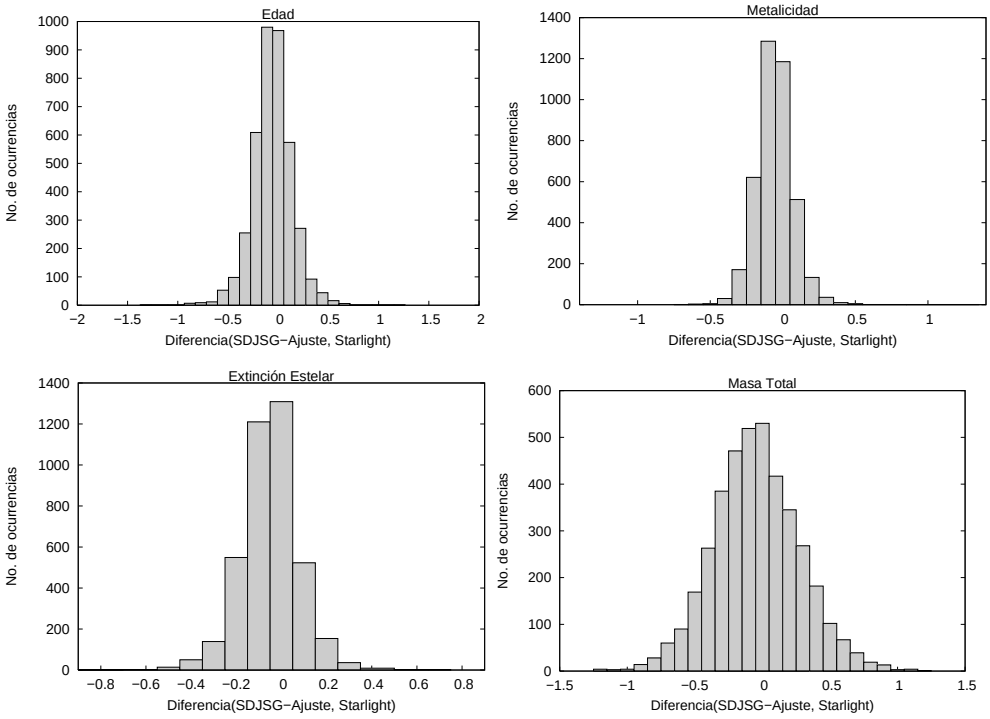


Figura 5.15.: Histogramas de frecuencias de las propiedades físicas correspondientes a los modelos de la figura 5.8

con ajuste en las funciones de pertenencia. Además, hemos comparado esta propuesta con otras metodologías habitualmente usadas para determinar las propiedades físicas de las galaxias. Tras efectuar un estudio experimental de cuatro propiedades físicas, concluimos que:

- El uso de un SDJS disminuye la dificultad de resolución de problemas complejos de regresión: con un número de variables reducido es posible determinar una precisión similar a la obtenida por otros métodos astrofísicos.
- El ajuste de las funciones de pertenencia permite afinar la precisión de las soluciones.
- El uso de SD con reglas de tipo Mamdani aporta un conocimiento adicional

que no tienen otros métodos, favoreciendo la interpretación de los datos. Esto implica el conocimiento de las relaciones existentes entre las variables y su importancia en el conjunto de datos.

En pocas palabras, nuestra propuesta resulta competitiva en términos de precisión e interpretabilidad frente a otros métodos astrofísicos.

Agradecimientos

Nuestro agradecimiento a William Schoenell, Enrique Pérez, ambos pertenecientes al Departamento de Astronomía Extragaláctica del Instituto de Astrofísica de Andalucía en Granada, y a Roberto Cid Fernandes, del Grupo de Astrofísica de la Universidad Federal de Santa Catarina en Florianópolis (Brasil), por proporcionarnos los datos para la experimentación, asesorarnos, orientarnos y ayudarnos en la realización de este capítulo.

Comentarios Finales

(...) parecía que habíamos llegado al final del camino y resulta que era sólo una curva abierta a otro paisaje y a nuevas curiosidades.

JOSÉ SARAMAGO (1922–2010), ESCRITOR, NOVELISTA, POETA, PERIODISTA Y DRAMATURGO PORTUGUÉS

Dedicaremos esta sección a resumir brevemente los resultados obtenidos y a destacar las conclusiones que esta memoria puede aportar. Además, comentaremos algunos aspectos sobre futuros trabajos que siguen la línea aquí expuesta y sobre otras líneas que se pueden derivar.

A. Resultados Obtenidos y Conclusiones

En este trabajo hemos presentado diferentes métodos para solventar problemas de regresión de alta dimensionalidad, manteniendo siempre un equilibrio entre precisión e interpretabilidad del sistema. Nuestro objetivo principal es el de proporcionar buena interpretabilidad en problemas complejos, manteniendo o mejorando la precisión. Los siguientes apartados contienen las conclusiones a las que hemos llegado durante esta labor.

Modelizado mediante Sistemas Difusos Jerárquicos en problemas de alta dimensionalidad

La resolución de problemas de alta dimensionalidad mediante sistemas difusos conlleva el incremento de forma exponencial del número de reglas, dificultando así

la interpretabilidad del conocimiento que contiene el sistema. El modelizado mediante sistemas difusos jerárquicos nos ayuda a paliarlo ya que los módulos de la jerarquía abordan el problema de forma parcial para obtener posteriormente la salida final del sistema. Como hemos visto en este trabajo, existen distintos tipos de jerarquías atendiendo a su topología (serie, paralelo e híbrido). El aprendizaje de estas estructuras mediante técnicas de optimización multiobjetivo hace que se mantenga un equilibrio entre la interpretabilidad y la precisión del sistema debido a que son magnitudes inversamente proporcionales. Por ello, el uso de estas técnicas en el modelizado del sistema aporta:

1. *Interpretabilidad.* La relación entre variables mediante módulos produce una reducción de la complejidad de los problemas de alta dimensionalidad ya que el número total de reglas disminuye, además de la complejidad a nivel de regla debido a que el número de variables por regla es menor. Esto da lugar a un sistema más simple.

2. *Extracción del conocimiento.* La jerarquía en sí nos proporciona información relevante del problema porque establece relaciones de dependencia entre las variables.

3. *Precisión.* Si el número de variables de entrada a cada módulo de la jerarquía es pequeño, esto permite obtener un conocimiento minucioso de cada zona del espacio de soluciones.

Para mejorar aún más la interpretabilidad del sistema, cada módulo posee un conjunto de reglas de tipo Mamdani que dotan al sistema de una mayor simplicidad para comprender mejor el conocimiento aprendido. Una aportación que reduce significativamente la complejidad en los sistemas jerárquicos frente a otras técnicas es que cada módulo de la jerarquía infiere variables naturales del problema y no genera variables adicionales en las capas intermedias. Además, el aprendizaje de cada estructura es flexible, es decir, la estructura se adapta a las necesidades del problema, paliando el problema de la alta dimensionalidad.

Por otro lado, hemos realizado un estudio experimental de cada una de las técnicas propuestas en donde se ha podido apreciar el comportamiento y los resultados obtenidos por las distintas topologías jerárquicas. Los modelos obtenidos en la experimentación con distintos problemas son interpretables tanto a nivel de regla, ya que el número de variables por regla se reduce, como el número total de reglas del

sistema. A pesar de que no era el objetivo de este trabajo, hemos propuesto un método para el modelizado de jerarquías en serie que permite la mejora de la precisión de este tipo de sistemas mediante el ajuste de las funciones de pertenencia. Como se aprecia en los resultados, esta propuesta puede ser comparable a otros modelos de la literatura que utilizan mecanismos de ajuste en el modelizado difuso.

Por último, hemos aplicado esta técnica a problemas de regresión del área de Astrofísica para calcular los modelos de cuatro propiedades físicas de las galaxias. Tal y como se muestra en la experimentación, el modelizado en serie genera modelos competitivos con respecto a los modelos obtenidos por técnicas convencionales de este campo.

Aprendizaje de modelos mediante Sistemas Difusos Jerárquicos

En este trabajo hemos presentado tres métodos de aprendizaje de sistemas jerárquicos, serie, paralelo e híbrido, para problemas de regresión de alta dimensionalidad con el objetivo de obtener soluciones equilibradas en precisión e interpretabilidad. Con una experimentación y un análisis exhaustivo de estas técnicas de modelizado utilizando distintos conjuntos de datos con diferente dimensionalidad, podemos deducir que el comportamiento de cada una de las tres propuestas algorítmicas de este trabajo reducen el número de reglas del sistema en términos de interpretabilidad. Además, el número de variables por regla es menor, lo que hace que este tipo de sistemas se caracterice por su simplicidad. Con respecto a la precisión, está claro que depende principalmente de la topología.

La *topología en paralelo* se caracteriza por no hacer uso de variables exógenas en las capas intermedias del sistema, sin importar el número de módulos existentes en cada capa. Como venimos comentando, la inferencia de cada módulo genera un error que se incrementa a medida que transcurre la información por la jerarquía hasta llegar al módulo de salida. El error acarreado depende directamente del número módulos, esto es, cuanto mayor sea número de módulos, el sistema genera un mayor error en la predicción. Esto repercute negativamente en la precisión del sistema. Este comportamiento se atenúa en la topología híbrida.

La *topología híbrida*, como se puede apreciar en su propia morfología, hace uso de variables exógenas que aportan información sin ruido al sistema a nivel de capa. De esta forma, el propio sistema mitiga la propagación del error a lo largo de las capas, aunque el número de módulos de la jerarquía sea elevado. Un problema importante que presenta este tipo de jerarquía es el incremento del espacio de búsqueda

de soluciones. Un modelizado híbrido contempla topología serie, paralela y ambos simultáneamente en una misma estructura, lo que incrementa en tres veces el espacio de búsqueda y hace que encontrar una solución buena sea una tarea ardua.

Por otro lado, la *topología en serie* es una estructura más simple, ya que por cada capa solo genera un módulo. Por ello, la generación de errores por módulo es menor y por supuesto, la propagación del mismo también es menor. A ésto se une la cualidad de que en cada capa puede existir una variable exógena, es decir, puede contar con la presencia de datos sin ruido que ayudan a mitigar la generación y propagación del error a través de las capas, lo que repercute positivamente en la precisión final del sistema. Dado el buen comportamiento que presenta esta topología frente a las otras dos, hemos podido comprobar que un ajuste en las funciones de pertenencia hace que las jerarquías en serie obtenidas afinen aún más en la precisión y pueda ser comparable a otros modelos de la literatura que utilizan mecanismos de ajuste en el modelado difuso.

Resumiendo, un sistema difuso jerárquico puede ser una buena apuesta en problemas de multidimensionalidad ya que nos proporciona simplicidad e interpretabilidad, pero si no queremos perder precisión, hemos de ser cautos en su diseño atendiendo tanto en el número de módulos como en la distribución del tipo de variables.

Aplicación del modelizado en serie con ajuste a problemas astrofísicos

Después de comprobar el buen funcionamiento de la metodología de aprendizaje de estructuras jerárquicas en serie y la mejora en precisión que supone el ajuste de las funciones de pertenencia, nos planteamos aplicar este algoritmo en el cálculo de cuatro propiedades físicas de las galaxias (problemas de alta dimensionalidad) y realizar una comparación con otros métodos usados habitualmente en este área. Esta propuesta resulta competitiva frente a otras técnicas ya que:

- *Disminuye la complejidad del problema.* Hemos podido comprobar que con el uso de un número reducido de variables distribuidas en un sistema jerárquico en serie se obtiene una precisión semejante a la de otros métodos empleados en el cálculo de propiedades físicas de galaxias.

- *Aumenta la precisión del modelo*, gracias al potencial del ajuste de las funciones de pertenencia y al estudio parcial del espacio de búsqueda de soluciones.

- *Mejora la interpretabilidad del conocimiento del modelo*. La topología en serie establece una relación entre las variables del problema y nos indica cuáles de ellas son independientes y cuáles no. En un espacio de búsqueda de 45 dimensiones, que es el tamaño de cada uno de los problemas, este tipo de estructuras producen una reducción dimensional indicando las variables que realmente son importantes. Este comportamiento implica que el sistema basado en reglas difusas contenido en cada módulo sea menos complejo, reduciendo a nivel global el número de reglas y caracterizándolo por su simplicidad a nivel de regla.

B. Líneas de Investigación Futuras

A continuación, consideraremos algunas posibles extensiones sobre los métodos propuestos en esta memoria y discutiremos varias líneas de trabajo inmediatas. Estas ideas nos permitirán analizar y mejorar el comportamiento de cada uno de los algoritmos contenidos en esta memoria.

Estudio del error en cada módulo

Como venimos comentando, cada módulo genera un error en la inferencia de la variable de salida. Nos resulta interesante el estudio de ese error que se propaga a través de las capas. Podemos comenzar contrastando los módulos en distintas topologías y con distintos problemas, analizando con detenimiento tanto el proceso de fuzzificación como el de defuzzificación. También nos cuestionamos el uso de técnicas de optimización sobre estos procesos. Esto permitirá realizar mejoras en la precisión de las jerarquías como lo han hecho Alcalá-Fdez y cols. en su trabajo de 2007 y Márquez y cols. en su trabajo de 2010 implementando una defuzzificación adaptativa para la mejora de la interpretabilidad.

Diseño de nuevos métodos que permitan mejorar la precisión de un sistema jerárquico

Como hemos comprobado, el ajuste de las funciones de pertenencia repercuten en la precisión muy positivamente, ya que el sistema afina aún más. Mejoraremos el método de ajuste que presentamos en el trabajo diseñando un mecanismo que, con poco coste computacional, aprenda el número de etiquetas. En este caso, cada variable llevaría asociada el número de etiquetas, al igual que hacemos con las funciones

de pertenencia. Ejemplos de uso de esta metodología adaptativa los encontramos en los trabajos de Acosta y cols. (2007a; 2007b).

Aplicar otros métodos para mejorar la precisión e interpretabilidad del sistema

Tenemos en mente aplicar otros métodos existentes en la literatura tales como la Metodología de Reglas Cooperativas (Casillas y cols. 2002), para la mejora de la precisión e interpretabilidad del sistema. Esta técnica considera la independencia entre reglas que se encuentran codificadas en una base de reglas. Esta propuesta se puede realizar de forma local a cada módulo o de forma global a la jerarquía. Dado que el método de WM no realiza un ajuste de etiquetas, pensamos que esta metodología aportaría precisión ya que se genera un conjunto de consecuentes candidatos para encontrar la mejor cooperación, sin empeorar la interpretabilidad del sistema, y construiría modelos con mayor nivel de jerarquía. Además, esta metodología incorpora un mecanismo de eliminación de reglas que permitirá mejorar la interpretabilidad en la medida de lo posible. Esta propuesta resulta beneficiosa para problemas con un número considerable de variables.

Aplicar cada una de las técnicas al resto de las propiedades físicas de las galaxias

Los datos del survey SDSS se componen de 16 propiedades físicas, de las que sólo hemos analizado cuatro por encontrarse aún bajo estudio. Por ello, pretendemos extender el trabajo y usar cada una de las propuestas anteriores en cada una de ellas. Nuestro objetivo es el de proporcionar una interpretabilidad los datos, a la par que precisión, ya que la mayoría de los métodos astrofísicos que se suelen usar son analíticos y poco interpretables.

También resulta interesante aplicar otras técnicas tales como métodos de optimización de reglas, o incluso otra metodología distinta de los sistemas difusos.

Apéndices

A. Métodos de Comparación

Con el conocimiento se acrecientan las dudas.

JOHANN WOLFGANG VON GOETHE
(1749–1832), ESCRITOR ALEMÁN

Este apéndice está dedicado a la descripción de los algoritmos implementados para la comparación con nuestras propuestas. Para ello, se dedica una sección a cada método para explicar su comportamiento.

