# Programación con Hormigas Multi-Objetivo para la Extracción de Reglas de Clasificación

J.L. Olmo, A. Cano, J.R. Romero, S. Ventura

*Resumen*— **La programación con hormigas (AP) es un tipo de programación automática que permite generar programas de ordenador para realizar una determinada tarea utilizando la metaheurística de optimización mediante colonias de hormigas. Recientemente ha demostrado una buena capacidad de generalización en la minería de reglas de clasificación. En este trabajo se extiende dicha investigación, proponiendo un nuevo algoritmo de AP que lleva a cabo la evaluación de los individuos desde una perspectiva multi-objetivo diseñada específicamente para la tarea de clasificación. El algoritmo desarrollado hace uso de una gramática de contexto libre que restringe el espacio de búsqueda y garantiza la creación de individuos válidos. El comportamiento del nuevo algoritmo se analiza con respecto al algoritmo de AP original y otros seis algoritmos de extracción de reglas sobre quince conjuntos de datos, demostrando que es capaz de alcanzar mejores resultados en cuanto a exactitud que el resto de los algoritmos considerados en el estudio, alcanzando al mismo tiempo una solución de compromiso entre exactitud y comprensibilidad.**

*Palabras clave*— **Minería de datos, clasificación, programación automática basada en gramática, programación con hormigas, optimización mediante colonias de hormigas multi-objetivo**

## I. Introducción

La extracción de conocimiento a partir de los datos se ha convertido en uno de los mayores retos en diferentes dominios, dada la ingente cantidad de datos disponible y su incesante crecimiento. A pesar de esta disponibilidad de datos, resulta complicado para los expertos en dichos dominios manejarlos y analizarlos para extraer información útil que les asesore y ayude en la toma de decisiones.

Los algoritmos de minería de datos (DM) permiten extraer información comprensible, no trivial y útil a partir de los datos. Entre las diversas tareas que comprende, una de las más estudiadas es la de clasificación, cuyo fin es la obtención de un modelo a partir de un conjunto de datos de entrenamiento etiquetados, es decir, cuya clase es conocida. Una vez inferido, este modelo se puede utilizar para clasificar nuevos datos cuya clase es desconocida, asociando cada instancia con una de las categorías existentes. El rendimiento de este modelo, también llamado clasificador, se mide normalmente con la exactitud predictiva obtenida cuando se aplica sobre un conjunto de datos de prueba que no se ha empleado durante la fase de construcción del mismo y donde se conoce la clase correspondiente a cada instancia.

Una metaheurística que ha sido aplicada con éxito en la extracción de clasificadores basados en reglas es la optimización mediante colonias de hormigas (ACO, Ant Colony Optimization) [1]. Se trata de una metaheurística bioinspirada que se fundamenta en el comportamiento y capacidad autoorganizativa que las colonias de hormigas exhiben en su búsqueda de alimento en la naturaleza. El algoritmo Ant-Miner [2] supuso la primera aplicación de ACO a la tarea de clasificación, y se ha convertido en un algoritmo de referencia en este campo.

Por otro lado, la programación automática es un método que emplea técnicas de búsqueda para construir automáticamente programas de ordenador que resuelven un problema dado, sin que haya que definir o incluso sea necesario conocer la forma o estructura de la solución. Simplemente hay que especificar los bloques básicos de los que se compone cualquier programa o individuo y la forma de evaluar cómo de bien se adapta una determinada solución al problema. Ejemplos típicos de esta técnica son la programación genética (GP, Genetic Programming) [3], [4], que usa algoritmos genéticos para guiar la búsqueda, y la programación automática con hormigas (AP, Ant Programming) [5], que emplea ACO como método de búsqueda. Diversos algoritmos de GP han demostrado un buen rendimiento en el diseño de clasificadores [6], y recientemente también se ha presentado un algoritmo de AP para clasificación con muy buenos resultados [7].

El diseño de un clasificador basado en reglas también se puede abordar desde el punto de vista de la optimización multi-objetivo: a diferencia de la optimización simple, trata de descubrir reglas de clasificación que optimizan simultáneamente varias medidas u objetivos, y en donde la mejora en los valores para uno frecuentemente obra en detrimento del resto de objetivos. Normalmente, los algoritmos de clasificación multi-objetivo se centran en obtener clasificadores con un buen compromiso entre exactitud y comprensibilidad. Cabe mencionar que se puede considerar que un clasificador es más comprensible que otro o en términos de su tamaño, si presenta menos reglas que el otro, o en términos de la longitud de sus reglas, si tiene un menor número medio de condiciones por regla. Se han presentado diversas propuesas basadas en la optimización multi-objetivo para la extracción de reglas utilizando algoritmos evolutivos [8], [9] y optimización con enjambres de partículas [10], [11]. Sin embargo, que sepamos no se han utilizado nunca algoritmos basados en ACO para abordar la tarea de clasificación desde esta perspectiva.

