UNIVERSIDAD DE CÓRDOBA

Escuela Politécnica Superior

Departamento de Informática y Análisis Numérico

Minería de Datos mediante Programación Automática con Colonias de Hormigas

Memoria de Tesis presentada por

Juan Luis Olmo Ortiz

como requisito para optar al grado

de Doctor en Informática

Córdoba Marzo de 2013

TITULO: *MINERÍA DE DATOS MEDIANTE PROGRAMACIÓN AUTOMÁTICA CON COLONIAS DE HORMIGAS.*

AUTOR: *JUAN LUÍS OLMO ORTIZ*

© Edita: Servicio de Publicaciones de la Universidad de Córdoba.
Campus de Rabanales
Ctra. Nacional IV, Km. 396 A
14071 Córdoba

www.uco.es/publicaciones
publicaciones@uco.es

TÍTULO DE LA TESIS: Minería de Datos mediante Programación Automática con Colonias de Hormigas

DOCTORANDO/A: Juan Luis Olmo Ortiz

INFORME RAZONADO DEL/DE LOS DIRECTOR/ES DE LA TESIS

(se hará mención a la evolución y desarrollo de la tesis, así como a trabajos y publicaciones derivados de la misma).

En su tesis, D. Juan Luis Olmo Ortiz ha abordado dos tareas de minería de datos desde una metaheurística bioinspirada, la programación automática mediante colonias de hormigas (AP): clasificación y obtención de reglas de asociación.

En total, ha propuesto cinco algoritmos de AP gramatical: dos para clasificación multiclase, otro para el problema de clasificación de datos no balanceados, y otros dos modelos para la extracción de patrones frecuentes.

Todos los algoritmos desarrollados han sido publicados en revistas internacionales de impacto y conferencias internacionales, lo que muestra la calidad científica del trabajo realizado. Por otra parte, la línea de investigación desarrollada en esta memoria no está aún agotada, existiendo algunas líneas de trabajo futuro que consideramos pueden también dar lugar a varias publicaciones científicas de calidad.

En conclusión, consideramos que la memoria presentada por D. Juan Luis Olmo Ortiz reúne, en nuestra opinión, las condiciones necesarias para su defensa y obtención del título de Doctor con Mención Internacional.

Por todo ello, se autoriza la presentación de la tesis doctoral.

Córdoba, 25 de febrero de 2013

Firma del/de los director/es

Fdo.: Sebastián Ventura Soto Fdo.: José Raúl Romero Salguero

UNIVERSIDAD DE CÓRDOBA

Minería de Datos mediante Programación Automática con Colonias de Hormigas

Memoria de Tesis presentada por

Juan Luis Olmo Ortiz

como requisito para optar al grado

de Doctor en Informática

Directores

Dr. Sebastián Ventura Soto

Dr. José Raúl Romero Salguero

Córdoba Marzo de 2013

La memoria titulada "*Minería de Datos mediante Programación Automática con Colonias de Hormigas*", que presenta Juan Luis Olmo Ortiz para optar al grado de Doctor, ha sido realizada dentro del programa de doctorado "*Ingeniería y Tecnología*" del Departamento de Informática y Análisis Numérico de la Universidad de Córdoba, bajo la dirección de los doctores Sebastián Ventura Soto y José Raúl Romero Salguero cumpliendo, en su opinión, los requisitos exigidos a este tipo de trabajos.

Córdoba, Marzo de 2013

El Doctorando

Fdo: Juan Luis Olmo Ortiz

El Director

Fdo: Sebastián Ventura Soto

El Director

Fdo: José Raúl Romero Salguero

Mención de Doctorado Internacional

Esta tesis cumple los criterios establecidos por la Universidad de Córdoba para la obtención del Título de Doctor con Mención Internacional:

1. Estancia predoctoral mínima de 3 meses fuera de España en una institución de enseñanza superior o centro de investigación de prestigio, cursando estudios o realizando trabajos de investigación relacionados con la tesis doctoral:

   Department of Computer Science, School of Engineering, Virginia Commonwealth University, Richmond, Virginia, United States. Responsable de la estancia: **Ph.D. Krzysztof Cios**, Professor and Chair.

2. La tesis cuenta con el informe previo de dos doctores o doctoras expertos y con experiencia investigadora acreditada pertenecientes a alguna institución de educación superior o instituto de investigación distinto de España:

   a. **Ph.D. Ajith Abraham**, Professor, IT for Innovations - EU Center of Excellence, VSB - Technical University of Ostrava, Ostrava, Poruba, Czech Republic

   b. **Ph.D. Vojislav Kecman**, Associate Professor, Department of Computer Science, Virginia Commonwealth University, Richmond, VA, United States

3. Entre los miembros del tribunal evaluador de la tesis se encuentra un doctor procedente de una institución de educación superior distinto de España y diferente del responsable de la estancia predoctoral:

   PhD. Alessandro Provetti, Assistant Professor, Dipartimento di Matematica e Informatica, University of Messina, Messina, Italy

4. Parte de la tesis doctoral se ha redactado y presentado en dos idiomas, castellano e inglés.

Córdoba, Marzo de 2012

El Doctorando

Fdo: Juan Luis Olmo Ortiz

Tesis Doctoral parcialmente subvencionada por la Comisión Interministerial de Ciencia y Tecnología (CICYT) con los proyectos **TIN2011-22408** y por la Junta de Andalucía con fondos FEDER con el proyecto de excelencia **P08-TIC-3720**.

Asímismo ha sido subvencionada por el programa predoctoral de Formación de Personal Docente e Investigador (FPDI, convocatoria publicada en el B.O.J.A. N^0 50 de 12 de marzo de 2008) de la Junta de Andalucía.

Si has construido un castillo en el aire,

no has perdido el tiempo, es allí donde debería estar.

Ahora debes construir los cimientos debajo de él.

George Bernard Shaw

Agradecimientos

He pensado largo y tendido acerca de cómo escribir la sección de agradecimientos y, a pesar de que trataré de no pasar por alto a nadie en ella, considero que no basta con escribir unas cuantas palabras, sino que la gratitud es un sentimiento que se demuestra y transmite con hechos a lo largo del tiempo.

En primer lugar quiero agradecer el apoyo de mis directores de tesis. En primer lugar, al Dr. Sebastián Ventura, por la confianza que depositó en mí cuando, hace ya cuatro años, me ofreció la posibilidad de trabajar con él y continuar mis estudios con la realización de la presente tesis doctoral. Gracias por tu dedicación y consejos durante estos años. En segundo lugar, al Dr. José Raúl Romero, por su ayuda y apoyo en este período.

Probablemente las personas con las que más tiempo he compartido estos últimos años, tanto de trabajo como de risas y llantos, y de quienes he recibido un apoyo más directo al estar junto a ellos prácticamente a diario, sean los *locos* del laboratorio. Por orden de llegada, Juan Ignacio, a quien admiro muchísimo y al que considero un ejemplo de padre y de amigo. En segundo lugar, Chemita (para mí vas a ser siempre Chemita, no Jose), que me ha ganado día a día por su integridad y veracidad, y con quien he compartido momentos inolvidables. El tercero, Alberto, un genio de la informática y mi compañero de viajes por los Estados Unidos. Finalmente, Aurora, la incorporación más reciente, y la más jovencita del grupo. De todos vosotros guardo momentos muy especiales, tanto a nivel individual como colectivo. No voy a olvidar que habéis estado apoyándome siempre, especialmente en los malos momentos. Gracias por traer positivismo a mi vida.

Quisiera agradecer el apoyo de todos los compañeros del grupo de investigación *KDIS*. Quiero hacer mención especial para Cristóbal Romero, por sus consejos y por conseguir hacer que me ría a carcajadas hasta en momentos de tensión o tristeza, y para Amelia Zafra, quien fue mi compañera en mi primera toma de contacto con la docencia en la universidad y que siempre ha tenido tiempo para explicarme dudas. Asimismo, gracias a los miembros del grupo de investigación *AYRNA*, con quienes compartí laboratorio cuando comenzó mi aventura universitaria.

Al Dr. Krzysztof Cios, de la Virginia Commonwealth University, por darme la oportunidad de trabajar en su laboratorio y facilitarme todos los trámites para la obtención del Doctorado con mención internacional. A Beata, Dat y Koray, por recibirme con los brazos abiertos en dicho laboratorio y hacerme más confortable mi estancia en los Estados Unidos. Y a Emine, una de las personas más maravillosas que he conocido nunca. Gracias por tu comprensión y apoyo incondicionales.

Por ayudarme a evadirme a del trabajo y poder recobrar energías para volver con más ganas (el período de barbecho es necesario en el proceso creativo), he de mencionar a mis amigos, Jesús, Juan Ángel, Rafa, David (y su esposa María), José María, Paco, Laura, Pedro, Inés, Jessica, Jose, Óscar y María Milman.

Mención especial para Menchu, gracias por tu valioso tiempo ayudándome a revisar los artículos en inglés, y para Curro, por permitírmelo. También para Juan Luque, mi profesor de la infancia, por su inestimable ayuda revisando este documento.

Dedico este trabajo a toda mi familia, en especial a mis padres, Juan Luis e Inmaculada, que han sido y son mis mejores maestros. Gracias por cómo sois y por apoyarme (aunque a veces no hayáis estado del todo de acuerdo) en todas y cada una de las decisiones que he tomado. También me quiero acordar de mi abuelo Paco y mi tía Lola, que siempre estaban orgullosos de mí y que seguro que me han echado un cable desde arriba.

Y por último quiero agradecer a Lour el hecho de haber irrumpido en mi vida, por infundirme fuerzas y motivarme para seguir adelante con multitud de proyectos. Haces que el tiempo que comparto contigo sea un tesoro.

Gracias de todo corazón.

Resumen

La presente tesis doctoral supone el primer acercamiento de la metaheurística de programación automática mediante colonias de hormigas (*Ant Programming*) a tareas de minería de datos. Esta técnica de aprendizaje automático ha demostrado ser capaz de obtener buenos resultados en problemas de optimización, pero su aplicación a la minería de datos no había sido explorada hasta el momento.

Específicamente, esta tesis cubre las tareas de clasificación y asociación. Para la primera se presentan tres modelos que inducen un clasificador basado en reglas. Dos de ellos abordan el problema de clasificación desde el punto de vista de evaluación monobjetivo y multiobjetivo, respectivamente, mientras que el tercero afronta el problema específico de clasificación en conjuntos de datos no balanceados desde una perspectiva multiobjetivo.

Por su parte, para la tarea de extracción de reglas de asociación se han desarrollado dos algoritmos que llevan a cabo la extracción de patrones frecuentes. El primero de ellos propone una evaluación de los individuos novedosa, mientras que el segundo lo hace desde un punto de vista basado en la dominancia de Pareto.

Todos los algoritmos han sido evaluados en un marco experimental adecuado, utilizando numerosos conjuntos de datos y comparando su rendimiento frente a otros métodos ya publicados de contrastada calidad. Los resultados obtenidos, que han sido verificados mediante la aplicación de test estadísticos no paramétricos, demuestran los beneficios de utilizar la metaheurística de programación automática con colonias de hormigas para dichas tareas de minería de datos.

Abstract

This Doctoral Thesis involves the first approximation of the ant programming metaheuristic to data mining. This automatic programming technique has demonstrated good results in optimization problems, but its application to data mining has not been explored until the present moment.

Specifically, this Thesis deals with the classification and association rule mining tasks of data mining. For the former, three models for the induction of rule-based classifiers are presented. Two of them address the classification problem from the point of view of single-objective and multi-objective evaluation, respectively, while the third proposal tackles the particular problem of imbalanced classification from a multi-objective perspective.

On the other hand, for the task of association rule mining two algorithms for extracting frequent patterns have been developed. The first one evaluates the quality of individuals by using a novel fitness function, while the second algorithm performs the evaluation from a Pareto dominance point of view.

All the algorithms proposed in this Thesis have been evaluated in a proper experimental framework, using a large number of data sets and comparing their performance against other published methods of proved quality. The results obtained have been verified by applying non-parametric statistical tests, demonstrating the benefits of using the ant programming metaheuristic to address these data mining tasks.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Lista de Acrónimos

ACO Ant Colony Optimization.

AI Artificial Intelligence.

AIS Artificial Immune System.

ANN Artificial Neural Network.

AP Ant Programming.

ARM Association Rule Mining.

AUC Area Under ROC Curve.

BNF Backus-Naur Form.

CFG Context-Free Grammar.

CGP Cartesian Genetic Programming.

CRS Collaborative Recommender System.

DM Data Mining.

EA Evolutionary Algorithm.

FNR False Negative Rate.

FPR False Positive Rate.

G3P Grammar Guided Genetic Programming.

GA Genetic Algorithm.

GP Genetic Programming.

IR Imbalance Ratio.

KD Knowledge Discovery.

MDLP Minimum Description Length Principle.

ML Machine Learning.

MOACO Multi-Objective Ant Colony Optimization.

MOO Multi-Objective Optimization.

NN Neural Network.

OVA One-Vs-All.

OVO One-Vs-One.

PSO Particle Swarm Optimization.

RBS Rule-Based System.

ROC Receiver Operating Characteristic.

SD Subgroup Discovery.

SI Swarm Intelligence.

STGP Strongly Typed Genetic Programming.

SVM Support Vector Machine.

TAG Tree-Adjoining Grammar.

TNR True Negative Rate.

TPR True Positive Rate.

TSP Traveling Salesman Problem.

1 Introducción

El volumen de datos mundial se incrementa diariamente a una tasa casi inimaginable. Por ejemplo, IBM afirma que en su compañía se generan 2.5 quintillones de bytes cada día [112]. Y quizás aún más asombroso, el 90 % de los datos que existen actualmente han sido generados en los últimos dos años.

A día de hoy es inevitable el no registrar y acumular datos en todos los ámbitos profesionales. De hecho, cada vez que se utiliza un motor de búsqueda, se adquiere un producto en un supermercado, se efectúa una transacción con una tarjeta de crédito, se emplea un teléfono móvil, o incluso se utiliza un GPS, se están generando grandes volúmenes de datos que las compañías almacenan. Muchos de esos datos no se están recolectando específicamente para un fin particular, pero es imposible no producirlos, con lo que los datos se van almacenando y acumulando de una forma natural. Los datos simplemente están ahí, y dados los avances en la tecnología y capacidad de almacenamiento de datos, es factible almacenarlos en grandes cantidades. Estos datos, en sí mismos, carecen de significado, y es necesario procesarlos para convertirlos en información, de manera que ofrezcan un significado, conocimiento, ideas o conclusiones. De esto se encargan los ordenadores, que también son asequibles. Sin embargo, muchas organizaciones no están empleando las herramientas adecuadas para procesar los datos de que disponen, con lo que

corren el riesgo de quedarse un paso por detrás con respecto a sus competidoras, al ser incapaces de llegar a comprender los datos y derivar dicho conocimiento en acciones que puedan ayudarlas a adoptar decisiones de negocio inteligentes.

La cuestión es, dado el acopio de datos de que disponemos, ¿podemos encontrar alguna información útil en ellos? Naturalmente, y de ello se encarga la **minería de datos** (*Data Mining*, DM). La DM es un área multidisciplinar donde se engloban ideas procedentes de diferentes disciplinas como el aprendizaje automático (*Machine Learning*, ML), la inteligencia artificial (*Artificial Intelligence*, AI), la estadística y los sistemas de bases de datos. A simple vista puede surgir la pregunta de en qué difiere la DM de la estadística, por ejemplo, que en su rama descriptiva también cubre la obtención de información a partir de los datos. Sin embargo, el objetivo de la DM difiere en gran medida de la estadística, que recoge datos específicamente para dar respuesta a alguna pregunta concreta, mientras que la DM pretende extraer información útil a partir de los datos de manera que sirva de apoyo en la toma de decisiones.

Debido a la cantidad ingente de datos, su alta dimensionalidad, heterogeneidad y distribución, puede no ser factible la aplicación de las técnicas tradicionales para, por ejemplo, encontrar patrones, tendencias o anomalías en los datos.

> Formalmente, la **DM** consiste en el proceso de descubrir o extraer conocimiento potencialmente útil, no trivial, y previamente desconocido a partir de los datos mediante la aplicación de algoritmos específicos [83]. Dicho proceso está dirigido hacia la obtención de un modelo, el cual dependerá del uso que se le que quiera dar a los datos.

Los objetivos que se marcan en un determinado proyecto donde se aplique la DM son los que determinan el modelo de conocimiento, así como el hecho de que un dato o una relación entre datos sean significativos. Un ejemplo de relación que podrá ser significativa o no dependiendo del objetivo podría ser el hecho de que los clientes que compran la marca X vivan en la periferia de la ciudad y los que compran la marca Y, en el centro. Dicho conocimiento puede ser útil si queremos

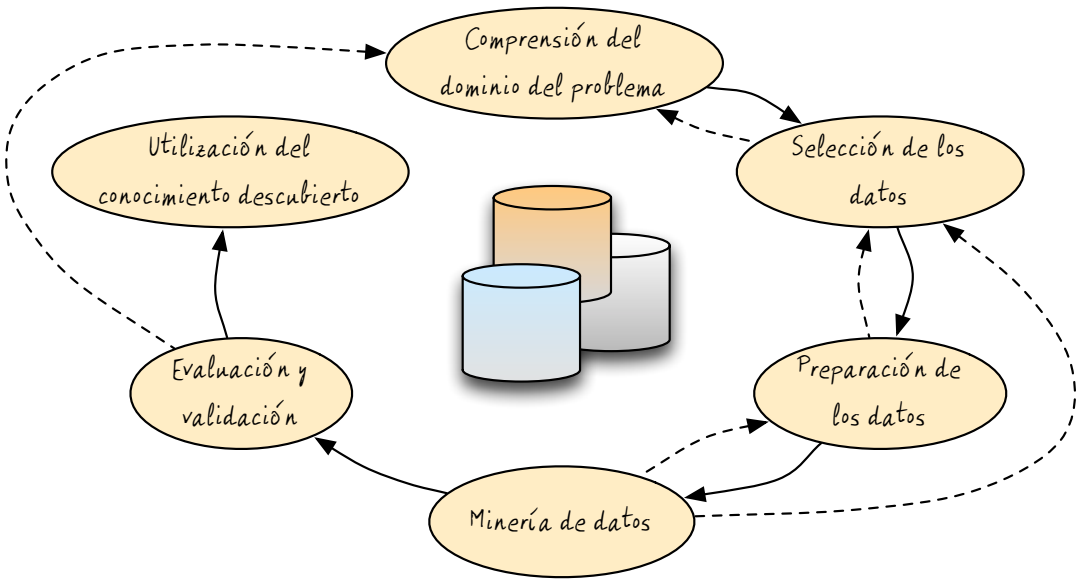


Figura 1.1: Fases del proceso de KD (adaptado de *Cios et al.* [48])

distinguir los patrones de compra de los ciudadanos dependiendo del área donde viven, pero será totalmente inservible para determinar la fidelidad de un cliente.

Mediante la DM se aporta un primer nivel de interpretación extrayendo relaciones de los datos en bruto. A partir de este nivel se puede extraer conocimiento aún más elaborado.

Previamente a la extracción de conocimiento útil a partir de los datos, es importante entender el **proceso global de descubrimiento de conocimiento** (KD, *Knowledge Discovery*), dentro del cual se encuadra la DM. Algunos autores utilizan ambos términos como sinónimos, pero en realidad el KD se refiere al proceso completo de extracción de conocimiento a partir de los datos, mientras que la DM es una etapa concreta de dicho proceso consistente en la aplicación de algoritmos específicos para extraer patrones a partir de los datos. Más en detalle, el proceso de KD consiste en una secuencia de pasos, eventualmente retroalimentados, que se deben seguir para encontrar patrones en los datos. Existen varios modelos en la literatura que describen el proceso de KD, normalmente focalizados en la perspectiva empresarial, académica o bien en una hibridación de ambas. En esta tesis consideramos el modelo de KD híbrido propuesto por *Cios et al.* [48], el cual contempla seis pasos, recogidos en la figura 1.1. Estas fases son: definición del problema

y comprensión del dominio del problema, selección de datos, preparación de los datos, DM propiamente dicha, evaluación y validación del modelo, interpretación del modelo e integración. Como se ha comentado anteriormente, no se trata de un proceso lineal, sino que se retroalimenta y continúa, y en el que nuevos cambios en la situación pueden provocar que el conocimiento deje de ser correcto, siendo preciso volver a extraer nuevo conocimiento.

La DM es la tarea más importante dentro del proceso de KD, al ser la encargada de encontrar información oculta en los datos, y contempla métodos de **aprendizaje supervisado**, donde los ejemplos o instancias están etiquetados, y **no supervisado**, donde se desconoce la clase de las instancias. Normalmente, los métodos supervisados tienen una naturaleza predictiva, mientras que los no supervisados tienden a ser descriptivos. Las tareas predictivas realizan inferencias a partir de los datos con el objetivo de hacer predicciones sobre nuevos datos (clase, categoría o valor numérico). Por su parte, las tareas descriptivas sirven para describir y caracterizar las propiedades generales de los datos según la observación de ciertos comportamientos (atributos).

Entre las tareas de DM **predictivas** podemos encontrar:

- **Clasificación** [66, 179]. Su finalidad es predecir la clase a la que pertenece una determinada instancia a partir de los valores de sus otros atributos, conocidos como atributos predictivos. Para ello, a partir de un conjunto de datos de entrenamiento se infiere un modelo o clasificador, que se utilizará para clasificar nuevas instancias o patrones no etiquetados en una de las categorías existentes.

- **Regresión** [181]. La tarea de regresión es similar a la de clasificación, pero en lugar de predecir la clase de una nueva instancia, se predice un valor numérico o resultado. En esencia, cada instancia contiene una serie de variables independientes o atributos que producen un resultado o variable dependiente. El objetivo es inducir un modelo para predecir el valor de una variable dependiente desconocida, dado el valor de los atributos. Un tipo específico de regresión, consistente en analizar secuencias de valores observados a lo largo

del tiempo (ordenados cronológicamente), es el **análisis de series temporales** [40]. En este caso, el objetivo es realizar una estimación de valores futuros en función del comportamiento pasado de la serie.

En cuanto a las tareas de índole **descriptiva**, podemos distinguir:

- **Minería de reglas de asociación** (ARM, *Association Rule Mining*) [5, 99]. Trata de descubrir relaciones de dependencia interesantes entre un conjunto de elementos. Suelen expresarse en forma de reglas *IF-THEN* mostrando parejas atributo-valor que ocurren frecuentemente juntas en un conjunto de datos dado. Un caso particular es la **minería de patrones secuenciales** [74], que consiste en la extracción de patrones frecuentes relacionados con el tiempo u otro tipo de secuencia. Estos patrones son similares a asociaciones donde las relaciones entre los datos se basan en el tiempo.

- **Agrupamiento** (*clustering*) [115]. Consiste en la agrupación de instancias en clases de objetos similares. Un *cluster* es una colección de ejemplos que comparten características comunes, pero que difieren de los ejemplos de otros *clusters*. Los algoritmos de clustering buscan dividir un conjunto de datos en subgrupos o *clusters* relativamente homogéneos, de manera que se maximiza la similitud de las instancias dentro del *cluster* y se minimiza la similitud con respecto a las instancias de fuera del mismo. Cuando el objetivo del agrupamiento es tratar de localizar instancias que no son similares a ninguna de las demás, éste se conoce con el nombre de **detección de desviaciones, casos extremos e instancias anómalas** [72, 157]. Estas instancias discordantes pueden representar casos potenciales de fraude, intrusos, fallos, etc.

Actualmente existen técnicas que se encuadran a medio camino entre DM descriptiva y predictiva, conocidas como **inducción supervisada de reglas descriptivas** [158], que comentamos a continuación:

- **Descubrimiento de subgrupos** (SD, *Subgroup Discovery*) [58, 103]. Dado un conjunto de datos y una propiedad de interés en ellos, el descubrimiento de subgrupos lleva a cabo la búsqueda de conjuntos de datos tan grandes como sea posible y que son estadísticamente más interesantes con respecto a la propiedad de interés.

- **Minería de contraste de conjuntos** [21]. Estrechamente relacionada con la ARM, trata de descubrir condiciones particulares que se producen entre los atributos y sus valores, de manera que sirvan para diferenciar los grupos de modo significativo. Se trata de pares atributo-valor que han de cumplir la restricción de que un atributo dado no puede aparecer más de una vez.
- **Minería de patrones emergentes** [9]. Se centra en la búsqueda de patrones cuyas frecuencias en dos clases difieren en una alta proporción.

Las tareas de DM y partes del proceso de KD se pueden abordar o enfocar como problemas de optimización y búsqueda, al tratarse de problemas difíciles de modelar y que presentan un espacio de soluciones grande. A tal efecto se pueden emplear **modelos computacionales bioinspirados** [79], un grupo de técnicas tolerantes a cierta imprecisión e incertidumbre que modelan de forma aproximada fenómenos existentes en la naturaleza.

Los **algoritmos evolutivos** (EAs, *Evolutionary Algorithms*) [121], por ejemplo, se inspiran en los principios Darwinianos de la evolución natural. Estos algoritmos imitan los mecanismos de selección natural y supervivencia del más apto para resolver problemas de la vida real, y se ha demostrado su aplicabilidad al desarrollo de algoritmos de DM [84].

Un modelo computacional encuadrado dentro del grupo de EAs son los **algoritmos genéticos** (GAs, *Genetic Algorithms*), propuestos por *Holland* [108]. A grandes rasgos, un GA consiste en una población de individuos (denominados cromosomas) que codifican una posible solución al problema. Cada cromosoma tiene asociada una función de aptitud que evalúa su validez para resolver el problema y que determina las posibilidades de reproducción del cromosoma. Así pues, la población de individuos evoluciona en el tiempo por medio de un proceso de selección y cruce de individuos, pudiéndose producir también con cierta probabilidad mutaciones en los cromosomas.

La **programación genética** (GP, *Genetic Programming*) [128] es otro tipo EA que se considera, en esencia, una variante de los GAs. De hecho, se puede definir

como un GA que usa un lenguaje de representación más complejo (normalmente un árbol) para la codificación de los individuos y cuyo objetivo es la evolución de programas de ordenador para resolver un problema concreto. Advirtamos que ni siquiera es necesario conocer de antemano la estructura o forma de la solución, sino que simplemente hay que especificar los bloques básicos de los que se compone cualquier programa o individuo y la forma de cuantificar su adecuación como solución al problema. Además de evolucionar programas de ordenador, la GP se usa para evolucionar otras abstracciones de conocimiento, tales como expresiones matemáticas o sistemas basados en reglas.

Otro paradigma de computación bioinspirado que sirve para abordar problemas de optimización y que se ha aplicado con éxito a DM es la **inteligencia colectiva** (SI, *Swarm Intelligence*) [26]. Dicha disciplina se ocupa del diseño de sistemas multiagente, es decir, compuestos de muchos individuos, y se inspira en las conductas descentralizadas y colectivas que algunas sociedades de insectos y otros animales presentan en la naturaleza. Concretamente, la SI se centra en el comportamiento colectivo que emerge de las interacciones simples que se producen entre los propios individuos y entre estos y el entorno. Ejemplos de sistemas que imita la SI pueden ser las colonias de hormigas [147], termitas [204], abejas [118], avispas [195], bancos de peces y bandadas de pájaros [107, 223].

Dentro de la SI se emplaza una metaheurística bioinspirada denominada **optimización mediante colonias de hormigas** (ACO, *Ant Colony Optimization*) [64], que se inspira en el comportamiento y capacidades autoorganizativas de las colonias de hormigas en la naturaleza. Diversos algoritmos de ACO han demostrado su habilidad y capacidad para la extracción de reglas de clasificación [147, 179], y también se ha adaptado esta metaheurística a la tarea de asociación bajo restricciones multidimensionales [132]. Sin embargo, su aplicación a la tarea de asociación y a otras tareas de DM no ha sido suficientemente explorada, como refleja el estudio abordado recientemente por *Martens et al.* [146].

Una variación de los algoritmos de ACO es la **programación automática con hormigas** (AP, *Ant Programming*) [1, 193], que incrementa la capacidad de construir programas automáticamente utilizando ACO como técnica de búsqueda, pero su aplicación al campo de la DM no ha sido investigada hasta la fecha. En cambio, otro tipo de programación automática, la GP, sí que ha sido aplicada con éxito a las tareas de clasificación y asociación de DM.

Dados los buenos resultados alcanzados por la metaheurística de ACO para la tarea de clasificación [179], y por la GP tanto para la minería de reglas de clasificación [70] como de asociación [144], consideramos que es un reto interesante explorar el comportamiento y desempeño de modelos basados en AP en su aplicación a dichas tareas, y su comparación con otros algoritmos de referencia. Además, no existe una revisión general de los métodos de AP existentes. Por todo ello, una de las metas de esta tesis consiste en efectuar una revisión bibliográfica de las publicaciones sobre AP existentes en la literatura. Una vez llevada a cabo, se desarrollarán modelos de AP específicos para las tareas de clasificación y asociación.

1.1. Objetivos

El **objetivo general** de esta tesis doctoral es estudiar el uso del paradigma de AP para el aprendizaje, en concreto para la extracción de reglas de clasificación y de asociación, desarrollando algoritmos basados en esta metaheurística para abordar dichas tareas.

El objetivo anterior puede desglosarse en los siguiente **subobjetivos**:

- Realizar un estudio teórico de los algoritmos basados en la metaheurística de ACO existentes para minería de reglas clasificación y asociación.

- Realizar una revisión bibliográfica de las distintas propuestas de AP existentes en la literatura.
- Desarrollar un modelo monobjetivo de AP gramatical para la tarea de clasificación que extraiga reglas de clasificación comprensibles, en forma *IF-THEN*.
- Abordar el problema de clasificación desde un enfoque multiobjetivo [12], útil para optimizar los diferentes objetivos contradictorios inherentes a los problemas de aprendizaje. Desarrollar una adaptación del modelo desarrollado a dicho enfoque que alcance un buen equilibrio entre exactitud y comprensibilidad.
- Desarrollar un modelo de AP para clasificación no balanceada capaz de abordar tanto conjuntos de datos binarios como multiclase.
- Desarrollar un modelo monobjetivo de AP basado en gramática para la tarea de asociación que busque reglas de asociación frecuentes.
- Abordar la extracción de reglas de asociación frecuentes desde un punto de vista multiobjetivo que busque maximizar el soporte y la confianza de las reglas simultáneamente.
- Analizar experimentalmente el rendimiento de los modelos desarrollados, comparándolos con los algoritmos más relevantes de otros paradigmas sobre diversos conjuntos de datos y analizando los resultados obtenidos estadísticamente.

1.2. Estructura

Para la consecución de los objetivos enumerados en el apartado anterior, la memoria se ha organizado del siguiente modo:

- **Capítulo 2**. **Aprendizaje basado en reglas**. Introduce las tareas de clasificación y asociación de DM, proporcionando una taxonomía de las mismas y una descripción de las medidas empleadas para evaluar la calidad de las reglas.

- **Capítulo 3. Programación automática con colonias de hormigas. Investigación del estado del arte**. Se introducen los algoritmos bioinspirados, prestando especial atención a la GP y a la metaheurística de ACO. También se presenta una revisión de los algoritmos de ACO aplicados a las tareas de clasificación y asociación. Por último, se define la AP, realizando una revisión de los trabajos existentes hasta la fecha.

- **Capítulo 4. Modelo de AP para clasificación multiclase**. En este capítulo se introduce un primer modelo de AP para clasificación multiclase, cuya principal característica radica en el uso de una gramática que guía la creación de individuos, la presencia de una doble componente en la función heurística y en un enfoque de nichos para para la asignación de un consecuente a las reglas y para seleccionar aquellas que formarán parte del clasificador final, ordenadas en forma de lista de decisión.

- **Capítulo 5. Modelo de AP multiobjetivo para clasificación multiclase**. Este capítulo comienza explicando la optimización multiobjetivo, para exponer a continuación una propuesta de AP basada en este tipo para la tarea de clasificación. Se trata de una evolución del modelo monobjetivo presentado en el capítulo 4 cuya principal aportación al campo estriba en la estrategia de evaluación de individuos propuesta, la cual puede ser adoptada por cualquier otro algoritmo de clasificación.

- **Capítulo 6. Modelo de AP para clasificación de datos no balanceados binarios y multiclase** describe el problema de clasificación de datos no balanceados y presenta la propuesta de AP para clasificación en dicho dominio, capaz de tener en cuenta la distribución de las clases. Nuestro algoritmo sirve tanto para la clasificación de conjuntos de datos binarios como multiclase, lo cual supone su principal ventaja respecto a otros algoritmos de clasificación no balanceada. Además, no requiere un preprocesamiento de los datos, sino que aborda el problema directamente.

- **Capítulo 7. Modelos de AP para minería de reglas de asociación**. Presenta dos modelos de AP desarrollados para la extracción de reglas de asociación frecuentes. Para el primer algoritmo, que sigue un enfoque monobjetivo, se ha presentado una función de fitness que considera la agregación

de las dos medidas que se busca maximizar. Por contra, el segundo modelo sigue un punto de vista multiobjetivo donde se maximizan simultáneamente ambas medidas.

- Capítulo 8. **Conclusiones y trabajo futuro**. En este capítulo se pone fin a la tesis doctoral resumiendo los conocimientos científicos resultantes e introduciendo posibles líneas de investigación futuras. Asimismo, presenta las publicaciones científicas asociadas a la tesis. Dado que la tesis se adecúa al formato exigido por la Mención Internacional del Título de Doctor, el resumen y las conclusiones también han sido redactados en inglés.

2

Aprendizaje basado en reglas

La idea que subyace del aprendizaje automático es que una computadora pueda resolver por sí misma determinados problemas que requieren ciertas habilidades más allá de una mera capacidad de cálculo. Para ello lleva a cabo un aprendizaje donde es necesario indicarle a partir de qué va a aprender (datos), cuál es el objetivo a cumplir y qué tipo de resultados pretendemos que ofrezca. Tal como se introdujo en el capítulo anterior, se distinguen dos tipos de aprendizaje, supervisado y no supervisado. La principal diferencia radica en que, en el primer caso, los ejemplos están etiquetados, mientras que en el segundo no se dispone de información relativa a la clase. Por tanto, el objetivo de las tareas de aprendizaje supervisado suele ser predictivo, mientras que el aprendizaje no supervisado se centra en tareas de índole descriptiva.

En ambos casos, la forma en que se presenta el conocimiento adquirido suele ser crucial, tomando especial relevancia en problemas como diagnóstico médico, finanzas, o *marketing*, donde los expertos del dominio pueden utilizar dicho conocimiento inferido como soporte y ayuda en la toma de decisiones, en lugar de confiar ciegamente en la salida de algoritmos que actúan como cajas negras. Debido a ello, a su capacidad de expresión, sencillez y escalabilidad, en los últimos años se ha extendido mucho el empleo de **sistemas basados en reglas** (RBSs, *Rule-Based*

Systems) [239]. En esencia, el componente fundamental de un RBS es una base o colección de reglas de la forma `IF` *antecedente* `THEN` *consecuente*, donde el antecedente representa las premisas que deben observarse para que la regla sea aplicable, y el consecuente, conclusiones que se pueden alcanzar o bien acciones que se derivan de la aplicación de la regla. Así, dependiendo de las circunstancias presentes en un momento dado, un RBS tratará de disparar o elegir una regla que se adecúe a las mismas, de forma que se adopten las acciones indicadas en su consecuente. Es importante resaltar que en función de su finalidad y la naturaleza del conocimiento que representan, se pueden distinguir reglas de clasificación, de asociación, de predicción, causalidad, etc.

Independientemente de la finalidad, las reglas también se pueden catalogar en base al tipo de los atributos que manejan. Así, podemos encontrarnos con reglas binarias, si contienen atributos de tipo binario; reglas categóricas o nominales, si el conjunto de datos presenta atributos de este tipo; reglas numéricas, cuando existen atributos numéricos que, por tanto, poseen un orden implícito (en este caso sí que existen algoritmos que necesitan transformar este tipo de atributos en categóricos aplicando un proceso de discretización); y reglas difusas, cuando el conjunto de datos contiene atributos numéricos que se tratan siguiendo los principios de la lógica difusa, esto es, dividiendo un atributo numérico en varios conjuntos y permitiendo la pertenencia parcial a los distintos grupos. Así, mediante esta representación se asume que un conjunto difuso corresponde a una serie de conjuntos donde cada uno de ellos contiene el valor verdadero del objeto con una probabilidad mayor o igual que un determinado umbral [57, 176]. Nótese que estas categorías no son excluyentes, con lo que una regla puede pertenecer a más de una a la vez.

La tarea de aprendizaje supervisado en la que nos centraremos en esta tesis será la de **minería de reglas de clasificación**. El objetivo de esta tarea es inducir un modelo o clasificador a partir de una serie de ejemplos etiquetados denominado conjunto de entrenamiento, de manera que tome en consideración las relaciones ocultas entre los valores de los atributos predictivos y el atributo clase. De esta forma, el modelo aprendido puede aplicarse posteriormente a nuevos datos no etiquetados para asignar a cada instancia una de las clases existentes.

Para abordar la tarea de clasificación se han empleado una gran variedad de algoritmos y técnicas, incluyendo árboles de decisión [98], reglas de decisión [127], *naïve*

Bayes [110], máquinas de vector soporte (SVMs, *Support Vector Machines*) [111], o redes neuronales (NNs, *Neural Networks*) [101], entre otras. Sin embargo, las técnicas con una representación de alto nivel como las reglas y los árboles de decisión [127] proporcionan a cualquier usuario o al propio experto del dominio la ventaja de ser comprensibles, lo que permite interpretar y entender el conocimiento extraído. Por ejemplo, en problemas médicos, las reglas de clasificación extraídas pueden ser verificadas por expertos médicos, proporcionando una mejor comprensión del problema [216]. En el caso de que un clasificador esté compuesto de reglas del tipo `IF` *antecedente* `THEN` *consecuente*, cada una de ellas contendrá en su antecedente una serie de requisitos en forma de condiciones, que son los que se deben producir para que se pueda considerar que una instancia pertenezca a la clase identificada por el consecuente de la regla.

La **minería de reglas de asociación** es la otra tarea de DM que abordamos en esta tesis, y que se encuadra dentro del aprendizaje no supervisado. Consiste en descubrir relaciones interesantes entre los atributos en grandes conjuntos de datos. Originalmente se trataba de una tarea fuertemente relacionada con el análisis de la cesta de la compra, en la que, examinando los datos relativos a las compras efectuadas por los clientes en los comercios, se hallaban relaciones del tipo: si el cliente compra ginebra y compra tónica, entonces también comprará limas. El departamento de *marketing* de estas grandes superficies utilizaba posteriormente las asociaciones encontradas por las reglas para cuestiones relativas a la ubicación de los productos, promociones, etc. Dado el potencial y aplicabilidad de las reglas de asociación, su uso se ha extendido a otros ámbitos tales como el descubrimiento de intrusos [211], el comercio electrónico [38], los sistemas de recomendación [178], la medicina y la medicina deportiva [16], entre otros.

Al igual que en la minería de reglas de clasificación, este tipo de relaciones también se prestan a codificarse utilizando reglas, aunque en este contexto se representan como $A \rightarrow C$, donde A hace referencia al antecedente de la regla y C al consecuente. Sin embargo, la diferencia estriba en que su objetivo no pasa por etiquetar los ejemplos que cubre el antecedente con una de las clases disponibles, sino en encontrar relaciones entre cualquier tipo de atributos, por lo que en el consecuente de la regla puede aparecer cualquier variable. El significado de una regla $A \rightarrow C$

es que si todas las condiciones de A están presentes en la transacción, entonces es bastante probable que las condiciones de C también lo estén.

Para evaluar la calidad de las reglas habrá que tener en cuenta la finalidad de las mismas, distinguiendo entre medidas de calidad de carácter predictivo, que son las empleadas en clasificación, y de carácter descriptivo, empleadas en las reglas de asociación.

En este capítulo nos centraremos en explicar las tareas de clasificación y asociación de DM, incluyendo las medidas para determinar la calidad de las reglas y la forma de evaluar el conocimiento extraído. En el anexo A se explica la fase del proceso de KD previa a la de DM, la de preparación de datos, y que fue introducida en la Figura 1.1, dado que supone un aspecto fundamental a considerar para que el proceso de DM sea más fácil y efectivo.

2.1. Minería de reglas de clasificación

Matemáticamente, el objetivo de un problema de clasificación compuesto por un conjunto de datos con i ejemplos o instancias $D = \{d_1, d_2, ..., d_i\}$, cada una descrita por una serie de observaciones o valores de sus atributos predictivos $X = \{x_1, x_2, ..., x_z\}$, y etiquetada con una de las clases existentes $C = \{c_1, c_2, ..., c_n\}$, es inducir una función o clasificador que permita representar la correspondencia existente entre los ejemplos y las clases:

$$f : X(D) \rightarrow C \tag{2.1}$$

Las reglas de clasificación son una forma simple e interpretable de representar el conocimiento [98]. Una regla estará compuesta de dos partes, el antecedente y el consecuente. El antecedente contiene una combinación de condiciones sobre los atributos predictivos, conectadas entre sí normalmente mediante el operador lógico AND, aunque cualquier otro operador lógico puede utilizarse para conectar las

condiciones. Por su parte, el consecuente indicará la clase que predice la regla. De esta forma, una regla cubre una instancia si esta satisface las condiciones contenidas en el antecedente, y en tal caso le asigna la clase predicha por el consecuente.

Las condiciones del antecedente suelen implicar comparaciones simples en las cuales un atributo se compara con un valor de su dominio mediante un operador relacional. En tal caso, los clasificadores reciben el nombre de clasificadores lineales. Sin embargo, hay ocasiones en las que se permiten comparaciones entre el valor de un atributo con el valor de otro, llevando a clasificadores multivariantes.

2.1.1. Taxonomía

En esta sección se presenta una taxonomía de la tarea de clasificación, donde se observa que se pueden distinguir varios tipos en función de distintos criterios:

- Dependiendo del número de clases en el conjunto de datos: **binaria** o **multiclase** [126]. En la clasificación binaria sólo existen dos clases no solapadas, excluyentes, mientras que en el tipo multiclase los problemas presentan más de dos clases, siempre sin solapamiento.

- En función del número de clases que se pueden asignar por instancia: **monoetiqueta** o **multietiqueta** [34]. El caso en el que cada ejemplo o patrón pertenece a una única clase se corresponde con el problema de clasificación clásico, y se denomina monoetiqueta. Sin embargo, en determinadas situaciones un patrón puede pertenecer a más de una clase. Así, si estuviéramos clasificando artículos científicos, cada uno tendrá asociado un conjunto de palabras clave o *keywords*. Por tanto, clasificar un artículo implicará etiquetarlo con una o más palabras del vocabulario controlado de *keywords*, es decir, etiquetarlo con una o más clases del conjunto de clases. Este tipo de aprendizaje donde cada patrón puede tener asociada simultáneamente una o más etiquetas no excluyentes se denomina multietiqueta.

- En función del número de instancias: **monoinstancia** o **multinstancia** [240]. La diferencia entre ambos reside en que en el primero existe un conjunto de instancias que se etiquetan independientemente, mientras que en el segundo existen bolsas compuestas por muchas instancias y lo que se etiqueta no es

cada una de ellas por separado, sino que la etiqueta de clase es observable para toda la bolsa de instancias. Una bolsa se etiqueta como positiva si existe al menos una instancia positiva, mientras que para que se etiquete como negativa todas las instancias que contenga han de ser negativas.

- En función de la pertenencia entre clases: **clasificación plana** o **jerárquica** [156]. La inmensa mayoría de problemas de clasificación consideran que cada instancia se asigna a una clase de un conjunto finito de clases planas. Sin embargo, existen problemas de clasificación más complejos donde las clases a predecir están relacionadas jerárquicamente. En este tipo de problemas, una o más clases pueden subdividirse en subclases, o bien ser agregadas en superclases.

2.1.2. Medidas de rendimiento

El rendimiento de un clasificador se mide normalmente en función de la **exactitud**, la **interpretabilidad** y la **eficiencia**. La primera de ellas es la que más importancia tiene a la hora de determinar la calidad de un clasificador, y se suele estimar tras ejecutar el clasificador sobre diferentes conjuntos de datos. La interpretabilidad de un clasificador, se trata, en cierto modo, de una cuestión subjetiva, ya que lo que puede ser un modelo comprensible para un usuario puede no serlo para otro. En cualquier caso, si el clasificador está compuesto por una serie de reglas, sí que se puede afirmar que cuanto menor sea su número y más cortas sean (menos condiciones tengan), más comprensible será el modelo [217]. Por último, la eficiencia hace referencia al tiempo que se necesita para inducir el clasificador y usarlo sobre nuevos datos.

Como se explicará posteriormente en el capítulo 3, los algoritmos bioinspirados que evolucionan clasificadores basados en reglas necesitan evaluar la bondad de las reglas que van extrayendo, asignándoles una **aptitud** o ***fitness***.

Para un problema de clasificación binaria, las medidas de rendimiento se pueden obtener de una **matriz de confusión** o tabla de contingencia de cuatro celdas que muestra las relaciones entre los valores reales y los predichos, como se muestra en la Tabla 2.1.

		Clase Predicha	
		Positivo	Negativo
Clase Real	Positivo	Verdaderos Positivos (T_P)	Falsos Negativos (F_N)
	Negativo	Falsos Positivos (F_P)	Verdaderos Negativos (T_N)

Tabla 2.1: Matriz de confusión para un problema de clasificación binaria

donde:

- T_P (*True Positive*) son los verdaderos positivos, es decir, las instancias positivas que son correctamente clasificadas como positivas,
- F_P (*False Positive*) son los ejemplos negativos clasificados erróneamente como positivos,
- T_N (*True Negative*) es el conjunto de instancias negativas clasificadas como negativas, y
- F_N (*False Negative*) son ejemplos positivos clasificados incorrectamente como negativos.

La matriz de confusión recoge los ejemplos correcta e incorrectamente reconocidos para cada clase. Para un problema multiclase de N clases, la matriz de confusión tendrá una dimensión $N \times N$, como se observa en la Tabla 2.2.

Sin embargo, este tipo de problemas se puede reducir a problemas de clasificación binaria si cada clase se considera de forma separada frente a la unión del resto de clases, obteniendo por tanto N matrices de confusión. Se puede observar un

ejemplo en la Figura 2.1, donde para un problema de cinco clases se muestra cómo se obtendrían los T_P, F_N, F_P y T_N para la clase C.

Clase Real \ Clase Predicha	C_1	C_2	...	C_N	Total
C_1	h_{11}	h_{12}	...	h_{1N}	T_{r1}
C_2	h_{21}	h_{22}	...	h_{2N}	T_{r2}
⋮			⋱		⋮
C_N	h_{N1}	h_{N2}	...	h_{NN}	T_{rN}
Total	T_{c1}	T_{c2}	...	T_{cN}	T

Tabla 2.2: Matriz de confusión para un problema de clasificación con N clases

donde:

- N es el número de clases,
- h_{ij} representa el número de ejemplos de la clase C_i predichos como C_j,
- T_{ri} es el número ejemplos en la fila i,
- T_{ci} es el número ejemplos en la columna i, y
- T es el número total de instancias

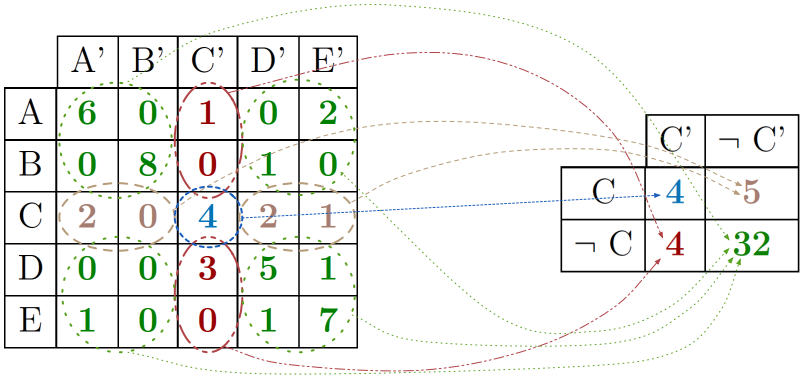


Figura 2.1: Reducción de una matriz de confusión 5x5

A partir de los valores recogidos en la matriz de confusión se pueden calcular diferentes medidas para evaluar la calidad de las reglas. Estas mismas medidas son

las que se emplean posteriormente para evaluar la calidad de un clasificador (ver Sección 2.1.3). A continuación se enumeran dichas medidas de rendimiento:

- **Exactitud predictiva** (*accuracy*): es la proporción del número de predicciones correctas sobre el total de predicciones.

$$accuracy = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} = \frac{predicciones\ correctas}{total\ de\ predicccciones} \tag{2.2}$$

- **Precisión** (*precision*): proporción de predicciones positivas correctas.

$$precision = \frac{T_P}{T_P + F_P} \tag{2.3}$$

- **Sensibilidad** (*recall* o *sensitivity*) o **tasa de verdaderos positivos** (TPR, *True Positive Rate*): indica la proporción de casos positivos correctamente identificados.

$$sensitivity = \frac{T_P}{T_P + F_N} \tag{2.4}$$

- **Especificidad** (*specificity*) o **tasa de verdaderos negativos** (TNR, *True Negative Rate*): proporción de casos negativos correctamente identificados. Tanto la sensibilidad como la especificidad son medidas ampliamente utilizadas en aplicaciones a medicina y biomedicina. Ambas medidas estiman el rendimiento del clasificador centrándose en una única clase (la sensibilidad en la positiva, y la especificidad en la negativa).

$$specificity = \frac{T_N}{T_N + F_P} \tag{2.5}$$

- **Tasa de falsos positivos** (FPR, *False Positive Rate*): proporción de casos negativos incorrectamente clasificados como positivos.

$$FPR = \frac{F_P}{T_N + F_P} = 1 - specificity \tag{2.6}$$

- **Tasa de falsos negativos** (FNR, *False Negative Rate*): proporción de casos positivos incorrectamente clasificados como negativos.

$$FNR = \frac{F_N}{T_P + F_N} = 1 - sensitivity \tag{2.7}$$

- **F-medida** (*F-measure*): combina precisión y *recall* como medida de la efectividad en el proceso de clasificación, en función de la importancia relativa de dichas medidas, controladas mediante un parámetro β. De esta manera, favorece la precisión cuando $\beta > 1$, y el *recall* en otro caso. Generalmente se establece $\beta = 1$, con lo que la F-medida será igual a la media armónica entre precisión y *recall*, recibiendo el nombre de F_1-medida.

$$F - measure = \frac{(1+\beta)^2 \cdot (precision \cdot recall)}{\beta^2 \cdot precision + recall} \tag{2.8}$$

$$F_1 - measure = \frac{2 \cdot precision \cdot recall}{precision + recall} = \frac{2 \cdot T_P}{2 \cdot T_P + F_P + F_N} \tag{2.9}$$

- **Coeficiente *Kappa*** [50]: tiene en consideración las distribuciones marginales para la matriz de confusión, compensando los aciertos aleatorios. Su propósito original era medir el grado de acuerdo o desacuerdo entre dos personas observando un mismo fenómeno. El coeficiente *Kappa* de Cohen se puede adaptar a tareas de clasificación, y su valor proporciona una idea acerca del porcentaje de clasificación correcta, una vez que se elimina el componente de aleatoriedad. Un valor de *Kappa* superior a 0 es indicativo de que el clasificador se está comportando mejor que un clasificador aleatorio. Para una matriz de confusión de $N \times N$ como la de la Tabla 2.2, el coeficiente *Kappa* se define como la concordancia observada, calculada como la suma de las frecuencias en la diagonal principal, con respecto al valor del estadístico bajo la concordancia aleatoria:

$$Kappa = \frac{T \sum_{i=1}^{N} h_{ii} - \sum_{i=1}^{N} T_{ri} \cdot T_{ci}}{T^2 - \sum_{i=1}^{N} T_{ri} \cdot T_{ci}} \tag{2.10}$$

  donde:

  h_{ii} son el número de ejemplos en la diagonal principal, es decir, el número de T_P para cada clase.

- **Área bajo la curva** ROC: la curva ROC es una gráfica cuyo eje X representa la FPR y el eje Y indica la TPR. El punto (0,1) representará el clasificador perfecto, ya que identifica todos los casos positivos y negativos

correctamente. El punto (0,0) representa un clasificador que predice todos los casos como negativos, mientras que el (1,1) corresponde a un clasificador que predice todos los casos como positivos. Por último, el punto (1,0) representa un clasificador que es incorrecto en todas las predicciones.

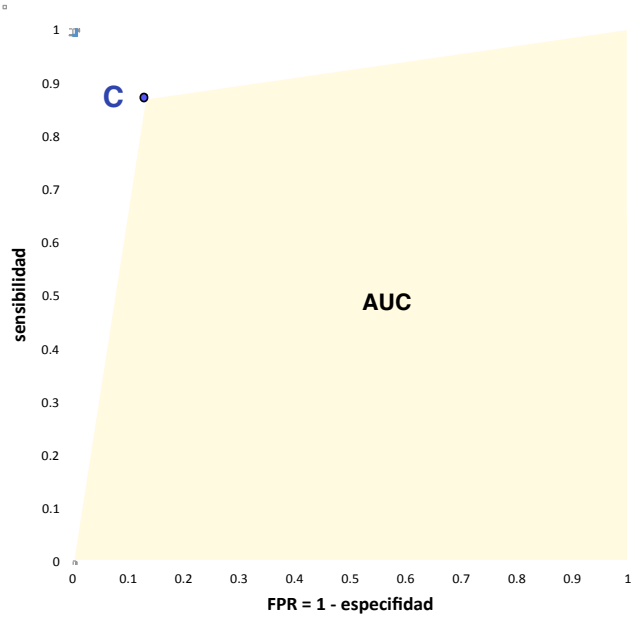


Figura 2.2: Curva ROC para un clasificador discreto

La curva ROC permite visualizar en dos dimensiones el rendimiento de un clasificador, mostrando la relación existente entre la sensibilidad y la especificidad. Sin embargo, para comparar algoritmos se necesita una medida escalar del rendimiento del clasificador. Por este motivo se propuso el área bajo la curva (AUC, *Area Under the ROC Curve*) [71], cuyo valor en la práctica estará siempre comprendido entre 0,5 y 1. Un clasificador aleatorio tendrá un valor de AUC de 0,5, y su curva ROC asociada coincidirá con la diagonal. En la Figura 2.2 se muestra la curva ROC para un clasificador discreto que obtiene una sensibilidad y una especificidad del 87 %, y donde el área de color representa el AUC. Para una única ejecución de un algoritmo, mediante transformaciones trigonométricas simples, se demuestra que el AUC es igual a la media aritmética entre sensibilidad y especificidad:

$$AUC = \frac{1 + TPR - FPR}{2} = \frac{sensitivity + specificity}{2} \tag{2.11}$$

2.1.3. Evaluación del rendimiento de un clasificador

La Sección 2.1.2 introduce las medidas empleadas habitualmente para determinar la capacidad del clasificador para etiquetar correctamente objetos nuevos, rendimiento que se calcula utilizando el mismo conjunto de datos que se usa para construir el clasificador.

Sin embargo, para inducir un clasificador no se utilizan todas las instancias del conjunto de datos, ya que a la hora de evaluar el rendimiento del mismo no se dispondría de instancias sobre las que evaluarlo. Esto es lógico, ya que si en su lugar se probase sobre un conjunto de instancias ya utilizadas para entrenar el clasificador, el resultado sería optimista, porque las particularidades de dichas instancias se habrían tenido en cuenta al inferir el clasificador. Este efecto es conocido con el nombre de **sobrentrenamiento**.

Previamente a la inducción de un clasificador, el conjunto de datos original se divide en subconjuntos de **entrenamiento** (***training***) y **prueba** (**test**), mutuamente excluyentes. Una vez hecho esto se induce el modelo empleando el conjunto de datos de entrenamiento. Finalmente, se calculan los resultados que el clasificador extraído arroja para dichas medidas sobre el conjunto de datos de test, compuesto de instancias etiquetadas que no fueron empleadas durante la fase de construcción del mismo.

Los clasificadores se construyen para ser utilizados sobre datos nuevos, desconocidos, por lo que es necesario validarlos. Para conocer cuál será la capacidad de **generalización** de un clasificador, esto es, para estimar cómo se va a comportar el clasificador ante nuevas instancias, es conveniente emplear un tercer conjunto de datos: el **conjunto de validación**. Dicho conjunto estará compuesto por ejemplos nuevos procedentes del mismo dominio que el conjunto de datos inicial. Sin embargo, es frecuente no disponer de dicho conjunto de validación. En cualquier caso, mientras se siga el procedimiento descrito en la Figura 2.3, particionando los datos de modo que se evite el sobrentrenamiento, las medidas de rendimiento

obtenidas para el clasificador nos indicarán con bastante precisión la capacidad de generalización del mismo.

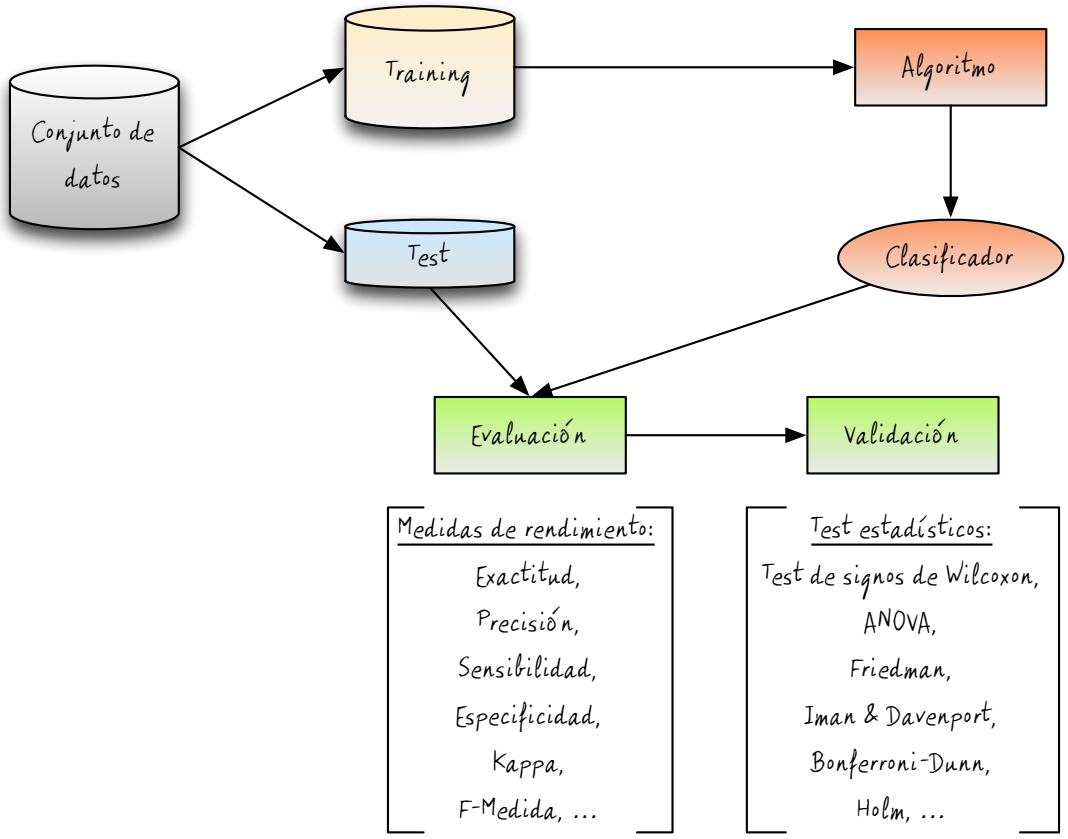


Figura 2.3: Procedimiento para calcular y comparar el rendimiento de un algoritmo

En la misma figura también se observa que es posible establecer comparaciones de rendimiento entre algoritmos. Para ello se llevan a cabo test estadísticos, que permiten extraer conclusiones significativas. La utilización de un test determinado u otro vendrá dada por el tipo de comparación a realizar, entre dos algoritmos o múltiple, y por el tipo de datos, que implicará la utilización de test paramétricos o no paramétricos [59]. Una explicación detallada de ello y de los test estadísticos más empleados se puede encontrar en el Anexo B.

El **principio de parsimonia** (***Occam's razor***) [241] se aplica frecuentemente en clasificación y establece que, en el caso de que haya que escoger entre clasificadores con un rendimiento similar en cuanto a exactitud, se debe elegir preferentemente el modelo menos complejo. Se debe hacer esto puesto que, además de ser más fácilmente interpretable, se demuestra que el clasificador posee una mayor capacidad de generalización.

En la práctica, el tamaño del conjunto de datos es limitado, por lo que para evaluar el rendimiento de un clasificador se hace necesario dividir dicho conjunto de datos en uno o más pares de conjuntos de entrenamiento y test. Existen diversas formas de particionar el conjunto de datos original [230], y a continuación presentamos las más conocidas.

2.1.3.1. Hold-out

El método de *hold-out* consiste en particionar el conjunto de datos en un único conjunto de entrenamiento y uno de prueba, siendo ambos subconjuntos mutuamente excluyentes. Es usual designar $^2/_3$ de los datos para el conjunto de entrenamiento y $^1/_3$ para el de test, de manera que el modelo se genera a partir del conjunto de entrenamiento y se prueba en el de test.

El problema de este método es que asume que los datos en ambos conjuntos van a tener las mismas características, cuando lo normal es que este hecho no se cumpla dado que el particionamiento se realiza al azar. Para solucionar esto último, se puede proceder generando los dos **conjuntos estratificados**, esto es, de forma que mantengan la misma distribución de instancias por clase que existía en el conjunto de datos completo. No obstante, con ello no es suficiente para eliminar el sesgo inherente al particionamiento aleatorio, por lo que en la práctica siempre existe cierta sensibilidad al particionamiento efectuado.

Una variación del método *hold-out* en la que éste se repite k veces es la denominada **remuestreo aleatorio** (***random subsampling***). En ella, el algoritmo se ejecutará k veces sobre unas particiones de entrenamiento y test diferentes, ya que

habrán sido generadas aleatoriamente con una semilla distinta, y para cada ejecución se calculará su rendimiento en cuanto a la medida en que estemos interesados. Así pues, se terminarán obteniendo k valores de dicha medida, y el rendimiento del clasificador vendrá dado por el cálculo de la media sobre los mismos.

2.1.3.2. Validación cruzada

En la **validación cruzada con k particiones** (***k-fold cross-validation***), el conjunto de datos inicial se divide aleatoriamente en k particiones o subconjuntos mutuamente exclusivos, cada uno con el mismo tamaño aproximado. Para hallar el rendimiento del algoritmo, éste se ejecuta k veces sobre un conjunto de entrenamiento formado por $k-1$ particiones, y dejando la partición restante como conjunto de test. Es decir, en la i-ésima evaluación, la partición P_i se reserva como conjunto de test, y la combinación de las particiones restantes forma el conjunto de entrenamiento sobre el que se entrena el clasificador.

En este caso, cada una de las k particiones se empleará el mismo número de veces para entrenar, y una única vez como conjunto de test. Como ventaja de la validación cruzada cabe destacar que se reduce la dependencia del resultado. Además, el hecho de disponer de k estimaciones parciales permite hacer el cálculo de la varianza de las mismas, con lo que se puede conocer la variabilidad del método de aprendizaje con respecto al conjunto de datos particular.

Suponiendo que la medida de interés fuese la *accuracy* predictiva, el cálculo sería el siguiente:

$$accuracy\ predictiva = \frac{\sum_{i=1}^{k} \frac{\#correctasP_i}{\#instanciasP_i}}{k} \cdot 100 \tag{2.12}$$

donde:

- $\#correctasP_i$ representa el número de instancias etiquetadas correctamente por el clasificador de la iteración i, y
- $\#instanciasP_i$ indica el número de instancias existentes en la partición P_i

En la validación cruzada también se puede dar el caso de que interese generar las k particiones de modo que mantengan la misma distribución de instancias por clase que existía en el conjunto de datos original. En tal caso, el procedimiento recibe el nombre de **validación cruzada estratificada**. La Figura 2.4 muestra gráficamente el proceso para una validación cruzada estratificada con k particiones (*stratified k-fold cross validation*).

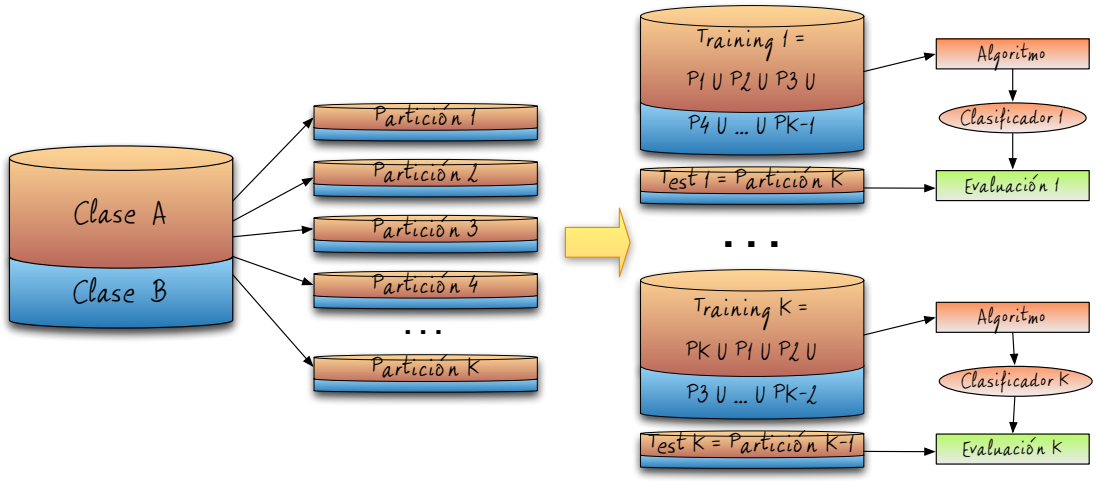


Figura 2.4: Validación cruzada estratificada con k particiones

Ocurre una situación particular de validación cruzada cuando el número de particiones es igual al número de ejemplos del conjunto de datos. En tal circunstancia, el conjunto de test estará compuesto por una única instancia, y el de entrenamiento, por el número de instancias menos una. Este caso se denomina **validación cruzada dejando uno fuera** (**leave-one-out cross-validation**).

2.1.3.3. Bootstrapping

Para validar el rendimiento de un clasificador sobre un conjunto de datos para el que se disponen de pocos ejemplos, sería más adecuado utilizar otro tipo de técnica como el *bootstrapping* [68]. En los casos anteriores las particiones generadas a partir del conjunto de datos original son mutuamente exclusivas. En cambio, en el *bootstrapping* se analizan repetidamente muestras aleatorias de los datos que pueden contener ejemplos repetidos. Suponiendo un conjunto de datos original de N instancias, se efectúa un muestreo aleatorio con reposición hasta que dicha

muestra contenga también N instancias, que compondrán el conjunto de *training*. El conjunto de test, por su parte, estará formado por aquellos ejemplos no elegidos en la muestra. Este proceso se repite un número determinado k de veces, actuando de manera análoga a la validación cruzada para calcular el promedio.

2.1.4. Clasificadores basados en reglas

Dado que la DM es un campo multidisciplinar, tal como se introdujo en el Capítulo 1, existen multitud de paradigmas para el diseño de algoritmos de DM. Es más, la tarea de descubrimiento de reglas ha sido abordada desde muchos de ellos, tales como los árboles de decisión, aprendizaje inductivo, aprendizaje basado en instancias y, más recientemente, algoritmos bioinspirados.