A.1. Método de Wang y Mendel

El aprendizaje basado en ejemplos con reglas de tipo Mamdani propuesto por L. Wang y Mendel en 1992 es muy conocido y usado gracias a su simplicidad. Nuestros algoritmos lo usan en el aprendizaje de la base de reglas de cada módulo. Se basa en contemplar un conjunto de parejas de datos entrada-salida que representa el comportamiento del problema que se va a resolver:

$$E = \{e_1, \ldots, e_N\}, e_l = (x_1^l, \ldots, x_n^l, y_1^l, \ldots, y_n^l),$$

siendo N el tamaño del conjunto de datos, n el número de variables de entrada y m el número de variables de salida. El algoritmo consiste en los siguientes pasos:

1. Considerar una partición difusa (definiciones de los parámetros de las funciones de pertenencia) para cada variable de entrada/salida.

2. Generar un conjunto de reglas difusas candidatas: Este conjunto se forma por el criterio de mejor cubrimiento de cada ejemplo contenido en E. De esta forma se obtienen N reglas difusas candidatas, RC^l. La estructura de cada regla

se genera tomando un ejemplo específico, es decir, un vector real $(n+m)$–dimensional y estableciendo cada una de las variables al término lingüístico (conjunto difuso asociado) con mejor cubrimiento de todos los componentes del vector:

RC^l: SI X_1 es A_1^l y ... y X_n es A_n^l ENTONCES Y_1 es B_1^l y ... y Y_m is B_m^l
$A_i^l = \arg\max_{A' \in A_i} \mu_{A'}(x_i^l)$, $B_j^l = \arg\max_{B' \in B_j} \mu_{B'}(y_j^l)$

3. Dar un grado de importancia a cada regla candidata:

$$G(RC^l) = \Pi_{i=1}^{n} \mu_{A_i^l}(x_i^l) \cdot \Pi_{j=1}^{m} \mu_{B_j^l}(y_j^l)$$

4. Obtener un conjunto final de reglas difusas a partir del conjunto de reglas difusas candidatas. Para ello, las N reglas candidatas son agrupadas en g grupos diferentes, cada uno de ellos formado por todas las reglas candidatas que contienen la misma combinación de antecedentes. Para construir el conjunto final de reglas difusas, se elige de cada grupo la regla con más grado de importancia. Por lo tanto, g será tanto el número de combinaciones diferentes de antecedentes en el conjunto de reglas candidatas como el número de reglas en el conjunto de reglas difusas final de tipo Mamdani.

A.2. Selección de Variables mediante un Algoritmo Genético

El algoritmo SVAG realiza una selección de variables por medio de un algoritmo elitista multiobjetivo basado en poblaciones con el objetivo de obtener un SD preciso e interpretable. El algoritmo 13 muestra un pseudocódigo.

Cada individuo es un SD con un módulo en donde se aplican dos operadores genéticos:

- Cruce: Es similar al operador de cruce del algoritmo SDJSG cuando ambos padres tienen un módulo (ver sección 3.1.3.4).

Algoritmo 13 Algoritmo SVAG

Entrada: Tamaño de la población, probabilidad de cruce y mutación. Conjunto de datos: $D = \{(x,y)|x \in \mathbb{R}^n, y \in \mathbb{R}^m\}$. Definición de las funciones de pertenencia.
Salida: Conjunto de soluciones no dominandas, cada una con un equilibrio distinto entre precisión/número de reglas.

```
Inicialización(P)
Evaluación(P)
Mientras (no se cumpla la condición de parada) hacer
    P1 ← Selección_Multiobjetivo(P)
    P2 ← Cruce(P1)
    P3 ← Mutación(P2)
    Evaluación(P3)
    P ← Reemplazamiento_Multiobjetivo(P3)
Fin Mientras
```

- Mutación: Este operador elige probabilísticamente entre añadir/eliminar una variable usada a/de un SD. Esto se muestra en el algoritmo 14.

La estrategia de reemplazamiento de individuos que se utiliza es la del algoritmo NSGA-II (Deb y cols. 2002), el mecanismo de inferencia es el esquema Max–Min y se consideran dos objetivos: el ECM y el número máximo de reglas. El SD aprende un conjunto de reglas difusas de tipo Mamdani mediante el método de L. Wang y Mendel (1992).

Algoritmo 14 Operador de mutación del algoritmo SVAG

Entrada: Sistema Difuso SD.

```
r = U[0,1] {Probabilidad uniforme}
Si (r < 0,5 y el número de variables no usadas >0) entonces
    Insertar_una_variable_aleatoria_no_usada(SD)
Sino
    Si (número de variables usadas >1) entonces
        Eliminar_una_variable_aleatoria_usada(SD)
    Fin Si
Fin Si
```

A.3. Algoritmo de Joo y Sudkamp

El algoritmo de Joo y Sudkamp (2009) consiste en la conversión de un SD de una sola capa en un sistema difuso jerárquico de dos capas. El primer paso establece el número de variables pertenecientes a cada capa: sea n el número de variables y k el

número de etiquetas, con $k \geq 2$; el número de variables en la primera capa se calcula mediante la ecuación:

$$p^* = \frac{n - \log_k 2}{3} \tag{A.1}$$

El número de variables que minimizan el número de reglas se calcula de la siguiente forma:

$$R(p)_{min} = \text{mín}\{R(\lfloor p^* \rfloor), R(\lfloor p^* \rfloor + 1)\} \tag{A.2}$$

donde $\lfloor x \rfloor$ es el mayor entero menor o igual a x y $R(x)$ se obtiene:

$$R(x) = k^x k^x + k^{n-x} \tag{A.3}$$

El siguiente paso es crear los módulos de la primera capa: se genera una base de reglas completa y las reglas se agrupan atendiendo a los antecedentes de las variables pertenecientes a la segunda capa con igual valor de etiqueta. Cada grupo crea un módulo en la primera capa. Si hay p variables en la primera etiqueta, el algoritmo generará k^{n-p} módulos y cada módulo tendrá k^p reglas. El algoritmo incluye un mecanismo de reducción de reglas jerárquico que elimina los módulos de la primera capa con una dependencia lineal mediante el cálculo del rango de la matriz llamada *BR*. Cada fila *i* de la matriz *BR* representa un módulo y cada columna es el consecuente de la regla j del módulo i. El número de módulos en la primera capa se obtiene mediante el rango de la matriz *BR*. Usa como fuzzificación un *singleton* (función A.4):

$$\mu_A(x) = \begin{cases} 1 & \text{si } x = a \\ 0 & \text{en otro caso} \end{cases} \tag{A.4}$$

donde $x \in A$ es la entrada a un SD y a es el centro de un conjunto *singleton* definido A. Como inferencia usa el producto y como defuzzificación por promedio entero.

A continuación, se crea la segunda capa. A diferencia de un SD estándar, la segunda capa del SD recibe los valores del resto de variables de entrada y los valores de los consecuentes de las reglas de la primera capa como entrada. El algoritmo genera una base de reglas completa de tipo TSK con $n - p$ variables. Los consecuentes de las reglas se construyen mediante la combinación de los valores inferidos de los módulos

de la primera capa con constantes reales. Estas constantes se obtienen de un conjunto de operaciones matriciales sobre la matriz *BR*.

Nuestra implementación de JS parte de un conjunto reducido de variables dado por SVAG por ejecución, en lugar de generar la base de reglas completa. El objetivo de esto es obtener un algoritmo de JS comparable en las mismas condiciones que el resto de los algoritmos implementados. El principal problema es la construcción de la matriz *BR* porque algunas celdas de la matriz no tienen un valor asociado debido a la falta de reglas. Por ello, decidimos rellenar las celdas con valores intermedios de la variable de salida con el objetivo de completar las celdas vacías de la matriz y después, insertar la regla asociada a la celda rellena en el módulo correspondiente. Esta inserción no produce cambios mayores en la salida de la primera capa por el uso de una inferencia FITA (Inferenciar Primero, Agregar Después), por lo que podemos considerarlo como un modelo TSK simplificado.

B. Aprendizaje Automático

Aprendemos, o por inducción o por demostración.
La demostración parte de lo universal; la inducción de lo particular.

ARISTÓTELES (384 A.C. – 322 A.C.), FILÓSOFO GRIEGO

Este apéndice realiza una pequeña introducción al aprendizaje automático de forma general, describiendo en qué consiste y el tipo de metodología que podemos encontrar en la literatura.

B.1. Conceptos básicos

El Aprendizaje Automático (Burke y Kendall (2005); H. Wang y cols. (2009); Yao y Liu (2014)) es el área de la Inteligencia Artificial que estudia el diseño y desarrollo de modelos de aprendizaje computacionales mediante un conjunto de métodos basados en un conocimiento acumulado. Su objetivo es simular el aprendizaje humano con métodos que, además de realizar el proceso de aprendizaje, lo mejoren de forma continua.

El modelo básico del aprendizaje automático establecido por An y Zhang (2007) consiste en un conjunto de procesos que se muestran en la figura B.1:

- Entorno: Proporciona información externa al sistema.
- Aprendizaje: Procesa la información recibida, la transforma en conocimiento y la deposita en un repositorio.

- Repositorio: Se encarga de almacenar el conocimiento para guiar las decisiones/acciones. La expresión del conocimiento puede venir dada mediante la producción de reglas, lógica de primer orden, autovectores, redes semánticas, etc.
- Ejecución: Lleva a la práctica el conocimiento aprendido.

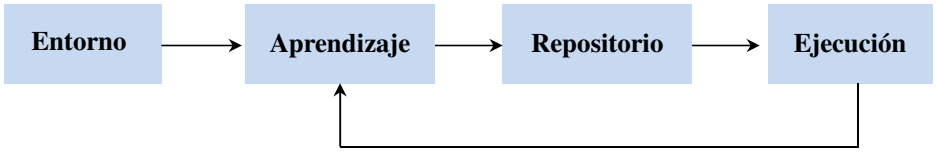


Figura B.1.: Modelo básico del aprendizaje automático (figura extraÃda del trabajo de H. Wang y cols. en 2009)

Por otro lado, existen distintos métodos de aprendizaje según H. Wang y cols. (2009):

- Por Repetición: Consiste en almacenar el nuevo conocimiento en una memoria y cuando el conocimiento se necesita, se hace uso de él. Su implementación se considera tan abstracta como una función.
- Inductivo: Aplica métodos inductivos a partir de un conjunto de ejemplos para establecer reglas generales sobre la información que se tiene.
- Analógico: Este aprendizaje se basa en la similaridad entre objetos mediante la comparación de cosas similares.
- Explicado: Analiza las instancias atendiendo al conocimiento contenido en cada campo y posteriormente se deduce una explicación de causa-efecto, es decir, se aprende nuevo conocimiento mediante la naturaleza, los tokens y las relaciones internas.
- Basado en redes neuronales: La naturaleza de las redes neuronales depende principalmente de la topología de su estructura y sus reglas de trabajo. Su principal problema es el ajuste de los valores de la red. La red más conocida es la que contiene el método de retropropagación.

- Descubrimiento de conocimiento: Es un proceso para identificar un modelo de entendimiento útil y efectivo con gran cantidad de datos. La selección de datos extrae datos relevantes de la base de datos basados en las necesidades de los usuarios. Un ejemplo sería la minería de datos.

Por otro lado, tanto el aprendizaje como el conocimiento supone un gran problema en el diseño de sistemas inteligentes ya que es una tarea tediosa y difícil que a veces no conduce a resultados satisfactorios. Un modelizado inductivo puede servir para predicción de observaciones futuras o para una descripción inteligible de dependencias entre variables dentro de un dominio. El modelizado de inducción comprende los siguientes pasos (Hüllermeier (2008)): 1) Adquisición de datos; 2) Preparación de datos (limpieza, transformación, selección, escalado, ...); 3) Modelo de inducción; 4) Modelo de interpretación y validación; y 5) Modelo de aplicación. Además, se pueden distinguir tres tipos de tareas empíricas en el aprendizaje automático: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo.

El aprendizaje no supervisado se caracteriza porque el proceso se realiza sobre un conjunto de datos. Su objetivo es detectar una estructura en los datos, como por ejemplo, las propiedades de la distribución, relaciones entre entidades de datos o dependencias entre atributos. Por otro lado, el aprendizaje supervisado asume una dependencia entre los datos. El aprendizaje se realiza a partir de un conjunto de ejemplos etiquetados que contienen la solución ideal para cada instancia. Podemos distinguir dos tipos de datos: categóricos y numéricos. Si la variable dependiente es categórica, se trata de un problema de clasificación. Si es numérica, el problema es de regresión. El aprendizaje por refuerzo se basa en la psicología conductista, de forma que se interesa en tomar decisiones atendiendo a una maximización de una recompensa.

El proceso de resolución de un problema implica la creación de un modelo y su utilización para generar una solución. Normalmente, el mejor modelo aporta la mejor solución, pero a veces la solución obtenida sólo abarca ese modelo en concreto sin gozar de una generalidad que le permita adecuarse a otros problemas de naturaleza similar. El modelizado mediante sistemas difusos, basados en un aprendizaje inductivo, puede solventar este problema ya que proporciona mayor generalidad, expresividad y tolerancia a la imprecisión.

C. Técnicas de Optimización

No es el más fuerte ni el más inteligente
el que sobrevive,
sino aquel que se adapta más a los cambios.

CHARLES DARWIN (1809–1882),
BIÓLOGO INGLÉS

Este apéndice explica las técnicas de optimización en las que se basa nuestro trabajo: algoritmos genéticos y algoritmos evolutivos multiobjetivo.

C.1. Algoritmos Genéticos

Los Algoritmos Genéticos (AG) son una familia de modelos computacionales inspirados en la evolución (Whitley (1994)). Estos algoritmos codifican soluciones potenciales de un problema en una estructura de datos, llamada *cromosoma* o *individuo*. Cada cromosoma está formado por un conjunto de unidades, en donde a cada una de ellas se le atribuye el nombre de *gen*. Cada gen podrá manifestarse de forma diferente, según los valores que tome. Estos se conocen con el nombre de *alelos*. El término *fenotipo* hace referencia al conjunto de parámetros candidatos a la solución del problema. La figura C.1 muestra un ejemplo de un cromosoma. Como se puede apreciar en este caso, el cromosoma es una representación vectorial, en donde cada componente del vector constituye un gen y el contenido de cada gen es el alelo. Estos algoritmos fueron desarrollados por Holland (1975).

Cualquier problema de adaptación se puede representar y ser resuelto mediante estos algoritmos, ya que tienen una gran capacidad a la hora de encontrar un óptimo debido a que no está limitado su espacio de búsqueda. Esto es posible gracias a que principalmente se parte de una población inicial de individuos, generada normalmen-

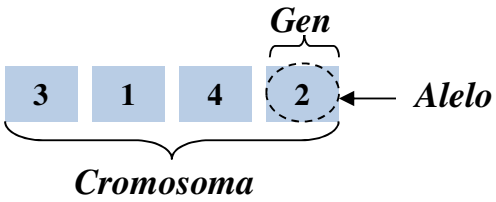


Figura C.1.: Ejemplo de representación de un cromosoma

te de forma aleatoria, a la cual se le va aplicando en cada generación una serie de operaciones, tales como selección de una serie de individuos, cruce y mutación de los mismos. Por último, se hace un reemplazamiento de individuos en la población por aquellos que posean mejores características. Para determinar la bondad de una solución generada, se ha de hacer uso de la *función objetivo*, también denominada *función de adaptación (fitness)*.

En estos algoritmos hemos de destacar que el concepto de explotación está relacionado con el cruce, y el de exploración con la mutación de individuos de una población.

La utilización de algoritmos genéticos aporta ventajas tales como la posibilidad de usar grandes dominios para las variables, capacidad para explorar la gradualidad de funciones con variables continuas. Además, la representación de las soluciones está muy cercana a la formulación natural de muchos problemas.