En este artículo se presenta un algoritmo de AP multi-objetivo para clasificación llamado MOGBAP (Multi-Objective Grammar-Based Ant Programming), que extrae un clasificador en forma de reglas IF-THEN. Nuestra propuesta combina las ventajas de AP basado en gramática con aquellas inherentes a las propuestas multi-objetivo, centrándose en obtener un buen balance entre exactitud y comprensibilidad. El algoritmo desarrollado aborda el descubrimiento de frentes de Pareto de un mo-

Departamento de Informática y Análisis Numérico, Universidad de Córdoba, Campus de Rabanales, Córdoba, España. Emails:{juanluisolmo, i52caroa, jrromero, sventura}@uco.es

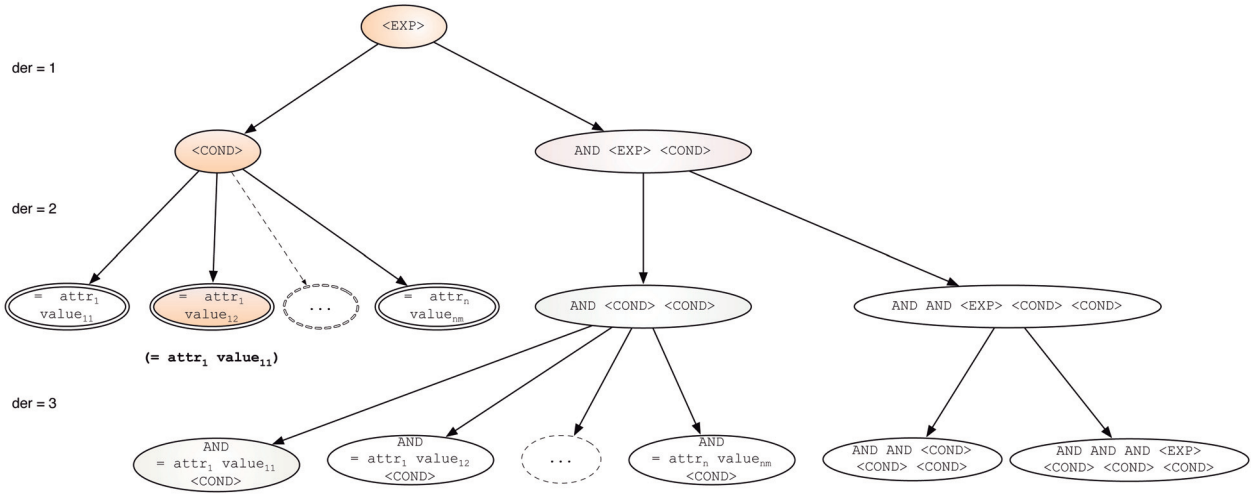



Fig. 1. Espacio de estados con una profundidad de tres derivaciones

do novedoso, encontrando un frente de Pareto para cada clase disponible en el conjunto de datos y combinando entonces las soluciones encontradas en cada frente por medio de un enfoque de nichos. El algoritmo se compara con otros siete algoritmos de inducción de reglas procedentes de diversos paradigmas, analizando los resultados estadísticamente. Los tests estadísticos prueban que nuestro algoritmo es significativamente más exacto que la mayoría de los algoritmos incluidos en el estudio, obteniendo asímismo un buen rendimiento en cuanto a comprensibilidad se refiere.

El resto del trabajo se organiza como sigue. En la próxima sección se describe el algoritmo MOGBAP. La sección III describe los experimentos realizados y los conjuntos de datos empleados. En la sección IV se detallan los resultados obtenidos y los tests estadísticos efectuados. Finalmente, a modo de conclusión, se plantean algunas observaciones en la sección V.

## II. Algoritmo MOGBAP

En los siguientes apartados se presentan las características particulares del algoritmo MOGBAP (Multi-Objective Grammar-Based Ant Programming) para clasificación. Se explican

Las características particulares del algoritmo se describen en los siguientes apartados, donde se explica cómo se construye el espacio de estados, la generación y codificación de los individuos, las medidas de heurística, el refuerzo de feromonas, las medidas de aptitud y el enfoque multi-objetivo.

### A. Entorno, generación de los individuos y codificación de las reglas

En MOGBAP, 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, como se muestra en la figura 1, y que comprenderá todas las posibles expresiones o programas que se pueden derivar a partir de la misma en un número máximo de derivaciones. En la figura se muestra el espacio de estados generado por la gramática hasta una profundidad de tres derivaciones. El camino sombreado representa un ejemplo de camino que podría seguir una determinada hormiga. Los estados con doble línea representan estados finales, compuestos únicamente por símbolos terminales, por lo que directamente codifican la expresión evaluable del antecente. Cumpliendo con las propiedades de una hormiga artificial [12], cada hormiga almacena el camino que ha explorado para poder realizar posteriormente una actualización de feromonas $offline$.

$G$ = ($\Sigma_N$, $\Sigma_T$, $P$, $EXP$)
$\Sigma_N$ = {<EXP>, <COND>}
$\Sigma_T$ = {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 válidas de la terna
= attr value}

Fig. 2. Gramática de contexto libre utilizada por MOGBAP

La gramática se encarga de garantizar que las hormigas sólo puedan visitar estados válidos, forzando asímismo que que las soluciones generadas sean válidas. Esta gramática se expresa en notación Backus-Naur, y se define en la figura 2 como $G$ = ($\Sigma_N$, $\Sigma_T$, $P$, $< EXP >$), donde $\Sigma_N$ es el conjunto de símbolos no terminales, $\Sigma_T$ es el conjunto de símbolos terminales, $P$ es el conjunto de reglas de producción y $EXP$ es el símbolo inicial. Nótese que las reglas de producción se expresan en notación prefija y siempre se derivan por la izquierda, lo que implica que cada transición de un estado $i$ a un estado $j$ se dispara tras aplicar una regla de producción de las disponibles para el primer símbolo no terminal del estado $i$.

En cuanto a la codificación de los individuos,

MOGBAP sigue el enfoque $hormiga = regla$, esto es, $individuo = regla$ [13]. Hay que resaltar que cuando una hormiga alcanza un estado final, en ese momento tan sólo representa el antecedente de una regla. El consecuente se asignará más tarde y se corresponderá con la clase que mejor cubra en cuanto a porcentaje de instancias.