En concreto, en esta sección nos centramos en los árboles de decisión y los paradigmas de inducción de reglas, ya que consideramos la comprensibilidad como una cuestión fundamental en el proceso de toma de decisiones. Asimismo, se realiza una breve introducción a la evolución de reglas mediante algoritmos bioinspirados, dado que el Capítulo 3 aborda esta cuestión en detalle.

2.1.4.1. Inducción de reglas de clasificación

Se pueden extraer reglas de decisión **a partir de un árbol de decisión**, traduciéndolo en un conjunto de reglas de forma que se crea una regla por cada camino desde la raíz a una hoja del árbol [185]. Otra posibilidad es la extracción de reglas por medio de algoritmos de **cubrimiento secuencial** [98].

Los **árboles de decisión** son una serie de condiciones o decisiones organizadas jerárquicamente en forma de árbol, que clasifican instancias ordenándolas en base a los valores de sus atributos. Cada nodo codifica una característica, y cada rama que parte de un nodo representa uno de los posibles valores que la característica puede adoptar. El proceso de clasificación comienza en la raíz, que representará la característica más significativa. Dada una instancia, partiendo de la raíz se va

siguiendo la rama que la cubre hasta llegar a un nodo hoja, que indicará la clase a asignarle.

Existen diversos algoritmos para construir árboles de decisión, como el CART [35], el ID3 [184], o una extensión de este último, el algoritmo C4.5 [186], que es el más utilizado en la literatura. No obstante, todos ellos siguen una filosofía similar, basada en la homogeneidad de los ejemplos en cada rama y en buscar árboles del mínimo tamaño posible.

La figura 2.5 muestra a modo de ejemplo el árbol de decisión que extrae el algoritmo C4.5 sobre un conjunto de datos clásico de DM, el conjunto de datos *weather* (tiempo), donde el objetivo es determinar si se practica un determinado deporte al aire libre dependiendo de las condiciones climatológicas.

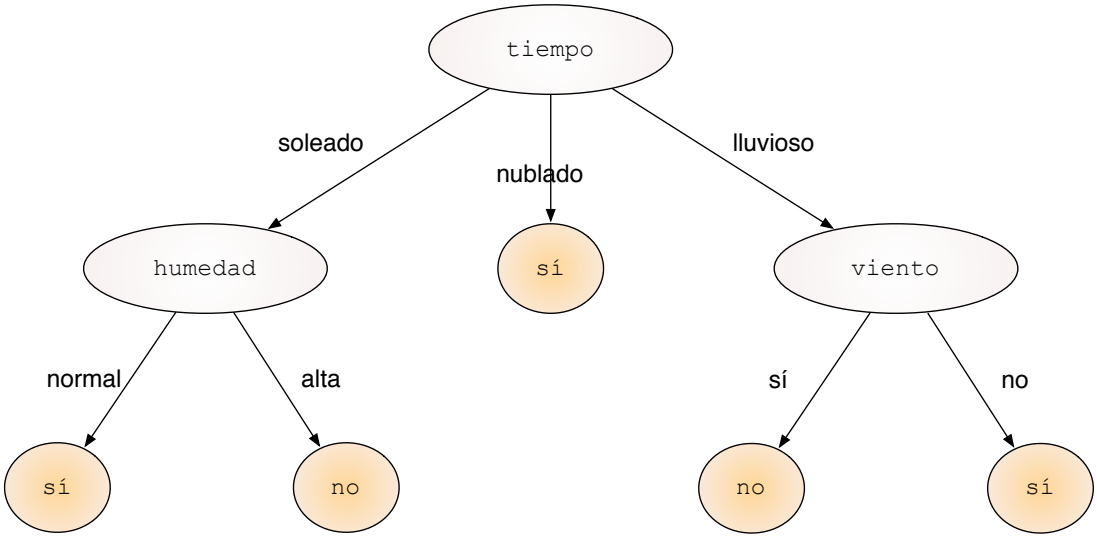


Figura 2.5: Ejemplo de árbol de decisión

La mayor ventaja de extraer reglas de clasificación a partir de un árbol de decisión radica en que son más fáciles de interpretar que un árbol completo. Las reglas resultantes son mutuamente excluyentes y exhaustivas, aunque pueden simplificarse mediante un proceso de generalización. No obstante, al hacerlo, dejarían de ser mutuamente excluyentes, pudiendo ser posible aplicar varias reglas para una instancia concreta, siendo necesario establecer un criterio para determinar cuál aplicar. Además, también dejarían de ser exhaustivas, con lo que también puede ocurrir que ninguna de las reglas cubra una instancia, siendo necesario incluir una

clase por defecto. Esto puede observarse en la lista de decisión de ejemplo para el conjunto de datos *weather* que se muestra en la Sección 2.1.4.2.

Por su parte, los algoritmos de cubrimiento secuencial (también denominados de aprendizaje iterativo), siguen el enfoque de *divide y vencerás*, aprendiendo una regla cada vez, secuencialmente. Van eliminando por cada regla extraída los ejemplos del conjunto de entrenamiento que cubre, y continúan aprendiendo nuevas reglas para cubrir las instancias que quedan. Idealmente, cada vez que se aprende una regla para la clase C_i, esta debe cubrir parte o todos los ejemplos de la clase C_i, y ninguno o el mínimo número posible que pertenezca a las demás clases. De esta manera las reglas aprendidas deben tener una exactitud predictiva alta. Una vez cubiertas todas las instancias, el clasificador se forma con las reglas creadas en cada iteración. Este enfoque genera nuevas reglas de un modo incremental, reduciendo al mismo tiempo el espacio de búsqueda puesto que en una determinada iteración no se tienen en cuenta las instancias que ya han sido eliminadas del conjunto de entrenamiento.

Un ejemplo ampliamente extendido de este tipo de algoritmos es el algoritmo **RIPPER** (*Repeated Incremental Pruning to Produce Error Reduction*) [51], que sigue un proceso repetido de crecimiento y poda. Durante la fase de crecimiento las reglas creadas son lo más restrictivas posibles, de manera que cubren un número muy específico de instancias del conjunto de entrenamiento. Luego, en la fase de poda, las reglas se modifican para que sean más generales y no se produzca sobreajuste (es decir, que las reglas sean tan específicas que se comporten mal sobre instancias no vistas durante el entrenamiento del modelo). El algoritmo RIPPER es capaz de tratar con problemas multiclase ordenando las clases desde mayor a menor prevalencia y tratando posteriormente cada una en orden como un problema binario diferente.

Un algoritmo que combina los árboles de decisión y el enfoque de cubrimiento secuencial para la extracción de reglas de decisión es el **PART** [230]. Este algoritmo infiere reglas de decisión generando repetidamente árboles de decisión parciales (uno en cada iteración). Una vez que un árbol parcial ha sido construido, se extrae una única regla del mismo, correspondiente al mejor nodo hoja.

2.1.4.2. Construcción del clasificador

El clasificador final resultante de ejecutar un determinado algoritmo de inducción de reglas estará compuesto por un conjunto de las mejores reglas aprendidas. Estas reglas pueden estar **ordenadas**, formando una lista de decisión que indica el modo en que tratan de cubrir y clasificar un nuevo ejemplo; o **desordenadas**, cuando el orden relativo entre las reglas no importa y debe instaurarse una política de asignación que será la encargada de indicar qué regla clasificará una determinada instancia en caso de conflicto (cuando dos o más reglas que predicen una clase diferente cubren la instancia).

En una **lista de decisión** las reglas se interpretan en orden. Se comienza comprobando si la primera regla cubre el nuevo ejemplo. Si es así, se le asigna la clase que predice la regla; si no, se trata de aplicar la segunda regla, y así sucesivamente. En caso de alcanzar el final de la lista de decisión sin que alguna regla se haya disparado, se etiqueta al ejemplo con la clase predicha por la regla por defecto (normalmente la clase mayoritaria del conjunto de *training*). La ventaja de las listas de decisión es que no existe ningún tipo de ambigüedad a la hora de etiquetar nuevas instancias. A continuación se muestra un ejemplo de clasificador en forma de lista de decisión para el conjunto de datos *weather*:

`IF` *tiempo* = *soleado* `AND` *humedad* = *alta* `THEN` *jugar* = *no*

`ELSE IF` *tiempo* = *lluvioso* `AND` *viento* = *sí* `THEN` *jugar* = *no*

`ELSE IF` *tiempo* = *nublado* `THEN` *jugar* = *sí*

`ELSE IF` *humedad* = *normal* `THEN` *jugar* = *sí*

`ELSE` *jugar* = *sí*

En el caso de que el clasificador esté formado por un **conjunto de reglas no ordenadas**, podemos considerarlas como pequeñas fracciones de conocimiento. No obstante, nos encontramos con la desventaja de que, si se produce una situación de conflicto, no está claro cuál de ellas aplicar. Además, puede darse el caso opuesto, es decir, que un nuevo ejemplo no sea cubierto por ninguna de ellas. Este último caso

se podría solventar asignando como categoría la clase mayoritaria del conjunto de *training*. En cambio, cuando ocurren conflictos, la clasificación de una nueva instancia se decide tras un proceso de votación que sigue los siguientes pasos:

- Se identifican las reglas que cubren la nueva instancia.
- Las reglas identificadas se ordenan de acuerdo a algún criterio. Por ejemplo, se pueden ordenar de mayor a menor por el número de ejemplos de *training* cubiertos, exactitud, AUC, o cualquier otra medida de rendimiento; o bien ordenarlas de menor a mayor complejidad de la regla (número de condiciones del antecedente). Se pueden encontrar otros enfoques de ordenación y una discusión de los mismos en *Wang et al.* [225].
- Se seleccionan las mejores k reglas ordenadas para votar la clase que se asignará a la instancia.
- Se cuentan los votos obtenidos para cada clase, y la nueva instancia queda etiquetada con aquella que obtiene más votos.

Hasta este punto se han descrito los dos tipos de clasificadores basados en reglas que pueden resultar de la ejecución de un algoritmo. Sin embargo, ambos pueden aparecer combinados junto con otro tipo de clasificadores, ya sean basados en reglas o no. La idea básica de los métodos de combinación de clasificadores, denominados ***ensembles*** [190], consiste en combinar las decisiones de clasificadores individuales, que en este contexto se denominan clasificadores base, con la finalidad de que la decisión final de a qué clase pertenece el ejemplo se tome como agregación del resultado de todos ellos. Así, cuando haya que clasificar un nuevo ejemplo, cada uno de los clasificadores base generará su predicción, y será necesario integrar posteriormente los resultados, siguiendo alguna política de asignación para determinar con qué clase se etiqueta la instancia. Normalmente se sigue una estrategia de votación similar al caso de los clasificadores compuestos por reglas desordenadas, de manera que la clase más votada por el conjunto de clasificadores es la que se emplea para categorizar la nueva instancia [86]. Hay que resaltar que los clasificadores base deben presentar cierta variabilidad en cuanto al algoritmo o método empleado para su inducción, los datos de entrenamiento utilizados, o bien en los parámetros de aprendizaje empleados en su configuración.

Los *ensembles* se suelen utilizar para mejorar el rendimiento de un clasificador, aunque su uso se hace indispensable cuando nos encontramos ante un conjunto de datos multiclase y queremos utilizar un algoritmo que únicamente es capaz de abordar problemas de clasificación binaria [86].

2.1.4.3. Evolución de clasificadores basados en reglas

Tal como se explica posteriormente en el Capítulo 3, la utilización de algoritmos de clasificación bioinspirados para la extracción de clasificadores basados en reglas presentan ciertas ventajas frente a los enfoques tradicionales [84]:

- Son métodos de búsqueda robustos y adaptativos que realizan una búsqueda global en el espacio de soluciones, en contraste con los métodos de búsqueda de tipo voraz o *greedy*, donde las reglas se construyen considerando un atributo cada vez, seleccionando el mejor a incluir [191].
- En el contexto de la DM, la búsqueda global de los algoritmos bioinspirados se asocia a una mejor forma de abordar y tratar con las interacciones entre atributos. Al igual que los algoritmos tradicionales de inducción de reglas, los algoritmos bioinspirados pueden representar reglas *IF-THEN*, pero su búsqueda global permite tratar mejor la interacción entre atributos y descubrir relaciones que podrían perderse en una búsqueda voraz.
- Evolucionan una población de individuos, donde cada individuo representa una solución al problema. Por tanto, trabajan con un conjunto de reglas, en vez de con una sola.
- Emplean operadores estocásticos, mientras que los enfoques tradicionales son deterministas.
- La búsqueda es conducida por una competición entre individuos, donde se utiliza una función de *fitness* o aptitud para evaluar la bondad de cada uno de ellos para resolver el problema, de forma independiente a cómo se ha construido la solución. En DM, esto proporciona cierta libertad y flexibilidad que permite incorporar conocimiento acerca del dominio del problema en el algoritmo bioinspirado, así como hibridar estos algoritmos con métodos de búsqueda locales especialmente diseñados para la tarea de DM.

Por otro lado, los algoritmos bioinspirados presentan una serie de inconvenientes relacionados principalmente con el tiempo de ejecución sobre conjuntos de datos de gran dimensionalidad, que normalmente suele ser superior al de los algoritmos tradicionales de inducción de reglas. No obstante, es posible paralelizar estos algoritmos, y muchos de los esfuerzos de la comunidad científica se centran en este aspecto. En particular, para la tarea de clasificación de DM, la generación de datos en cualquier dominio de aplicación se ve incrementada diariamente a un ritmo exponencial. Por este motivo, se considera un punto crucial el diseño de algoritmos paralelos capaces de lidiar con esta enorme cantidad de datos.

Tal como se ha comentado, los algoritmos bioinspirados basados en reglas evolucionan una población de individuos para extraer un clasificador. Uno de los aspectos cruciales en el diseño de este tipo de algoritmos consiste en la elección del **esquema de codificación** de las reglas dentro de la población de individuos.

En la literatura se presentan dos enfoques principales para la codificación de los individuos [70]:

- Enfoque ***individuo = regla*** (también denominado enfoque de *Michigan*) [69], en el que cada individuo de la población codifica una única regla de decisión y el clasificador se obtiene al combinar varios individuos de la población. Para ello será necesario emplear algún mecanismo que considere un subconjunto de la población a partir de la cual se forme el clasificador.
- Enfoque ***individuo = conjunto de reglas*** (también denominado enfoque de *Pittsburgh*) [69], en el que cada individuo codifica un clasificador completo con diversas reglas. El clasificador final será el mejor individuo evolucionado durante la ejecución del algoritmo.

2.2. Minería de reglas de asociación

Dado un conjunto de datos con N ejemplos o instancias $D = \{d_1, d_2, ..., d_N\}$, cada una descrita por una serie de observaciones o patrones $X = \{x_1, x_2, ..., x_z\}$, la tarea de asociación consiste en la extracción de relaciones frecuentes y confiables entre patrones, de modo que proporcionen conocimiento de interés para el usuario. Estas relaciones se pueden trasladar a reglas del tipo `IF` A `THEN` C, representado como $A \rightarrow C$, del conjunto de datos D, con las restricciones de que $A,\ C \subseteq X$ y $A \cap C = \emptyset$. Tanto el antecedente, A, como el consecuente, C, se interpretan como un conjunto de uno o más patrones: $A = \{x_1 = a \wedge ... \wedge x_i = d \wedge ... \wedge x_z = w\}$.

Por tanto, las reglas de asociación expresan relaciones que se dan en un conjunto de datos de acuerdo a la aparición conjunta de valores de dos o más atributos con una determinada frecuencia. Atributos que en el contexto de la ARM se conocen con el nombre de *items*. Es decir, se trata de proposiciones probabilísticas sobre la ocurrencia de ciertos valores de los *items* en un conjunto de datos [39]. Un conjunto de uno o más *items* se denomina *itemset*, y un *k-itemset* será un *itemset* compuesto por k *items*.

En ARM, a diferencia de lo que ocurría en la tarea de clasificación, la parte derecha de la regla puede estar compuesta por uno o más atributos, sin necesidad de especificar un atributo de clase. Es más, en el consecuente pueden aparecer varios atributos, no sólo uno. De hecho, un tipo específico de reglas de asociación son las reglas de asociación de clase (*class association rules*), donde, siempre siguiendo con la filosofía de ARM y empleando los algoritmos específicos, se introduce la restricción de que en el consecuente de la regla únicamente pueda aparecer un atributo, el correspondiente a la clase de un conjunto de datos etiquetado [46, 192, 219].

Por lo general, el proceso de ARM se descompone en dos subprocesos, la **extracción de patrones o items frecuentes**, es decir, aquellos que aparecen por encima de una probabilidad, y la posterior **obtención de reglas de asociación a partir de los patrones** previamente obtenidos. El primer subproceso implica una carga

computacional elevada, puesto que conlleva la exploración de muchas combinaciones de *items*, lo que supone una tarea compleja, especialmente cuando se trata de conjuntos de datos de alta dimensionalidad. Dado que la mayoría de las bases de datos que manejan las compañías son enormes, existe una necesidad acuciante de encontrar algoritmos eficientes que puedan extraer conjuntos de *items* frecuentes en un tiempo de cómputo razonable. Es por este motivo que a veces se llega a un compromiso entre el tiempo de cómputo y la búsqueda de todos los conjuntos de *items* frecuentes, considerando generalmente aquellos que sobrepasan un determinado umbral de soporte. El soporte y la confianza son las dos medidas de calidad más importantes para evaluar el interés de las reglas de asociación.

El **soporte** implica la frecuencia de aparición de patrones. Así, el soporte de un patrón x_i será igual al número de transacciones T del conjunto de datos D que lo contienen, denotado por $|x_i \subseteq T, T \in D|$. Por su parte, el soporte de una regla de asociación del tipo $A \rightarrow C$ indica la proporción de transacciones T del conjunto de datos D que incluyen tanto el antecedente como el consecuente:

$$support(A \rightarrow C) = \frac{|\{A \cup C \subseteq T, T \in D\}|}{|D|} \tag{2.13}$$

Además, una propiedad de esta medida establece que si un *itemset* no es frecuente, entonces la adición de otro *item* a este *itemset* no lo hará más frecuente, como se demuestra en la Ecuación 2.14. Este principio se conoce con el nombre de **propiedad antimonótona**:

$$\begin{aligned} 0 \leq support(Condition_1) \leq 1 \\ 0 \leq support(Condition_2) \leq 1 \\ 0 \leq support(Condition_1 \cup Condition_2) \leq 1 \\ support(Condition_1 \cup Condition_2) \leq support(Condition_1) \\ support(Condition_1 \cup Condition_2) \leq support(Condition_2) \end{aligned} \tag{2.14}$$

Es importante destacar que el interés de extraer reglas de asociación sobre patrones frecuentes radica en la extracción de reglas con un alto soporte, es decir, reglas que satisfacen un gran número de instancias de un conjunto de datos. Sin embargo, si

una regla de asociación tiene el máximo valor de soporte posible y, por tanto, cubre todas las instancias del conjunto de datos, no proporciona nueva información.

La **confianza**, en cambio, es una medida de la fuerza de la implicación de una regla de asociación. Para una regla $A \rightarrow C$, la confianza representa la proporción del número de transacciones que incluyen tanto el antecedente como el consecuente entre aquellas transacciones que sólo contienen el antecedente. En otras palabras, es el soporte de la regla dividido por el soporte del antecedente.

$$confidence(A \rightarrow C) = \frac{support(A \rightarrow C)}{support(A)} = \frac{|\{A \cup C \subseteq T, T \in D\}|}{|\{A \subseteq T, T \in D\}|} \tag{2.15}$$

2.2.1. Taxonomía

A continuación comentaremos los distintos tipos de reglas de asociación que encontramos en la literatura:

- Dependiendo de que nos interese extraer reglas de asociación con un alto soporte o no:

  Frecuentes [5], con soporte y confianza altos.

  Infrecuentes o **raras** [145], que son aquellas cuyo soporte es bajo pero su confianza alta. Son especialmente interesantes cuando existen situaciones en las que el objetivo es descubrir comportamientos anómalos o inusuales, buscando por tanto encontrar aquellos patrones que no siguen la tendencia de la mayoría.

- En ocasiones suele ocurrir que los *items* pertenezcan a una determinada jerarquía. Así, según los niveles de abstracción podemos encontrar:

  De **un único nivel**, son aquellas asociaciones entre atributos que están en el mismo nivel de abstracción. Ejemplo: $(Cerveza, Aceitunas) \rightarrow (Pan)$.

  Multinivel, donde hay envueltos atributos pertenecientes a distintos niveles de la jerarquía. Supongamos que consideramos los atributos $Negra$ como subtipo del atributo *Cerveza*, *Rellenas* como suptipo de *Patatas* y *Anchoa* como subtipo de *Rellenas*. Una relación multinivel sería, por ejemplo, la siguiente: (*Cerveza:Negra,Aceitunas:Rellenas:Anchoa*)$\rightarrow$ (Pan). La razón de

utilizar jerarquías puede venir motivada por el hecho de que reglas que involucren atributos más bajos de la jerarquía puede que no tengan soporte suficiente para aparecer con una frecuencia mínima, dado que se trata de reglas muy específicas.

- **Reglas de asociación restringidas** (*constraint-based ARM*). Con el objetivo de incrementar la eficiencia de los algoritmos de asociación, se pueden considerar restricciones durante el proceso de extracción de reglas de manera que se generen únicamente las reglas interesantes para el usuario, en lugar de todas las posibles. Podemos contemplar restricciones referentes a umbrales mínimos de algunas medidas, como el soporte o la confianza, o bien aquellas relativas a los *items* en los que está interesado el usuario o a la disposición de los mismos. Así, por ejemplo, si estamos ante un conjunto de datos etiquetado y se introduce la restricción de que en el consecuente de la regla sólo puede aparecer el atributo correspondiente a la clase, nos encontramos ante las **reglas de asociación de clase** [46, 192, 219].

2.2.2. Medidas de rendimiento

En esta sección se introducen las medidas de calidad de reglas de asociación que aparecen más frecuentemente en la literatura.

Al igual que en la tarea de clasificación se podía emplear una matriz de confusión para recoger los ejemplos clasificados correcta e incorrectamente por cada clase, en ARM se puede emplear una **tabla de contingencia** para contabilizar las frecuencias con que una determinada regla de asociación se satisface en un conjunto de datos, tal como recoge la Tabla 2.3.

	C	$\neg C$	
A	$n(AC)$	n$(A\neg C)$	$n(A)$
$\neg A$	$n(\neg AC$	$n(\neg A\neg C)$	$n(\neg A)$
	$n(C)$	$n(\neg C)$	N

Tabla 2.3: Tabla de contingencia para una regla de asociación $A \rightarrow C$

donde:

$n(AC)$ se refiere al número de instancias que satisfacen tanto A como C, y

N es el número total de instancias

Partiendo de la tabla de contingencia también se pueden definir el soporte y la confianza, además de otras medidas [92, 137] que se emplean en distintas áreas como la estadística, teoría de la información y recuperación de información. A continuación definiremos algunas de ellas, utilizando también medidas de probabilidad, teniendo en cuenta que $P(A) = \frac{n(A)}{N}$ denota la probabilidad de que ocurra A, y que $P(C|A) = \frac{P(A \cap C)}{P(A)}$ representa la probabilidad de C dada A (probabilidad condicionada de que ocurra C sabiendo que ocurre A):

- **Soporte** (*support*): el soporte de un conjunto de *items* A se define como el porcentaje de instancias que contienen A o, lo que es lo mismo, la probabilidad de A.

$$support(A) = P(A) \tag{2.16}$$

  El soporte de una regla $A \rightarrow C$ es el porcentaje de instancias que contienen A y C simultáneamente.

$$support(A \rightarrow C) = P(A \cap C) \tag{2.17}$$

- **Confianza** (*confidence*): es la probabilidad de que las transacciones que contienen A también contengan C.

$$confidence(A \rightarrow C) = P(C|A) \tag{2.18}$$

- **Cobertura** (*coverage*): mide con qué frecuencia se puede aplicar la regla en el conjunto de datos. Es equivalente al soporte del antecedente de la regla.

$$coverage(A \rightarrow C) = support(A) = P(A) \tag{2.19}$$

- **Interés** (*lift*) [36]: mide cuál es el grado de independencia de A y C. Su rango de valores está comprendido en el intervalo $[0, \infty)$. Los valores cercanos a 1 indicarán que tanto A como C son independientes y que, por tanto, la regla no es interesante, mientras que los valores alejados del 1 querrán decir que la evidencia de A proporciona información sobre C. La unión de *itemsets* poco frecuentes puede producir valores muy altos de esta medida. Además, se trata de una medida simétrica, ya que $lift(A \rightarrow C) = lift(C \rightarrow A)$.

$$lift(A \rightarrow C) = \frac{confidence(A \rightarrow C)}{support(C)} = \frac{P(C|A)}{P(C)} = \frac{P(A \cap C)}{P(A) \cdot P(C)} \tag{2.20}$$

- **Convicción** (*conviction*) [36]: trata de determinar el grado de implicación de una regla y, a diferencia del $lift$, sí es sensible a la dirección de la regla. Se calcula como la probabilidad de que A aparezca sin C en caso de que sean estadísticamente independientes con respecto a la probabilidad condicionada de A sin C. El valor de la convicción será infinito si existe una implicación absoluta, en cuyo caso la confianza será 1. Asimismo, adoptará el valor 1 cuando A y C sean estadísticamente independientes.

$$conviction(A \rightarrow C) = \frac{1 - support(C)}{1 - confidence(A \rightarrow C)} = \frac{P(A) \cdot P(\neg C)}{P(A \cap \neg C)} \tag{2.21}$$

- **Novedad** (*leverage*) [180]: mide la proporción de casos adicionales cubiertos por A y C sobre aquellos esperados si A y C fueran estadísticamente independientes.

$$\begin{aligned} leverage(A \rightarrow C) = \; & support(A \cap C) - support(A) \cdot support(C) \\ & = P(C|A) - P(A) \cdot P(C) \end{aligned} \tag{2.22}$$

2.2.3. Enfoques clásicos de búsqueda exhaustiva

El primer algoritmo que se propuso para la extracción de reglas de asociación fue el **Apriori** [5], que se compone de dos pasos:

1. Dado un conjunto de datos con k atributos o *items*, genera todos los conjuntos de *items* con un elemento que sobrepasan un umbral mínimo de soporte. Esto permite ir eliminando posibles combinaciones, al no ser necesario considerarlas todas. A continuación, usa los *items* extraídos para *itemsets* con dos elementos, volviendo a tomar sólo los que cumplen con el soporte mínimo, y así sucesivamente hasta llegar a los conjuntos de *items* de tamaño $k-1$.

2. Se generan las reglas de asociación a partir de los *items* frecuentes extraídos en el paso anterior. El nivel de soporte mínimo está ya garantizado al emplearse los conjuntos de *items* frecuentes, de manera que el proceso se limita a generar todas las reglas de asociación posibles, descartando posteriormente aquellas que no satisfacen el umbral de confianza mínimo.

Otro enfoque importante basado en el Apriori es el **FP-Growth** [99], que se basa en una estructura de árbol frecuente que evita la generación de grandes conjuntos de *items* candidatos, minimizando el tiempo de cómputo del primero. De hecho, este algoritmo elimina los cuellos de botella del Apriori, generando los conjuntos de *items* frecuentes en tan sólo dos pasadas sobre el conjunto de datos, y sin requerir de una fase de generación de candidatos. El proceso de generación de patrones frecuentes incluye dos subprocesos: la construcción de la estructura de árbol y la generación de patrones frecuentes a partir del mismo.

Ambos algoritmos entran dentro de la categoría de enfoques de búsqueda exhaustiva [221]. Este tipo de algoritmos siguen dos pasos. El primero consiste en la identificación de los conjuntos de *items* frecuentes, es decir, aquellos que sobrepasan un umbral mínimo de frecuencia ϕ. Entonces, combinando estos conjuntos de *items* frecuentes, extraen aquellas reglas de asociación que satisfacen un umbral mínimo de soporte y de confianza, establecidos por el usuario o el experto.

Una desventaja de este tipo de métodos estriba en que no son apropiados para tratar con conjuntos de datos de gran dimensionalidad debido a sus requerimientos de memoria y tiempo de cómputo. Además, únicamente son capaces de trabajar

con datos categóricos, por lo que es necesario discretizar los datos numéricos. Por otro lado también tienen el problema de que con umbrales de soporte bajos el número de items y, por tanto, de reglas, crece considerablemente, devolviendo un conjunto de reglas difícilmente comprensible para el usuario final. Por todos estos motivos, el interés en la aplicación de otro tipo de metodologías a la ARM se ha visto acrecentado. Así, se han aplicado EAs [144, 148, 236] o algoritmos de SI [130], que obtienen buenas soluciones cuando el espacio de búsqueda es demasiado grande para utilizar métodos de búsqueda determinísticos.

3

Programación automática con colonias de hormigas. Investigación del estado del arte

Los **algoritmos bioinspirados** [79] son una clase de algoritmos basados en sistemas biológicos y que simulan las propiedades de dichos sistemas en la naturaleza. Son atractivos desde el punto de vista computacional al tener muchos dominios de aplicación y proporcionar buenos resultados a un coste aceptable. De hecho, se trata de uno de los campos de investigación más activos y prometedores en el diseño de algoritmos, e incluyen áreas como las **redes neuronales artificiales** (ANNs, *Artificial Neural Networks*), los **algoritmos evolutivos** (EAs, *Evolutionary Algorithms*), los **sistemas inmunes** (AISs, *Artificial Immune Systems*), y la **inteligencia colectiva** (SI, *Swarm Intelligence*), entre otros.

Este capítulo se centra en explicar y realizar una revisión bibliográfica de la **programación automática con hormigas** (AP, *Ant Programming*), paradigma bioinspirado en el que se basan los algoritmos desarrollados en esta tesis. Para ello se efectúa en primer lugar una introducción a los algoritmos bioinspirados, marco donde se encuadra el paradigma de AP. Acto seguido se explica una clase de EA,

la programación genética (GP, *Genetic Programming*), al tratarse de otro tipo de programación automática que además comparte ciertas características comunes con la AP. Por otro lado, se introduce la metaheurística de optimización mediante colonias de hormigas (ACO, *Ant Colony Optimization*), ya que se trata de la técnica de búsqueda empleada por la AP, examinando asimismo los algoritmos de ACO propuestos hasta la fecha para las tareas de clasificación y asociación de DM.

3.1. Algoritmos bioinspirados

Los algoritmos bioinspirados emulan, de manera aproximada, el comportamiento de un sistema natural para el diseño de métodos heurísticos para la resolución de problemas. Las características que presentan son las siguientes [81]:

- Son **estocásticos** o **no determinísticos**, esto es, tienen un componente aleatorio en la toma de decisiones.
- A menudo presentan, de forma implícita, una **estructura paralela** (múltiples agentes).
- Son **adaptativos**, utilizando retroalimentación con el entorno para modificar el modelo y los parámetros.

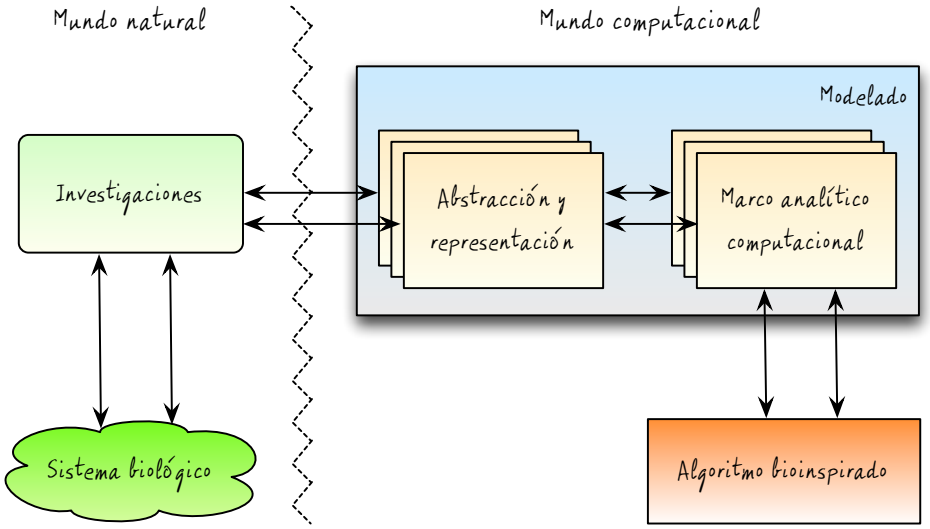


Figura 3.1: Conexión entre los algoritmos bioinspirados y la biología (adaptado de *Stepney et al.* [209])

El paso de un modelo biológico a un algoritmo bioinspirado implica, al menos, que concurran las áreas de biología, matemáticas y ciencias de la computación. *Stepney et al.* [209] propusieron un marco conceptual para el desarrollo de modelos computacionales bioinspirados sofisticados, exponiendo la relación existente entre la biología y los algoritmos bioinspirados. Una aproximación a dicho proceso se muestra en la Figura 3.1, donde la parte izquierda se refiere a la investigación en el campo de la biología, mientras que la derecha muestra el desarrollo de algoritmos bioinspirados. A modo de resumen, las observaciones, experimentos e investigaciones proporcionan una vista del sistema biológico. A partir de esta vista se realizan representaciones abstractas o modelos de la misma, validados por medio de análisis matemáticos, problemas de referencia y demostraciones. Los modelos sirven para construir un marco analítico computacional, que a su vez proporciona los principios necesarios para desarrollar algoritmos bioinspirados. A continuación se introducen brevemente algunos paradigmas de computación bioinspirados.

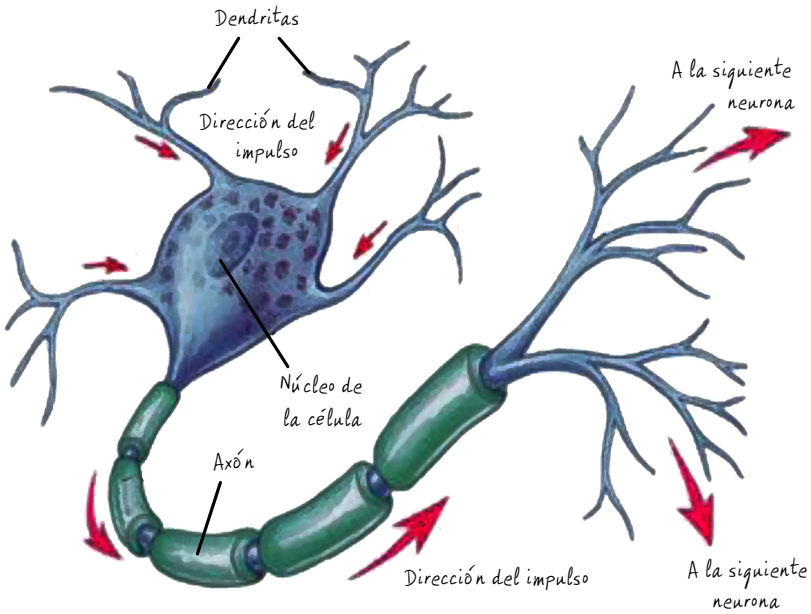


Figura 3.2: Diagrama de una neurona real

Las ANNs [243] son un paradigma de computación matemática que se fundamenta en la emulación de los sistemas nerviosos biológicos, cuya capacidad de cómputo se desarrolla mediante un proceso adaptativo de aprendizaje. En la Figura 3.2 se

muestra cómo las neuronas reales se comunican entre sí, mediante transmisiones electroquímicas que tienen lugar en la sinapsis, es decir, los contactos entre el axón de una neurona con las dendritas de otra. La información pasa entonces al cuerpo de la célula, donde tiene lugar el sumatorio de los impulsos eléctricos que lo alcanzan, aplicándose una función de activación. De esta forma, la neurona se activará si el resultado es superior a un determinado umbral, enviando una señal por su axón para comunicarse con otras neuronas.

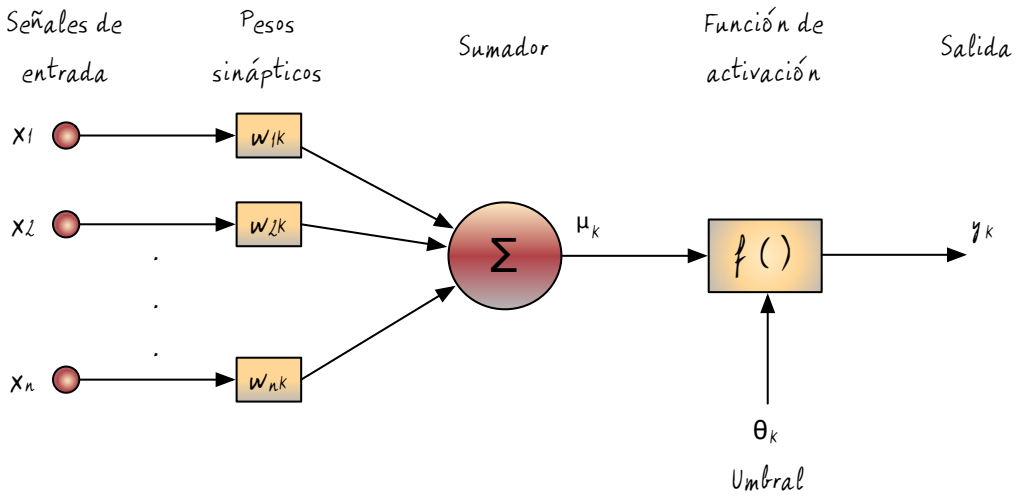


Figura 3.3: Diagrama de una neurona artificial

El modelado de una neurona biológica se recoge en la Figura 3.3. Se puede apreciar que incluye una entrada adicional, denominada polarización y denotada por Θ, la cual permite aumentar o disminuir el umbral de excitación de la neurona. Las entradas se representan por el vector de entrada, $\boldsymbol{x}$, y el rendimiento de las sipnasis se modela mediante un vector de pesos, $\boldsymbol{w}$. De este modo, el valor de salida de la neurona se corresponde con la siguiente función de activación:

$$y = f\left(\sum_i w_i \cdot x_i\right) \tag{3.1}$$

Una única neurona es una unidad de procesamiento muy simple. Su potencial radica en el empleo de muchas de estas unidades simples y robustas actuando en paralelo. En una red neuronal, las salidas de unas neuronas se conectan con las entradas de otras. Si el peso entre dos neuronas conectadas es positivo, el efecto producido es el

de excitación. En cambio, si es negativo, el efecto es de inhibición. El conocimiento se adquiere por la red mediante un proceso de aprendizaje o entrenamiento, el cuál es un procedimiento de adaptación de los pesos mediante un algoritmo.

Por su parte, los EAs [69, 121] pueden producir procesos altamente optimizados simulando la evolución basada en poblaciones, siguiendo los principios de la selección natural.

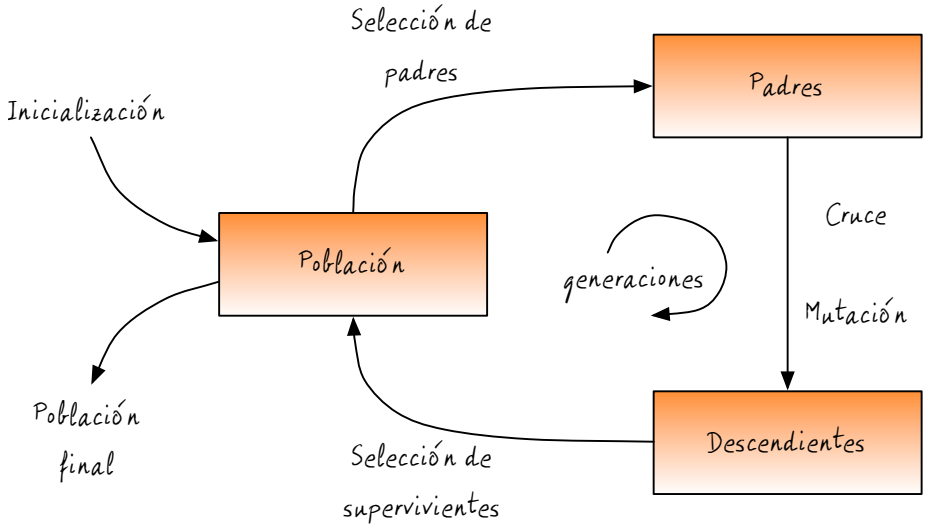


Figura 3.4: Esquema general de los EAs (adaptado de *Eiben y Smith* [69])

Los EAs utilizan una estrategia de aprendizaje colaborativo sobre una población de individuos. El **esquema general de los EAs** se observa en la Figura 3.4. A grandes rasgos, se inicializa una población de individuos al azar y, dada la función de aptitud a optimizar, que permite evaluar la adaptación de cada individuo al entorno, un mecanismo de selección se encarga de escoger los mejores candidatos como padres para dar lugar a la siguiente generación de individuos. Esto se realiza por medio de las operaciones de cruce y mutación, operaciones pseudoaleatorias encargadas de introducir diversidad en la población. El cruce es un operador que se aplica sobre dos o más padres y genera uno o más hijos, intercambiando la información genética de los progenitores; en cambio, la mutación es un proceso de autorreplicación errónea que se aplica sobre un único individuo y que genera, por tanto, un nuevo individuo, que difiere en pequeños aspectos del original. Una vez obtenida la descendencia, se produce la selección de supervivientes o remplazo, donde los descendientes producidos por ambas operaciones compiten junto con los

individuos de la generación anterior para formar parte de la nueva generación. Si se conoce el valor de una solución óptima para el problema, el proceso se repite hasta que se encuentra un individuo con una calidad aceptable, en cuyo caso se tomaría como solución. En caso contrario, el algoritmo sigue ejecutándose hasta que se alcanza una condición de parada, que puede ser un límite establecido de iteraciones, de evaluaciones, de tiempo, o bien la pérdida de diversidad tras el paso de generaciones.

Los EAs engloban cuatro paradigmas fundamentales que difieren principalmente en la representación de los individuos y en el tipo de operadores que emplean:

- Los **algoritmos genéticos** (GAs, *Genetic Algorithms*) [108], donde los individuos se codifican como cadenas llamadas cromosomas (genotipo), y representan una solución (fenotipo). Los cromosomas pueden presentarse en forma de cadenas binarias o bien utilizar estructuras más intrincadas, como cadenas de números reales o permutaciones, entre otras. Los GAs enfatizan el rol del operador de cruce como elemento fundamental de búsqueda, dotando de menor importancia a la mutación, que se aplica con menos probabilidad.

- La **programación genética** (GP, *Genetic Programming*) [128], que es una técnica de programación automática donde los individuos son programas de ordenador. Está basada en los GAs, pero a diferencia de estos, los individuos se representan con estructuras más elaboradas (normalmente, árboles). Los individuos no sólo contienen datos, sino que también contienen expresiones o funciones que se aplican sobre los datos.

- La **programación evolutiva** [80], una abstracción de la evolución al nivel de las especies. La principal diferencia entre ella y el resto de enfoques de EA es que no se produce intercambio de material genético entre los individuos de la población, utilizando únicamente el operador de mutación.

- Las **estrategias evolutivas** [201], centradas en simular las modificaciones en el comportamiento al nivel de los individuos. Suelen emplear vectores reales para codificar los individuos y utilizan la mutación como operador principal a la hora de generar la descendencia. Además, los parámetros de este operador cambian durante la ejecución del algoritmo.

Otro de los tipos de algoritmos bioinspirados mencionados previamente, los AIS [244], se fundamentan en la simulación del comportamiento del sistema inmunológico, sistemas adaptativos complejos compuestos de células y moléculas cuyo propósito es proteger al huésped y eliminar patógenos invasores como bacterias y virus. A tal efecto, hacen uso de la memoria y el aprendizaje para resolver tareas de reconocimiento y clasificación. De hecho, aprenden a reconocer patrones relevantes, recordar patrones vistos anteriormente y construir detectores de patrones eficientes. Éstas habilidades de procesamiento de información de los sistemas inmunes proporcionan aspectos deseables en el campo de la computación, resultando en sistemas escalables, robustos, muy flexibles y con una degradación elegante.

La SI [26, 29], comprende un grupo de algoritmos que se inspiran en el comportamiento descentralizado y colectivo de sistemas multiagente tales como colonias de insectos sociales u otras sociedades de animales.

Los objetivos generales del grupo, colonia o enjambre, como la búsqueda de alimento, se persiguen a través de acciones independientes de los individuos basadas en métodos de comunicación indirectos. A partir de estas acciones independientes emerge un comportamiento global, inteligente y coordinado. Como ejemplos más representativos de sistemas de SI podemos citar, entre otros, la **optimización mediante colonias de hormigas** (ACO, *Ant Colony Optimization*) [154], que simula el comportamiento de las hormigas en la naturaleza, la **inteligencia de enjambres de abejas** (*bee swarm intelligence*) [118], que imitan muchas de las características que presentan las colmenas de abejas en la naturaleza, la **optimización mediante bacterias** *bacterial foraging optimization* [53], que se fundamenta en el comportamiento de las bacterias en su búsqueda de áreas con altos contenidos en nutrientes, o la **optimización mediante nubes de partículas** (PSO, *Particle Swarm Optimization*) [223], basada en el movimiento de los bancos de peces o las bandadas de aves, como se observa en la Figura 3.5.

Millonas [151] determinó los cinco principios que debe satisfacer un enjambre para ser considerado inteligente:

1. Proximidad. El grupo debe ser capaz de realizar cálculos espaciales y temporales.

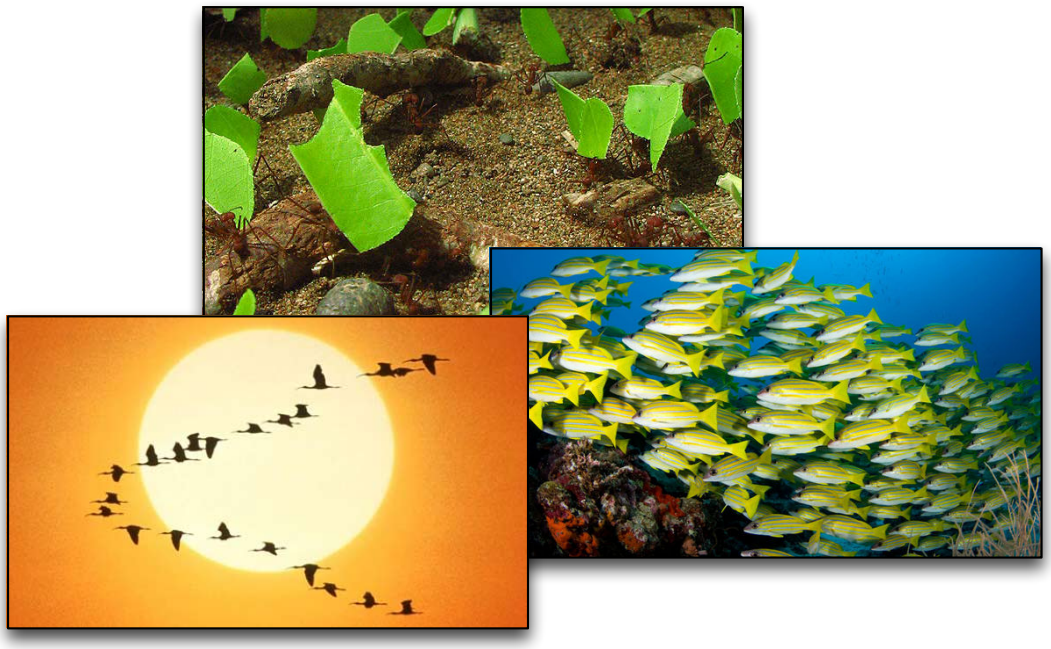


Figura 3.5: Ejemplo de colonias de hormigas, bandadas de pájaros y bancos de peces en la naturaleza

2. Calidad. La población debe ser capaz de responder a factores de calidad como la importancia de la fuente de alimento o la seguridad de la localización que ocupa.

3. Respuesta diversa. El grupo no debe colocar todos sus recursos a lo largo de canales de comunicación estrechos, sino que debe distribuirlos en muchos nodos, a modo de seguro ante una fluctuación repentina del entorno.

4. Estabilidad. El grupo no debe variar su comportamiento cada vez que el entorno cambia, ya que puede no compensar el gasto energético que acarrea.

5. Adaptabilidad. La población debe ser adaptativa, esto es, debe ser capaz de modificar su comportamiento cuando haya alguna señal que así lo recomiende desde el punto de vista de ahorro computacional o de mejora en la precisión.

El concepto básico de cualquier algoritmo de SI es el empleo de un alto número de individuos simples, de poco peso. El objetivo global de resolver un determinado problema nunca va a depender de los individuos de manera singular, sino que, de

hecho, dichos individuos son prescindibles, lo que garantiza la robustez del proceso. En general, cada individuo sólo tiene una vista limitada de sus vecinos, y tiene una memoria limitada, lo que le lleva a tomar decisiones basándose únicamente en la información de que dispone localmente mediante la ayuda de reglas simples. El proceso de decisión para un individuo concreto se puede describir en unas pocas reglas de comportamiento simples. Todo ello hace a los sistemas de SI **escalables** con respecto al número de individuos, **robustos** respecto a la pérdida de miembros del grupo, y **adaptativos** a entornos sometidos a un cambio continuo. Un individuo no tiene conocimiento suficiente para resolver la tarea a la que se enfrenta el grupo, pero sí que tiene capacidad para comunicarse indirectamente con el resto de individuos produciendo cambios en el entorno. Así pues, se permite que surja una solución global como respuesta a todas los estímulos y acciones simples que los individuos llevan a cabo, independientemente, sobre el entorno.

3.2. Programación genética

La GP, propuesta por *John R. Koza* [128], se ha definido como un tipo de EA para evolucionar programas de ordenador capaces de realizar una tarea determinada definida por el usuario. Está inspirada en los GAs, y se considera como un tipo especial de GAs donde los individuos son objetos que pueden ser ejecutados, por lo que muchos investigadores la consideran una rama de los GAs. Así pues, se trata de una técnica de aprendizaje automático donde se optimiza una población de programas de ordenador, estando la evolución guiada por una heurística que determina la aptitud de los individuos o programas.

La principal diferencia entre ambas estrategias estriba en la codificación de los individuos, ya que en los GAs la estructura de los mismos se corresponde normalmente con cadenas binarias de una longitud fija, mientras que en GP los cromosomas no tienen una estructura lineal, sino que habitualmente se representan en forma en forma de árbol, con lo que además tienen una longitud variable. Por otra parte, ambas técnicas difieren también en el ámbito de aplicación, dado que los GAs se suelen adoptar para tareas de optimización, mientras que la GP se emplea más frecuentemente en problemas de aprendizaje [23].

Para representar los individuos o programas de ordenador que evolucionan en la población, la mayoría de las propuestas de GP utilizan una estructura de árbol. Los árboles se componen de una serie de **primitivas**, de las que se distinguen dos tipos:

- **Terminales**, que son datos (variables o constantes) sobre los que actúan los programas. Proporcionan un valor al sistema, y se corresponden con los nodos hoja.
- **Funciones o no terminales**, que consisten en acciones de los programas (operaciones aritméticas, booleanas, funciones matemáticas, etc.). Por lo tanto, procesan un valor que ya está en el sistema, y se corresponden con los nodos interiores.

Para poder asegurar la aplicación del paradigma de GP a un determinado problema, hay que garantizar que los conjuntos de terminales y funciones escogidos satisfagan las propiedades de cierre y suficiencia. El **cierre** establece que cada función ha de ser capaz de manejar cualquier valor que pueda recibir como entrada, incluyendo salidas producidas por otra función u operador del conjunto de no terminales. El incumplimiento de esta propiedad repercutiría en la generación de individuos no válidos, con la consiguiente pérdida de eficiencia que supone este hecho en el proceso evolutivo. Por su parte, la **suficiencia** implica que el poder expresivo del conjunto de terminales y no terminales debe ser suficiente para poder representar una solución al problema.

Además de la GP estándar basada en árboles, se pueden encontrar otros dos tipos de representaciones:

- **GP lineal** [18]. En este tipo de GP, los individuos presentan una estructura lineal que codifica una serie de instrucciones de un lenguaje de programación imperativo o lenguaje máquina.

- **GP cartesiana** (CGP, *Cartesian Genetic Programming*) [150]. Esta extensión de la GP genera programas con estructura de grafo. El genotipo consiste en una lista de enteros que representan las primitivas del programa y cómo se conectan. Utilizando dicha representación, las soluciones pueden codificarse utilizando programas más compactos.

El cruce y la mutación son los dos **operadores** principales en GP. El operador de **cruce** es capaz de generar mucha diversidad por sí mismo [23]. En el cruce estándar, se generan dos sujetos a partir de dos progenitores combinando al azar partes de los mismos. Se selecciona aleatoriamente un nodo en ambos individuos y a continuación se intercambian los subárboles que tienen como raíz dichos nodos. En la Figura 3.6 se muestra un ejemplo donde se puede observar que los dos descendientes han intercambiado el material genético coloreado de sus progenitores.

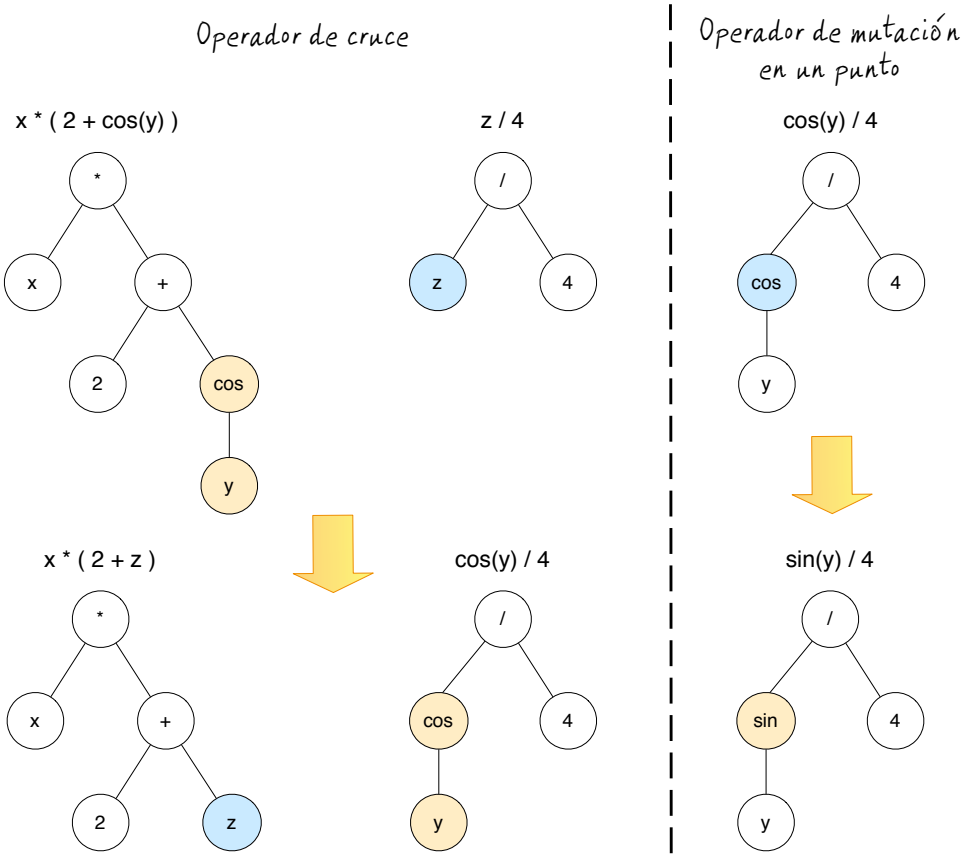


Figura 3.6: Operadores de cruce y mutación en un punto en GP

El operador de **mutación** actúa sobre un único individuo, produciendo en él cambios aleatorios. Se selecciona un nodo al azar y, bien se sustituye por otro de similares características (mutación en un punto), o bien se reemplaza todo el subárbol que cuelga de él por otro generado al azar (mutación por subárbol aleatorio). En la parte derecha de la Figura 3.6 se incluye un ejemplo de mutación en un punto.

Aparte del cruce y la mutación, podemos distinguir los operadores de reproducción, permutación, edición, encapsulación y decimación. El de **reproducción** consiste simplemente en seleccionar un individuo de la población actual para insertar una copia del mismo en la siguiente generación. En la **permutación** se remplaza una lista de argumentos de una función por una permutación de los mismos. El operador de **edición** se usa para simplificar una expresión, remplazándola por otra más sencilla, simplificada. La **encapsulación** consiste en convertir un subárbol en una nueva función que puede intervenir así en otros árboles, lo cuál permite restringir los posibles puntos de cruce de un árbol. Y la **decimación** se encarga de eliminar un porcentaje de los individuos de la población según la aptitud de los mismos, con el fin de evitar evaluar soluciones que van a ser desechadas más adelante en posteriores generaciones.

A partir de la GP original, han surgido otras variantes que hacen uso de distintos formalismos para satisfacer la propiedad de cierre. Como los árboles se construyen de forma aleatoria, todos los nodos del árbol deben recibir y devolver el mismo tipo de datos, lo cuál supone un problema a la hora de trabajar con tipos de datos heterogéneos. Se pueden distinguir cuatro modificaciones para solucionar este problema:

- **GP fuertemente tipada** (STGP, *Strongly Typed Genetic Programming*) [152]. La STGP fuerza la propiedad de cierre para que permita la utilización de variables de cualquier tipo de datos. La propiedad de cierre implica que cualquier elemento puede ser nodo hijo de otro elemento en la estructura de árbol, sin que existan conflictos en cuanto al tipo de datos. Por tanto, en la STGP, las variables, constantes, argumentos y valores devueltos por las funciones pueden ser de cualquier tipo de datos, pero el tipo de dato de cada elemento ha de especificarse de antemano. De este modo se asegura que en el proceso

de inicialización de la población y en la generación de nuevos individuos mediante los operadores genéticos sólo se construyen individuos sintácticamente correctos.

- **GP con restricciones gramaticales** [27]. En este tipo de GP se usa **gramática de contexto libre** (CFG, *Context-Free Grammar*) para chequear la corrección de los individuos.
- **GP guiada por gramática** (G3P, *Grammar Guided Genetic Programming*) [227]. En la G3P, los individuos presentan como genotipo un árbol sintáctico perteneciente a una gramática definida por el usuario, la cual restringe el espacio de búsqueda de modo que sólo se generen individuos gramaticalmente correctos. La gramática aporta un mayor control sobre la estructura de los individuos y sobre los operadores genéticos que se aplican a estos. Los sistemas de G3P emplean habitualmente una CFG, pero existen representaciones de la gramática alternativas tales como la gramática de adjunción de árboles (TAG, *Tree-Adjoining Grammar*) [105], gramáticas de atributo [167], gramáticas lógicas [234] y otras. La extensión de los sistemas de G3P donde los individuos se representan con una estructura lineal variable se conoce con el nombre de **evolución gramatical** (*Grammatical Evolution*) [168].

3.2.1. Algoritmos de GP para minería de reglas de clasificación

La GP ofrece un gran potencial para la inducción de clasificadores, por lo que se han desarrollado multitud de modelos para el aprendizaje de reglas de clasificación utilizando esta metaheurística. Los primeros modelos tienen una semántica más reducida, dado que emplean la GP tradicional [128], mientras que los modelos más recientes han hecho uso de la GP con restricciones gramaticales, STGP y G3P.

La mayoría de problemas de GP que abordan esta tarea generan un clasificador para un conjunto de datos específico, en lugar de extraer una solución genérica que se pueda aplicar posteriormente a otros conjuntos de datos. Encontramos una excepción en el trabajo de *Pappa y Freitas* [177], donde el objetivo es evolucionar un

algoritmo de inducción de reglas genérico que pueda aplicarse a cualquier conjunto de datos, en lugar de extraer reglas directamente.

Se puede encontrar una revisión bibliográfica sobre los algoritmos de GP aplicados a clasificación en el trabajo publicado por *Espejo et al.* [70]. Entre ellos, nos interesan especialmente los algoritmos de G3P, al tratarse del tipo de GP que más relación comparte con los algoritmos de AP que se proponen en esta tesis, que también siguen un enfoque basado en gramática. Existen multitud de propuestas de algoritmos de G3P basados en CFG para clasificación en la literatura [54, 215, 216, 233].

A continuación explicaremos brevemente el algoritmo propuesto por *Bojarczuk et al.* [27], al tratarse de un algoritmo que ha obtenido buenos resultados en cuanto a comprensibilidad y exactitud en su aplicación a conjuntos de datos de medicina, por lo que se utiliza a fines comparativos en algunos estudios experimentales de esta tesis. Se trata de un algoritmo restringido por sintaxis que representa las reglas definiendo un conjunto de funciones consistentes tanto en operadores lógicos (AND, OR) como relacionales ($=, \neq, \leq, >$). El algoritmo de **Bojarczuk** sigue un enfoque mixto $individuo = regla/conjunto\ de\ reglas$ para la codificación de los individuos, en el cual cada individuo codifica un conjunto de reglas disyuntivas para predecir la misma clase. El clasificador generado para un determinado problema contará con k individuos, donde k se corresponde con el número de clases en el conjunto de datos. Los operadores genéticos que considera este algoritmo son el cruce y la reproducción, por lo que no se produce mutación alguna durante la evolución. Con el objetivo de evolucionar reglas comprensibles, la función de *fitness* tiene en cuenta tres componentes:

$$fitness = sensibilidad \cdot especificidad \cdot simplicidad \tag{3.2}$$

donde la sensibilidad y la especificidad se calculan como se indica en la ecuaciones 2.4 y 2.5, respectivamente, y la simplicidad se calcula como sigue:

$$simplicidad = \frac{maxnodos - 0{,}5 \cdot numnodos - 0{,}5}{maxnodos - 1} \tag{3.3}$$

donde $maxnodos$ es el número máximo de nodos permitido y $numnodos$ es el número de nodos actual. Por tanto, el objetivo de la función de $fitness$ es maximizar tanto sensibilidad como especificidad, minimizando simultáneamente la

complejidad del conjunto de reglas. Cuando el proceso evolutivo finaliza se forma el clasificador tomando el mejor individuo encontrado para cada clase, por lo que quedará compuesto por k individuos.

3.2.2. Algoritmos de GP para minería de reglas de asociación

Para la tarea de ARM no se dispone del elenco de algoritmos de GP que existen para clasificación. De hecho, ha sido recientemente cuando *Luna et al.* [144] presentaron la primera propuesta de GP para la tarea de ARM, que denominaron **G3PARM**, y que se fundamenta en el uso de una CFG. Los resultados demostraron subsanar los principales inconvenientes que presentan los algoritmos clásicos de ARM, como son el alto tiempo de ejecución de los mismos, la imposibilidad de trabajar con atributos numéricos y la complejidad de las soluciones. Cada solución encontrada por el algoritmo satisface el lenguaje definido en la gramática. Este algoritmo se describió como un algoritmo totalmente configurable, donde existen parámetros tales como los umbrales de soporte y confianza, las probabilidades de cruce y mutación, el número de generaciones o el tamaño de la población. Es altamente recomendable el uso de este tipo de algoritmos en diversas situaciones, especialmente si son ejecutados por expertos en minería de datos.

También existen otros dos algoritmos de G3P para ARM, variaciones del algoritmo G3PARM que abordan la optimización del *fitness* de los individuos desde una perspectiva multiobjetivo, tratando de optimizar varias medidas simultáneamente. Fueron denominados **NSGA-G3PARM** y **SPEA-G3PARM**, ya que siguen el enfoque de optimización multiobjetivo de dos algoritmos de optimización clásicos, el NSGA2 [55] y el SPEA2 [246], respectivamente.

3.3. Optimización mediante colonias de hormigas

La metaheurística de ACO es un paradigma de optimización bioinspirado basado en agentes emplazado dentro de la SI [29]. La ACO basa el diseño de sistemas

inteligentes multiagente en el comportamiento y la organización que adoptan las colonias de hormigas en su búsqueda de alimento, cuando las hormigas se comunican entre sí de manera indirecta a través del entorno —fenómeno conocido como **estigmergia**—, secretando a su paso una sustancia química denominada **feromona**. Inicialmente, las hormigas exploran aleatoriamente el área que rodea su hormiguero. Tan pronto como alguna hormiga encuentra alimento, evalúa la cantidad y la calidad del mismo y transporta parte de vuelta al nido. Es durante el viaje de retorno cuando la hormiga deja un rastro de feromonas. La concentración de feromonas en un camino dado se incrementará conforme más hormigas sigan dicho camino, y se decrementará a medida que las hormigas dejen de seguirlo. Cuanto mayor sea el nivel de feromonas en un camino, mayor será la probabilidad de que una hormiga vaya por él.

Desde un punto de vista de sistemas multiagentes [75], las hormigas son **agentes muy simples** que se comunican indirectamente entre sí únicamente por medio del entorno, que funciona como si de una **memoria colectiva** se tratara. La combinación de las acciones locales independientes representa un mecanismo de optimización global basado en la retroalimentación.

Pero, ¿cómo encuentran las hormigas el camino más corto hacia el alimento? Para estudiar dicho comportamiento, *Goss et al.* [28, 94] llevaron a cabo un experimento con hormigas de la especie *Iridomyrmex humilis*, colocando una fuente de comida y un hormiguero conectados entre sí mediante un puente con dos ramas, ejemplificado en la Figura 3.7. Así pues, existían dos posibles caminos para llegar del hormiguero al alimento, siendo estos de diferente longitud, y en los cuales no existía previamente concentración alguna de feromonas, de tal modo que las hormigas pudieran seleccionar inicialmente un camino u otro con idéntica probabilidad. Con este experimento se demostró que, si el camino largo poseía la longitud suficiente (al menos el doble de longitud que el corto), las hormigas tendían a escoger la rama más corta. Este comportamiento se explica porque las hormigas dejan feromonas en el entorno y, a su vez, son atraídas por los rastros de feromona existentes. Así pues, las primeras hormigas en volver al nido desde la fuente de alimento son aquellas

que van por la rama más corta, con lo que refuerzan el rastro de feromonas en ella. Por tanto, el mayor nivel de feromonas existente tiende a reclutar a las nuevas hormigas que parten del hormiguero. Cuanto más tiempo pasa, más se refuerza este camino más corto y menos hormigas viajan por la rama larga.

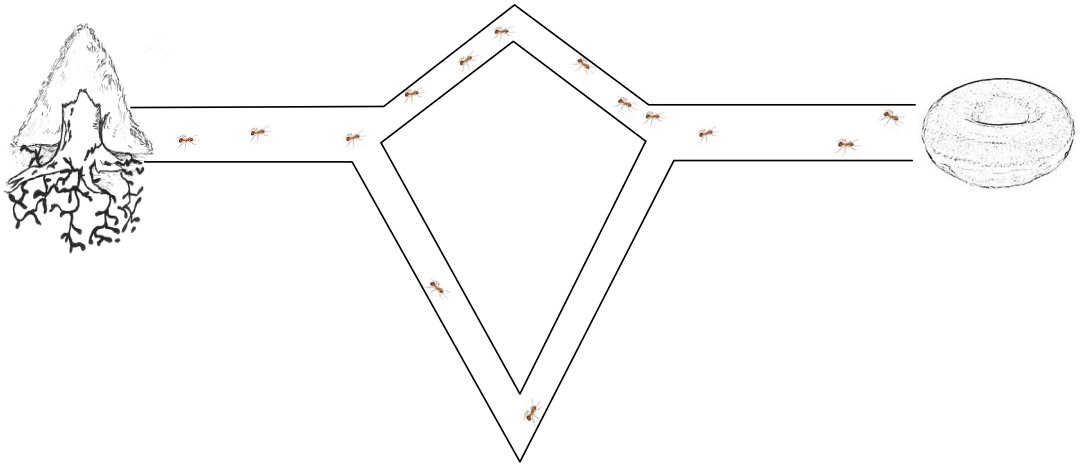


Figura 3.7: Experimento *shortest bridge*

En este experimento también se comprobó que si al inicio se obstruye el camino corto durante un período de tiempo, una vez se habilita su paso por él las hormigas no lo recorrerán al estar ya la otra rama marcada con feromonas. Este problema que se produce en la naturaleza se puede solventar en un sistema artificial mediante la introducción de la evaporación: si las feromonas se evaporan suficientemente rápido, se dificulta el mantenimiento de un rastro de feromonas estable a lo largo de un camino largo. Así, la rama más corta terminará siendo la más recorrida incluso si se presenta más tarde que la otra. Esta propiedad es especialmente deseable en optimización, donde se busca evitar la convergencia hacia soluciones mediocres.