C.1.1. Funcionamiento general de los AG

Normalmente, se suele representar a cada individuo que compone la población mediante una cadena de longitud determinada, según el problema que deseamos abordar. Se han propuesto varios esquemas algorítmicos para los algoritmos evolutivos. La idea más generalizada sobre el mecanismo de un algoritmo genético (Goldberg (1989); Davis (1991); Chambers (1998)) que trabaja sobre una población P se presenta esquemÃ¡ticamente en el algoritmo 15.

El algoritmo genético trabaja sobre individuos que representan potenciales soluciones al problema, codificados de acuerdo a un mecanismo prefijado. Los individuos son evaluados mediante la función de fitness que tiene en cuenta la adecuación de cada solución al problema que se intenta resolver. La operativa del algoritmo genético

Algoritmo 15 Esquema general de un AG

Entrada: Tamaño de la población, Probabilidad de cruce y mutación
Salida: Mejor Solución Encontrada
  Inicializar(P(0))
  generación ← 0
  Mientras (*no se cumpla la condición de parada*) **hacer**
    Evaluar(P(generación))
    Padres ← Seleccionar(P(generación))
    Hijos ← Aplicar_Operadores_Evolutivos(Padres)
    NuevaPoblación ← Reemplazar(Hijos, P(generación))
    generación = generación + 1
    P(generación) ← NuevaPoblación
  Fin Mientras

comienza con una etapa de inicialización de los individuos, que puede ser completamente aleatoria, muestreando al azar diferentes secciones del espacio de soluciones, o guiada de acuerdo a características del problema a resolver. El algoritmo genético podría incluso tomar como población inicial individuos resultantes como salida de algún otro algoritmo heurístico de resolución que permitiera calcular buenas soluciones iniciales aproximadas para el problema. La evolución propiamente dicha se lleva a cabo en el ciclo que genera nuevos individuos a partir de la población actual mediante un procedimiento de aplicación de operadores estocásticos. En este ciclo se distinguen cuatro etapas:

- *Evaluación:* Etapa que consiste en asignar un valor de adecuación (fitness) a cada individuo en la población. Este valor evalúa que tan bien resuelve cada individuo el problema en cuestión. Se utiliza para guiar el mecanismo evolutivo.

- *Selección:* Proceso que determina candidatos adecuados, de acuerdo a sus valores de fitness, para la aplicación de los operadores evolutivos con el objetivo de engendrar la siguiente generación de individuos.

- *Aplicación de los operadores evolutivos:* Genera un conjunto de descendientes a partir de los individuos seleccionados en la etapa anterior.

- *Reemplazo:* Mecanismo que realiza el recambio generacional, sustituyendo individuos de la generación anterior por descendientes creados en la etapa anterior.

Diversas políticas para la selección y el reemplazo de individuos permiten modificar las características del algoritmo evolutivo. Aplicando políticas adecuadas es posible privilegiar a los individuos más adaptados en cada generación (estrategias de elitismo), aumentar la presión selectiva sobre individuos mejor adaptados, generar un número reducido de descendientes en cada generación (modelos de estado estacionario), y otras muchas variantes.

Los operadores evolutivos determinan el modo en que el algoritmo explora el espacio de soluciones del problema. Una gran diversidad de propuestas de operadores evolutivos han surgido en los cuarenta años de vida de la computación evolutiva. Los diferentes operadores y las particularidades en su modo de aplicación dan características peculiares a las distintas variantes de algoritmos evolutivos. Los operadores de recombinación, que permiten combinar características de dos o más individuos con la idea de obtener descendientes mejor adaptados y los operadores de mutación, que introducen diversidad mediante modificaciones aleatorias, son los operadores evolutivos más difundidos.

La condición de parada de la fase iterativa del algoritmo evolutivo usualmente tiene en cuenta la cantidad de generaciones procesadas, deteniéndose el ciclo evolutivo tras alcanzar un número prefijado de generaciones. Otras alternativas consideran la variación de los valores de fitness (deteniendo el ciclo evolutivo cuando el proceso se estanca y no obtiene mejoras considerables en los valores de fitness) o estimaciones del error cometido respecto al valor óptimo del problema o una aproximación, en caso de conocerse. También, y como ocurre en nuestro caso, se suele dar un tiempo de ejecución al algoritmo.

En la literatura podemos encontrar dos modelos de algoritmos genéticos:

- *Generacional:* Este tipo de algoritmo genético consiste en que en cada generación, se crea una generación intermedia (con igual tamaño que la actual) mediante la selección de padres de la población actual, se aplican los operadores genéticos sobre los individuos de esta población intermedia y los descendientes resultantes pasan a formar la población actual, reemplazando a la antigua. Con todo esto podemos decir que en cada iteración hay una generación nueva. En este modelo, los individuos con mayor probabilidad de selección tienden a tener un mayor número de copias; los descendientes resultantes de aplicar los operadores de cruce y mutación a la población intermedia forman una nueva población.

- *Estacionario:* Este modelo evolutivo escoge un cierto número de padres de la población (normalmente dos) en cada generación usando cualquier mecanismo de selección. A continuación, se aplican los operadores genéticos sobre ellos, y los descendientes (o el único hijo) pugnan por reemplazar a dos cromosomas (un cromosoma) de la población. Este modelo proporciona más intensificación y más convergencia.

C.1.2. Representación de soluciones

Los AG no trabajan directamente sobre las soluciones del problema en cuestión, sino que lo hacen sobre una abstracción de los objetos solución, usualmente denominadas cromosomas por analogía con la evolución natural biológica. Un cromosoma puede ser un vector de genes, mientras que el valor asignado a un gen se denomina alelo. En la terminología biológica, genotipo denota al conjunto de cromosomas que definen las características de un individuo. El genotipo sometido al medio ambiente se denomina fenotipo. En términos de los algoritmos genéticos el genotipo también esta constituido por cromosomas, utilizándose generalmente un único cromosoma por individuo solución al problema. Por ello suelen utilizarse indistintamente los términos genotipo, cromosoma e individuo. Por su parte, el fenotipo representa un punto del espacio de soluciones del problema.

Dado que un AG trabaja sobre cromosomas, se debe definir una función de codificación sobre los puntos del espacio de soluciones, que representa todos puntos del espacio de soluciones en un genotipo. La función inversa de la codificación, denominada decodificación permite obtener el fenotipo asociado a un cromosoma.

Además, los mecanismos de codificación de individuos solución resultan importantes para el proceso de búsqueda de los algoritmos genéticos. Habitualmente los algoritmos genéticos utilizan codificaciones binarias de longitud fija. Los individuos se codifican por un conjunto de cardinalidad conocida de valores binarios (ceros y unos) conocido como string de bits o *bitstring*. Cada *bitstring* representa a una solución potencial del problema de acuerdo al mecanismo de codificación predefinido, en general dependiente del problema.

Existen otros esquemas de codificación como las codificaciones basadas en números reales que son útiles para representar soluciones cuando se resuelven problemas sobre espacios de cardinalidad no numerable. Este es el caso de la determinación de parámetros en problemas de control. Los esquemas basados en permutaciones de enteros son útiles para problemas de optimización combinatoria que involucran hallar

ordenamientos óptimos, como los problemas de scheduling o el conocido problema del Viajante de Comercio.

C.1.3. Función fitness

Todo cromosoma tiene un valor asociado de fitness que evalúa la aptitud del individuo para resolver el problema en cuestión. La función de fitness tiene el mismo tipo que la función objetivo del problema, lo cual implica que el cálculo del valor de fitness se realiza sobre el fenotipo correspondiente al cromosoma. La figura C.2 muestra un esquema de este cálculo.

Figura C.2.: Esquema del cálculo de la función de fitness

La función de fitness tiene una influencia importante en el mecanismo del AG. Si bien actúa como una caja negra para el proceso evolutivo, la función de fitness guía el mecanismo de exploración, al actuar representando al entorno que evalúa la bondad de un individuo solución para la resolución del problema.

C.1.4. Operadores

La gran mayoría de las variantes de algoritmos genéticos utiliza como principales los siguientes operadores: selección, cruce y mutación.

El mecanismo de selección determina el modo de perpetuar buenas características, que se asumen, son aquellas presentes en los individuos más adaptados. El mecanismo de *selección proporcional* o *selección por ruleta* elige aleatoriamente individuos utilizando una ruleta sesgada, en la cual la probabilidad de ser seleccionado es proporcional al fitness de cada individuo. Otros mecanismos de selección introducen diferentes grados de elitismo, conservando un cierto número prefijado de los mejores individuos a través de las generaciones. En el caso de la *selección por torneo*, se escogen aleatoriamente un determinado número de individuos de la población, los cuales compiten entre ellos para determinar cuáles se seleccionarán para reproducirse, de acuerdo a sus valores de fitness. El mecanismo de *selección basado en el*

rango introduce el mayor grado de elitismo posible, al mantener entre generaciones, un porcentaje generalmente elevado, de los mejores individuos de la población.

Estas diferentes políticas de selección, conjuntamente con políticas similares utilizadas para determinar los individuos reemplazados por los descendientes generados, posibilitan el diseño de diferentes modelos evolutivos para los algoritmos genéticos.

Los algunos esquemas de codificación, como el esquema de bits, tienen como ventaja principal la definición operadores evolutivos simples sobre ellos. En la formulación canónica de un algoritmo genético, se propone como operador de recombinación el cruzamiento de un punto, que consiste en obtener dos descendientes a partir de dos individuos padres seleccionando un punto al azar, cortando los padres e intercambiando los trozos de cromosoma (figura C.3).

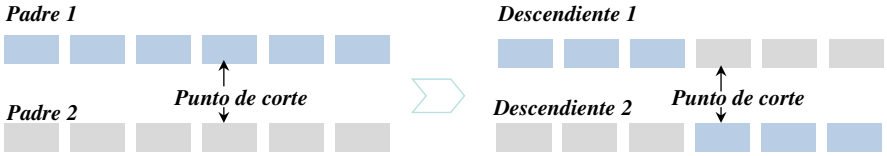
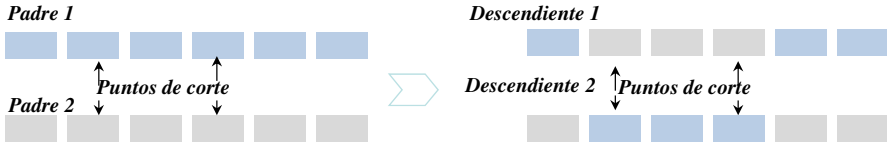


Figura C.3.: Esquema de cruce de un Algoritmo Genético

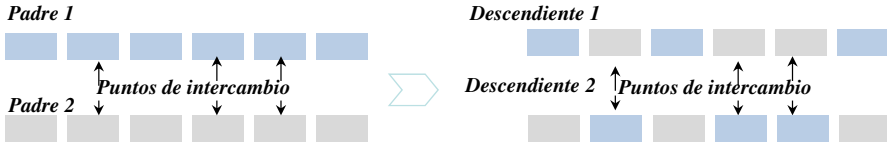
El operador clásico de mutación introduce diversidad en el mecanismo evolutivo, simplemente modificando aleatoriamente uno de los valores de un gen contenido en un cromosoma. Sobre un esquema de codificación binaria, la modificación consiste en invertir el valor binario de un alelo.

Tanto el cruce como la mutación son operadores probabilísticos, de forma que se aplican o no, teniendo en cuenta una tasa de aplicación del operador. Generalmente la tasa de aplicación del operador de cruce es elevada en un algoritmo genético simple (entre 0,5 y 0,9) mientras que la tasa de aplicación del operador de mutación suele ser muy baja (entre 0,1 y 0,2).

Para modificar el comportamiento del mecanismo de exploración del espacio de soluciones se han propuesto operadores evolutivos más complejos como alternativa. Es habitual encontrar operadores de cruce multipunto en donde se utilizan dos (figura C.4(a)) o más puntos de corte o uniformes (figura C.4(b)) en donde para cada posición en el cromosoma se decide intercambiar material genético de acuerdo a una probabilidad prefijada.



(a) Cruce de dos puntos



(b) Cruce uniforme

Figura C.4.: Esquemas de cruce multipunto

C.1.5. Algoritmos genéticos frente a otras técnicas heurísticas

Si bien no existe una opinión unánime sobre la aplicabilidad de los algoritmos genéticos, la amplia gama de problemas resueltos exitosamente en diversas áreas han popularizado a esta técnica evolutiva como una de las más versátiles y robustas.

Los algoritmos genéticos presentan ciertas características que los hacen ventajosos cuando se los compara con otras técnicas de resolución de problemas. La operativa del mecanismo evolutivo de los algoritmos genéticos es independiente de particularidades del dominio, permitiendo definir esquemas genéricos capaces de abordar diferentes clases de problemas. Esta característica, conjuntamente con el hecho de que el mecanismo evolutivo se aplique sobre representaciones, hace a los algoritmos genéticos aplicables a una amplia gama de problemas.

Desde el punto de vista de la resolución del problema, el algoritmo genético funciona como una caja negra que solamente utiliza como dato de entrada la función de fitness definida para el problema con el objeto de evaluar a los individuos en el ciclo evolutivo. No utiliza información adicional sobre las características del problema, aunque ciertas particularidades de la codificación pueden complementar la operativa simple de los operadores evolutivos y dirigir la búsqueda. La independencia del mecanismo de búsqueda evolutivo de las características del problema permite a los algoritmos genéticos evitar, en ocasiones, el estancamiento en mínimos locales del problema en los cuales son proclives a caer ciertos algoritmos tradicionales basados en gradientes u otras búsquedas locales dirigidas.

Otras de las ventajas que presentan los AG con respecto a otras técnicas de optimización clásicas son:

- No requieren que la función objetivo tenga un "buen comportamiento", ya que son capaces de tolerar sin ningún problema discontinuidades o funciones polinomiales.
- Se prestan para implementaciones distribuidas o en paralelo.
- Son simples de implementar, ya que la única información necesaria es la función objetivo y las restricciones correspondientes.

C.2. Algoritmos Multiobjetivo

C.2.1. Conceptos generales

Muchos problemas reales se caracterizan por la existencia de múltiples medidas de actuación, las cuales deberían ser optimizadas, o al menos ser satisfechas simultáneamente. Un ejemplo de esto podría ser el diseño de un sistema de control de aire acondicionado, para el cual tuviéramos que optimizar un conjunto de parámetros, como minimizar el consumo de energía, maximizar el confort de los usuarios, maximizar la estabilidad del sistema de control, etc.

Un problema de optimización multiobjetivo plantea la optimización (minimización o maximización) de un conjunto de funciones, habitualmente en conflicto entre sí (Collette y Siarry (2004); Coello y cols. (2007)). La existencia de múltiples funciones objetivo plantea una diferencia fundamental con un problema en el que sólo obtenemos una solución: no existirá una única solución al problema, sino un conjunto de soluciones que plantearán diferentes compromisos entre los valores de las funciones a optimizar. La formulación general de un problema de optimización multiobjetivo es la siguiente:

Maximizar / Minimizar la función

$$F(x) = (f_1(x), f_2(x), \ldots, f_M(x)) \tag{C.1}$$

sujeto a las funciones:

$$G(x) = (g_1(x), g_2(x), \ldots, g_S(x)) \geq 0$$
$$H(x) = (h_1(x), h_2(x), \ldots, h_g(x)) = 0 \tag{C.2}$$

La solución al problema de optimización multiobjetivo corresponderá a un vector de variables de decisión $x = (x_1, x_2, \ldots, x_N)$ que satisfaga las restricciones impuestas por las funciones G y H, ofreciendo valores que representen un compromiso adecuado para las funciones $f_1, f_2, \ldots, f_M$.

Cabe mencionar que la mayor parte de los problemas de optimización subyacentes a problemas del mundo real tienen una formulación de este tipo, aunque en muchos casos son abordados siguiendo un enfoque en el que solo se obtiene una sola solución.

Considerando el caso de un problema de minimización de funciones, un punto x^* es *óptimo de Pareto* si para todo x en Ω, la región factible del problema, se cumple que $f_i(x) = f_i(x^*)\ \forall i \in 1, \ldots, M$, o para al menos un valor de i se cumple que $f_i(x) > f_i(x^*)$. Esto significa que no existe un vector factible que sea *mejor* que el óptimo de Pareto en alguna función objetivo sin que empeore los valores de alguna de las restantes funciones objetivo.