### *B. Medidas de heurística*

Otra característica importante de MOGBAP es que considera dos heurísticas complementarias. La primera es la cardinalidad de las reglas de producción ($P_{card}$). Dado un estado $i$ y todos sus posibles estados siguientes, el valor de $P_{card}$ para una de las posibles transiciones se calcula como el cociente entre el número de soluciones que podría encontrar la hormiga si tomara esa transición y la suma total de soluciones posibles que se pueden alcanzar entre todas las posibles transiciones que parten del estado $i$. Para poder realizar este cálculo, es necesario que, previamente, cuando el algoritmo inicializa la gramática, se calcule una tabla de cardinalidad para cada regla de producción en función del número máximo de derivaciones permitidas. Con esta heurística se pretende guiar a las hormigas hacia transiciones que permitan llegar a un abanico mayor de soluciones. Esta medida incrementa, por tanto, la probabilidad de escoger transiciones de este tipo y está basada en la medida de cardinalidad propuesta en [14]. Nótese que esta función sólo es aplicable en transiciones intermedias, es decir, transiciones que no implican la selección de atributos del dominio del problema.

Para este tipo de transiciones, GBAP usa otra medida heurística: la ganancia de información. Esta medida complementa a la anterior, ya que se aplica en transiciones donde no puede aplicarse $P_{card}$, es decir, en aquellas que suponen la aplicación de reglas de producción que seleccionan atributos del dominio del problema (en el caso de la gramática definida en la figura 2, $< COND >:= operator - attribute - value$). Esta heurística es similar a la empleada por Ant-Miner.

El uso de ambas heurísticas afecta durante el proceso de creación de cada nueva hormiga, en la elección del siguiente movimiento de la misma, como se observa en la siguiente sección.

### *C. Regla de transición y actualización de feromonas*

Cualquier algoritmo basado en la metaheurística ACO sigue un método de construcción de soluciones probabilístico donde la hormiga, hasta encontrar una solución final, pasa de un estado dado a uno adyacente. Esta secuencia de transiciones viene guiada por cierta información. La influencia de dicha información es considerada en la regla de transición, que define la probabilidad de que una determinada hormiga se mueva de un estado $i$ a otro estado válido $j$:

$$P_{ij}^{k} = \frac{(\eta_{ij})^{\alpha} \cdot (\tau_{ij})^{\beta}}{\Sigma_{k \in allowed} (\eta_{ik})^{\alpha} \cdot (\tau_{ik})^{\beta}} \qquad (1)$$

donde $k$ es el número de posibles estados siguientes, $\alpha$ es el exponente que permite dar mayor importancia a la heurística, $\beta$ hace lo propio con el nivel de feromonas, $\eta$ es la función heurística y $\tau$ indica la intensidad del rastro de feromonas.

La regla de transición se utiliza para determinar el siguiente movimiento de una hormiga en su proceso de búsqueda de una solución. Asignará una probabilidad a cada posible estado siguiente y controlará que únicamente se escojan transiciones válidas, es decir, aquellas que permiten alcanzar al menos un estado final en un número de derivaciones inferior o igual al mayor número de pasos de derivación disponibles en ese punto. En caso contrario, a estas transiciones se les asigna una probabilidad de 0, lo que impedirá que la hormiga las escoja.

En cuanto al mantenimiento de feromonas, los algoritmos multi-objetivo de ACO se pueden encuadrar en dos categorías, según se almacene la información relativa a las feromonas [15], [16]. Así, pueden usar una única matriz de feromonas, o bien usar múltiples matrices, una por objetivo. MOGBAP pertenece al primer grupo, lo cual implica un beneficio en cuanto a requerimientos de memoria y tiempo de cómputo.

MOGBAP considera dos operaciones de mantenimiento: evaporación y refuerzo. La evaporación tiene lugar sobre el espacio de estados completo:

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot (1 - \rho) \qquad (2)$$

donde $\rho$ representa la tasa de evaporación.

A su vez, el refuerzo de feromonas sólo es realizado por aquellas hormigas que no están dominadas, es decir, aquellas pertenecientes a los frentes de Pareto. Así pues, una hormiga no dominada reforzará el rastro de feromonas en las transiciones que ha seguido de forma proporcional a la calidad de la solución medida por la exactitud de Laplace e inversamente proporcional a la longitud de su solución:

$$\tau_{ij}(t+1) = \tau_{ij}(t) \cdot Q \cdot LaplaceAccuracy \qquad (3)$$

donde $\tau_{ij}$ indica la cantidad de feromonas en la transición del estado *i* al *j*, y *Q* es una medida calculada que favorece las soluciones más comprensibles. De hecho, el valor de esta medida se calcula como el cociente entre el máximo número de derivaciones permitidas en esta generación y la longitud del camino seguido por la hormiga.

La exactitud de Laplace también se utiliza en el algoritmo GBAP mono-objetivo para medir la calidad de los individuos y se define como:

$$LaplaceAccuracy = \frac{1 + TP}{k + TP + FP} \qquad (4)$$

donde $TP$ y $FP$ hacen referencia a verdaderos positivos y falsos positivos, respectivamente, y $k$ es el número de clases en el conjunto de datos.

Cuando las operaciones de actualización de feromonas han finalizado tiene lugar un proceso de normalización para restringir los niveles de feromonas de cada transición al rango $[\tau_{min}, \tau_{max}]$.

Debido a que la tasa de feromonas queda restringida al intervalo mencionado y a que en el espacio de estados las feromonas iniciales para cada transición corresponden al valor máximo de feromonas permitido, el algoritmo clásico de ACO con el que MOGBAP comparte más características es el Max-Min Ant System (MMAS) [17]. Además, en MOGBAP las hormigas no dominadas son las encargadas de realizar el refuerzo, al igual que en el MMAS, donde sólo la mejor hormiga lleva a cabo dicha tarea.