Goss et al. [94] fueron capaces de describir el comportamiento observado mediante una ecuación matemática. En ella, asumiendo que en un momento dado m_s hormigas han pasado ya por la ruta corta, y m_l han ido por la larga, la probabilidad P_s de que una nueva hormiga escoja el camino más corto viene definida por:

$$P_s = \frac{(m_s + \tau_s)^r}{(m_s + \tau_s)^r + (m_l + \tau_l)^r} \tag{3.4}$$

donde:

- m_s indica el número de hormigas que han pasado por el camino corto,
- m_l indica el número de hormigas que han pasado por el camino largo,
- τ_s es la concentración de feromonas en el camino corto,
- τ_l indica la concentración de feromonas en el camino largo,
- r es el coeficiente que indica la longitud del camino largo respecto al corto, siendo $r \geq 2$

Transferir esta idea desde la naturaleza a un modelo artificial supone hacer uso de agentes computacionales simples que trabajen cooperativamente, comunicándose entre sí mediante los rastros de feromonas. Se trata de una abstracción de algunos patrones de comportamiento de las hormigas, pero también se consideran ciertas características que no tiene su homólogo en la naturaleza, como por ejemplo el uso de información heurística. Teniendo en cuenta que, a diferencia de las hormigas reales, las hormigas artificiales se van a mover entre estados discretos [52], cada hormiga se moverá de un estado S_i a otro S_j en base a dos factores:

- **Información heurística**: mide la preferencia heurística para moverse del estado S_i al S_j, y se denota como η_{ij}. La información pertenece al dominio del problema y se conoce antes de la ejecución del algoritmo.
- **Rastro de feromonas artificial**: mide la concentración de feromonas en la transición del estado S_i al S_j, y se denota como τ_{ij}. Este valor vendrá dado por la historia pasada de la colonia de hormigas, es decir, por la deposición de feromonas realizadas por hormigas que han pasado por dicha transición previamente. Normalmente suele existir un mecanismo de evaporación, similar a la evaporación real de feromonas en la naturaleza, que modifica esta información a lo largo del tiempo. En cierto modo, la evaporación permite a la colonia ir olvidando poco a poco su historia de forma que pueda dirigir su búsqueda hacia nuevas direcciones, sin quedar limitada por decisiones pasadas.

Una **hormiga artificial** es un agente simple que intenta construir soluciones factibles a un problema explotando los rastros de feromonas disponibles y la información heurística.

Las **características** que presenta una hormiga artificial son las siguientes [65, 154]:

- Tiene una **memoria interna** para almacenar los estados que ha visitado (el camino que ha seguido).
- A partir de un **estado inicial** del entorno (espacio de estados), una hormiga trata de encontrar una solución factible al problema.
- El movimiento de la hormiga viene guiado por los factores comentados anteriormente –heurística y feromonas–, que se tienen en cuenta en la **regla de transición**.
- Deposita una cantidad de feromonas que viene dada por una regla específica de **actualización de feromonas**. Las feromonas depositadas se pueden asociar a los estados o bien a las transiciones entre estados.
- Si la deposición de feromonas tiene lugar al mismo tiempo que se construye la solución, es decir, durante cada transición entre estados, se denomina **actualización de feromonas en línea paso a paso** (*online step-by-step pheromone update*). Si en cambio se produce una vez que la solución ha sido encontrada, volviendo por los estados que ha seguido la hormiga, la actualización se conoce como **actualización de feromonas en línea retrasada** (*online delayed pheromone update*). También se puede dar el caso de que la actualización tenga lugar una vez todas las hormigas de una generación han sido generadas, en cuyo caso se denomina **actualización de feromonas fuera de línea** (*offline pheromone update*).
- **Características adicionales**. Las hormigas artificiales también pueden presentar otras características que no poseen sus homólogas en la naturaleza pero que permiten mejorar su rendimiento, como por ejemplo la hibridación con procedimientos de optimización locales.

Inicialmente, los algoritmos de ACO fueron aplicados a problemas de optimización combinatoria, obteniendo soluciones óptimas o muy próximas al óptimo. Desde entonces, los algoritmos de ACO se han empleado en un creciente rango de problemas, demostrando su efectividad para abordar problemas de DM.

La estructura genérica que sigue la metaheurística de ACO se recoge en la Figura 3.8, y engloba a todas las propuestas de ACO para problemas de optimización discretos [26].

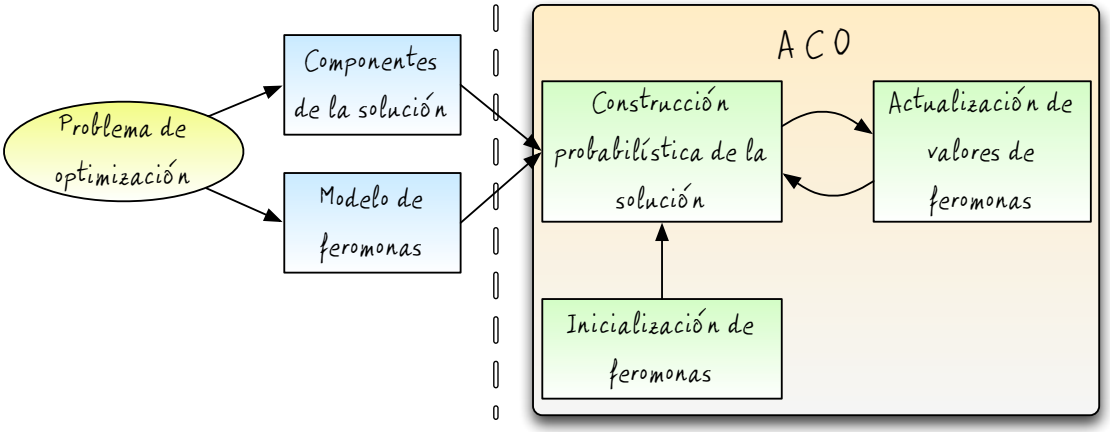


Figura 3.8: Marco computacional de ACO (adaptado de *Blum y Merkle* [26])

Conocido el problema de optimización a resolver, en primer lugar hay que determinar el conjunto de componentes que se pueden emplear para formar una solución. Asimismo, hay que definir el modelo de feromonas que se va a utilizar, el cual guiará la generación probabilística de soluciones a partir del conjunto de componentes disponibles. Una vez inicializados los valores de feromonas, que normalmente se asocian a los componentes de las soluciones o bien a las conexiones entre ellos, se generan los individuos y se modifican los niveles de feromonas para guiar a las siguientes generaciones hacia soluciones de alta calidad.

3.3.1. Evolución de los algoritmos de ACO

El primer algoritmo de ACO fue propuesto por *Dorigo* en su tesis doctoral, y posteriormente publicado bajo la denominación de **Ant System** (**AS**) [64]. Fue aplicado al problema de optimización del viajante de comercio (TSP, *Travelling Salesman Problem*), cuyo objetivo es encontrar un camino cerrado que pase por n ciudades, visitando cada una de ellas una única vez. Todas las ciudades están conectadas entre sí, formando pues un grafo completo de $\frac{n\cdot(n-1)}{2}$ transiciones. En cada iteración del algoritmo, m hormigas buscan soluciones moviéndose de una ciudad a otra hasta completar una gira, siendo siempre $m < n$. Cada hormiga es un agente cuyo movimiento para viajar de una ciudad a otra viene determinado por:

- La restricción de no haber visitado la ciudad destino previamente.
- Una función heurística calculada como el inverso de la distancia entre las dos ciudades (cuanto más cerca se encuentren las ciudades, mayor será el valor de la heurística).
- La cantidad de feromonas existente en el camino entre ambas ciudades.

Así, la probabilidad de que la hormiga pase de la ciudad i a la j viene determinada por:

$$P_{ij}^{k} = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\Sigma_{k \in no\_visitadas} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \tag{3.5}$$

donde:

- η representa la función heurística,
- τ la concentración de feromonas,
- α controla la importancia de la heurística,
- β permite controlar la importancia de las feromonas, y
- k representa cualquier ciudad que la hormiga tenga pendiente visitar.

Una vez que todas las hormigas han completado su gira, se actualizan los niveles de feromonas en el entorno. Primero tiene lugar el proceso de evaporación sobre todas las transiciones existentes, y a continuación cada hormiga deposita feromonas en las transiciones que ha seguido.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \triangle\tau_{ij}^{k}, \forall \tau_{ij} \in \mathcal{T} \tag{3.6}$$

donde:

- ρ indica el valor del coeficiente de evaporación,
- $\triangle\tau_{ij}^{k}$ es la cantidad de feromonas depositada por la hormiga k en (i, j), y
- $\mathcal{T}$ representa el conjunto de transiciones o aristas del grafo.

Todas las hormigas depositan feromonas en cada una de las transiciones que han seguido al construir su gira de manera inversamente proporcional a la longitud del camino encontrado. Por tanto, cuanto más corto sea el camino encontrado para recorrer las n ciudades, mayor cantidad de feromonas se deposita:

$$\triangle\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L^{k}} & \text{si la hormiga } k \text{ toma la transición } (i, j) \\ 0 & \text{en otro caso} \end{cases} \tag{3.7}$$

donde:

- Q es una constante, y
- L^{k} indica la longitud del camino encontrado por la hormiga k.

A pesar de que el algoritmo AS no era competitivo resolviendo el problema del TSP con respecto a algoritmos de otros paradigmas, sí que resultó esencial para promover y estimular el trabajo de otros investigadores en el área de ACO, los cuales propusieron extensiones del mismo que perseguían una mejora en el rendimiento.

La primera mejora que se propuso consistía simplemente en dotar de una mayor importancia al mejor camino encontrado hasta el momento por el algoritmo, lo que se traducía en efectuar una deposición de feromonas adicional en cada una de las transiciones de dicho camino. El nombre con el que se denominó a este algoritmo fue el de **AS elitista** [64].

Dorigo y Gambardella presentaron el algoritmo **Ant Colony System (ACS)** [63], cuya contribución más interesante fue la introducción de una actualización de feromonas adicional, a nivel local, que se va produciendo conforme las hormigas

construyen la solución y que se conoce con el nombre de regla de actualización local. El objetivo de la misma radica en buscar la diversificación del proceso de búsqueda tanto como sea posible durante la fase de construcción de soluciones:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \tag{3.8}$$

donde:

τ_0 es la concentración de feromonas inicial de todas las transiciones del entorno.

Otra diferencia estriba en que la regla de actualización global sólo se aplica a las transiciones que forman parte de la mejor solución global encontrada hasta el momento:

$$\tau_{ij} = \begin{cases} (1 - \rho) \cdot \tau_{ij} + \rho \cdot \triangle\tau_{ij} & \text{si la mejor hormiga ha recorrido } (i, j) \\ \tau_{ij} & \text{en otro caso} \end{cases} \tag{3.9}$$

Además, la regla de transición del algoritmo ACS se ve modificada para permitir buscar un balance entre la exploración de nuevas conexiones y la explotación del conocimiento acumulado previamente sobre el problema. Se la conoce como regla de transición pseudoaleatoria, ya que depende del valor de una variable q generada aleatoriamente en el intervalo $[0, 1]$:

$$P_{ij}^k = \begin{cases} 1 & \text{si } q \leq q_0 \\ \dfrac{(\eta_{ij})^\alpha \cdot (\tau_{ij})^\beta}{\Sigma_{k \in no\_visitadas} (\eta_{ik})^\alpha \cdot (\tau_{ik})^\beta} & \text{en otro caso} \end{cases} \tag{3.10}$$

donde:

q_0 es el parámetro que permite dar mayor prioridad a la explotación o a la exploración.

Otra modificación al algoritmo AS fue la propuesta de *Bullnheimer et al.*, denominada **Rank-Based Ant System (RBAS)** [37], donde se introduce la actualización de feromonas basada en rangos. Una vez que las m hormigas encuentran una solución se ordenan según la calidad de la misma, indicada por la longitud

del camino que han seguido. Así, la contribución de feromonas de cada hormiga al entorno viene dada por la posición de la hormiga o rango, y sólo las mejores ω hormigas son responsables de reforzar los niveles de feromonas. La regla de transición permanece idéntica a la del algoritmo original, pero la actualización de feromonas se calcula como:

$$\tau_{ij} = \rho \cdot \tau_{ij} + \triangle\tau_{ij}^{rank} + \triangle\tau_{ij}^{gb},$$

$$\triangle\tau_{ij}^{rank} = \begin{cases} \sum_{\mu=1}^{\omega-1}(\omega-\mu)\dfrac{Q}{L^{\mu}} & \text{si la } \mu\text{-ésima mejor hormiga pasa por } (i,j) \\ 0 & \text{en otro caso,} \end{cases} \tag{3.11}$$

$$\triangle\tau_{ij}^{gb} = \begin{cases} \omega\dfrac{Q}{L^{gb}} & \text{si la mejor hormiga ha pasado por la transición } (i,j) \\ 0 & \text{en otro caso} \end{cases}$$

donde:

$\triangle\tau_{ij}^{rank}$	se refiere al refuerzo de las hormigas elitistas,
$\triangle\tau_{ij}^{gb}$	representa el refuerzo hecho por la mejor hormiga global encontrada,
μ	indica el índice del rango, y
ω	es el número de hormigas elitistas consideradas.

Otra variante fue el algoritmo **Max-Min Ant System (MMAS)** [210], introducido por *Stützle y Hoos* con el objetivo de evitar la convergencia prematura hacia óptimos locales que se observaba en el algoritmo AS. El MMAS difiere del original en tres cuestiones principales. La primera es que únicamente la mejor hormiga puede reforzar los rastros de feromonas, al igual que ocurría en el algoritmo ACS, por lo que la regla de actualización de feromonas es idéntica a la de la Ecuación 3.9. La segunda diferencia viene impuesta por la restricción de que los niveles de feromonas han de permanecer dentro de un determinado umbral, $[\tau_{min}, \tau_{max}]$. Y la tercera es que, al comienzo del algoritmo, el nivel de feromonas en todas las posibles transiciones es inicializado a τ_{max}, logrando así que se produzca una mayor exploración del espacio de soluciones durante las primeras iteraciones.

3.3.2. Algoritmos de ACO para minería de reglas de clasificación

El interés de la comunidad científica en los algoritmos de ACO se ha visto considerablemente incrementado en los últimos años, ya que los buenos resultados obtenidos a nivel académico los ha hecho atractivos para la industria. De hecho, se está aplicando esta metaheurística en un amplio abanico de áreas y problemas, y su uso está creciendo a nivel empresarial para su implantación en sistemas del mundo real [43]. Para ver una lista de aplicaciones de ACO agrupadas por el tipo de problema, referimos al lector a [25, 62, 154]. En esta sección nos centraremos en su aplicación a la tarea de clasificación de DM.

El primer algoritmo en incorporar los principios y conceptos de ACO a la inducción de reglas de clasificación fue **Ant-Miner** [179], propuesto por *Parpinelli et al.*, y se han propuesto numerosas extensiones y modificaciones al mismo, llegando a ser el algoritmo de ACO más referenciado en este campo.

Ant-Miner sigue un enfoque iterativo o de cubrimiento secuencial (ver Sección 2.1.4.1) donde, a partir del conjunto de entrenamiento, el algoritmo extrae nuevas reglas para añadirlas al conjunto de reglas descubiertas. Conforme lo va haciendo, elimina aquellas instancias del conjunto de entrenamiento que son cubiertas por cada una de las reglas encontradas, reduciendo así el tamaño del mismo. Durante la construcción de cada nueva regla, aplicando la regla de transición, Ant-Miner escoge un término para añadirlo a la regla parcial actual, cuyo antecedente estará compuesto por la conjunción de dichos términos. Un término no es más que una combinación de un atributo del conjunto de datos con uno de sus posibles valores, y el algoritmo únicamente considera la inclusión de aquellos términos que no hayan sido seleccionados previamente en la misma regla. Como función heurística, emplea una medida de información basada en la entropía [124]. La clase que el algoritmo asigna como consecuente a la regla actual será la clase mayoritaria de entre las instancias cubiertas por el antecedente. El algoritmo continúa descubriendo nuevas reglas hasta que el conjunto de entrenamiento queda vacío o bien hasta que el número de reglas

descubiertas supera el número máximo de reglas permitido para el clasificador. La calidad de las reglas se mide con el producto entre sensibilidad y especificidad:

$$fitness = \underbrace{\frac{T_P}{T_P + F_N}}_{\text{sensitivity}} \cdot \underbrace{\frac{T_N}{T_N + F_P}}_{\text{specificity}} \tag{3.12}$$

donde T_P, F_P, T_N y F_N indican verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos, respectivamente.

La Tabla 3.1 lista cronológicamente las distintas modificaciones y extensiones del algoritmo Ant-Miner para la extracción de reglas de clasificación que han sido publicadas hasta la fecha. En ella se resumen las principales diferencias de estas propuestas respecto a la original, agrupándolas en la tabla bajo uno de los siguientes grupos: mecanismos de poda para las reglas, actualización de las feromonas, función heurística, regla de transición, función de *fitness*, reglas intervalares, atributos continuos, reglas difusas y clasificación multietiqueta y jerárquica [17, 41, 42, 45, 88, 116, 139, 140, 147, 155, 170, 172–174, 197, 207, 214, 226].

Muchas de estas extensiones suponen cambios menores y sus resultados difieren ligeramente de los obtenidos por el algoritmo Ant-Miner original. Por ejemplo, *Liu et al.* [139] presentaron el algoritmo **Ant-Miner2**, donde aplicaron una función heurística mucho más simple asumiendo que el refuerzo de feromonas tiene suficiente capacidad para compensar los posibles errores inducidos por el uso de dicha función heurística, más simple pero a su vez menos efectiva. Además de dicho cambio en la función heurística, los mismos autores modificaron la regla de transición y el refuerzo de feromonas en el algoritmo **Ant-Miner3** [140].

En cambio, **Ant-Miner+**, presentado por *Martens et al.* [147], ha demostrado que obtiene resultados de *accuracy* superiores a las versiones previas de Ant-Miner. Este algoritmo define el entorno como un grafo directo acíclico que permite seleccionar transiciones mejores e incluir intervalos en las reglas. Además, el rendimiento que obtiene Ant-Miner+ es superior por dos motivos: el algoritmo clásico de ACO en que se basa es el MMAS, y además hace uso de una función heurística más dependiente de la clase y más exacta. La función de *fitness* se define como la suma de la confianza ($confidence$) y la cobertura ($coverage$) del individuo, como se puede

Referencia	Poda	Feromonas	Heurística	Regla de transición	Función de *fitness*	Reglas intervalares	Atributos continuos	Reglas difusas	Multi-etiqueta	Jerárquica
Liu et al. [139]			√							
Liu et al. [140]		√	√	√						
Wang, Feng [226]		√	√	√						
Chen et al. [45]	√	√	√							
Chan, Freitas [41]	√									
Chan, Freitas [42]					√				√	
Smaldon, Freitas [207]	√	√	√		√					
Jin et al. [116]	√	√								
Galea, Shen [88]		√	√					√		
Swaminathan [214]		√				√				
Martens et al. [147]	√	√	√		√	√				
Otero et al. [172]		√	√				√			
Nalini et al. [155]	√	√	√		√		√			
Otero et al. [170, 173]	√	√	√		√		√			√
Otero et al. [174]		√	√				√			
Salama, Abdelbar [197]		√	√	√	√					
Baig, Shahzad [17]	√		√		√					
Otero et al. [171]		√	√				√			

Tabla 3.1: Modificaciones y extensiones de Ant-Miner para la inducción de reglas de clasificación

apreciar en la fórmula siguiente:

$$fitness = \underbrace{\frac{\mid A \cup C \subseteq I, I \in D \mid}{\mid A \subseteq I, I \in D \mid}}_{\text{confianza}} + \underbrace{\frac{\mid A \cup C \subseteq I, I \in D \mid}{\mid Cov = 0 \mid}}_{\text{cobertura}}$$

donde:

A	representa el antecedente de la regla,
C	representa el consecuente de la regla,
I	es el número de instancias,
D	es el conjunto de datos, y
$Cov = 0$	indica el número de instancias no cubiertas por ninguna regla.

Por tanto, la confianza se refiere al cociente entre el número de instancias pertenecientes al conjunto de datos que incluyen tanto el antecedente como el consecuente, y el número de instancias que incluyen el antecedente. La cobertura se calcula como el número de instancias que incluyen antecedente y consecuente, dividido por el número de instancias que no son cubiertas por ninguna de las reglas extraídas.

Otra diferencia importante de Ant-Miner+ radica en el valor seleccionado para los exponentes de la heurística y de las feromonas (α y β). De hecho, introduce un rango de valores permitidos para cada parámetro y permite a las hormigas elegir valores adecuados de manera autónoma.

Otro algoritmo interesante por su capacidad para trabajar directamente con atributos numéricos es el **cAnt-Miner**, propuesto por *Otero et al.* [172]. Ya que la metaheurística de ACO no se presta para su utilización en dominios continuos, lo que cAnt-Miner hace durante su ejecución es crear internamente intervalos discretos para este tipo de atributos. Los mismos autores presentaron posteriormente una modificación de dicho algoritmo a la que denominaron **cAnt-Miner2-MDL** [174] donde incorporaron y aplicaron en la fase de construcción de reglas el **principio de minimización de la longitud de descripción** (MDLP, *Minimum Description Length Principle*) [19, 187], explicado en el Anexo A.3. Otra diferencia de esta segunda versión es que los valores de feromonas se asocian a las aristas del grafo en lugar de a los vértices, que es como proceden cAnt-Miner y el propio Ant-Miner.

Salama y Abdelbar [196, 197] estudiaron diversas modificaciones del algoritmo Ant-Miner. En primer lugar, incorporaron la idea de emplear múltiples tipos de feromonas. También usaron un nuevo procedimiento de actualización de feromonas cuyo propósito era intensificar las diferencias entre soluciones malas, buenas, excepcionalmente buenas y aquellas soluciones no visitadas durante la construcción de una regla. A diferencia del algoritmo original, donde sólo se permiten condiciones representando combinaciones entre los atributos y los posibles valores, propusieron la utilización del operador lógico de negación en los antecedentes de las reglas, simulando así el operador distinto. Por otro lado, también incorporaron el concepto de hormigas testarudas [2], donde las hormigas están guiadas e influenciadas también por las decisiones que han tomado previamente. Finalmente, permitieron que cada hormiga tuviera valores propios para los exponentes α y β, de manera que ciertas hormigas pudieran considerar más importante la información heurística y otras, en cambio, hicieran lo propio con la información procedente de las feromonas.

Otra de las propuestas que se han publicado es la del algoritmo **Ant-Miner-C**, presentada por *Baig y Shazhad* [17]. Entre sus principales características podemos destacar el empleo de una función heurística basada en el nivel de correlación entre atributos, mientras que la función de $fitness$ es similar a la empleada en el algoritmo Ant-Miner+. Cabe destacar el modo de asignar el consecuente a las reglas, decidiendo de antemano la clase que predecirá la regla que se va a construir. También se incorporaron un nuevo criterio de parada para no seguir añadiendo términos al antecedente de la regla, así como una estrategia de poda que únicamente se aplica a la mejor regla de una iteración, en lugar de a todas las reglas encontradas en ella.

La última modificación directa al algoritmo original para minería de reglas de clasificación de la cuál tenemos constancia es la de *Otero et al.* [171], cuya principal diferencia radica en el uso de una nueva estrategia de cubrimiento secuencial que trata de mitigar los problemas inherentes a la que emplean Ant-Miner y el resto de sus extensiones. En ellas, aunque el descubrimiento de una regla se puede ver como un problema de búsqueda independiente, una vez que la regla es construida se eliminan los ejemplos cubiertos por la misma en el conjunto de entrenamiento, lo que modifica el espacio de búsqueda, afectando a las reglas que se descubren

posteriormente. La idea que se propone en este artículo es tener en cuenta la interacción entre reglas, incorporando ideas del enfoque de Pittsburgh, introducido en la Sección 2.1.4.3. De ahí el nombre del algoritmo, **cAnt-Miner**$_{PB}$ (cAnt-Miner basado en el enfoque de Pittsburgh). A pesar de que sigue utilizando un enfoque de cubrimiento secuencial, cada individuo crea una lista completa de reglas en cada iteración del algoritmo en lugar de una única regla. Además, las feromonas se actualizan en base a la lista de mejores reglas candidatas de entre las creadas en la iteración.

Recientemente ha aparecido una propuesta de algoritmo de ACO para la inducción de árboles de decisión, combinando estrategias de ambas técnicas. El rendimiento del algoritmo, denominado **Ant-Tree-Miner**, se comparó frente a algoritmos de árboles de decisión y de ACO, obteniendo buenos resultados. Ant-Tree-Miner sigue un enfoque descendente que realiza una selección probabilística de los atributos que se añaden como nodos de decisión basándose en la cantidad de feromonas y la información heurística.

Aparte de estas modificaciones, existen otras extensiones relacionadas con la hibridación de ACO con otras metaheurísticas. Entre ellas, nos llama la atención especialmente el algoritmo **PSO/ACO2**, desarrollado por *Holden et al.* [107]. También se trata de un algoritmo de cubrimiento secuencial, y puede trabajar tanto con atributos nominales como numéricos. La capacidad de utilizar ambos tipos de atributos se debe al carácter híbrido del algoritmo, puesto que, por un lado utiliza ACO para manejar los atributos nominales, y por otro emplea PSO para tratar con los numéricos. La función de *fitness* viene dada por la precisión corregida de Laplace:

$$fitness = \frac{1 + T_P}{1 + T_P + F_P} \tag{3.13}$$

3.3.3. Algoritmos de ACO para minería de reglas de asociación

Tal como se indica en la investigación llevada a cabo por *Martens et al.* [146] sobre la aplicación de algoritmos de SI a DM, las propuestas de este tipo de algoritmos en la literatura se centran en las tareas de clasificación y *clustering*, aunque sería

de esperar que obtuvieran buenos resultados en otras. La misma conclusión es alcanzada en el trabajo de *Michelakos et al.* [149], donde hacen una revisión de las propuestas de ACO para DM.

En nuestra búsqueda bibliográfica únicamente hemos encontrado un algoritmo de ACO para llevar a cabo la tarea de asociación, propuesto por *Kuo y Shih* [132], denominado **ACS-based ARM**. Este algoritmo se utilizó sobre una base de datos de pacientes mantenida por el gobierno de Taiwan, comparando sus rendimiento con el del algoritmo clásico Apriori. Los resultados arrojados por el estudio experimental indicaron que que el tiempo computacional con respecto al algoritmo Apriori era menor, y que el algoritmo propuesto era capaz de extraer menos reglas pero con información más representativa.

El algoritmo ACS-based ARM fue aplicado posteriormente a la misma base de datos, pero empleando previamente dos algoritmos de clustering basados en hormigas, con el objetivo de disminuir el tiempo de cómputo [131]. Esto supuso la generación de tres clusters, sobre los que se aplicó el algoritmo ACS-based ARM. Los resultados obtenidos indicaron que procediendo de esta forma no sólo se extraían las reglas más rápidamente, sino que las reglas permitían a los médicos dedicar más atención a grupos de pacientes más importantes, descubriendo las relaciones ocultas en dichos grupos más fácilmente.

3.4. Programación automática con colonias de hormigas

La AP es un enfoque que sigue los principios de la programación automática para la construcción de programas de ordenador, encontrando soluciones cuasi óptimas para problemas de optimización en un tiempo de cómputo razonable. Presenta ciertas similitudes con la GP, pero mientras que esta última utiliza GAs como técnica de búsqueda, la AP utiliza la metaheurística de ACO.

La construcción de soluciones en GP se basa en la combinación o modificación de otras soluciones mediante el uso de los operadores de cruce y mutación. Este comportamiento presenta varios inconvenientes que tienen que ver con la pérdida de información acerca de la distribución del espacio de búsqueda, su dificultad en

adaptarlo a un entorno en constante cambio, y el mapeo genotipo-fenotipo que implica que un pequeño cambio en el genotipo puede tener un gran impacto en el fenotipo [1]. En cambio, la aplicación de ACO a la programación automática no presenta los problemas mencionados.

Al igual que en la GP, cualquier programa en AP estará compuesto de primitivas, denominados igualmente terminales y funciones. Cualquier algoritmo de AP ha de satisfacer también las propiedades de cierre y suficiencia.

En la literatura existen diferentes propuestas que utilizan la AP. A continuación efectuamos una revisión de las mismas.

El primer trabajo que combinó el paradigma de hormigas con la generación automática de programas fue presentado por *Roux y Fonlupt* [193], aunque no consideramos que se trate de un algoritmo de AP "puro"puesto que estaba fuertemente relacionado con la GP y empleaba algunas de sus características. De hecho, este algoritmo comienza con la creación de una población de programas (árboles) al azar, utilizando el método de inicialización *ramped half-and-half* [69] y almacenando una tabla de feromonas para cada nodo del árbol. Cada tabla de feromonas mantiene la cantidad de feromonas asociada a todos los posibles elementos (también denominados terminales y funciones, al igual que en la GP). Una vez que se evalúa cada programa, la tabla de feromonas se actualiza mediante evaporación y refuerzo basado en la calidad de las soluciones. Estos pasos se repiten hasta que se alcanza un criterio de terminación, pero las nuevas poblaciones de programas ya no se generan como la población inicial, sino de acuerdo a las tablas de feromonas. Este algoritmo fue utilizado con cierto éxito para resolver problemas de regresión simbólica y de multiplexión.

Una idea de codificación de individuos similar fue empleada en [47], trabajo en el que cada hormiga construye y modifica árboles teniendo en cuenta la cantidad de feromonas existente en cada nodo, y donde cada nodo almacena la tabla de feromonas. En este caso los árboles representaban árboles neuronales, y el objetivo del trabajo consistía en evolucionar redes neuronales flexibles. Para este fin, los autores combinaron la AP con la PSO, siendo la primera responsable de evolucionar la arquitectura de redes neuronales flexibles, y la PSO encargada de optimizar los parámetros codificados en el árbol neuronal. Aplicaron las redes neuronales

flexibles desarrolladas a problemas de predicción de series temporales, mostrando la efectividad de su algoritmo.

Boryczka et al. [32, 33] presentaron una serie de métodos englobados bajo la denominación de **Ant Colony Programming** (**ACP**), fundamentados en la utilización del algoritmo clásico ACS. Inicialmente aplicaron este tipo de AP para resolver problemas de regresión simbólica, proponiendo dos versiones diferentes, el enfoque de expresión y el enfoque de programa.

En el enfoque de expresión, el espacio de búsqueda consistía en un grafo definido como $G = (N, E)$, donde N es el conjunto de nodos, que pueden representar bien una variable o un operador, y E es el conjunto de aristas, cada una con un valor de feromonas asociado. Las hormigas se mueven por el grafo generando programas estructurados jerárquicamente. Así, el objetivo era encontrar una función de aproximación expresada en notación prefija. Prácticamente de forma paralela, *Green et al.* [96] también presentaron una técnica de AP similar al enfoque de expresión de ACP, donde el grafo se generaba siguiendo un proceso aleatorio. Un ejemplo de grafo puede verse en la Figura 3.9, donde partiendo del nodo inicial se observa el movimiento de las hormigas por el resto del grafo. Nótese que no se trata de un grafo dirigido, pero para ejemplificar la construcción de un programa se ha indicado el movimiento de las hormigas desde el nodo inicial.

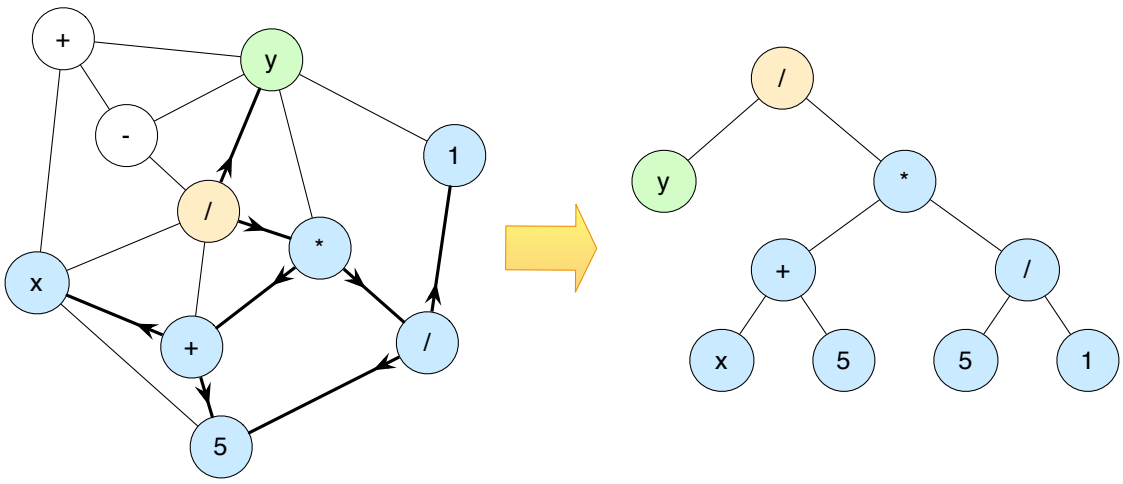


Figura 3.9: Árbol de programa obtenido a partir del grafo en el enfoque de de expresión de ACP (adaptado de *Green et al.* [96])

Por otro lado, en el enfoque de programa, la función de aproximación se construía como un programa de ordenador. En este caso los nodos del grafo representan instrucciones de asignación, y la solución consiste en una secuencia de asignaciones que evalúan la función.

Boryczka también presentó extensiones a estos trabajos con el objetivo de mejorar el rendimiento de ACP. En [30], el autor mejoró la efectividad y logró obtener soluciones más simplificadas eliminando los intrones, mientras que en [31] redujo el tiempo computacional necesario para la evaluación de las reglas de transición.

Basándose en las ideas presentadas previamente por otros autores, Rojas y Bentley desarrollaron un nuevo algoritmo de AP denominado **Grid Ant Colony Programming** (**GACP**) [189], cuya principal característica suponía la consideración del espacio de estados como una rejilla, en lugar de un grafo. La principal ventaja de utilizar una rejilla residía en que permitía a las hormigas almacenar información temporal acerca de los pasos que habían seguido. De esta manera, en el caso de que una misma hormiga visitase un nodo concreto varias veces, podía discriminar qué concentración de feromonas depositar en cada caso. Los autores demostraron que con GACP era posible alcanzar soluciones óptimas encontrando programas para resolver funciones booleanas.

Otro intento de evolucionar programas utilizando ACO fue **AntTAG**, propuesto por *Abbass et al.* [1] como un método de programación automática empleando ACO como estrategia de búsqueda y una TAG para construir programas. Los autores probaron su rendimiento en problemas de regresión simbólica y lograron mejores resultados que los obtenidos por G3P [227] y GP guiada por TAG [104].

Keber y Schuster publicaron otro trabajo basado en gramática llamado **Generalized Ant Programming** (**GAP**) [120], que usa una CFG en lugar de un TAG, y donde las hormigas generan un programa siguiendo un camino por el espacio de estados. *Salehi-Abari y White* [198] continuaron este trabajo proponiendo una modificación del algoritmo que denominaron **Enhanced Generalized Ant Programming** (**EGAP**). En concreto, EGAP introduce una nueva forma de emplazar las feromonas que consiste en situar en un paso de derivación una cantidad de feromonas proporcional a la profundidad del camino; asimismo utiliza una función heurística específica para controlar la terminación de un camino. Posteriormente, *Salehi-Abari y White* publicaron otro trabajo [199] comparando el rendimiento de

GP frente a su algoritmo EGAP en tres problemas diferentes: regresión simbólica, multiplexor y rastro de hormigas de Santa Fe.

Más recientemente, *Shirakawa et al.* [206] propusieron el algoritmo **Dynamic Ant Programming** (**DAP**). Su principal diferencia con respecto a la ACP radica en el uso de una tabla de feromonas que cambia dinámicamente y en un número variable de nodos, lo que deriva en un espacio de estados más compacto. No obstante, los autores únicamente comprobaron el rendimiento de DAP frente a GP sobre problemas de regresión simbólica.

Otro algoritmo que hace uso de AP fue el presentado por *Kumaresan* [129], donde se aplica a control y modelado. En él, el espacio de estados se representa por medio de un grafo donde los nodos representan funciones y terminales y las conexiones entre los mismos almacenan la información relativa a las feromonas. AP se emplea para resolver ecuaciones algebraicas diferenciales, con el objetivo de calcular la solución a la ecuación Ricatti, punto central en la teoría de control óptimo. El autor demuestra que la solución obtenida mediante AP se acerca mucho a la solución exacta del problema.

La última propuesta de AP que encontramos fue presentada por *Hara et al.* [100]. Se trata de un método que combina la CGP con ACO y que denominaron **Cartesian Ant Programming** (**CAP**). En ella, las hormigas construyen programas de ordenador moviéndose por la red de nodos empleada en la CGP. Los autores exploraron el rendimiento de la CAP en problemas de regresión simbólica clásicos y en el problema de espirales entrecruzadas, que no deja de ser otro problema de regresión en el que se consideran dos espirales de puntos y donde hay que identificar a qué espiral pertenece cada uno de los puntos.

4

Modelo monobjetivo de AP para clasificación multiclase

Tras el estudio realizado de las propuestas desarrolladas para abordar la tarea de clasificación de DM, de una parte, y de la revisión efectuada del paradigma de AP, concluimos que aún no ha sido explorada la aplicación de AP a dicha tarea, a pesar de que a priori las expectativas en cuanto a su rendimiento son, cuanto menos, interesantes. Estos resultados son de esperar dadas las buenas soluciones que demuestran alcanzar clasificadores basados en la metaheurística de ACO y por aquellos basados en GP.

Así pues, en este capítulo se presenta el algoritmo **GBAP** (*Grammar-Based Ant Programming*), un algoritmo de AP basado en gramática que ha sido el primero de este paradigma en abordar la tarea de clasificación de DM.

GBAP extrae reglas de clasificación fácilmente comprensibles y que pueden ayudar a expertos en la toma de decisiones. Está guiado por una CFG que garantiza la creación de individuos válidos. Para calcular la probabilidad de transición a cada estado disponible, el algoritmo aplica dos funciones heurísticas complementarias, a diferencia de los algoritmos clásicos de ACO que emplean una sola. La selección

del consecuente de la regla y la selección de las reglas que conforman el clasificador final se realiza siguiendo un enfoque de nichos.

El rendimiento del algoritmo GBAP se compara frente a otras técnicas de clasificación basadas en reglas sobre 18 conjuntos de datos de características y dimensionalidad variadas. Los resultados demuestran que el algoritmo propuesto es capaz de extraer reglas comprensibles y valores de exactitud predictiva competitivos e incluso superiores al resto de algoritmos considerados.

4.1. Entorno y codificación de los individuos

Para asegurar que los individuos generados codifican programas válidos, Koza introdujo el requerimiento de cierre en GP, que también se aplica en la metaheurística de AP. A pesar de tratarse de un concepto simple y uniforme, su uso plantea dos inconvenientes, principalmente. En primer lugar, dado que toda función y terminal puede combinarse con cualquier otro, y a que algunas de esas combinaciones no tienen ningún sentido conceptualmente, ésto produce un incremento innecesario y no deseable en el tamaño del espacio de búsqueda. Y, en segundo lugar, la propiedad de cierre es difícil de satisfacer (y en ciertas ocasiones incluso imposible) en aquellos problemas que requieren tipos de datos diferentes.

El algoritmo GBAP hace uso de una **gramática** para representar el antecedente de la regla codificada por cada individuo. Se trata de una CFG expresada en notación Backus-Naur form (BNF), y que viene definida por $G = (\Sigma_N, \Sigma_T, P, < EXP >)$, donde Σ_N es el conjunto de no terminales, Σ_T es el conjunto de terminales, P es el conjunto de reglas de producción, y $< EXP >$ es el símbolo inicial de la gramática.

El conjunto de símbolos no terminales es aquél que expresa conceptos, y eventualmente son remplazados por los símbolos terminales para ser funcionales. Además, un no terminal puede ser remplazado por otro no terminal. Por su parte, los símbolos terminales denotan un significado concreto. Las reglas de producción definen cómo los no terminales se relacionan con los terminales y otros no terminales.

La gramática empleada por GBAP se expresa como:

G = (Σ_N, Σ_T, P, <EXP>)

Σ_N = {<EXP>, <COND>}

Σ_T = {AND, =, !=

attr$_1$, attr$_2$, ..., attr$_n$,

value$_{11}$, value$_{12}$, ..., value$_{1m}$,

value$_{21}$, value$_{22}$, ..., value$_{2m}$,

..., value$_{n1}$, value$_{n2}$, ..., value$_{nm}$}

P = {<EXP>:= <COND>| AND <EXP><COND>,

<COND>:= todas las posibles combinaciones de la terna

operator attr value}

Cualquier regla de producción consta de una parte izquierda y una parte derecha. La parte izquierda siempre hace referencia al símbolo no terminal que se puede remplazar con la parte derecha de la regla, y que estará compuesta por una combinación de símbolos terminales y no terminales. Esta acción se denomina derivación inmediata. Matemáticamente, dada una CFG definida por $G = (\Sigma_N, \Sigma_T, P, S)$, y donde $A \in \Sigma_N$ y $\delta, \eta \in \{\Sigma_N \cup \Sigma_T\}$ y siendo $A \to \beta$ una regla de producción de la gramática, entonces una derivación inmediata se expresa como sigue:

$$\delta A \eta \underset{G}{\Longrightarrow} \delta \beta \eta \tag{4.1}$$

En GBAP las reglas de producción se expresan en notación prefija y se derivan siempre por la izquierda. Las derivaciones por la izquierda siempre aplican una regla asociada al símbolo no terminal situado más a la izquierda. Formalmente, una derivación $\alpha_0, \alpha_1, \cdots, \alpha_n (n \geq 0)$, es por la izquierda si

$$\forall i \in \{0, \cdots, n-1\} \exists A_i \to \beta \in P / \alpha_i = x A_i \eta \underset{I}{\Longrightarrow} x \beta_i \eta = \alpha_{i+1} \tag{4.2}$$

donde $x \in \Sigma_T$, $A_i \in \Sigma_N$ y $\eta, \beta \in \{\Sigma_N \cup \Sigma_T\}$. Así pues, dado un estado i, cualquier transición de dicho estado a otro j estará asociada a la aplicación de una determinada regla de producción sobre el primer símbolo no terminal del estado i.

Los sistemas de programación automática guiados por gramática, al igual que ocurre en la GP y la AP, también emplean la nomenclatura de terminales y no terminales, pero en este caso hacen referencia a los símbolos de la gramática, en lugar de a los nodos hoja y funciones en el caso de la representación de árbol. La diferencia que encontramos es que en la G3P la gramática controla la creación de la población inicial de individuos y las operaciones de cruce, mutación y reproducción; en cambio, en la AP basada en gramática no existen operadores de ningún tipo, por lo que aquí **la gramática controla los movimientos de las hormigas**, siendo responsable de que estas sigan un camino válido y puedan encontrar una solución factible al problema.

En cualquier algoritmo inspirado en hormigas es necesario especificar un **entorno** donde estas puedan cooperar unas con otras. En GBAP, este entorno consiste en el espacio de búsqueda que comprende todas las posibles expresiones o programas que se pueden derivar a partir de la gramática en el número de derivaciones disponibles. El entorno adopta la forma de un **árbol de derivación**, como se muestra en la Figura 4.1 de ejemplo a una profundidad de tres derivaciones. Por tanto, este número de derivaciones permitidas para la gramática es el que nos va a permitir conocer la profundidad del árbol de derivación.

Cada hormiga trata de construir una solución factible al problema partiendo del estado inicial, que se corresponde con el símbolo inicial de la gramática. Cualquier solución encontrada toma la forma de un **camino desde el nodo raíz a un estado final** del árbol de derivación, como se ejemplifica con el camino coloreado de la Figura 4.1. El camino se compone de una secuencia de estados, donde cada paso entre estados viene dado por un **paso de derivación**, es decir, la aplicación de una de las reglas de producción aplicables al estado origen. Los estados finales, representados en la figura por un doble óvalo, únicamente pueden contener símbolos terminales, y representan una expresión evaluable del antecedente de la regla que codifican. A pesar de que los estados finales codifican un antecedente evaluable, cumpliendo con las propiedades de las hormigas artificiales [154], cada individuo

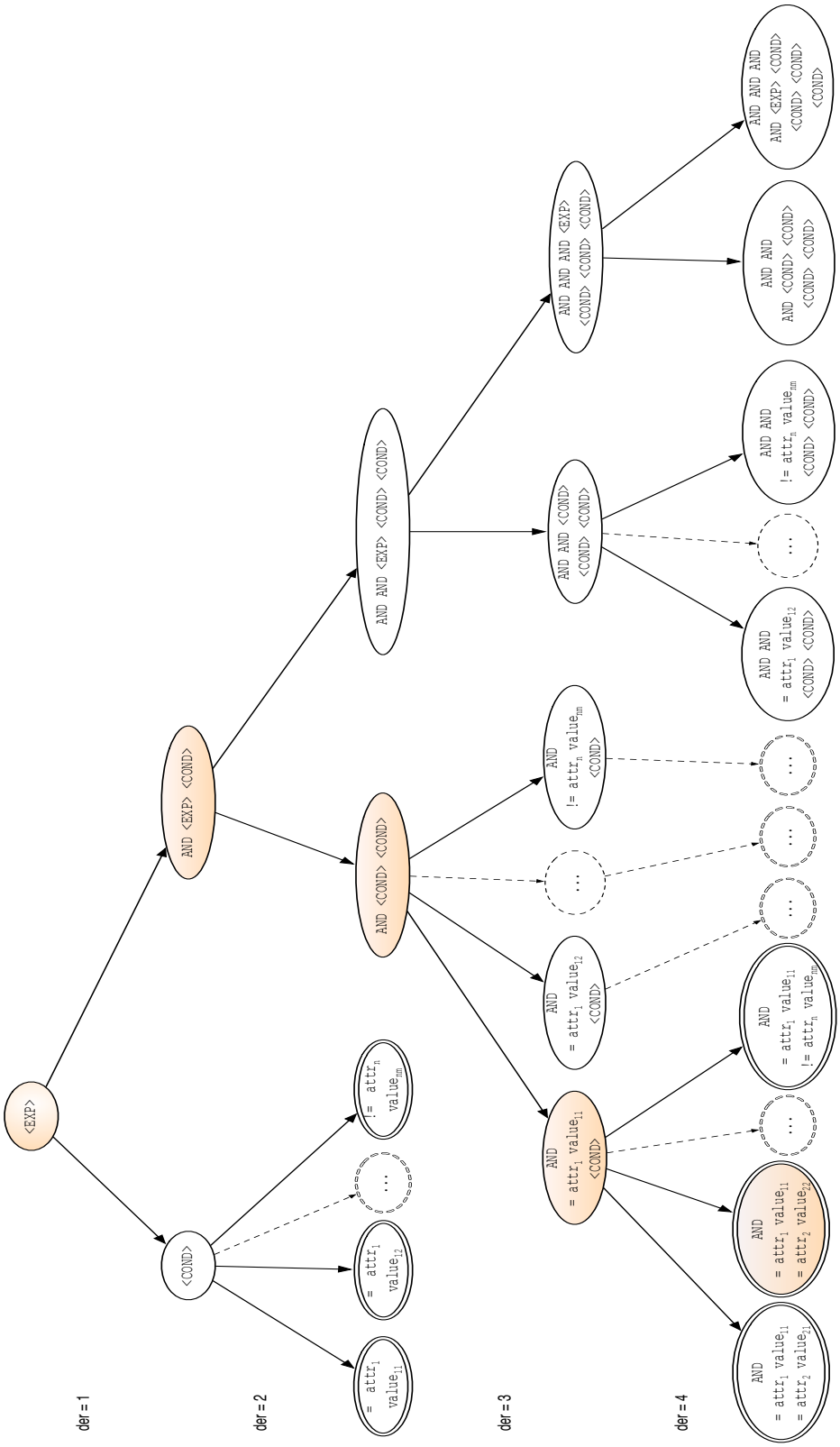


Figura 4.1: Espacio de estados a una profundidad de 4 derivaciones.

posee una **memoria interna** para almacenar el camino que ha seguido, con lo que se posibilita la realización de una actualización de feromonas fuera de línea.

El tratamiento del entorno es un punto clave del algoritmo. De hecho, dependiendo tanto de la dimensionalidad del conjunto de datos abordado como del número de derivaciones que se permiten para la gramática, puede suponer un coste computacional excesivo mantener en memoria el espacio de estados completo. Por ello, en lo concerniente a la generación del espacio de estados, GBAP sigue un planteamiento de **construcción incremental**. Así, al comienzo del algoritmo, la estructura de datos que codifica el entorno contendrá sólo el estado inicial, y todas las posibles transiciones poseen la misma cantidad de feromonas. La estructura de datos se va rellenando conforme se crean las hormigas, insertando los estados visitados por las hormigas. Esta estructura de datos también contiene atributos que permiten llevar la cuenta del efecto de los procesos de normalización y evaporación de feromonas sobre el entorno y, especialmente, sobre los estados que aún no han sido visitados.

Con respecto a la codificación de los individuos, GBAP sigue el **enfoque** *individuo* = *regla* [70]. Como se ha mencionado previamente, cuando las hormigas se crean tan sólo representan el antecedente de la nueva regla. El consecuente será asignado posteriormente en la ejecución del procedimiento de nichos, tal como se describe en la Sección 6.2.4.

4.2. Heurística

Otro factor diferenciador de GBAP con respecto a los algoritmos de ACO estriba en el uso de una **función heurística con dos componentes complementarios** que no se pueden aplicar simultáneamente en una transición. Para distinguir cuál aplicar, GBAP necesita averiguar de qué tipo de transición se trata, existiendo dos posibles tipos: **transiciones intermedias**, que son aquellas que no conllevan la aplicación de reglas de producción que implican la selección de atributos del dominio del problema; y **transiciones finales**, que suponen lo contrario, es decir, la aplicación de reglas de producción del tipo `COND = 'operator', 'attribute', 'value';`

Para el caso de las transiciones intermedias, la función heurística que se aplicaría está vinculada a la **cardinalidad de las reglas de producción**, y se denota como P_{card}. Este componente aumenta la probabilidad de que una hormiga seleccione como movimiento siguiente aquellas transiciones que permiten ampliar el abanico de soluciones candidatas. Se basa en la medida de cardinalidad propuesta por *Geyer-Schulz* [93] y, para poder computarla, es necesario que el algoritmo calcule una tabla de cardinalidad por cada regla de producción existente en la gramática, tarea que lleva a cabo cuando inicializa la misma. La tabla de cardinalidad para una regla concreta contendrá tantas entradas como número de derivaciones haya permitidas en la ejecución del algoritmo, donde cada entrada mapea el número de derivación al número de soluciones que pueden ser alcanzadas a esa profundidad. Así pues, dado un estado origen i con k posibles estados siguientes, siendo j un sucesor específico entre dichos k estados, y restando aún d derivaciones disponibles en ese punto, $P_{card_{ij}^k}$ se calcula como el ratio entre el número de soluciones candidatas que una hormiga puede alcanzar trasladándose al estado j en *d-1* derivaciones, y la suma de todas las posibles soluciones a las que se puede llegar desde el estado fuente i, tal como se expresa en la Ecuación 4.3.

$$P_{card_{ij}^k} = \frac{cardinality(state_j, d-1)}{\Sigma_{k \in allowed}(cardinality(state_k, d-1))} \tag{4.3}$$

Hay que recalcar que la aplicación de P_{card} en las transiciones finales no tendría sentido, dado que cada posible nodo destino englobaría el mismo número de soluciones, una, con lo que la aplicación de P_{card} sería equiprobable en cada uno de los posibles nodos siguientes.

En las transiciones finales se aplica la otra componente de la función heurística, la **ganancia de información** [179], denotada como $G(A_i)$, y que mide la fuerza de cada atributo para separar los ejemplos de entrenamiento con respecto al objeto de clasificación. La ganancia de información se calcula como:

$$G(A_i) = I - I(A_i) \tag{4.4}$$

donde:

I representa la entropía de las clases en el conjunto de entrenamiento, y

$I(A_i)$ es la entropía de las clases dado los valores del atributo A_i.

A su vez, estas se calculan como sigue:

$$I = -\sum_{c=1}^{\#classes} (\frac{n_c}{n} \cdot log_2 \frac{n_c}{n}) \tag{4.5}$$

donde:

n_c se refiere al número de instancias de la clase c, y

n indica el número de instancias en el conjunto de entrenamiento.

$$I(A_i) = \sum_{j=1}^{\#valuesA_i} (\frac{n_{ij}}{n} \cdot I(A_{ij})) \tag{4.6}$$

donde:

n_{ij} es igual al número de instancias con el valor j para el atributo A_i, y

$I(A_{ij})$ es la entropía de las clases dado el valor j para el atributo A_i, que se calcula como:

$$I(A_{ij}) = -\sum_{c=1}^{\#classes} (\frac{n_{ijc}}{n_{ij}} \cdot log_2 \frac{n_{ijc}}{n_{ij}}) \tag{4.7}$$

donde:

n_{ijc} indica el número de instancias de la clase c con valor j en el atributo A_i.

4.3. Regla de transición

La metaheurística de ACO sigue un método constructivo en el cual cada solución se construye de acuerdo a una secuencia de transiciones entre estados guiadas

por cierta información. Dado que el paradigma de AP utiliza ACO como técnica de búsqueda, dicho enfoque se mantiene. La información que afecta a cada paso se considera en la regla de transición, que calcula la probabilidad con que una determinada hormiga se moverá de un estado i a otro j:

$$P_{ij}^{k} = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\Sigma_{k \in allowed}(\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \tag{4.8}$$

donde:

- k es el número de posibles estados siguientes,
- α es el exponente de la función heurística,
- β es el exponente del nivel de feromonas,
- η es el valor de la función heurística, calculada como $G(A_i) + P_{card}$ (siendo siempre uno de los dos componentes igual a cero), y
- τ indica la fuerza del rastro de feromonas en la transición.

La regla de transición asignará una probabilidad a cada estado siguiente disponible. El algoritmo controla que la transición de una hormiga a cada uno de ellos permita alcanzar al menos un estado final o solución en el número de derivaciones que quedan disponibles en ese punto. En caso contrario, asignará a dichos estados una probabilidad de cero, con lo que se impide a la hormiga seleccionar estos movimientos.

4.4. Actualización de feromonas

En lo concerniente al mantenimiento de los niveles de feromonas en el espacio de estados, se consideran dos operaciones: refuerzo y evaporación. Aquellas hormigas de la generación actual cuya solución tenga una calidad superior a un determinado valor umbral pueden reforzar los niveles de feromonas. Para ello incrementan en la misma cantidad de feromonas todas las transiciones que han seguido en su camino hasta hallar la solución que codifican, como se muestra en la Ecuación 4.9. El valor umbral establecido ha sido fijado a 0.5 experimentalmente, de manera que las

hormigas que representan soluciones de poca calidad no influyan negativamente en el entorno.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot Q \cdot fitness \tag{4.9}$$

donde:

- τ_{ij} representa la cantidad de feromonas en la transición del estado i al j, y
- Q es una medida calculada para favorecer las soluciones comprensibles. De hecho, su valor depende de la longitud de la solución codificada por la hormiga, y se calcula como el ratio entre el número máximo de derivaciones permitidas en la generación actual y la longitud del camino seguido por la hormiga. Así, las soluciones más cortas recibirán un mayor refuerzo.

A diferencia del proceso de refuerzo, que sólo tiene lugar cuando la calidad de la hormiga excede el valor umbral, la evaporación tiene lugar siempre, y sobre todas las transiciones del espacio de estados, tal como se indica en la Ecuación 7.5.

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1-\rho) \tag{4.10}$$

donde:

- ρ indica la tasa de evaporación.

Una vez que los rastros de feromonas del entorno se han visto sometidos a los procesos de refuerzo y evaporación, se lleva a cabo una normalización de los valores en los mismos con el objetivo de limitar la cantidad de feromonas existente en cada transición al rango $[\tau_{min}, \tau_{max}]$.

Hay que resaltar la complejidad inherente al almacenamiento de los valores de feromonas del entorno en una matriz de feromonas, debido a la forma en árbol de derivación que adopta el mismo y a que no se conoce de antemano su profundidad, la cual depende del número de derivaciones permitidas para la gramática. En lugar de emplear una matriz, el valor de feromonas asociado a una determinada transición se almacena en un campo que posee a tal efecto la estructura de datos que se corresponde con el estado destino de dicha transición.

4.5. Algoritmo

La secuencia principal de GBAP se detalla en el pseudocódigo del Algoritmo 1. El primer paso consiste en la inicialización de la gramática y del espacio de estados. Para ello, por un lado se crea una tabla de cardinalidad por cada regla de producción y, por otro, se inserta el nodo que representa el estado inicial en el espacio de estados. El algoritmo también crea un objeto vacío que representa al clasificador final, el cual contendrá las hormigas que queden (ganen) en la competición que se lleva a cabo en cada generación en el procedimiento de nichos.

En la primera iteración del algoritmo, los individuos buscarán codificar reglas con el mínimo número de derivaciones que, dada la gramática, permiten alcanzar estados finales o soluciones. Por tanto, previamente a la entrada en el bucle generacional, el número máximo de derivaciones se establece a dos. También se calcula el paso de derivación para cada generación, que incrementará la variable $maxDerivations$ para ir extendiendo la búsqueda hacia soluciones más complejas.

Una nueva lista de hormigas se inicializa al comienzo de cada generación. El algoritmo rellena esta lista creando el número de hormigas que el usuario especifica con el parámetro $numAnts$. Los estados que no habían sido visitados previamente por otras hormigas y que sí son visitados por las hormigas recién creadas se almacenan en la estructura que representa el espacio de estados. A continuación, el algoritmo calcula k valores de $fitness$ por hormiga, donde k es el número de clases del conjunto de datos. Nótese que en este punto aún no se ha procedido a la asignación del consecuente, por lo que cada hormiga únicamente codifica el antecedente de una regla.

Una vez que todas las hormigas han sido creadas, estas hormigas junto con aquellas asignadas al clasificador en la generación previa compiten en el algoritmo de nichos. En él, las hormigas tratan de capturar tantas instancias del conjunto de entrenamiento como pueden, tal como se explica en la Sección 4.5.1. Una vez hecho esto, se asigna el consecuente idóneo para cada hormiga. Al concluir el algoritmo de nichos, las hormigas vencedoras se asignan al clasificador, remplazando a las que contenía de la generación anterior.

A continuación, cada hormiga creada en la presente generación del algoritmo refuerza la cantidad de feromonas en las transiciones seguidas sólo si su $fitness$ es

Algoritmo 1 Pseudocódigo de GBAP

Require: $trainingSet, testSet, numGenerations, numAnts, maxDerivations$

```
Inicializar gramática a partir de trainingSet y espacio de estados
Inicializar clasificador
derivationStep ← (maxDerivations−2) / numGenerations
maxDerivations ← 2
for i = 0 to i = numGenerations inc 1 do
    Crear lista ants ← {}
    for j = 0 to j = numAnts inc 1 do
        ant ← Crear nueva hormiga (ver Procedimiento 2)
        Almacenar los estados del camino seguido por ant en el espacio de estados
        Evaluar ant, calculando k fitness, uno por cada clase disponible en el conjunto
        de datos, suponiendo que en su consecuente vaya a predecir dicha clase
        Añadir ant a la lista ants
    end for
    Enfoque de nichos para asignar el consecuente a las hormigas y establecer las reglas
    del clasificador (ver Procedimiento 3)
    for each ant in ants do
        if fitness > threshold then
            Actualizar la tasa de feromonas en el camino seguido por ant de manera
            proporcional a su fitness e inversamente proporcional a la longitud de la
            solución que codifica
        end if
    end for
    Evaporar las feromonas en el espacio de estados
    Normalizar los valores de feromonas
    maxDerivations ← maxDerivations + derivationStep
end for
Establecer la regla por defecto en classifier
predictiveAccuracy ← Calcular la exactitud predictiva obtenida al ejecutar sobre
testSet el clasificador construido
return predictiveAccuracy
```

superior al valor umbral establecido. Para completar la generación, los procesos de evaporación y normalización tienen lugar. El número máximo de derivaciones también se incrementa en el valor indicado por *derivationStep*.

Tras finalizar todas las generaciones, la regla por defecto se añade al clasificador y éste se ejecuta sobre el conjunto de *test*, calculando la exactitud predictiva.

El proceso de creación de una determinada hormiga se describe en el Procedimiento 2. En primer lugar, el algoritmo inicializa una nueva lista que representará el camino seguido por la hormiga, y donde se almacenarán, por tanto, aquellos nodos o estados por los que pase la nueva hormiga. Acto seguido crea un nuevo nodo n

Procedimiento 2 Creación de hormigas

Require: $maxDerivations$
1: Crear lista $path \leftarrow \{\}$
2: $n \leftarrow$ Estado inicial
3: Añadir n a la lista $path$
4: **repeat**
5: $maxDerivations \leftarrow maxDerivations - 1$
6: $n \leftarrow$ Escoger siguiente movimiento en espacio de estados, siendo n el nodo origen y $maxDerivations$ el número de derivaciones que quedan disponibles
7: Añadir n a la lista $path$
8: **until** (n es un nodo final)
9: $ant \leftarrow$ Nueva hormiga con su camino igual a $path$
10: **return** ant

que se corresponde con el estado inicial del entorno, y lo añade a la lista. Siguiendo un proceso secuencial, el bucle principal del algoritmo se encarga de seleccionar el siguiente movimiento de la hormiga a partir del estado actual, decrementando en uno el número de derivaciones que quedan disponibles. También añade el nuevo estado a la lista *path*. Finaliza cuando el movimiento escogido lleva a la hormiga a un estado final, lo cual indica que se ha alcanzado una solución. Finalmente, la hormiga se crea a partir de la lista de estados visitados *path*.

4.5.1. *Función de* fitness

Dado que el algoritmo GBAP puede abordar tanto problemas de clasificación binarios como multiclase, a la hora de determinar la aptitud de los individuos resulta conveniente tener en cuenta el número de clases que posee el conjunto de datos.

La función de *fitness* que GBAP utiliza durante la fase de entrenamiento para conducir el proceso de búsqueda es la **exactitud de Laplace** [49], que se se define como:

$$fitness = \frac{1 + T_P}{k + T_P + F_P} \tag{4.11}$$

donde T_P y F_P representan los verdaderos positivos y falsos positivos, respectivamente, y k es igual al número de clases del conjunto de datos.

4.5.2. Asignación del consecuente

Con respecto a la asignación del consecuente, GBAP sigue un **enfoque de nichos** que se asemeja en cierta medida al empleado en la propuesta de *Berlanga et al.* [24], y cuyo propósito es evolucionar diferentes reglas para predecir cada clase en el conjunto de datos manteniendo la diversidad de la población.

> A veces, debido a la distribución que presenta el conjunto de datos, no es posible que una única regla cubra todos los ejemplos de su clase, por lo que es necesario descubrir reglas adicionales para predecir dicha clase. La estrategia de nichos se ocupa de seleccionar las reglas para clasificar una determinada categoría sin que se solapen con instancias de otras clases.

Por otro lado, el algoritmo de nichos resulta apropiado para desechar las reglas que son redundantes. Además carece de las desventajas que presentan los algoritmos de cubrimiento secuencial en lo referente al descarte de instancias: las instancias que cubren por las reglas descubiertas hasta el momento son eliminadas del conjunto de entrenamiento y, por ende, el $fitness$ de las reglas descubiertas a posteriori se calcula empleando un menor número de instancias. Por todo ello, se suele decir que los algoritmos de cubrimiento secuencial se centran en buscar reglas con un buen rendimiento en el *sub*conjunto de datos.

El pseudocódigo del algoritmo de nichos implementado se muestra en el Procedimiento 3. Cada instancia existente en el conjunto de entrenamiento se considera un recurso, denominado *token* en este contexto. Todas las hormigas de la población competirán para capturar los *tokens*, teniendo en cuenta que cada *token* sólo puede ser capturado como máximo por un individuo. Por cada hormiga se han calculado previamente k posibles $fitness$, uno por cada clase (asumiendo que la clase respectiva se asigna como consecuente al individuo). El algoritmo de nichos repite el siguiente procedimiento por cada clase del conjunto de entrenamiento, siguiendo el orden en que aparecen en la sección de metadatos:

Procedimiento 3 Algoritmo de nichos de GBAP

Require: $ants, classifier, minCasesPerRule, trainingSet$

Crear lista $winnerAnts \leftarrow \{\}$
$numInstances \leftarrow$ número de instancias en $trainingSet$
$numClasses \leftarrow$ número de clases en $trainingSet$
for $k \leftarrow 0$ **to** $k < numClasses$ **inc** 1 **do**
  Añadir las hormigas de $classifier$ a la lista $ants$
  Ordenar $ants$ por el $fitness$ correspondiente a la clase k
  $flagsArray \leftarrow$ array binario de $numInstances$ elementos
  for $i \leftarrow 0$ **to** $i < |flagsArray|$ **inc** 1 **do**
    $flagsArray[i] \leftarrow$ **false**
  end for
  for each ant **in** $ants$ **do**
    $idealTokens \leftarrow 0$
    $capturedTokens \leftarrow 0$
    for $j \leftarrow 0$ **to** $j < numInstances$ **inc** 1 **do**
      $instance \leftarrow$ Instancia j de $trainingSet$
      $cat \leftarrow$ Categoría de $instance$
      if $cat = k$ **then**
        $idealTokens \leftarrow idealTokens + 1$
        if $flagsArray[j] =$ **false and** ant cubre $instance$ **then**
          $flagsArray[j] \leftarrow$ **true**
          $capturedTokens \leftarrow capturedTokens + 1$
        end if
      end if
    end for
    if $capturedTokens >= minCasesPerRule$ **then**
      $fitness_{adj}[k] \leftarrow \frac{capturedTokens}{idealTokens}$
    else
      $fitness_{adj}[k] \leftarrow 0$
    end if
  end for
end for
for each ant **in** $ants$ **do**
  $max \leftarrow 0$
  for $k \leftarrow 0$ **to** $k < numClasses$ **inc** 1 **do**
    if $fitness_{adj}[k] > fitness_{adj}[max]$ **then**
      $max \leftarrow k$
    end if
  end for
  Establecer $consequent$ de ant igual a clase max
  if $fitness_{adj}[max] > 0$ **then**
    Añadir ant a $winnerAnts$
  end if
end for
return $winnerAnts$

1. Los individuos se preparan para competir, ordenándose de mayor a menor fortaleza según su valor de $fitness$ prediciendo la clase actual. De esta manera, los individuos más fuertes tendrán más posibilidades de capturar *tokens*.