Asociada con la definición anterior, se introduce una relación de orden parcial denominada dominancia entre vectores solución del problema de optimización multiobjetivo. Un vector $w = (w_1, w_2, \ldots, w_N)$ domina a otro $v = (v_1, v_2, \ldots, v_N)$ si:

$$w_i \leq v_i \forall i \in \{1, \ldots, M\} \wedge \exists i \in \{1, \ldots, M\} / w_i < v_i \tag{C.3}$$

En este caso la relación de dominancia se nota $w \prec v$.

Dado que diferentes valores de las variables de decisión representan diferentes compromisos, la resolución de un problema de optimización multiobjetivo no se concentra en hallar un único valor solución, sino que se plantea hallar un conjunto de soluciones *no dominadas*, de acuerdo a la definición presentada (ecuación C.3).

El conjunto de soluciones óptimas al problema de optimización multiobjetivo se compone de los vectores factibles *no dominados*. Este conjunto se denomina conjunto óptimo de Pareto (P^*) y está definido por:

$$P^* = \{x \in \Omega / \neg x' \in \Omega, f(x') \preceq f(x)\} \tag{C.4}$$

La región de puntos definida por el conjunto óptimo de Pareto en el espacio de valores de las funciones objetivo se conoce como *Frente* o *Frontera del Pareto (FP∗)*. Formalmente, el Frente del Pareto está definido por:

$$FP^* = \{u = (f_1(x), f_2(x), \ldots, f(x_M)/x \in P^*)\} \tag{C.5}$$

La figura C.5 muestra la frontera del Pareto para maximizar dos funciones objetivo.

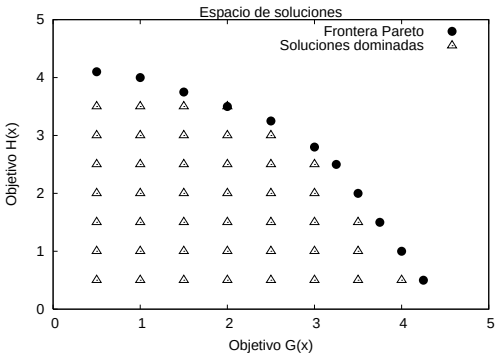


Figura C.5.: Ejemplo de espacio de soluciones en optimización multiobjetivo

La complejidad inherente a los problemas de optimización multiobjetivo plantea un difícil reto para su resolución mediante algoritmos exactos deterministas a medida que crece la dimensión del espacio de soluciones. De este modo, las técnicas clásicas como los métodos exactos de búsqueda local, basados en gradientes o que utilizan las técnicas estándar de programación determinista métodos *greedy*, técnicas de *branch and bound*, etc., si bien pueden ser aplicables para problemas de complejidad reducida, exigen un costo computacional excesivo para la resolución de problemas multiobjetivo complejos con aplicación en el mundo real.

En este contexto, se propusieron técnicas heurísticas estocásticas como alternativa para la resolución de problemas de optimización multiobjetivo, con el objeto de alcanzar soluciones aproximadas de buena calidad en tiempos de cómputo razonables. Enmarcados en esta categoría, las técnicas de computación evolutiva se han manifestado como métodos robustos y efectivos para resolución de los problemas de optimización multiobjetivo y se han popularizado como consecuencia de su éxito.

C.2.2. Algoritmos Evolutivos Multiobjetivo

Los *Algoritmos Evolutivos para Optimización Multiobjetivo (AEOM)* (Deb (2001)) surgen como una extensión de los Algoritmos Evolutivos (AE) para problemas que obtienen más de una solución, utilizando fundamentalmente varios conceptos relacionados con el tratamiento de funciones multimodales por parte de los AE en los que sólo se obtiene una solución.

Los AE basan su funcionamiento en la simulación del proceso de evolución natural. Consisten en una técnica iterativa que aplica operadores estocásticos sobre un conjunto de individuos la población con el propósito de mejorar su *fitness*, una medida relacionada con la función objetivo del problema en cuestión. Cada individuo de la población representa una solución potencial del problema, codificada de acuerdo a un esquema de representación, generalmente basado en números binarios o reales.

Inicialmente la población se genera de forma aleatoria y luego evoluciona mediante la aplicación iterativa de interacciones denominadas operadores de reproducción. Esta evolución es guiada por una estrategia de selección de los individuos más adaptados a la resolución del problema, de acuerdo a sus valores de fitness.

Dado que trabajan en paralelo sobre un conjunto de soluciones, los AE tienen la potencialidad de tratar problemas con objetivos múltiples, hallando en cada ejecución un conjunto de soluciones aproximadas al frente del Pareto. Esto representa una importante ventaja respecto a los algoritmos tradicionales, que solamente generan una solución por ejecución. Además, los AE tienen otras ventajas respecto a los algoritmos tradicionales, como ser menos sensibles a la forma o a la continuidad del frente del Pareto o permitir abordar problemas con espacio de soluciones de gran dimensión.

Un AEOM debe diseñarse para lograr dos propósitos en forma simultánea: lograr buenas aproximaciones al frente del Pareto y mantener la diversidad de las soluciones. Debe de muestrear adecuadamente el espacio de soluciones y no converger a una solución única o a una sección acotada del frente.

El mecanismo evolutivo de los AE permite lograr el primer propósito, mientras que para preservar la diversidad los AEOM utilizan las técnicas de nichos, sharing, crowding o similares, utilizadas tradicionalmente por los AE en la optimización de funciones multimodales.

El algoritmo 16 describe, sobre una población P, dos operadores característicos de un AEOM, que no aparecen en la estructura genérica de un AE. El *operador de diversidad* aplica la técnica utilizada para evitar la convergencia a un sector del

frente del Pareto (nichos, fitness sharing, crowding, etc.). Asimismo, se incluye un procedimiento de *asignación de fitness*, orientado a dar mayor oportunidad a aquellos individuos con mejores características, considerando los valores de las funciones objetivo y los resultados de la métrica utilizada para evaluar la diversidad.

Algoritmo 16 Esquema general de un AEOM

Entrada: Tamaño de la población, probabilidad de cruce y mutación.
Salida: Mejor solución hallada
```
Inicializar(P(0))
generación ← 0
Evaluar(P(0))
Mientras (no se cumpla la condición de parada) hacer
    Operador de diversidad(P(generación))
    Asignar_fitness(P(generación))
    Padres ← Seleccionar(P(generación))
    Hijos ← Aplicar_Operadores_Evolutivos(Padres)
    NuevaPoblación ← Reemplazar(Hijos, P(generación))
    generación ← generación + 1
    P(generación) ← NuevaPoblación
    Evaluar(P(generación))
Fin Mientras
```

La computación evolutiva utiliza los modelos de cómputo de procesos evolutivos como elementos dominantes en el diseño y puesta en práctica en los sistemas de computación para solucionar un problema. Hay una variedad de modelos evolutivos que se han propuesto y estudiado y que se refieren a los AE. Concretamente, cuatro bien definidos AE que han servido como base en este campo: AG, estrategias de evolución, programación genética y programación evolutiva.

Un AE mantiene una población de soluciones temporales, impone cambios al azar a estas soluciones e incorpora la selección para determinar si van a ser mantenidas en las generaciones futuras o no. Pero hay también importantes diferencias entre ellos. Los AG acentúan los modelos de operadores genéticos según lo observado dentro de la naturaleza, tal como el cruce y la mutación, y aplica éstos a los cromosomas abstraídos con diferente representación según el problema que estemos considerando.

Los AE son muy apropiados para solucionar problemas multiobjetivo. Gracias al uso de una población de soluciones, los AE pueden buscar muchas soluciones Pareto-óptimas en la misma ejecución. Generalmente, los AEOM solamente difieren del resto de AE en la función de fitness y/o en el mecanismo de la selección.

Los acercamientos evolutivos en la optimización multiobjetivo pueden ser clasi-

ficados en tres grupos:

1. Acercamientos agregando planos.
2. Acercamientos basados en población sin Paretos.
3. Acercamientos basados en Paretos.

El primer grupo se corresponde con los llamados modelos evolutivos utilizando pesos para la agregación de objetivos. El segundo y tercer grupo entran dentro del grupo de modelos evolutivos que generan poblaciones de soluciones no dominadas.

El primer grupo constituye la extensión de métodos clásicos a AE. Los objetivos están combinados, o agregados, en una función escalar según una cierta comprensión del problema, y entonces el AE se aplica de la manera general. La optimización de una combinación de los objetivos tiene la ventaja de producir una sola solución pero tiene los siguientes inconvenientes:

- Puede ser difícil definir la combinación de pesos para que se obtengan soluciones aceptables.
- Si la solución óptima generada no puede finalmente ser validada, una nueva ejecución del AE puede ser requerida hasta que se encuentre una solución conveniente.

Los modelos evolutivos que generan poblaciones de soluciones no dominadas permiten que explotemos las características especiales del AE.

Los acercamientos basados en Paretos parecen ser el área más activa de la investigación en los AEOM. Promueven la generación de soluciones no dominadas múltiples, pero hacen uso directamente de la definición del Pareto óptimo.

Las soluciones generadas se comparan por medio de la relación de dominancia existente entre ellas. Se definen grupos de equivalencia de diferentes dependiendo de la dominancia de sus individuos y de esos individuos que pertenecen a las clases prometedoras (esos grupos incluyendo los individuos que dominan una gran cantidad del resto) se les asigna una probabilidad más alta de selección que a los individuos del grupo de las clases menos prometedoras.

La diferencia entre el primer y segundo grupo en los modelos evolutivos que generan poblaciones de soluciones no dominadas se presenta en el uso del elitismo. Los algoritmos incluidos dentro del primer grupo (acercamientos de población basados

sin Paretos), por ejemplo Niched Pareto Genetic Algorithm (NPGA, Horn y cols. (1994)), Non-dominated Sorting Genetic Algorithm (NSGA, Srinivas y Deb (1995)) y Multiple-Objective Genetic Algorithm (MOGA, Murata y Ishibuchi (1995)), que no presenta esta característica. Por otra parte, segundo grupo en los modelos evolutivos que generan poblaciones de soluciones no dominadas (acercamientos de población basados en Paretos) se basan en la consideración de una población auxiliar donde las soluciones no dominadas generadas entre las generaciones diferentes se salvan. Los ejemplos de esta familia son Strength Pareto Evolutionary Algorithm (SPEA, Zitzler y Thiele (1998)), SPEA-II (Zitzler y cols. (2001)) y NSGA-II, entre otros.

C.2.3. Nondominated Sorting Genetic Algorithm II (NSGA-II)

La presencia de múltiples objetivos en un problema da lugar a un conjunto de soluciones Pareto-óptimas, en lugar de una sola solución óptima. En ausencia de más información, una de estas soluciones Pareto-óptimas no se puede considerar mejor que otra. Esto exige encontrar tantas soluciones Pareto-óptimas como sea posible. Los métodos clásicos de optimización (incluyendo los métodos de toma de decisiones de múltiple criterio) sugieren convertir el problema de optimización multiobjetivo en un problema de optimización de un solo objetivo por medio de enfatizar solamente una solución Pareto-optima al mismo tiempo. Cuando un método tiene que usarse para encontrar múltiples soluciones, tiene que aplicarse muchas veces, y es de esperar que se obtengan resultados diferentes en cada simulación.

Los AEOM permiten encontrar múltiples soluciones Pareto-óptimas en una sola simulación. El algoritmo NSGA (Srinivas y Deb (1995)) fue uno de los primeros AEOM, pero fue criticado por Deb y cols. (2002):

- La alta complejidad computacional de organización de no dominancia: el algoritmo de ordenación de soluciones no dominadas que se usa tiene una complejidad computacional de $O(MN^3)$ (donde M es el número de objetivos y N es el tamaño de la población). Ello hace que NSGA sea computacionalmente costoso para poblaciones de gran tamaño ya que esta ordenación se realiza en cada generación.

- Ausencia de elitismo: El elitismo puede acelerar el rendimiento del AG de forma significativa, lo cual puede ayudar a prevenir la pérdida de buenas soluciones una vez que se han encontrado.

- Necesidad de especificar el parámetro de reparto: Los mecanismos tradicionales que aseguran la diversidad en una población como para conseguir una amplia variedad de soluciones equivalentes se basan en el concepto de reparto. El principal problema del reparto es que requiere la especificación de un parámetro. Aunque el parámetro de reparto se estudió, es conveniente implementar un mecanismo para preservar menor diversidad.

Por ello, surgió una versión mejorada del NSGA, que se denomina NSGA-II. El NSGA-II es una buena apuesta para encontrar un conjunto diverso de soluciones y para converger cerca del Pareto-óptimo.

El proceso de ordenación de NSGA-II parte de un enfoque simple. Para identificar las soluciones de la primera frontera del Pareto en una población de tamaño N, cada solución se compara con todas las soluciones de la población para saber si es dominada. Para encontrar los individuos no dominados de la siguiente frontera, se repite el mismo proceso pero sin considerar las soluciones de la primera frontera. Este proceso se repite para niveles de no dominancia más altos.

Para cada solución se calcula un contador de dominancia (n_p), que es el número de soluciones que dominan a la solución p, y un conjunto de soluciones (S_p) dominadas por la solución p.

Todas las soluciones en la primera frontera de no dominados tiene su contador n_p a cero. A continuación, por cada solución p con $n_p = 0$ se compara con cada miembro q de su conjunto S_p que reduce su dominancia incrementando el contador en uno. El contador n_p es cero en algún miembro q, entonces q se sitúa en un grupo separado Q. Los miembros de este grupo pertenecen a la segunda frontera de no dominados. Después se sigue el mismo procedimiento con cada miembro del grupo Q para identificar la tercera frontera. El proceso continúa hasta que se identifican todas las fronteras. El algoritmo 17 muestra el procedimiento de ordenación de las soluciones no dominadas.

Con respecto a la preservación de la diversidad, los autores proponen una mejora basada en la cantidad de soluciones que no ha de ser establecida por el usuario. Esta nueva métrica consiste en una estimación de la densidad de la soluciones y un operador de comparación de densidad. Para estimar la densidad de las soluciones alrededor de una determinada solución en la población, se calcula la distancia media de dos puntos a ambos lados de esta solución para cada objetivo. Este valor $i_{distance}$ es un estimador de perímetro del cuboide formado por los vecinos más cercanos como vértices. A esta distancia se le denomina *distancia de crowding*. La figura C.6 ilustra

Algoritmo 17 Ordenación de las soluciones no dominadas de NSGA-II

Entrada: P población
Salida: P población ordenada
Para cada $p \in P$ **hacer**
$S_p = 0$
$n_p = 0$
Para cada $q \in P$ **hacer**
{Si p domina a q}
Si $(p \prec q)$ **entonces**
$S_p = S_p \cup \{q\}$ {Añade q al conjunto de soluciones dominadas por p}

Sino
Si $(q \prec p)$ **entonces**
{Incrementa el contador de dominancia de p}
{p pertenece a la primera frontera}
$n_p = n_p + 1$
Fin Si
Fin Si
Fin Para
Si $(n_p == 0)$ **entonces**
$p_{rank} = 1$
$F_1 = F_1 \cup \{p\}$
Fin Si
Fin Para
$i = 0$ {Inicialización del contador de la frontera}
Mientras $(F_i \neq \emptyset)$ **hacer**
$Q = \emptyset$ {Para almacenar los miembros de la siguiente frontera}
Para cada $p \in F_i$ **hacer**
Para cada $q \in S_p$ **hacer**
$n_q = n_q - 1$
Si $(n_q == 0)$ **entonces**
$q_{rank} = i + 1$ {Q pertenece a la siguiente frontera}
$Q = Q \cup \{q\}$
Fin Si
Fin Para
Fin Para
$i = i + 1$
$F_i = Q$
Fin Mientras

el cálculo de la distancia de crowding.