### D. Objetivos utilizados en la función de aptitud

La calidad de los individuos generados en MOGBAP se calcula en base a tres objetivos contradictorios: sensibilidad, especificidad y comprensibilidad.

La sensibilidad y la especificidad son dos medidas ampliamente utilizadas en problemas de clasificación, incluso como una agregación escalar de las mismas. Por ejemplo, el algoritmo Ant-Miner utiliza como función de $fitness$ el producto de ambas. La sensibilidad indica cómo de bien identifica una regla los casos positivos. Por contra, la especificidad informa de la efectividad de una regla identificando los casos negativos o aquellos casos que no pertenecen a la clase estudiada. Si el valor de sensibilidad de una regla incrementa, predecirá un mayor número de ejemplos positivos pero a veces a expensas de clasificar como positivos algunos casos que, en realidad, son negativos. El objetivo es maximizar ambas medidas.

$$Sensibilidad = \frac{T_P}{T_P + F_N} \tag{5}$$

$$Especificidad = \frac{T_N}{T_N + F_P} \tag{6}$$

Dado que MOGBAP es un algoritmo de clasificación basado en reglas, se pretende que, además de extraer reglas de elevada exactitud, dichas reglas sean comprensibles. Por tanto, de alguna manera, es conveniente optimizar la complejidad de las reglas extraídas. Aunque la comprensibilidad es un concepto subjetivo, existen distintas formas de medir la comprensibilidad de las reglas y de un clasificador, normalmente contando el número de condiciones por regla y el número de reglas que aparecen en el clasificador final, respectivamente. Esta última no se puede considerar aquí como un objetivo, dado que MOGBAP sigue el enfoque $hormiga = regla$, como se mencionó en la Sección II-A. Por otro lado, si el número de condiciones por regla se usa como objetivo, habría que minimizarlo. No obstante, suponiendo que una regla puede llegar a tener hasta un número máximo fijo de condiciones, la comprensibilidad se puede medir como:

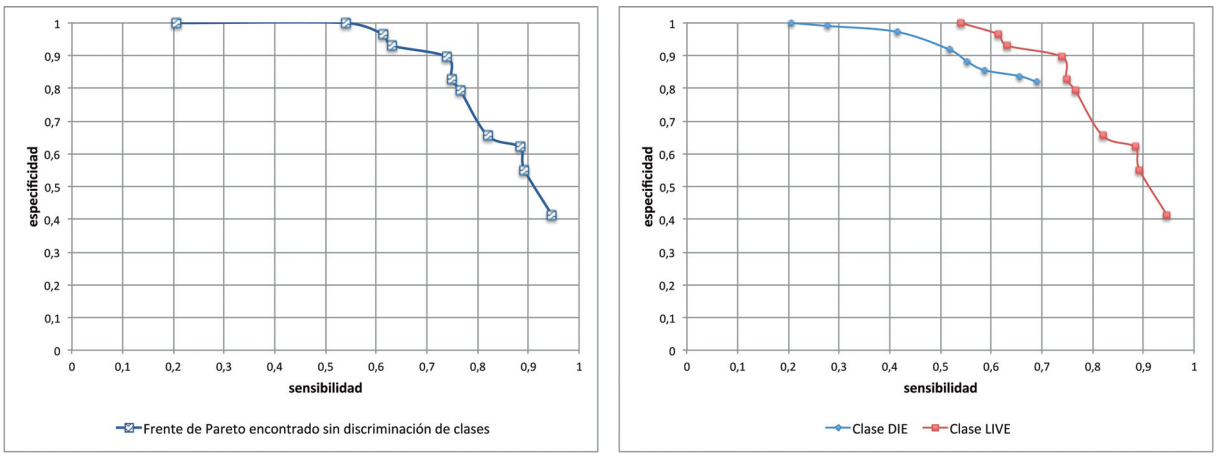
$$Comprensibilidad = 1 - \frac{numCondiciones}{maxCondiciones} \tag{7}$$

donde $numCondiciones$ hace referencia al número de condiciones que aparecen en la regla codificada por el individuo, mientras que $maxCondiciones$ es el número máximo de condiciones que puede tener una regla.

En MOGBAP, es fácil conocer el número máximo de condiciones que puede tener un individuo, ya que la gramática se conoce de antemano y el número máximo de derivaciones también. La ventaja de usar esta medida de comprensibilidad como objetivo radica en que sus valores estarán contenidos en el intervalo [0,1], y cuanto más se acerque a 1 su valor, más comprensible será el individuo. Por lo tanto, este objetivo debe maximizarse, al igual que la sensibilidad y la especificidad. Teniendo en cuenta estos tres objetivos, una determinada regla $hormiga_i$ dominará a otra $hormiga_j$, denotado como $hormiga_i \succ hormiga_j$, si la $hormiga_i$ no es peor que la $hormiga_j$ en ningún objetivo y es mejor en al menos un objetivo.

### E. Estrategia multi-objetivo

MOGBAP sigue una estrategia multi-objetivo especialmente diseñada para la tarea de clasificación. La idea de este esquema es distinguir los individuos en función de la clase que predicen, ya que determinadas clases pueden ser más difíciles de predecir que otras. En realidad, si




(a)Frente de Pareto que encontraría un enfoque clásico

(b)Frentes de pareto encontrados mediante el enfoque de MOGBAP

Fig. 3. Comparación entre el nuevo enfoque de $k$ frentes de Pareto frente al clásico sobre el conjunto de datos $hepatitis$ de dos clases

individuos pertenecientes a diferentes clases se ordenan por frentes de acuerdo a la dominancia de Pareto puede darse solapamiento, como se ilustra en las figuras 3 y 4, que muestran los frentes de Pareto encontrados al ejecutar el algoritmo sobre los conjuntos de datos $hepatitis$ y $lymphography$. Para ilustrar con mayor claridad el concepto se han considerado dos objetivos, sensibilidad y especificidad.