2. Cada uno de los individuos, secuencialmente y siguiendo el orden establecido, tratan de capturar cada uno de los *tokens* del conjunto de entrenamiento correspondientes a la clase actual, si es que no han sido capturados previamente. Para ello, cada instancia tiene asociada una bandera que se pone a *true* cuando un individuo logra capturarla.

3. Si la regla codificada por cada hormiga ha sido capaz de capturar el número de *tokens* establecido en el parámetro *minCasesPerRule*, se calcula su valor de *fitness* penalizado o ajustado para esta clase, tal como se indica en la Ecuación 4.12. En otro caso, el valor de $fitness_{adj}$ del individuo para esta clase será de cero.

$$fitness_{adj} = fitness \cdot \frac{\#capturedTokens}{\#idealTokens} \tag{4.12}$$

donde:

$capturedTokens$	indica el número ideal de *tokens* capturados, y
$idealTokens$	representa el número ideal de *tokens* que la hormiga podría capturar, es decir, el número de instancias que pertenecen a la clase actual (sin importar que otras hormigas las hayan capturado previamente o no)

Hay que resaltar que el número de $idealTokens$ es siempre mayor o igual que el de $capturedTokens$. Por tanto, cuanto más cercanos sean los valores de ambos, menor penalización tendrá la hormiga. De hecho, si $capturedTokens = idealTokens$, la hormiga no sufre penalización alguna, con lo que su $fitness_{adj}$ para esta clase será equivalente al que tenía antes de entrar en la competición de nichos.

Una vez se han calculado los k valores de fitness ajustados para cada hormiga, el consecuente que se le asignará será aquel correspondiente al mejor valor de *fitness* ajustado. Para finalizar, los individuos que tienen un valor de *fitness* ajustado superior a cero y, por tanto, cubren al menos una instancia del conjunto de entrenamiento, se añaden al clasificador.

4.6. Experimentación

En esta sección se introducen, en primer lugar, los conjuntos de datos empleados en el estudio experimental, junto con las acciones de preprocesamiento realizadas sobre los mismos. A continuación, se describe el proceso de validación cruzada empleado y la forma de calcular el rendimiento de los clasificadores. Por último, se listan los algoritmos incluidos en la comparativa, indicando la configuración de parámetros utilizada en cada uno, y se presenta asimismo un análisis de sensibilidad de los parámetros del algoritmo de AP propuesto.

4.6.1. Conjuntos de datos y preprocesado

El rendimiento de GBAP ha sido estudiado sobre 18 conjuntos de datos, tanto reales como artificiales, disponibles públicamente en el repositorio de ML de la universidad de California en Irvine (UCI) [82] [1]. Los conjuntos de datos seleccionados presentan dimensionalidad variada con respecto al número de instancias, atributos y clases. El listado de los mismos junto con sus características se proporciona en la Tabla 5.1.

Debido al hecho de que algunos de los conjuntos de datos considerados contienen valores perdidos y atributos numéricos, se han realizado dos acciones de preprocesamiento utilizando Weka[2]. La primera acción supuso el remplazo de los valores perdidos por la moda, en el caso de atributos nominales, o la media aritmética, para los numéricos. La otra tarea implicó la discretización de los conjuntos de datos con

[1] Todos los conjuntos de datos se pueden descargar de la página web de la UCI, http://archive.ics.uci.edu/ml/datasets.html

[2] El software de ML Weka está disponible públicamente en http://www.cs.waikato.ac.nz/ml/index.html

Conjunto de Datos	Valores Perdidos	Instancias	Atributos			Clases	Distribución de Clases
			Cont.	*Bin.*	*Nom.*		
Hepatitis	yes	155	6	13	0	2	32 / 123
Sonar	no	208	60	0	0	2	97 / 111
Breast-c	sí	286	0	3	6	2	201 / 85
Heart-c	sí	303	6	3	4	2	165 / 138
Ionosphere	no	351	33	1	0	2	126 / 225
Horse-c	sí	368	7	2	13	2	232 / 136
Australian	sí	690	6	0	9	2	307 / 383
Breast-w	sí	699	9	0	0	2	458 / 241
Diabetes	no	768	0	8	0	2	500 / 268
Credit-g	no	1000	6	3	11	2	700 / 300
Mushroom	sí	8124	0	0	22	2	4208 / 3916
Iris	no	150	4	0	0	3	50 / 50 / 50
Wine	no	178	13	0	0	3	59 / 71 / 48
Balance-scale	no	625	4	0	0	3	288 / 49 / 288
Lymphography	no	148	3	9	6	4	2 / 81 / 61 / 4
Glass	no	214	9	0	0	6	70 / 76 / 17 / 13 / 9 / 29
Zoo	no	101	1	15	0	7	41 / 20 / 5 / 13 / 4 / 8 / 10
Primary-tumor	sí	339	0	14	3	21	84 / 20 / 9 / 14 / 39 / 1 / 14 / 6 / 2 / 28 / 16 / 7 / 24 / 2 / 1 / 10 / 29 / 6 / 2 / 1 / 24

Tabla 4.1: Características de los conjuntos de datos

atributos numéricos, aplicando el algoritmo de discretización de Fayyad e Irani [73], cuyos fundamnetos se explican brevemente en el Anexo A.3. Hay que mencionar que la sustitución de los valores perdidos se realizó previamente al particionado del conjunto de datos. Por su parte, la discretización se aplicó en un primer momento sobre cada conjunto de entrenamiento específico, para pasar a discretizar a continuación el conjunto de test correspondiente a cada uno los mismos empleando los intervalos encontrados en el primer paso.

4.6.2. Validación cruzada

Para evaluar el rendimiento de un algoritmo particular sobre cada conjunto de datos se ha realizado un procedimiento de **validación cruzada estratificada con 10 particiones (*10-fold cross validation*)**. En él, dividimos al azar cada conjunto de datos en diez particiones mutuamente excluyentes, P_1,...,P_{10}, cada una manteniendo la distribución de instancias por clase del conjunto de datos original. A continuación, diez ejecuciones del algoritmo son lanzadas utilizando $\bigcup_{j \neq i} P_j$ como conjunto de *training* en el experimento i-ésimo y P_i como conjunto de *test*. Por tanto, la exactitud predictiva obtenida sobre un determinado conjunto de datos por un algoritmo se calcula como la media de los valores de exactitud obtenidos sobre las diez particiones, tal como se indica en la Ecuación 4.13.

$$predAcc = \frac{\Sigma_{i=1}^{10}(\#correctlyClassifiedP_i)}{\#instances} \cdot 100 \tag{4.13}$$

donde:

$\#correctlyClassifiedP_i$	indica el número de instancias correctamente clasificadas utilizando P_i como conjunto de *test*, y
$\#instances$	representa el número de instancias del conjunto de datos original.

Además, en los algoritmos con componentes estocásticos, con el objetivo de evitar la posible influencia en los resultados que puede conllevar la utilización de una determinada semilla en su ejecución, se llevaron a cabo diez ejecuciones por cada partición, utilizando en cada una una semilla de números aleatorios diferente.

4.6.3. Algoritmos y configuración de parámetros

En el estudio experimental se han considerado otros seis algoritmos de inducción de reglas de clasificación: tres basados en la metaheurística de ACO, Ant-Miner [3], Ant-Miner+ [4] y PSO/ACO2 [5], los cuales fueron introducidos en la Sección 3.3.2; un algoritmo de GP, Bojarczuk-GP [27] [6], explicado con anterioridad en la Sección 3.2.1; y dos clasificadores clásicos, JRIP (la implementación de Weka del algoritmo de cubrimiento secuencial RIPPER), y PART, el cual extrae reglas a partir del árbol de decisión generado por el algoritmo J48 de Weka. Es importante mencionar en este punto que cada algoritmo utilizado en la experimentación fue ejecutado sobre las mismas particiones discretizadas mencionadas previamente, incluso aquellos capaces de tratar valores numéricos directamente.

Los valores empleados para los parámetros que presentan estos algoritmos fueron aquellos reportados como óptimos por sus autores en las referencias correspondientes. Dicha configuración de parámetros queda recogido en la Tabla 4.2, donde también se indica la configuración para el algoritmo GBAP.

A simple vista puede parecer que GBAP tiene más parámetros que el resto de algoritmos de ACO, lo cual iría en detrimento de la usabilidad. No obstante, esto se debe al hecho de que los otros algoritmos de hormigas poseen parámetros similares pero ocultos para el usuario final, con un valor prefijado de antemano. Por ejemplo, en el artículo donde se propuso el algoritmo Ant-Miner+ [147], los autores mencionan los parámetros α, β, *early stopping criterion* (criterio de parada rápida), así como parámetros que son implícitos al enfoque MMAS que sigue este algoritmo, τ_0, τ_{min} and τ_{max}. Sin embargo, los autores establecieron el valor de dichos parámetros en el código del algoritmo, no permitiendo su modificación al usuario final. En el caso del algoritmo GBAP, procediendo de un modo similar se podría haber limitado el número de parámetros configurables por el usuario a cuatro $numAnts$, $numGenerations$, $maxDerivations$ y $minCasesPerRule$, fijando en el código el

[3] Ant-Miner fue ejecutado utilizando el *framework* Myra (version 2.1.2), que se puede descargar de http://myra.sourceforge.net/

[4] El código fuente de Ant-Miner+ fue facilitado por sus autores

[5] El algoritmo PSO/ACO2 fue ejecutado utilizando la implementación disponible en http://sourceforge.net/projects/psoaco2

[6] Bojarczuk-GP fue ejecutado empleando la implementación existente en el *framework* de computación evolutiva JCLEC [224], disponible públicamente en http://jclec.sourceforge.net

Algoritmo	Parámetro	Descripción	Valor
GBAP	numAnts	Número de hormigas	20
	numGenerations	Número de generaciones	100
	maxDerivations	Número máximo de derivaciones para la gramática	15
	minCasesPerRule	Número mínimo de instancias cubiertas por regla	3
	$[\tau_0]$	*Cantidad de feromonas inicial en las transiciones*	*1,0*
	$[\tau_{min}]$	*Cantidad mínima de feromonas permitida*	*0,1*
	$[\tau_{max}]$	*Cantidad máxima de feromonas permitida*	*1,0*
	$[\rho]$	*Tasa de evaporación*	*0,05*
	$[\alpha]$	*Exponente para la función heurística*	*0,4*
	$[\beta]$	*Exponente para las feromonas*	*1,0*
Ant-Miner	number of ants	Número de hormigas	1
	min. cases per rule	Número mínimo de instancias cubiertas por regla	10
	max. uncovered cases	Número máximo de instancias no cubiertas	10
	rules for convergence	Número de reglas para convergencia	10
	number of iterations	Número máximo de iteraciones	3000
Ant-Miner+	nAnts	Número de hormigas	1000
	rho	Tasa de evaporación	0,85
PSO/ACO2	numParticles	Número de partículas	10
	numIterations	Número de iteraciones	200
	maxUncovExampPerClass	Número máximo de ejemplos no cubiertos por clase	2
GP	population-size	Número de individuos	200
	max-of-generations	Número de generaciones	100
	max-deriv-size	Número máximo de derivaciones para la gramática	20
	recombination-prob	Probabilidad de cruce	0,8
	reproduction-prob	Probabilidad de reproducción	0,05
	parents-selector	Método de selección de padres	Roulette
JRIP	checkErrorRate	Si se incluye comprobación para tasa de error >= 1/2 en criterio de parada	True
	folds	Determina las particiones usadas para la poda. Una se usa para poda, el resto para hacer crecer las reglas	3
	minNo	Mínimo de peso total de las instancias en una regla	2,0
	optimizations	Número de ejecuciones para optimización	2
	pruning	Si se realiza poda	True
PART	binarySplits	Indica si se usan particiones binarias en atributos nominales al construir los árboles parciales	False
	confidenceFactor	Factor de confianza utilizado para la poda (valores menores suponen mayor poda)	0,25
	minNumObj	Número mínimo de instancias por regla	2
	numFolds	Determina la cantidad de datos para poda de error-reducida. Una partición se usa para poda, el resto para hacer crecer las reglas	3
	reducedErrorPruning	Si se usa poda de error reducida en lugar de la del C4.5	False
	unpruned	Si no se realiza poda	False

Tabla 4.2: Configuración de parámetros configurables por el usuario

valor de los parámetros restantes a los valores indicados en la Tabla 4.2. Sin embargo, esta idea fue desechada al considerar que minaba las posibilidades de ejecución del algoritmo en manos de usuarios expertos, dificultándoles sacar todo el provecho posible del potencial del algoritmo. Por ello, los cuatro parámetros mencionados se

consideran obligatorios, mientras que los otros seis, representados en letra cursiva y entre corchetes, poseen un valor por defecto que puede ser modificado por el usuario si así lo considera oportuno.

4.6.4. Análisis de sensibilidad de los parámetros de GBAP

Con el objetivo de determinar la configuración de parámetros de GBAP se llevó a cabo un análisis de sensibilidad de los parámetros del algoritmo. Así, la configuración indicada en la Tabla 4.2 fue adoptada tras efectuar un procedimiento de validación cruzada sobre tres conjuntos de datos representativos de entre los empleados en el estudio, con características y dimensionalidad variadas: *primary-tumor*, *hepatitis* y *wine*. Por cada parámetro se emplearon diferentes rangos de valores y a continuación se analizó qué configuración específica permitía alcanzar un mayor rendimiento. En las Figuras 4.2 y 4.3 se visualiza el rendimiento del algoritmo sobre cada conjunto de datos de acuerdo con distintas configuraciones de cada parámetro. Se puede observar que no necesariamente los valores finalmente adoptados para cada parámetro tenían el mejor comportamiento para los tres conjuntos de datos simultáneamente, como era de esperar.

Algunas conclusiones arrojadas por el estudio son las siguientes:

- Una vez que el parámetro $numAnts$ sobrepasa los 20 individuos, el rendimiento del algoritmo no mejora en ningún conjunto de datos. Es más, cuanto mayor sea el valor de este parámetro, más hormigas tendrán que ser creadas y evaluadas en cada generación, lo que incrementa el coste computacional.
- Un comportamiento similar se puede asociar al parámetro $numGenerations$, cuyos valores óptimos se centran en el intervalo $[10, 100]$.
- El parámetro $maxDerivations$ se ha mostrado sensible a cada conjunto de datos específico, por lo que optamos por un valor intermedio.
- $minCasesPerRule$ también se muestra como un parámetro altamente sensible. Consideramos que su valor debe estar comprendido en el intervalo $[1, 5]$.
- Los parámetros τ_0 y τ_min parecen ser los menos sensibles. Por otro lado, el valor para τ_max debe pertenecer al rango $[0{,}8, 1]$.

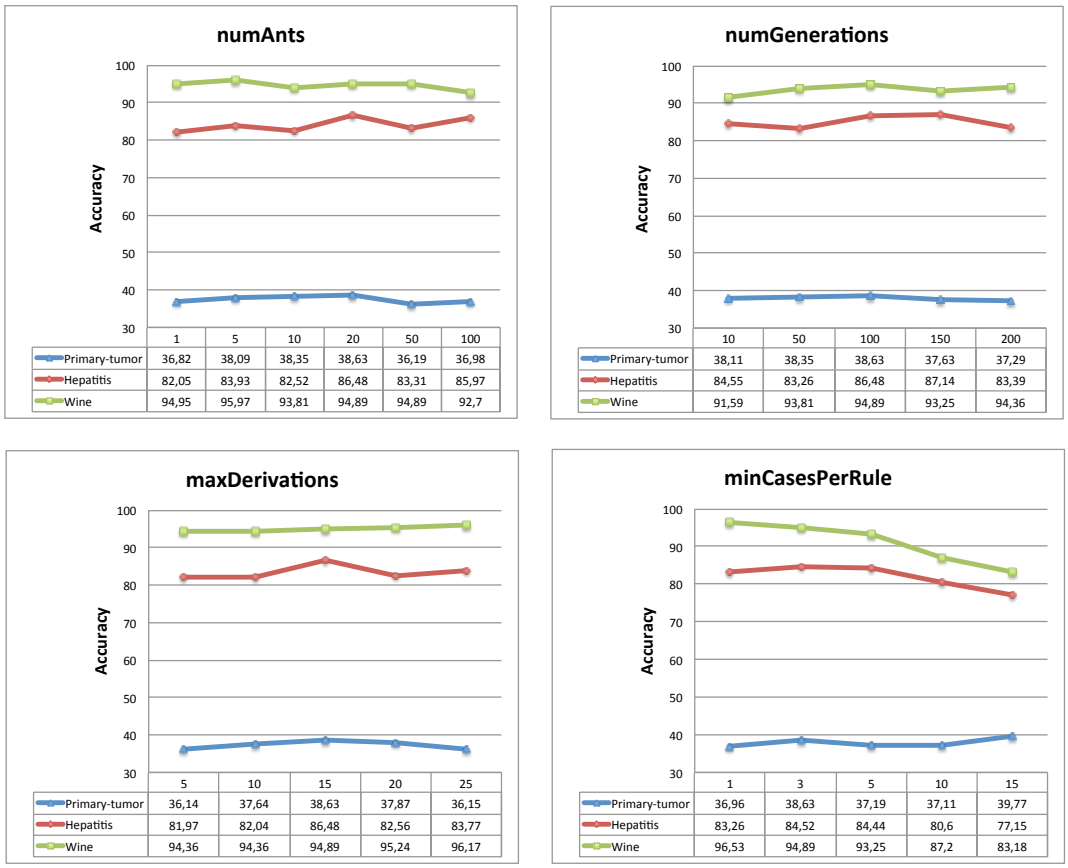


Figura 4.2: Análisis de sensibilidad de los parámetros *numAnts*, *numGenerations*, *maxDerivations* y *minCasesPerRule*

- Los valores apropiados para el factor de evaporación ρ están comprendidos entre $[0{,}01, 0{,}1]$. Cuanto más bajo sea el valor del mismo, mejores resultados se espera obtener, dado que el proceso de convergencia será más lento.

- Finalmente, los valores óptimos para los exponentes α y β se encuadran en los intervalos $[0{,}2, 0{,}4]$ y $[0{,}8, 1]$, respectivamente.

Es importante comentar que, pese a que la configuración adoptada por GBAP se mantuvo durante el estudio experimental aquí presentado, a la hora de aplicar el algoritmo sobre un problema específico se puede optimizar su configuración de parámetros con el objeto de que alcance un comportamiento óptimo en dicho problema.

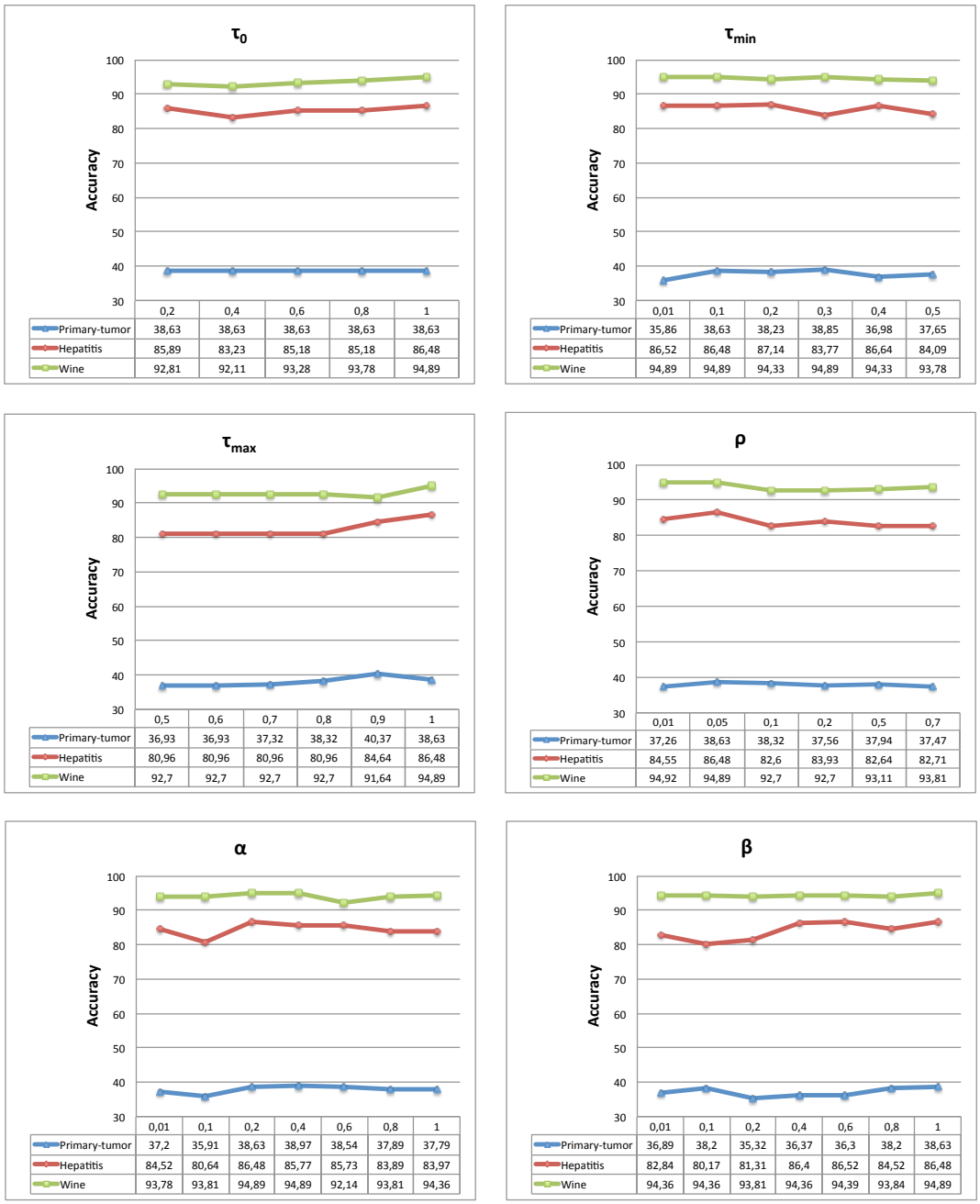


Figura 4.3: Análisis de sensibilidad de los parámetros tau_0, tau_{min}, tau_{max}, ρ, α y β

4.7. Resultados

En esta sección se compara el rendimiento y la comprensibilidad del modelo propuesto con respecto a los algoritmos de clasificación basados en reglas introducidos

en la Sección 4.6.3, analizando estadísticamente las diferencias entre ellos. Es importante tener en mente que en DM no existe ningún algoritmo que se comporte, en general, de forma superior a todos los demás para cada conjunto de datos, tal como enuncia el teorema de *no free lunch* [231, 232].

4.7.1. Estudio de exactitud predictiva

El primer criterio considerado para comparar los algoritmos es la exactitud predictiva. La Tabla 4.3 muestra los valores medios obtenidos por cada algoritmo sobre cada conjunto de datos. Los mejores resultados para cada conjunto de datos se muestran resaltados en negrita. Analizando la tabla, es posible darse cuenta de que GBAP es competitivo con respecto a los demás algoritmos considerados, obteniendo incluso los mejores resultados en el 50 % de los problemas considerados en la experimentación. Además, en aquellos conjuntos de datos en los que GBAP no alcanza los mejores valores de exactitud, sus resultados son muy competitivos. Con respecto a los valores de desviación típica podemos observar, asimismo, que GBAP arroja valores intermedios en términos de estabilidad.

Aunque GBAP obtiene los mejores valores medios de *accuracy*, como puede observarse en la última fila de la Tabla 4.7.2 o bien fijándonos en las líneas centrales de los diagramas de cajas de la Figura 4.4, con el objetivo de comparar los resultados y determinar si existen diferencias significativas entre los clasificadores, llevamos a cabo el test de Iman&Davenport. Se trata de un test no paramétrico que compara los rangos medios de los k algoritmos sobre los N conjuntos de datos. La asignación de rangos se efectúa de la siguiente manera: el algoritmo con exactitud más alta para un conjunto de datos concreto recibe el rango 1, al siguiente con mayor valor se le asigna el rango 2, y así hasta asignar las posiciones a todos los algoritmos. Este procedimiento se realiza para cada conjunto de datos y se calcula el rango medio obtenido por cada algoritmo en todos ellos. Estos rangos permiten conocer el algoritmo que obtiene los mejores resultados considerando todos los conjuntos de datos. Así, aquél cuyo valor sea más cercano a 1 será el que mejor se comporta sobre la mayoría de los conjuntos de datos.

El rango medio obtenido por cada clasificador se puede observar en la última fila de la Tabla 4.3. Prestando atención a los valores de los mismos se puede apreciar

	GBAP		Ant-Miner		Ant-Miner+		PSO/ACO2		GP		JRIP		PART	
Dataset	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}
Hepatitis	82,17	12,04	83,27	10,32	81,79	10,30	84,59	9,33	71,05	14,45	81,54	12,05	**84,64**	7,66
Sonar	**81,98**	7,44	76,95	6,89	76,05	7,22	78,49	8,05	79,82	9,24	80,33	6,61	77,84	8,10
Breast-c	71,40	7,86	**73,42**	7,29	73,05	6,86	68,63	6,87	68,63	10,94	72,00	6,41	68,48	7,90
Heart-c	**82,84**	5,24	78,01	6,69	82,41	5,10	82,25	5,36	70,02	7,08	82,20	5,12	80,13	6,39
Ionosphere	**93,02**	4,07	84,39	6,73	92,89	4,02	89,97	4,99	76,48	8,19	91,70	5,14	88,93	4,02
Horse-c	82,97	6,34	82,71	4,73	81,79	6,03	82,06	4,93	82,52	6,06	**83,72**	6,35	81,5	3,72
Australian	85,47	4,49	85,30	4,12	83,48	3,38	85,19	4,69	85,52	4,50	**86,70**	5,15	84,66	4,48
Breast-w	**96,50**	1,68	94,69	2,04	94,28	2,86	95,86	1,91	87,39	2,75	95,71	1,81	95,71	1,82
Diabetes	**75,80**	4,12	72,48	3,76	74,58	4,81	74,16	4,47	61,94	4,72	75,56	2,34	75,66	2,52
Credit-g	70,79	4,27	70,55	3,72	70,80	3,87	70,36	3,55	63,02	7,03	70,70	3,26	**72,70**	3,26
Mushroom	98,26	0,76	98,15	0,71	98,89	0,63	99,90	0,11	86,22	6,11	99,99	0,04	**100,00**	**0,00**
Iris	**96,00**	4,10	95,20	5,47	94,00	3,59	95,33	6,70	91,73	10,46	**96,00**	5,33	95,33	6,70
Wine	**97,01**	4,37	91,86	5,08	93,86	4,61	90,20	2,86	83,69	9,44	95,61	5,37	95,03	3,89
Balance-scale	75,49	4,97	68,36	5,30	**77,75**	6,31	77,14	4,93	58,38	7,76	73,42	5,66	76,50	3,51
Lymphography	**81,00**	10,35	75,51	9,59	77,23	10,91	76,59	12,20	77,78	12,77	78,84	11,49	78,43	14,30
Glass	69,13	8,66	65,52	9,26	62,03	9,80	71,16	10,54	39,23	11,34	69,00	8,70	**73,91**	8,43
Zoo	**95,60**	4,21	92,55	7,93	93,09	10,65	92,32	7,19	64,20	18,88	86,85	7,25	94,84	9,02
Primary-tumor	37,91	6,55	37,75	5,27	37,26	5,43	37,19	5,88	16,41	4,96	38,11	3,75	**38,36**	5,09
Ranking	**2,25**		4,78		4,39		4,11		6,08		3,06		3,33	

Tabla 4.3: Resultados comparativos de exactitud predictiva (%) en test

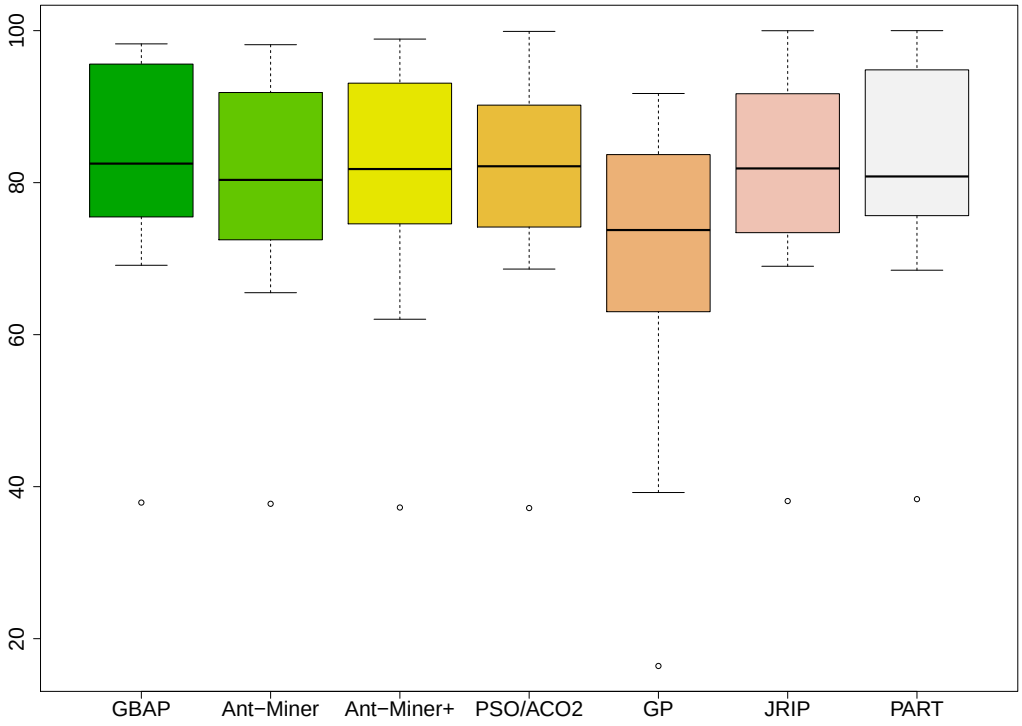


Figura 4.4: Diagramas de caja de la exactitud predictiva (%) en test

que el resultado más bajo, es decir, la mejor posición global, fue obtenida por nuestra propuesta. El valor calculado para el estadístico de Iman&Davenport de acuerdo a una distribución F con $k-1$ y $(k-1)(N-1)$ grados de libertad es igual a 8,7404. Este valor excede el valor crítico a un nivel de significación de $\alpha = 0{,}1$, $C_0 = [0, (F_F)_{0,1,6,102} = 1{,}8327]$. Por tanto, la hipótesis nula de que todos los algoritmos se comportan igualmente bien puede rechazarse para este nivel de confianza $\alpha = 0{,}1$.

Al ser rechazada la hipótesis nula por el test de Iman&Davenport, podemos afirmar que existen diferencias significativas entre los algoritmos, pero este test no nos permite concretar entre qué algoritmos se producen. Para ello es necesario llevar a cabo un test a posteriori y, dado que todos los clasificadores se comparan frente a uno de control (el de rango inferior, en este caso, GBAP), podemos aplicar el test de Bonferroni–Dunn [59] centrándonos en todas las posibles comparaciones por parejas entre GBAP y el resto de algoritmos. Así, al mismo nivel de confianza que se realizó previamente el test de Iman&Davenport, $\alpha = 0{,}1$, el valor crítico obtenido

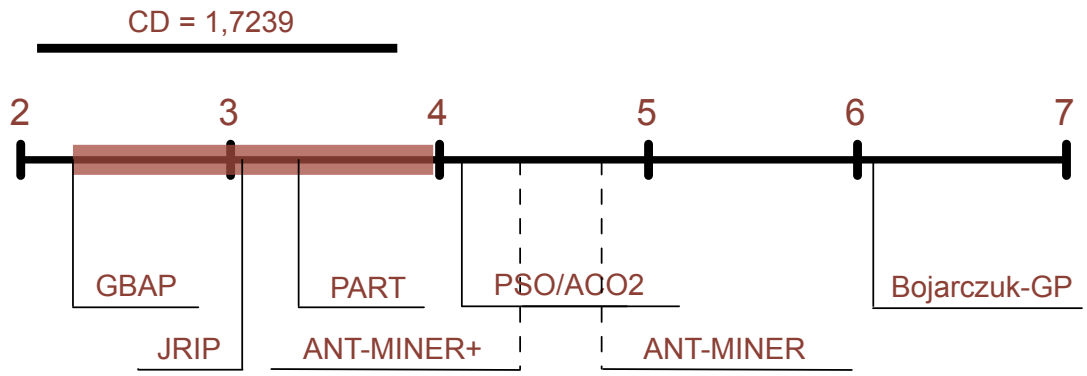


Figura 4.5: Test de Bonferroni–Dunn. GBAP presenta diferencias significativas con respecto a aquellos clasificadores cuyo *ranking* está fuera del intervalo sombreado ($p < 0{,}1$)

por Bonferroni–Dunn es de 1,7239, lo que significa que para que las diferencias entre dos algoritmos sean significativas, estas han de ser al menos de 1,7239 unidades [205]. Así pues, se comprueba que el rendimiento de GBAP es estadísticamente superior al de los algoritmos PSO/ACO2, Ant-Miner+, Ant-Miner y Bojarczuk-GP, en este orden, ya que la diferencia entre sus rangos medios y el de GBAP es mayor que el valor umbral mencionado. Estos resultados se recogen gráficamente en la Figura 4.5, donde es posible observar, asimismo, que GBAP obtiene incluso mejores resultados de exactitud predictiva que PART y que JRIP.

Para niveles de confianza más restrictivos como $\alpha = 0{,}05$ y $\alpha = 0{,}01$, el test de Iman&Davenport rechaza igualmente la hipótesis nula. En el primer caso, el valor crítico de Bonferroni–Dunn es de 1,8996, lo que indica que GBAP es significativamente más exacto que Ant-Miner+, Ant-Miner y GP. En el segundo, para $\alpha = 0{,}01$, el valor crítico de Bonferroni–Dunn es igual a 2,2639, por lo que con una probabilidad del 99 % se puede afirmar que GBAP tiene un rendimiento significativamente superior que Ant-Miner y GP.

Con el objetivo de contrastar los resultados obtenidos tras la aplicación del procedimiento de Bonferroni–Dunn, podemos utilizar el test de Holm, que es más preciso que el primero y que además no hace asunciones adicionales sobre las hipótesis [59]. La ventaja que supone la utilización del test de Bonferroni–Dunn radica en que es más fácil de describir y visualizar los resultados ya que utiliza la misma diferencia crítica para comparar cualesquiera dos algoritmos. En cambio, el test de Holm

comprueba las hipótesis ordenándolas por significación, comparanado cada p_i con $\alpha/(k-i)$ desde el *p-value* más significativo. La Tabla 4.4 recoge todas las posibles hipótesis de comparación entre el algoritmo de control y el resto, ordenándolas por su *p-value* y asociándoles su nivel de significación α. El test de Holm rechaza las hipótesis cuyo *p-value* es menor o igual que su valor ajustado de α. Así, el algoritmo PART estaría en el umbral, dado que su *p-value* (0,132464) no es inferior a su valor α ajustado (0,025). A partir de él, el resto de algoritmos sí que se comportan estadísticamente peor. Por ello, a un nivel de $\alpha = 0{,}05$, GBAP es estadísticamente más exacto que los algoritmos PSO/ACO2, Ant-Miner+ y Bojarczuk-GP.

i	Algoritmo	z	p	α/i	Hipótesis nula
6	GP	5,323465	1,0180E-7	0,008333	Rechazada
5	Ant-Miner	3,510401	4,4743E-4	0,01	Rechazada
4	Ant-Miner+	2,970339	0,002974	0,0125	Rechazada
3	PSO/ACO2	2,584581	0,009749	0,016666	Rechazada
2	PART	1,504457	0,132463	0,025	Aceptada
1	JRIP	1,118699	0,263268	0,05	Aceptada

Tabla 4.4: Resultados del test de Holm para $\alpha = 0{,}05$

4.7.2. Estudio de comprensibilidad

Un segundo criterio de evaluación es la evaluación de la comprensibilidad del clasificador inducido. Al contrario que la exactitud, se trata de un concepto subjetivo que se asocia frecuentemente a la simplicidad sintáctica del clasificador [27]. Por ello, cuantas menos reglas tenga el clasificador y menos condiciones aparezcan en ellas, menor será la complejidad del clasificador y, por tanto, mayor será su interpretabilidad.

La Tabla 4.5 resume los resultados de número de reglas del clasificador, presentando el número de reglas medio extraído para cada conjunto de datos, y la complejidad de las reglas, indicando el número medio de condiciones por regla. La penúltima fila de la tabla muestra el rango medio obtenido por cada algoritmo con respecto al número de reglas en el clasificador, mientras que la última fila hace lo propio para el número de condiciones por regla. En ambos casos el algoritmo de control es GP, al ser el que obtiene el rango más bajo. Por su parte, las Figuras 4.6 y 4.7 muestran

los diagramas de cajas para el número de reglas y el número de condiciones por regla, respectivamente.

Antes de analizar los resultados obtenidos, hay que tener en mente que todos los algoritmos excepto GP extraen las reglas de un modo similar, como una conjunción de condiciones. En cambio, GP utiliza, además del operador *AND*, el operador *OR*. Por tanto, debido a la codificación en forma de árbol de los individuos en GP, para calcular de una manera justa el número de reglas y el número de condiciones en ellas, es necesario dividir la regla en dos por cada operador *OR*, dejando de contabilizar los nodos *OR* como condiciones.

El primer análisis estadístico llevado a cabo considera el número medio de reglas en el clasificador final. A un nivel de significación de $\alpha = 0{,}05$, la aplicación del test de Iman&Davenport rechaza la hipótesis nula, ya que el valor del estadístico, 23,4734, no pertenece al intervalo crítico $C_0 = [0, (F_F)_{0,05,6,102} = 2{,}1888]$. Para mostrar las diferencias significativas aplicamos el test a posteriori de Bonferroni–Dunn, cuyo valor crítico resultante es de 1,8995 para $\alpha = 0{,}05$. Esto quiere decir que GP, JRIP y Ant-Miner+ se comportan estadísticamente mejor que GBAP. No obstante, GBAP no se comporta estadísticamente peor que los algoritmos Ant-Miner, PSO/ACO2 y PART.

El mejor resultado posible en cuanto a este criterio supondría extraer una única regla por clase, lo cual puede llevar a no obtener buenos resultados en aquellos conjuntos de datos donde la distribución de instancias de una clase no se encuentre localizada en la misma región del espacio. Este fenómeno se puede observar en el comportamiento del algoritmo GP, ya que prácticamente extrae de media una regla por clase, lo que va en detrimento de la exactitud obtenida por este algoritmo, al no ser un número de reglas suficiente para obtener buenos resultados en muchos de los conjuntos de datos (ver Tabla 4.3).

Es interesante comprobar que, a pesar de que los nodos *OR* no se consideran condiciones, sino como una forma de unir dos reglas diferentes que predicen la misma clase, el algoritmo GP tiende a minimizar el uso de este operador (los resultados medios de número de reglas por conjunto de datos así lo corroboran). Esto se debe a que, tal como se explicó en la Sección 3.2.1, este algoritmo presenta un componente de simplicidad en la función de *fitness*, con lo que la presencia

	GBAP		Ant-Miner		Ant-Miner+		PSO/ACO2		GP		JRIP		PART	
Dataset	#R	#C/R	#R	#C/R	#R	#C/R	#R	#C/R	#R	#C/R	#R	#C/R	#R	#C/R
Hepatitis	8,1	1,89	4,8	1,99	3,9	3,25	7,4	2,28	3,1	1,22	3,8	2,15	8,4	2,30
Sonar	12,3	1,81	5,2	2,07	4,0	3,48	6,1	2,92	3,0	1,00	4,6	2,21	13,9	2,98
Breast-c	13,2	1,91	6,0	1,28	5,4	2,82	11,8	1,75	3,5	1,01	3,3	1,70	17,1	2,12
Heart-c	14,5	1,67	5,9	1,20	4,4	2,82	11,9	3,81	3,0	3,02	5,3	2,32	17,3	2,35
Ionosphere	11,1	1,18	5,7	1,61	8,8	1,41	4,5	4,03	3,1	1,14	7,7	1,48	8,2	1,83
Horse-c	9,0	1,46	6,3	1,49	4,7	3,41	20,1	3,39	3,0	1,00	3,5	1,74	13,2	2,38
Australian	10,1	1,08	6,5	1,53	3,3	2,08	25,8	6,96	3,0	1,00	5,2	1,80	19,4	2,01
Breast-w	6,6	1,65	7,2	1,04	6,4	1,92	10,5	1,1	3,0	1,00	6,5	1,74	10,9	1,63
Diabetes	9,9	1,53	8,6	1,03	5,5	3,71	35,2	3,61	3,0	1,33	4,6	2,88	17,9	2,21
Credit-g	22,9	1,82	9,1	1,51	3,3	3,31	52,8	4,2	3,3	1,17	7,1	2,54	57,8	2,70
Mushroom	6,7	1,33	7,7	1,19	8,6	1,27	9,1	2,04	3,3	1,12	8,5	1,58	10,0	1,72
Iris	3,7	1,06	4,3	1,03	3,9	1,8	3,0	1,20	4,3	1,29	3,0	1,00	4,6	1,00
Wine	7,2	1,50	5,1	1,33	2,5	2,19	4,0	1,73	4,1	1,27	4,2	1,56	6,3	1,77
Balance-scale	16,7	1,92	12,4	1,01	9,1	3,35	27,0	2,69	5,2	1,56	12,4	1,84	28,6	1,55
Lymphography	10,2	1,60	4,7	1,69	4,6	2,83	15,6	2,11	5,1	1,02	6,9	1,53	10,2	2,30
Glass	21,6	1,79	8,4	1,76	12,4	4,10	24,5	3,13	8,2	1,48	8,0	2,03	13,7	2,32
Zoo	8,7	1,97	6,1	1,32	6,7	4,07	7,1	1,47	8,0	1,42	7,4	1,58	7,7	1,57
Primary-tumor	45,9	2,60	12,1	3,35	9,3	8,50	86,5	6,01	23,7	1,37	8,3	3,13	48,7	3,23
#R Ranking	5,30		3,67		2,69		5,25		**2,06**		2,72		6,31	
#C/R Ranking	3,22		2,5		6,33		5,55		**1,78**		3,86		4,75	

Tabla 4.5: Resultados comparativos del tamaño del clasificador y la complejidad de las reglas

de este operador reduce sustancialmente dicho componente, lo que provoca que la calidad de los individuos decaiga en el proceso evolutivo.

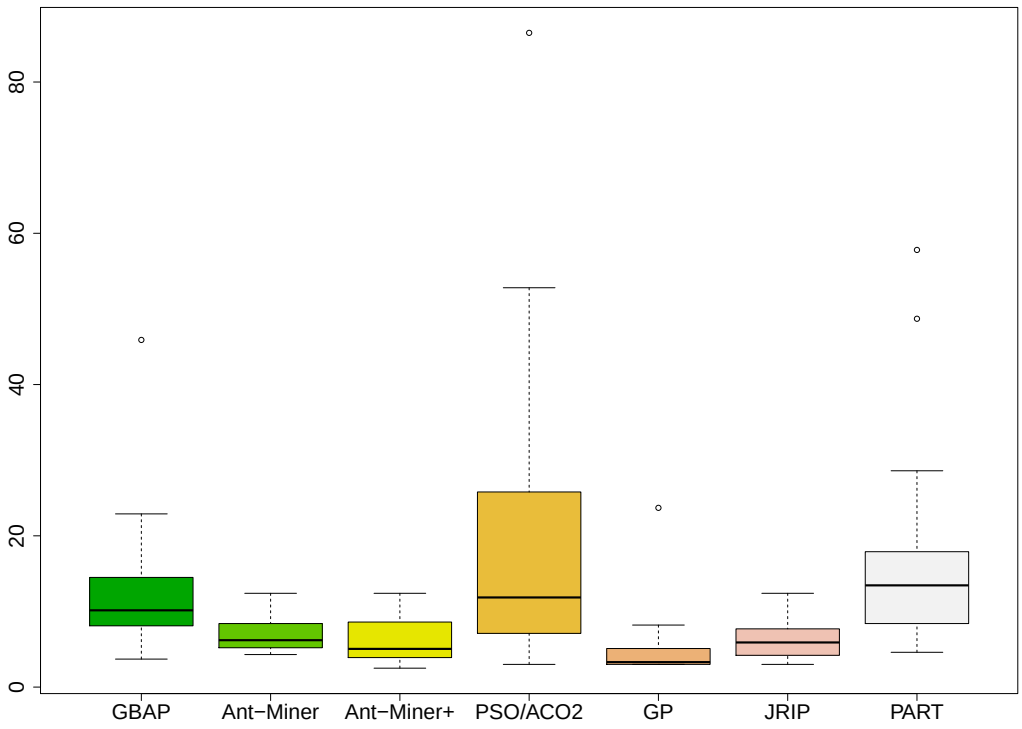


Figura 4.6: Diagramas de caja del número medio de reglas de los clasificadores

Al contrario que el algoritmo de GP, gracias a la aplicación del algoritmo de nichos descrito en la Sección 4.5.1, GBAP se asegura la selección de un número de reglas suficiente para cubrir los ejemplos de cada clase, alcanzando asimismo muy buenos resultados de clasificación.

El segundo estudio realizado conllevó el análisis del número de condiciones medio que presentan las reglas en los clasificadores. Para saber si los algoritmos presentaban diferencias al respecto, aplicamos el test de Iman&Davenport al mismo nivel de confianza considerado previamente, es decir, $\alpha = 0{,}05$. El valor del estadístico para la distribución F es igual a 22,0539, que tampoco pertenece al intervalo crítico $C_0 = [0, (F_F)_{0,05,6,102} = 2{,}1888]$, con lo que se rechaza igualmente la hipótesis nula de que todos los clasificadores presentan un comportamiento similar. La aplicación del test de Bonferroni–Dunn reveló que GBAP se comporta significativamente mejor que Ant-Miner+ y PSO/ACO2. Otra conclusión arrojada por este test es que,

aunque GBAP no presenta diferencias significativas con los demás algoritmos, tampoco existe ninguno entre ellos que haga lo propio con respecto a nuestra propuesta.

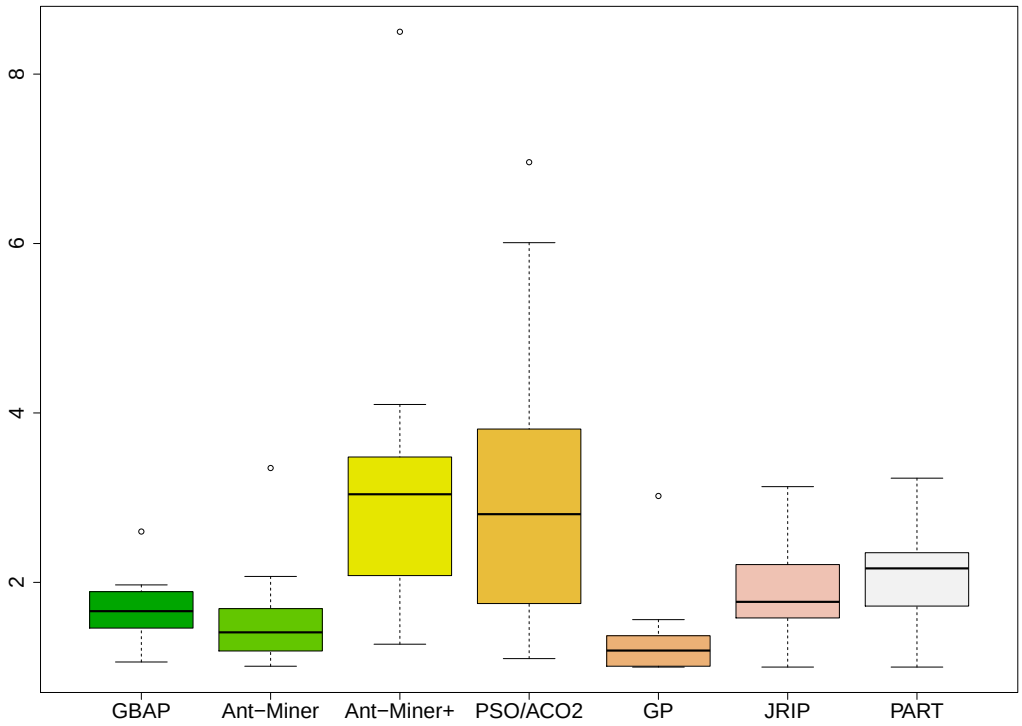


Figura 4.7: Diagramas de caja del número medio de condiciones por regla

En cuanto a esta medida, cabe resaltar que el uso de una CFG en GBAP supone un beneficio, dado que restringe la complejidad de cada regla mediante el número de derivaciones máximas permitidas para la gramática. Es por ello que nuestra propuesta permite modificar el compromiso entre la complejidad y el rendimiento del clasificador modificando el valor de este parámetro, de manera que reglas con más condiciones pueden conllevar obtener mejores resultados, al ser capaces de descubrir relaciones más complejas entre los atributos. Como se puede deducir de la Figura 4.7 o en la Tabla 4.7.2, el algoritmo GBAP es el tercero que obtiene menos condiciones por regla, sólo superado por GP y Ant-Miner. El motivo por el cual GP obtiene el número de condiciones por regla más bajo también se debe, al igual que en el caso del número de reglas en el clasificador, al uso del componente de simplicidad que considera en la función de aptitud. GBAP también tiene en

cuenta la complejidad de las reglas a la hora de efectuar el refuerzo de feromonas, como se explicó en la Sección 4.4.

Algoritmo	Accuracy	#R	#C/R
GBAP	**81,85**	13,41	1,65
Ant-Miner	79,25	7,01	1,52
Ant-Miner+	80,29	5,93	3,13
PSO/ACO2	80,63	20,16	3,02
GP	70,22	**5,16**	**1,30**
JRIP	80,99	6,13	1,93
PART	81,26	17,44	2,11

Tabla 4.6: Resultados medios de los algoritmos

La importancia del balance entre comprensibilidad y exactitud queda perfectamente ilustrada en los resultados obtenidos por el algoritmo de GP: es el algoritmo más comprensible pero, sin embargo, también es el que peores resultados de exactitud reporta. En cambio, podemos concluir que GBAP sí que presenta un buen compromiso entre exactitud y comprensibilidad, dado que es el que mejor rango de exactitud presenta, sin que ello le lleve a obtener malos resultados de comprensibilidad, alcanzando resultados bastante competitivos en este sentido.

Por último, se muestra un ejemplo de clasificador obtenido al ejecutar el algoritmo GBAP sobre una partición de entrenamiento del conjunto de datos *hepatitis*.

4.8. Conclusiones

En este capítulo se ha presentado un nuevo algoritmo de programación automática basado en ACO guiado por una CFG para la tarea de clasificación multiclase. Este algoritmo, denominado GBAP, utiliza dos medidas de heurística complementarias que conducen el proceso de búsqueda hacia la obtención de soluciones válidas. El algoritmo ofrece la oportunidad al usuario de modificar la complejidad de las reglas extraídas variando el número de derivaciones permitidas para la gramática. Además, hace uso de un algoritmo de nichos específicamente implementado para asignar el consecuente a las reglas y seleccionar aquellas que van a formar parte del

```
IF (!= ALBUMIN (-inf-2,65] ) THEN LIVE
ELSE IF (AND(!= PROTIME (44,5-inf) )
            (= ASCITES yes) ) THEN DIE
ELSE IF (= ALBUMIN (-inf-2,65] ) THEN DIE
ELSE IF (AND(!= AGE (-inf-29] )
            (= VARICES yes) ) THEN DIE
ELSE IF (= ANTIVIRALS no) THEN DIE
ELSE IF (AND(!= PROTIME (44,5-inf) )
            (= ASCITES no) ) THEN DIE
ELSE IF (AND(= PROTIME (-inf-44,5] )
            (= SPIDERS no) ) THEN DIE
ELSE LIVE
```

Tabla 4.7: Ejemplo de clasificador sobre el conjunto de datos *hepatitis*

clasificador final sin eliminar o descartar ejemplos del conjunto de entrenamiento, evitando así las desventajas de los algoritmos de cubrimiento secuencial.

Aunque este algoritmo se ha desarrollado para la tarea de clasificación de DM, puede ser aplicado a otro tipo de problemas, estableciendo otra forma de evaluar los individuos y diseñando una gramática acorde al problema objetivo.

GBAP se ha comparado sobre dieciocho conjuntos de datos diferentes con otros algoritmos representativos de inducción de reglas: tres algoritmos basados en hormigas (Ant-Miner, Ant-Miner+ y PSO/ACO2), un algoritmo de GP, y otros dos clasificadores ampliamente conocidos (JRIP) y PART). Los métodos estadísticos no paramétricos empleados para analizar la exactitud y la comprensibilidad del algoritmo permiten concluir que, por un lado, GBAP es más exacto estadísticamente que PSO/ACO2, Ant-Miner+, Ant-Miner y que el algoritmo de GP con una probabilidad del 95 %; y, por otro, que GBAP es competitivo en cuanto a interpretabilidad se refiere. Estos resultados demuestran que el paradigma de AP también puede emplearse satisfactoriamente para abordar problemas de clasificación, tal como otro paradigma de programación automática, la GP, o los algoritmos de ACO han demostrado previamente.

5

Modelo multiobjetivo de AP para clasificación multiclase

En ocasiones es necesario encontrar clasificadores cuyas reglas satisfagan simultáneamente varias medidas. Así, el diseño de un clasificador basado en reglas también se puede abordar desde la perspectiva de la optimización multiobjetivo (MOO, *Multi-Objective Optimization*). A diferencia de la optimización simple, trata de descubrir reglas de clasificación que optimizan simultáneamente varios objetivos, pero la mejora en los valores para uno habitualmente obra en detrimento del resto. La MOO se explica en detalle en la primera sección de este capítulo.

Los algoritmos de clasificación multiobjetivo se orientan habitualmente hacia la obtención de clasificadores con un buen compromiso entre exactitud y comprensibilidad. Cabe mencionar que se puede considerar que un clasificador es más comprensible que otro en términos de su tamaño, si presenta menos reglas que otro, o en función de la longitud de sus reglas, si presenta un menor número de condiciones por regla. Se han presentado diversas propuestas basadas en la MOO para la extracción de reglas utilizando EAs [56, 114] y PSO [6, 223]. Sin embargo, la aplicación de algoritmos multiobjetivo basados en ACO (MOACO, *Multi-Objective Ant Colony Optimization*) a la tarea de clasificación no ha sido estudiada hasta

la fecha, cuando sí que se ha utilizado este enfoque en ACO con éxito para otros problemas de optimización, como el del TSP [91]. Se puede encontrar una revisión exhaustiva de los algoritmos de MOACO en [12].

Partiendo de los buenos resultados obtenidos por el algoritmo GBAP, presentado en el Capítulo 4, buscamos ahora combinar las ventajas de la AP guiada por gramática con aquellas inherentes a las propuestas multiobjetivo. Así, presentamos un nuevo algoritmo denominado **MOGBAP** (*Multi-Objective Grammar-Based Ant Programming*) para la extracción de clasificadores compuestos de reglas del tipo *IF-THEN*, centrándose en la obtención de un buen balance entre exactitud y comprensibilidad. El algoritmo propuesto puede considerarse como el primer algoritmo basado en hormigas que aborda la tarea de clasificación. Además, su principal contribución al campo estriba en abordar el descubrimiento de frentes de Pareto de un modo novedoso, encontrando un frente por cada clase del conjunto de datos para combinar posteriormente las soluciones encontradas en cada frente mediante un enfoque de nichos.

El algoritmo presentado se compara frente a otros siete algoritmos de inducción de reglas y frente al algoritmo GBAP, analizando los resultados estadísticamente. Los test estadísticos prueban que MOGBAP es significativamente más exacto que la mayoría de los otros algoritmos incluidos en el estudio y, lo que es más importante, lo consigue alcanzando al mismo tiempo un buen compromiso en cuanto a comprensibilidad se refiere.

5.1. Optimización multiobjetivo

5.1.1. Definición

Un problema de MOO general se puede definir mediante un conjunto de n funciones objetivo a optimizar, un conjunto de m parámetros o variables de decisión y un conjunto de c restricciones [246]. Dado un espacio de soluciones X y un espacio objetivo Y, las funciones objetivo y las restricciones son funciones dependientes de las variables de decisión:

$$\begin{aligned} max/min \quad & y = f(x) = (f_1(x), f_2(x), ..., f_n(x)) \\ restricciones \quad & r(x) = (r_1(x), r_2(x), ..., r_c(x)) \leq 0 \\ siendo \quad & \mathbf{x} = (x_1, x_2, ..., x_m) \in X \\ & \mathbf{y} = (y_1, y_2, ..., y_n) \in Y \end{aligned} \tag{5.1}$$

donde:

$\mathbf{x}$	es el vector de decisión,
$\mathbf{y}$	es el vector objetivo, y
$r(x) \leq 0$	determina el conjunto de soluciones factibles.

5.1.2. Dominancia de Pareto

Los objetivos a optimizar pueden ser independientes, aunque lo habitual es que exista una relación de dependencia entre ellos. En este caso, los objetivos pueden estar en **conflicto** o bien en **armonía** [182]. Se habla de objetivos en conflicto si el aumento en el rendimiento de un objetivo deteriora el rendimiento del otro, mientras que se dice que dos objetivos están en armonía si la mejora del rendimiento de uno no afecta o bien lo hace favorablemente al rendimiento del otro.

Cuando los objetivos están en conflicto existen múltiples soluciones de compromiso, en lugar de existir una única solución óptima para todos los objetivos. En este caso, cada una de las soluciones se dice que es una **solución óptima de Pareto**[1]. Una solución óptima de Pareto presenta el mejor compromiso posible entre las funciones objetivo, de manera que no existe ninguna modificación posible que permita mejorar en alguna de las dimensiones sin una degradación en las otras.

[1]El término recibe su nombre a partir del economista *Vilfredo Pareto*, quien desarrolló este concepto para sus estudios de eficiencia económica y distribución de la renta. El concepto de eficiencia de Pareto fue posteriormente aplicado en ingeniería y ciencias sociales.

Suponiendo un problema de maximización con dos vectores de decisión $a, b \in X$, se dice que ***a* domina a *b***, representado como $a \succ b$, si:

$$\begin{aligned} &\forall i \in \{1, 2, ...n\} : f_i(a) \geq f_i(b) \quad \wedge \\ &\exists j \in \{1, 2, ...n\} : f_j(a) > f_j(b) \end{aligned} \tag{5.2}$$

Este concepto, demoninado **dominancia de Pareto**, es el que permite en MOO comparar las soluciones y ordenarlas. Siguiendo con la suposición de un problema de maximización, se pueden producir las siguientes relaciones de dominancia entre dos soluciones s_1 y s_2:

- **Dominancia fuerte**, denotada como $s_1 \succ\succ s_2$, si s_1 es mejor que s_2 en todos los objetivos.
- **Dominancia**, $s_1 \succ s_2$, si s_1 no es peor que s_2 en todos los objetivos y es mejor en al menos uno.
- **Dominancia débil**, $s_1 \succeq s_2$, cuando s_1 no es peor que s_2 en todos los objetivos.

Así, un conjunto de soluciones no dominadas estará formado por soluciones que no están dominadas débilmente por ninguna otra solución. El conjunto óptimo de Pareto es un conjunto de soluciones no dominadas, aunque la relación inversa no tiene que darse necesariamente. Todas las soluciones óptimas de Pareto pertenecen al conjunto de Pareto, y los puntos que estas soluciones forman en el espacio de soluciones se conoce con el nombre de **frente de Pareto**.

El número de soluciones óptimas de Pareto que conforman el conjunto de Pareto y la forma que adopta el consiguiente frente de Pareto depende de cada problema de optimización. En la Figura 5.1 se ilustran, a modo de ejemplo, tres posibles formas que pueden adoptar los frentes de Pareto.

Se pueden considerar tres tipos de algoritmos de MOO, en función de cómo se presenta la solución al experto:

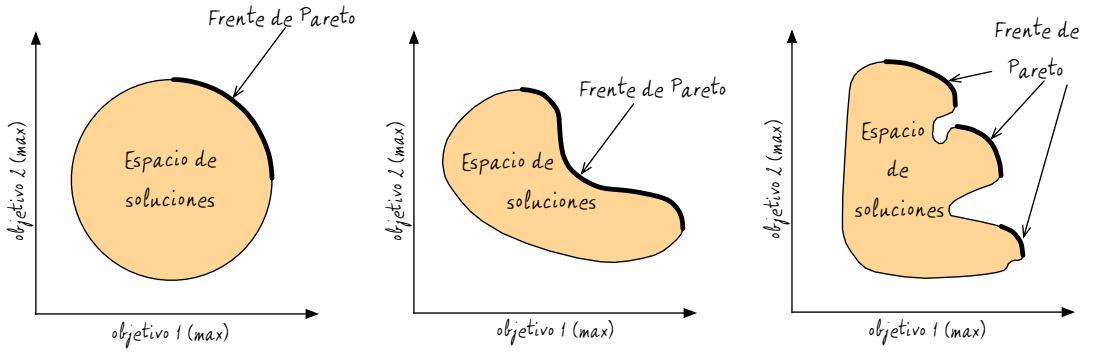


Figura 5.1: Ejemplo de formas que pueden adoptar los frentes de Pareto

- **A priori**. Tienen conocimiento previo acerca de la forma que adoptará la solución del problema. Así, combinan todos los objetivos en uno solo mediante una función de agregación y un vector de pesos, convirtiendo pues el problema multiobjetivo en uno monobjetivo.

- **Progresivos**. Pueden disponer de cierta información previa, que utilizan para ajustar el proceso de búsqueda conforme se van interpretando los resultados de la misma. Normalmente operan en tres fases: (1) encuentran una solución no dominada, (2) modifican las preferencias de los objetivos de manera acorde, y (3) repiten los dos pasos previos hasta que el experto queda satisfecho o no es posible encontrar ninguna mejora.

- **A posteriori**. Tienen como objetivo la obtención del conjunto óptimo de Pareto, sin presuponer nada, presentando al experto un conjunto de soluciones candidatas a partir de las que elegir.

Cuando no se dispone de información suficiente acerca del frente de Pareto, un enfoque progresivo o a posteriori puede ser más útil al ser capaz de arrojar cierta luz sobre la forma y la naturaleza del espacio de soluciones. Estos enfoques habitualmente se esfuerzan por alcanzar dos objetivos: encontrar soluciones cercanas al frente de Pareto y obtener buena diversidad de soluciones, es decir, mantener un buen cubrimiento de soluciones a lo largo del frente de Pareto.

5.1.3. Métricas para la comparación de frentes de Pareto

En MOO no existe un único criterio que permita discernir si un frente de Pareto es mejor que otro. Un frente de Pareto debería cumplir los siguientes **requisitos**:

- Estar formado por un alto número de soluciones diferentes.
- Minimizar la distancia con respecto al frente de Pareto óptimo del problema, es decir, el conjunto real de soluciones de dicho problema, y que normalmente no se conoce.
- Las soluciones deben estar bien distribuidas a lo largo del frente de Pareto.
- Maximizar la extensión del frente de Pareto obtenido. Para cada objetivo, las soluciones no dominadas deben cubrir la mayor amplitud de valores posible.

En la literatura han sido propuestas numerosas métricas para la comparación de frentes de Pareto, aunque algunas de ellas requieren conocer de antemano las soluciones reales para poder ser calculadas. En *Zitzler et al.* [247] y *Knowles et al.* [123], los autores ofrecen un análisis detallado de varios indicadores de calidad unarios que miden la distribución de soluciones y su distancia con respecto al frente de Pareto óptimo real. Estas medidas incluyen la distancia generacional, el espaciado, el ratio de error máximo del frente de Pareto y la extensión del conjunto de aproximación. En ambos artículos, los autores coinciden en afirmar que, en determinados casos, dichas medidas pueden conducir a resultados erróneos en cuanto a la calidad de las distribuciones. De hecho, ambos trabajos concluyen que incluso la combinación de estas medidas no es suficiente para obtener resultados fiables. En su lugar, afirman que es preferible hacer uso de un único indicador unario fiable en vez de emplear varios que pueden conllevar error, recomendando el uso del indicador epsilon y del hipervolumen.

El **ϵ-indicador**, dados dos frentes de Pareto a y b, calcula la cantidad mínima $\epsilon \in \mathbb{R}$ con la que es necesario manipular uno de los dos frentes para establecer una relación de dominancia débil entre ambos. Se define como:

$$\epsilon - indicador = min\{\epsilon \in \mathbb{R} | \forall b \in B \ \exists a \in A : a \succ_\epsilon b\} \tag{5.3}$$

El **hipervolumen** proporciona una medida volumen de cubrimiento del frente de Pareto conocido con respecto al espacio objetivo. Suponiendo un problema de MOO con dos objetivos, es igual a la suma de todas las áreas rectangulares limitadas por algún punto de referencia y las funciones objetivo $f_1(x), f_2(x)$. Matemáticamente se expresa como:

$$HV \triangleq \left\{ \bigcup_i vol_i | vec_i \in FrentePareto_{conocido} \right\} \tag{5.4}$$

donde:

- vol_i es el volumen entre el origen y el vector vec_i, y
- vec_i es un vector no dominado del frente de Pareto conocido.

Se asume que el punto de referencia para el hipervolumen es el valor mínimo de cada objetivo.

5.2. Introducción al algoritmo MOGBAP

En MOGBAP, al igual que se explicó en el Capítulo 4 para el algoritmo GBAP, el entorno que permite a las hormigas comunicarse indirectamente con las demás simulando el proceso de estigmergia que se produce en la naturaleza viene dado por el árbol de derivación que se puede generar a partir de la gramática. La gramática que emplea MOGBAP viene definida por:

G = (Σ_N, Σ_T, P, <EXP>)

Σ_N = {<EXP>, <COND>}

Σ_T = {AND, =

attr$_1$, attr$_2$, ..., attr$_n$,

value$_{11}$, value$_{12}$, ..., value$_{1m}$,

value$_{21}$, value$_{22}$, ..., value$_{2m}$,

..., value$_{n1}$, value$_{n2}$, ..., value$_{nm}$}

P = {<EXP>:= <COND>| AND <EXP><COND>,

<COND>:= todas las posibles combinaciones de la terna

= attr value}

La principal diferencia que presenta respecto al algoritmo original radica en la ausencia del operador '!='. Esta decisión fue adoptada tras la fase experimental del algoritmo, donde se observó un mejor comportamiento del mismo cuando no se incluía dicho operador en la gramática.

El algoritmo MOGBAP sigue el enfoque de codificación de individuos *individuo* = *regla*, al igual que GBAP. Otras características comunes con dicho algoritmo son la generación progresiva del espacio de estados (Sección 4.1), la función heurística con dos componentes (Sección 4.2) y la regla de transición (Sección 4.3), con lo que no nos detendremos en explicarlas de nuevo en este capítulo. Sí que conviene detenerse en analizar el proceso de actualización de feromonas y la evaluación de los individuos, ya que difieren en gran medida al tratarse, en este caso, de un modelo multiobjetivo.

5.3. Actualización de feromonas

En general, los algoritmos de MOACO se pueden encuadrar en dos categorías, según almacenen la información relativa a las feromonas [12]. Así, pueden usar una única matriz de feromonas o bien utilizar múltiples matrices, una por objetivo. El algoritmo MOGBAP pertenece al primer grupo, lo cual entraña un beneficio en lo que se refiere a requerimientos de memoria y tiempo de cómputo.