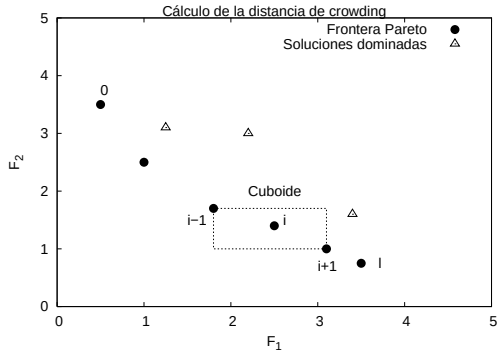


Figura C.6.: Cálculo de la distancia de crowding

Para calcular la distancia de crowding la población se ordena según el valor de cada función objetivo de forma ascendente. Después, por cada función objetivo, a las soluciones pertenecientes a la frontera se les asigna un valor de distancia infinito y las soluciones intermedias son inicializadas a una distancia igual al valor de a diferencia normalizada absoluta en los valores de la función de dos soluciones adyacentes. Posteriormente, se continúa el cálculo con el resto de funciones objetivo. El valor de la distancia de crowding general se calcula como la suma de los valores de la distancia individual correspondientes a cada objetivo. Cada función objetivo se normaliza antes de calcular la distancia de crowding. El algoritmo 18 muestra el procedimiento de cálculo de la distancia de crowding de todas las soluciones en un conjunto I de no dominados.

El operador de comparación de crowding ($\prec_n$) guía el proceso de selección hacia la frontera del Pareto óptimo. Considera que cada individuo i de la población tiene dos atributos: 1) el grado de no dominancia (i_{grado}); 2) la distancia de crowding ($i_{distancia}$). NSGA-II define el orden parcial $\prec_n$ como:

$$i \prec_n j \text{ si } (i_{grado} < j_{grado}) \text{ o } ((i_{grado} == j_{grado}) \text{ y } (i_{distancia} > j_{distancia}))$$

Es decir, entre dos soluciones con diferente grado de no dominancia, la mejor es la que tiene un menor grado. En otro caso, si ambas soluciones pertenecen a la misma frontera, la mejor solución es la que está en una región con menor distancia de crowding.

Algoritmo 18 Cálculo de la distancia de crowding

Entrada: I conjunto de soluciones no dominadas
Salida: Distancia de crowding para cada solución
  $l = |I|$ {Número de soluciones en I}
  Para cada i **hacer**
    Establecer $I[i]_{distancia} = 0$ {Inicializa la distancia}
  Fin Para
  Para cada objetivo m **hacer**
    $I = ordenar(I, m)$ {Ordena usando cada valor objetivo}
    $I[1]_{distancia} = I[l]_{distancia} = \infty$ {Para seleccionar siempre los valores de la frontera}
    Para $i = 2$ hasta $(l-1)$ **hacer**
      {Para el resto de puntos}
      $I[i]_{distancia} = I[i]_{distancia} + (I[i+1].m - I[i-1].m)/(f_m^{max} - f_m^{min})$
    Fin Para
  Fin Para

Dados los conceptos anteriores, a continuación pasamos a describir el algoritmo NSGA-II. El algoritmo parte de una población inicial $P(0)$ generada aleatoriamente y se ordena en base al criterio de no dominancia. A cada solución se le asigna el fitness (o grado) igual a su nivel de no dominancia (1 es el mejor nivel, 2 es el siguiente mejor nivel, y así sucesivamente). En este caso, asumimos que minimizamos la función de fitness. El algoritmo comienza con una seleción por torneo binario y los operadores genéticos de cruce y mutación para crear una población de descendientes Q_0 de tamaño N. El elitismo se hace latente mediante la comparación de la población actual con las mejores soluciones no dominadas encontradas, aunque el procedimiento es diferente para las sucesivas generaciones. El algoritmo 19 muestra la generación de las sucesivas generaciones a partir de la generación de la población inicial.

Como se puede apreciar, se forma una población combinada $R_t = P_t \cup Q_t$ de tamaño $2N$ y se ordena según el criterio de no dominancia. De esta forma se asegura el elitismo. Ahora, las soluciones pertenecientes al mejor conjunto de no dominados F_1 son las mejores soluciones de la combinación de poblaciones. Si el tamaño de F_1 es menor que N, se eligen todos los miembros de F_1 para formar la población P_{t+1}. El resto de miembros de la población P_{t+1} se eligen de la subsiguiente frontera según su grado. A continuación, se eligen las soluciones del conjunto F_2, posteriormente del conjunto F_3, y así sucesivamente. El procedimiento continua hasta que no haya mas conjuntos. Para elegir exactamente N miembros de la población, se ordenan las soluciones de la frontera F_l mediante el operador de comparación $\prec_n$ en orden descendente. Seguidamente, la nueva población P_{t+1} de tamaño N se usa para selección

Algoritmo 19 Algoritmo NSGA-II

Entrada: P población inicial ordenada con criterio de no dominancia, Q población generada a partir de P
Salida: Pareto óptimo de soluciones
$R_t = P_t \cup Q_t$ {Combinar las poblaciones de padres y descendientes}
F=Ordenación_criterio_no_dominancia(R_t) {$F = (F_1, F_2, \dots)$, fronteras no dominadas de R_t}
$P_{t+1} = \emptyset$
$i = 1$
{Hasta que la población de padres se rellene}
Mientras $|P_{t+1}| + |F_i| \leq N$ **hacer**
  Asignación_distancia_crowding(F_i) {Calcular la distancia de crowding en F_i}
  $P_{t+1} = P_{t+1} \cup F_i$ {Incluir la i-ésima frontera no dominada en la población de padres}
  $i = i + 1$ {Comprobar la siguiente frontera para la inclusión}
Fin Mientras
Ordenar($F_i, \prec_n$) {Ordenar descendentemente usando $\prec_n$}
$P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ {Elegir los primeros $(N - |P_{t+1}|)$ elementos de F_i}
Q_{t+1}=Nueva_población(P_{t+1}) {Selección, cruce y mutación para crear una nueva población Q_{t+1}}
$t = t + 1$ {Incrementar el contador de generaciones}

(por torneo binario), cruce y mutación de individuos, creando una nueva población Q_{t+1} de tamaño N.

En resumen, NSGA-II incorpora elitismo. La ordenación de cada individuo está basada en el criterio de no dominancia. Además, introduce un algoritmo más rápido de ordenación. Se evalúa una distancia de crowding, considerando el tamaño del mayor paralelogramo que circunda cada individuo. Este parámetro mantiene la diversidad en la población, evitando el uso del factor de reparto.

D. Modelizado Difuso

Saber que se sabe lo que se sabe y que no sabe lo que no se sabe; he aquí el verdadero saber.

CONFUCIO (551 A.C.–479 A.C.), PENSADOR CHINO

Este apéndice pone de manifiesto la metodología para realizar un modelizado mediante Sistemas Basados en Reglas Difusas y su hibridación con técnicas de optimización.

D.1. Sistemas Basados en Reglas Difusas

Los Sistemas Basados en Reglas Difusas (SBRD) se asientan en el conocimiento del sentido común, es decir, en un conocimiento intuitivo (inductivo), fácil y rápido de obtener y estructurar (Adnan y cols. (2015)). Podemos definir un sistema difuso como una caja gris (Babuška (1998)) que es capaz de inferir una salida dadas unas variables de entrada. Para realizar esta inferencia, se han introducido un conjunto de metodologías inteligentes que atienden a sistemas biológicos o a la inteligencia humana, como por ejemplo reglas, redes semánticas o modelos cualitativos. De esta forma se obtiene una solución aproximada e interpretable al problema en la medida de lo posible.

El modelo de representación de las variables se realiza de forma que cada variable toma un conjunto de valores cualitativos que poseen una interpretación lingüística. Cada valor que toma la variable se denomina etiqueta o término lingüístico. El significado de los términos lingüísticos por cada variable de entrada y salida constituyen un conjunto difuso, concretamente la función de pertenencia. Las funciones de pertenencia establecen una correspondencia entre las variables numéricas de las

variables de entrada y salida y las variables difusas de las reglas (L. Zadeh (1965)). Pueden ser triangulares, trapezoidales, gaussianas, sigmoidales o polinomiales.

Para modelizar el conocimiento dentro de un sistema difuso, se establece una relación entre las variables de entrada y salida al sistema mediante las reglas difusas que pueden ser de dos tipos (Alcalá y cols. (2000)): Mamdani y Takagi-Sugeno-Kang (TSK). Ambos tipos de reglas difusas permiten representar tanto problemas de clasificación como de regresión y se basan en un esquema SI-ENTONCES. La parte del SI se denomina antecedente y la parte del ENTONCES es el consecuente de la regla.

Las reglas de tipo TSK se caracterizan por tener en la parte del consecuente una ecuación de salida lineal. Por ejemplo:

$$\textbf{SI } X(t) \text{ es } A \textbf{ ENTONCES } Z \text{ es } Y=k_{io} + k_{i1}X(t)$$

donde A es un conjunto difuso, X son los atributos que se pueden medir del sistema, Z son los atributos controlables del sistema e Y es una ecuación de salida lineal. Es evidente que en este tipo de reglas la representación de los datos es principalmente cuantitativa.

Las reglas de tipo Mamdani carecen de una ecuación de salida lineal en la parte del consecuente. A continuación mostramos un ejemplo de reglas Mamdani para un sistema de control del nivel de agua de un tanque:

SI el *nivel de agua* es muy bajo **ENTONCES** *abrir bastante la válvula*
SI el *nivel de agua* es bajo **ENTONCES** *abrir poco la válvula*
SI el *nivel de agua* es medio **ENTONCES** *no abrir ni cerrar la válvula*
SI el *nivel de agua* es alto **ENTONCES** *cerrar un poco la válvula*
SI el *nivel de agua* es muy alto **ENTONCES** *cerrar bastante la válvula*

En estas reglas encontramos *nivel de agua* como variable de entrada y la *apertura del tanque* como variable de salida del sistema difuso. Como se puede apreciar, este tipo de reglas nos permite una mejor representación cualitativa de la información. La principal ventaja de las reglas de tipo Mamdani frente a las TSK es que son más interpretables.

Por otro lado, para representar un modelo basado en datos cuantitativos en un modelo basado en datos cualitativos, como son las funciones de pertenencia, necesitamos implementar dos operaciones: 1) Fuzzificación, que es la conversión de una

variable real en un grado de pertenencia correspondiente a un término lingüístico del conjunto difuso; y 2) Defuzzificación, que realiza la operación inversa a la fuzzificación.

Para saber si un modelo difuso es representativo, hemos de tener en cuenta dos métricas (Casillas, Cordón, y cols. (2003); Alcalá y cols. (2006)): la interpretabilidad, que nos indica la habilidad del sistema difuso para expresar el comportamiento del sistema de manera razonable e inteligible, y la precisión, que se refiere al grado de representación del sistema modelizado. Destacar que la precisión está íntimamente relacionada con las funciones de pertenencia, los términos lingüísticos y las reglas. Para que el sistema difuso sea viable, debe existir un equilibrio entre interpretabilidad y precisión debido a que son magnitudes inversamente proporcionales: a mayor precisión, menor interpretabilidad del sistema difuso, y viceversa.

Además existe otro factor importante que influye directamente en la interpretabilidad del sistema. Este factor determinante es el número de variables que contiene el problema que vamos a resolver. Un problema con un número considerable de variables de entrada reduce la interpretabilidad del sistema, ya que en un sistema difuso se traduce como un incremento exponencial de número de reglas además de un aumento de la complejidad a nivel de regla.

No obstante, en la literatura del aprendizaje de reglas podemos encontrar métodos híbridos que combinan métodos difusos con otras estrategias, como por ejemplo algoritmos evolutivos. Por ejemplo, los algoritmos evolutivos se suelen usar para optimizar una base de reglas difusas o para buscar el espacio de bases de reglas potenciales de forma sistemática.

D.1.1. Sistemas Basados en Reglas Difusas Lingüísticos

Un Sistema Basado en Reglas Difusas Lingüístico es un SBRD se compone principalmente de:

- Una Base de Conocimiento (BC), formada por un conjunto de reglas difusas con el objetivo de guiar el comportamiento del mismo.
- Una interfaz de fuzzificación, para transformar los datos de entrada precisos en valores para ser utilizados en el proceso de razonamiento difuso.
- Un sistema de inferencia, que emplea estos valores y la información contenida en la base para llevar a cabo dicho proceso de inferencia.

- Una interfaz de defuzzificación, que transforma la acción difusa resultante del proceso de inferencia en una acción precisa que constituye la salida global del SBRD.

En las siguientes subsecciones, analizaremos detalladamente cada uno de estos componentes.

D.1.2. La Base de Conocimiento

La BC es la parte más importante del SBRD ya que el resto de componentes interpretan y usan las reglas para la resolución del problema que estemos considerando. Está formada por:

- Una Base de Reglas (BR), constituida por un conjunto de reglas lingüísticas de tipo SI-ENTONCES que, en el caso de los SBRD con múltiples entradas y una sola salida, presentan la siguiente estructura:

  R_1 : SI X_1 es A_{11} $y \ldots y$ X_n es A_{1n} ENTONCES Y es B_1
  ADEMÁS
  R_2 : SI X_1 es A_{21} $y \ldots y$ X_n es A_{2n} ENTONCES Y es B_2
  ADEMÁS
  $\ldots$
  ADEMÁS
  R_m : SI X_1 es A_{m1} $y \ldots y$ X_n es A_{mn} ENTONCES Y es B_m

  donde X_i e Y son variables lingüísticas de entrada y salida respectivamente, y A_{ij} y B_i son etiquetas lingüísticas asociadas con conjuntos difusos que determinan su semántica en cada una de las reglas. La BR está compuesta por una serie de reglas de este tipo unidas mediante el operador ADEMÁS, lo que indica que todas las reglas pueden dispararse ante una entrada concreta.

- Una Base de Datos (BD) que almacena la definición de los conjuntos difusos asociados a los términos lingüísticos empleados en las reglas de la BR, así como los valores de los factores de escala que realizan las transformaciones necesarias para trasladar los universos de discurso en los que están definidos

dichos conjuntos a aquellos en que se definen las variables de entrada y salida del sistema.

D.1.3. La Interfaz de Fuzzificación

La *interfaz de fuzzificación* permite al SBRD lingüístico trabajar con entradas y salidas reales. Su tarea es la de establecer una correspondencia entre cada valor preciso del espacio de entrada y un conjunto difuso definido en el universo de discurso de dicha entrada. Así, la *interfaz de fuzzificación* trabaja del siguiente forma:

$$A' = F(x_0),$$

donde x_0 es un valor preciso de entrada al SBRD definido en el universo de discurso U, $A^{'}$ es un conjunto difuso definido sobre el mismo dominio y F es un operador de fuzzificación.

El conjunto difuso $A^{'}$ puede construirse al menos de dos maneras diferentes (Driankov y cols. (1993)). En el primero de los casos, A' se construye como un conjunto difuso puntual con soporte en x_0, es decir, con la siguiente función de pertenencia:

$$\mu_{A'}(x) \begin{cases} 1, & \text{si } x = x_0 \\ 0, & \text{en otro caso,} \end{cases}$$

La otra posibilidad comprende la fuzzificación no puntual o aproximada. En ella, $A'(x_0) = 1$ y el grado de pertenencia de los valores restantes de U va disminuyendo según se alejan de x_0. Este tipo de operador de fuzzificación permite el uso de distintos tipos de funciones de pertenencia, por ejemplo, en el caso de una función de pertenencia triangular:

$$\mu_{A'}(x) \begin{cases} 1 - \frac{x - x_0}{\varepsilon}, & \text{si } |x - x_0| \leq \varepsilon \\ 0, & \text{en otro caso.} \end{cases}$$

D.1.4. El Sistema de Inferencia

El *sistema de inferencia* lleva a cabo el proceso de inferencia difuso. Para ello, se hace uso de los principios de la Lógica Difusa para establecer una relación entre los conjuntos difusos definidos en $U = U_1 \times U_2 \times ... \times U_m$ y conjuntos difusos definidos en V, correspondientes a los dominios de las variables de entrada y salida,

respectivamente.