Por ejemplo, para el conjunto de datos $hepatitis$, si se emplease un enfoque de Pareto clásico se encontraría un único frente de soluciones, como se muestra en la figura 3(a). Dicho frente de Pareto estaría compuesto por todos los individuos que predicen la clase $LIVE$ y tan sólo un individuo de la clase $DIE$ (el individuo cuya especificidad es 1.0). Para tener en cuenta al resto de individuos de la clase $DIE$ sería necesario encontrar frentes adicionales por debajo del frente de Pareto, y los individuos de estos frentes tendrían menor probabilidad de pertenecer a la lista de decisión del clasificador. En cambio, el enfoque multi-objetivo de MOGBAP (ver figura 3(b)) garantiza que todas las soluciones no dominadas para cada clase disponible sean consideradas, de forma que se asegura la inclusión de reglas prediciendo cada clase en el clasificador final.

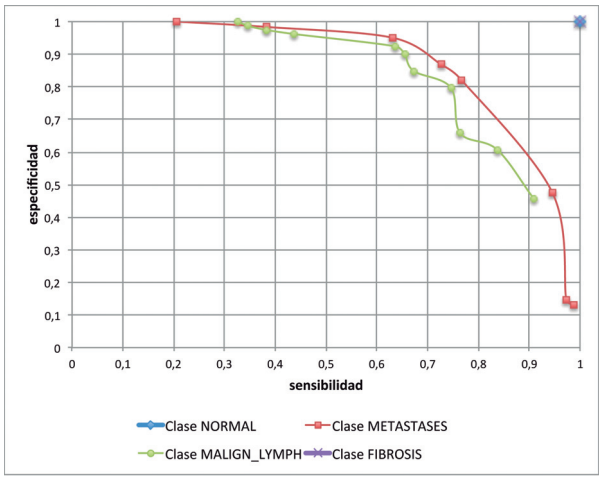



Fig. 4. Frentes de Pareto encontrados para el problema *lymphography*

Se puede justificar un comportamiento similar en el caso de la figura 4, correspondiente al conjunto de datos $lymphography$, donde muchas soluciones no dominadas de las clases $METASTASES$ y $MALIGN\_LYMPH$ quedarían ocultas por las soluciones de las clases $NORMAL$ y $FIBROSIS$ si todas las clases se consideraran simultáneamente y se encontrara un único frente de Pareto. En este caso, el frente de Pareto consistiría en un único punto, el (1.0, 1.0). Además, para llegar a contemplar reglas prediciendo la clase $MALIGN\_LYMPH$ habría que encontrar al menos tres frentes (ya que la figura muestra sólo los frentes de Pareto de cada clase y, por tanto, puede haber frentes intermedios formados por otras soluciones no correspondientes a los frentes de Pareto).

En resumen, el enfoque multi-objetivo diseñado para MOGBAP consiste en descubrir un frente separado de soluciones no dominadas para cada clase disponible en el conjunto de datos. Para ello, una vez que los individuos de la generación actual han sido creados y evaluados para cada uno de los objetivos considerados, se dividen en $k$ grupos en función de su consecuente, siendo $k$ el número de clases en el conjunto de entrenamiento. A continuación, cada grupo de individuos se combina con las soluciones almacenadas en su correspondiente frente de Pareto, obtenido en la generación previa del algoritmo, y se clasifican todos los individuos de acuerdo a la dominancia de Pareto, hallando un nuevo frente de Pareto para cada clase. Por tanto, existirán $k$ frentes de Pareto, y sólo las soluciones no dominadas participarán en el refuerzo de feromonas.

El clasificador final estará compuesto de los individuos no dominados que existen en cada uno de los $k$ frentes de Pareto una vez que la última generación del algoritmo concluye. Para seleccionar adecuadamente las reglas del clasificador final, se lleva a cabo sobre cada frente un enfoque de nichos similar al que utiliza el algoritmo GBAP original. La diferencia es que en GBAP dicho procedimiento no se ejecuta una sola vez para seleccionar las reglas del clasificador final, sino que se ejecuta en todas las generaciones, y además participan todos los individuos creados en cada generación.

Experimentalmente se ha comprobado que este nuevo enfoque multi-objetivo para clasificación supera en rendimiento a utilizar un procedimiento similar al utilizado por el algoritmo NSGA-II [18].

## III. Experimentación

### A. Conjuntos de datos y preprocesado

La simulación ha sido realizada utilizando quince problemas estándar de clasificación del repositorio de la UCI [1]. Esta colección de conjuntos de datos considera problemas con un variado rango de características en cuanto a dimensionalidad, tipo de atributos y número de clases. Los conjuntos de datos con valores perdidos han sido preprocesados reemplazando dichos valores por la moda (en caso de atributos nominales) o por la media aritmética (para atributos numéricos) del atributo en el conjunto de datos. Así mismo, aquellos conjuntos de datos con atributos numéricos fueron discretizados para tratar sólo con atributos categóricos.

Con respecto a la evaluación del algoritmo, se ha aplicado un proceso de validación cruzada sobre diez particiones estratificadas. Además, dado que algunos algoritmos utilizados en la experimentación tienen algún componente estocástico y, por ende, una ejecución aislada puede verse influenciada por la semilla utilizada, sus experimentos se han repetido diez veces con diferentes semillas, y los resultados que se muestran son la media de las diez ejecuciones.