La evaporación y el refuerzo son las dos operaciones consideradas en cuanto al mantenimiento de feromonas en el entorno. La evaporación tiene lugar sobre el espacio de estados completo:

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1-\rho) \tag{5.5}$$

donde:

- τ_{ij} representa la cantidad de feromonas en la transición del estado i al j, y
- ρ indica la tasa de evaporación.

En cambio, el refuerzo de feromonas está localizado únicamente en las transiciones de las **hormigas no dominadas** obtenidas al final de cada generación. Cada una de ellas reforzará todas las transiciones de su camino con una cantidad de feromonas equivalente, siendo el valor de este refuerzo dependiente de la longitud de la solución codificada y de la calidad de la misma, representada por la exactitud de Laplace, la medida de *fitness* utilizada por el algoritmo GBAP:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot Q \cdot LaplaceAccuracy \tag{5.6}$$

donde:

- Q es una medida para favorecer las soluciones comprensibles, calculada como se indica en la Sección 4.4.

Del mismo modo que en el algoritmo GBAP, una vez que en cada generación se ha producido el refuerzo y la evaporación, un proceso de normalización tiene lugar para limitar los niveles de feromonas a los niveles $[\tau_{min}, \tau_{max}]$. Además, al comienzo del algoritmo todas las transiciones se inicializan con la cantidad máxima de feromonas permitida.

5.4. *Función de* fitness *multiobjetivo*

La calidad de los individuos que se generan en el algoritmo MOGBAP se calcula en base a tres objetivos en conflicto: sensibilidad, especificidad y comprensibilidad.

La **sensibilidad** y la **especificidad** son dos medidas ampliamente utilizadas en problemas de clasificación, incluso como una función escalar de agregación, tal como hace por ejemplo el algoritmo Ant-Miner (ver Ecuación 3.12). La sensibilidad indica cómo de bien la regla identifica los casos positivos, mientras que la especificidad permite conocer la efectividad de la regla identificando tanto los casos negativos como los que no pertenecen a la clase estudiada. Si el valor de sensibilidad de una regla se ve incrementado, esta será capaz de predecir un mayor número de instancias positivas, aunque normalmente a expensas de clasificar como positivos ejemplos que de hecho pertenecen a la clase negativa. MOGBAP se esfuerza por maximizar ambos objetivos simultáneamente.

Ya que el algoritmo MOGBAP está enfocado a obtener un clasificador basado en reglas en forma de lista de decisión, además de alcanzar un buen poder de generalización, se pretende que las reglas extraídas sean lo más interpretables posibles. Por tanto, de alguna manera, es necesario minimizar su complejidad. Aunque la comprensibilidad es un concepto subjetivo, existen diversas formas de medir la comprensibilidad de las reglas y del clasificador final, normalmente contando el número de condiciones por regla y el número de reglas que aparecen en el clasificador. Esta última opción no se puede considerar como objetivo a optimizar en MOGBAP, ya que sigue el enfoque de codificación de individuos $hormiga = regla$, como se explicó en la Sección 4.1. Por otro lado, si el número de condiciones por regla se emplea directamente como objetivo a optimizar, habría que minimizar dicho número. No obstante, suponiendo que una determinada regla puede tener hasta un máximo fijado de condiciones, la **comprensibilidad** se puede calcular como:

$$Comprehensibility = 1 - \frac{numConditions}{maxConditions} \tag{5.7}$$

donde:

- $numConditions$ es igual al número de condiciones que aparecen en la regla codificada por el individuo, y
- $maxConditions$ es el número máximo de condiciones que puede tener una regla [56].

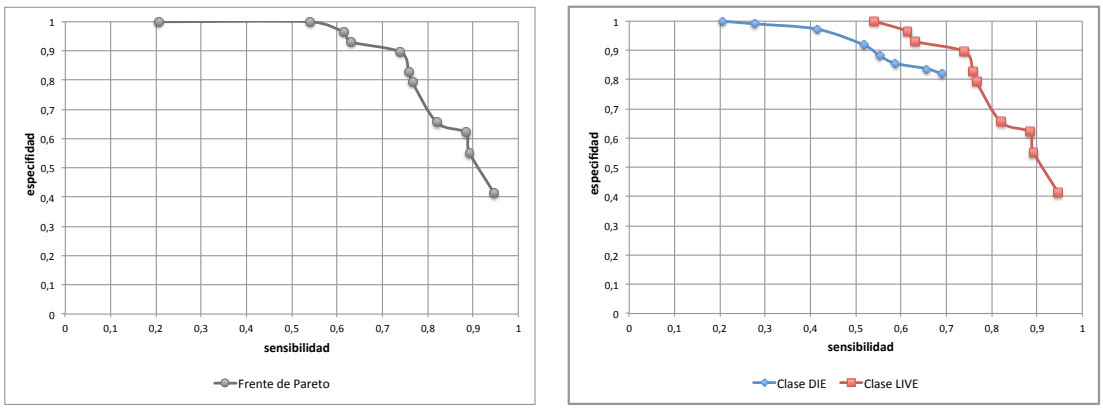
Este número máximo de condiciones posible para una regla se puede hallar sin dificultad en el algoritmo MOGBAP, dado que la gramática se establece de antemano y el número máximo de derivaciones permitidas también es conocido. La ventaja de utilizar esta métrica de comprensibilidad estriba en que sus valores van a estar comprendidos en el intervalo [0,1], y cuanto más cerca esté una regla del límite superior del intervalo, más comprensible será. Por tanto, al igual que con los objetivos de sensibilidad y especificidad, también hay que maximizar este objetivo. En la Sección 5.8.1, se analizan dos versiones de MOGBAP, una optimizando sólo sensibilidad y especificidad, y la otra considerando simultáneamente estos objetivos junto con la comprensibilidad.

Considerando el caso de tres objetivos, cabe decir que una determinada regla ant_i se dice que domina a otra ant_j, denotado por $ant_i \succ ant_j$, si ant_i no es peor que ant_j en dos objetivos y es mejor en al menos un objetivo.

5.5. *Estrategia de los k frentes de Pareto*

La idea subyacente a la estrategia de evaluación multiobjetivo que se propone para MOGBAP es la siguiente: resulta más conveniente **distinguir las soluciones o individuos en función de la clase que predicen**, debido a que ciertas clases son más difíciles de predecir que otras. De hecho, si individuos pertenecientes a diferentes clases se ordenan de acuerdo a la dominancia de Pareto, puede producirse solapamiento, como se aprecia en las Figuras 5.2, 5.3 y 5.4, para los conjuntos de datos *hepatitis*, *breast-cancer* y *lymphography*, respectivamente (considerando la maximización de la sensibilidad y la especificidad de los individuos). Por tanto, en el enfoque propuesto se encuentran ***k* frentes de Pareto, uno por cada clase**.

La estrategia multiobjetivo de los k frentes de Pareto que ha sido diseñada para MOGBAP puede ser aplicada en cualquier EA o algoritmo bioinspirado de clasificación multiobjetivo.



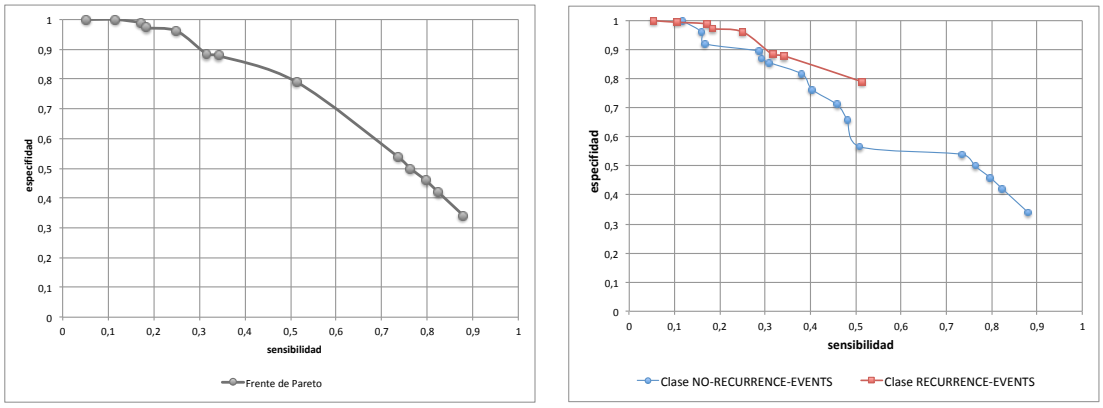
(a) Enfoque clásico (b) Enfoque de los k-frentes de Pareto

Figura 5.2: Comparación entre un enfoque clásico y el de los k-frentes de Pareto sobre el conjunto de datos binario *hepatitis*

Suponiendo el problema de *hepatitis*, si se empleara un enfoque clásico de ordenación en frentes de Pareto, el primer frente de soluciones no dominadas sería el mostrado en la Figura 5.2(a). Como se puede apreciar, dicho frente estaría compuesto por todos los individuos de la clase 'LIVE' y por un único individuo perteneciente a la clase 'DIE' (aquel con especificidad de 1,0). Por tanto, para considerar el resto de individuos de esta clase, sería necesario encontrar frentes adicionales, cuyos individuos tendrían menor probabilidad de formar parte de la lista de decisión del clasificador final. Sin embargo, el enfoque multiobjetivo aquí propuesto (ver Figura 5.2(b)) encuentra un conjunto de individuos no dominados por cada clase, con lo que garantiza la inclusión de reglas prediciendo cada una de ellas en el clasificador final.

Se puede argumentar un comportamiento similar para la Figura 5.3, donde muchas de las reglas no dominadas que predicen la clase 'RECURRENCE-EVENTS' quedarían ocultas en caso de seguir un enfoque multiobjetivo clásico donde se consideraran los individuos de ambas clases a la vez y, por tanto, se encontraría un único frente de Pareto.

Centrándonos ahora en la Figura 5.4, que representa los frentes de Pareto encontrados para el conjunto de datos *lymphography*, se puede ver que si se considerasen simultáneamente todos los individuos generados en el algoritmo con independencia del tipo de clase que predicen en su consecuente, el frente de Pareto estaría compuesto únicamente por un sólo punto, aquel con sensibilidad y especificidad



(a) Enfoque clásico (b) Enfoque de los k-frentes de Pareto

Figura 5.3: Comparación entre un enfoque clásico y el de los k-frentes de Pareto sobre el conjunto de datos binario *breast-cancer*

máxima (1,0). De hecho, para llegar a contemplar individuos prediciendo la clase 'MALIGN_LYMPH', previamente sería necesario encontrar como mínimo otros tres frentes, dado que la figura sólo muestra las soluciones que pertenecen al frente de Pareto de cada clase, con lo que hasta llegar al frente de Pareto de la clase 'MALIGN_LYMPH' pueden existir soluciones dominadas de otras clases que conformen frentes intermedios.

Resumiendo, el enfoque multiobjetivo propuesto para el algoritmo MOGBAP consiste en descubrir un conjunto de soluciones no dominadas para cada clase existente en el conjunto de datos. Para ello, una vez que los individuos de la generación actual han sido creados y evaluados por cada objetivo considerado, se separan en k grupos en función del consecuente que predicen, y donde k es el número de clases del conjunto de entrenamiento. Entonces, cada grupo de individuos se combina con las soluciones almacenadas en la generación anterior correspondientes al frente de Pareto de dicha clase, y se ordenan todos en base a la dominancia, encontrando un nuevo frente de Pareto. Así pues, en cada iteración del algoritmo se encuentran k frentes de Pareto, y únicamente las soluciones contenidas en ellos participarán en el refuerzo de feromonas.

El clasificador final se crea a partir de los individuos que existen en los k frentes de Pareto encontrados en la última generación del algoritmo. Para seleccionar los individuos más apropiados, se lleva a cabo un **enfoque de nichos sobre cada frente**, como se describe en la Sección 5.6.

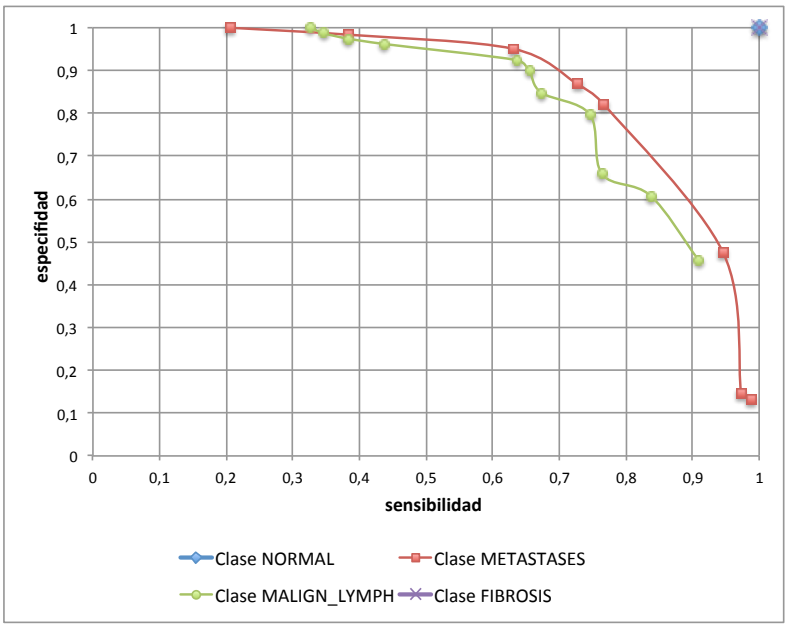


Figura 5.4: Enfoque de los k-frentes de Pareto sobre el conjunto de datos multiclase *lymphography*

La decisión de enfocar la optimización multiobjetivo considerando un frente de Pareto para cada clase fue adoptada tras comprobar que su rendimiento es superior al obtenido mediante un procedimiento de ordenación de individuos basado en la dominancia similar al empleado en el algoritmo NSGA-II [55]. En este último no se hace distinción entre los individuos por la clase que predicen, por lo que el frente de Pareto contendrá individuos prediciendo distintas clases.

El enfoque de los k frentes de Pareto propuesto puede encuadrarse dentro de los algoritmos de MOO **progresivos**, por lo que se buscan múltiples soluciones óptimas de Pareto. Esto hace que surjan cuestiones en cuanto al modo en que se van a almacenar las soluciones de Pareto durante el proceso de búsqueda. Entre las diversas formas que se proponen en la literatura de MOACO [12], el algoritmo MOGBAP sigue una **estrategia de almacenamiento en línea** porque, aunque se almacenan para ser utilizadas cuando finaliza la ejecución del algoritmo, también se usan durante la evolución del mismo:

- Se emplean durante la ejecución del algoritmo para actualizar las tasas de feromonas en el entorno.
- Se usan vez que todas las generaciones han terminado para seleccionar los individuos que formarán parte del clasificador final.

5.6. Algoritmo

Los pasos principales del algoritmo MOGBAP se muestran en el pseudocódigo del algoritmo 4, que comienza inicializando la gramática e insertando el estado inicial en el espacio de estados. También crea un objeto vacío que representa el clasificador, así como una tabla hash para almacenar los frentes de Pareto de cada generación. El número de derivaciones se establece al mínimo necesario que permite encontrar un antecedente para las reglas, y se calcula el paso de derivación para cada generación. En el caso de la gramática definida serán necesarias al menos dos derivaciones para alcanzar una solución a partir del estado inicial.

Cuando empieza cada generación, se inicializa una tabla hash $ants$, que tendrá tantas entradas como clases tenga el conjunto de datos. Cada entrada estará relacionada con una lista de hormigas que contendrá las hormigas creadas para la clase correspondiente durante la generación actual. A continuación el algoritmo crea $numAnts$ individuos y asigna a cada uno de ellos un consecuente, para lo que se calcula el porcentaje de instancias de cada clase cubiertas por el antecedente en el $trainingSet$, y se selecciona la clase más cubierta en proporción. El algoritmo evalúa cada hormiga de acuerdo a las funciones objetivo y la añade a la lista de hormigas almacenada en la tabla hash $ants$ que corresponda a su consecuente.

El siguiente paso realizado por el algoritmo se repite por cada categoría del conjunto de datos: los individuos creados para la categoría actual se juntan con aquellos no dominados procedentes de la generación previa que también predecían esta clase. Se encuentra un nuevo frente de Pareto a partir de ambos grupos de individuos, y las soluciones del frente reforzarán la concentración de feromonas en el camino que siguen. Una vez que se ha encontrado un frente de Pareto para cada categoría, tienen lugar los procesos de evaporación y normalización, incrementándose también el número máximo de derivaciones en el paso de derivación.

Algoritmo 4 Pseudocódigo de MOGBAP

Require: $trainingSet, testSet, numGenerations, numAnts, maxDerivations$
1: Inicializar gramática a partir de $trainingSet$ e inicializar espacio de estados
2: Inicializar clasificador
3: Crear tabla hash $paretoFronts \leftarrow \{\}$
4: $derivationStep \leftarrow \frac{maxDerivations-2}{numGenerations}$
5: $maxDerivations \leftarrow 2$
6: **for** $i \leftarrow 0$ **to** $i < numGenerations$ **inc** 1 **do**
7: Crear tabla hash $ants \leftarrow \{\}$
8: Crear lista $categories \leftarrow$ Clases en $trainingSet$
9: **for each** $category$ **in** $categories$ **do**
10: Crear lista $antList \leftarrow \{\}$
11: Introducir entrada $(category, antList)$ en tabla hash $ants$
12: **end for**
13: **for** $j \leftarrow 0$ **to** $j < numAnts$ **inc** 1 **do**
14: $ant \leftarrow$ Crear nueva hormiga (ver Procedimiento 2)
15: Almacenar los estados del camino seguido por ant en el espacio de estados
16: $consequent \leftarrow$ Asignar un consecuente a la hormiga
17: Evaluar ant para cada función objetivo
18: $antList \leftarrow$ Recuperar valor de entrada con clave $consequent$ de la tabla $ants$
19: Añadir ant a la lista $antList$
20: Introducir entrada $(consequent, antList)$ en tabla hash $ants$
21: **end for**
22: **for each** $category$ **in** $categories$ **do**
23: Crear lista $aux \leftarrow \{\}$
24: Añadir a lista aux los individuos de las listas obtenidas de las tablas hash $ants$ y $paretoFronts$, dada la clave $category$, comprobando duplicidad
25: Crear lista $newParetoFront \leftarrow$ Soluciones no dominadas en aux (ver Sección 5.5)
26: **for each** ant **in** $newParetoFront$ **do**
27: Actualizar la tasa de feromonas en el camino seguido por ant de manera proporcional a la accuracy de Laplace e inversamente proporcional a la longitud de la solución que codifica
28: **end for**
29: Introducir entrada $(category, newParetoFront)$ en tabla hash $paretoFronts$
30: **end for**
31: Evaporar las feromonas del espacio de estados
32: Normalizar los valores de feromonas
33: $maxDerivations \leftarrow maxDerivations + derivationStep$
34: **end for**
35: **for each** $category$ **in** $categories$ **do**
36: Crear lista $paretoFront \leftarrow$ Recuperar lista de tabla hash $paretoFronts$ a partir clave $category$
37: Añadir al clasificador hormigas $winnerAnts$ devueltas tras llevar a cabo el procedimiento de nichos sobre las hormigas de la lista $paretoFront$ (ver Procedimiento 5)
38: **end for**
39: Establecer la regla por defecto en $classifier$
40: $predictiveAccuracy \leftarrow$ Calcular la exactitud predictiva obtenida al ejecutar sobre $testSet$ el clasificador construido
41: **return** $predictiveAccuracy$

Procedimiento 5 Enfoque de nichos

Require: $ants, cat, percentageOfCoverage, trainingSet$
Crear lista $winnerAnts \leftarrow \{\}$
$numInstances \leftarrow$ número de instancias en $trainingSet$
$numInstancesClass \leftarrow$ número de instancias de la clase cat en $trainingSet$
Ordenar lista $ants$ según $LaplaceAccuracy$
$flagsArray \leftarrow$ array binario de $numInstances$ elementos
for $i \leftarrow 0$ **to** $i < numInstances$ **inc** 1 **do**
    $flagsArray[i] \leftarrow$ **false**
end for
for each ant **in** $ants$ **do**
    $idealTokens \leftarrow 0$
    $capturedTokens \leftarrow 0$
    for $j \leftarrow 0$ **to** $j < numInstances$ **inc** 1 **do**
        $instance \leftarrow$ Instancia j de $trainingSet$
        $catInstance \leftarrow$ Categoría de $instance$
        if ant cubre $instance$ **then**
            $idealTokens \leftarrow idealTokens + 1$
            if $flagsArray[j] \leftarrow$ **false and** $catInstance = cat$ **then**
                $flagsArray[j] \leftarrow$ **true**
                $capturedTokens \leftarrow capturedTokens + 1$
            end if
        end if
    end for
    if $capturedTokens >= (percentageOfCoverage \cdot numInstancesClass)$ **then**
        $LaplaceAccuracy_{adj} \leftarrow LaplaceAccuracy \cdot \frac{capturedTokens}{idealTokens}$
        Añadir ant a $winnerAnts$
    else
        $LaplaceAccuracy_{adj} \leftarrow 0$
    end if
end for
return $winnerAnts$

Tras finalizar el bucle principal del algoritmo, después de realizar $numGenerations$ generaciones, hay que seleccionar un subconjunto de hormigas de cada frente de Pareto final por medio del enfoque de nichos descrito en el Procedimiento 5. En él, las hormigas se ordenan de acuerdo a su exactitud de Laplace y entonces compiten por capturar tantas instancias, aquí denominadas *tokens*, como sean capaces. Una hormiga puede capturar un *token* en caso de que lo cubra y que no haya sido capturado previamente por otra hormiga. Por último, sólo las hormigas cuyo número de *tokens* capturados supere el porcentaje de cubrimiento mínimo establecido por el usuario se añaden a la lista $winnerAnts$, teniendo una exactitud de Laplace

ajustada calculada como sigue:

$$LaplaceAccuracy_{adj} = LaplaceAccuracy \cdot \frac{capturedTokens}{idealTokens} \tag{5.8}$$

donde:

$idealTokens$ representa el número de *tokens* que cubre la hormiga (independientemente de que los haya capturado o no).

Una vez que el enfoque de nichos se lleva a cabo sobre los individuos de cada frente de Pareto, las hormigas resultantes se añaden al clasificador. La lista de decisión del clasificador final se establece con estas hormigas, ordenándolas según su exactitud de Laplace ajustada en orden descendente. Finalmente, la regla por defecto prediciendo la clase mayoritaria del $trainingSet$ se añade al final de la lista de decisión y el clasificador se ejecuta sobre $testSet$ para calcular su exactitud predictiva.

5.7. Experimentación

En esta sección se introducen los conjuntos de datos que han sido utilizados en el estudio experimental y cómo han sido preprocesados antes de lanzar las ejecuciones de los algoritmos. Asimismo, se describen los algoritmos que toman parte en el estudio y su configuración de parámetros.

5.7.1. Conjuntos de datos y preprocesado

El estudio experimental ha sido realizado utilizando 15 conjuntos de datos públicos del repositorio de la Universidad de California en Irvine (UCI) [82]. Dicha colección de conjuntos de datos comprende problemas con dimensionalidad variada. Sus características particulares se pueden observar en la Tabla 5.1.

Se ha llevado a cabo un procedimiento de validación cruzada estratificada con diez particiones (*10-fold cross validation*), donde cada partición contiene aproximadamente la misma proporción de instancias por clase existente en el conjunto de

Conjunto de Datos	Valores Perdidos	Instancias	Atributos			Clases	Distribución de Clases
			Cont.	*Bin.*	*Nom.*		
Hepatitis	yes	155	6	13	0	2	32 / 123
Sonar	no	208	60	0	0	2	97 / 111
Breast-c	yes	286	0	3	6	2	201 / 85
Heart-c	yes	303	6	3	4	2	165 / 138
Ionosphere	no	351	33	1	0	2	126 / 225
Horse-c	yes	368	7	2	13	2	232 / 136
Vote	no	435	6	0	17	2	267 / 168
Australian	yes	690	6	0	9	2	307 / 383
Breast-w	yes	699	9	0	0	2	458 / 241
Credit-g	no	1000	6	3	11	2	700 / 300
Iris	no	150	4	0	0	3	50 / 50 / 50
Wine	no	178	13	0	0	3	59 / 71 / 48
Lymphography	no	148	3	9	6	4	2 / 81 / 61 / 4
Glass	no	214	9	0	0	6	70 / 76 / 17 / 13 / 9 / 29
Primary-tumor	yes	339	0	14	3	21	84 / 20 / 9 / 14 / 39 / 1 / 14 / 6 / 2 / 28 / 16 / 7 / 24 / 2 / 1 / 10 / 29 / 6 / 2 / 1 / 24

Tabla 5.1: Características de los conjuntos de datos

datos original. Cada algoritmo se ejecuta diez veces con una partición diferente excluida cada una de las veces y empleada como conjunto de *test*, mientras que las otras nueve conformarían el conjunto utilizado para entrenamiento. La exactitud predictiva del clasificador se calcula considerando la exactitud media de los diez experimentos, tal como se indica en la Ecuación 4.13.

Hay que mencionar que para evaluar el rendimiento de los algoritmos no deterministas se han empleado diez semillas diferentes, con lo que se ha repetido el procedimiento de validación cruzada diez veces, con el objetivo de evitar cualquier posibilidad de obtener resultados sesgados por el valor de una única semilla de generación de números aleatorios. Así pues, estos algoritmos han sido ejecutados un total de cien veces por cada conjunto de datos.

Por cada conjunto de datos conteniendo valores perdidos, dichos valores han sido sustituidos por la moda (en el caso de atributos nominales) o la media aritmética

(en el caso de numéricos). Además, dado que varios de los algoritmos no son capaces de procesar los atributos numéricos directamente, aplicando el algoritmo de discretización de Fayyad e Irani [73] (explicado en la Sección A.3), cada conjunto de entrenamiento ha sido discretizado, empleando a continuación los intervalos discretos encontrados al discretizar su respectivo conjunto de test. Esta separación es necesaria para poder obtener resultados fiables, porque si se aplicase el algoritmo sobre el conjunto de datos completo antes de generar las particiones para la validación cruzada, el algoritmo tendría en cuenta los datos de test, lo que invalidaría los resultados obtenidos. Ambas acciones de preprocesado se efectuaron haciendo uso de la herramienta Weka.

5.7.2. Algoritmos y configuración de parámetros

En esta sección se presentan los algoritmos utilizados en el estudio experimental, indicando los valores empleados para configurar sus parámetros en la Tabla 5.2. La comparativa considera algunos algoritmos de inducción de reglas de clasificación representativos pertenecientes a diversos paradigmas:

- AP. Además del algoritmo presentado en este capítulo, MOGBAP, también se usa su precursor, GBAP, explicado en el Capítulo 4.
- ACO. Se han considerado tres algoritmos de esta metaheurística, Ant-Miner, cAnt-Miner2-MDL y Ant-Miner+. Además, también se incluye en la experimentación el híbrido PSO/ACO2. Todos estos algoritmos fueron introducidos en la Sección 3.3.2.
- GP. Bojarczuk-GP, guiado por gramática (Sección 3.2.1).
- *Reduced error pruning*. JRIP, la implementación Weka del algoritmo de cubrimiento secuencial RIPPER (Sección 2.1.4.1).
- Árboles de decisión. PART, que extrae reglas de decisión a partir de la implementación Weka del algoritmo J4.8 (Sección 2.1.4.1).

Algoritmo	Parámetro	Descripción	Valor
MOGBAP	numAnts	Número de hormigas	20
&	numGenerations	Número de generaciones	100
GBAP	maxDerivations	Número máximo de derivaciones para la gramática	15
	percentageOfCoverage	Porcentaje de cubrimiento mínimo de instancias	3
		de la clase predicha por la regla (MOGBAP)	3
	minCasesPerRule	Número mínimo de instancias cubiertas por regla (GBAP)	3
	$[\tau_0]$	*Cantidad de feromonas inicial en las transiciones*	*1,0*
	$[\tau_{min}]$	*Cantidad mínima de feromonas permitida*	*0,1*
	$[\tau_{max}]$	*Cantidad máxima de feromonas permitida*	*1,0*
	$[\rho]$	*Tasa de evaporación*	*0,05*
	$[\alpha]$	*Exponente para la función heurística*	*0,4*
	$[\beta]$	*Exponente para las feromonas*	*1,0*
Ant-Miner	number of ants	Número de hormigas	1
	min. cases per rule	Número mínimo de instancias cubiertas por regla	10
	max. uncovered cases	Número máximo de instancias no cubiertas	10
	rules for convergence	Número de reglas para convergencia	10
	number of iterations	Número máximo de iteraciones	3000
cAnt-Miner2	number of ants	Número de hormigas	60
-MDL	min. cases per rule	Número mínimo de instancias cubiertas por regla	5
	max. uncovered cases	Número máximo de instancias no cubiertas	10
	rules for convergence	Número de reglas para convergencia	10
	number of iterations	Número máximo de iteraciones	1500
Ant-Miner+	nAnts	Número de hormigas	1000
	rho	Tasa de evaporación	0,85
PSO/ACO2	numParticles	Número de partículas	10
	numIterations	Número de iteraciones	200
	maxUncovExampPerClass	Número máximo de ejemplos no cubiertos por clase	2
GP	population-size	Número de individuos	200
	max-of-generations	Número de generaciones	100
	max-deriv-size	Número máximo de derivaciones para la gramática	20
	recombination-prob	Probabilidad de cruce	0,8
	reproduction-prob	Probabilidad de reproducción	0,05
	parents-selector	Método de selección de padres	Roulette
JRIP	checkErrorRate	Si se incluye comprobación para tasa de error >= 1/2 en	True
		criterio de parada	
	folds	Determina las particiones usadas para la poda. Una	3
		se usa para poda, el resto para hacer crecer las reglas	
	minNo	Mínimo de peso total de las instancias en una regla	2,0
	optimizations	Número de ejecuciones para optimización	2
	pruning	Si se realiza poda	True
PART	binarySplits	Indica si se usan particiones binarias en atributos	False
		nominales al construir los árboles parciales	
	confidenceFactor	Factor de confianza utilizado para la poda	0,25
		(valores menores suponen mayor poda)	
	minNumObj	Número mínimo de instancias por regla	2
	numFolds	Determina la cantidad de datos para poda de	3
		error-reducida. Una partición se usa para poda, el resto	
		para hacer crecer las reglas	
	reducedErrorPruning	Si se usa poda de error reducida en lugar de la del C4.5	False
	unpruned	Si no se realiza poda	False

Tabla 5.2: Configuración de parámetros configurables por el usuario

En esta experimentación hemos querido comprobar, además, si el hecho de que algoritmos capaces de tratar directamente con atributos numéricos utilicen únicamente las particiones discretizadas podría ir en detrimento de su rendimiento. Por tanto, hemos llevado a cabo los experimentos respectivos por duplicado, como si considerásemos dos algoritmos diferentes por cada uno de ellos: por un lado utilizando las particiones originales conteniendo valores continuos y, por otro, empleando las particiones discretizadas. Para diferenciar entre ambos casos, el sufijo 'D' se ha añadido al nombre de cada algoritmo cuando se ha ejecutado sobre los datos discretizados.

La Tabla 5.2 recoge los valores empleados para los parámetros de cada algoritmo. MOGBAP y GBAP comparten los mismos parámetros, a excepción de *percentageOfCoverage*, que remplaza el parámetro *minCasesPerRule* que existía en GBAP. Así pues, empleamos la misma configuración de parámetros en ambos algoritmos. Como se puede observar, los dos tienen cuatro parámetros obligatorios, mientras que los otros seis, representados entre corchetes, son opcionales. Son opcionales puesto que en el estudio de sensibilidad de parámetros del algoritmo GBAP (Sección 4.6.4) se determinaron como óptimos. Para el algoritmo MOGBAP se han mantenido los mismos valores.

Hay que resaltar que no se ha efectuado optimización alguna para los valores de los parámetros de MOGBAP. Por consiguiente, sería posible ajustar los mismos para cada conjunto de datos particular, de manera que se obtuviesen mejores resultados.

5.8. Resultados

En esta sección se muestran los resultados obtenidos en los estudios experimentales. Inicialmente se estudian dos versiones del algoritmo MOGBAP, una optimizando simultáneamente dos objetivos y la otra tres. La versión que mejores resultados obtiene es la que se utiliza en los dos estudios siguientes, que versan sobre la exactitud predictiva y la comprensibilidad.

5.8.1. Comparación entre enfoques optimizando dos y tres objetivos

Antes de entrar en el estudio experimental propiamente dicho, se ha efectuado un análisis preliminar para determinar los objetivos a optimizar por MOGBAP. Así, se han considerado dos versiones: una primera, donde el objetivo es la maximización de dos objetivos, sensibilidad y especificidad; y una segunda, que considera además la maximización de la medida de comprensibilidad explicada en la Ecuación 5.7.

	MOGBAP 2-objetivos						MOGBAP 3-objetivos					
Data set	Acc_{tra}	σ_{tra}	Acc_{tst}	σ_{tst}	#R	#C/R	Acc_{tra}	σ_{tra}	Acc_{tst}	σ_{tst}	#R	#C/R
Hepatitis	89,00	1,40	84,59	10,93	8,7	2,24	89,33	1,52	85,15	11,51	9,1	1,99
Sonar	86,45	1,63	78,20	9,63	11,2	2,52	86,72	1,52	79,49	9,26	11,1	2,04
Breast-c	76,05	1,58	72,45	8,72	12,0	1,88	76,04	1,73	72,02	9,62	11,8	1,69
Heart-c	85,15	1,14	81,11	4,42	10,3	2,22	85,31	1,09	83,13	4,24	9,9	2,25
Ionos.	91,23	0,77	90,46	5,52	6,6	1,61	91,25	0,71	90,55	5,67	6,8	1,49
Horse-c	85,81	1,02	84,66	4,62	10,5	2,14	85,56	1,12	83,78	4,67	9,6	2,03
Vote	96,10	0,59	95,05	2,85	6,8	2,35	96,10	0,59	94,89	2,92	6,6	2,12
Australian	87,67	0,84	88,10	4,10	8,9	2,17	87,71	0,81	87,38	4,27	9,1	2,00
Breast-w	94,81	0,72	94,58	2,72	6,0	1,95	95,10	0,58	95,41	2,31	6,1	1,77
Credit-g	73,59	0,93	71,68	3,13	11,4	2,03	74,02	0,99	70,82	3,33	11,6	1,82
Iris	96,00	0,59	95,33	6,00	5,8	1,15	96,00	0,59	95,33	6,00	5,8	1,15
Wine	98,31	0,88	97,11	4,07	6,1	1,65	98,86	0,71	98,24	2,75	6,1	1,47
Lymph.	87,44	1,98	79,97	10,53	12,4	1,86	87,03	1,83	80,55	9,74	11,9	1,55
Glass	74,55	2,28	68,29	9,40	18,4	2,36	76,86	2,20	71,03	8,45	17,5	2,22
Primary	50,52	1,39	41,83	6,79	35,6	2,89	50,16	1,40	42,18	7,19	34,9	2,53
Avg	84,85	1,18	81,56	6,23	11,38	2,07	84,67	1,16	82,00	6,13	11,19	1,87

Tabla 5.3: Resultados obtenidos por las versiones de MOGBAP optimizando dos y tres objetivos

La Tabla 5.3 muestra los resultados obtenidos por cada versión. En ambos casos se puede observar que los resultados de exactitud predictiva obtenidos en entrenamiento son superiores a los de test, como cabía esperar. La versión de tres objetivos obtiene, no obstante, valores de *accuracy* superiores y extrae un modelo más comprensible, como se deduce a partir de los valores medios de número de reglas en el

clasificador y de número de condiciones por regla, que son inferiores al caso de dos objetivos. Sin embargo, este análisis no permite extraer conclusiones determinantes, al estar basado en la media de los valores absolutos de diversas medidas sobre diferentes conjuntos de datos que además tienen características variadas.

Por tanto, hemos procedido realizando el test de ranking de signos de Wilcoxon con la corrección de continuidad sobre los resultados de exactitud predictiva en test en ambas versiones. Los resultados indican que, a un nivel de significación de α=0.05 no se aprecian diferencias significativas entre ambas versiones. También hemos analizado estadísticamente las diferencias que presentan ambas versiones en cuanto al número de condiciones por regla. El *p-value* obtenido fue igual a 0,001363, que es inferior a 0,01. Esto prueba que la versión de MOGBAP optimizando tres objetivos es significativamente más comprensible que la versión de dos objetivos con una probabilidad del 99 %.

Como conclusión, adoptamos como versión definitiva de nuestro algoritmo el modelo de MOGBAP que optimiza simultáneamente sensibilidad, especificidad y comprensibilidad, y es el que hemos utilizado en el estudio experimental descrito en la siguiente sección.

5.8.2. *Estudio de exactitud predictiva*

En esta sección se compara el rendimiento del algoritmo propuesto frente a otros algoritmos de inducción de reglas de clasificación, centrándonos en los resultados de exactitud predictiva que alcanzan. Las Tablas 5.4 y 5.5 recogen los resultados de *accuracy* medios, con la desviación típica, obtenidos por cada algoritmo sobre cada conjunto de datos. Cuando un valor se encuentra marcado en negrita indica que se trata del mejor resultado alcanzado para ese conjunto de datos particular, y se corresponderá con el algoritmo de la cabecera de su columna. Debido a la extensión del estudio experimental, lo hemos separado en dos tablas, considerando por un lado los algoritmos que tratan con los conjuntos de datos discretizados (Tabla 5.4) y por otro los que trabajan con los datos continuos (Tabla 5.5). Nótese que, a pesar de esta división en dos tablas, no se han realizado los test estadísticos en dos grupos, sino que se han considerado todos los algoritmos a la vez.

	MOGBAP		GBAP		Ant-Miner		Ant-Miner+		cAnt-Miner2$_D$		PSO/ ACO2$_D$		GP$_D$		JRIP$_D$		PART$_D$	
Data set	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}
Hepatitis	**85,15**	1,52	82,17	12,04	83,27	10,32	81,79	10,30	84,89	1,27	84,59	9,33	71,05	14,45	81,54	12,05	84,64	7,66
Sonar	79,49	9,26	**81,98**	7,44	76,95	6,89	76,05	7,22	76,75	10,76	78,49	8,05	79,82	9,24	80,33	6,61	77,84	8,10
Breast-c	72,02	9,62	71,40	7,86	73,42	7,29	73,05	6,86	**74,15**	1,22	68,63	6,87	68,63	10,94	72,00	6,41	68,48	7,90
Heart-c	**83,13**	4,24	82,84	5,24	78,01	6,69	82,41	5,10	75,54	9,77	82,25	5,36	70,02	7,08	82,20	5,12	80,13	6,39
Ionos.	90,55	5,67	**93,02**	4,07	84,39	6,73	92,89	4,02	86,17	10,67	89,97	4,99	76,48	8,19	91,70	5,14	88,93	4,02
Horse-c	**83,78**	4,67	82,97	6,34	82,71	4,73	81,79	6,03	81,98	7,12	82,06	4,93	82,52	6,06	83,72	6,35	81,5	3,72
Vote	94,89	2,92	94,37	3,57	94,29	3,27	94,66	3,72	94,33	4,86	94,30	3,81	**95,67**	2,78	95,44	3,52	94,51	3,08
Australian	**87,38**	4,27	85,47	4,49	85,30	4,12	83,48	3,38	85,38	4,5	85,19	4,69	85,52	4,50	86,70	5,15	84,66	4,48
Breast-w	95,41	2,31	**96,50**	1,68	94,69	2,04	94,28	2,86	94,44	1,79	95,86	1,91	87,39	2,75	95,71	1,81	95,71	1,82
Credit-g	70,82	3,33	70,79	4,27	70,55	3,72	70,80	3,87	70,65	5,67	70,36	3,55	63,02	7,03	70,70	3,26	72,70	3,26
Iris	95,33	6,00	**96,00**	4,10	95,20	5,47	94,00	3,59	95,33	7,83	95,33	6,70	91,73	10,46	**96,00**	5,33	95,33	6,70
Wine	**98,24**	2,75	97,01	4,37	91,86	5,08	93,86	4,61	92,48	7,76	90,20	2,86	83,69	9,44	95,61	5,37	95,03	3,89
Lymph.	80,55	9,74	**81,00**	10,35	75,51	9,59	77,23	10,91	74,15	18,34	76,59	12,20	77,78	12,77	78,84	11,49	78,43	14,30
Glass	71,03	8,45	69,13	8,66	65,52	9,26	62,03	9,80	64,51	14,51	71,16	10,54	39,23	11,34	69,00	8,70	**73,91**	8,43
Primary	**42,18**	7,19	37,91	6,55	37,75	5,27	37,26	5,43	36,21	9,23	37,19	5,88	16,41	4,96	38,11	3,75	38,36	5,09
Ranking	**3,0667**		4,0333		8,8		8,1		8,9		7,7333		10,3		4,1667		6,5333	

Tabla 5.4: Resultados comparativos de exactitud predictiva (%) en test sobre conjuntos de datos discretizados

	cAnt-Miner2		PSO/ ACO2		GP		JRIP		PART	
Data set	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}
Hepatitis	79,93	10,61	85,03	13,81	73,72	11,40	80,95	9,96	78,08	13,99
Sonar	73,86	8,83	74,56	9,75	79,11	8,49	78,80	13,99	75,49	6,89
Breast-c	**74,15**	1,22	68,63	6,87	68,63	10,94	72,00	6,41	68,48	7,90
Heart-c	77,33	6,77	80,68	7,00	73,57	8,24	76,59	5,26	80,93	7,38
Ionos.	87,59	5,49	84,82	4,88	77,98	8,57	90,03	4,57	89,17	4,68
Horse-c	82,45	8,37	80,82	6,79	76,53	8,11	83,65	7,75	80,62	7,93
Vote	94,33	4,86	94,80	3,81	**95,67**	2,78	95,44	3,52	94,51	3,08
Australian	85,42	3,81	84,88	3,27	85,51	2,04	86,03	2,87	84,49	3,85
Breast-w	95,03	2,30	96,02	1,38	91,61	4,08	95,56	2,60	94,70	3,00
Credit-g	71,20	4,53	70,04	3,65	65,76	7,66	**73,50**	4,57	70,50	4,50
Iris	93,93	5,53	95,80	6,07	89,59	11,36	95,33	4,27	94,00	6,29
Wine	89,23	8,04	88,87	7,15	80,46	11,97	93,22	6,11	90,96	4,53
Lymph.	75,23	11,02	79,28	8,03	77,66	19,04	77,76	9,26	78,15	10,48
Glass	66,24	10,86	68,91	10,34	38,78	11,67	65,70	8,37	68,50	10,65
Primary	36,21	9,23	37,19	5,88	16,41	4,96	38,11	3,75	38,36	5,09
Ranking	9,1667		8,0		11,0333		5,9		9,2666	

Tabla 5.5: Resultados comparativos de exactitud predictiva (%) en test sobre conjuntos de datos sin discretizar

En primer lugar, para evaluar estadísticamente las diferencias existentes entre los clasificadores, llevamos a cabo el test de Iman&Davenport, que compara los rangos medios de k algoritmos sobre N conjuntos de datos. Se pueden observar los rangos de cada algoritmo en la última fila de cada tabla, con el algoritmo de control (el que mejor se comporta) resaltado en negrita.

De acuerdo con el test de Iman&Davenport, diremos que todos los algoritmos son equivalentes si se acepta la hipótesis nula. En cambio, si esta es rechazada, afirmaremos que existen diferencias significativas entre los mismos. Así, a un nivel de significación de $\alpha = 0{,}05$, el valor obtenido para el estadístico de Iman&Davenport sobre los rangos medios distribuidos con respecto a una distribución F con $k-1$ y $(k-1)\cdot(N-1)$ grados de libertad es igual a 7,1368. Este valor se encuentra

fuera del intervalo crítico $C_0 = [0, (F_F)_{0,05,13,182} = 1{,}7743]$, con lo que se rechaza la hipótesis nula con una probabilidad del 95 %.

Para desvelar con respecto a qué clasificadores MOGBAP se comporta estadísticamente mejor es necesario efectuar un test a posteriori. El test de Bonferroni-Dunn se puede aplicar dado que todos los algoritmos se comparan frente a un algoritmo de control, nuestra propuesta. El valor crítico que revela este test al mismo nivel $\alpha = 0{,}05$ es de 4,4161, lo que implica que, en cuanto a exactitud predictiva, MOGBAP es estadísticamente mejor que PSO/ACO2$_D$, PSO/ACO2, Ant-Miner+, Ant-Miner, cAnt-Miner2-MDL$_D$, cAnt-Miner2-MDL, GP, GP$_D$ y PART. Estos resultados pueden observarse en la Figura 5.5, donde también se comprueba que MOGBAP alcanza valores de *accuracy* competitivos e incluso mejores que los algoritmos GBAP, PART$_D$, JRIP$_D$ y JRIP. Estos resultados demuestran que MOGBAP es estadísticamente más exacto que el resto de los algoritmos basados en hormigas considerados, con la excepción de GBAP. Para cerciorarnos de que efectivamente no existían diferencias entre MOGBAP y GBAP, realizamos el test de rangos de Wilcoxon. El test arrojó un *p-value* de 0,3303, con lo que se puede afirmar que no existen diferencias significativas.

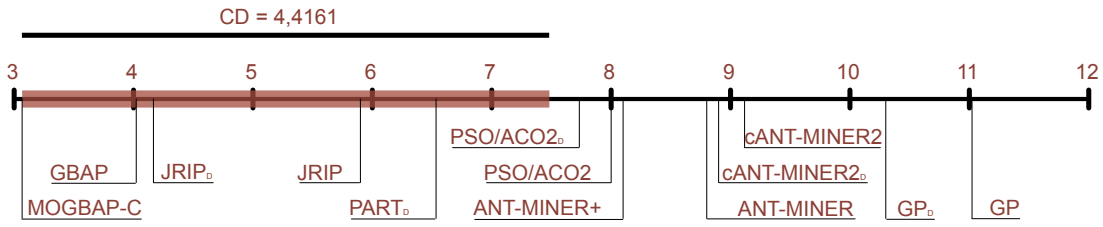


Figura 5.5: Test de Bonferroni–Dunn. MOGBAP presenta diferencias significativas con respecto a aquellos clasificadores cuyo *ranking* está fuera del intervalo sombreado ($p < 0{,}05$)

5.8.3. Estudio de comprensibilidad

Como se ha mencionado anteriormente, todos los algoritmos incluidos en el estudio son clasificadores basados en reglas. Por tanto, la complejidad del conjunto de reglas obtenido por cada algoritmo y la longitud de las reglas se pueden comparar apropiadamante. El número medio de reglas por clasificador y el número medio de condiciones por regla obtenidos por los algoritmos se presentan también en

dos tablas, la Tabla 5.6 para los algoritmos que utilizan los conjuntos de datos discretizados, y la Tabla 5.7 para los que usan los datos numéricos.

Hay que mencionar en este punto que el algoritmo de GP emplea el operador OR, con lo que a efectos de cómputo es necesario considerar que dicho operador está uniendo dos reglas diferentes, y no considerarlo como una condición más.

La Tabla 5.8 resume los resultados medios de todos los algoritmos sobre los quince conjuntos de datos con respecto a todas las medidas estudiadas: exactitud predictiva, número de reglas en el clasificador y número de condiciones por regla. No obstante, estos resultados no son relevantes estadísticamente hablando, con lo que hemos procedido del mismo modo que en la sección anterior realizando el test de Iman&Davenport para ambas medidas. La penúltima fila de las Tablas 5.6 y 5.7 muestra el rango obtenido por cada algoritmo en cuanto al número medio de reglas en el clasificador, y la última fila hace lo propio para el número medio de condiciones por regla.

El primer estudio estadístico considera el número medio de reglas en el clasificador (mientras menor sea, más comprensible será el mismo). Siguiendo con un nivel de significación de $\alpha = 0{,}05$, el valor del estadístico de Iman&Davenport de acuerdo con una distribución F es igual a 18,1522, y el intervalo crítico calculado es $[0, 1{,}7743]$. Dado que el estadístico no pertenece a este intervalo, se rechaza la hipótesis nula de que los algoritmos se comportan igual en cuanto a esta medida. Según el test de Bonferroni–Dunn a este mismo nivel de significación, una diferencia entre rangos superior a 4,4161 implica la existencia de diferencias significativas entre ese par de algoritmos concretos. Encontramos una diferencia superior a dicho valor entre los rangos de los algoritmos GP_D, GP, JRIP, Ant-Miner+ y MOGBAP, con lo que la comprensibilidad de estos algoritmos en lo referente al número de reglas en el clasificador es significativamente mejor que la obtenida por nuestra propuesta. No obstante, MOGBAP se comporta significativamente mejor que GBAP, Ant-Miner, PSO/ACO2_D y PART_D.

La interpretación de estos resultados sigue una línea similar a la del estudio experimental de la Sección 4.7.2, donde se argumentaba que el algoritmo de GP era capaz de obtener muy buenos valores de comprensibilidad a costa de obtener el peor rendimiento en cuanto a exactitud predictiva. En este caso ocurre lo mismo: el algoritmo GP-Bojarczuk extrae aproximadamente una regla por cada clase existente

	MOGBAP		GBAP		Ant-Miner		Ant-Miner+		cAnt-Miner2$_D$		PSO/ ACO2$_D$		GP$_D$		JRIP$_D$		PART$_D$	
Data set	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}
Hepatitis	9,1	1,99	8,1	1,89	4,8	1,99	3,9	3,25	6,0	2,04	7,4	2,28	3,1	1,22	3,8	2,15	8,4	2,30
Sonar	11,1	2,04	12,3	1,81	5,2	2,07	4,0	3,48	6,6	1,99	6,1	2,92	3,0	1,00	4,6	2,21	13,9	2,98
Breast-c	11,8	1,69	13,2	1,91	6,0	1,28	5,4	2,82	6,9	1,25	11,8	1,75	3,5	1,01	3,3	1,70	17,1	2,12
Heart-c	9,9	2,25	14,5	1,67	5,9	1,20	4,4	2,82	7,2	1,41	11,9	3,81	3,0	3,02	5,3	2,32	17,3	2,35
Ionos.	6,8	1,49	11,1	1,18	5,7	1,61	8,8	1,41	7,7	1,59	4,5	4,03	3,1	1,14	7,7	1,48	8,2	1,83
Horse-c	9,6	2,03	9,0	1,46	6,3	1,49	4,7	3,41	7,5	1,57	20,1	3,39	3,0	1,00	3,5	1,74	13,2	2,38
Vote	6,6	2,12	17,2	2,19	5,6	1,36	5,2	2,34	6,8	1,59	6,1	1,33	3,0	1,00	3,1	1,38	7,7	1,84
Australian	9,1	2,00	10,1	1,08	6,5	1,53	3,3	2,08	7,6	1,52	25,8	6,96	3,0	1,00	5,2	1,80	19,4	2,01
Breast-w	6,1	1,77	6,6	1,65	7,2	1,04	6,4	1,92	8,1	1,12	10,5	1,10	3,0	1,00	6,5	1,74	10,9	1,63
Credit-g	11,6	1,82	22,9	1,82	9,1	1,51	3,3	3,31	9,7	1,37	52,8	4,20	3,3	1,17	7,1	2,54	57,8	2,70
Iris	5,8	1,15	3,7	1,06	4,3	1,03	3,9	1,80	5,2	1,02	3,0	1,20	4,3	1,29	3,0	1,00	4,6	1,00
Wine	6,1	1,47	7,2	1,50	5,1	1,33	2,5	2,19	6,5	1,41	4,0	1,73	4,1	1,27	4,2	1,56	6,3	1,77
Lymph.	11,9	1,55	10,2	1,60	4,7	1,69	4,6	2,83	6,9	1,74	15,6	2,11	5,1	1,02	6,9	1,53	10,2	2,30
Glass	17,5	2,22	21,6	1,79	8,4	1,76	12,4	4,10	10,4	2,25	24,5	3,13	8,2	1,48	8,0	2,03	13,7	2,32
Primary	34,9	2,53	45,9	2,60	12,1	3,35	9,3	8,50	21,8	4,43	86,5	6,01	23,7	1,37	8,3	3,13	48,7	3,23
#R Ranking	9,9		10,9		6,5333		4,7667		8,6667		9,7667		**3,0333**		4,5667		12,1333	
#C/R Ranking	7,2		5,5333		5,1		12,7333		5,8333		10,9333		**2,8**		6,7667		9,8333	

Tabla 5.6: Resultados comparativos del tamaño del clasificador y la complejidad de las reglas en test sobre conjuntos de datos discretizados

	cAnt-Miner2		PSO/ ACO2		GP		JRIP		PART	
Data set	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}	Acc	σ_{Acc}
Hepatitis	6,5	1,71	3,6	1,79	3,0	1,02	4,3	2,23	10,7	2,18
Sonar	6,6	1,99	4,5	3,09	3,0	1,00	4,9	1,92	12,6	2,51
Breast-c	3,3	1,70	11,8	1,75	3,5	1,01	3,3	1,70	17,1	2,12
Heart-c	7,4	1,71	12,8	3,18	4,0	3,39	4,0	2,05	20,0	1,76
Ionos.	6,8	1,81	4,8	2,95	3,6	1,16	5,9	1,20	7,3	2,41
Horse-c	7,0	1,78	15,3	3,60	3,1	1,72	4,2	1,81	18,7	2,53
Vote	6,8	1,59	6,1	1,33	3,0	1,00	3,1	1,38	7,7	1,84
Australian	8,5	1,56	3,8	1,51	3,0	1,00	4,9	2,01	17,2	2,05
Breast-w	6,7	1,87	7,1	1,93	3,0	1,00	5,8	1,86	9,5	2,46
Credit-g	9,9	1,57	54,1	4,25	3,1	1,07	6,3	2,33	70,1	3,07
Iris	4,3	1,14	3,0	1,03	4,2	1,21	3,8	1,08	3,3	1,30
Wine	4,9	1,46	4,0	1,88	4,4	1,33	4,2	1,52	4,5	1,57
Lymph.	6,8	1,68	13,3	2,11	5,2	1,03	6,3	1,60	12,1	2,54
Glass	10,4	1,94	20,5	2,92	8,3	1,12	7,4	2,07	15,2	2,81
Primary	21,8	4,43	86,5	6,01	23,7	1,37	8,3	3,13	48,7	3,23
#R Ranking	7,7		7,9667		3,3		4,0667		11,5	
#C/R Ranking	6,9		10,1667		3,1667		7,2		10,8333	

Tabla 5.7: Resultados comparativos del tamaño del clasificador y la complejidad de las reglas en test sobre conjuntos de datos sin discretizar

en el conjunto de datos, pero los resultados de *accuracy* decaen estrepitosamente. En cambio, nuestra propuesta es capaz de logar un buen compromiso, obteniendo los mejores resultados de *accuracy* y manteniendo, a su vez, niveles competitivos de comprensibilidad.

El segundo estudio estadístico considera el número medio de condiciones por regla. Tras aplicar el test de Iman&Davenport se obtiene un valor para el estadístico de 14,8741, que tampoco se halla dentro del intervalo crítico, con lo que se vuelve a rechazar la hipótesis nula. El test de Bonferroni–Dunn llevado a cabo posteriormente arroja dos conclusiones. Por un lado, que MOGBAP es significativamente mejor que el algoritmo Ant-Miner+ en lo concerniente a esta métrica. Y, por otro, que pese a que MOGBAP no se comporta estadísticamente mejor que el resto de algoritmos, tampoco se comporta peor.

Algorithm	Accuracy	#R	#C/R
MOGBAP	**82,00**	11,19	1,87
GBAP	81,50	14,24	1,68
Ant-Miner	79,29	6,46	1,62
Ant-Miner+	79,71	5,47	3,08
cAnt-Miner2-MDL$_D$	79,13	8,32	1,75
cAnt-Miner2-MDL	78,81	7,85	1,86
PSO/ACO2$_D$	80,14	19,37	3,06
PSO/ACO2	79,36	16,75	2,62
GP$_D$	72,60	**5,09**	**1,27**
GP	72,73	5,21	1,30
JRIP$_D$	81,17	5,37	1,89
JRIP	80,18	5,11	1,86
PART$_D$	80,68	17,16	2,18
PART	79,13	18,31	2,29

Tabla 5.8: Resultados medios de los algoritmos

También hay que mencionar que la longitud de las reglas minadas por el algoritmo depende directamente del parámetro *maxDerivations*, al igual que ocurría en el algoritmo GBAP. Así, mientras más número de derivaciones se permitan, habrá mayor probabilidad de que se encuentren soluciones más complejas, con mayor número de condiciones, que tienen en cuenta relaciones más intrincadas entre los atributos predictivos.

Por lo tanto, tras efectuar estos dos test relacionados con la comprensibilidad, es posible afirmar que el algoritmo presentado obtiene resultados competitivos, tanto en lo referente al número de reglas en el clasificador final como al número de condiciones por regla. Es más, presenta un buen compromiso entre exactitud predictiva y comprensibilidad, al ser el algoritmo que mejor se comporta en cuanto a *accuracy* a la vez que alcanza un nivel de comprensibilidad competitivo. La complejidad de las reglas extraídas se puede observar en la Tabla 5.9, obtenidas mediante la ejecución de MOGBAP sobre una partición de entrenamiento del conjunto de datos *iris*.

```
IF      (= petalwidth (-inf-0,8]) THEN setosa
ELSE IF (AND (= petalwidth (1,75-inf))
             (= sepallength (6,15-inf)))
             THEN virginica
ELSE IF (= petallength (2,45-4,75])
             THEN versicolor
ELSE IF (= petalwidth (1,75-inf))
             THEN virginica
ELSE IF (= petalwidth (0,8-1.75])
             THEN versicolor
ELSE    virginica
```

Tabla 5.9: Ejemplo de clasificador sobre el conjunto de datos *iris*

5.9. Conclusiones

En este capítulo hemos extendido el algoritmo de AP monobjetivo original, GBAP, adaptándolo a un enfoque multiobjetivo que busca alcanzar un compromiso entre exactitud predictiva y comprensibilidad. El nuevo algoritmo, denominado MOGBAP (*Multi-Objective Grammar-Based Ant Programming*), optimiza simultáneamente tres objetivos, sensibilidad, especificidad y una medida de la comprensibilidad. Una de las aportaciones principales del algoritmo es la estrategia de evaluación multiobjetivo con k frentes de Pareto apta para cualquier problema de clasificación, que se encarga de extraer reglas óptimas por cada clase a predecir en el conjunto de datos, evitando el solapamiento que se puede producir al ordenar por dominancia de Pareto individuos de diferentes clases al mismo tiempo. Para ello, MOGBAP encuentra en cada generación, y por cada posible clase del conjunto de entrenamiento, un grupo de individuos no dominados que almacena en un frente de Pareto específico para dicha clase hasta la última generación del algoritmo, momento en que se terminan por combinar en el clasificador final los individuos los k frentes siguiendo un enfoque de nichos.

6

Modelo de AP para clasificación binaria y multiclase de datos no balanceados

En muchos de los dominios de aplicación de la tarea de clasificación de DM, tales como medicina, finanzas, detección de anomalías o telecomunicaciones, los problemas suelen presentar una distribución de datos compleja, donde las clases aparecen desbalanceadas. Dado que la mayoría de los clasificadores tradicionales abordan la tarea de clasificación sin considerar la distribución relativa de instancias por clase, la clasificación de conjuntos de datos no balanceados está atrayendo la atención de los investigadores de este campo [102], considerándose un tema crucial del reconocimiento de patrones. Al tratar con este tipo de datos, centrarse únicamente en la optimización de la exactitud predictiva puede llevar a resultados engañosos debido a la naturaleza del problema, donde los casos minoritarios, en cuya identificación se centra habitualmente el interés de los expertos, tienden a ser ignorados por los clasificadores clásicos. De hecho, el impacto o importancia de la clase minoritaria con respecto a la exactitud no es relevante si se compara al de las clases mayoritarias. Supongamos, por ejemplo, un conjunto de datos de diagnóstico de cáncer que

presente 100 instancias cancerosas y 10.000 no cancerosas. En este caso hipotético, el ratio de desbalanceo (IR, *Imbalance Ratio*) sería de 1 a 100, por lo que un clasificador que prediga cualquier ejemplo como no canceroso obtendría resultados de exactitud cercanos al 100 %.

Los algoritmos de clasificación que no han sido específicamente diseñados para problemas no balanceados, al abordar conjuntos de datos de este tipo, tienden a inferir un modelo que clasifica incorrectamente ejemplos de test de la **clase positiva (minoritaria)** con mayor frecuencia que aquellos pertenecientes al resto de clases, lo que supone un alto coste para sus dominios de aplicación. Esto se debe al hecho de que los algoritmos clásicos buscan reglas que clasifiquen correctamente un mayor número de ejemplos, con lo que tienen a favorecer a la **clase negativa (mayoritaria)**. Este desbalanceo entre clases no es el único responsable del deterioro en el rendimiento de un algoritmo, sino que influyen otros factores como el solapamiento de clases, que se presenta frecuentemente en estos dominios. Para revertir esta situación, se han propuesto varias soluciones, aunque aún existen algunos temas pendientes [76]. Entre ellos, este capítulo se centra en el problema de la **separación entre soluciones binarias y multiclase**, dado que la mayoría de los enfoques propuestos para problemas binarios no se pueden aplicar en problemas con más de dos clases. Además, a pesar de que existen multitud de dominios en los que los datos presentan múltiples clases no balanceadas, la mayoría de los esfuerzos en este campo se centran sólo en la clasificación no balanceada de conjuntos de datos binarios. De hecho, se han propuesto pocos trabajos para tratar el problema multiclase [213].

Dados los buenos resultados obtenidos por los algoritmos de AP para clasificación desarrollados, consideramos que explorar su aplicación a dominios no balanceados constituye un paso natural a los capítulos anteriores. Para ello se presenta el algoritmo **APIC** (*Ant Programming for Imbalanced Classification*), que sigue un enfoque multiobjetivo para la clasificación no balanceada binaria y multiclase. Las principales ventajas del modelo propuesto se pueden resumir como sigue. En primer lugar, se encuadra dentro de las soluciones para datos no balanceados **a nivel del algoritmo**, y como se ha comentado se puede aplicar tanto a conjuntos de datos binarios como multiclase. En segundo lugar, centrándonos en el caso binario, no requiere llevar a cabo un remuestreo, sino que **emplea los conjuntos de**

datos originales en el proceso de aprendizaje. Teniendo en cuenta ahora la perspectiva multiclase, no requiere reducir el problema utilizando un **enfoque uno a uno** (OVO, *One-Vs-One*) o **uno a muchos** (OVA, *One-Vs-All*), donde se induce un clasificador por cada combinación. En su lugar, trata con el problema directamente, simplicando pues la complejidad del modelo. Además, proporciona mejores resultados en situaciones fronterizas, ya que el clasificador se entrena conociendo la existencia de todas las clases [175]. Por última, también presenta otras ventajas inherentes al uso de una gramática y de un enfoque multiobjetivo, y genera un clasificador comprensible basado en reglas IF-THEN.

Para evaluar el rendimiento de nuestro modelo hemos efectuado un estudio experimental sobre un gran número de conjuntos de datos variados, llevando a cabo una validación cruzada estratificada con cinco particiones utilizando, además, diez grupos diferentes de particiones generadas con diferentes semillas, en vez de uno solo, con el objetivo de minimizar los efectos del problema del ***shift*** [153] de los conjuntos de datos que aparece frecuentemente en la clasificación de datos no balanceados.

A continuación, la Sección 6.1 revisa los métodos existentes para tratar con los problemas de clasificación de datos no balanceados binarios y multiclase. La Sección 6.2 describe el modelo propuesto para tratar este tipo de problemas de clasificación. La Sección 6.3 presenta los estudios experimentales realizados, analizando los resultados obtenidos. Finalmente, la Sección 6.5 expone las conclusiones alcanzadas.

6.1. Clasificación de datos no balanceados

En esta sección presentamos las principales propuestas que existen en la literatura actualmente para abordar la tarea de clasificación de datos no balanceados. Para ello, hacemos una división entre enfoques para tratar conjuntos de datos binarios y multiclase.

6.1.1. Conjuntos de datos binarios

La taxonomía que proponen *Sun et al.* [213] para clasificar las propuestas que tratan el problema de clasificación binaria de conjuntos de datos no balanceados es la siguiente:

- **Enfoques externos, a nivel de datos, o de remuestreo** [90]. Las instancias del conjunto de entrenamiento se rebalancean para obtener una distribución de clases equitativa, bien mediante **sobremuestreo**, generando más instancias para la clase minoritaria, mediante **bajomuestreo**, eliminando instancias correspondientes a la mayoritaria, o bien mediante una combinación de ambas. Se puede considerar a estos métodos como técnicas de preprocesamiento, y es ahí donde radica su principal ventaja: una vez que el conjunto de datos de entrenamiento ha sido preprocesado, es posible aplicar algoritmos clásicos sobre ellos.

- **Enfoques internos, a nivel del algoritmo** [102]. Suponen la modificación de los algoritmos de manera que tengan en cuenta el efecto de la distribución sesgada de los datos en la fase de aprendizaje, mejorando el rendimiento del algoritmo en test. APIC se encuadra en este grupo.

- **Enfoques de aprendizaje sensibles al coste** [4]. Se introducen costes en las fórmulas empleadas para calcular el error del modelo. Una matriz de costes codifica la penalización de clasificar ejemplos de una clase como si fueran de otra. El algoritmo tiene así en cuenta que el coste de cometer un error clasificando una instancia de la clase positiva como negativa es mayor que en el caso opuesto, con lo que trata de minimizar dicho coste en el modelo entrenado.

- **Enfoques de combinación de modelos (*ensembles*) y de *boosting*** [87]. Se entrenan múltiples clasificadores a partir del conjunto de entrenamiento original, agregando sus predicciones para asignar una etiqueta al clasificar una nueva instancia.

6.1.2. Conjuntos de datos multiclase

Cuando un conjunto de datos no balanceado presenta más de dos clases puede que las soluciones descritas en la sección anterior no sean aplicables. De hecho, existen pocas técnicas de clasificación capaces de tratar con el problema directamente y la mayoría requieren, además, ciertas modificaciones o extensiones para adecuarse a clasificación multiclase [102]. Además, la existencia de múltiples clases plantea una dificultad adicional a los algoritmos de DM, ya que la separación entre clases puede no estar bien definida, solapándose unas con otras y provocando un descenso en el nivel de rendimiento. En esta situación, sí que se puede seguir un **esquema de descomposición** o **dicotomización** de manera que el conjunto de datos multiclase se reduzca a varios binarios y, por tanto, sea posible la aplicación de este tipo de clasificadores [208]. Estos métodos consisten en dos pasos:

1. Se divide el problema multiclase original en subproblemas binarios más sencillos.

2. Por cada subproblema obtenido se aplican aquellas soluciones que han sido desarrolladas para tratar con conjuntos de datos binarios no balanceados.

Concretamente, existen dos métodos para reducir un problema multiclase a uno binario:

1. Enfoque **OVO**. Dado un conjunto de entrenamiento multiclase con k categorías, se entrenan $k(k-1)/2$ clasificadores binarios, cada uno dedicado a distinguir entre cada par de clases (i, j), donde $i \neq j$.

2. Enfoque **OVA**. Dado un conjunto de entrenamiento multiclase con k categorías, se entrenan k clasificadores binarios, cada uno dedicado a distinguir una clase específica del resto de clases.