El proceso de inferencia difusa está basado en la aplicación del *modus ponens generalizado*, extensión del modus ponens de la lógica clásica, propuesto por Zadeh según la siguiente expresión (L. A. Zadeh (1973)):

$$\frac{\begin{array}{l}\text{SI } X \text{ es } A \text{ ENTONCES } Y \text{ es } B \\ X \text{ es } A'\end{array}}{Y \text{ es } B'}$$

Para llevar implementar esta expresión, es necesario interpretar el tipo de regla que emplea el SBRD. Una regla con la forma: SI X es A ENTONCES Y es B, puede ser representada como una relación difusa entre A y B definida en $U \times V$. Dicha relación se expresa mediante el conjunto difuso R cuya función de pertenencia $\mu_R(x,y)$ presenta la siguiente forma:

$$\mu_R(x,y) = I(\mu_A(x), \mu_B(y)),\ \forall x \in U,\ y \in V,$$

donde $\mu_A(x)$ y $\mu_B(y)$ son funciones de pertenencia de los conjuntos difusos A y B, respectivamente, e I es un operador de implicación difuso que modela la relación difusa existente.

Dada una relación R, la función de pertenencia del conjunto difuso B' resultante de la aplicación del modus ponens generalizado, se obtiene a partir de la *regla composicional de inferencia* introducida por L. A. Zadeh (1973) del siguiente modo: «Si R es una relación difusa definida de U a V y A' es un conjunto difuso definido en U, entonces el conjunto difuso B', inducido por A', viene dado por la composición de R y A', esto es:

$$B' = A' \circ R,$$

donde A' juega el papel de una relación unaria» (Driankov y cols. (1993)).

De este modo, la aplicación de la regla composicional de inferencia sobre reglas que implícitamente denotan una relación difusa, toma la forma de la siguiente expresión:

$$\mu_{B'}(y) = Sup_{x \in U} \left\{ T(\mu_{A'}(x), I(\mu_A(x), \mu_B(y))) \right\},$$

donde T es un operador de conjunción de la familia de las *t-normas* asociado al

operador de implicación I.

El diseño de un mecanismo de inferencia requiere de la elección de los distintos operadores que intervendrán en dicho proceso, esto es, definir:

- *el operador que permite realizar la intersección* de conjuntos difusos propios de las premisas de las reglas difusas. Para ello se dispone de distintos operadores pertenecientes a la familia de las funciones denominadas *t-normas* (Gupta y Qi (1991); Trillas, Valverde, Kracpryzk, y Yager (1985)).

- *la implicación difusa*, correspondiente al condicional de las reglas ENTONCES. En su inicio, Mamdani empleó la t-norma *mínimo* (Mamdani (1974)), y en consecuencia varios operadores de esa familia han sido aplicados *a posteriori* (Gupta y Qi (1991)). Por otro lado, la familia de funciones de implicación difusa ofrece una amplia variedad de operadores clasificados en distintos grupos, dependiendo del modo en que interpretan la implicación difusa (Trillas, Valverde, Kracpryzk, y Yager (1985)). Además de las mencionadas funciones, otros autores sugieren el uso de *t-conormas* y operadores externos a las anteriores definiciones (Cao y Kandel (1989); Cordón y cols. (1997); Kiszka y cols. (1985)). En los trabajos de Alsina y Trillas (1992), Trillas, Valverde, Kracpryzk, y Yager (1985) y Trillas, Valverde, Gupta, y cols. (1985) se pueden encontrar estudios específicos de la implicación difusa.

- *la agregación ADEMÁS*, que se emplea para combinar las distintas salidas individuales en una final en conjunción con un método de defuzzificación. La composición de este operador depende del tipo de defuzzificación que el SBRD emplee. En la siguiente subsección, se describirán detalladamente tanto este operador como el método de defuzzificación asociado.

D.1.5. La Interfaz de Defuzzificación

Dado que el proceso de inferencia difusa se aplica a nivel de reglas individuales, una vez aplicada la inferencia sobre las m reglas que componen la BR, se obtienen m conjuntos difusos B'_i que representan las acciones difusas que ha deducido el SBRD a partir de las entradas que recibió.

La salida del sistema ha de ser precisa, por lo que la interfaz de defuzzificación debe asumir la tarea de agregar la información aportada por cada uno de los conjuntos difusos individuales y transformarla en un valor preciso. Existen dos formas

diferentes de implementar esta agregación (Bárdossy y Duckstein (1995); Cordón y cols. (1997); L. X. Wang (1994)):

1. *Modo A: agregar primero, defuzzificar después*. En este caso, la interfaz de defuzzificación lleva a cabo las siguientes tareas:

   - Agrega los conjuntos difusos individuales inferidos B'_i, para obtener un conjunto difuso final B', empleando para ello un *operador de agregación difuso* G que, modelando el operador *ADEMÁS*, relaciona las reglas de la base:
   $$\mu_{B'}(y) = G\left\{\mu_{B'_i}(y), ..., \mu_{B'_m}(y)\right\}.$$
   - Mediante un *método de defuzzificación* D, transforma el conjunto difuso B' obtenido en un valor preciso y_0, que será proporcionado como salida global del sistema:
   $$y_0 = D(\mu_{B'}(y)).$$

2. *Modo B: defuzzificar primero, agregar después.* Este modo considera individualmente la contribución de cada conjunto difuso inferido y el valor preciso final se obtiene mediante una operación (una media, una suma ponderada o la selección de uno de ellos, entre otras) sobre un valor preciso característico de cada uno de los conjuntos difusos individuales.

   Así, se evita el cálculo del conjunto difuso final B', lo que permite ahorrar bastante tiempo computacional. Este modo de operación supone una aproximación distinta al concepto representado por el operador *ADEMÁS*.

Inicialmente fue propuesto el *Modo A*, el cual fue empleado por Mamdani en su primera aproximación al control difuso (Mamdani (1974)). En los últimos años, la modalidad *B* ha sido muy empleada, sobre todo en sistemas de tiempo real, donde se requieren tiempos de respuesta rápidos.

Cuando de trabaja en *Modo A*, la función del operador de agregación *ADEMÁS* es unir todos los conjuntos difusos, resultantes de la inferencia de cada regla, en un único conjunto difuso global. Para definir matemáticamente este operador, se emplean distintos operadores, principalmente t-normas y t-conormas, los cuales están descritos y analizados en detalle por Bárdossy y Duckstein (1995).

En cuanto a la definición matemática del método de defuzzificación a emplear para transformar el conjunto difuso global resultante del proceso de inferencia en un valor preciso de salida, encontramos que los más habituales cuando se trabaja en *Modo A* son: el *centro de gravedad*, el *centro de sumas* (aproximación al centro de gravedad computacionalmente más rápida de obtener) y la *media de los máximos* (Driankov y cols. (1993)).

Por otro lado, en caso de emplear el *Modo B*, los *operadores de agregación* más utilizados son la *media*, la *media ponderada* o la selección de algún *valor característico* de los conjuntos difusos en función del grado de importancia de la regla que los ha generado en el proceso de inferencia (Cordón y cols. (1997)). Como métodos para extraer valores representativos se suelen emplear el *centro de gravedad* y el *punto de máximo criterio*; y como grados de importancia de la regla, el *área* y la *altura del conjunto difuso* inferido o el *grado de emparejamiento de los antecedentes* de la misma con la entrada al sistema. El operador más empleado dentro de este grupo es la *media ponderada por el grado de emparejamiento*, que se suele combinar con el *centro de gravedad* como valor característico del conjunto difuso (Cordón y cols. (1997); Hellendoorn y Thomas (1993)).

D.2. Sistemas Difusos Genéticos

En la literatura podemos encontrar numerosos trabajos que resuelven problemas de regresión, control, clasificación y minería de datos mediante sistemas difusos. Esto se debe, principalmente, a que los sistemas difusos poseen una habilidad para manejar la imprecisión y la incertidumbre. Además, son capaces de describir el comportamiento de sistemas complejos sin necesidad de un modelo matemático.

Los modelos difusos más comunes se basan en un conjunto de reglas difusas lógicas. Estas reglas pueden ser diseñadas de forma automática porque se puede considerar como una tarea de optimización. Para ello, se han usado AEs y AGs ya que son capaces de encontrar buenas soluciones cercanas al óptimo sin necesidad de una descripción precisa del problema. La hibridación de los sistemas difusos y los AGs se denomina Sistemas Difusos Genéticos (SDG).

El principal problema de los sistemas difusos reside en establecer un equilibrio entre la precisión y la interpretabilidad. Este problema se puede abordar fácilmente mediante la aplicación directa de los AEOMs para el diseño de los sistemas difusos. Esta hibridación recibe el nombre de Sistemas Difusos Genéticos Evolutivos Multi-

objetivo (SDGEM).

D.2.1. Taxonomía de los SDGEMs

Los SDGEMs se pueden clasificar atendiendo a distintos criterios (Herrera (2008); Herrera y Lozano (2009); Cordón (2011)), pero la clasificación realizada por Fazzolari y cols. (2013) nos ofrece una idea más general que será descrita en las siguientes subsecciones.

D.2.1.1. Diseño de SDGEMs para generar SDBRD

Este diseño hace referencia al problema principal de los sistemas difusos: el equilibrio entre la precisión y la interpretabilidad. En este caso se pueden diseñar e implementar algoritmos multiobjetivo que puedan manejar este equilibrio. Los componentes de un SBRD que pueden optimizar son:

- Los componentes del SBRD (combinados o no) con un proceso de ajuste del conjunto de reglas. En este caso, una BC predefinida se puede ajustar mediante un proceso de optimización, como por ejemplo, las funciones de pertenencia de la BD, los parámetros de la inferencia, etc. Además, se puede realizar un proceso de ajuste del conjunto de reglas para reducir la complejidad del sistema. En la literatura podemos encontrar propuestas para el ajuste de las funciones de pertenencia como las de Alcalá y cols. (2007) y Gacto y cols. (2010). El primer trabajo usa una versión mejorada del algoritmo SPEA-II y tiene en cuenta dos objetivos: la precisión, minimizando el error cuadrático medio, y la complejidad que atiende al número de reglas del sistema. En el segundo trabajo, se propone un índice para preservar la interpretabilidad de la semántica y simultáneamente se realiza un ajuste de las funciones de pertenencia. Ambos resuelven problemas de regresión. Por otro lado, Márquez y cols. (2010) proponen un método de defuzzificación adaptativo que mejora la precisión. Además, usa una adaptación del algoritmo NSGA-II en donde establece tres objetivos: el error cuadrático medio, el número final de reglas del sistema y un índice de interpretabilidad.

- Aprendizaje de la BC. En este punto cabe mencionar tanto el aprendizaje de la BR como el de la BD por separado o simultáneamente. Botta y cols. (2009) presentan un método de ajuste de los parámetros de la BD. En esta propuesta, se

aplica el algoritmo NSGA-II para maximizar la precisión y la interpretabilidad de un SBRD lingüístico.

D.2.1.2. SDGEMs para problemas de control multiobjetivo

El rendimiento de un controlador depende de su precisión. En el diseño de un controlador encontramos un par de problemas. El primero aparece si los procesos se describen de forma imprecisa o son controlados por humanos. Por otro lado, encontramos el problema de cómo diseñar un modelo que se adapte al cambio de parámetros del sistema. El modelizado difuso es una poderosa herramienta en el diseño de un sistema de control, pero contempla varios objetivos que a veces están en conflicto. Los AEOMs pueden solventar estos problemas si los codificamos en una estructura y en parámetros dentro de un cromosoma. Para identificar los parámetros del controlador podemos encontrar el trabajo de Ahlawat y Ramaswamy (2001), donde diseñan un sistema difuso con reglas Mamdani para el control de la vibración de una estructura de ingeniería civil en zonas sísmicas. Aquí consideran dos objetivos: la seguridad y el nivel de confort del usuario. Por otro lado, Serra y Bottura (2006) realizan un aprendizaje de la estructura del controlador para plantas no lineales. El controlador se basa en un sistema difuso, redes neuronales y AGs. En este trabajo, un AEOM diseña de forma óptima un controlador integral proporcional difuso. Establece tres objetivos: minimización del tiempo de encendido, minimización del tiempo de estabilización y el tiempo de respuesta.

D.2.1.3. SDGEMs para minería de reglas

La hibridación de AEOMs y sistemas difusos permite la extracción automática del conocimiento de los datos. Los métodos predictivos se aplican en clasificación, regresión y ,en algunos casos, a problemas de control. En las reglas de asociación difusas de minería, los objetivos se basan en la calidad de las reglas que se extraen. Las reglas deben ser precisas, suficientemente generales o específicas, interesantes, etc. Debido a la gran cantidad de métricas, los AEOMs han resuelto problemas con este tipo de reglas de asociación de forma satisfactoria. Thilagam y Ananthanarayana (2008) proponen una técnica para optimizar reglas de asociación difusas para minería. Esta optimización tiene como objetivo detectar intrusos en una red. En esta propuesta utilizan un algoritmo genético multiobjetivo (Fonseca y Fleming (1993)) para generar y optimizar las reglas de asociación difusas maximizando dos objetivos:

la confidencialidad y el apoyo para identificar la generalidad de la regla.

Referencias

Acosta, J., Nebot, A., Villar, P., y Fuertes, J. (2007a). Learning fuzzy partitions in fir methodology. *International Journal of General Systems*, *36*, 703–731.

Acosta, J., Nebot, A., Villar, P., y Fuertes, J. (2007b). Optimization of fuzzy partitions dor inductive reasoning using genetic algorithms. *International Journal of Systems Science*, *38*, 991–1011.

Adnan, M. M., Sarkheyli, A., Adnan, A. M., y Haron, H. (2015). Fuzzy logic for modeling machining process: a review. *Artificial Intelligence Review*, *43*(3), 345–379.

Ahlawat, A., y Ramaswamy, A. (2001). Multiobjective optimal structural vibration control using fuzzy logic control system. *Journal of Structural Engineering*, *127*(11), 1330–1337.

Aja-Fernández, S., y Alberola-López, C. (2008). Matriz modeling of hierarchical fuzzy systems. *IEEE Transactions on Fuzzy Systems*, *16*(3), 585–599.

Alcalá, R., Casillas, J., Cordón, O., Herrera, F., y Zwir, J. (2000). *Learning and tuning fuzzy rule-based systems for linguistic modeling* (Vol. 3; C. Leondes, Ed.). San Diego, EE.UU.: Academic Press.

Alcalá, R., Gacto, M., Herrera, F., y Alcalá-Fdez, J. (2007). A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *15*(5), 539–557.

Alcalá, R., y Nojima, Y. (2009). Special issue on genetic fuzzy systems: new advances. *Evolutionary Intelligence*, *2*, 1–3.

Alcalá, R., Alcalá-Fdez, J., Casillas, J., Cordón, O., y Herrera, F. (2006). Hybrid learning models to get the interpretability-accuracy trade-off in fuzzy modeling. *Soft Computing*, *10*, 717–734.

Alcalá, R., Alcalá-Fdez, J., Gacto, M., y Herrera, F. (2007). Rule base reduction and genetic tuning of fuzzy systems based on the linguistic 3-tuples representation. *Soft Computing*, *11*, 401–419.

Alcalá, R., Alcalá-Fdez, J., Herrera, F., y Otero, J. (2007). Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. *International Journal of Approximate Reasoning*, *44*(1), 46–64.

Alcalá, R., Gacto, M., y Herrera, F. (2011). A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. *IEEE Transactions on Fuzzy Systems*, *19*, 666–681.