[1]El repositorio de la Universidad de California en Irvine está disponible en `http://www.ics.uci.edu/ml/datasets.html`

TABLA I
RESULTADOS COMPARATIVOS DE EXACTITUD PREDICTIVA ( %)

| | MOGBAP | | GBAP | | ANT-MINER | | ANT-MINER+ | | PSO/ACO2 | | GP | | JRIP | | PART | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conjunto datos | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ | Acc | $\sigma_{Acc}$ |
| Hepatitis | **85.15** | 1.52 | 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 | 79.49 | 9.26 | **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 | 72.02 | 9.62 | 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 | **83.13** | 4.24 | 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 | 90.55 | 5.67 | **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 | **83.78** | 4.67 | 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 |
| Vote | 94.89 | 2.92 | 94.37 | 3.57 | 94.29 | 3.27 | 94.66 | 3.72 | 94.80 | 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 | 85.19 | 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 | 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 | 70.36 | 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 | 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 | 90.20 | 2.86 | 83.69 | 9.44 | 95.61 | 5.37 | 95.03 | 3.89 |
| Lymphography | 80.55 | 9.74 | **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 | 71.03 | 8.45 | 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 |
| Primary | **42.18** | 7.19 | 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 | **2.4** | | 3.0003 | | 5.7333 | | 5.4666 | | 5.0333 | | 6.3666 | | 3.4 | | 4.5666 | |

### *B. Comparativa con otros algoritmos de extracción de reglas*

En este trabajo se compara el algoritmo desarrollado frente a otros basados en reglas pertenecientes a diferentes paradigmas: el algoritmo de AP original, llamado GBAP [7]; dos algoritmos de ACO, Ant-Miner [2] y Ant-Miner+ [19]; un algoritmo que sigue un enfoque híbrido entre ACO y la optimización mediante enjambres de partículas, PSO/ACO2 [20]; un algoritmo de programación genética, Bojarczuk-GP [6]; la implementación de WEKA[2] del algoritmo de cubrimiento secuencial RIPPER [21], denominado JRIP; y el algoritmo PART [22], que extrae reglas del árbol de decisión generado por el algoritmo J48 de WEKA.

Dichos algoritmos se han configurado con los valores de los parámetros propuestos originalmente por sus autores. La configuración del algoritmo MOGBAP fue la siguiente: 20 hormigas, 100 iteraciones, 15 derivaciones máximas para la gramática, porcentaje de cubrimiento mínimo de instancias de la clase predicha por la regla del 5 %, cantidad inicial y cantidad máxima de feromonas de 1.0, mínima cantidad de feromonas de 0.1, tasa de evaporación de 0.05, valor de $\alpha$ de 0.4 y valor de $\beta$ de 1.0.

## IV. RESULTADOS

El rendimiento de MOGBAP se compara en primer lugar frente a los demás algoritmos en términos de exactitud predictiva. La tabla I muestra los valores de exactitud predictiva y desviación típica obtenidos por cada algoritmo sobre cada uno de los conjuntos de datos del estudio. Los resultados resaltados en negrita indican el algoritmo que mejor se comporta en este aspecto para un determinado conjunto de datos.

[2]El software de aprendizaje automático de WEKA está disponible en `http://www.cs.waikato.ac.nz/ml/index.html`

Para evaluar estadísticamente las diferencias entre los algoritmos se ha realizado el test de Iman-Davenport [23]. Se trata de un test no paramétrico que compara la media en rango de $k$ algoritmos sobre $N$ conjuntos de datos. Estos rangos indican qué algoritmo obtiene los mejores resultados teniendo en cuenta todos los conjuntos de datos. Para calcularlos, un rango de 1 se le asigna al algoritmo con mayor exactitud en un problema, un 2 al algoritmo con segundo mejor valor de exactitud, y así sucesivamente. Finalmente se calcula la media de rangos para cada algoritmo sobre todos los conjuntos de datos, como se muestra en la última fila de la tabla.

El test de Iman-Davenport plantea la hipótesis nula de que todos los algoritmos se comportan igual, es decir, son equivalentes. Si la rechaza, existirán diferencias significativas entre los algoritmos. Para un nivel de relevancia de $\alpha = 0.05$, el estadístico de los rangos medios distribuidos de acuerdo a una distribución $F$ con $(k-1)$ y $(k-1)\cdot(N-1)$ grados de libertad es igual a 7.0, el cuál no pertenece al intervalo crítico, que es $C_0 = [0, (F_F)_{0,05,7,98} = 2{,}1044]$. Por tanto, el test de Iman-Davenport rechaza la hipótesis nula.

Para revelar dónde se producen las diferencias significativas es necesario realizar un test $a\ posteriori$. Podemos aplicar el test de Bonferroni-Dunn ya que todos los algoritmos se comparan frente a un algoritmo de control [23], MOGBAP, centrándose en todas las posibles comparaciones por parejas que implican al algoritmo MOGBAP. El valor crítico que indica este test para el nivel $alpha = 0.05$ es de 2.4060 y, por tanto, el rendimiento de MOGBAP es estadísticamente mejor en cuanto a exactitud predictiva que el de los algoritmos PSO/ACO2, Ant-Miner+, Ant-Miner y GP. Estos resultados quedan reflejados en la figura 5, donde se observa también que MOGBAP obtiene valores de exactitud competitivos e incluso mejores que GBAP, PART y JRIP.