Para determinar la salida predicha al clasificar una nueva instancia, se pueden emplear diversos métodos de agregación [87]. Entre ellas, mencionamos aquí el enfoque seguido en el estudio experimental, denominado **estrategia de votación**. En el caso de un sistema OVO, cada clasificador binario vota la clase que predice para la nueva instancia (cada uno podrá votar entre dos clases diferentes). Así,

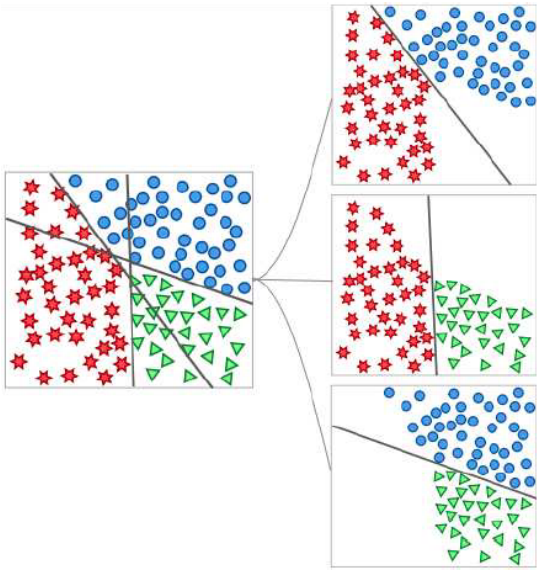


Figura 6.1: Ejemplo de binarización de un conjunto de datos de 3 clases mediante OVO (adaptado de *Fernández et al.* [77])

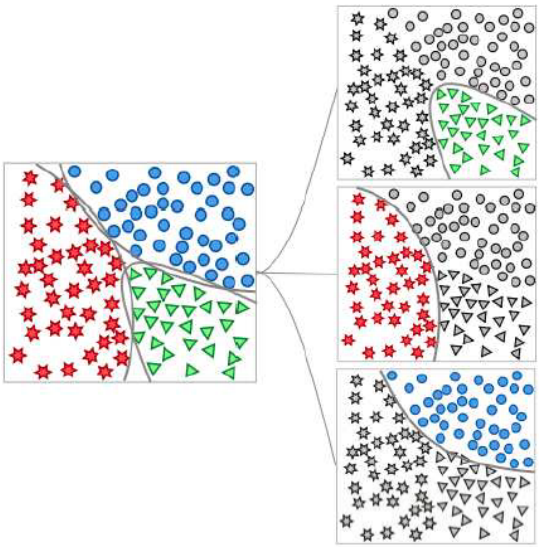


Figura 6.2: Ejemplo de binarización de un conjunto de datos de 3 clases mediante OVA (adaptado de *Fernández et al.* [77])

se cuentan los votos que recibe cada clase y se asigna aquella que ha recibido un mayor número de votos. En el caso de los sistemas OVA, el clasificador que predice como salida la clase positiva será el que determine la clase a asignar. En caso de que la salida positiva no sea única y haya empate, se tomará como salida la del clasificador con mayor confianza. Es por ello que en estos sistemas, esta estrategia de agregación también recibe el nombre de estrategia de confianza máxima.

6.1.3. Medidas de rendimiento

A efectos de medir el rendimiento de un clasificador en dominios no balanceados, tal como se ha comentado con anterioridad, no se debe emplear la exactitud predictiva al encontrarse muy influenciada por la clase mayoritaria y, por ende, no arrojar resultados imparciales y objetivos en estas situaciones. Esta predisposición hacia la clase mayoritaria cobra más presencia cuanto más distorsión existe entre las clases. En su lugar, el **área bajo la curva ROC**, denotada como **AUC** [71], es una medida de evaluación de rendimiento extendida para clasificación no balanceada. Ambas fueron explicadas en detalle en la Sección 2.1.2. A modo de recordatorio, la curva ROC presenta el compromiso entre la tasa de verdaderos positivos (TPR, *True Positive Rate*) y la tasa de falsos positivos (FPR, *False Positive Rate*). Un algoritmo evolutivo que emplee esta medida como función de *fitness* tenderá a capturar más verdaderos positivos, lo que generalmente suele conllevar que aumente su tasa de falsos positivos, al identificar incorrectamente instancias que son negativas como si fueran positivas. El AUC se calcula a partir de los valores de la matriz de confusión, como se indica en la Ecuación 6.1:

$$AUC = \frac{1 + \frac{T_P}{T_P + F_N} - \frac{F_P}{F_P + T_N}}{2} \tag{6.1}$$

Esta medida busca pues el equilibro entre el ratio de verdaderos positivos y el de falsos positivos. Sin embargo, esta definición es únicamente válida para problemas de clasificación binarios. En conjuntos de datos con más de dos clases se hace necesario extender esta fórmula para considerar las relaciones por parejas de clases. Esta extensión se denomina como **AUC probabilística**, y en ella se calcula un valor único por cada par de clases, tomando una clase como positiva y la otra como negativa. Así, el valor medio se obtiene como indica la Ecuación 6.2:

$$PAUC = \frac{1}{k(k-1)} \sum_{i=1}^{k} \sum_{j \neq i}^{k} AUC(i,j) \tag{6.2}$$

donde:

$AUC(i,j)$	es el AUC que se calcula tomando la clase i como positiva y la clase j como negativa, y
k	indica el número de clases.

6.2. Algoritmo APIC

El algoritmo propuesto en este capítulo, APIC, induce un clasificador a partir de un proceso de aprendizaje sobre un conjunto de entrenamiento. Este clasificador adopta la forma de una lista de decisión compuesta de reglas *IF-THEN*. El pseudocódigo se muestra en el Algoritmo 6, y sus principales características se describen en la siguientes secciones, deteniéndonos especialmente en los puntos que lo diferencian de los otros dos modelos de AP desarrollados hasta el momento y que se han explicado en los dos capítulos previos. Dichas modificaciones son naturales al propósito del algoritmo APIC, que no es otro que la clasificación de conjuntos de datos no balanceados. Entre otras, cabe destacar que este nuevo algoritmo es **multicolonia**, mientras que en los otros existía una población de hormigas común. Por otra parte, sigue un enfoque **multiobjetivo** donde se busca maximizar el valor de dos funciones objetivo utilizadas con frecuencia en dominios no balanceados. La función heurística que se venía utilizando hasta ahora, la ganancia de información, no se ajusta a este tipo de dominios, por lo que ha sido sustituida por la **confianza de la clase**, más apta para este tipo de problemas. Aparte de estas diferencias, presenta otras que están relacionadas con el refuerzo de las feromonas y la construcción del clasificador final.

6.2.1. Entorno y codificación de los individuos

Los individuos creados durante las distintas generaciones del algoritmo se comunican entre sí de un modo indirecto, mediante las feromonas que van dejando en el entorno. El entorno está formado por todas las posibles expresiones que se pueden derivar desde el estado correspondiente al símbolo inicial de la gramática, mediante

la aplicación de las reglas de producción en el número de pasos de derivación disponibles, definidos por el parámetro que indica el número máximo de derivaciones. La gramática que utiliza el algoritmo es la siguiente:

G = (Σ_N, Σ_T, P, <EXP>)

Σ_N = {<EXP>, <COND>}

Σ_T = {AND, =

attr$_1$, attr$_2$, ..., attr$_n$,

value$_{11}$, value$_{12}$, ..., value$_{1m}$,

value$_{21}$, value$_{22}$, ..., value$_{2m}$,

..., value$_{n1}$, value$_{n2}$, ..., value$_{nm}$}

P = {<EXP>:= <COND>| AND <EXP><COND>,

<COND>:= todas las posibles combinaciones de la terna

= attr value}

Así pues, el entorno adquiere la forma de un árbol de derivación, y cada vez que una hormiga es creada y alcanza una solución, correspondiente a una hoja del árbol de derivación, los estados del camino seguido hasta llegar a ella son almacenados en la estructura de datos que representa el espacio de estados.

Dado que el algoritmo APIC es multicolonia, el camino codificado por cada individuo representará el antecedente de la regla, y el consecuente será conocido dado que existen tantas colonias de individuos como clases tenga el conjunto de datos. Así, cada colonia se dedicará a generar individuos para predecir la clase correspondiente. Cada colonia evoluciona en paralelo, ya que los individuos generados en una de ellas no interfieren en ningún momento con aquellos creados en el resto. Esto se consigue, además, simulando la existencia de **distintos tipos de feromonas**, tantos como clases haya que considerar.

Los movimientos de los individuos durante su creación, hasta que alcanzan una solución válida, se basan en un método probabilístico que considera información procedente del entorno (las feromonas) y del dominio del problema (la heurística) que se incluye en la regla de transición. Esta regla define la probabilidad con que una hormiga podrá seleccionar cada una de las transiciones de que disponga en

un momento dado. Suponiendo que una hormiga se encuentre en el estado i del entorno, la probabilidad de moverse al estado j se define como:

$$P^k_{class_{ij}} = \frac{(\eta_{ij})^\alpha \cdot (\tau_{class_{ij}})^\beta}{\sum_{k \in allowed}(\eta_{ik})^\alpha \cdot (\tau_{class_{ik}})^\beta} \tag{6.3}$$

donde:

- k indica el número de estados válidos disponibles,
- α es el exponente para la heurística,
- β es el exponente para las feromonas,
- η representa el valor de la función heurística, y
- τ_{class} el nivel de feromonas en la transición de la colonia a la que pertenece el individuo.

6.2.2. Información heurística

La información heurística en APIC considera dos componentes complementarios que no se pueden aplicar simultáneamente, sino cuya aplicación depende del tipo de transición en cuestión. Al igual que los algoritmos de AP desarrollados en los capítulos anteriores, en el caso de las transiciones intermedias APIC utiliza la probabilidad de cardinalidad, que incrementa la probabilidad de seleccionar transiciones cuya selección amplía el rango de posibles soluciones para la hormiga. En cambio, existe una diferencia con respecto a las transiciones finales, y es que APIC ya no utiliza la ganancia de información, sino una medida denominada **confianza de la clase** (*class confidence*) [142]. La ganancia de información ha sido descartada al no ser adecuada para entornos no balanceados, ya que guía la búsqueda hacia la clase mayoritaria. En cambio, la confianza de clase, definida en la Ecuación 6.4, permite centrarse en los antecedentes más interesantes para cada clase, ya que emplea el soporte del consecuente en el denominador.

$$CC(x \rightarrow y) = \frac{Support(x \cup y)}{Support(y)} \tag{6.4}$$

donde:

- x representa el antecedente de la regla, e
- y su consecuente.

6.2.3. Mantenimiento de feromonas

En los algoritmos multiobjetivo de ACO se distinguen dos tipos según cómo se trata la información relativa a las feromonas, los que emplean una única matriz de feromonas o aquellos que usan una por cada objetivo a optimizar [12]. El algoritmo APIC pertenece a la primera categoría. No obstante, dado que se trata de un algoritmo multicolonia, existirán k matrices de feromonas diferentes, una independiente por cada colonia. Así pues, cada colonia se centra en evolucionar individuos o reglas para su clase específica.

Las dos operaciones consideradas en cuanto al mantenimiento de feromonas son el refuerzo y la evaporación. Esta última tiene lugar sobre todo el entorno:

$$\tau_{class_{ij}}(t+1) = \tau_{class_{ij}}(t) \cdot (1-\rho) \tag{6.5}$$

donde ρ representa la tasa de evaporación.

Por su parte, el refuerzo solamente puede producirse sobre el camino que han seguido las hormigas del frente de Pareto de cada colonia (ver Sección 6.2.4). Para un determinado individuo, todas las transiciones de su camino percibirán igual cantidad de feromonas, cantidad que vendrá determinada por la longitud del camino seguido (las soluciones más cortas recibirán mayor concentración de feromonas) y por la calidad de la regla codificada, representada por el valor de AUC calculado para dicho individuo sobre el conjunto de entrenamiento:

$$\tau_{class_{ij}}(t+1) = \tau_{class_{ij}}(t) + \tau_{class_{ij}}(t) \cdot \frac{maxDerivations}{pathLength} \cdot AUC \tag{6.6}$$

Cuando las operaciones de actualización de feronomas terminan, tiene lugar un proceso de normalización para limitar los niveles de feromonas existentes en cada transición al intervalo $[\tau_{min}, \tau_{max}]$. Además, en la primera generación, el espacio de

estados de cada colonia se inicializa con la máxima cantidad de feromonas permitida.

6.2.4. *Cálculo de* fitness *multiobjetivo*

La calidad de los individuos generados en APIC se calcula en base a dos objetivos, precisión y *recall* o sensibilidad (explicadas en la Sección 2.1.2). Ambas medidas han sido ampliamente utilizadas en dominios no balanceados, ya que cuando se usan conjuntamente permanecen sensibles al rendimiento en cada clase [134]. Así pues, son apropiadas para usarse como funciones objetivo, tratando de maximizar su valor.

Como explicamos en el capítulo anterior, la dominancia de Pareto establece que un determinado individuo ant_1 domina a otro ant_2, representado como $ant_1 \succ ant_2$, si ant_1 no es peor que ant_2 en todos los objetivos y es mejor en al menos uno de ellos. El conjunto de soluciones no dominadas de una población conforma el frente de Pareto.

Las medidas de precisión y *recall* se usan para evolucionar un frente de Pareto de individuos por cada clase existente en el conjunto de entrenamiento, ya que cada colonia es responsable de generar individuos prediciendo una única clase, por lo que en total habrá tantos frentes de Pareto como colonias, k. En una determinada colonia de hormigas hay que tener en cuenta que sólo los individuos no dominados pueden reforzar su camino y ser una de las reglas que compongan el clasificador final. Como se observa en el pseudocódigo, también se calcula el AUC para cada individuo, ya que la intensidad con que se refuerza el camino de una solución depende del valor del mismo, y dado que también se emplea como mecanismo de ordenación de las reglas en la lista de decisión final.

Una vez que el proceso evolutivo termina en todas las colonias, para seleccionar apropiadamente las reglas que formarán parte del clasificador, se lleva a cabo un procedimiento de nichos sobre el frente de Pareto final de cada colonia. De esta forma se seleccionan las mínimas reglas necesarias para cubrir los ejemplos de entrenamiento de cada clase, mezclándolas luego en el clasificador. Dado que éste actúa como una lista de decisión, las reglas se ordenan en orden descendente según su AUC.

Algoritmo 6 Pseudocódigo de APIC

Require: $trainingSet, testSet, numGenerations$, $numAnts, maxDerivations$

Inicializar gramática a partir de $trainingSet$
Crear lista $categories \leftarrow$ Clases en $trainingSet$
$numClasses \leftarrow$ Longitud de $categories$
Crear tabla hash $paretoFronts \leftarrow \{\}$
for $k \leftarrow 0$ **to** $k < numClasses$ **do**
    Inicializar espacio de estados para colonia k
    Crear lista $paretoFront \leftarrow \{\}$
    Introducir entrada (k, $paretoFront$) en tabla hash $paretoFronts$
end for
$derivationStep \leftarrow \frac{maxDerivations-2}{numGenerations}$
for $k \leftarrow 0$ **to** $k < numClasses$ in $trainingSet$ **do**
    $maxDerivations \leftarrow 2$
    for $i \leftarrow 0$ **to** $i < numGenerations$ **inc** 1 **do**
        Crear lista $antList \leftarrow \{\}$
        for $j \leftarrow 0$ **to** $j \leftarrow numAnts$ **inc** 1 **do**
            $ant \leftarrow$ Crear nueva hormiga, asignando como consecuente la clase k
            Almacenar los estados del camino seguido por ant en el espacio de estados de la colonia k
            Evaluar ant para cada función objetivo
            Añadir ant a la lista $antList$
        end for
        Añadir a la lista $antList$ los individuos obtenidos de la tabla $paretoFronts$, dada la clave $numClass$, comprobando duplicidad
        Crear lista $newParetoFront \leftarrow$ Soluciones no dominadas en $antList$
        for cada ant en $newParetoFront$ **do**
            Actualizar la tasa de feromonas en el camino seguido por ant de manera proporcional al AUC e inversamente proporcional a la longitud de la solución que codifica
        end for
        Introducir entrada (k, $paretoFront$) en tabla hash $paretoFronts$
        Evaporar las feromonas y normalizar valores en espacio de estados k
        $maxDerivations \leftarrow maxDerivations + derivationStep$
    end for
end for
for $k \leftarrow 0$ **to** $k < numClasses$ **do**
    Crear lista $paretoFront \leftarrow$ Recuperar lista de tabla hash $paretoFronts$ a partir de clave k
    Añadir al clasificador hormigas devueltas tras llevar a cabo el procedimiento de nichos sobre las hormigas de la lista $paretoFront$ (ver Procedimiento 5).
end for
Ordenar reglas en el clasificador por AUC y establecer regla por defecto
$AUCtest \leftarrow$ Calcular el AUC obtenido al ejecutar sobre $testSet$ el clasificador
return $AUCtest$

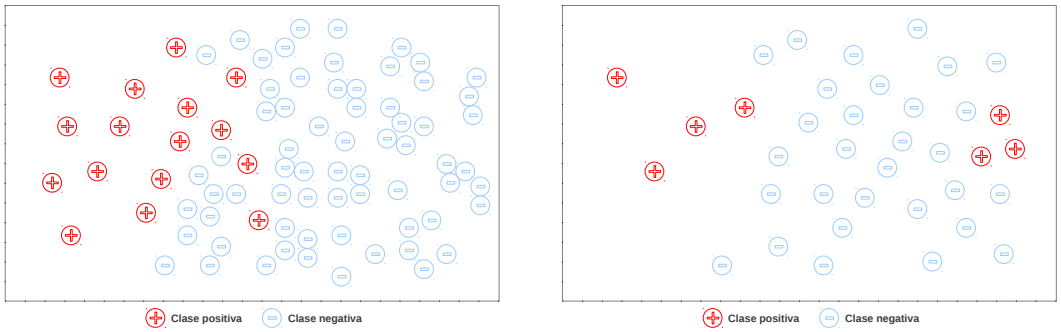
6.3. Experimentación

En esta sección se explica en primer lugar el procedimiento seguido para ejecutar los algoritmos y evaluar su rendimiento. Posteriormente se indican los algoritmos empleados y su configuración.

6.3.1. Efecto $shift$ en conjuntos de datos y validación cruzada

Resulta conveniente comentar otra dificultad que puede afectar a cualquier problema de clasificación, pero que cobra especial relevancia cuando se trata de clasificar datos no balanceados: el **problema del *shift* en el conjunto de datos**. Este problema se produce cuando la distribución que sigue el conjunto de entrenamiento y que, por tanto, es la que emplea el algoritmo para construir un modelo, es diferente de la distribución que siguen los datos en el conjunto de test [153]. En la Figura 6.3 se puede observar este problema gráficamente: debido a la baja presencia de instancias pertenecientes a la clase minoritaria en un conjunto de entrenamiento, el modelo aprendido puede clasificar incorrectamente las instancias positivas en el conjunto de test correspondiente, que además suelen ser las de mayor interés. De hecho, un simple error puede provocar un decaimiento significativo del rendimiento en ciertos casos [76]. En la figura se observa cómo en el conjunto de test aparecen instancias positivas en una zona del espacio de soluciones donde no aparecían instancias de esta clase en el conjunto de entrenamiento, lo que hace imposible entrenar un modelo que clasifique correctamente dichas instancias en test.

Con el objetivo de minimizar los efectos del problema del *shift*, no nos limitamos a efectuar un procedimiento de validación cruzada típico. En su lugar hemos diseñado el estudio experimental de manera que se empleen diez grupos de particiones diferentes. Cada grupo consta de cinco particiones mutuamente exclusivas, y preservan todo lo posible la proporción original de instancias por clase, esto es, están estratificadas con el fin de introducir el mínimo efecto *shift* posible y evitar la influencia que puede introducir una determinada semilla al generar un grupo de particiones [153]. Así pues, por cada uno de los grupos se ha llevado a cabo un procedimiento de *5-fold cross validation*, donde cada algoritmo se ejecuta en cinco



(a) Conjunto de entrenamiento (b) Conjunto de test

Figura 6.3: Ejemplo de distribuciones desbalanceadas de entrenamiento y test mostrando el problema del *shift*

ocasiones, con una partición utilizada como conjunto de test cada vez y la unión de las otras cuatro como conjunto de entrenamiento. El AUC global obtenido por un clasificador sobre un determinado problema se calcula considerando el AUC medio de los cincuenta experimentos (cinco experimentos por diez grupos de particiones):

$$AUC_{test} = \sum_{i=1}^{10} \frac{\sum_{j=1}^{5} \frac{AUCP_{ij}}{5}}{10} \tag{6.7}$$

donde:

P_{ij}	se refiere a la partición j del grupo de particiones i, y
$AUCP_{ij}$	representa el AUC obtenido por el clasificador cuando la partición P_{ij} se utiliza como conjunto de test.

El motivo por el cual se ha empleado un procedimiento de *5-fold cross validation*, cuando hasta ahora hemos venido utilizando un *10-fold cross validation*, radica en que determinados conjuntos de datos tienen menos de diez instancias para la clase minoritaria. Esto implica que, si se usan diez particiones, haya casos en que la partición de test no contenga ningún ejemplo de la clase positiva.

Además, hay que tener en cuenta que al evaluar el rendimiento de los algoritmos no deterministas (todos los algoritmos empleados en la experimentación son de

este tipo, excepto NN CS [245] y C-SVM CS [218]), se han considerado diez semillas adicionales, lanzando el procedimiento de validación cruzada nueve veces más para evitar cualquier posibilidad de obtener resultados sesgados debido a la semilla de números aleatorios que utiliza el algoritmo internamente en sus cálculos. Así pues, en estos casos, el algoritmo ha sido ejecutado quinientas veces en total por cada conjunto de datos, mientras que para los algoritmos deterministas han sido realizadas cincuenta ejecuciones.

6.3.2. Algoritmos y configuración de parámetros

La experimentación ha sido separada en dos secciones, una binaria y una multiclase. La configuración de APIC es idéntica en ambas, y se muestra en la Tabla 6.1.

Parámetro	Valor
numAnts	20
numGenerations	100
maxDerivations	15
percentageOfCoverage	3
$[\tau_0]$	*1,0*
$[\tau_{min}]$	*0,1*
$[\tau_{max}]$	*1,0*
$[\rho]$	*0,05*
$[\alpha]$	*0,4*
$[\beta]$	*1,0*

Tabla 6.1: Parámetros y configuración del algoritmo APIC

Para la experimentación binaria hemos considerado otras ocho técnicas: una correspondiente a *boosting*, ADAC2 [212]; tres sensibles al coste, NN CS, C-SVM CS y C4.5 CS [220]; y cuatro de rebalanceo, RUS [229], SBC [237], SMOTE [44] y SMOTE-Tomek Links [222], todas ellas utilizando el algoritmo C4.5 como clasificador base. Estos algoritmos, su descripción y los parámetros empleados se pueden observar en la Tabla 6.2. Finalmente, para la experimentación multiclase hemos comparado APIC frente a los esquemas de descomposición OVO y OVA utilizando como clasificador base el algoritmo C4.5 CS con la misma configuración de parámetros empleada en el caso binario. Se ha elegido este método sensible al coste

para compararlo con nuestro algoritmo dado que ha demostrado ser la técnica que mejores resultados obtiene en conjunción con los esquemas de descomposición [77].

Algoritmo	Parámetro	Valor
ADAC2, un algoritmo de *boosting* que genera un *ensemble* de árboles de decisión C4.5 a partir de un conjunto de ejemplos. Cada instancia tiene un coste asociado dependiendo de la clase. Para resolver el problema del desbalanceo, incrementa el peso de las instancias de la clase minoritaria en cada iteración del algoritmo. Los costes se seleccionan de manera automática considerando el ratio de desbalanceo.	Pruned Confidence Instances per leaf Number of classifiers Train method Cost set-up	true 0,25 2 10 cost-sensitive adaptive
NN CS, una red neuronal sensible al coste	Hidden layers Hidden nodes Transfer function η α λ Cycles	2 15 Htan 0,15 0,1 0 10000
C-SVM CS, que construye un modelo SVM con los datos de entrenamiento teniendo en cuenta el coste asociado a la distribución de clases y entonces clasifica los datos de test empleando dicho modelo	Kernel C eps Degree γ Coef0 Shrinking	polynomial 100 0,001 1 0,01 0 true
C4.5 CS, un árbol de decisión C4.5 sensible al coste	Pruned Confidence Instances per leaf Minimum expected cost	true 0,25 2 true
RUS+C4.5, un algoritmo de bajomuestreo aleatorio de instancias sobrerrepresentadas, con la posterior aplicación del algoritmo C4.5	Pruned Confidence Instances per leaf	true 0,25 2
SBC+C4.5, bajomuestreo basado en *clustering*, y aplicación posterior del algoritmo C4.5	Ratio Ncluster Pruned Confidence	0,5 5 true 0,25
SMOTE+C4.5, que hace uso de SMOTE para generar instancias de la clase bajorrepresentada, aplicando a continuación el algoritmo C4.5	Number of neighbours Distance Pruned Confidence Instances per leaf	5 HVDM true 0,25 2
SMOTE-TL+C4.5, que hace uso de SMOTE para generar instancias de la clase minoritaria, elimina a continuación instancias cercanas a los límites entre clases mediante el algoritmo TL, y finalmente aplica el árbol de decisición C4.5	Number of neighbours Distance Pruned Confidence Instances per leaf	5 HVDM true 0,25 2

Tabla 6.2: Algoritmos para clasificación de conjuntos de datos no balanceados binarios. Configuración de parámetros

Teniendo en cuenta el hecho de que el algoritmo APIC sólo puede tratar con atributos nominales, para poder llevar a cabo una comparación justa entre los algoritmos, los conjuntos de entrenamiento fueron discretizados empleando el algoritmo de discretización de Fayyad&Irani. Así, empleando los puntos de corte encontrados sobre cada conjunto de entrenamiento fueron discretizadas las particiones de test respectivas.

Por otro lado, tanto los conjuntos de datos no balanceados binarios como los multiclase empleados en el estudio han sido descargados del repositorio KEEL [7]. Para el estudio experimental binario, los experimentos fueron ejecutados utilizando las implementaciones de los algoritmos que proporciona la herramienta software KEEL [8]. En cuanto a la experimentación multiclase, hemos utilizado nuestras propias implementaciones para descomponer los conjuntos de datos apropiadamente, ejecutar el clasificador base por cada uno de ellos, y la estrategia de votación para determinar la clase asignada a una instancia de test.

6.4. Resultados

En esta sección se analizan los resultados obtenidos en los dos estudios experimentales, uno para conjuntos de datos binarios y otro para multiclase.

6.4.1. Resultados del estudio experimental sobre conjuntos de datos binarios

Cada fila de la Tabla 6.3 muestra los resultados medios de AUC obtenidos por cada algoritmo sobre un conjunto de datos binario determinado, tras llevar a cabo los experimentos tal como describe la Sección 6.3.1. Los conjuntos de datos están ordenados en base a su IR en orden ascendente, como se observa en la segunda columna de la tabla. APIC obtiene los mejores resultados sobre ocho de los conjuntos de datos, obteniendo también el mejor resultado en otros tres, aunque en este caso empatado con otros enfoques. La última fila de la tabla muestra los rangos medios obtenidos por cada algoritmo.

Dataset	IR	APIC	AdaC2	NN CS	C-SVM CS	C4.5 CS	RUS C4.5	SBC C4.5	SMOTE C4.5	SMOTE TL+C4.5
Ecoli0vs1	1,86	0,9811	0,9742	0,9808	0,9811	0,9811	0,9811	0,5000	0,9792	0,9792
Iris0	2,00	1,0000	0,9850	1,0000	1,0000	1,0000	1,0000	0,5000	0,9855	0,9852
Haberman	2,78	0,6410	0,5720	0,5052	0,5000	0,4992	0,6311	0,5528	0,6350	0,6343
Glass0123vs456	3,20	0,9200	0,9193	0,9161	0,7923	0,9131	0,9066	0,6823	0,9181	0,9245
Ecoli1	3,36	0,8917	0,8834	0,7735	0,5003	0,8965	0,8941	0,5378	0,8861	0,8857
Ecoli2	5,46	0,8493	0,8952	0,6892	0,7248	0,8469	0,8423	0,5000	0,8834	0,8772
Glass6	6,38	0,9012	0,8898	0,7800	0,8365	0,9115	0,8994	0,5789	0,9090	0,9093
Yeast2vs4	9,08	0,8598	0,8950	0,7404	0,6573	0,8963	0,9242	0,5016	0,8861	0,8974
Ecoli067vs5	9,09	0,8495	0,7645	0,8207	0,8072	0,5385	0,8741	0,5114	0,8488	0,8434
Yeast0256vs3789	9,14	0,7760	0,5843	0,5575	0,5007	0,4981	0,8011	0,6068	0,7673	0,7850
Ecoli01vs235	9,17	0,8507	0,5597	0,7459	0,7104	0,8612	0,8454	0,5504	0,8423	0,8534
Glass04vs5	9,22	0,9719	0,9927	0,9916	0,9689	0,9927	0,9276	0,5000	0,9709	0,9681
Yeast05679vs4	9,35	0,7830	0,7596	0,5000	0,5000	0,5000	0,7787	0,5099	0,7753	0,7785
Ecoli067vs35	10,00	0,8352	0,5803	0,792	0,7403	0,5842	0,8383	0,5236	0,8461	0,8440
Led7digit02456789vs1	10,97	0,8883	0,7076	0,5595	0,5000	0,8379	0,8667	0,5000	0,8850	0,8703
Cleveland0vs4	12,62	0,7121	0,5874	0,5711	0,5379	0,6487	0,6942	0,5001	0,7489	0,8745
Shuttle0vs4	13,87	1,0000	0,9897	0,9999	1,0000	1,0000	1,0000	0,5000	0,9992	0,9994
Glass4	15,47	0,7885	0,8306	0,5923	0,5082	0,7848	0,8079	0,5660	0,8971	0,8952
Ecoli4	15,80	0,8789	0,8801	0,7345	0,7314	0,7403	0,8068	0,5000	0,8698	0,8764
Abalone9vs18	16,40	0,7095	0,6434	0,5159	0,4999	0,4801	0,6906	0,6624	0,6768	0,6909
Glass016vs5	19,44	0,9131	0,9375	0,5279	0,4980	0,9273	0,8721	0,5009	0,9565	0,9431
Shuttle2vs4	20,50	0,9950	0,9650	0,9959	0,9850	0,9150	0,9150	0,5000	0,9950	0,9985
Glass5	22,78	0,9185	0,9367	0,5246	0,4988	0,9230	0,8812	0,5000	0,9392	0,9338
Yeast2vs8	23,10	0,7739	0,6462	0,7339	0,5614	0,5000	0,7739	0,5045	0,7960	0,8001
Yeast4	28,10	0,8048	0,7279	0,5000	0,5000	0,6033	0,8029	0,5007	0,7643	0,7770
Yeast1289vs7	30,57	0,6285	0,6037	0,5574	0,5031	0,5000	0,5832	0,5000	0,6266	0,6406
Yeast5	32,73	0,9521	0,8567	0,5828	0,5000	0,9481	0,9429	0,5000	0,9418	0,9494
Ecoli0137vs26	39,14	0,7997	0,7774	0,6410	0,5285	0,6836	0,7775	0,5000	0,7977	0,8184
Yeast6	41,40	0,8533	0,7822	0,5860	0,5000	0,5275	0,8504	0,5000	0,8272	0,8243
Abalone19	129,44	0,7100	0,5185	0,5027	0,5000	0,5281	0,6958	0,5808	0,5670	0,5721
Ranking		2,7333	5,1833	6,2500	7,2333	5,3166	3,8000	8,0333	3,5000	2,9500

Tabla 6.3: Resultados de AUC para conjuntos de datos binarios

Para analizar estadísticamente estos resultados, hemos llevado a cabo el test de Iman&Davenport [59]. Como ya hemos visto en experimentaciones anteriores, para estudiar el comportamiento de los algoritmos con respecto a una medida, este test calcula el *ranking* medio obtenido por los k algoritmos sobre N conjuntos de datos distribuidos de acuerdo a una distribución F con $(k-1)$ y $(k-1)(N-1)$ grados de libertad. La hipótesis nula que plantea es la de que todos los algoritmos se comportan igualmente bien. En el caso que nos ocupa, el intervalo crítico para un nivel de probabilidad de $\alpha = 0{,}01$ es igual a $C_0 = [0, (F_F)_{0,01,8,232} = 3{,}1982]$. Dado que el valor para el estadístico de Iman&Davenport es de 26,9765, que queda fuera del intervalo C_0, podemos rechazar la hipótesis nula, diciendo pues que existen diferencias significativas entre los algoritmos con respecto a los resultados de AUC.

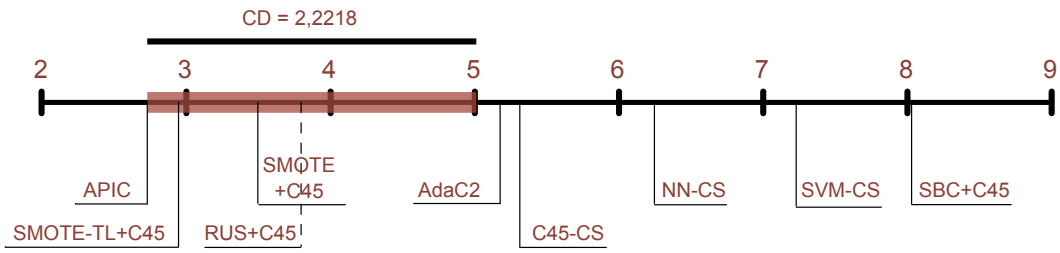


Figura 6.4: Test de Bonferroni–Dunn. APIC presenta diferencias significativas con respecto a aquellos clasificadores cuyo *ranking* está fuera del intervalo sombreado ($p < 0{,}01$)

Para descubrir entre qué algoritmos se dan diferencias significativas es necesario proceder con un test a posteriori. Elegimos a tal efecto el test de Bonferroni–Dunn, que se puede aplicar dado que todos los clasificadores se comparan frente a uno de control. La Figura 6.4 muestra gráficamente los resultados de este test siguiendo con el mismo nivel de confianza, $\alpha = 0{,}01$. La diferencia crítica obtenida es de 2,2818, representada por una línea horizontal más gruesa que el eje. Así pues, APIC se comporta significativamente mejor que aquellos algoritmos cuyo rango queda fuera de dicha línea: AdaC2, C45-CS, NN-CS, CSVM-CS, SBC-C4.5, en este orden. Además, nuestra propuesta obtiene resultados de AUC competitivos o incluso superiores a los de SMOTE-TL+C4.5, SMOTE+C4.5 y RUS+C4.5.

6.4.2. Resultados del estudio experimental sobre conjuntos de datos multiclase

La Tabla 6.4 reúne los resultados obtenidos por cada algoritmo sobre los conjuntos de datos multiclase. En este caso, el valor de IR de la segunda columna de la tabla corresponde a la razón de desbalanceo más alto entre cualquier par de clases, e igualmente se puede observar que los conjuntos de datos se encuentran ordenados de menor a mayor ratio de desbalanceo. Nuestra propuesta, APIC, obtiene los mejores resultados de AUC en diez de los quince conjuntos de datos.

Para comprobar si existen diferencias significativas, hemos aplicado el test de ranking de signos de Wilcoxon [59] por parejas. En la Tabla 6.5 se muestran los resultados del test de Wilcoxon sobre el AUC para calcular las comparaciones por

Data set	IR	APIC	OVO C4.5-CS	OVA C4.5-CS
Penbased	1,09	**0,9639**	0,9561	0,9285
Hayes-roth	1,70	0,7279	**0,8333**	0,7902
Contraceptive	1,88	**0,6765**	0,6626	0,6045
Wine	2,71	**0,9688**	0,9389	0,9257
New-thyroid	5,00	**0,9621**	**0,9621**	0,9404
Dermatology	5,55	0,9608	**0,9746**	0,9599
Balance	5,87	0,7381	0,7125	**0,7399**
Glass	8,44	**0,9486**	0,8423	0,8591
Autos	16,00	0,8406	0,8493	**0,8820**
Thyroid	39,17	0,9686	**0,9847**	0,9725
Lymphography	40,50	**0,9397**	0,7247	0,8070
Ecoli	71,50	**0,9465**	0,8100	0,8516
Pageblocks	164,00	**0,9370**	0,8984	0,8787
Yeast	295,80	**0,8988**	0,8166	0,7219
Shuttle	853,00	**0,9850**	0,9281	0,9695

Tabla 6.4: Resultados de AUC para conjuntos de datos multiclase

parejas entre el algoritmo APIC y los esquemas de descomposición OVO y OVA utilizando C4.5 CS como clasificador base. Como se puede observar, el método de descomposición OVO se comporta ligeramente mejor que el OVA, pero en cualquier caso nuestra propuesta se comporta estadísticamente mejor que ambas con niveles de confianza superiores al 95 % (reporta un p-value de 0,0472 para el OVO y de 0,03016 para el OVA, con lo que a estos niveles de significación se rechaza la hipótesis nula de que los algoritmos presentan un comportamiento similar).

VS	R^+	R^-	P-value	Hipótesis nula
OVO+C4.5CS	95,0	25,0	0,04792	Rechazada
OVA+C4.5CS	98,0	22,0	0,03016	Rechazada

Tabla 6.5: Test de Wilcoxon para el AUC

6.5. Conclusiones

En este capítulo se ha adaptado el algoritmo multiobjetivo de AP para clasificación desarrollado previamente de manera que tenga en cuenta las particularidades de los conjuntos de datos no balanceados, extrayendo reglas de clasificación adecuadas para este tipo de problemas.

Internamente, el nuevo algoritmo, que ha sido denominado APIC (*Ant Programming for Imbalanced Classification*), presenta numerosas diferencias con respecto a los algoritmos GBAP y MOGBAP. Su principal ventaja respecto a las técnicas propuestas hasta la fecha para clasificación no balanceada radica en que, a diferencia de éstas, permite tratar simultáneamente tanto conjuntos de datos binarios como multiclase. Además, APIC trabaja con el conjunto de datos directamente, sin necesidad de efectuar un remuestreo para balancear la distribución de instancias por clase.

Los resultados obtenidos demuestran que APIC alcanza resultados competitivos en ambos dominios de clasificación no balanceada.

7

Modelos de AP para minería de reglas de asociación

La tarea de ARM fue originalmente propuesta para el análisis de la bolsa de la compra, obteniendo relaciones frecuentes del tipo $A \rightarrow C$. Por ejemplo, en un supermercado, una regla del tipo *pañales* $\rightarrow$ cerveza sería indicativa de que si un cliente compra pañales entonces probablemente comprará también cerveza. De esta manera, es factible par el comercio explotar estas relaciones, situando los productos estratégicamente para favorecer sus intereses.

En la Sección 2.2.3 se introdujeron dos algoritmos clásicos para la extracción de reglas de asociación, el Apriori [5] y el FP-Growth [99]. Ambos algoritmos entran dentro de la categoría de métodos de búsqueda exhaustiva, que abordan la extracción de reglas en dos pasos: descubrimiento de *itemsets* frecuentes, y generación de reglas a partir de los mismos. Una desventaja de este tipo de enfoques radica en el hecho de que no son adecuados para tratar con conjuntos de datos grandes debido a sus requerimientos de memoria y tiempo de ejecución [194]. Por este motivo, ha existido un creciente interés en aplicar otro tipo de metodologías a la extracción de reglas de asociación, tales como los EAs [144, 148, 236] o PSO [130], que permiten

alcanzar buen rendimiento en situaciones donde el espacio de búsqueda es demasiado extenso como para emplear métodos de búsqueda deterministas. Además, estos métodos omiten la descomposición en dos pasos que emplean los basados en búsqueda exhaustiva, generando las reglas directamente.

Un ejemplo de GA para esta tarea es el **ARMGA** (*Association Rule Mining Genetic Algorithm*) [236]. Dicho algoritmo codifica cada individuo como una regla de asociación en un único cromosoma. Cada gen, a excepción del primero del cromosoma, representa un *item* del conjunto de datos. El primer gen es el encargado de indicar la separación entre los *items* que forman parte del antecedente y los que componen el consecuente. Al igual que cualquier GA, este algoritmo utiliza operadores genéticos para obtener nuevos individuos, evaluando su confianza relativa, que establece el nivel de relación entre el antecedente y el consecuente.

En los capítulos anteriores ha quedado patente la capacidad de la metaheurística de AP para la extracción de reglas de clasificación. Todos los algoritmos propuestos, ya sean monobjetivo o multiobjetivo que abordan la clasificación desde un punto de vista general o bien desde uno específico de clasificación no balanceada, se fundamentan en el uso de una CFG que define el espacio de búsqueda y guía la creación de individuos válidos. Dados los buenos resultados que hemos logrado alcanzar aplicando AP a la tarea de clasificación, y a que es posible adaptar la CFG que utilizan a la tarea de ARM, en este capítulo presentamos la primera aproximación para la extracción de reglas de asociación utilizando esta metaheurística. En concreto, se presentan dos algoritmos: el primero de ellos sigue un enfoque monobjetivo donde se introduce una función de *fitness* novedosa para medir la calidad de las reglas extraídas. El segundo algoritmo propuesto evalúa la calidad de los individuos desde un punto de vista basado en la dominancia de Pareto, optimizando simultáneamente el soporte y la confianza de las reglas y asignándoles un *fitness* basado en *ranking*. El rendimiento de ambos se verifica sobre 16 conjuntos de datos, comparando sus resultados frente a diferentes paradigmas tales como búsqueda exhaustiva, GAs y GP. Los resultados obtenidos son muy prometedores, y demuestran que AP también es una técnica apropiada para la tarea de asociación de DM, presentando algunas ventajas y careciendo de los problemas que presentan los métodos exhaustivos.

7.1. Minería de reglas de asociación con AP basada en gramática

Esta sección analiza en detalle las características comunes de los dos algoritmos de AP que se proponen. Concretamente, describe la CFG que emplean, la generación del espacio de estados, la función heurística que usan y la regla de transición que se aplica en el movimiento de las hormigas.

7.1.1. Entorno y codificación de los individuos

Los modelos de AP que se proponen se basan en una CFG que define todos los posibles estados que los individuos pueden visitar. Esta gramática juega el papel del conjunto de terminales y funciones en programación automática, restringiendo el espacio de búsqueda y asegurando la validez de cualquier solución encontrada. La gramática se expresa en BNF y se define como $G = (\Sigma_N, \Sigma_T, P, S)$, donde Σ_N es el conjunto de no terminales, Σ_T es el conjunto de terminales, P es el conjunto de reglas de producción, y S es el símbolo inicial de la gramática.

El entorno definido por la gramática permite a las hormigas comunicarse entre sí, y adopta la forma de un árbol de derivación, tal como se muestra en la Figura 7.1. A modo de ejemplo, esta figura muestra el árbol de derivación utilizando el conjunto de datos *weather* hasta una profundidad de cuatro derivaciones. Desde el estado inicial, las hormigas siguen un camino hasta alcanzar un estado final o solución, representado mediante un doble óvalo. Cada individuo almacena el camino seguido, cumpliendo con las propiedades de una hormiga artificial [154]. Por tanto, cada individuo codifica una regla de asociación, y el último estado del camino se corresponde con la expresión evaluable de la regla, expresada en notación prefija.

La gramática empleada por los algoritmos de AP para ARM se expresa como:

G = (Σ_N, Σ_T, P, S)

Σ_N = {<Rule>, <Antecedent>, <Consequent>, <Condition>}

Σ_T = {$\rightarrow$, AND, =, !=

attr$_1$, attr$_2$, ..., attr$_n$,

value$_{11}$, value$_{12}$, ..., value$_{1m}$,

value$_{21}$, value$_{22}$, ..., value$_{2m}$,

..., value$_{n1}$, value$_{n2}$, ..., value$_{nm}$}

P = {<Rule>:= <Antecedent><Consequent>,

<Antecedent>:= <Condition>| AND <Antecedent><Condition>,

<Consequent>:= <Condition>,

<Condition>:= todas las posibles combinaciones de la

terna operator attr value}

S = <Rule>

El tratamiento del espacio de estados es similar al seguido por los algoritmos de AP propuestos para la tarea de clasificación en esta tesis doctoral. Así, el algoritmo sigue un enfoque de construcción incremental de manera que, conforme se crean las hormigas, los estados que éstas visitan se almacenan en una estructura de datos a tal efecto. De esta manera no es necesario crear de antemano ni almacenar en memoria todo el espacio de búsqueda, que dependiendo de la dimensionalidad del conjunto de datos y del número de derivaciones que se fijen para la gramática puede ser realmente extenso. Esto se traduce en menos requerimientos computacionales.

7.1.2. Medidas de heurística

Los algoritmos de AP diseñados consideran una función heurística con dos componentes complementarias. La primera es una medida ligada a la cardinalidad de las

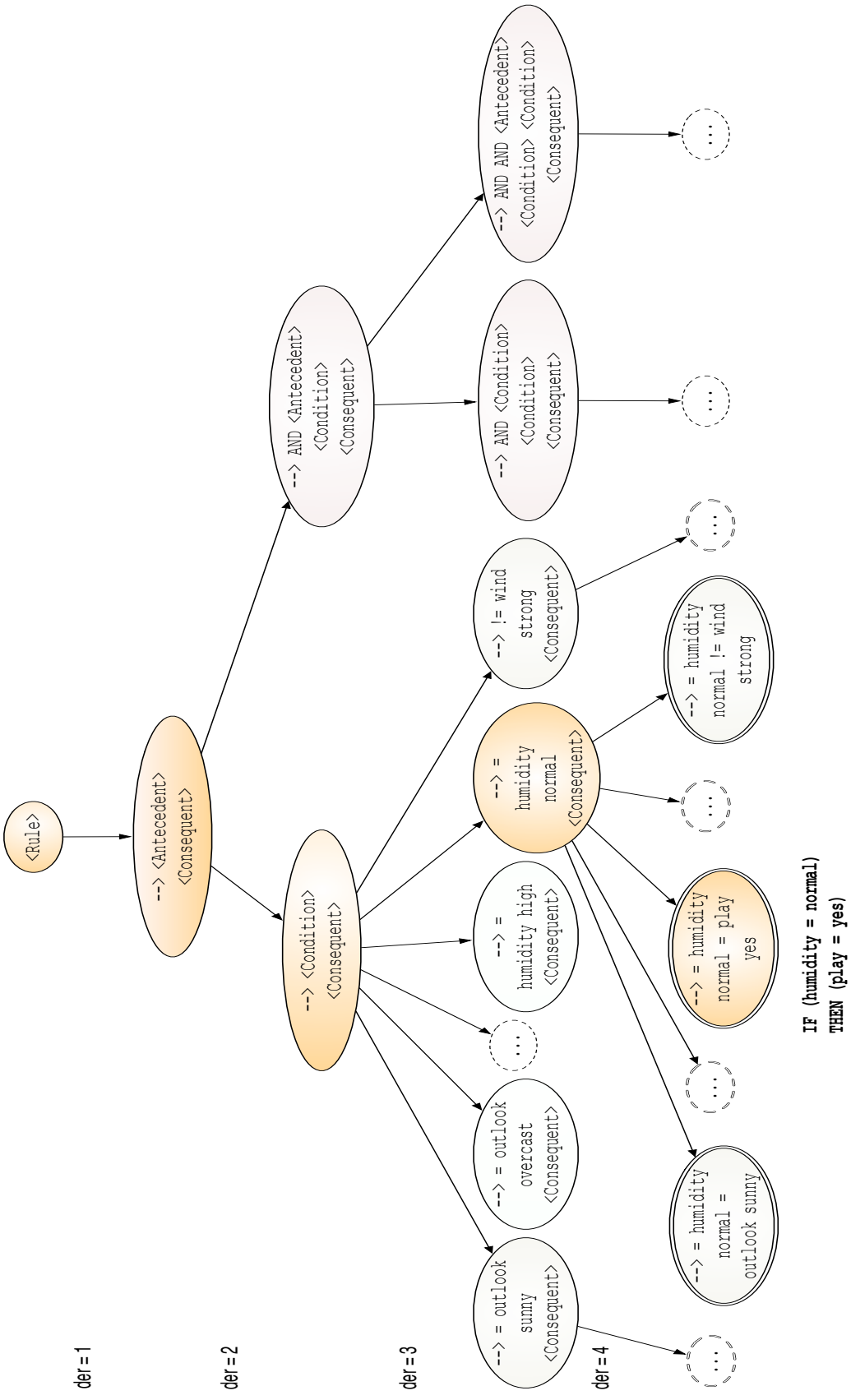


Figura 7.1: Ejemplo de espacio de estados a una profundidad de 4 derivaciones. El camino coloreado representa la regla de asociación codificada por una hormiga.

reglas de producción que se ven envueltas en una transición de un estado a otro, se basa en la medida de cardinalidad propuesta por *Geyer-Schulz* [93], y es la misma que se ha empleado en los algoritmos de clasificación de AP presentados en esta tesis. Como se ha indicado en los mismos, esta medida se aplica en las transiciones intermedias, que son aquellas que no suponen la selección de atributos del dominio del problema, dado que las transiciones que sí lo hacen permiten alcanzar siempre un número de soluciones equivalente, independientemente del estado que seleccione la hormiga como movimiento siguiente.

Dado que el objetivo de los algoritmos de ARM basados en AP que proponemos es la extracción de reglas frecuentes, que son aquellas que tienen un alto soporte, en las transiciones intermedias la métrica que se aplica es, precisamente, el soporte de la condición que codifica el nodo destino, cuyo cálculo se indica en la ecuación 2.13. El soporte se calcula previamente por cada posible combinación de operador, atributo y valor, en la fase de inicialización de la gramática. De la definición de soporte también se demuestra la propiedad antimonótona, recogida en la ecuación 2.14, que establece que el soporte de un *item* es siempre mayor o igual que el soporte de la unión de este *item* con otro. Esta propiedad se puede aprovechar en la fase de inicialización de la gramática, simplificando el cálculo de esta heurística, ya que dicha propiedad implica que no es posible encontrar reglas conteniendo ambos *items* y que tengan un soporte mayor que el de estos. Así, si uno de ellos ya tiene un soporte inferior al valor umbral, sabemos que no existirá regla alguna que lo contenga con soporte igual o superior que el umbral, con lo que se puede descartar. Nótese que, de acuerdo con la gramática que se ha definido en la sección previa, se permiten condiciones con los operadores '='y '!='. El operador '!=' favorece la obtención de reglas con un soporte alto en ciertos dominios donde hay muchos *itemsets* frecuentes.

7.1.3. Probabilidad de transición

Cada paso adelante en la búsqueda de un individuo por alcanzar una solución al problema supone la transición del estado actual a uno de los estados siguientes disponibles, y obedece una regla de probabilidad. Dicha regla tiene en cuenta tanto las feromonas esparcidas por el entorno por las hormigas de las generaciones previas como los valores proporcionados por la función heurística explicada en la sección

anterior. La regla recibe el nombre de regla de transición, y la probabilidad de que una determinada hormiga situada en el estado i en un momento determinado se mueva al estado j viene dada por:

$$P_{ij}^{k} = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\Sigma_{k \in allowed}(\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \tag{7.1}$$

donde:

- k es el número de posibles estados siguientes,
- α es el exponente de la función heurística,
- β es el exponente del nivel de feromonas,
- η es el valor de la función heurística, calculada como $sup(Condition) + P_{card}$ (siendo siempre uno de los dos componentes igual a cero), y
- τ indica la fuerza del rastro de feromonas en la transición.

7.2. El algoritmo GBAP-ARM

En esta sección se describen las características específicas del algoritmo monobjetivo de AP implementado, que ha sido denominado **GBAP-ARM**, presentando también el pseudocódigo del mismo.

7.2.1. Función de fitness

Tal y como se ha explicado en la Sección 7.1.1, el último estado de un camino encontrado por una hormiga codifica directamente la expresión de una regla de asociación. El primer paso para evaluar dicha regla consiste en comprobar si dicha expresión es consistente con una regla válida, es decir, si tanto el antecedente como el consecuente están compuestos por *itemsets* disjuntos. En caso contrario, la regla se descartaría. Una vez comprobado esto, se pasa a medir la aptitud del individuo.

GBAP-ARM evalúa la calidad de los individuos mediante una única función de *fitness*, que se muestra en la Ecuación 7.2, la cual mide la media armónica entre el soporte y la confianza.

$$fitness = \frac{2 \cdot support \cdot confidence}{support + confidence} \tag{7.2}$$

Durante la ejecución del algoritmo, lo que se pretende es que los individuos maximicen esta función de *fitness*, y su valor quedará comprendido en el intervalo $[0, 1]$. Esta función está inspirada en la F-medida (*F-measure*), ampliamente utilizada en problemas de recuperación de información y cuya formulación puede verse en la Ecuación 2.9, haciendo uso de sensibilidad y precisión. Como puede observarse, la función de *fitness* es la media armónica entre soporte y confianza. Por tanto, el valor de la función de *fitness* tenderá a ser más cercano a la medida que menor valor tenga (soporte o confianza). Así, un individuo que obtenga un *fitness* cercano a 1 implica que tanto el soporte como la confianza del mismo serán razonablemente elevados. También podemos considerar un parámetro γ para dar más énfasis al soporte que a la confianza o viceversa, como se indica en la Ecuación 7.3. Por tanto, para valores altos de dicho parámetro, los individuos tenderán a buscar reglas con mejores valores de confianza. Asimismo, para valores de γ inferiores a 1, se promueve el soporte. En cualquier caso, el valor de este parámetro durante la experimentación ha sido fijado a 1, como se observa en la Ecuación 7.2, dado que se pretende encontrar soluciones con un soporte y confianza balanceados.

$$fitness = \frac{(1 + \gamma^2) \cdot support \cdot confidence}{\gamma^2 \cdot support + confidence} \tag{7.3}$$

7.2.2. Población externa

El algoritmo GBAP-ARM mantiene una población externa con las mejores soluciones encontradas hasta el momento durante el proceso evolutivo. El experto que haga uso del algoritmo puede fijar el tamaño de esta población de reglas. Los individuos que ésta contenga al finalizar todas las generaciones del algoritmo serán devueltos al usuario como el mejor conjunto de reglas de asociación descubiertas.

Es importante recalcar que únicamente las reglas de asociación no redundantes se guardan en esta población externa. Así, cuando un individuo es candidato a ser

incluido en ella, se comprueba si existe ya alguna regla que sea equivalente a la que codifica dicho individuo, de manera que si esto sucede, su inclusión se descarta. Dos reglas se consideran equivalentes si cubren las mismas instancias del conjunto de datos, ya que, aunque pueden presentar diferentes atributos, se trata de diferentes formas de representar el mismo conocimiento.

La población se ordena de forma descendente por el *fitness* de los individuos y, en caso de empate, aquellos individuos con un soporte más alto tienen preferencia.

7.2.3. Mantemiento de feromonas

En el mantenimiento de feromonas se consideran dos operaciones, refuerzo y evaporación. Con respecto a la primera, todas las hormigas que se crean en la generación actual, junto con aquellas de la población externa, pueden depositar feromonas en su camino, siempre que su calidad sobrepase un umbral fijado experimentalmente en 0,5. Este valor umbral impide la influencia negativa en el entorno de las soluciones que no se consideran suficientemente buenas. La cantidad de feromonas esparcidas por una determinada hormiga será proporcional a su *fitness*, como se indica en la Ecuación 7.4.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot fitness \tag{7.4}$$

donde:

- $\tau_{ij}(t)$ indica la cantidad de feromonas que ya había en la transición del estado i al j,
- $\tau_{ij}(t)$ es la nueva cantidad de feromonas que habrá tras la la deposición de feromonas, y
- $fitness$ representa la calidad del individuo (Ecuación 7.2).

Todas las transiciones en el camino de un individuo reciben una cantidad de feromonas equivalente.

La evaporación tiene lugar sobre todas las transiciones del espacio de estados, y tras su aplicación, para una transición concreta, el valor de feromonas queda como

indica la Ecuación 7.5:

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1-\rho) \tag{7.5}$$

donde:

ρ indica la tasa de evaporación.

Por último, se introduce un proceso de normalización para limitar los niveles de feromonas de cada transición al rango $[\tau_{min}, \tau_{max}]$.

7.2.4. Algoritmo

La secuencia principal del algoritmo GBAP-ARM se detalla en el pseudocódigo del Algoritmo 7. Al principio, se inicializa la gramática y el espacio de estados, creando también una lista vacía de individuos *ants*. A continuación, el algoritmo lleva a cabo *numGenerations* iteraciones, efectuando las operaciones que se detallan a continuación en cada una de ellas.

En primer lugar, se crean *numAnts* individuos de un modo similar a como procedían el resto de algoritmos prensentados en esta tesis, y cuyo pseudocódigo se recoge en el Procedimiento 2. Conforme se van creando, estos individuos son evaluados y se rellena la estructura de datos que representa el espacio de estados con los estados que visita cada individuo. Los individuos se añaden a la lista *ants*. Nótese que en caso de que el individuo no alcance el soporte mínimo, sus estados no se almacenan, y tampoco se añade a la lista *ants*.

En segundo lugar, GBAP-ARM ordena la población de individuos actual en orden descendente por $fitness$, y a continuación crea una nueva lista denominada $externalPopulation$, donde almacena los mejores $extPopSize$ individuos encontrados. Para ello, empezando desde el individuo con mejor $fitness$, va insertándolos en la lista recién creada comprobando que no existía previamente otro individuo equivalente. Continúa insertando individuos hasta que la lista llega a su máxima capacidad, indicada por el parámetro $extPopSize$. Al mismo tiempo, va actualizando las feromonas del entorno de manera proporcional a la aptitud de cada individuo, siempre que su valor de $fitness$ sea superior a 0,5.

Algoritmo 7 Pseudocódigo de GBAP-ARM

Require: $dataSet, numGenerations, numAnts, maxDerivations, minSupport, extPopSize$

```
1: Inicializar gramática a partir de dataSet y espacio de estados
2: Lista ants ← {}
3: for i ← 0 to i < numGenerations inc 1 do
4:    for j ← 0 to j < numAnts inc 1 do
5:       ant ← Crear nueva hormiga (ver Procedimiento 2)
6:       Evaluar ant empleando la función de fitness (ver Ecuación 7.2)
7:       if soporte de ant ≥ minSupport then
8:          Almacenar estados visitados por ant en espacio de estados
9:          Añadir ant a la lista ants
10:      end if
11:   end for
12:   Ordenar individuos de ants en orden descendente por fitness
13:   Lista externalPopulation ← {}
14:   for each ant in ants do
15:      if tamaño de externalPopulation <= extPopSize then
16:         if externalPopulation no contiene ningún individuo igual o equivalente
            a ant then
17:            Insertar ant en externalPopulation
18:         end if
19:      end if
20:      if fitness > 0,5 then
21:         Actualizar la tasa de feromonas en el camino seguido por ant de manera
            proporcional a su fitness
22:      end if
23:   end for
24:   Evaporar las feromonas en el espacio de estados
25:   Normalizar los valores de feromonas
26:   ants ← externalPopulation
27: end for
28: return ants
```

7.3. El algoritmo MOGBAP-ARM

En lugar de utilizar una función de *fitness* con la agregación de las medidas de soporte y confianza, podemos guiar la evolución del algoritmo evaluando los individuos simultáneamente conforme a ambas medidas, siguiendo un enfoque multiobjetivo. Por este motivo se ha implementado el algoritmo **MOGBAP-ARM**.

Esta sección explica el algoritmo, centrándose en las características específicas que lo diferencian de la versión monobjetivo.

7.3.1. Evaluación multiobjetivo

La calidad de los individuos generados en MOGBAP-ARM se calcula en base a dos objetivos, soporte y confianza. Antes de calcular estas medidas, la regla de asociación codificada por el individuo ha de ser validada.

La dominancia de Pareto fue explicada en la Sección 5.1.2, y establece que un determinado individuo ant_1 domina a otro ant_2, representado como $ant_1 \succ ant_2$, en caso de que ant_1 no sea peor que ant_2 en ningún objetivo, y al menos sea mejor en uno de ellos. El conjunto de soluciones no dominadas de una población conforma el frente de Pareto.

Una vez que todos los individuos han sido evaluados de acuerdo a las dos medidas a optimizar, se calcula un valor de $fitness$ para cada uno utilizando rangos, organizando la población en frentes de individuos no dominados, de un modo similar a como procede el algoritmo NSGA2 [55]. El fitness asignado a un determinado individuo corresponde al valor inverso del número de frente en que se encuentre, empezando a numerar los frentes por 1 desde el frente de Pareto. Así, los individuos de dicho frente tendrán un $fitness$ de 1, los del segundo frente un valor de 1/2, y así sucesivamente. Una vez que se asigna un $fitness$ de rango a cada individuo, las operaciones de refuerzo y evaporación tienen lugar.

7.3.2. Almacenamiento de Pareto

MOGBAP-ARM emplea una población separada para almacenar los individuos del frente de Pareto descubiertos en cada generación. El tamaño de esta población no puede ser fijado. Además, a diferencia del comportamiento de la versión monobjetivo, la existencia de soluciones similares no se comprueba en este punto, sino que se posterga hasta el final del algoritmo, antes de devolver las soluciones encontradas al usuario. Así, esta población externa contendrá todas las soluciones no dominadas. En una generación dada, los individuos de la población externa encontradas en la generación previa se unen a los que se crean, calculando los nuevos frentes

que forman todos los individuos. Al finalizar el algoritmo, el conjunto de reglas de asociación devueltas se selecciona a partir de las hormigas no dominadas que componen la población externa una vez se eliminan las soluciones equivalentes.

7.3.3. Mantemiento de feromonas

MOGBAP-ARM pertenece al grupo de algoritmos multiobjetivo basados en hormigas que utilizan una única matriz de feromonas [12], lo que supone un beneficio en cuanto a tiempo de cómputo y requerimientos de memoria.

Otra diferencia con la versión monobjetivo radica en que en MOGBAP-ARM la cantidad de feromonas que un individuo puede depositar en su camino es inversamente proporcional al número de frente al que dicho individuo pertenece, como se indica en la Ecuación 7.6.

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \cdot rankFitness \tag{7.6}$$

donde:

$rankFitness$ es el valor de $fitness$ calculado para cada individuo en base al frente que ocupa, como se explica en la Sección 7.3.1.

En el algoritmo MOGBAP-ARM no hay ningún umbral que limite los individuos que pueden reforzar las feromonas, ya que la función de $fitness$ basada en el rango es suficiente para mantener fuera del entorno la influencia de las soluciones poco prometedoras (los individuos del segundo frente tienen un valor de *fitness* de 0,5; los del tercero, 0,33; y conforme menor sea el frente que ocupa el individuo, menos afecta el individuo al entorno).

Por último, los procesos de evaporación y normalización suceden tal como se ha descrito en la Sección 7.2.3.

7.3.4. Algoritmo

El pseudocódigo del algoritmo MOGBAP-ARM se muestra en el Algoritmo 8. Al igual que su homólogo monobjetivo, comienza por inicializar la gramática a partir de los metadatos que contiene el conjunto de datos *dataSet*, para crear a continuación la estructura que representa el espacio de estados, que en este punto únicamente contendrá el estado inicial. También crea una lista vacía de individuos, *ants*. A continuación se describen las operaciones que efectúa el algoritmo a lo largo de *numGeneraciones* ciclos.

Primero crea el número de hormigas especificadas por el experto, *numAnts*. Conforme se van creando, se evalúan de acuerdo a las medidas de soporte y confianza que hay que maximizar. Siempre que el soporte del individuo recién creado sea mayor que el umbral de soporte configurado, los estados que recorre en su camino hasta alcanzar la solución se guardan en la estructura del espacio de estados, y el individuo se añade a la lista *ants*.

Tras finalizar la fase de creación de los individuos, estos se ordenan de acuerdo a la dominancia de Pareto en diferentes frentes de individuos, de forma que los individuos que pertenezcan al frente N perciban un $rankFitness$ de $1/N$.

A continuación se inicializa una nueva lista denominada $fronts$, donde se incluirán aquellos individuos que forman el frente de Pareto, esto es, cuyo $rankFitness$ es igual a uno. Asimismo, con el objetivo de evitar la pérdida de diversidad, todos los individuos refuerzan la cantidad de feromonas de su camino proporcionalmente a su $rankFitness$, de manera que cada transición en el camino de una hormiga recibe idéntico refuerzo. Para concluir una generación del algoritmo, los procesos de evaporación y normalización tienen lugar, y la lista *ants* se remplaza con los individuos de la lista $fronts$, implementando de este modo el elitismo.

Una vez han concluido el número de generaciones indicadas por el parámetro *numGenerations*, con el propósito de seleccionar un conjunto apropiado de reglas de asociación para retornar al usuario, los individuos de la lista *ants* se ordenan descendentemente en base al soporte y, en caso de empate, por su confianza. Acto seguido, comenzando desde el primer individuo de la lista ordenada, se van añadiendo hormigas a la lista *associationRules*. Únicamente se añaden si no existe otra regla equivalente que se haya añadido con anterioridad. Por lo tanto, el

número de reglas de asociación devueltas se simplifica sin pérdida de información, de manera que no se dificulte la comprensibilidad al usuario. Por último, la lista *associationRules* es devuelta como resultado de la ejecución del algoritmo.

Algoritmo 8 Pseudocódigo de MOGBAP-ARM

Require: *dataSet*, *numGenerations*, *numAnts*, *maxDerivations*, *minSupport*

Inicializar gramática a partir de *dataSet* y espacio de estados
Lista *ants* ← {}
for $i \leftarrow 0$ **to** $i <$ *numGenerations* **inc** 1 **do**
  for $j \leftarrow 0$ **to** $j <$ *numAnts* **inc** 1 **do**
    ant ← Crear nueva hormiga (ver Procedimiento 2)
    Evaluar *ant* para las dos funciones objetivo
    if soporte de *ant* $\geq$ *minSupport* **then**
      Almacenar estados visitados por *ant* en espacio de estados
      Añadir *ant* a la lista *ants*
    end if
  end for
  for each *ant* **in** *ants* **do**
    rankFitness ← Asignar un $fitness$ de rango de acuerdo a la dominancia de Pareto
  end for
  Lista $fronts$ ← {}
  for each ant in *ants* **do**
    if rankFitness = 1,0 **then**
      Añadir *ant* a *fronts*
    end if
    Actualizar tasa de feromonas en el camino de *ant* proporcionalmente a su *rankFitness*
  end for
  Evaporar las feromonas en el espacio de estados
  Normalizar los valores de feromonas
  ants ← $fronts$
end for
Ordenar *ants* por *support*, y en caso de empate, por $confidence$
Lista *associationRules* ← {}
while $i <$ tamaño de *ants* **do**
  currentAnt ← individuo en posición i en *ants*
  if *associationRules* no contiene reglas equivalentes a *currentAnt* **then**
    Añadir *currentAnt* a *associationRules*
  end if
  $i \leftarrow i + 1$
end while
return *associationRules*

7.4. Experimentación

Esta sección presenta los conjuntos de datos y los algoritmos empleados en el estudio experimental, con la configuración de parámetros empleada para cada uno de ellos.

7.4.1. Conjuntos de datos y preprocesamiento

Para analizar la efectividad de los dos algoritmos de AP que se han propuesto, se han considerado dieciséis conjuntos de datos del repositorio de ML de la universidad de California en Irvine (UCI) [82]. Sus características se recogen en la Tabla 7.1, donde se muestran ordenados de menor a mayor número de instancias. Algunos de los conjuntos de datos tienen etiquetas y se pueden utilizar para aprendizaje supervisado. Por este motivo, el atributo que indica la clase se utiliza en los experimentos de manera análoga al resto, de forma que puede aparecer o bien en el antecedente o en el consecuente de una regla.