Alcalá, R., Nojima, Y., Herrera, F., y Ishibuchi, H. (2011). Multiobjective genetic fuzzy rule selection of single granularity-based fuzzy classification rules and its interaction with the lateral tuning of membership functions. *Soft Computing*, *15*(12), 2303–2318.

Alcalá-Fdez, J., Herrera, F., Márquez, F., y Peregrín, A. (2007). Increasing fuzzy rules cooperation based on evolutionary adaptive inference systems. *International Journal on Intelligent Systems*, *22*, 1035–1064.

Alonso, J., y Magdalena, L. (2011). Special issue on interpretable fuzzy systems. *Information Sciences*, *181*, 4331–4339.

Alsina, C., y Trillas, E. (1992). Synthesizing implications. *International Journal of Intelligent Systems*, *7*, 705–713.

An, Z., y Zhang, Y. (2007). The application study of machine learning. *Journal of Changzhi University*, *24*(2), 21–24.

Antonelli, M., Ducange, P., Lazzerini, B., y Marcelloni, F. (2011). Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity, and partition integrity. *Soft Computing*, *15*(12), 2335–2354.

Arroyo, J., y Armentano, V. (2005). Genetic local search for multi-objective flow shop scheduling problems. *European Journal of Operational Research*, *167*, 717–738.

Ata, R. (2015). Artificial neural networks applications in wind energy systems: a review. *Renewable and Sustainable Energy Reviews*, *49*, 534–562.

Babuška, R. (1998). *Fuzzy modeling for control*. Norwell, Massachusetts, EE. UU.: Kluwer Academic.

Bárdossy, A., y Duckstein, L. (1995). *Fuzzy rule-based modeling with application to geophysical, biological and engineering systems*. Boca Raton, Florida, EE. UU.: CRC Press.

Bäck, T., Foussette, C., y Krause, P. (2013). *Taxonomy of evolution strategies*. Springer Berlin Heidelberg.

Benítez, A., y Casillas, J. (2009). Genetic learning of serial hierarchical fuzzy systems for large-scale problems. En *Proceedings of joint 2009 international fuzzy systems association world congress and 2009 european society of fuzzy logic and technology conference (ifsa-eusflat 2009)* (pp. 1751–1756). Lisboa, Portugal.

Beyer, H., y Schwefel, H. (2002). Evolution strategies. a comprehensive introduction. *Natural Computing*, *1*(1), 3–52.

Botta, A., Lazzerini, B., Marcelloni, F., y Stefanescu, D. (2009). Context adaptation of fuzzy systems through a multi-objective evolutionary approach based on a novel interpretability index. *Soft Computing*, *13*(5), 437–449.

Bouchaala, L., Masmoudi, A., Gargouri, F., y Rebai, A. (2010). Improving algorithms for structure learning in bayesian networks using a new implicit score. *Expert Systems with Applications*, *37*, 5470–5475.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L., Friedman, J., Olshen, R., y Stone, C. (1984). *Classification and regression trees* (Chapman y Hall, Eds.). Monterey, CA, EE.UU.: Wadsworth International Group.

Burke, E., y Kendall, G. (Eds.). (2005). *Search methodologies*. Spring Street, New York, U.S.A: Springer US.

Cala, S., y Moreno-Velo, F. (2010). Xfhl: A tool for the induction of hierarchical fuzzy systems. En *Ieee international conference on fuzzy systems* (pp. 1–6). Barcelona, España: IEEE.

Cao, Z., y Kandel, A. (1989). Applicability of some fuzzy implication operators. *Fuzzy Sets and Systems*, *31*, 151–186.

Casillas, J., y Carse, B. (2009). Genetic fuzzy systems: Recent developments and future directions. *Soft Computing*, *13*, 417–418. (Special Issue)

Casillas, J., Cordón, O., del Jesús, M., y Herrera, F. (2001). Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. *Information Sciences*, *136*, 135–157.

Casillas, J., Cordón, O., Herrera, F., y Magdalena, L. (Eds.). (2003a). *Accuracy improvements to find the balance interpretability-accuracy in linguistic fuzzy modeling: An overview*. Springer Berlin Heidelberg.

Casillas, J., Cordón, O., Herrera, F., y Magdalena, L. (Eds.). (2003b). *Interpretability improvements to find the balance interpretability-accuracy in fuzzy modeling: An overview*. Springer Berlin Heidelberg.

Casillas, J., Cordón, O., del Jesús, M., y Herrera, F. (2005). Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems*, *13*(1), 13–29.

Casillas, J., Cordón, O., y Herrera, F. (2002). Cor: a methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, *32*(4), 526–537.

Casillas, J., Cordón, O., Herrera, F., y Magdalena, L. (Eds.). (2003). *Cor methodology: a simple way to obtain linguistic fuzzy models with good interpretability and accuracy*. Heidelberg, Alemania: Accuracy Improvements in Linguistic Fuzzy Modeling, Studies in Fuzziness and Soft Computing, Springer.

Casillas, J., Herrera, F., Pérez, R., del Jesús, M., y Villar, P. (2007). Special issue on genetic fuzzy systems and the interpretability-accuracy trade-off. *International Journal on Approximate Reasonint*, *44*, 1–3.

Casillas, J., Martínez, P., y Benítez, A. (2009). Learning consistent, complete and compact sets of fuzzy rules in conjunctive normal form for regression problems. *Soft Computing*, *13*, 451–465.

Castro, J., Castro-Sánchez, J., y Zurita, J. (1999). Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems. *Fuzzy Sets and Systems*, *101*, 331–342.

Chambers, L. (Ed.). (1998). (Vol. 3). EE.UU.: CRC Press Inc.

Chen, L., y Wang, D. (2007). Multivariate decision trees based on regression and discriminant analysis. En *International conference on convergence information technology* (pp. 1733–1741). Gyeongju, Corea del Sur: IEEE.

Chen, Y., y Abraham, A. (2010). *Hierarchical fuzzy systems* (J. Kacprzyk y L. C. Jain, Eds.). Intelligent Systems Reference Library, Springer Berlin Heidelberg.

Chen, Y., Dong, J., y Yang, B. (2004). Automatic design of hierarchical ts-fs model using ant programming and pso algorithm. En C. Bussler y D. Fensel (Eds.), *Proceedings 12th international conference on artificial intelligence, methodology, systems and applications, lecture notes on artificial inteligence, lnai 3192* (pp. 285–294).

Chen, Y., Yang, B., Abraham, A., y Peng, L. (2007). Automatic design of hierarchical takagi-sugeno type fuzzy systems using evolutionary algorithms. *IEEE Transactions on Fuzzy Systems*, *15*(3), 385–397.

Cheong, F. (2007). A hierarchical fuzzy system with high input dimensions for forecasting foreign exchange rates. *IEEE Congress on Evolutionary Computation, CEC*, 1642–1647.

Chiu, S. (1996). Selecting input variables for fuzzy models. *Journal of Intelligent and Fuzzy Systems*, *4*(4), 243–256.

Cid-Fernandes, R., Mateus, A., Jr, L. S., Stasińska, G., y Gomes, J. (2005). Semi-empirical analysis of sloan digital sky survey galaxies - i. spectral synthesis method. *Monthly Notices of the Royal Astronomical Society*, *358*, 363–378. doi: 10.1111/j.1365-2966.2005.08752.x

Cid-Fernandes, R., Stasińska, G., Schlickmann, M., Mateus, A., Vale-Asari, N., Schoenell, W., y Jr, L. S. (2010). Alternative diagnostic diagrams and forgotten population of weak line galaxies in the sdss. *Monthly Notices of the Royal Astronomical Society*, *403*, 1036–1053. doi: 10.1111/j.1365-2966.2009.16185.x

Coello, C., Lamont, G., y Veldhuizen, D. V. (Eds.). (2007). Springer US.

Collette, Y., y Siarry, P. (Eds.). (2004). Springer-Verlag Berlin Heidelberg.

Contreras, J., Misa, R., y Ureta, L. (2007). Algoritmos para identificación de modelos difusos interpretables. *IEEE Latin America Transactions*, *8*, 346–351.

Cordón, O. (2011). A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *International Journal of Approximate Reasoning*, *52*, 894–913.

Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., y Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, *141*, 5–31. doi: 10.1016/S0165-0114(03)00111-8

Cordón, O., Herrera, F., Magdalena, L., y Villar, P. (2001). A genetic learning process for the scaling factors, granularity and contexts of the fuzzy rule-based system data base. *Information Sciences*, *136*, 85–107.

Cordón, O., Herrera, F., y Peregrín, A. (1997). Applicability of the fuzzy operators in the design of fuzzy logic controllers. *Fuzzy Sets and Systems*, *86*(1), 15–41.

Cordón, O., Herrera, F., y Sánchez, L. (1999). Solving electrical distribution problems using hybrid evolutionary data analysis techniques. *Applied Intelligence*, *10*(1), 5–24.

Cordón, O., Herrera, F., y Villar, P. (2001). Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. *IEEE Transactions on Fuzzy Systems*, *9*(4), 667–674.

Davis, L. (Ed.). (1991). Nueva York, EE.UU.: Van Nostrand Reinhold.

Deb, K. (Ed.). (2001). *Multi-objective optimization using evolutionary algorithms*. Boston, Massachusetts, EE.UU.: Wiley Custom.

Deb, K., Pratap, A., Agarwal, S., y Meyarevian, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Driankov, D., Hellendoorn, H., y Reinfrank, M. (1993). *An introduction to fuzzy control*. Heidelberg, Alemania: Springer-Verlag.

Duan, J., y Chung, F. (2002). Multilevel fuzzy relational systems: structure and identification. *Soft Computing*, *6*(2), 71–86.

Essabri, A., Gzara, M., y Loukil, T. (2006). Parallel multi-objective evolutionary algorithm with multi-front equitable distribution. En *Fifth international conference on grid and cooperative computing* (pp. 241–244). Hunan, China: IEEE.

Fakhari, A., y Moghadam, A. (2013). Combination of classification and regression in decision tree for multi-labeling image annotation and retrieval. *Applied Soft Computing*, *13*, 1292–1302.

Fazzolari, M., Alcalá, R., Nojima, Y., Ishibuchi, H., y Herrera, F. (2013). A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE Transactions on Fuzzy Systems*, *21*(1), 45–65.

Fernández, A., López, V., Jesús, M. D., y Herrera, F. (2015). Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges. *Knowledge-Based Systems*, *80*, 109–121.

Fogel, D., y Fogel, L. (1996). *An introduction to evolutionary programming* (J. Alliot, E. Lutton, E. Ronald, M. Schoenauer, y D. Snyers, Eds.). Springer Berlin Heidelberg.

Fogel, G. (2012). *Evolutionary programming* (G. Rozenberg, T. Bäck, y J. Kok, Eds.). Springer Berlin Heidelberg.

Fonseca, C., y Fleming, P. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. En (pp. 416–423).

Gacto, M., Alcalá, R., y Herrera, F. (2010). Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems. *IEEE Transactions on Fuzzy Systems*, *18*(3), 515–531.

Gacto, M., Alcalá, R., y Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, *181*, 4340–4360.

Gacto, M., Alcalá, R., y Herrera, F. (2009). Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems. *Soft Computing*, *13*, 419–436.

Garg, A., y Tai, K. (2012). Review of genetic programming in modeling of machining processes. En *Proceedings of international conference on modelling, identification and control (icmic)* (pp. 653–658). IEEE.

Gaweda, A., y Scherer, R. (2004). Fuzzy number-based hierarchical fuzzy system. *Lecture Notes in Computer Science*, *3070*, 302–307.

Giordana, A., y Neri, F. (1995). Search-intensive concept induction. *Evolutionary Computation*, *3*, 375–416.

Goldberg, D. (Ed.). (1989). *Genetic algorithms in search, optimization and machine learning*. Boston, Massachusetts, EE.UU.: Addison-Wesley Longman Publishing Co., Inc.

González, A., y Pérez, R. (2001). Selection of relevant features in a fuzzy genetic learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, *31*(3), 417–425.

Greene, D., y Smith, S. (1993). Competition-based induction of decision models from examples. *Machine Learning*, *3*, 229–257.

Gunn, J., Siegmund, W., Mannery, E., y otros. (2006). The 2.5 m telescope of the sloan digital sky survey. *The Astronomical Journal*, *131*, 2332–2359.

Gupta, M. M., y Qi, J. (1991). Design of fuzzy logic controllers based on generalized t-operators. *Fuzzy Sets and Systems*, *40*, 473–489.

Hagras, H., Callaghan, V., Colley, M., y Carr-West, M. (1999). A behaviour based hierarchical fuzzy control architecture for agricultural autonomous mobile robots. En *Proceedings of the international conference on computational intelligence for modelling, control and automation* (pp. 166–171). Viena, Austria.

Heckerman, D. (1998). *A tutorial on learning with bayesian networks* (Vol. 89; M. Jordan, Ed.). Boston, EE.UU.: Kluwer Academic Publishers.

Heckerman, D., Geiger, D., y Chickering, D. (1995). Learning bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, *20*, 197–243.

Hellendoorn, H., y Thomas, C. (1993). Deffuzification in fuzzy controllers. *Journal of Intelligent Fuzzy Systems*, *1*, 109–123.

Hermoso-Gutiérrez, J., y Hernández-Bastida, A. (Eds.). (1997). *Curso básico de estadística descriptiva y probabilidad*. Granada, España: Némesis.

Herrera, F. (2008). Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evolutionary Intelligence*, *1*, 27–46.

Herrera, F., y Lozano, M. (2009). Fuzzy evolutionary algorithms and genetic fuzzy systems: A positive collaboration between evolutionary algorithms and fuzzy systems. *Computational Intelligence*, *1*, 83–130.

Herrera, F., y Magdalena, L. (1997). Genetic fuzzy systems: A tutorial. *Tatra Mountains Mathematical Publications (Slovakia)*, *13*, 93–121.

Hüllermeier, E. (2008). Fuzzy methods for data mining and machine learning: State of the art and prospects. *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, 357–375.

Ho, T. K., y Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), 289–300.

Ho, T. K., Basu, M., y Law, M. (2006). Measures of geometrical complexity in classification problems. En *Data complexity in pattern recognition* (pp. 1–23). Springer.

Hoimafar, A., y Mccormic, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, *3*, 129–139.

Holland, H. (Ed.). (1975). *Adaptation in natural and artificial systems*. Ann Arbor, Michigan, EE.UU.: University of Michigan Press.

Holmes, G., Hall, M., y Frank, E. (1999). Generating rule sets from model trees. En *Twelfth australian joint conference on artificial intelligence* (p. 1-12). Springer.

Holve, R. (1998). Investigation of automatic rule generation for hierarchical fuzzy systems. *Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence*, *2*, 973–978.

Hong, T., y Chen, J. (1999). Finding relevant attributes and membership functions. *Fuzzy Sets and Systems*, *103*(3), 389–404.

Hong, T., y Lee, C. (1999). Effect of merging order on performance of fuzzy induction. *Intelligent Data Analysis*, *103*, 389–404.

Hong, X., y Harris, C. (2001). Variable selection algorithm for the construction of mimo operating point dependent neurofuzzy networks. *IEEE Transactions on Fuzzy Systems*, *9*(1), 88–101.

Horn, J., Nafpliotis, N., y Goldberg, D. (1994). A niched pareto genetic algorithm for multiobjective optimization. En (pp. 82–87). IEEE.

Huang, Z., Li, J., Su, H., Watts, G., y Chen, H. (2007). Large-scale regulatory network analysis from microarray data: modified bayesian network learning and association rule mining. *Decision Support Systems*, *43*, 1207–1225.

Ikonomovska, E., Gama, J., y Dzeroski, S. (2015). Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing*, *150*, 458–470.

Ishibuchi, H., K. Nozaki, Yamamoto, N., y Tanaka, H. (1995). Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, *3*(3), 260–270.

Ishibuchi, H., Masuda, H., Tanigaki, Y., y Nojima, Y. (2014). Review of coevolutionary developments of evolutionary multi-objective and many-objective algorithms and test problems. En *Ieee symposium on computational intelligence in multi-criteria decision-making* (pp. 178–184). Orlando, Florida, EE.UU.: IEEE.