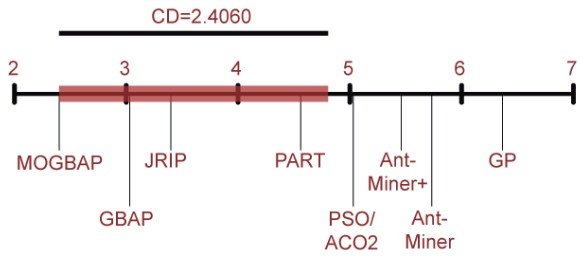



Fig. 5. Test de Bonferroni-Dunn para la exactitud predictiva

Como se mencionó anteriormente, todos los algoritmos incluidos en la comparación son algoritmos basados en reglas. El número medio de reglas y el número medio de condiciones por regla obtenidos por cada algoritmo en cada conjunto de datos se detallan en la tabla II. La penúltima fila de la tabla indica el rango medio obtenido por cada algoritmo respecto a la longitud del clasificador, y la última fila especifica el rango medio obtenido para el número medio de condiciones por regla. En ambos casos, el algoritmo con el rango más bajo y, por tanto, con mejor comportamiento, se corresponde con el algoritmo de GP.

Cabe mencionar que todos los algoritmos extraen las reglas utilizando conjunciones de condiciones, excepto el algoritmo de GP, que también utiliza el operador lógico de disyunción como conector de condiciones. En el caso de este último algoritmo, para calcular correctamente el número de reglas del clasificador, es necesario considerar cada disyunción como la conexión de dos reglas diferentes, sin tener en cuenta los nodos OR como condiciones.

Para analizar estos resultados se realiza un primer análisis concerniente al número de reglas en el clasificador (a menor número, más comprensible será el clasificador). Suponiendo un nivel de significación de 0.05, el valor del estadístico de Iman-Davenport de acuerdo a una distribución $F$ es igual a 22.3511, y el intervalo crítico es, al igual que en el caso anterior, $C_0 = [0, (F_F)_{0,05,7,98} = 2{,}1044]$. Por tanto, dado que queda fuera de dicho intervalo, el test de Iman-Davenport rechaza la hipótesis nula, lo que significa que existen diferencias significativas entre los algoritmos para esta medida. Siguiendo con el mismo nivel de significación, la diferencia entre los rangos de GP, JRIP, Ant-Miner+ y el rango de MOGBAP es superior al valor crítico del test de Bonferroni-Dunn, lo que quiere decir que dichos algoritmos obtienen estadísticamente un clasificador más comprensible en cuanto a número de reglas de MOGBAP. Sin embargo, no existen diferencias significativas entre MOGBAP y el resto de algoritmos.

Idealmente, el mejor resultado posible en cuanto a complejidad del clasificador sería obtener una única regla para predecir cada clase. Sin embargo, esto puede ser contraproducente para la exactitud obtenida por el algoritmo. Este efecto se puede observar en los resultados que obtiene el algoritmo de GP, ya que el clasificador contiene, en media, casi una regla por cada clase, pero sus resultados en cuanto a exactitud son los más bajos. En contraste, MOGBAP llega a un buen compromiso entre ambas medidas, obteniendo los valores de exactitud más altos y al mismo tiempo comportándose de forma competitiva en lo que respecta a la longitud del clasificador.

También se ha efectuado un segundo análisis para analizar la complejidad de las reglas extraídas. En este caso, el valor del estadístico es 17.6205, que tampoco pertenece al intervalo crítico, por lo que vuelven a existir diferencias significativas. El test de Bonferroni-Dunn nos conduce a dos conclusiones. Por un lado, MOGBAP es estadísticamente mejor que Ant-Miner+ en cuanto a esta métrica. Y, por otro lado, MOGBAP no es significativamente mejor ni peor que los algoritmos GBAP, Ant-

TABLA II

RESULTADOS COMPARATIVOS DEL TAMAÑO DEL CLASIFICADOR Y LA COMPLEJIDAD DE LAS REGLAS

| | MOGBAP | | GBAP | | ANT-MINER | | ANT-MINER+ | | PSO/ACO2 | | GP | | JRIP | | PART | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Conjunto datos | #R | #C/R | #R | #C/R | #R | #C/R | #R | #C/R | #R | #C/R | #R | #C/R | #R | #C/R | #R | #C/R |
| Hepatitis | 9.1 | 1.99 | 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 | 11.1 | 2.04 | 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 | 11.8 | 1.69 | 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 | 9.9 | 2.25 | 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 | 6.8 | 1.49 | 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 | 9.6 | 2.03 | 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 |
| Vote | 6.6 | 2.12 | 17.2 | 2.19 | 5.6 | 1.36 | 5.2 | 2.34 | 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 | 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 | 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 | 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 | 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 | 4.0 | 1.73 | 4.1 | 1.27 | 4.2 | 1.56 | 6.3 | 1.77 |
| Lymphography | 11.9 | 1.55 | 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 | 17.5 | 2.22 | 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 |
| Primary | 34.9 | 2.53 | 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 | 5.6333 | | 6.3 | | 3.9666 | | 2.7666 | | 5.7333 | | **1.9333** | | 2.6333 | | 7.0333 | |
| #C/R RANKING | 4.1333 | | 3.4333 | | 3.1666 | | 7.4 | | 6.2 | | **1.8** | | 4.1 | | 5.76667 | |

Miner, PSO/ACO2, GP, JRIP y PART.

Cabe resaltar que la longitud de las reglas extraídas por MOGBAP depende directamente del parámetro para el número de derivaciones máximo. Cuantas más derivaciones se permitan para la gramática, el algoritmo será capaz de encontrar relaciones más complejas entre los atributos, lo que puede conllevar la extracción de reglas con mayor número de condiciones.

Tras realizar todos los tests estadísticos es posible concluir que MOGBAP presenta muy buen compromiso entre exactitud y comprensibilidad, dado que es al algoritmo que mejores resultados alcanza en exactitud predictiva, obteniendo también resultados muy competitivos de comprensibilidad.