7.4.2. Algoritmos y configuración de parámetros

Para realizar una comparativa justa, se ha efectuado un estudio experimental independiente para cada uno de los algoritmos propuestos, considerando en el primero algoritmos monobjetivo y en el segundo multiobjetivo. Esta decisión ha sido adoptada debido a que los algoritmos multiobjetivo están pensados para obtener un conjunto de soluciones óptimas teniendo en cuenta varios objetivos a la vez, por lo que en el caso que nos ocupa se centran en buscar reglas de asociación con un buen compromiso entre los objetivos que optimizan.

Conjunto de Datos	Valores Perdidos	Instancias	Atributos	
			Cont.	*Nom.*
Lymphography	no	148	3	16
Wine	no	178	13	1
Sonar	no	208	60	1
Primary-tumor	yes	339	0	18
Soybean	yes	683	0	36
Australian	yes	690	6	10
Vehicle	no	846	18	1
Vowel	no	990	10	4
Credit-g	no	1000	6	15
Contraceptive	no	1473	2	8
Segment	yes	2310	19	1
Splice	no	3190	0	61
Chess	no	3196	0	37
Musk	no	6598	166	3
Nursery	no	12960	0	9
Connect-4	no	67557	0	43

Tabla 7.1: Características de los conjuntos de datos

Por un lado, en lo concerniente a la experimentación monobjetivo, el algoritmo GBAP-ARM se ha evaluado frente a otros algoritmos de ARM de diversos paradigmas: un algoritmo de GP basado en gramática, G3PARM[1]; un algoritmo genético, ARMGA[2]; y dos algoritmos de búsqueda exhaustiva, Apriori[3] y FP-Growth[4].

La configuración de parámetros para el algoritmo GBAP-ARM corresponde a un tamaño de población de 20 hormigas, 100 generaciones, un número máximo de 10 derivaciones, una cantidad de feromonas inicial de 1,0, una cantidad mínima de feromonas de 0,1, una tasa de evaporación de 0,05, un valor de 0,4 para el exponente

[1]El algoritmo G3PARM ha sido ejecutado empleando la implementación facilitada por los autores

[2]El algoritmo ARMGA fue ejecutado utilizando la implementación del software KEEL, disponible en http://www.keel.es

[3]El algoritmo Apriori fue ejecutado con la implementación del mismo existente en el software WEKA

[4]El algoritmo FP-Growth fue ejecutado utilizando la implementación de F. Coenen (2003), de la Universidad de Liverpool, http://www.csc.liv.ac.uk/~frans/KDD/Software/FPgrowth/fpGrowth.html

α, un valor de 1,0 para el exponente β, un 70 % para el umbral de soporte, y una tamaño máximo para el conjunto de reglas devuelto ha sido fijado a 20. Nótese que este algoritmo no requiere establecer umbral de confianza.

Para el algoritmo G3PARM se emplearon sus valores óptimos reportados por los autores: tamaño de población de 50 individuos, 100 generaciones, probabilidades del 70 % y 14 % para cruce y mutación, respectivamente, un número máximo de 24 derivaciones, un umbral de confianza del 90 % y un umbral de soporte del 70 %.

Los valores óptimos de los parámetros del algoritmo ARMGA son: tamaño de población de 50 individuos, 100 generaciones, 100 % de probabilidad de selección, 90 % de probabilidad de cruce, 1 % de mutación, longitud máxima de las reglas de 4 atributos, y 90 % para el umbral de confianza. ARMGA no usa umbral de soporte.

Para los algoritmos Apriori y FP-Growth se usaron los mismos umbrales de confianza y soporte empleados en el resto de algoritmos. Además, dado que ambos son métodos de búsqueda exhaustivos, se ha limitado el número máximo de reglas que pueden devolver a 2000000.

Por otro lado, en lo que que respecta a la configuración de la experimentación multiobjetivo, dos algoritmos de GP han sido considerados para su comparación con MOGBAP-ARM. Ambos son versiones multiobjetivo del algoritmo G3PARM que se incluye en la experimentación monobjetivo, NSGA-G3PARM [143], que está basado en el algoritmo NSGA2 (*Non dominated Sort Genetic Algorithm*) [55], y SPEA-G3PARM [143], basado en el algoritmo SPEA2 (*Strength Pareto Evolutionary Algorithm*) [246][5].

El algoritmo MOGBAP-ARM utiliza la misma configuración que la versión monobjetivo, a excepción del parámetro que limita el tamaño del conjunto de reglas devuelto, ya que en este algoritmo se devuelven todas las reglas no equivalentes del frente de Pareto encontrado en la última generación del algoritmo. Por su parte, la configuración para los algoritmos NSGA-G3PARM y SPEA-G3PARM ha sido la siguiente: población de 50 individuos, 100 generaciones, 90 % de probabilidad de cruce, 16 % de probabilidad de mutación, y tamaño máximo de derivaciones para la gramática de 24. Tan sólo difieren en un parámetro, el tamaño de la población

[5]NSGA-G3PARM y SPEA-G3PARM fueron ejecutados utilizando el código proporcionado por los autores.

auxiliar, ya que el segundo de ellos es el que presenta dicho parámetro. En este caso, se ha adoptado un tamaño de 20.

7.5. Resultados

En esta sección se muestran e interpretan los resultados obtenidos en ambos estudios experimentales.

7.5.1. Resultados del estudio experimental monobjetivo

Los resultados del primer estudio experimental incluyen comparaciones de los resultados medios de: soporte, en la Tabla 7.2; confianza, en la Tabla 7.3; número medio de reglas devuelto, en la Tabla 7.4; y cobertura, en la Tabla 7.5. Nótese que, en primer lugar, los resultados medios por cada conjunto de datos son calculados y, en segundo, en la última fila de cada tabla se indica la media obtenida para cada uno de ellos sumando las medias y diviendo por el número de conjuntos de datos. Los algoritmos estocásticos (GBAP-ARM, G3PARM y ARMGA) fueron ejecutados diez veces con diferentes semillas de números aleatorios, para no obtener resultados sesgados. En cambio, los de búsqueda exhaustiva fueron ejecutados una sola vez. Cada fila de las tablas indicadas muestra el resultado obtenido por cada algoritmo sobre cada conjunto de datos para la medida correspondiente. Las cifras en negrita indican el algoritmo que logra el mejor resultado para un conjunto de datos específico. A simple vista es posible observar que GBAP-ARM obtiene los mejores resultados de soporte en un 93 % de los problemas, y asimismo supera al resto de algoritmos en un 80 % en lo que respecta a la confianza del conjunto de reglas de asociación obtenidas. El algoritmo ARMGA es el que obtiene los mejores resultados en cuanto a confianza para todos los conjuntos de datos.

Por otro lado, teniendo en cuenta que Apriori y FP-Growth son métodos de búsqueda exhaustiva, encuentran todas las relaciones posibles que se adecúan a las restricciones impuestas de soporte y confianza mínimos. Sin embargo, no son capaces de encontrar ninguna regla de asociación para los conjuntos de datos *wine, vehicle, splice, musk* y *nursery* (indicado como n/a, *not available*). Esto se debe a que dichos algoritmos únicamente consideran el operador '=' en las condiciones de

Conjunto de datos	GBAP-ARM	G3PARM	ARMGA	Apriori	FP-Growth
Lymphography	**0,9649**	0,9301	0,1643	0,7605	0,7605
Wine	**0,8486**	0,8162	0,1675	n/a	n/a
Sonar	**0,8052**	0,7952	0,2142	0,7501	0,7501
Primary-tumor	**0,9732**	0,8435	0,0354	0,7568	0,7568
Australian	**0,9876**	0,9068	0,0889	0,7510	0,7510
Soybean	**0,9679**	0,9198	0,0892	0,7301	0,7368
Vehicle	**0,9719**	0,9335	0,0090	n/a	n/a
Vowel	**0,9420**	0,8910	0,0347	0,7881	0,7881
Credit-g	**0,9513**	0,9484	0,0180	0,7599	0,7599
Contraceptive	**0,8985**	0,8434	0,0075	0,7821	0,7821
Segment	**0,9771**	0,9754	0,0091	0,8828	0,8828
Splice	0,9993	**0,9997**	0,0301	n/a	n/a
Chess	**0,9876**	0,9010	0,1368	0,7595	0,7595
Musk	**0,9987**	0,9417	0,1616	n/a	n/a
Nursery	**0,7784**	0,7606	0,0038	n/a	n/a
Connect-4	**0,9977**	0,9520	0,1271	0,9173	0,9173
Media	**0,9418**	0,8971	0,0821	0,5370	0,5373

Tabla 7.2: Resultados de soporte obtenidos por los algoritmos monobjetivo

Conjunto de datos	GBAP-ARM	G3PARM	ARMGA	Apriori	FP-Growth
Lymphography	0,9919	0,9989	**1,0000**	0,9722	0,9722
Wine	0,9579	0,9645	**1,0000**	n/a	n/a
Sonar	0,9450	0,9431	**1,0000**	0,9432	0,9432
Primary-tumor	0,9887	0,9964	**1,0000**	0,9420	0,9420
Australian	0,9962	0,9997	**1,0000**	**1,0000**	**1,0000**
Soybean	0,9886	0,9990	**1,0000**	0,8856	0,9456
Vehicle	0,9948	0,9998	**1,0000**	n/a	n/a
Vowel	0,9863	0,9915	**1,0000**	0,9535	0,9535
Credit-g	0,9878	**1,0000**	**1,0000**	0,9193	0,9193
Contraceptive	0,9716	0,9703	**1,0000**	0,9193	0,9193
Segment	0,9952	0,9955	**1,0000**	0,9773	0,9773
Splice	**1,0000**	**1,0000**	**1,0000**	n/a	n/a
Chess	0,9952	0,9997	**1,0000**	0,9385	0,9385
Musk	0,9996	0,9999	**1,0000**	n/a	n/a
Nursery	0,9915	0,9921	**1,0000**	n/a	n/a
Connect-4	0,9991	0,9999	**1,0000**	0,9761	0,9761
Media	0,9877	0,9907	**1,0000**	0,6529	0,6567

Tabla 7.3: Resultados de confianza obtenidos por los algoritmos monobjetivo

las reglas. En cambio, tanto G3PARM como GBAP-ARM hacen uso del operador '!=', que permite encontrar resultados de soporte elevados en conjuntos de datos con *itemsets* no frecuentes. En el caso del algoritmo ARMGA, sí que es capaz de encontrar reglas en los conjuntos de datos indicados porque no considera umbral de soporte mínimo.

Conjunto de datos	GBAP-ARM	G3PARM	ARMGA	Apriori	FP-Growth
Lymphography	20,0	20,0	20,0	104,0	104,0
Wine	20,0	20,0	20,0	n/a	n/a
Sonar	20,0	20,0	20,0	98	98
Primary-tumor	20,0	20,0	20,0	209	209
Australian	20,0	20,0	20,0	2,0	2,0
Soybean	20,0	20,0	20,0	112650	47304
Vehicle	20,0	20,0	20,0	n/a	n/a
Vowel	20,0	20,0	20,0	5,0	5,0
Credit-g	20,0	20,0	20,0	11,0	11,0
Contraceptive	20,0	20,0	20,0	1,0	1,0
Segment	20,0	20,0	20,0	7,0	7,0
Splice	20,0	20,0	20,0	n/a	n/a
Chess	20,0	20,0	20,0	$2 \cdot 10^6$	$2 \cdot 10^6$
Musk	20,0	20,0	20,0	n/a	n/a
Nursery	20,0	20,0	20,0	n/a	n/a
Connect-4	20,0	20,0	20,0	$2 \cdot 10^6$	$2 \cdot 10^6$
Media	20,0	20,0	20,0	274205,8	269849,4

Tabla 7.4: Número de reglas medio obtenido por los algoritmos monobjetivo

Conjunto de datos	GBAP-ARM	G3PARM	ARMGA	Apriori	FP-Growth
Lymphography	**1,0000**	0,9959	0,8311	0,9865	0,9865
Wine	**1,0000**	0,9974	0,7865	n/a	n/a
Sonar	0,9731	0,9668	0,7115	**0,9807**	**0,9807**
Primary-tumor	0,9971	0,9999	0,2286	**1,0000**	**1,0000**
Australian	**1,0000**	**1,0000**	0,6551	0,7510	0,7510
Soybean	**1,0000**	**1,0000**	0,7288	**1,0000**	**1,0000**
Vehicle	**1,0000**	0,9856	0,1017	n/a	n/a
Vowel	**1,0000**	**1,0000**	0,4329	0,9889	0,9889
Credit-g	**1,0000**	**1,0000**	0,3340	0,9900	0,9900
Contraceptive	**0,9985**	0,9976	0,2186	0,7821	0,7821
Segment	**1,0000**	**1,0000**	0,2597	0,9961	0,9961
Splice	0,9999	**1,0000**	0,2342	n/a	n/a
Chess	**1,0000**	**1,0000**	0,9651	**1,0000**	**1,0000**
Musk	**1,0000**	**1,0000**	0,5779	n/a	n/a
Nursery	**1,0000**	0,9781	0,0449	n/a	n/a
Connect-4	**1,0000**	**1,0000**	0,8843	**1,0000**	**1,0000**
Media	**0,9980**	0,9950	0,4997	0,6547	0,6547

Tabla 7.5: Resultados de cobertura obtenidos por los algoritmos monobjetivo

Los algoritmos Apriori y FP-Growth devuelven un gran conjunto de reglas de asociación, obviamente. En cambio, los algoritmos bioinspirados extraen un conjunto reducido de reglas de alta calidad, indicativo de *itemsets* más interesantes. Por ejemplo, ARMGA extrae 20 reglas que tienen la máxima confianza media, a expensas de obtener valores de soporte muy bajos. Por su parte, GBAP-ARM extrae

20 reglas de asociación con una media de soporte muy alta que poseen, al mismo tiempo, resultados muy elevados de confianza.

Deteniéndonos en los resultados de cobertura, GBAP-ARM y G3PARM consiguen el mejor resultado, a pesar de que extraen menos reglas debido al límite impuesto por los tamaños de la población externa que utilizan. Como conclusión se puede afirmar que ambos algoritmos son capaces de minar reglas de asociación más representativas.

Para analizar estadísticamente estos resultados, hemos aplicado el test no paramétrico de comparación múltiple de Iman&Davenport [59]. El valor del estadístico para el soporte medio distribuido de acuerdo con una distribución F es igual a 63,7692; para la confianza, su valor es igual a 41,9733, y 12,0804 para el cobertura. A un nivel de significación de $\alpha = 0{,}01$, ninguno de ellos pertenece al intervalo crítico $C_0 = [0, (F_F)_{0,01,4,60} = 3{,}6490]$. Por tanto, la hipótesis nula de que los algoritmos se comportan igualmente bien puede ser rechazada para las tres medidas consideradas. Así, para revelar dónde se dan las diferencias de rendimiento, es necesario proceder con un test a posteriori. Los *rankings* medios obtenidos por cada algoritmo sobre cada medida estudiada se indican en la Tabla 7.11.

Algoritmo	Soporte	Confianza	Cobertura
GBAP-ARM	**1,06**	2,94	1,81
G3PARM	1,94	2,28	2,06
ARMGA	4,37	**1,16**	4,37
Apriori	3,84	4,34	3,37
FP-Growth	3,78	4,28	3,47

Tabla 7.6: Resultados medios de los algoritmos

Dado que todos los clasificadores se comparan frente a uno de control, es posible llevar a cabo el test de Bonferroni-Dunn [59]. La diferencia crítica al mismo nivel de significación $\alpha = 0{,}01$ es igual a 1,6905, con lo que se comprueba estadísticamente que GBAP-ARM y G3PARM obtienen estadísticamente mejores resultados de soporte que los demás algoritmos.

Para la medida de confianza, el algoritmo con *ranking* inferior es el ARMGA, que obtiene diferencias significativas frente a GBAP-ARM. Sin embargo, estos resultados se deben al hecho de que ARMGA obtiene reglas con una confianza muy alta

pero valores de soporte muy bajos, dado que se centra en maximizar la confianza sin prestar atención al soporte. Este comportamiento no es adecuado a la hora de extraer reglas frecuentes, dado que la mayoría de reglas de asociación que extrae este algoritmo, a pesar de obtener reglas muy fiables, corresponden a *itemsets* poco frecuentes. Por otro lado, para un nivel de $\alpha = 0{,}10$, GBAP-ARM es significativamente mejor en cuanto a la confianza que los algoritmos Apriori y FP-Growth. En este último caso la diferencia crítica según el test de Bonferroni-Dunn es de 1,2533.

Por último, fijándonos en la capacidad de cobertura de los algoritmos, GBAP-ARM es significativamente mejor que ARMGA con un 99 % de probabilidad, y mejor que Apriori y FP-Growth con un 95 % de probabilidad, donde la diferencia crítica de Bonferroni-Dunn se establece en 1.3964.

Al margen de este análisis, dado que el algoritmo GBAP-ARM ha sido ejecutado utilizando la función de *fitness* explicada en la Ecuación 7.2, que se corresponde directamente con la media armónica entre soporte y confianza, para ilustrar cómo afecta el parámetro γ hemos realizado diversos experimentos variando el valor del mismo. Así, utilizando los conjuntos de datos *nursery* y *soybean* a modo de ejemplo, se han probado diferentes valores de este parámetro, recogiendo los valores medios de soporte y confianza. Como se puede observar en las Figuras 7.2 y 7.3, estos resultados demuestran cómo el usuario puede determinar qué le interesa más a la hora de extraer las reglas, el soporte o bien la confianza. De hecho, para obtener el soporte máximo, se necesitan valores bajos de γ, mientras que cuanto mayor sea su valor, mayor será la confianza del conjunto de reglas de asociación extraído.

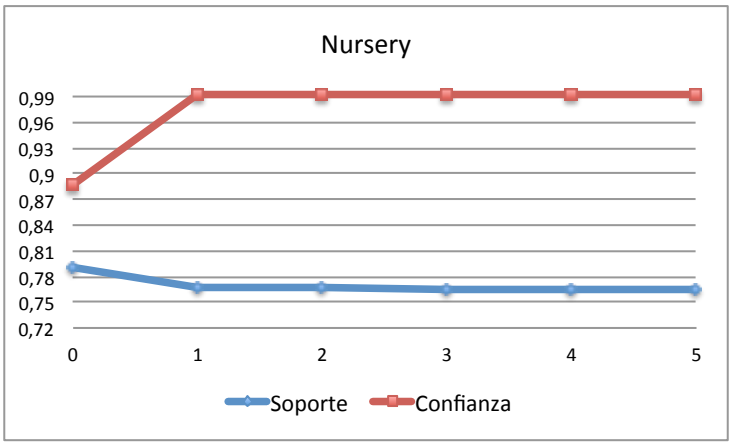


Figura 7.2: Análisis de sensibilidad para el parámetro γ en *nursery*

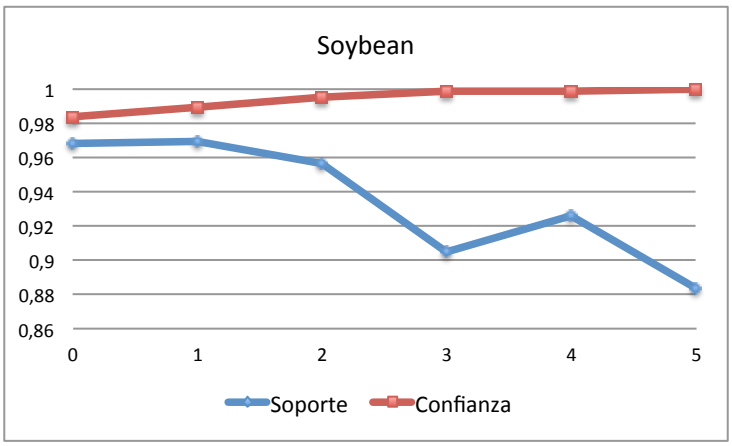


Figura 7.3: Análisis de sensibilidad para el parámetro γ en *soybean*

7.5.2. Resultados del estudio experimental multiobjetivo

En esta sección se muestran e interpretan los resultados del segundo estudio experimental, que abarca a los algoritmos multiobjetivo. Los tres algoritmos envueltos en este estudio son estocásticos, por lo que fueron ejecutados diez veces por cada conjunto de datos, calculando para cada uno la media obtenida en estas diez ejecuciones para las medidas de soporte (Tabla 7.7), confianza (Tabla 7.8), número de reglas (Tabla 7.9) y cobertura (Tabla 7.10). El ranking medio de estos tres algoritmos sobre cada medida se muestra en la Tabla.

El test de Iman&Davenport fue aplicado para comprobar si los algoritmos presentaban un comportamiento similar, en términos estadísticos. Para el caso particular del soporte, el estadístico calculado es igual a 20,5556; para la confianza, 20,2293; y para el cobertura, 2,5143. A un nivel de significación de $\alpha = 0{,}01$, el intervalo crítico es $C0 = [0, (F_F)_{0,01,2,30} = 5{,}3903]$, por lo que la hipótesis nula se rechaza para las medidas de soporte y confianza, mientras que se acepta para el cobertura que los tres algoritmos tienen un comportamiento similar. El test de Bonferroni-Dunn fue aplicado como test a posteriori, obteniendo las siguientes diferencias críticas: 0,6930 para una probabilidad del 90 %, 0.7926 para el 95 % y 0.9924 para el 99 %.

MOGBAP-ARM es significativamente mejor de acuerdo al soporte que el algoritmo SPEA-G3PARM con una probabilidad del 99 %. Aunque no obtiene diferencias

Conjunto de datos	GBAP-ARM	NSGA-G3PARM	SPEA-G3PARM
Lymphography	**0,9865**	0,9851	0,9361
Wine	0,7764	**0,8003**	0,6837
Sonar	**0,7888**	0,7410	0,6296
Primary-tumor	**0,9459**	0,8998	0,8084
Australian	0,9957	**0,9959**	0,9443
Soybean	0,9778	**0,9862**	0,9271
Vehicle	**0,9825**	0,9818	0,9291
Vowel	**0,9322**	0,7991	0,7442
Credit-g	0,8890	**0,9961**	0,9230
Contraceptive	**0,8400**	0,7081	0,5897
Segment	**0,9565**	0,9345	0,8637
Splice	**0,9997**	**0,9997**	**0,9997**
Chess	**0,9774**	0,9698	0,9265
Musk	0,9585	**0,9977**	0,6084
Nursery	**0,8000**	0,7306	0,6570
Connect-4	**0,9921**	0,9911	0,9638
Media	**0,9249**	0,9073	0,8209

Tabla 7.7: Resultados de soporte obtenidos por los algoritmos multiobjetivo

Conjunto de datos	GBAP-ARM	NSGA-G3PARM	SPEA-G3PARM
Lymphography	**1,0000**	0,9995	0,9942
Wine	**0,9873**	**0,9873**	0,9791
Sonar	0,9850	**0,9873**	0,9757
Primary-tumor	**0,9980**	0,9963	0,9897
Australian	**1,0000**	**1,0000**	0,9963
Soybean	**0,9994**	0,9989	0,9935
Vehicle	0,9989	0,9989	0,9957
Vowel	0,9954	**0,9967**	0,9906
Credit-g	0,9927	**1,0000**	0,9909
Contraceptive	0,9868	**0,9899**	0,9880
Segment	0,9983	**0,9992**	0,9965
Splice	**1,0000**	**1,0000**	**1,0000**
Chess	0,9997	0,9997	0,9988
Musk	**0,9999**	0,9994	0,9968
Nursery	**1,0000**	0,9885	0,9578
Connect-4	**0,9999**	0,9998	0,9996
Media	**0,9963**	**0,9963**	0,9902

Tabla 7.8: Resultados de confianza obtenidos por los algoritmos multiobjetivo

significativas respecto a NSGA-G3PARM, nuestra propuesta de AP consigue resultados competitivos e incluso mejores para esta medida.

Centrándonos en los resultados de confianza, MOGBAP-ARM y NSGA-G3PARM rinden de un modo similar, logrando idéntico *ranking*. Ambos obtienen diferencias significativas frente a SPEA-G3PARM a un nivel de confianza del 99 %.

Conjunto de datos	GBAP-ARM	NSGA-G3PARM	SPEA-G3PARM
Lymphography	1,9	2,1	20,0
Wine	5,4	5,1	20,0
Sonar	6,4	12,9	20,0
Primary-tumor	7,8	6,1	20,0
Australian	1,6	2,3	20,0
Soybean	2,1	2,7	20,0
Vehicle	1,8	2,3	20,0
Vowel	3,2	7,0	20,0
Credit-g	5,7	1,4	20,0
Contraceptive	12,2	14,7	20,0
Segment	3,7	4,2	20,0
Splice	17,2	48,9	20,0
Chess	4,0	5,0	20,0
Musk	2,4	3,3	20,0
Nursery	1,0	2,6	20,0
Connect-4	3,2	6,7	20,0
Media	5,12	8,29	20,0

Tabla 7.9: Número de reglas medio obtenido por los algoritmos multiobjetivo

Conjunto de datos	GBAP-ARM	NSGA-G3PARM	SPEA-G3PARM
Lymphography	0,9865	0,9865	**0,9966**
Wine	**0,9949**	0,9573	0,9938
Sonar	**0,9475**	0,9240	0,9284
Primary-tumor	**1,0000**	0,9847	**1,0000**
Australian	0,9957	0,9959	**0,9992**
Soybean	0,9855	0,9939	**1,0000**
Vehicle	0,9897	0,9912	**0,9941**
Vowel	0,9876	0,9844	**0,9985**
Credit-g	0,9994	0,9961	**1,0000**
Contraceptive	**0,9924**	**0,9924**	0,9888
Segment	0,9993	0,9961	**1,0000**
Splice	**0,9997**	**0,9997**	**0,9997**
Chess	**0,9997**	**0,9997**	0,9994
Musk	**1,0000**	**1,0000**	**1,0000**
Nursery	0,8000	0,9138	**1,0000**
Connect-4	**1,0000**	**1,0000**	**1,0000**
Media	0,9798	0,9822	**0,9937**

Tabla 7.10: Resultados de cobertura obtenidos por los algoritmos multiobjetivo

Por último, fijándonos en los resultados de cobertura, pese a que el test de Iman&Davenport indica que no hay diferencias significativas entre los algoritmos, SPEA-G3PARM logra los mejores resultados, probablemente a consecuencia del tamaño del conjunto de reglas de asociación devuelto como resultado, que es el mayor, compuesto de 20 reglas. MOGBAP-ARM queda segundo en *ranking*, empatado con el algoritmo

Algoritmo	Soporte	Confianza	Cobertura
MOGBAP-ARM	**1,5**	1,56	2,06
NSGA-G3PARM	1,62	**1,56**	2,34
SPEA-G3PARM	2,87	2,87	**1,59**

Tabla 7.11: Resultados medios de los algoritmos

NSGA-G3PARM, a pesar del hecho de extraer el menor número de reglas de los tres algoritmos. Por tanto, se puede concluir que, aunque el conjunto de reglas devuelto por MOGBAP-ARM contiene sólo aquellos individuos no equivalentes que componen el frente de Pareto de la última generación del algoritmo, dichos individuos codifican reglas muy representativas.

7.5.3. Comprensibilidad en el proceso de descubrimiento de conocimiento

Como se ha señalado anteriormente, los algoritmos presentados en este capítulo producen un conjunto de reglas de asociación en el formato *IF-THEN*. El proceso se lleva a cabo de manera automática a partir del conjunto de datos abordado y la configuración de parámetros empleada, siendo la salida el conjunto de las mejores reglas descubiertas durante la evolución que se adhieren a la gramática. Hay que resaltar que el único operador considerado en la gramática utilizada para la experimentación ha sido el operador AND. No obstante, la gramática se puede modificar para que tenga en cuenta otro tipo de operadores lógicos, de manera que otro tipo de reglas puedan ser extraídas de acuerdo al dominio de aplicación.

Las reglas obtenidas poseen varias propiedades deseables: son simples, intuitivas, fácilmente interpretables y proporcionan información representativa. A modo de ejemplo se muestran algunas reglas obtenidas por el algoritmo GBAP-ARM sobre el conjunto de datos *soybean*, que es un conjunto de datos de enfermedades ampliamente utilizado:

`IF` *class != rhizoctonia-root-rot* `THEN` *mycelium = present*

`IF` *int-discolor != black* `AND` *class != diaporthepod-&-stem-blight*

`THEN` *sclerotia = absent*

Asimismo se muestran algunas reglas obtenidas por el mismo algoritmo sobre el conjunto de datos *contraceptive method choice*, el cual contiene ejemplos acerca de mujeres en estado de embarazo o no junto con sus características socioeconómicas y demográficas:

`IF` *media-exposure != 1* `AND` *standard-of-living-index != 1*

`THEN` *husbands-education != 1*

`IF` *wifes-education != 1* `THEN` *husbands-ocupation != 4*

`IF` *number-of-children-ever-born* $!= (-inf, 0{,}5]$ `THEN` *husbands-education != 1*

7.6. Conclusiones

En este capítulo se ha explorado la aplicación de AP al descubrimiento de reglas de asociación frecuentes que poseen un nivel de cobertura de instancias y una fiabilidad altas. A tal efecto se han propuesto dos algoritmos diferentes, uno bajo un enfoque de evaluación monobjetivo y el otro, multiobjetivo. Un beneficio inherente de ambos, dado que se incluyen en la categoría de algoritmos bioinspirados, es que obtienen un buen rendimiento para la tarea de ARM sin requerir una gran cantidad de memoria, en contraste con los métodos de búsqueda exhaustivos. Dos estudios experimentales separados se han llevado a cabo, uno por cada perspectiva, ejecutando experimentos sobre diversos conjuntos de datos de diferente dimensionalidad y comparando los resultados con los obtenidos por otros algoritmos de ARM.

Como puntos más característicos de los algoritmos propuestos cabe resaltar el empleo de una CFG y de una función heurística con dos componentes. La gramática restringe el espacio de búsqueda y proporciona mayor flexibilidad, permitiendo utilizar diversos operadores lógicos. En cuanto a la versión monobjetivo, denominada GBAP-ARM, se ha presentado una función de $fitness$ que optimiza una agregación escalar de las medidas de soporte y confianza. El algoritmo multiobjetivo, llamado MOGBAP-ARM, calcula la calidad de los individuos generados de acuerdo a ambas medidas, soporte y confianza, simultáneamente, siguiendo un enfoque de Pareto.

Los resultados obtenidos en los dos estudios demuestran que la AP es una técnica muy competitiva para abordar la tarea de ARM.

8

Conclusiones y trabajos futuros

El trabajo efectuado durante la realización de la presente tesis doctoral y que se recoge en esta memoria no puede considerarse como cerrado, y mucho menos si se tiene en cuenta la amplitud de los temas abordados, su constante evolución, y sus posibles aplicaciones a otras áreas. Así, aunque en este capítulo se resumen brevemente los resultados alcanzados y las principales conclusiones derivadas de la consecución del presente trabajo, que permiten poner un punto y final a la tesis, también se introducen posibles ampliaciones y líneas de investigación que se pueden abordar en el futuro a partir de la misma. Finalmente se incluye una sección de aportaciones, donde se enumeran las publicaciones científicas asociadas a esta tesis.

8.1. Conclusiones

En general, en esta tesis se ha explorado la aplicación de la metaheurística de programación automática mediante colonias de hormigas a tareas de aprendizaje de DM, como son la clasificación (supervisado) y la extracción de reglas de asociación (no supervisado). Los resultados obtenidos son prometedores, y ponen de manifiesto que la AP es una técnica adecuada para abordar dichas tareas.

Todos los modelos de AP desarrollados en esta tesis implementan el enfoque de codificación de individuos de **enfoque** *individuo* = *regla* [70]. Además, se fundamentan en la utilización de una CFG que guía a los individuos hacia la búsqueda de soluciones válidas, y que dota de un **gran poder de representación** a las reglas extraídas. Es más, aparte de la **interpretabilidad** inherente a la extracción de reglas del tipo `IF` *antecedente* `THEN` *consecuente* frente a otras técnicas de caja negra como las NNs, métodos estadísticos o SVMs, entre otros, la modificación de la gramática permite **adaptar** los algoritmos a multitud de problemas. El uso de la gramática permite controlar factores relativos a la comprensibilidad, como por ejemplo especificar el tipo de condiciones que pueden aparecer en el antecedente de la regla, así como los operadores que conectan estas condiciones (mediante conjunciones, disyunciones, y cualquier otro operador lógico).

Por otro lado, y directamente relacionado con la interpretabilidad de las reglas, es posible forzar al algoritmo a encontrar reglas más sencillas a partir del parámetro que establece el **número máximo de derivaciones** para la gramática, ya que permite reducir el número de condiciones que aparecen en las reglas.

Otro factor diferenciador de los algoritmos de AP desarrollados se centra en la aplicación de **dos funciones heurísticas complementarias** en la regla de transición, según el tipo de transición en el que se encuentre el individuo que se está creando en ese momento. Así, se distinguen dos tipos de transiciones: finales e intermedias. Las primeras implican la selección de atributos del dominio del problema, mientras que las segundas no lo hacen. Donde coinciden todos los algoritmos es en la aplicación de la probabilidad de cardinalidad en las transiciones intermedias, mientras que difieren en el tipo de medida aplicada en las finales.

Todos los resultados obtenidos por los modelos de AP propuestos en la tesis han sido comparados frente a otros algoritmos de referencia de diferentes metaheurísticas. Estos resultados han sido contrastados mediante la utilización de **test estadísticos no paramétricos**, altamente recomendados para el análisis de resultados en problemas de DM, y que se recogen en el Anexo B.

Los siguientes puntos resumen de forma escueta los resultados conseguidos, presentando las conclusiones más relevantes.

- **Revisión bibliográfica**. Se ha efectuado una extensa revisión bibliográfica de los temas que se tratan en esta tesis. Como temas fundamentales mencionar el estudio realizado de las propuestas de ACO para clasificación y asociación, así como de los algoritmos de AP existentes en la literatura.

- **Algoritmo GBAP**. Se trata del primer modelo de AP que ha sido desarrollado para la tarea de clasificación de DM, tanto binaria como multiclase. Para evolucionar la población de individuos emplea una función de *fitness* monobjetivo. Implementa un enfoque de nichos para asignar un consecuente a las reglas y establecer su orden en el clasificador final, sin los inconvenientes que presentan los algoritmos de cubrimiento secuencial, con lo que no elimina instancias del conjunto de entrenamiento. Utiliza la ganancia de información como función heurística para las transiciones finales. Obtiene muy buenos resultados de exactitud predictiva y comprensibilidad frente a otros seis algoritmos sobre dieciocho conjuntos de datos.

- **Algoritmo MOGBAP**. Se ha presentado una versión multiobjetivo del algoritmo original, que mide la calidad de los individuos en la base de varios objetivos a optimizar simultáneamente, utilizando mecanismos para seleccionar un conjunto o frente de soluciones que abarcan un buen compromiso entre los mismos. En concreto, optimiza tres objetivos: sensibilidad, especificidad y una medida de la comprensibilidad. Además, se puede considerar como el primer algoritmo basado en hormigas para clasificación que sigue un enfoque de este tipo. Como contribución más significativa al campo cabe reseñar que se ha propuesto una estrategia multiobjetivo que puede ser adoptada por cualquier otro algoritmo bioinspirado de clasificación multiobjetivo, denominada estrategia de los k frentes de Pareto, y que distingue a los individuos en base a la clase que predicen. En cuanto a los resultados, obtiene resultados competitivos e incluso mejores que el algoritmo original monobjetivo.

- **Algoritmo APIC**. El problema de clasificación de datos no balanceados también se ha abordado, partiendo de los modelos desarrollados anteriormente. Así, se ha propuesto un algoritmo multiobjetivo y multicolonia que evoluciona los individuos para predecir cada clase de forma separada, existiendo una colonia por cada clase en el conjunto de datos. Con respecto a los otros dos algoritmos de AP para clasificación, APIC aplica una función heurística

diferente en las transiciones intermedias, la confianza de la clase, más adecuada para entornos no balanceados. APIC optimiza objetivos ampliamente utilizados en este tipo de dominio, como son la precisión y la sensibilidad, haciendo uso asimismo del AUC para seleccionar las reglas de cada colonia que integrarán el clasificador final, así como para ordenarlas adecuadamente en la lista de decisión. En cuanto a la experimentación, cabe destacar su exhaustividad, teniendo en cuenta el efecto *shift* que aparece frecuentemente en conjuntos de datos no balanceados. Las ventajas más significativas que presenta el algoritmo son su posibilidad de trabajar directamente con los conjuntos de datos, sin necesidad de llevar a cabo remuestreo alguno, además de poder aplicarse tanto a conjuntos de datos binarios como multiclase.

- **Algoritmo GBAP-ARM**. La extracción de reglas de asociación ha sido explorada mediante la metaheurística de AP. El primer algoritmo desarrollado descubre reglas de asociación frecuentes, evolucionando la población de individuos mediante una función de *fitness* novedosa que optimiza una agregación escalar de las medidas de soporte y confianza, inspirándose en la F-medida utilizada en problemas de recuperación de información. Los resultados obtenidos frente a otros algoritmos de extracción de reglas de asociación frecuentes, muestran que la metaheurística de AP es adecuada para abordar esta tarea de DM.

- **Algoritmo MOGBAP-ARM**. Se trata de un algoritmo multiobjetivo para el descubrimiento de reglas de asociación frecuentes. Este algoritmo optimiza simultáneamente el soporte y la confianza de los individuos, siguiendo un enfoque de Pareto. Su rendimiento ha sido comparado frente a otros algoritmos de ARM multiobjetivo pertenecientes al paradigma de GP, mostrando su capacidad de descubrir reglas muy representativas, obteniendo los mejores resultados en cuanto a soporte y confianza.

8.2. Líneas de trabajo futuras

A continuación se presentan posibles líneas de ampliación que se pueden plantear en el futuro partiendo de la base de los métodos propuestos en esta tesis doctoral.

8.2.1. Modelos de clasificación multietiqueta y de clasificación jerárquica

El algoritmo de ACO Ant-Miner ha sido adaptado con éxito a la tarea de clasificación multietiqueta, donde existen dos o más atributos de clase a predecir. En concreto, *Chan y Freitas* presentaron el algoritmo **MuLAM** (Multi-Label Ant-Miner) [41], realizando las modificaciones pertinentes sobre el algoritmo original de forma que en el consecuente de las reglas minadas pudiese aparecer más de un atributo de clase. Ello requirió un rediseño de muchas de las partes del algoritmo original.

Por otra parte, también se han presentado extensiones de Ant-Miner para la tarea de clasificación jerárquica, estrechamente relacionada con la multietiqueta. *Otero et al.* [170, 173] aplicaron su modificación del algoritmo a la predicción de funciones proteicas. Este problema es un campo de investigación muy activo, dado el inmenso número de proteínas no caracterizadas disponibles para su análisis y la importancia de determinar sus funciones para mejorar el conocimiento biológico actual. Las proteínas pueden llevar a cabo varias funciones y muchos de los esquemas funcionales disponen de una jerarquía, con lo que este problema se corresponde con el de clasificación jerárquica multietiqueta. En él, cada ejemplo puede pertenecer a múltiples etiquetas de clase, y las etiquetas de clase se organizan en una estructura jerárquica.

Asimismo, nuestro grupo de investigación ha desarrollado modelos de GP para clasificación multietiqueta [15] que también se basan en el esquema de codificación *individuo=regla*, empleado por los algoritmos presentados en esta tesis. Por este motivo y por el hecho de que algoritmos de ACO para clasificación han sido adaptados a este problema, consideramos factible desarrollar un modelo de AP para el problema de clasificación multietiqueta, así como para el problema de clasificación jerárquica, y esperamos obtener resultados de suficiente calidad en comparación con los publicados hasta la fecha.

8.2.2. Modelos coevolutivos de clasificación

Un paradigma relacionado especialmente con los modelos multiobjetivo aquí desarrollados es la coevolución de varias poblaciones de individuos [61, 136], donde varias poblaciones evolucionan en paralelo para resolver subproblemas, combinándose posteriormente para componer una solución global al problema. La coevolución guía el proceso evolutivo hacia un conjunto de cambios recíprocos en cada población, de manera que se mantengan adaptadas entre sí.

Existen diferentes formas de coevolución, que van desde la cooperación mutua a la competitiva. La cooperación entre las diferentes poblaciones se consigue, generalmente, utilizando funciones de evaluación conjuntas donde es necesario que intervenga un individuo de cada población para realizar la evaluación. Ésto permite beneficiar a ciertos individuos que tengan un comportamiento adecuado a la hora de colaborar con el resto de poblaciones, a diferencia de las funciones de evaluación monobjetivo que evalúan la calidad de los individuos respecto a un único objetivo. Por su parte, la coevolución competitiva significa que si el *fitness* de un individuo se incrementa, entonces el del competidor disminuye.

En el ámbito de la GP se han desarrollado algoritmos para la tarea de clasificación que hacen uso de la coevolución competitiva [13, 14], promoviendo la competición entre clasificadores y ejemplos con la finalidad de mejorar los resultados focalizándose en aquellos ejemplos que más dificultad presentan para ser categorizados. Por este motivo, y por los beneficios asociados a evolucionar varias poblaciones en paralelo que compitan o colaboren entre sí, planteamos la posibilidad de desarrollar modelos de este tipo para la metaheurística de AP.

8.2.3. Extracción de reglas de asociación poco frecuentes

A pesar de que, tradicionalmente, la minería de reglas de asociación ha sido estudiada desde el punto de vista del descubrimiento de relaciones frecuentes entre patrones, existen infinidad de campos donde estas relaciones no tienen por qué ser frecuentes (anomalías médicas [169], detección de fraudes bancarios [200], detección de intrusos [188], etc.). Los algoritmos que existen para extraer este tipo de patrones frecuentes están basados en métodos de búsqueda exhaustiva, con los inconvenientes que ello conlleva.

Se nos abre aquí un posible campo de estudio donde el descubrimiento de fuertes relaciones entre patrones, pero cuya ocurrencia sea rara o poco frecuente, es de claro interés para el usuario. Dados los buenos resultados obtenidos por los métodos de AP para asociación desarrollados en esta tesis, y las ventajas que presentan frente a los algoritmos de búsqueda exhaustiva, creemos conveniente modificarlos para abordar el descubrimiento de reglas de asociación poco frecuentes.

8.2.4. Descubrimiento de subgrupos

En la presente tesis doctoral se han propuesto una serie de modelos correspondientes a tareas predictivas así como otros correspondientes a tareas descriptivas. En la actualidad, existen tareas de DM que recaen en la intersección entre predictivas y descriptivas, entre las que se encuentra el descubrimiento de subgrupos (SD).

La tarea de SD fue propuesta por *Wrobel et al.* [235] y se definió como: "dada una población de individuos (clientes, objetos, etc.) y una propiedad de los mismos en la que estamos interesados, la tarea de descubrimiento de subgrupos consiste encontra. Por ejemplo, subgrupos tan grandes como sea posible que presenten las características estadísticas más inusuales con respecto a un atributo de interés". Desde su introducción, muchos investigadores han estudiado este concepto, presentando diferentes algoritmos [103]. Entre ellos, algunos son extensiones de algoritmos para clasificación [135], mientras que otros provienen de algoritmos de ARM [119]. Actualmente se trata de una tarea que recibe mucho interés por parte de la comunidad científica, y también se ha visto acrecentado el número de trabajos que la abordan desde un punto de vista evolutivo [58, 103].

En esta tesis se ha demostrado el potencial de la AP gramatical para extraer reglas de clasificación y asociación con un gran poder de expresión y una estructura flexible. Consideramos que es interesante desarrollar algún método de SD que se beneficie de dichas características, descubriendo subgrupos de interés para el usuario en modo de reglas con una estructura fácilmente comprensible.

8.3. Publicaciones asociadas a la tesis

8.3.1. Revistas internacionales

1. Título: *Using Ant Programming Guided by Grammar for Building Rule-Based Classifiers* [164]
   Autores: *J.L. Olmo, J.R. Romero and S. Ventura*

IEEE Transactions on Systems, Man & Cybernetics, Part B: Cybernetics, *Volume 41, Issue 6, pp. 1585–1599, December 2011*

Ranking:

Índice de impacto (JCR 2011): 3.080

Área de conocimiento:

Computer Science, Artificial Intelligence: 10/111

Computer Science, Cybernetics: 1/20

DOI: 10.1109/TSMCB.2011.2157681

2. Título: *Classification rule mining using ant programming guided by grammar with multiple Pareto fronts* [165]
   Autores: *J.L. Olmo, J.R. Romero and S. Ventura*

Soft Computing - A Fusion of Foundations, Methodologies and Applications, *Volume 16, Issue 12, pp. 2143-2163, 2012*

Ranking:

Índice de impacto (JCR 2011): 1.880

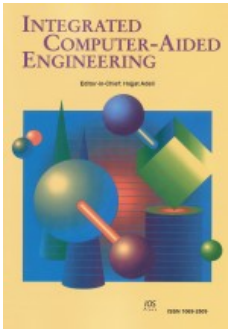
Área de conocimiento:

Computer Science, Artificial Intelligence: 30/111

Computer Science, Interdisciplinary Applications: 24/99

DOI: 10.1007/s00500-012-0883-8

3. Título: *Mining association rules with single and multi-objective grammar guided ant programming* [162]
Autores: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*



Integrated Computer-Aided Engineering, *pp. 1–17, 2013 (accepted)*

Ranking:

Índice de impacto (JCR 2011): 3.451

Área de conocimiento:

Computer Science, Artificial Intelligence: 7/111

Computer Science, Interdisciplinary Applications: 6/99

8.3.2. Conferencias internacionales

1. Título: *An automatic programming ACO-based algorithm for classification rule mining* [160]
   Autores: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*
   Trends in Practical Applications of Agents and Multiagent Systems - 8th International Conference on Practical Applications of Agents and Multiagent Systems (PAAMS), *Advances in Soft Computing Volume 70, pp. 649–656, 2010. Salamanca, Spain.*
   DOI:10.1007/978-3-642-12433-4_76

2. Título: *A grammar based ant programming algorithm for mining classification rules* [163]
   Autores: *J.L. Olmo, J.R. Romero and S. Ventura*
   IEEE Congress on Evolutionary Computation (IEEE CEC), *pp. 225–232, 2010. Barcelona, Spain.*
   DOI:10.1109/CEC.2010.5586492

3. Título: *Association rule mining using a multi-objective grammar-based ant programming algorithm* [161]
   Autores: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*
   11th International conference on Intelligent Systems Design and Applications (ISDA), *pp. 971–977, 2011. Córdoba, Spain.*
   DOI:10.1109/ISDA.2011.6121784

4. Título: *Multi-objective ant programming for mining classification rules* [166]
   Autores: *J.L. Olmo, J.R. Romero and S. Ventura*
   15th European Conference on Genetic Programming (EuroGP), *LNCS Volume 7244/2012, pp. 146–157, 2012. Málaga, Spain.*
   DOI:10.1007/978-3-642-29139-5_13

5. Título: *Binary and multi-class imbalanced classification using multi-objective ant programming* [159]
   Autores: *J.L. Olmo, A. Cano, J.R. Romero and S. Ventura*
   12th International conference on Intelligent Systems Design and Applications (ISDA), *pp. 70–76, 2012. Kochi, India.*
   DOI:10.1109/ISDA.2012.6416515

8.3.3. Conferencias nacionales

1. Título: *Minería de reglas de clasificación mediante un algoritmo de programación automática con hormigas*
   Autores: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*
   VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), *pp. 243–250, 2010. Valencia, Spain.*

2. Título: *Programación con hormigas multi-objetivo para la extracción de reglas de clasificación*
   Autores: *J.L. Olmo, A. Cano, J.R. Romero and S. Ventura*
   VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), *pp. 219–226, 2012. Albacete, Spain.*

9

Conclusions and future work

The work developed along the time that lasted the research for this Doctoral Thesis and that appears in this memory, cannot be thought of as concluded and even less if we have in mind the extent of the topics treated, their constant evolution and their possible application to other areas. Therefore, although in this chapter we will find a brief summary of the results obtained and the main conclusions that could derive from the consecution of this work, which allows to put an end to the Thesis, there is also a hint to possible extensions and new research lines that can be built up taking the present work as a basis. Finally, it is important to mention that the scientific publications associated to this Thesis are enumerated in Appendix 8.3.

9.1. Conclusions and contributions

In this Thesis we have mainly explored the application of the metaheuristic of automatic programming by using ant colonies to some DM tasks, such as classification (supervised) and the extraction of association rules (unsupervised). The results obtained are highly promising and show that AP is an appropriate technique to undertake these tasks.

All the AP models developed in this thesis encode individuals following the **enfoque** *individual* = *rule* scheme [70]. Besides, they get their foundation in the use of a CFG that guides the individuals towards the search of valid solutions, and that provides a **great power of representation** to the rules obtained. Even more, apart from the **interpretability** inherent to the extraction of rules of the type `IF` *antecedent* `THEN` *consequent*, as opposed to other black-box techniques such as ANNs, statistical methods or SVMs, among others, the modification of the grammar lets us **adapt** the algorithms to a great quantity of problems. The use of the grammar lets us, as well, control factors related to the comprehensibility, such as for example to specify the type of conditions that may appear in the antecedent of the rule, and the operators that connect these conditions (by means of conjuntions, disjunctions and any other type of logical operator).

On the other hand, and related directly to the interpretability of the rules, it is possible to force the algorithm to find simpler rules by configuring the parameter that establishes the **maximum number of derivations** for the grammar, since it permits to reduce the number of conditions that appear in the rules.

Another pattern of difference of the AP algorithms that have been here developed is centred in the application of **two complementary heuristic functions** in the transition rule, depending on the type of transition in which the individual that is being created at that moment is. Thus, two types of transitions are distinguished: final and intermediate. The former involves the selection of attibutes of the problem domain, while the latter does not do that. Where all the algorithms coincide is in the application of the probability of cardinality in the intermediate transitions, although they differ in the type of measure that is applied in the final ones.

All the results obtained by the AP models proposed in this Thesis have been compared in opposition to other algorithms of reference of different metaheuristics. These results have been checked statistically using **non-parametric tests**, highly reccomended for the analysis of results in problems of DM. These tests are explained in Appendix B.

The following points summarize the most relevant results achieved in the different chapters of this Thesis and the conclusions obtained in each.

- **Bibliographical revision**. A thorough bibliographical revision related to the topics treated in this Thesis has been done. As main topics it includes a study of the proposals of both ACO for classification and association and AP that exist in literature.

- **GBAP algorithm**. GBAP is the first AP model that has been developed for the classification task of DM, so much for the binary as for the multiclass. To evolve the population of individuals, it employs a single-objective fitness function. The algorithm implements a niching approach to assign a consequent to the rules and establishes their order in the final classifier, without the drawbacks that sequential covering algorithms present, since it does not remove instances of the training set. It uses the well-known information gain as heuristic function for the final transitions. GBAP gets very good results regarding predictive accuracy and comprehensibility of the rules mined, and its performance is compared against other six algorithms over eighteen data sets.

- **MOGBAP algorithm**. A multi-objective version of the original algorithm is brought up, measuring the quality of individuals in the basis of different objectives that are to be optimized simultaneously, using mechanisms to select a set or front of solutions that embrace a trade-off among them. Thus, the foundations of MOGBAP are presented, which optimizes three objectives: sensitivity, specificity and a comprehensibility measure. Apart from that, it can be considered as the first ant-based algorithm for classification that follows a multi-objective approach. We can consider as a significant contribution to the field of research the proposition of a new multi-objective strategy, that can be adopted by any other multi-objective bioinspired classification algorithm, called strategy of the k Pareto fronts and that distinguishes the individuals according to the class they predict. MOGBAP achieves competitive and even better results than the best obtained by the original single-objective algorithm.

- **APIC algorithm**. The problem of imbalanced classification data has also been addressed taking the previously developed models as a starting point. APIC is a multi-objective algorithm that evolves individuals to predict each class in a separate way, since it is a multicolony algorithm, having a colony for

each type of class in the data set. As for the other two algorithms of AP for classification, APIC applies a different heuristic function in the intermediate transitions, the class confidence, better adapted to imbalanced domains. APIC evolves individuals optimizing simultaneously precision and sensibility, which are more suitable objectives for imbalanced problems. It also makes use of the AUC to select the rules evolved by each colony that will make up the final classifier, sorting them out correctly in the decision list. A thorough experimentation has been carried out, having in mind the shift effect that appears frequently in imbalanced data sets. The most significant advantages that the algorithm presents are its possibility to work directly with the data sets without needing to resample any of them, and its possible application either for binary or multiclass data sets.

- **GBAP-ARM algorithm**. The extraction of association rules has been explored by means of the AP metaheuristic. This is the first algorithm that has been developed for discovering frequent association rules. It evolves the population of individuals by means of a novel fitness function, inspired from the information retrieval F-measure, that optimizes a scalar aggregation of support and confidence. The results obtained have been compared against other frequent ARM algorithms, showing that the AP metaheuristic is adequate to deal with this task of DM.

- **MOGBAP-ARM algorithm**. It is a multi-objective algorithm for discovering frequent association rules. The algorithm optimizes simultaneously the support and confidence of individuals by following a Pareto appproach. Its results have been compared to other multi-objective algorithms of ARM that belong to the paradigm of GP, showing its ability to discover very representative rules, obtaining the best results in what support and confidence are concerned.

9.2. Future research

In the following sections, the possible paths of expansion that can be proposed in a future moment having as a basis the methods offered in this Thesis are presented.

9.2.1. Multi-Label and hierarchical classification models

The Ant-Miner algorithm has been successfully adapted to the multi-label classification task, where there are two or more class attributes to predict. *Chan and Freitas* presented the MuLAM (Multi-Label Ant-Miner) algorithm [41], carrying out the necessary modifications on the original algorithm so that in the consequent of the rules mined could appear more than one class attribute. This required the redesign of many of the the parts of Ant-Miner.

On the other hand, extensions of Ant-Miner for the task of hierarchical classification have been presented, closely related to the multi-label. *Otero et al* [170, 173] applied their modification of the algorithm to the prediction of protein functions. This problem means a very active path of research, given the huge number of non characterized proteins ready for being analyzed and the importance of determining their function so as to improve the existent biological knowledge. Proteins can perform various actions and many of the functional schemes follow a hierarchy, and thus this problem can be addressed with multi-label hierarchical classification techniques. In it, each example can belong to multiple class labels, and the class labels get organized in a hierarchical structure.

Also, our research group has developed GP models for multi-label classification [15] that are based in the *individual=rule* encoding approach, used by the algorithms presented in this Thesis. Because of that and due to the fact that ACO algorithms have been addressed this problem with success, we find it feasible to develop AP models for the multi-label and hierarchical classification problems; and we hope to obtain results of enough quality compared to those published up to now.

9.2.2. Coevolutive classification models

A paradigm closely related to the multi-objective models presented in this work is the coevolution of populations of individuals [61, 136], where several populations evolve in parallel to solve subproblems, combining later to build a global solution to the problem. The coevolution guides the evolutive process towards a group of reciprocal changes in all the populations, so that they will keep adapted together.

There are different forms of coevolution that range from the mutual to the competitive cooperation. Cooperation between the different populations is obtained, generally, using functions of joint assessment where it is necessary that an individual from each population takes part to be able to perform the evaluation. This allows to promote certain individuals that have a suitable behavior when collaborating with the rest of populations, differently from the single-objective evaluation functions, that gauge the quality of individuals with respect to a single objective. On their part, the competitive coevolution means that if the fitness of an individual increasess, then that of the competitor decreases.

In the field of GP there have been developed algorithms for the classification task that make use of the competitive coevolution [13, 14], promoting the competition between classifiers and examples with the aim of improving the results putting the focus on those examples that show more difficulties to be categorized. This is why, and because of the associated benefits to the evolution of several populations in parallel, either by competing or collaborating, we pose the possibility of developing models of this type by using the AP metaheuristic.

9.2.3. Rare association rule mining

To date, association rule mining has mainly focused on the discovery of frequent patterns. Nevertheless, there are many domains where it is often interesting to focus on those relations that do not frequently occur (medical anomalies [169], bank fraud detection [200], intrusion detection [188], etc.). In addition, existing algorithms for mining this kind of infrequent patterns are mainly based on exhaustive search methods, with the drawbacks associated to this technique.

Therefore, it is interesting to explore the extraction of strong relations between patterns that appear scarcely or infrequently. Owing to the good results achieved by the AP methods developed in this Thesis for the ARM task, we think it addequate to modify and adapt them to the discovery of rare association rules.

9.2.4. Subgroup discovery

In this Thesis we have proposed AP models for the classification task of DM, which has a predictive nature, and others for the ARM task, which is a descriptive one. Nowadays, there are also tasks that lie in the intersection between predictive and descriptive DM, such as the subgroup discovery (SD).

The task of SD was originally proposed by *Wrobel et al.* [235], who defined it as: "given a population of individuals (customers, objects, etc.) and a property of those individuals that we are interested in, the task of subgroup discovery is to find population subgroups that are statistically most interesting for the user, e.g., subgroups that are as large as possible and have the most unusual statistical characteristics with respect to a target attribute of interest". Since its definition, researchers have studied this concept, presenting several algorithms [103]. Among them, some are extensions of classification algorithms [135], while others have been adapted from ARM methods [119]. At present, SD receives the interest of the research community, and the number of papers that address this task from an evolutionary perspective has been increased [58, 103].

On the other hand, the power that grammar-based AP presents for the extraction of comprehensible and flexible classification and association rules has been demonstrated in this Thesis. Therefore, we find it interesting to develop a SD algorithm based on this metaheuristic which profits of its advantages, discovering interesting subgroups for the user consisting of comprehensible rules.

9.3. Related publications

9.3.1. International journals

1. Title: *Using Ant Programming Guided by Grammar for Building Rule-Based Classifiers* [164]
   Authors: *J.L. Olmo, J.R. Romero and S. Ventura*

IEEE Transactions on Systems, Man & Cybernetics, Part B: Cybernetics, *Volume 41, Issue 6, pp. 1585–1599, December 2011*

RANKING:

Impact factor (JCR 2011): 3.080

Knowledge area:

Computer Science, Artificial Intelligence: 10/111

Computer Science, Cybernetics: 1/20

DOI: 10.1109/TSMCB.2011.2157681

2. TITLE: *Classification rule mining using ant programming guided by grammar with multiple Pareto fronts* [165]
AUTHORS: *J.L. Olmo, J.R. Romero and S. Ventura*

Soft Computing - A Fusion of Foundations, Methodologies and Applications, *Volume 16, Issue 12, pp. 2143-2163, 2012*

RANKING:

Impact factor (JCR 2011): 1.880

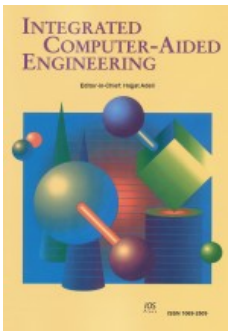
Knowledge area:

Computer Science, Artificial Intelligence: 30/111

Computer Science, Interdisciplinary Applications: 24/99

DOI: 10.1007/s00500-012-0883-8

3. TITLE: *Mining association rules with single and multi-objective grammar guided ant programming* [162]
   AUTHORS: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*



Integrated Computer-Aided Engineering, *pp. 1–17, 2013 (*In press*)*

RANKING:

Impact factor (JCR 2011): 3.451

Knowledge area:

Computer Science, Artificial Intelligence: 7/111

Computer Science, Interdisciplinary Applications: 6/99

9.3.2. International conferences

1. TITLE: *An automatic programming ACO-based algorithm for classification rule mining* [160]
   AUTHORS: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*
   Trends in Practical Applications of Agents and Multiagent Systems - 8th International Conference on Practical Applications of Agents and Multiagent Systems (PAAMS), *Advances in Soft Computing Volume 70, pp. 649–656, 2010. Salamanca, Spain.*
   DOI:10.1007/978-3-642-12433-4_76

2. TITLE: *A grammar based ant programming algorithm for mining classification rules* [163]
   AUTHORS: *J.L. Olmo, J.R. Romero and S. Ventura*
   IEEE Congress on Evolutionary Computation (IEEE CEC), *pp. 225–232, 2010. Barcelona, Spain.*
   DOI:10.1109/CEC.2010.5586492

3. TITLE: *Association rule mining using a multi-objective grammar-based ant programming algorithm* [161]
   AUTHORS: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*
   11th International conference on Intelligent Systems Design and Applications (ISDA), *pp. 971–977, 2011. Córdoba, Spain.*
   DOI:10.1109/ISDA.2011.6121784

4. TITLE: *Multi-objective ant programming for mining classification rules* [166]
   AUTHORS: *J.L. Olmo, J.R. Romero and S. Ventura*
   15th European Conference on Genetic Programming (EuroGP), *LNCS Volume 7244/2012, pp. 146–157, 2012. Málaga, Spain.*
   DOI:10.1007/978-3-642-29139-5_13

5. TITLE: *Binary and multi-class imbalanced classification using multi-objective ant programming* [159]
   AUTHORS: *J.L. Olmo, A. Cano, J.R. Romero and S. Ventura*
   12th International conference on Intelligent Systems Design and Applications (ISDA), *pp. 70–76, 2012. Kochi, India.*
   DOI:10.1109/ISDA.2012.6416515

9.3.3. National conferences

1. TITLE: *Minería de reglas de clasificación mediante un algoritmo de programación automática con hormigas*
   AUTHORS: *J.L. Olmo, J.M. Luna, J.R. Romero and S. Ventura*
   VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), *pp. 243–250, 2010. Valencia, Spain.*

2. TITLE: *Programación con hormigas multi-objetivo para la extracción de reglas de clasificación*
   AUTHORS: *J.L. Olmo, A. Cano, J.R. Romero and S. Ventura*
   VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB), *pp. 219–226, 2012. Albacete, Spain.*

A

Preparación y preprocesado de datos

La fase de preparación de datos supone una parte muy importante del tiempo y el esfuerzo que se dedica a un proceso de KD. Se trata de una cuestión esencial a tener en cuenta en el aprendizaje, dado que en ella se puede generar un conjunto de datos reducido del original, lo que puede repercutir en una mejora de la eficiencia del proceso; los datos originales pueden ser impuros, lo que puede desembocar en la extracción de conocimiento erróneo o poco útil; y permite generar datos de mayor calidad, conducentes a la obtención de conocimiento de mayor calidad o más útil.

Así, la preparación y preprocesado de datos incluye todas aquellas técnicas de análisis y transformación de datos que permiten mejorar la calidad de los mismos, de tal manera que los algoritmos de DM sean capaces de inducir conocimiento de mayor precisión e interpretabilidad.

El tamaño, la dimensionalidad del conjunto de datos y las características de los atributos son algunos de los factores que más influyen en el esfuerzo computacional.

La dimensionalidad se refiere al número de atributos y la complejidad de los mismos (nominales, numéricos, presencia de valores perdidos, etc.). La **maldición de la dimensionalidad** (*the curse of dimensionality*), expresión acuñada por *Bellman* [22], plantea que el espacio de búsqueda crece de manera exponencial con respecto al número de características, presentando un obstáculo a la hora de resolver los problemas. Por tanto, a veces se hace necesario simplificar los datos que el algoritmo va a procesar, reduciendo el número de atributos iniciales o su número de valores.

Cada instancia de un determinado conjunto de datos se representa utilizando un mismo conjunto de características o atributos, que pueden ser **binarios**, **nominales/categóricos**, o **continuos** [183]: los atributos nominales o categóricos pueden tomar una serie de valores discretos que no siguen necesariamente un orden lineal; los binarios son un tipo de atributo categórico que pueden tomar solo dos valores discretos, 0 o 1; y los continuos, un conjunto de valores infinito no numerable. Existen multitud de algoritmos de DM que requieren que los datos sean de un determinado tipo para poder tratar con ellos. Por este motivo, existen acciones de preprocesado que van encaminadas a transformar los datos originales al tipo con que pueden trabajar dichos algoritmos.

En este capítulo se introducen tres de las cuestiones que más frecuentemente se presentan en la fase de preparación de datos de un proceso de KD: la ausencia de datos, la existencia de multitud de atributos o características, y la necesidad de transformar datos cuantitativos en datos cualitativos.

A.1. Ausencia de datos

Uno de los problemas más habituales que se presentan durante la fase de preparación de datos es la falta de valores para un atributo determinado, lo cual puede deberse a un error en la toma de los datos o a que no se ha podido observar esta característica en ellos o en una serie de casos [20].

La solución más fácil supone ignorar la instancia que presenta el valor perdido, pero obviamente esto implica la pérdida de la información que contiene. Así pues, la forma habitual de proceder en estas situaciones consiste en remplazar el valor que falta por otro, que dependerá del tipo de atributo de que se trate. Se presentan diversas alternativas [122]:

- En caso de que estemos ante un atributo numérico, el valor perdido podría sustituirse por la media del resto de valores del mismo; mientras que si se trata de un atributo nominal, se sustituiría por el valor que más se repite (moda).
- Un método más elaborado consiste en inducir un modelo de clasificación o regresión, dependiendo de si el valor perdido pertenece a un atributo categórico o a uno numérico, respectivamente. Este modelo tratará el atributo del valor perdido como salida, y el resto de características como atributos predictivos. Una vez extraído el modelo a partir del conjunto de datos completo, se determina el valor perdido de la instancia en cuestión.
- Identificar la instancia del conjunto de datos más semejante a la que tiene el valor perdido, sustituyéndolo por el valor que presenta en ella.
- Modificar el rango de valores posibles para la característica que contiene valores perdidos, incluyendo un nuevo valor que englobe los casos perdidos.

No obstante, hay situaciones en las que un atributo concreto presenta un número elevado de valores perdidos, en cuyo caso no es aconsejable sustituirlos como se acaba de indicar, ya que no existe un número suficientemente representativo de ejemplos sobre los que calcular un valor fiable con que remplazarlo. Lo normal en estas circunstancias es no tratar este atributo, descartándolo.

A.2. Reducción de dimensionalidad

Una forma de limitar los efectos de la alta dimensionalidad consiste en reducir la dimensión del espacio de trabajo. Con las operaciones de reducción de dimensionalidad se pretende mantener las características de los datos iniciales, eliminando

aquellos que no son relevantes para la tarea de DM a realizar. De esta forma se reduce la dimensionalidad, permitiendo que la ejecución de algoritmos de DM sea más rápida y más efectiva.

En este contexto de reducción de dimensionalidad se pueden considerar, generalmente, tres **tipos de características** [238]:

- Relevantes, que son aquellas que influyen en la salida y cuyo rol no es asumible por las demás.
- Irrelevantes, si no tienen ningún tipo de influencia en la salida.
- Redundantes, si existe otro atributo que ya describe el mismo comportamiento en los datos.

Las acciones de reducción de dimensionalidad o de transformación de datos tienen sentido tanto en tareas de aprendizaje supervisado como no supervisado. Los motivos que llevan a efectuarlas son la eliminación de redundancia; la obtención de conjuntos de datos con información más condensada; o la proyección de datos con una alta dimensionalidad en un espacio de dimensionalidad reducida, de modo que sea posible descubrir visualmente *clusters* y otras relaciones en los datos, imposibles de observar en el espacio de dimensiones original. Podemos encontrar otro motivo adicional para el caso de tareas de aprendizaje supervisado, que sería la obtención de conjuntos de datos transformados conteniendo exclusivamente características relevantes para el diseño de clasificadores.

La reducción de dimensionalidad se puede llevar a cabo seleccionando un subconjunto de atributos o bien mediante transformaciones, remplazando los atributos iniciales por proyecciones de los mismos.