Ishibuchi, H., Yoshida, T., y Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multi-objective permutation flow shop. *IEEE Transactions on Evolutionary Computation*, *7*, 204–223.

Jaimes, A., y Coello, C. C. (2005). Mrmoga: parallel evolutionary multi-objective optimization using multiple resolutions. En *Ieee congress on evolutionary computation* (pp. 2294–2301). Kanpur, India: IEEE.

Jarraya, A., Leray, P., y Masmoudi, A. (2014). Discrete exponential bayesian networks: Definition, learning and application for density estimation. *Neurocomputing*, *137*, 142–149.

Jayaram, B. (2008). Rule reduction for efficient inferencing in similarity based reasoning. *Approximate Reasoning*, *48*, 156–173.

Jelleli, T., y Alimi, A. (2005). Improved hierarchical fuzzy control scheme. *Adaptive and Natural Computing*, *1*, 128–131.

Jelleli, T., y Alimi, A. (2010). Automatic design of a least complicated hierarchical fuzzy system. *6th IEEE World Congress on Computational Intelligence*, 1–7.

Jin, Y. (2000). Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement. *IEEE Transactions on Fuzzy Systems*, *8*(2), 212–221.

Jing, C., Jing-qi, F., y Wei, S. (2011). Learning bayesian network parameters based on iterative learning control. En *International conference on consumer electronics, communications and networks* (pp. 4161–4165). XianNing, China: IEEE.

Jong, K. D., Spears, W., y Gordon, D. (1993). Using genetic algorithms for concept learning. *Machine Learning*, *13*, 161–188.

Joo, M., y Lee, J. (1999). Hierarchical fuzzy control scheme using structured takagi-sugeno type fuzzy inference. En *Proceedings of ieee international fuzzy systems conference* (pp. 78–83). Seúl, Corea.

Joo, M., y Lee, J. (2002). Universal approximation by hierarchical fuzzy system with constrains on the fuzzy rule. *Fuzzy Sets and Systems*, *130*(2), 175–188.

Joo, M., y Sudkamp, T. (2009). A method of converting a fuzzy system to a two-layered hierarchical fuzzy system and its run-time efficiency. *IEEE Transactions on Fuzzy Systems*, *17*(1), 93–103.

Kim, D., Choi, Y., y Lee, S. (2002). An accurate cog defuzzier design using lamarckian co-adaptation of learning and evolution. *Fuzzy Sets Systems*, *130*, 207–225.

Kiszka, J., Kochanska, M., y Sliwinska, D. (1985). The influence of some fuzzy implication operators on the accuracy of a fuzzy model. Partes I y II *Fuzzy Sets and Systems*, *15,15*, 111–128,223–240.

Kleeman, M., y Lamont, G. (2006). Co-evolutionary multi-objective eas: the next frontier? En *Ieee congress on evolutionary computation* (pp. 1726–1735). Vancouver, Canadá: IEEE.

Kotsiantis, S. (2013). Decision trees: a recent overview. *Artificial Intelligence Review*, *39*, 261–283.

Kovacs, T. (2004). Springer-Verlag London.

Lee, H., Chen, C., Chen, J., y Jou, Y. (2001). An efficient fuzzy classifier with feature selection based on fuzzy entropy. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, *31*(3), 426–432.

Lee, M., Chung, H., y Yu, F. (2003). Modeling of hierarchical fuzzy systems. *Fuzzy Sets and Systems*, *138*(2), 343–361.

Mackiewicz, A., y Ratajczak, W. (1993). Principal components analysis. *Computers and Geosciences*, *13*(3), 303–342.

Maeda, H. (1996). An investigation on the spread of fuzziness in multi-fold multistage approximate reasoning by pictorial representation under sup-min composition and triangular type membership function. *Fuzzy Sets and Systems*, *80*(2), 133–148.

Mamdani, E. H. (1974). Applications of fuzzy algorithms for control a simple dynamic plant. *Proceedings of the IEE 121*, *12*, 1585–1588.

Masmoudi, N. K., Rekik, C., Djemel, M., y Derbel, N. (2010). Optimal control for discrete large scale nonlinear systems using hierarchical fuzzy systems. En *Proceedings 2nd international conference on machine learning and computing* (pp. 91–95). Bangalore, India: IEEE.

Mencar, C. (2013). Interpretability of fuzzy systems. En *Proceedings 10th international workshop on fuzzy logic and applications* (pp. 22–35). Genova, Italia: Springer International Publishing.

Mikut, R., Jäkel, J., y Gröll, L. (2005). Interpretability issues in data-based learning of fuzzy systems. *Fuzzy Sets and Systems*, *150*, 179–197.

Montazer, G., y Giveki, D. (2015). An improved radial basis function neural network for object image retrieval. *Neurocomputing*, *168*, 221–223.

Márquez, A., Márquez, F., y Peregrín, A. (2010). A multi-objective evolutionary algorithm with an interpretability improvement mechanism for linguistic fuzzy systems with adaptative defuzzification. En *Proceedings of ieee international conference on fuzzy systems* (pp. 1–7).

Murata, T., y Ishibuchi, H. (1995). Moga: multi-objective genetic algorithms. En (Vol. 1). Perth, WA, Australia: IEEE.

Nauk, D., y Kruse, R. (1999). Neuro-fuzzy systems for function approximation. *Fuzzy Sets and Systems*, *101*, 261–271.

Nojima, Y., Alcalá, R., Ishibuchi, H., y Herrera, F. (2011). Special issue on evolutionary fuzzy systems. *Soft Computing*, *15*(12), 2299–2301.

Patel, R., y Raghuwanshi, M. (2010). Review on real coded genetic algorithms used in multiobjective optimization. En *3rd international conference on emerging trends in engineering and technology* (pp. 610–613). Goa, India: IEEE.

Pedrycz, W. (1994). Why triangular membership functions? *Fuzzy Sets and Systems*, *64*, 21–30.

Pedrycz, W. (2001). Fuzzy equalization in the construction of fuzzy sets. *Fuzzy Sets and Systems*, *119*, 329–335.

Peng, X., y Xu, D. (2013). A local information-based feature-selection algorithm for data regression. *Pattern Recognition*, *46*, 2519–2530.

Quinlan, J. (1986). Introduction to decision tree. *Machine Learning*, *1*, 81–106.

Quinlan, R. (1992). Learning with continuous classes. En *5th australian joint conference on artificial intelligence* (pp. 343–348). Singapur, Malasia: World Scientific.

Raju, G., Zhou, J., y Kisner, R. (1991). Hierarchical fuzzy control. *International Journal Control*, *54*(5), 1201–1216.

Rattasiri, W., y Halgamuge, S. (2000). Computational complexity of hierarchical fuzzy systems. En *Proceedings 19th international conference of the north american.* (pp. 383–387). Atlanta, EE.UU.: IEEE.

Rojas, I., Pomares, H., Ortega, J., y Prieto, A. (2000). Self-organized fuzzy systems generation from training examples. *IEEE Transactions on Fuzzy Systems*, *8*, 23–36.

Rokach, L. (2016). Decision forest: Twenty years of research. *Information Fusion*, *27*, 111–123.

Rosenblatt, F. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanism* (G. Palm y A. Aertsen, Eds.). Chicago, EE.UU.: Spartan Books.

Salgado, P. (2005). Clustering and hierarchization of fuzzy systems. *Soft Computing*, *9*(10), 715–731.

Salgado, P. (2008). Rule generation for hierarchical collaborative fuzzy system. *Applied Mathematical Modelling, Science Direct*, *32*(7), 1159–1178.

Salimans, T. (2012). Variable selection and functional form uncertainty in cross-country. *Journal of Econometrics*, *171*, 267–280.

Schoenell, W., Cid-Fernandes, R., Benítez, N., y Vale-Asari, N. (2013, Mayo). Recovering physical properties from narrow-band photometry. En J. C. Guirado, L. M. Lara, V. Quilis, y J. Gorgas (Eds.), *Highlights of spanish astrophysics vii* (pp. 405–410). Valencia, España.

Serra, G., y Bottura, C. (2006). Multiobjective evolution based fuzzy pi controller design for nonlinear systems. *Engineering Applications of Artificial Intelligence*, *19*(2), 157–167.

Shapiro, A. (2005). Fuzzy regression models. *Article of Penn State University*.

Shi, G. (2014). *Decision trees* (G. Shi, Ed.). San Diego, California, EE.UU.: Elsevier.

Shimojima, K., Fukuda, T., y Hasegawa, Y. (1995). Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm. *Fuzzy Sets and Systems*, *71*(3), 295–309.

Shukla, P., y Tripathi, S. (2012). A review on the interpretability-accuracy trade-off in evolutionary multi-objective fuzzy systems (emofs). *Information*, *3*, 256–277.

Sivanandam, S., y Deepa, S. (2008). Springer Berlin Heidelberg.

Smith, S. (1980). A learning system based on genetic algorithms. *PhD Thesis, University of Pittsburg, Pittsburg*.

Srinivas, N., y Deb, K. (1995). Multiobjective function using nondominated sorting genetic algorithms. *Evolutionary Computation*, *2*(3), 221–248.

Stulp, F., y Sigaud, O. (2015). Many regression algorithms, one unified model: A review. *Neural Networks*, *69*, 60–79.

Sun, S. (2013). A review of deterministic approximate inference techniques for bayesian machine learning. *Neural Computing and Applications*, *23*, 2039–2050.

Takagi, T., y Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, *15*, 116–132.

Tan, F., Fu, X., Zhang, Y., y Bourgeois, A. (2008). A genetic algorithm-based method for feature subset selection. *Soft Computing*, *12*, 111–120.

Taniguchi, T., Tanaka, K., Ohtake, H., y Wang, H. (2001). Model construction, rule reduction, and robust compensation for generalized form of takagi-sugeno fuzzy systems. *IEEE Transactions on Fuzzy Systems*, *9*(4), 525–538.

Thilagam, P., y Ananthanarayana, V. (2008). Extraction and optimization of fuzzy association rules using multi-objective genetic algorithm. *Pattern Analysis and Applications*, *11*(2), 159–168.

Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. En *4th international conference on genetic algorithms* (pp. 509–513). San Diego, California, EE.UU.: Morgan Kaufmann.

Torra, V. (2002). A review of the construction of hierarchical fuzzy systems. *International Journal of Intelligent Systems*, *17*(5), 531–543.

Trillas, E., Valverde, L., Gupta, M. M., Kandel, A., Bandler, W., y Kiszka, J. B. (1985). One mode and implication in aproximate reasoning. En *Approximate reasoning in expert systems.* Amsterdam, Holanda: North-Holland.

Trillas, E., Valverde, L., Kracpryzk, J., y Yager, R. (1985). On implications and indistinguishability in the setting of fuzzy logic. En *Managements decision support systems using fuzzy logic and possibility theory* (pp. 198–212). Colonia, Alemania: Verlag TUV Rheinland.

Uysal, I., y Altay-Güvenir, H. (1999). An overview of regression techniques for knowledge discovery. *The Knowledge Engineering Review*, *14*(4), 319–340.

Vehtari, A., y Lampinen, J. (1999). Bayesian neural networks for industrial applications. En J. Martikainen (Ed.), *Proceedings of the 1999 ieee midnight-sun workshop on soft computing methods in industrial applications* (pp. 63–68). Kuusamo, Finlandia: IEEE.

Venturini, G. (1993). Sia: a supervised inductive algorithm with genetic search for learning attribute based concepts. En *European conference on machine learning* (pp. 280–296). Viena, Austria: Morgan Kaufmann.

Wang, D., Zeng, X., y Keane, J. (2006). Learning for hierarchical fuzzy systems based on gradient-descent method. En *Proceedings of ieee international conference on fuzzy systems* (pp. 92–99).

Wang, H., Zhou, L., y Ma, C. (2009). A brief review of machine learning and its application. En *Proceedings of ieee international conference on information engineering and computer science* (pp. 1–4).

Wang, L. (1998). Universal approximation by hierarchical fuzzy systems. *Fuzzy Sets and Systems*, *93*(2), 223–230.

Wang, L. (1999). Analysis and design of hierarchical fuzzy systems. *IEEE Transactions on Fuzzy Systems*, *7*(5), 617–624.

Wang, L., y Mendel, J. (1992). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(6), 1414–1427.

Wang, L. X. (1994). *Adaptive fuzzy systems and control, design and stability analysis*. Engleweed Cliffs, NJ, EE. UU.: Prentice Hall.

Wang, R., Fleming, P., y Purshouse, R. (2014). General framework for localised multi-objective evolutionary algorithms. *Information Sciences*, *258*, 29–53.

Wang, S., F.L.Chung, Wang, J., y J.Wu. (2015). A fast learning method for feedforward neural networks. *Neurocomputing*, *149*, 295–307.

Weigend, A., y Gershenfeld, N. (Eds.). (1993). *Time series prediction: forecasting the future and understanding the past*. Santa Fe, Nuevo Méjico: 1992 NATO Advanced Research Workshop on Comparative Time Series Analysis, Addison-Wesley.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, *4*, 65–85.

Wright, S. (1921). Correlation and causation. *Journal of Agricultural Research*, *20*, 557–585.

Xiong, N., y Funk, P. (2006). Construction of fuzzy knowledge bases incorporating feature selection. *Soft Computing*, *10*, 796–804.

Xiong, N., y Litz, L. (2000). Fuzzy modeling based on premise optimization. En *Proceedings of the 9th ieee transactional conference on fuzzy systems* (pp. 859–964). San Antonio, TX, EE.UU.: Springer Berlin Heidelberg.

Yang, S., Guo, S., y Jie, L. (2005). A novel evolution strategy for multi-objective optimization problem. *Applied Mathematics and Computation*, *170*, 850–873.

Yao, X., y Liu, Y. (2014). *Machine learning* (E. K. Burke y G. Kendall, Eds.). Springer US.

Yen, V. (1999). Rule selections in fuzzy expert systems. *Expert Systems with Applications*, *16*, 79–84.

York, D., Adelman, J., Jr., J. A., Anderson, S., Annis, J., y otros. (2000). The sloan digital sky survey: Technical summary. *The Astronomical Journal*, *120*, 1579–1587.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*, 338–353.

Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, *1*, 28–44.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning. Partes I, II y III *Information Science*, *8, 8, 9*, 199–249, 301–357, 43–80.

Zajaczkowski, J., y Verma, B. (2012). Selection and impact of different topologies in multi-layered hierarchical fuzzy systems. *Applied Intelligence*, *36*(3), 564–584.

Zeng, X., Goulermas, J., Liatsis, P., Wang, D., y Keane, J. (2008). Hierarchical fuzzy systems for function approximation on discrete input spaces with application. *IEEE Transactions on Fuzzy Systems*, *16*(5), 1197–1215.

Zhang, X., Onieva, E., Perallos, A., Osaba, E., y Lee, V. (2014). Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term trafic congestion prediction. *Transportation Research. Part C.*, *43*(1), 127–142. doi: http://dx.doi.org/10.1016/j.trc.2014.02.013

Zhang, X., y Zhang, N. (2006). Universal approximation of binary-tree hierarchical fuzzy system with typical flus. *Lecture Notes in Computer Science*, *4114*, 177–182.

Zhang, Y., Li, Y., Sun, J., y Ji, J. (2015). Estimates on compressed neural networks regression. *Neural Networks*, *63*, 10–17.

Zheng, Y., y Lei, D. (2008). New progresses and prospect of multi-objective evolutionary algorithm. En *International conference on machine learning and cybernetics* (pp. 962–968). Kunming, China: IEEE.

Zitzler, E., Laumanns, M., y Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. En K. Giannakoglou (Ed.), (pp. 12–21). Atenas, Grecia.

Zitzler, E., y Thiele, L. (1998). *An evolutionary algorithm for multiobjective optimization: The strength pareto approach* (Inf. Téc. n.º 43). Swiss Federal Institute of Technology, TIK-Report.