## V. Conclusiones

En este artículo se ha presentado un nuevo algoritmo de programación automática con ACO multi-objetivo guiado por una gramática de contexto libre para extraer reglas de clasificación. La propuesta se sustenta en una doble función heurística y en la posibilidad de modificar la complejidad de las reglas variando el número de derivaciones permitidas por la gramática. Otra contribución del trabajo a la tarea de clasificación es la propuesta de una estrategia de evaluación multi-objetivo que permite evitar el problema del solapamiento y ocultación entre individuos de diferentes clases a la hora de ordenarlos por rangos de dominancia. Para ello, se encuentra un conjunto de individuos no dominados por cada posible clase en el conjunto de entrenamiento. Estas soluciones se guardan en frentes de Pareto separados hasta la última generación del algoritmo, cuando se combinan en el clasificador final mediante un enfoque de nichos.

El rendimiento de MOGBAP se ha comparado frente a otros siete algoritmos de extracción de reglas de clasificación, utilizando para ello tests estadísticos no paramétricos. Los resultados indican que MOGBAP es estadísticamente más exacto que los otros algoritmos basados en hormigas considerados (Ant-Miner, Ant-Miner+ y PSO/ACO2) y que el algoritmo de GP. Además, nuestro algoritmo obtiene resultados competitivos con respecto a la complejidad del clasificador final y de las reglas extraídas, alcanzando un buen compromiso entre exactitud y comprensibilidad. Estos resultados confirman que la AP multi-objetivo es una técnica adecuada para abordar problemas de clasificación.

## Agradecimientos

Este trabajo ha sido financiado conjuntamente por los proyectos P08-TIC-3720 de la Junta de Andalucía, TIN2008-06681-C06-03 del Ministerio de Ciencia y Tecnología, y por la fundación FEDER.

## Referencias

[1] Marco Dorigo and Thomas Stützle, *The Ant Colony Optimization metaheuristic: Algorithms, Applications and Advances*, International Series in Operations Research and Management Science. Kluwer Academic Publishers, 2002.

[2] R. Parpinelli, A. A. Freitas, and H. S. Lopes, "Data mining with an ant colony optimization algorithm," *IEEE Trans on Evolutionary Computation*, vol. 6, pp. 321–332, 2002.

[3] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, The MIT Press, Cambridge, MA, 1992.

[4] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone, *Genetic Programming - An Introduction; On the Automatic Evolution of Computer Programs and its Applications*, Morgan Kaufmann, San Francisco, CA, USA, January 1998.

[5] M. Birattari, G. Di Caro, and M. Dorigo, "Toward the formal foundation of ant programming," *LNCS*, vol. 2463, pp. 188–201, 2002.

[6] Celia C. Bojarczuk, Heitor S. Lopes, Alex Alves Freitas, and Edson L. Michalkiewicz, "A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets," *Artificial Intelligence in Medicine*, vol. 30, pp. 27–48, 2004.

[7] J. L. Olmo, J. R. Romero, and S. Ventura, "Using ant programming guided by grammar for building rule-based classifiers," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–15, 2011.

[8] S. Dehuri, S. Patnaik, A. Ghosh, and R. Mall, "Application of elitist multi-objective genetic algorithm for classification rule generation," *Appl. Soft Comput.*, vol. 8, pp. 477–487, January 2008.

[9] Hisao Ishibuchi and Yusuke Nojima, "Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning," *International Journal of Approximate Reasoning*, vol. 44, no. 1, pp. 4 – 31, 2007, Genetic Fuzzy Systems and the Interpretability-Accuracy Trade-off.

[10] Bilal Alatas and Erhan Akin, "Multi-objective rule mining using a chaotic particle swarm optimization algorithm," *Knowledge-Based Systems*, vol. 22, no. 6, 2009.

[11] A. Torácio, *Multiobjective Particle Swarm Optimization in Classification-Rule Learning*, chapter 3, pp. 37–64, Springer-Verlag, 2009.

[12] R. J. Mullen, D. Monekosso, S. Barman, and P. Remagnino, "A review of ant algorithms," *Expert Systems with Applications*, vol. 36, pp. 9608–9617, 2009.

[13] P.G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 2, pp. 121–144, march 2010.

[14] Andreas Geyer-Schulz, *Fuzzy Rule-Based Expert Systems and Genetic Machine Learning*, vol. 3 of *Studies in Fuzziness*, Physica-Verlag, Heidelberg, 1995.

[15] Daniel Angus and Clinton Woodward, "Multiple objective ant colony optimisation," *Swarm Intelligence*, vol. 3, no. 1, pp. 69–85, March 2009.

[16] C. García-Martínez, O. Cordón, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp," *European Journal of Operational Research*, vol. 180, no. 1, pp. 116 – 148, 2007.

[17] Thomas Stützle and Holger H. Hoos, "MAX-MIN Ant System," *Future Generation Computer Systems*, vol. 16, pp. 889–914, 2000.

[18] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[19] D. Martens, M. De Backer, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Trans on Evolutionary Computation*, vol. 11, pp. 651–665, 2007.

[20] Nicholas Holden and Alex A. Freitas, "A hybrid PSO/ACO algorithm for discovering classification rules in data mining," *J. Artif. Evol. App.*, vol. 2008, pp. 2:1–2:11, January 2008.

[21] William W. Cohen, "Fast Effective Rule Induction," in *In Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 115–123.

[22] Eibe Frank and Ian H. Witten, "Generating accurate rule sets without global optimization," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, ICML '98, pp. 144–151.

[23] Janez Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.

[24] M. Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53 –66, apr 1997.