La selección de un subconjunto de atributos para reducir la dimensionalidad del conjunto de datos, eliminar el ruido y mejorar el desempeño recibe el nombre de **selección de características** [97]. Este proceso consiste en detectar el subconjunto óptimo de atributos o características fuertemente relevantes y las débilmente relevantes pero que no sean redundantes.

Existen diversas técnicas para realizar proyecciones, como el **análisis de componentes principales** [3], que usa proyecciones lineales, o el uso de **mapas autorganizativos de Kohonen** [125], entre otras, que proyecta los patrones de una forma no lineal.

A.3. Discretización de atributos numéricos

La discretización es una técnica de preprocesamiento de datos consistente en establecer un criterio por el cual el rango de valores de un atributo continuo se transforma en dos o más valores discretos que representan intervalos, siempre formando conjuntos disjuntos.

Entre los motivos por los cuales puede ser interesante discretizar un conjunto de datos podemos citar que determinados algoritmos de DM solo pueden operar en espacios de búsqueda discretos; que se busque una reducción del coste computacional y un aumento en la velocidad del proceso de aprendizaje; o que el modelo resultante sea de menor tamaño y más comprensible. De hecho, la discretización también puede considerarse como una forma de reducir la dimensionalidad del problema.

En cualquier caso, hay que tener en cuenta que no se puede producir una pérdida de información relevante durante el proceso de discretización. Por ejemplo, si clasificando un determinado conjunto de datos sin discretizar obtenemos una tasa de error inferior a la conseguida con los datos discretizados, de poco sirve efectuar la discretización.

Desde el punto de vista de los clasificadores basados en reglas, que son en los que nos centramos en esta tesis, su mayor ventaja radica en la comprensibilidad que ofrecen. Además, pese a que algunos de ellos pueden tratar directamente con atributos numéricos, algunos expertos aconsejan discretizar dichas características antes de inducir un modelo, de manera que se reduzca el tiempo de la fase de entrenamiento y se incremente la exactitud [10].

Existen métodos de discretización no supervisados y supervisados, aunque se pueden distinguir otros criterios bajo los que agruparlos, tal como se recoge en el trabajo de *Liu et al.* [141]. Los métodos supervisados solo son aplicables cuando

los datos tienen una etiqueta de clase, y los intervalos que producen están de alguna forma correlacionados con el atributo clase. En cambio, en los métodos no supervisados no se dispone de información acerca de la tarea que tiene que cumplir el modelo que se construirá a partir de los datos discretizados. Dos técnicas de este tipo son la partición en intervalos de la misma amplitud y en intervalos de igual frecuencia [230].

En la discretización en **intervalos de la misma amplitud**, por cada atributo continuo se localizan los valores mínimo y máximo, y entonces se divide este rango en un número de intervalos n de igual amplitud especificados por el usuario. El problema de este método es que se pueden producir desequilibrios, de manera que la mayoría de las instancias se concentren en un único intervalo y, a su vez, que existan intervalos que no estén representados por ninguna.

Del mismo modo, en la discretización en **intervalos de igual frecuencia** se buscan en primer lugar los valores mínimo y máximo para una determinada característica. Sin embargo, a continuación se ordenan todos los valores que presentan las instancias en orden ascendente. Una vez hecho esto, se divide el rango en un número de intervalos definido por el usuario, de manera que cada intervalo contenga aproximadamente el mismo número de valores ordenados. Generalmente, este método es preferible ya que evita desequilibrios en el balanceo entre valores.

Los métodos de discretización supervisados están enfocados hacia la tarea de clasificación, ya que se dispone de información relativa a la clase. En este caso, los algoritmos utilizan medidas estadísticas de separabilidad o de teoría de la información para encontrar los puntos de corte óptimos para un determinado atributo continuo, dividiendo así su rango de valores en los intervalos más adecuados para llevar a cabo dicha tarea [117]. Un ejemplo es el algoritmo **CAIM** (*Class-Attribute Interdependency Maximization Algorithm*) [133], el cual divide cada uno de los intervalos existentes en dos nuevos intervalos, de tal forma que se consigue una interdependencia óptima entre la clase y el atributo tras la división. Empieza con el intervalo completo de un atributo continuo y lo va dividiendo iterativamente hasta que cada atributo discretizado tenga la menor cantidad de intervalos posible que minimice la pérdida entre clase y atributo, número que al menos será igual al número de clases que haya en el conjunto de datos. A pesar de que CAIM obtiene muy buenos resultados frente al resto de técnicas, presenta dos inconvenientes. Por

un lado, genera esquemas de discretización donde el número de intervalos es menor o igual que el número de clases. Y por otro, para cada intervalo discretizado, el algoritmo considera únicamente la clase mayoritaria e ignora todas las demás, lo cual tiene efectos negativos cuando se trata de discretizar conjuntos de datos no balanceados.

Entre los algoritmos de discretización supervisados, destaca la propuesta de *Fayyad e Irani* [73], conocida como **discretización basada en la entropía** [202]. También sigue un enfoque descendente, empezando por el intervalo completo de un atributo numérico. Para determinar el punto de corte, selecciona el valor del atributo que tiene mínima entropía, y recursivamente particiona los intervalos resultantes hasta llegar a un criterio de parada. Se consideran dos criterios de parada: alcanzar el número máximo de intervalos permitido, o que la entropía caiga por debajo de un umbral establecido. Además, se demuestra que cualquier punto de corte candidato que minimice la entropía de la partición ha de ser seleccionado como punto de corte, y no solo eso, sino que los puntos de corte así escogidos siempre se van a encontrar entre valores que pertenecen a clases diferentes.

Una mejora de este algoritmo es la **discretización basada en la entropía con el criterio de parada MDLP** [73]. El **principio de minimización de la longitud de descripción** (MDLP, *Minimum Description Length Principle*) [19, 187] puede resumirse diciendo que, dado un conjunto de datos, el mejor modelo es aquel más compacto y que requiere la mínima cantidad de información para ser construido. Teniendo en cuenta que tanto modelo como datos pueden codificarse mediante bits, ante un modelo y unos datos, se trata de minimizar la longitud de la codificación. Por tanto, en el caso del algoritmo de discretización que nos ocupa, dependiendo de la homogeneidad de los intervalos resultantes valdrá la pena o no introducir un punto de corte. Si la codificación obtenida introduciendo el punto de corte es más corta que si no se hace, lo introduciremos; en caso contrario, no lo haremos. Podemos minimizar la longitud de descripción total si en cada paso de discretización elegimos aquel punto que maximice la ganancia de información.

B

Análisis estadísticos

A lo largo de la presente tesis se ha mencionado en diversas ocasiones que, en general, no existe ningún algoritmo que alcance un comportamiento superior a todos los demás para todos los conjuntos de datos. Esta afirmación es la que se conoce con el nombre de **teorema del *no free lunch*** [231, 232]. Este teorema parte de la hipótesis de que no se tiene ningún conocimiento parcial del problema, lo cuál, en ocasiones, no se cumple, permitiendo diseñar algoritmos con características específicas que los hagan más eficaces a la hora de resolverlo. Teniendo esta premisa en mente, se hace necesario el uso de test estadísticos para analizar empíricamente los resultados obtenidos por los algoritmos, comparando así su rendimiento. En este punto juega un papel fundamental el análisis empírico de los resultados, haciéndose uso de la estadística inferencial, que permite conocer si la muestra de resultados apoya una determinada hipótesis y si las conclusiones alcanzadas pueden generalizarse más allá del estudio efectuado.

En este capítulo se describen los fundamentos para la comparación de heurísticas mediante técnicas estadísticas, explicando las condiciones que hacen conveniente aplicar unas técnicas u otras. En la literatura se distingue entre dos grupos de test estadísticos: **paramétricos** y **no paramétricos** [205]. Los primeros asumen

que los datos deben cumplir las propiedades de independencia, normalidad y homocedasticidad, y que pertenecen a una determinada distribución. Cuando estas condiciones se cumplen, es preferible aplicar este tipo de test. Sin embargo, resulta muy complicado que estas propiedades se satisfagan cuando se trata de algoritmos bioinspirados, con lo que se hace necesario el uso de test no paramétricos.

Es más, es necesario distinguir entre **comparaciones por pares** y **múltiples** [59]. La primera sirve de marco para la comparación de dos algoritmos, mientras que la última es la que se debe usar cuando se comparan más de dos métodos. Así, en la Sección B.2 nos centramos en explicar las condiciones iniciales que deben satisfacerse para poder aplicar test paramétricos. Por su parte, la Sección B.3 describe los test no paramétricos de comparaciones por pares, y la Sección B.4 describe los test no paramétricos para detectar grupos de diferencias entre los resultados obtenidos por más de dos algoritmos. Finalmente, la Sección B.5 introduce los test no paramétricos que hay que aplicar posteriormente a los de comparación múltiple para identificar las diferencias (*post-hoc*).

B.1. Introducción a la estadística inferencial

En el campo de la estadística inferencial, los **contrastes de hipótesis** se pueden utilizar para inferir conclusiones acerca de una población de muestras (resultados). En la situación de partida de un estudio experimental se plantea una hipótesis nula, H_0, que representa la situación donde las propuestas contempladas en el estudio no suponen novedad o mejora significativa, frente a la hipótesis alternativa, H_1, que implica la presencia de diferencias (en nuestro caso, diferencias significativas en el rendimiento de los algoritmos).

Cuando se aplica un procedimiento estadístico para rechazar una hipótesis, es necesario establecer un **nivel de confianza o significación** del test, denotado por $1-\alpha$, para determinar con qué probabilidad se puede rechazar la hipótesis. Los valores usuales para α son 0,01, 0,05 y 0,10. Dichos valores, expresados en porcentajes, corresponden a niveles de confianza del 99 %, 95 % y 90 %, respectivamente.

En vez de establecer un nivel de confianza a priori, es posible calcular el menor nivel de significación que resulta en el rechazo de H_0. Este nivel es el que se conoce

con el nombre de ***p-value***. Este valor proporciona información acerca de si un test es significativo o no, indicando además cómo de significativo es: cuanto menor sea el *p-value*, mayor evidencia hay del rechazo de H_0 y, lo que es más importante, lo hace sin comprometerse o estar ligado a un nivel de confianza.

B.2. Condiciones iniciales para la aplicación de test paramétricos

La distinción que hace *Sheskin* [205] entre test estadísticos paramétricos y no paramétricos se basa en el tipo de datos envueltos en el estudio. Así, un test paramétrico utiliza datos con valores reales pertenecientes a un intervalo, mientras que uno no paramétrico utiliza datos nominales o que representan un orden en forma de rango. Sin embargo, el hecho de disponer de datos reales no implica que se pueda aplicar un test paramétrico, sino que deben cumplirse además otras condiciones, que en caso de no satisfacerse harían que el análisis estadístico perdiera credibilidad.

Los test paramétricos han sido utilizados en muchos estudios estadísticos de algoritmos bioinspirados. Por ejemplo, la forma habitual de comprobar las diferencias entre dos algoritmos es hacer uso del **test *t* de Student** [95], que comprueba si la diferencia media en su rendimiento sobre los problemas estudiados es significativamente diferente de cero. Para comparar un conjunto de algoritmos, el método estadístico paramétrico utilizado comúnmente ha sido el **test de las medidas repetidas o ANOVA** [78]. La hipótesis nula a contrastar es que todos los algoritmos rinden igual y que las diferencias observadas entre los rendimientos de los algoritmos son debidas al azar. ANOVA divide la variabilidad total del indicador de rendimiento en tres sumandos: variabilidad de los algoritmos, variabilidad entre instancias y variabilidad residual. Si la variabilidad entre algoritmos es significativamente mayor que la variabilidad residual, se rechaza la hipótesis nula y se concluye que existen diferencias entre los algoritmos. Este contraste se efectúa con la distribución F de Fisher-Snedecor y en caso de rechazar la igualdad en el rendimiento de los algoritmos se puede proceder con un test a posteriori que determine dónde se encuentran esas diferencias, entre qué algoritmos. Entre los muchos test

para realizar esto en ANOVA, los dos más corrientes son el **test de Tuckey** para comparar todos los algoritmos entre sí y el **test de Dunnett** para compararlos frente a un algoritmo de control. Ambos métodos usan la estimación de la desviación típica de la diferencia entre los rendimientos de dos algoritmos dividiendo la variabilidad residual por el número de instancias menos uno. Para comparar pares de algoritmos, las correspondientes diferencias en rendimiento se dividen por esta estimación y se comparan con el valor crítico.

Sin embargo, los test paramétricos están basados en suposiciones que en el caso de algoritmos estocásticos de ML se violan con toda seguridad [59, 60, 89]. Es más, los test no paramétricos permiten efectuar comparaciones entre algoritmos de diferentes dominios, es decir, estocásticos y determinísticos, cosa que no pueden hacer los paramétricos. Las condiciones que se han de cumplir para que se puedan aplicar los test paramétricos son las que se listan a continuación:

- **Independencia**. Dos sucesos se consideran independientes cuando la probabilidad de ocurrencia de uno de ellos no se ve afectada por que haya ocurrido el otro. Esta condición suele ser siempre cierta en estudios experimentales que comparan resultados de algoritmos, dado que son ejecuciones independientes de los algoritmos con semillas iniciales generadas aleatoriamente.

- **Normalidad**. Una observación se considera normal si su comportamiento se asemeja a una distribución normal o de Gauss con una media μ y varianza σ. Para comprobar si los datos observados siguen una distribución gaussiana, será necesaria la aplicación de un test de normalidad sobre la muestra. Existen diversas alternativas a la hora de verificar la normalidad de una distribución, entre las que se encuentran las dos siguientes:

  Test de Shapiro-Wilk [203]. Calcula el nivel de simetría y el coeficiente de curtosis (o forma de la curva) para pasar a calcular posteriormente su diferencia con respecto a los de una distribución normal, obteniendo el valor de p a partir de la suma de cuadrados de esas discrepancias:

$$W = \frac{(\Sigma_{i=1}^{n} a_i x_{(i)})^2}{\Sigma_{i=1}^{n}(x_i - \bar{x}^2)} \tag{B.1}$$

donde:

$x_{(i)}$	es la i-ésima menor observación de la muestra,
$\bar{x}$	representa la media, y
las constantes a_i	se obtienen como sigue:

$$(a_1, ..., a_n) = \frac{m^T V^{-1}}{(m^T V^{-1} V^{-1} m)^{1/2}} \tag{B.2}$$

donde:

$m = (m_1, ..., m_n)^T$	son los valores esperados, de menor a mayor, de observaciones de una muestra de una distribución normal idéntica, y
V	es la matriz de covarianza de éstos.

Test de Anderson-Darling [11]. Sirve para comprobar si una muestra viene de una determinada distribución, normalmente la gaussiana. Compara la distribución de probabilidades acumulada empírica, resultado de los datos observados, con la distribución acumulada teórica esperada. El estadístico A^2 se calcula en base a la siguiente fórmula:

$$A^2 = -n - S \tag{B.3}$$

donde:

$$S = \Sigma_{i=1}^{n} \frac{2i-1}{n} [lnF(Y_i) + ln(1 - F(Y_{n+1-i}))] \tag{B.4}$$

donde:

F	es la función de distribución acumulada de la distribución especificada, y
Y_i	son los datos ordenados.

- **Homocedasticidad**. Esta propiedad indica que las varianzas de las muestras deben ser iguales. El **test de Levene** [138] se usa para discernir si k muestras presentan o no esta homogeneidad en las varianzas.

B.3. Test no paramétricos de comparaciones por pares

Los test de comparaciones por pares son el tipo de test estadísticos más sencillos que un investigador puede aplicar en el marco de un estudio experimental. Están destinados a comparar el comportamiento (rendimiento) de dos algoritmos al ejecutarse sobre un mismo conjunto de problemas, y son la alternativa al test t de Student paramétrico. En primer lugar centramos la atención sobre el test de los signos, un método rápido y sencillo, aunque poco potente, para pasar a explicar el test de los rangos con signo de Wilcoxon, mucho más robusto y seguro.

B.3.1. El test de los signos

Una forma sencilla y usual de comparar dos algoritmos es contar el número de veces en las que uno de ellos tiene mejor comportamiento que el otro. Bajo la hipótesis nula de que ambos tienen el mismo rendimiento, este número se aproximaría a la mitad de las veces, esto es, cada uno debería ganar en $n/2$ de n problemas.

El número de veces que cada algoritmo gana obedecería a una distribución binomial con probabilidad de éxito 0,5 y número de pruebas n. Bajo la hipótesis nula, para un número mayor de casos, el número de victorias se distribuye de acuerdo a $n(n/2, \sqrt{n}/2)$, lo cuál permite el uso del estadístico z (1,96 para un nivel de confianza $\alpha = 0{,}05$). Así, si el número de victorias es al menos $n/2 + 1{,}96 \cdot \sqrt{n}/2$, lo que ha dado origen a la regla rápida de $n/2 + \sqrt{n}$, entonces el algoritmo es significativamente mejor con $p < 0{,}05$.

En este test deben tenerse en cuenta los casos de empate, dado que respaldan la hipótesis nula, pero poniéndolos como victorias alternativamente para uno u otro algoritmo. En caso de que hubiera un número impar de empates, uno de ellos debe ignorarse.

B.3.2. Test de los rangos con signo de Wilcoxon

El test de los rangos con signos de Wilcoxon [228] se basa en la ordenación de las diferencias. Para ello se calculan los valores absolutos de las diferencias en el rendimiento de dos algoritmos para cada instancia y se ordenan de mayor a menor, comparando los lugares que ocupan las diferencias a favor de cada algoritmo.

Formalmente, el test de los rangos con signo de Wilcoxon se define como sigue. Sea $rank(d_i)$ la posición que ocupa $|d_i|$ en esta ordenación, denominado rango de d_i. Sean R^+ *y* R^- las sumas respectivas de los rangos de las diferencias positivas y negativas. Los rangos de las diferencias nulas ($d_i = 0$) se reparten entre ambas sumas. Si hay un número impar de ellas, se ignora una. Por tanto:

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \qquad (B.5)$$
$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$$

Sea $T = min(R^+, R^-)$ la menor de las sumas. Si T es menor o igual que el valor de la distribución de Wilcoxon para n grados de libertad, la hipótesis nula de igualdad se rechaza. Esto implica que un algoritmo es significativamente mejor que el otro para la medida de rendimiento estudiada, con el respectivo *p-value* asociado.

B.4. Test no paramétricos de comparaciones múltiples

Los tests de comparaciones por pares no han sido diseñados para obtener conclusiones sobre varias variables. En este tipo de situaciones, las comparaciones por pares son insuficientes, ya que los *p-values* obtenidos en cada comparación no se pueden acumular. El error que se comete si se lleva a cabo la comparación como una familia de hipótesis relacionadas en vez de considerarlas hipótesis individuales hace deseable la utilización de test de comparaciones múltiples.

En el caso habitual en que se presenta un nuevo algoritmo bioinspirado para alguna tarea de DM y el investigador desea comparar su rendimiento frente a otros enfoques similares, es necesario utilizar test no paramétricos de comparaciones múltiples [59]. En estos casos, el interés radica en detectar grupos de diferencias entre los resultados obtenidos por los algoritmos envueltos en el estudio experimental. En estadística, estos grupos se denominan **bloques** y se corresponden normalmente con los problemas del estudio. Por ejemplo, suponiendo un estudio experimental múltiple para clasificación con muchos conjuntos de datos, cada bloque corresponde a los resultados obtenidos por los algoritmos sobre un determinado conjunto de datos. Así, un bloque estará compuesto de tres o más resultados, cada uno correspondiente al rendimiento de un algoritmo sobre el conjunto de datos [89].

En un estudio experimental con varios algoritmos, el **número de problemas o conjuntos de datos para hacer un análisis estadístico adecuado** debe ser superior al al doble del número de algoritmos, como mínimo. Con un número inferior de problemas a $n < 2 \cdot k$, donde k es el número de algoritmos, existe una alta probabilidad de que no se pueda rechazar ninguna hipótesis. Por otro lado, no existe un máximo de conjuntos de datos que puedan ser utilizados en el estudio, aunque una proporción muy elevada entre el número de problemas y de algoritmos puede llevar a resultados ineficaces de acuerdo con el teorema central del límite [205]. Como norma, se puede considerar adecuado utilizar hasta un tope de $n = 8 \cdot k$ conjuntos de datos [60].

En estadística no paramétrica, el procedimiento más conocido para probar si existen diferencias significativas entre más de dos muestras es el test de Friedman. A continuación explicamos este test y su extensión, el test de Iman&Davenport.

B.4.1. Test de Friedman

El test de Friedman [85] es un test no paramétrico correspondiente al test F en un ANOVA de bloques. El primer paso para calcular el estadístico consiste en convertir los resultados originales a rangos. Para ello se ordenan los k algoritmos separadamente para cada problema según el rendimiento alcanzado, asignando un rango de 1 al mejor algoritmo, 2 al segundo mejor, y así sucesivamente. En caso de que varios algoritmos obtengan el mismo rendimiento para esa instancia se

asignarían rangos intermedios. Así, sea r_j^i el rango del j-ésimo algoritmo sobre el i-ésima problema. El test de Friedman compara los rangos medios de los algoritmos calculados según:

$$R_j = \frac{1}{n}\sum_{i=1}^{n} r_j^i \tag{B.6}$$

Bajo la hipótesis nula, el rendimiento de todos los algoritmos es el mismo y los rangos R_j deben ser similares. El estadístico

$$\chi_F^2 = \frac{12n}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{B.7}$$

propuesto por Friedman, se distribuye de acuerdo a una distribución χ^2 con $k-1$ grados de libertad, cuando n y k son suficientemente grandes (como regla experimental práctica, $n > 10$ y $k > 5$). Para un número menor de algoritmos y casos se han calculado los valores críticos exactos en [205, 242].

Con el test de Friedman es posible establecer una ordenación clara entre los algoritmos, calculando incluso el **rango medio** obtenido por cada algoritmo sobre todos los conjuntos de datos, lo que da una perspectiva interesante de la comparativa. En caso de rechazarse la hipótesis nula, se puede concluir que al menos existe una diferencia significativa entre los rendimientos de los algoritmos, pero no nos indica qué algoritmos ofrecen diferencias significativamente relevantes. Para ello es necesario aplicar un test a posteriori de los que se indican en la Sección B.5.

B.4.2. Test de Iman&Davenport

Iman y Davenport [113] comprobaron que el estadístico χ_F^2 de Friedman presenta un **comportamiento conservador** no deseado, y propusieron un estadístico mejor basado en la distribución F de Fisher-Snedecor, derivado del de Friedman. Por este motivo, en los estudios experimentales abordados en esta tesis hemos utilizado este test en lugar del de Friedman. Con $k-1$ y $(k-1)(n-1)$ grados de libertad, el valor del estadístico se calcula como:

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1)-\chi_F^2} \tag{B.8}$$

En [205] se pueden encontrar las tablas estadísticas para los valores críticos. Al igual que para el test de Friedman, en caso de rechazar la hipótesis nula, se puede proceder con un test a posteriori para revelar dónde se producen las diferencias significativas.

B.5. Test no paramétricos a posteriori

El mayor inconveniente de los test de Friedman y de Iman&Davenport es que únicamente detectan si existen diferencias significativas entre los algoritmos del estudio, pero no las identifican. Cuando el objetivo de la aplicación del test múltiple es efectuar una comparación entre un algoritmo de control (habitualmente el algoritmo propuesto en el trabajo) y un conjunto de algoritmos, si se rechaza la hipótesis nula de equivalencia de los rendimientos de los algoritmos que postulan el test de Friedman y el de Iman&Davenport, puede definirse una familia de hipótesis, todas asociadas al método de control. Entonces es posible aplicar un test no paramétrico a posteriori (*post-hoc*) que lleve a la obtención de un *p-value* que determine el grado de rechazo de cada hipótesis individual.

Puede ocurrir, aunque es poco probable, que los test no paramétricos de comparaciones múltiples encuentren diferencias significativas pero los test a posteriori no lleguen a detectarlas, no siendo posible identificarlas, debido a la menor potencia de estos últimos. En estos casos sólo se puede concluir que el comportamiento de algunos algoritmos es diferente [59].

Se pueden distinguir tres grupos de test no paramétricos a posteriori, según si hacen todas las comparaciones al mismo tiempo en un único paso (***one-step***), si siguen un procedimiento secuencial hacia arriba (***step-up***), o bien si son secuenciales hacia abajo (***step-down***). A continuación se explica un test no paramétrico *post-hoc* por cada uno de estos grupos.

B.5.1. Test de Bonferroni-Dunn

El test de Bonferroni-Dunn [67] divide el valor de α por el número de comparaciones realizadas, igual a $k-1$ en nuestro caso. La calidad de dos algoritmos es significativamente diferente si la diferencia entre los rankings de ambos excede el valor proporcionado por la diferencia crítica:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}} \tag{B.9}$$

donde:

q_α es el valor crítico de Q‘ para una múltiple comparación no paramétrica con un algoritmo de control (Tabla B.16 en [242])

La mayor ventaja que supone la aplicación del test de Bonferroni-Dunn radica en que realiza todas los contrastes de vez (*one-step*), utilizando la misma diferencia crítica, lo que permite visualizar gráficamente los resultados.

B.5.2. Test de Holm

El test de Holm [109] contrasta secuencialmente las hipótesis ordenadas según su significancia de mayor a menor (*step-down*). Denotaremos los *p-values* ordenados por $p_1, p_2, \ldots, p_{k-1}$ de forma que $p_1 \leq p_2 \ldots \leq p_{k-1}$, siendo $H_1, H_2, \ldots, H_{k-1}$ las correspondientes hipótesis. El test de Holm compara cada p_i con $\alpha/(k-i)$ comenzando desde el $p-value$ más significativo. Si p_1 está por debajo de $\alpha/(k-1)$, se rechaza la correspondiente hipótesis nula y se procede a comparar p_2 con $\alpha/(k-2)$. Si la segunda hipótesis se rechaza, se procede a comprobar la tercera, y así sucesivamente. En el momento en que una hipótesis nula no se pueda rechazar, se detiene el proceso y no se rechaza ninguna de las restantes, manteniéndolas como aceptadas. El estadístico para comparar el valor i-ésimo con el j-ésimo es:

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6n}} \tag{B.10}$$

El valor z se usa para localizar la probabilidad correspondiente a partir de la tabla de la distribución normal, la cuál se compara con el correspondiente valor de α.

B.5.3. Test de Hochberg

El test de Hochberg [106] ajusta el valor de α de menor a mayor (*step-up*). Al contrario que el de Holm, este test compara el *p-value* más alto con α, el siguiente más alto con $\alpha/2$, el siguiente con $\alpha/3$, y así sucesivamente hasta encontrar una hipótesis que pueda rechazar. Todas las hipótesis nulas con *p-values* más pequeños son asimismo rechazadas.

Bibliografía

[1] Abbass, H. A., Hoai, X., and Mckay, R. I. AntTAG: A new method to compose computer programs using colonies of ants. In *IEEE Congress on Evolutionary Computation (IEEE CEC)* (2002), pp. 1654–1659.

[2] Abdelbar, A. Stubborn ants. In *IEEE Swarm Intelligence Symposium (IEEE SIS)* (2008), pp. 1–5.

[3] Abdi, H., and Williams, L. J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics 2*, 4 (2010), 433–459.

[4] Abe, N., Zadrozny, B., and Langford, J. An iterative method for multi-class cost-sensitive learning. In *ACM SIGKDD* (2004), pp. 3–11.

[5] Agrawal, R., Imieliński, T., and Swami, A. Mining association rules between sets of items in large databases. *ACM SIGMOD Record 22*, 2 (1993), 207–216.

[6] Alatas, B., and Akin, E. Multi-objective rule mining using a chaotic particle swarm optimization algorithm. *Knowledge-Based Systems 22*, 6 (2009), 455–460.

[7] Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., and García, S. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing 17*, 2-3 (2011), 255–287.

[8] Alcalá-Fdez, J., Sánchez, L., García, S., del Jesus, M., Ventura, S., Garrell, J., Otero, J., Romero, C., Bacardit, J., Rivas, V., Fernández, J., and Herrera, F. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications 13* (2009), 307–318.

[9] Alhammady, H., and Ramamohanarao, K. Mining emerging patterns and classification in data streams. In *IEEE/WIC/ACM International Conference on Web Intelligence* (2005), pp. 272–275.

[10] An, A., and Cercone, N. Discretization of continuous attributes for learning classification rules. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (1999), pp. 509–514.

[11] Anderson, T., and Darling, D. Asymptotic theory of certain 'goodness-of-fit'criteria based on stochastic processes. *Annals of Mathematical Statistics 23* (1952), 193–212.

[12] Angus, D., and Woodward, C. Multiple objective ant colony optimisation. *Swarm Intelligence 3*, 1 (March 2009), 69–85.

[13] Augusto, D. A., Barbosa, H. J., and Ebecken, N. F. Coevolution of data samples and classifiers integrated with grammatically-based genetic programming for data classification. In *Genetic and Evolutionary Computation Conference (GECCO)* (New York, NY, USA, 2008), ACM, pp. 1171–1178.

[14] Augusto, D. A., Barbosa, H. J. C., and Ebecken, N. F. F. Coevolutionary multi-population genetic programming for data classification. In *Genetic and Evolutionary Computation Conference (GECCO)* (New York, NY, USA, 2010), ACM, pp. 933–940.

[15] Ávila-Jiménez, J., Gibaja, E., Zafra, A., and Ventura, S. A gene expression programming algorithm for multi-label classification. *Multiple-Valued Logic and Soft Computing 17*, 2-3 (2011), 183–206.

[16] Bahnert, A., Norton, K., and Lock, P. Association between post-game recovery protocols, physical and perceived recovery, and performance in elite afl players. *Journal of Science and Medicine in Sport (In press) 16*, 2 (Jun 2012), 151–156.

[17] Baig, A., and Shahzad, W. A correlation-based ant miner for classification rule discovery. *Neural Computing & Applications 21* (2012), 219–235.

[18] Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, San Francisco, CA, USA, January 1998.

[19] Barron, A., Rissanen, J., and Yu, B. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on 44*, 6 (1998), 2743–2760.

[20] Batista, G. E. A. P. A., and Monard, M. C. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence 17*, 5-6 (2003), 519–533.

[21] Bay, S. D., and Pazzani, M. J. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery 5*, 3 (2001), 213–246.

[22] Bellman, R. *Adaptive Control Processes*. Princeton University Press, 1961.

[23] Berlanga, F. *Aprendizaje de sistemas basados en reglas difusas compactos y precisos con programació genética*. PhD thesis, Universidad de Granada, 2010.

[24] Berlanga, F. J., Rivera, A. J., del Jesus, M. J., and Herrera, F. GP-COACH: Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems. *Information Sciences 180*, 8 (Apr 2010), 1183–1200.

[25] Blum, C. Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews 2*, 4 (2005), 353–373.

[26] Blum, C., and Merkle, D. *Swarm Intelligence: Introduction and Applications*, 1 ed. Springer, 2008.

[27] Bojarczuk, C. C., Lopes, H. S., Freitas, A. A., and Michalkiewicz, E. L. A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. *Artificial Intelligence in Medicine 30* (2004), 27–48.

[28] Bonabeau, E., Dorigo, M., and Theraulaz, G. Inspiration for optimization from social insect behaviour. *Nature 406*, 6791 (Jul 2000), 39–42.

[29] Bonabeu, E., Eric, T., and Dorigo, M. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University, 1999.

[30] Boryczka, M. Eliminating introns in ant colony programming. *Fundamenta Informaticae 68*, 1-2 (2005), 1–19.

[31] Boryczka, M. Ant colony programming with the candidate list. In *KES International conference on Agent and Multi-agent Systems: Technologies and Applications (KES-AMSTA)* (2008), vol. 4953 of *LNAI*, pp. 302–311.

[32] Boryczka, M., and Czech, Z. J. Solving approximation problems by ant colony programming. In *Genetic and Evolutionary Computation Conference (GECCO)* (2002), pp. 39–46.

[33] Boryczka, M., Czech, Z. J., and Wieczorek, W. Ant colony programming for approximation problems. In *Genetic and Evolutionary Computation Conference (GECCO)* (2003), pp. 142–143.

[34] Boutell, M. R., Luo, J., Shen, X., and Brown, C. M. Learning multi-label scene classification. *Pattern Recognition 37*, 9 (2004), 1757–1771.

[35] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth, 1984.

[36] Brin, S., Motwani, R., Ullman, J., and Tsur, S. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD International Conference on Management of Data* (1997), vol. 26, pp. 255–264.

[37] Bullnheimer, B., Hartl, R. F., and StrauÃü, C. A new rank based version of the ant system - a computational study. *Central European Journal for Operations Research and Economics 7* (1997), 25–38.

[38] Carmona, C., Ramírez-Gallego, S., Torres, F., Bernal, E., Del Jesus, M., and García, S. Web usage mining to improve the design

of an e-commerce website: Orolivesur.com. *Expert Systems with Applications 39*, 12 (2012), 11243–11249.

[39] Ceglar, A., and Roddick, J. F. Association mining. *ACM Computing Surveys 38*, 2 (July 2006), 1–42.

[40] Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. Locally adaptive dimensionality reduction for indexing large time series databases. *Database Systems, ACM Transactions on 27*, 2 (Jun 2002), 188–228.

[41] Chan, A., and Freitas, A. A new classification-rule pruning procedure for an ant colony algorithm. In *Artificial Evolution*, vol. 3871 of *LNCS*. Springer Berlin / Heidelberg, 2006, pp. 25–36.

[42] Chan, A., and Freitas, A. A. A new ant colony algorithm for multi-label classification with applications in bioinformatics. In *Genetic and Evolutionary Computation Conference (GECCO)* (2006), pp. 27–34.

[43] Chandra Mohan, B., and Baskaran, R. A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications 39*, 4 (2012), 4618–4627.

[44] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. SMOTE: Synthetic minority over-sampling techniques. *Journal of Artificial Intelligence Research 16* (2002), 321–357.

[45] Chen, C., Chen, Y., and He, J. Neural network ensemble based ant colony classification rule mining. In *International Conference on Innovative Computing, Information and Control (ICICIC)* (2006), vol. 3, pp. 427–430.

[46] Chen, G., Liu, H., Yu, L., Wei, Q., and Zhang, X. A new approach to classification based on association rule mining. *Decision Support Systems 42*, 2 (2006), 674–689.

[47] Chen, Y., Yang, B., and Dong, J. Evolving flexible neural networks using ant programming and PSO algorithms. In *Advances in Neural Networks ISSN* (2004), vol. 3173 of *LNCS*.

[48] Cios, K., Pedrycz, W., Swiniarski, R., and Kurgan, L. *Data Mining: A Knowledge Discovery Approach*. Springer, 2010.

[49] Clark, P., and Boswell, R. Rule induction with CN2: Some recent improvements. In *European Working Session on Machine Learning (EWSL)* (1991), pp. 151–163.

[50] Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20* (1960), 37–46.

[51] Cohen, W. Fast Effective Rule Induction. In *International Conference on Machine Learning (ICML)* (1995), pp. 115–123.

[52] Cordon, O., Herrera, F., and Stützle, T. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing 9* (2002), 141–175.

[53] Dasgupta, S., Das, S., Abraham, A., and Biswas, A. Adaptive computational chemotaxis in bacterial foraging optimization: An analysis. *Evolutionary Computation, IEEE Transactions on 13*, 4 (2009), 919–941.

[54] de Falco, I., della Cioppa, A., and Tarantino, E. Discovering interesting classification rules with genetic programming. *Applied Soft Computing 1*, 4 (May 2002), 257–269.

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

[56] Dehuri, S., Patnaik, S., Ghosh, A., and Mall, R. Application of elitist multi-objective genetic algorithm for classification rule generation. *Applied Soft Computing 8* (January 2008), 477–487.

[57] del Jesus, M. J., Gómez, J. A., González, P., and Puerta, J. M. On the discovery of association rules by means of evolutionary algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1*, 5 (2011), 397–415.

[58] del Jesus, M. J., González, P., Herrera, F., and Mesonero, M. Evolutionary fuzzy rule induction process for subgroup discovery: A case study in marketing. *Fuzzy Systems, IEEE Transactions on 15*, 4 (Aug. 2007), 578–592.

[59] Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research 7* (2006), 1–30.

[60] Derrac, J., García, S., Molina, D., and Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation 1*, 1 (2011), 3–18.

[61] Derrac, J., Triguero, I., Garcia, S., and Herrera, F. Integrating instance selection, instance weighting, and feature weighting for nearest neighbor classifiers by coevolutionary algorithms. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 42*, 5 (2012), 1383–1397.

[62] Dorigo, M., Birattari, M., and Stützle, T. Ant colony optimization artificial ants as a computational intelligence technique. *Computational Intelligence Magazine, IEEE 1*, 4 (2006), 28–39.

[63] Dorigo, M., and Gambardella, L. Ant colony system: A cooperative learning approach to the traveling salesman problems. *IEEE Transactions on Evolutionary Computation 1*, 1 (1997), 53–66.

[64] Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: Optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on 26* (1996), 29–41.

[65] Dorigo, M., and Stützle, T. *The Ant Colony Optimization metaheuristic: Algorithms, Applications and Advances.* International Series in Operations Research and Management Science. Kluwer Academic Publishers, 2002.

[66] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern classification.* Wiley, New York, 2000.

[67] Dunn, O. Multiple comparisons among means. *Journal of the American Statistical Association 56* (1961), 52–64.

[68] Efron, B. Estimating the error rate of a prediction rule: Improvement on Cross-Validation. *Journal of the American Statistical Association 78*, 382 (1983), 316–331.

[69] Eiben, A. E., and Smith, J. E. *Introduction to Evolutionary Computing*, 2nd ed. Natural Computing Series. Springer, 2007.

[70] Espejo, P., Ventura, S., and Herrera, F. A survey on the application of genetic programming to classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 40*, 2 (march 2010), 121–144.

[71] Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters 27*, 8 (2006), 861 – 874.

[72] Fawcett, T., and Provost, F. Adaptive fraud detection. *Data Mining and Knowledge Discovery 1* (1997), 291–316.

[73] Fayyad, U. M., and Irani, K. B. Multi-interval discretization of continuous-valued attributes for classification learning. In *International Joint Conference on Uncertainly in Artificial Intelligence (IJCAI)* (1993), pp. 1022–1029.

[74] Febrer-Hernández, J., and Hernández-Palancar, J. Sequential pattern mining algorithms review. *Intelligent Data Analysis 16*, 3 (01 2012), 451–466.

[75] Ferber, J. *Multi-agent systems. An introduction to distributed artificial intelligence*. Addison-Wesley-Longman, 1999.

[76] Fernández, A., García, S., and Herrera, F. Addressing the classification with imbalanced data: Open problems and new challenges on class distribution. In *HAIS*, vol. 6678 of *LNCS*. Springer, 2011, pp. 1–10.

[77] Fernández, A., López, V., Galar, M., del Jesus, M. J., and Herrera, F. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, 0 (2013).

[78] Fisher, R. *Statistical methods and scientific inference*. Hafner Publishing Co., New York, 1959.

[79] Floreano, D., and Mattiussi, C. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

[80] Fogel, D. An evolutionary approach to the travelling salesman problem. *Biological Cybernetics 60*, 2 (1988), 139–144.

[81] Forbes, N. *Imitation of life. How biology is inspiring computing*. The MIT Press, 2004.

[82] Frank, A., and Asuncion, A. UCI machine learning repository, 2010.

[83] Frawley, W. J., Piatetsky-Shapiro, G., and Matheus, C. J. Knowledge discovery in databases: an overview. *AI Magazine 13*, 3 (Sept. 1992), 57–70.

[84] Freitas, A. A. A review of evolutionary algorithms for data mining. In *Soft Computing for Knowledge Discovery and Data Mining*, O. Maimon and L. Rokach, Eds. Springer, 2008, pp. 79–111.

[85] Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics 11*, 1 (1940), 86–92.

[86] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition 44*, 8 (2011), 1761–1776.

[87] Galar, M., Fernández, A., Barrenechea, E., Bustince, H., and Herrera, F. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on PP*, 99 (2011), 1–22.

[88] Galea, M., and Shen, Q. Simultaneous ant colony optimization algorithms for learning linguistic fuzzy rules. In *Swarm Intelligence in Data*

Mining, vol. 34 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg, 2006, pp. 75–99.

[89] García, S., Fernández, A., Luengo, J., and Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Information Sciences 180*, 10 (2010), 2044–2064.

[90] García, V., Sánchez, J. S., and Mollineda, R. A. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems 25*, 1 (2012), 13–21.

[91] García-Martínez, C., Cordón, O., and Herrera, F. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research 180*, 1 (2007), 116 – 148.

[92] Geng, L., and Hamilton, H. J. Interestingness measures for data mining: A survey. *ACM Computing Surveys 38*, 3 (Sep 2006), 1–32.

[93] Geyer-Schulz, A. *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning*, vol. 3 of *Studies in Fuzziness*. Physica-Verlag, Heidelberg, 1995.

[94] Goss, S., Aron, S., Deneubourg, J., and Pasteels, J. Self-organized shortcuts in the argentine ant. *Naturwissenschaften 76*, 12 (December 1989), 579–581.

[95] Gosset, W. The probable error of a mean. *Biometrika 6*, 1 (1908), 1–25.

[96] Green, J., Whalley, J., and Johnson, C. Automatic programming with ant colony optimization. In *UK Workshop on Computational Intelligence* (2004), pp. 70–77.

[97] Guyon, I., and Elisseeff, A. An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (Mar. 2003), 1157–1182.

[98] Han, J., and Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kauffman, 2006.

[99] Han, J., Pei, J., Yin, Y., and Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery 8*, 1 (2004), 53–87.

[100] Hara, A., Watanabe, M., and Takahama, T. Cartesian ant programming. In *IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC)* (2011), pp. 3161–3166.

[101] Haykin, S. *Neural Networks and Learning Machines*, 3rd ed. Pearson, 2009.

[102] He, H., and García, E. A. Learning from Imbalanced Data. *Knowledge and Data Engineering, IEEE Transactions on 21*, 9 (Sept. 2009), 1263–1284.

[103] Herrera, F., Carmona, C., GonzÃ¡lez, P., and del Jesus, M. An overview on subgroup discovery: foundations and applications. *Knowledge and Information Systems 29* (2011), 495–525.

[104] Hoai, N., and McKay, R. A framework for tree adjunct grammar guided genetic programming. In *Post-Graduate ADFA Conference on Computer Science (PACCS)* (2001), pp. 93–99.

[105] Hoai, N. X., Mckay, R. I., and Abbass, H. A. Tree adjoining grammars, language bias, and genetic programming. In *European Conference on Genetic Programming (EuroGP)* (2003), vol. 2610 of *LNCS*, pp. 340–349.

[106] Hochberg, Y. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika 75* (1988), 800–803.

[107] Holden, N., and Freitas, A. A. A hybrid PSO/ACO algorithm for discovering classification rules in data mining. *Journal of Artificial Evolution and Applications 2008* (January 2008), 2:1–2:11.

[108] Holland, J. Genetic algorithms. *Scientific American 267*, 1 (1992), 66–72.

[109] Holm, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics 6*, 2 (1979), 65–70.

[110] Huang, H.-J., and Hsu, C.-N. Bayesian classification for data from the same unknown class. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 32*, 2 (2002), 137–145.

[111] Huang, T.-M., Kecman, V., and Kopriva, I. Support vector machines in classification and regression - an introduction. In *Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semi-supervised, and Unsupervised Learning*, Studies in Computational Intelligence. Springer, 2006.

[112] IBM Zikopoulos, P., Eaton, C., and Zikopoulos, P. *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Companies, Incorporated, 2011.

[113] Iman, R., and Davenport, J. Approximations of the critical region of the Friedman statistic. In *Communications in Statistics* (1980), pp. 571–595.

[114] Ishibuchi, H., and Nojima, Y. Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning 44*, 1 (2007), 4 – 31.

[115] Jain, A. K., Murty, M. N., and Flynn, P. J. Data clustering: a review. *ACM Computing Surveys 31*, 3 (Sept. 1999), 264–323.

[116] Jin, P., Zhu, Y., Hu, K., and Li, S. Classification rule mining based on ant colony optimization algorithm. In *Intelligent Control and Automation*, vol. 344 of *LNCIS*. Springer Berlin / Heidelberg, 2006, pp. 654–663.

[117] Jin, R., Breitbart, Y., and Muoh, C. Data discretization unification. *Knowledge and Information Systems 19*, 1 (2009), 1–29.

[118] Karaboga, D., and Akay, B. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review 31* (2009), 61–85.

[119] Kavšek, B., Lavrač, N., and Jovanoski, V. APRIORI-SD: Adapting association rule learning to subgroup discovery. In *Advances in Intelligent Data Analysis V*, M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, Eds., vol. 2810 of *LNCS*. Springer Berlin Heidelberg, 2003, pp. 230–241.

[120] Keber, C., and Schuster, M. G. Option valuation with generalized ant programming. In *Genetic and Evolutionary Computation Conference (GECCO)* (2002), pp. 74–81.

[121] Kicinger, R., Arciszewski, T., and Jong, K. D. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures 83* (2005), 1943–1978.

[122] Kim, W., Choi, B.-J., Hong, E.-K., Kim, S.-K., and Lee, D. A taxonomy of dirty data. *Data Mining and Knowledge Discovery 7*, 1 (2003), 81–99.

[123] Knowles, J. D., Thiele, L., and Zitzler, E. A tutorial on the performance assessment of stochastive multiobjective optimizers. Tech. rep., Computer Engineering and Networks Laboratory, ETH Zurich, 2006.

[124] Kohavi, R., and Sahami, M. Error-based and entropy-based discretization of continuous features. In *International Conference on Knowledge Discovery and Data Mining* (1996), pp. 114–119.

[125] Kohonen, T. *Self-organizing maps*, 3rd ed., vol. 30 of *Springer Series in Information Sciences*. Springer, 2001.

[126] Kolo, B. *Binary and Multiclass Classification*. Weatherford Press, 2011.

[127] Kotsiantis, S. B., Zaharakis, I. D., and Pintelas, P. E. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Reviews 26* (2006), 159–190.

[128] Koza, J. R. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, Cambridge, MA, 1992.

[129] Kumaresan, N. Optimal control for stochastic linear quadratic singular periodic neuro Takagi-Sugeno (T-S) fuzzy system with singular cost using ant colony programming. *Applied Mathematical Modelling 35*, 8 (2011), 3797 – 3808.

[130] Kuo, R. J., Chao, C. M., and Chiu, Y. T. Application of particle swarm optimization to association rule mining. *Applied Soft Computing 11*, 1 (2011), 326–336.

[131] Kuo, R. J., Lin, S. Y., and Shih, C. W. Mining association rules through integration of clustering analysis and ant colony system for health insurance

database in taiwan. *Expert Systems with Applications. 33*, 3 (Oct 2007), 794–808.

[132] Kuo, R. J., and Shih, C. W. Association rule mining through the ant colony system for national health insurance research database in taiwan. *Computers & Mathematics with Applications 54*, 11-12 (Dec 2007), 1303–1318.

[133] Kurgan, L., and Cios, K. CAIM discretization algorithm. *Knowledge and Data Engineering, IEEE Transactions on 16*, 2 (2004), 145–153.

[134] Landgrebe, T., Paclik, P., Duin, R., and Bradley, A. Precision-recall operating characteristic (P-ROC) curves in imprecise environments. In *International Conference on Pattern Recognition (ICPR)* (2006), vol. 4, pp. 123–127.

[135] Lavrač, N., Kavšek, B., Flach, P., and Todorovski, L. Subgroup discovery with CN2-SD. *Journal of Machine Learning Research 5* (2004), 153–188.

[136] Lemczyk, M. *Pareto-coevolutionary Genetic Programming Classifier*. PhD thesis, Dalhousie University, Halifax, Nova Scotia, 2006.

[137] Lenca, P., Meyer, P., Vaillant, B., and Lallich, S. On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. *European Journal of Operational Research 184*, 2 (2008), 610–626.

[138] Levene, H. In *Contributions to probability and Statistics: Essays in Honor of Harold Hotelling*, I. Olkin et al., Ed. Stanford University Press, 1960, pp. 278–292.

[139] Liu, B., Abbass, H. A., and McKay, B. Density-based heuristic for rule discovery with ant-miner. In *Australasia-Japan Joint Workshop on Intelligent Evolutionary Systems* (2002), pp. 180–184.

[140] Liu, B., Abbass, H. A., and McKay, B. Classification rule discovery with ant colony optimization. In *IEEE/WIC International Conference on Intelligent Agent Technology (IEEE IAT)* (2003), pp. 83–88.

[141] Liu, H., Hussain, F., Tan, C. L., and Dash, M. Discretization: An enabling technique. *Data Min. Knowl. Discov. 6*, 4 (Oct. 2002), 393–423.

[142] Liu, W., Chawla, S., Cieslak, D. A., and Chawla, N. V. A robust decision tree algorithm for imbalanced data sets. In *SDM* (2010), pp. 766–777.

[143] Luna, J. M., Romero, J. R., and Ventura, S. G3PARM: A grammar guided genetic programming algorithm for mining association rules. In *IEEE Congress on Evolutionary Computation (IEEE CEC)* (2010), pp. 2586–2593.

[144] Luna, J. M., Romero, J. R., and Ventura, S. Design and behaviour study of a grammar guided genetic programming algorithm for mining association rules. *Knowledge and Information Systems 32*, 1 (2011), 53–76.

[145] Luna, J. M., Romero, J. R., and Ventura, S. Mining and representing rare association rules through the use of genetic programming. In *World Congress on Nature & Biologically Inspired Computing (NaBIC)* (2011), pp. 86–91.

[146] Martens, D., Baesens, B., and Fawcett, T. Editorial survey: swarm intelligence for data mining. *Machine Learning 82* (January 2011), 1–42.

[147] Martens, D., De Backer, M., Vanthienen, J., Snoeck, M., and Baesens, B. Classification with ant colony optimization. *Evolutionary Computation, IEEE Transactions on 11* (2007), 651–665.

[148] Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F., and Riquelme, J. C. Mining quantitative association rules based on evolutionary computation and its application to atmospheric pollution. *Integrated Computer-Aided Engineering 17*, 3 (Aug 2010), 227–242.

[149] Michelakos, I., Mallios, N., Papageorgiou, E., and Vassilakopoulos, M. Ant colony optimization and data mining. *Studies in Computational Intelligence 352* (2011), 31–60.

[150] Miller, J. F. Cartesian genetic programming. In *Cartesian Genetic Programming*, J. F. Miller, Ed., Natural Computing Series. Springer Berlin Heidelberg, 2011, pp. 17–34.

[151] Millonas, M. M. Swarms, phase transitions and collective intelligence. In *Artificial Life III*, C. Langton, Ed. Addison-Wesley, 1994, pp. 417–445.

[152] Montana, D. J. Strongly typed genetic programming. *Evol. Comput. 3*, 2 (Jun 1995), 199–230.

[153] Moreno-Torres, J., Raeder, T., Alaiz-Rodríguez, R., Chawla, N., and Herrera, F. A unifying view on dataset shift in classification. *Pattern Recognition 45*, 1 (2012), 521–530.

[154] Mullen, R. J., Monekosso, D., Barman, S., and Remagnino, P. A review of ant algorithms. *Expert Systems with Applications 36* (2009), 9608–9617.

[155] Nalini, C., and Balasubramanie, P. Discovering unordered rule sets for mixed variables using an ant-miner algorithm. *Data Science Journal 7* (May 2008), 76–87.

[156] Nascimento Silla Jr., C., and Freitas, A. A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery 22*, 1-2 (2011), 31–72.

[157] Ning, P., Cui, Y., Reeves, D. S., and Xu, D. Techniques and tools for analyzing intrusion alerts. *Information and System Security, ACM Transactions on 7*, 2 (May 2004), 274–318.

[158] Novak, P. K., Webb, G. I., and Wrobel, S. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research 10* (2009), 377–403.

[159] Olmo, J. L., Cano, A., Romero, J. R., and Ventura, S. Binary and multiclass imbalanced classification using multi-objective ant programming. In *International Conference on Intelligent Systems Design and Applications (ISDA 2012)* (2012), pp. 70–76.

[160] Olmo, J. L., Luna, J. M., Romero, J. R., and Ventura, S. *Trends in Practical Applications of Agents and Multiagent Systems*. LNAI. Springer Berlin / Heidelberg, Apr 2010, ch. An Automatic Programming ACO-Based Algorithm for Classification Rule Mining, pp. 649–656.

[161] Olmo, J. L., Luna, J. M., Romero, J. R., and Ventura, S. Association rule mining using a multi-objective grammar-based ant programming algorithms. In *International Conference on Intelligent Systems Design and Applications (ISDA 2011)* (Nov 2011), pp. 971–977.

[162] Olmo, J. L., Luna, J. M., Romero, J. R., and Ventura, S. Mining association rules with single and multi-objective grammar guided ant programming. *Integrated Computer-Aided Engineering* (2013), 1–17.

[163] Olmo, J. L., Romero, J. R., and Ventura, S. A grammar based ant programming algorithm for mining classification rules. In *IEEE Congress on Evolutionary Computation (IEEE CEC)* (Jul 2010), pp. 225–232.

[164] Olmo, J. L., Romero, J. R., and Ventura, S. Using ant programming guided by grammar for building rule-based classifiers. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 41*, 6 (Dec 2011), 1585–1599.

[165] Olmo, J. L., Romero, J. R., and Ventura, S. Classification rule mining using ant programming guided by grammar with multiple Pareto fronts. *Soft Computing - A Fusion of Foundations, Methodologies and Applications (In press) 0* (Jul 2012), 1–21.

[166] Olmo, J. L., Romero, J. R., and Ventura, S. Multi-objective ant programming for mining classification rules. In *European Conference on Genetic Programming (EuroGP)* (Apr 2012), vol. 7244 of *LNCS*, pp. 146–157.

[167] OÑeill, M., Cleary, R., and Nikolov, N. Solving knapsack problems with attribute grammars. In *Genetic and Evolutionary Computation Conference (GECCO)* (2004), R. P. et al., Ed., pp. 26–30.

[168] OÑeill, M., and Ryan, C. Grammatical evolution. *Evolutionary Computation, IEEE Transactions on 5*, 4 (2001), 349–358.

[169] Ordoñez, C., Ezquerra, N., and Santana, C. Constraining and summarizing association rules in medical data. *Knowledge and Information Systems 9*, 3 (2006), 259–283.

[170] Otero, F., Freitas, A., and Johnson, C. A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing 2* (2010), 165–181.

[171] Otero, F., Freitas, A., and Johnson, C. A new sequential covering strategy for inducing classification rules with ant colony algorithms. *Evolutionary Computation, IEEE Transactions on 17*, 1 (2013), 64–76.

[172] Otero, F., Freitas, A. A., and Johnson, C. cAnt-Miner: An ant colony classification algorithm to cope with continuous attributes. In *International conference on Swarm Intelligence (ANTS)* (2008), vol. 5217 of *LNCS*, pp. 48–59.

[173] Otero, F. E., Freitas, A. A., and Johnson, C. G. A hierarchical classification ant colony algorithm for predicting gene ontology terms. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (EvoBIO)* (2009), pp. 68–79.

[174] Otero, F. E. B., Freitas, A. A., and Johnson, C. G. Handling continuous attributes in ant colony classification algorithms. In *IEEE symposium on Computational Intelligence and Data Mining (IEEE CIDM)* (2009), pp. 225–231.

[175] Ou, G., and Murphey, Y. L. Multi-class pattern classification using neural networks. *Pattern Recognition 40* (Jan. 2007), 4–18.

[176] Palacios, A. M., Gacto, M. J., and Alcalá-Fdez, J. Mining fuzzy association rules from low-quality data. *Soft Computing - A Fusion of Foundations, Methodologies and Applications 16*, 5 (May 2012), 883–901.

[177] Pappa, G. L., and Freitas, A. A. Evolving rule induction algorithms with multi-objective grammar-based genetic programming. *Knowledge and Information Systems 19*, 3 (2009), 283–309.

[178] Park, D., Kim, H., Choi, I., and Kim, J. A literature review and classification of recommender systems research. *Expert Systems with Applications 39*, 11 (2012), 10059–10072.

[179] Parpinelli, R., Freitas, A. A., and Lopes, H. S. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on 6* (2002), 321–332.

[180] Piatetsky-Shapiro, G. Discovery, analysis and presentation of strong rules. In *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991, pp. 229–248.

[181] Prasad, A., Iverson, L., and Liaw, A. Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems 9* (2006), 181–199.

[182] Purshouse, R. C., and Fleming, P. J. Conflict, harmony, and independence: relationships in evolutionary multi-criterion optimisation. In *International conference on Evolutionary Multi-criterion Optimization (EMO)* (2003), LNCS, pp. 16–30.

[183] Pyle, D. *Data preparation for data mining*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[184] Quinlan, J. R. Learning efficient classification procedures and their application to chess end games. In *Machine Learning. An Artificial Intelligence Approach*. Tioga Publishing Co., 1983, pp. 463–482.

[185] Quinlan, J. R. Induction of decision trees. *Machine Learning 1*, 1 (Mar 1986), 81–106.

[186] Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[187] Quinlan, J. R., and Rivest, R. L. Inferring decision trees using the minimum description length principle. *Information and Computation 80*, 3 (1989), 227–248.

[188] Rahman, A., Ezeife, C., and Aggarwal, A. Wifi miner: An online apriori-infrequent based wireless intrusion system. In *Proceedings of the 2nd International Workshop in Knowledge Discovery from Sensor Data (Sensor-KDD)* (2008), pp. 76–93.

[189] Rojas, S. A., and Bentley, P. J. A grid-based ant colony system for automatic program synthesis. In *Genetic and Evolutionary Computation Conference (GECCO)* (2004), R. P. et al., Ed., Late Braking Papers, pp. 1–12.

[190] Rokach, L. Ensemble-based classifiers. *Artificial Intelligence Reviews 33*, 1-2 (Feb 2010), 1–39.

[191] Romero, C. *Aplicación de técnicas de adquisición de conocimiento para la mejora de cursos hipermedia adaptativos basados en webs.* PhD thesis, Universidad de Granada, 2003.

[192] Romero, C., Ventura, S., Vasilyeva, E., and Pechenizkiy, M. Class association rules mining from students'test data. In *International Conference on Educational Data Mining (EDM)* (2010), pp. 317–318.

[193] Roux, O., and Fonlupt, C. Ant programming: or how to use ants for automatic programming. In *International conference on Swarm Intelligence (ANTS)* (2000), pp. 121–129.

[194] Roychowdhury, S. Association rule computation using lexicographic order combination. *Integrated Computer-Aided Engineering 10*, 4 (2003), 387–397.

[195] Runkler, T. Wasp swarm optimization of the c-means clustering model. *International Journal of Intelligent Systems 23*, 3 (2008), 269–285.

[196] Salama, K., Abdelbar, A., and Freitas, A. Multiple pheromone types and other extensions to the ant-miner classification rule discovery algorithm. *Swarm Intelligence 5*, 3-4 (2011), 149–182.

[197] Salama, K. M., and Abdelbar, A. M. Extensions to the ant-miner classification rule discovery algorithm. In *International conference on Swarm Intelligence (ANTS)* (2010), vol. 6234 of *LNCS*, Springer, pp. 167–178.

[198] Salehi-Abari, A., and White, T. Enhanced generalized ant programming (EGAP). In *Genetic and Evolutionary Computation Conference (GECCO)* (2008), pp. 111–118.

[199] Salehi-Abari, A., and White, T. The uphill battle of ant programming vs. genetic programming. In *International Joint Conference on Computational Intelligence (IJCCI)* (2009), pp. 171–176.

[200] Sánchez, D., Serrano, J., Cerda, L., and Vila, M. Association rules applied to credit card fraud detection. *Expert Systems with Applications 36* (2008), 3630–3640.

[201] Schwefel, H.-P., and Rudolph, G. Contemporary evolution strategies. In *European Conference on Artificial Life (ECAL)* (1995), pp. 893–907.

[202] Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal 27* (1948), 379–423.

[203] Shapiro, S., and Wilk, M. An analysis of variance test for normality (complete samples). *Biometrika 52*, 3-4 (1965), 591–611.

[204] Sherafat, V., Nunes de Castro, L., and Hruschka, E. Termitant: An ant clustering algorithm improved by ideas from termite colonies. In *Neural Information Processing*, vol. 3316 of *LNCS*. Springer Berlin / Heidelberg, 2004, pp. 1088–1093.

[205] Sheskin, D. J. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, 2007.

[206] Shirakawa, S., Ogino, S., and Nagao, T. Dynamic ant programming for automatic construction of programs. *Electrical and Electronic Engineering, IEEE Transactions on 3*, 5 (Aug 2008), 540–548.

[207] Smaldon, J., and Freitas, A. A. A new version of the ant-miner algorithm discovering unordered rule sets. In *Genetic and Evolutionary Computation Conference (GECCO)* (2006), pp. 43–50.

[208] Soda, P., and Iannello, G. Decomposition methods and learning approaches for imbalanced dataset: An experimental integration. In *International Conference on Pattern Recognition (ICPR)* (2010), pp. 3117–3120.

[209] Stepney, S., Smith, R. E., Timmis, J., Tyrrell, A. M., Neal, M. J., and Hone, A. N. W. Conceptual frameworks for artificial immune systems. *International Journal of Unconventional Computing 1*, 3 (Jul 2005), 315–338.

[210] Stützle, T., and Hoos, H. H. MAX-MIN Ant System. *Future Generation Computer Systems 16* (2000), 889–914.

[211] Su, M.-Y., Yu, G.-J., and Lin, C.-Y. A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach. *Computers and Security 28*, 5 (2009), 301–309.

[212] Sun, Y., Kamel, M. S., Wong, A. K. C., and Wang, Y. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition 40* (December 2007), 3358–3378.

[213] Sun, Y., Wong, A. K. C., and Kamel, M. S. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence 23*, 4 (2009), 687–719.

[214] Swaminathan, S. Rule induction using ant colony optimization for mixed variable attributes. Master's thesis, Texas Tech University, 2006.

[215] Tan, K. C., Tay, A., Lee, T. H., and Heng, C. M. Mining multiple comprehensible classification rules using genetic programming. In *IEEE Congress on Evolutionary Computation (IEEE CEC)* (2002), vol. 2, pp. 1302–1307.

[216] Tan, K. C., Yu, Q., Heng, C. M., and Lee, T. H. Evolutionary computing for knowledge discovery in medical diagnosis. *Artificial Intelligence in Medicine 27*, 2 (2003), 129–154.

[217] Tan, P.-N., Steinbach, M., and Kumar, V. *Introduction to Data Mining*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[218] Tang, Y., Zhang, Y.-Q., Chawla, N., and Krasser, S. SVMs modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39*, 1 (feb. 2009), 281–288.

[219] Thabtah, F. A review of associative classification mining. *Knowledge Engineering Review 22*, 1 (2007), 37–65.

[220] Ting, K. M. An instance-weighting method to induce cost-sensitive trees. *Knowledge and Data Engineering, IEEE Transactions on 14*, 3 (2002), 659–665.

[221] Tiwari, A., Gupta, R., and Agrawal, D. A survey on frequent pattern mining: Current status and challenging issues. *Information Technology Journal 9*, 7 (2010), 1278–1293.

[222] Tomek, I. Two modifications of CNN. *Systems, Man and Cybernetics, IEEE Transactions on 6*, 11 (Nov. 1976), 769–772.

[223] Torácio, A. *Swarm intelligence for multi-objective problems in data mining*, vol. 242/2009. Springer-Verlag, 2009, ch. Multiobjective Particle Swarm Optimization in Classification-Rule Learning, pp. 37–64.

[224] Ventura, S., Romero, C., Zafra, A., Delgado, J. A., and Hervás, C. JCLEC: a java framework for evolutionary computation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications 12*, 4 (2007), 381–392.

[225] Wang, Y. J., Xin, Q., and Coenen, F. A novel rule weighting approach in classification association rule mining. In *IEEE International Conference on Data Mining Workshops (IEEE ICDMW)* (2007), pp. 271–276.

[226] Wang, Z., and Feng, B. Classification rule mining with an improved ant colony algorithm. In *AI 2004: Advances in Artificial Intelligence*, G. Webb and X. Yu, Eds., vol. 3339 of *LNCS*. Springer Berlin / Heidelberg, 2005, pp. 177–203.

[227] Whigham, P. Grammatically biased genetic programming. In *Workshop on Genetic Programming: from Theory to Real-World Applications* (1995), pp. 33–41.

[228] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics 1* (1945), 80–83.

[229] Wilson, D. R., and Martinez, T. R. Reduction techniques for instance-based learning algorithms. *Machine Learning 38* (2000), 257–286.

[230] Witten, I. H., Frank, E., and Hall, M. A. *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Morgan Kaufmann, 2011.

[231] Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Computation 8*, 7 (1996), 1341–1390.

[232] Wolpert, D. H., and Macready, W. G. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on 1*, 1 (April 1997), 67–82.

[233] Wong, M. L., Lam, W., Leung, K. S., Ngan, P. S., and Cheng, J. C. Y. Discovering knowledge from medical databases using evolutionary algorithms. *IEEE Engineering in Medicine and Biology Magazine 19*, 4 (2000), 45–55.

[234] Wong, M. L., and Leung, K. S. The logic grammars based genetic programming system. In *Genetic and Evolutionary Computation Conference (GECCO)* (1996), pp. 433–433.

[235] Wrobel, S., Wettschereck, D., Verkamo, I. A., Siebes, A., Mannila, H., Kwakkel, F., and Klösgen, W. User Interactivity in Very Large Scale Data Mining. In *Annual Workshop of the GI Special Interest Group Machine Learning (FGML)* (09111 Chemnitz, Aug. 1996), W. Dilger, M. Schlosser, J. Zeidler, and A. Ittner, Eds., TU Chemnitz-Zwickau, pp. 125–130.

[236] Yan, X., Zhang, C., and Zhang, S. Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support. *Expert Systems with Applications 36*, 2, Part 2 (2009), 3066–3076.

[237] Yen, S.-J., and Lee, Y.-S. Cluster-based sampling approaches to imbalanced data distributions. In *Data Warehousing and Knowledge Discovery*, A. Tjoa and J. Trujillo, Eds., vol. 4081 of *LNCS*. Springer, 2006, pp. 427–436.

[238] Yu, L., and Liu, H. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research 5* (Dec 2004), 1205–1224.

[239] Zafra, A. *Modelos de programación genética gramatical para aprendizaje con múltiples instancias*. PhD thesis, Universidad de Córdoba, 2009.

[240] Zafra, A., and Ventura, S. G3P-MI: A genetic programming algorithm for multiple instance learning. *Information Sciences 180*, 23 (2010), 4496–4513.

[241] Zahálka, J., and Železný, F. An experimental test of Occam's razor in classification. *Machine Learning 82*, 3 (Mar 2011), 475–481.

[242] Zar, J. *Biostatistical Analysis*. Prentice Hall, 1999.

[243] Zhang, G. Neural networks for classification: A survey. *Systems, Man and Cybernetics Part C: Applications and Reviews, IEEE Transactions on 30*, 4 (2000), 451–462.

[244] Zhao, W., and Davis, C. E. A modified artificial immune system based pattern recognition approach. an application to clinical diagnostics. *Artificial Intelligence in Medicine 52*, 1 (May 2011), 1–9.

[245] Zhou, Z.-H., and Liu, X.-Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *Knowledge and Data Engineering, IEEE Transactions on 18*, 1 (Jan. 2006), 63–77.

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

[247] Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on 7*, 2 (2003), 117–132.