Clustering dinámico con hormigas artificiales

Diego Alejandro Ingaramo, Guillermo Leguizamón, Marcelo Errecalde
LIDIC - Departamento de Informática. Universidad Nacional de San Luis
Ejército de los Andes 950. (D5700HHW) San Luis, Argentina
{daingara,legui,merreca}@unsl.edu.ar
VI Workshop de Agentes y Sistemas Inteligentes (WASI)

Resumen

La tarea de *clustering* consiste en la clasificación no supervisada de patrones (observaciones, datos, vectores, etc.) en grupos. Este problema ha sido analizado en varios contextos y por investigadores de distintas disciplinas, reflejando su amplia utilidad. Si bien se han propuesto distintas alternativas para abordar las tareas de clustering, existe un área particularmente interesante y novedosa que ha planteado distintos enfoques bio-inspirados que incluyen los algoritmos genéticos y algoritmos basados en la metáfora del comportamiento de las hormigas. En este trabajo, analizamos la utilización de algoritmos basados en el comportamiento de hormigas en la Minería de Datos, y más específicamente, en la tarea de clustering. Entre estos algoritmos podemos mencionar al *AntTree*, inspirado en las posibilidades de auto-ensamblaje de las hormigas reales. También se propone una extensión a este algoritmo que incluye la capacidad de desconexión del árbol por parte de las hormigas con el objeto de posicionarse en otro grupo más adecuado. Ésto permite flexibilizar el proceso del descubrimiento de clusters dentro de los datos a analizar. La factibilidad del enfoque propuesto es analizada experimentalmente considerando distintas instancias del problema de clustering. Los resultados obtenidos son comparados con los del algoritmo AntTree original y los de *K-means*, uno de los algoritmos de clustering tradicional más utilizados.

Palabras claves: Sistemas Inteligentes, Técnicas bio-inspiradas, Clustering, Minería de Datos.

1. Introducción

La tarea de *clustering* [4] es la clasificación no supervisada de patrones (observaciones, datos, vectores, etc.) en grupos. Este problema ha sido analizado en varios contextos y por investigadores de distintas disciplinas, reflejando su amplia utilidad. Es un problema de gran dificultad combinatoria y, dado a que se ha utilizado en diferentes áreas, los métodos obtenidos carecen de generalidad.

La tarea de clustering consiste en la organización de una colección de patrones (usualmente representados como vectores de atributos o puntos en un espacio multidimensional) en clusters (o grupos) basándose en la similitud que existe entre los mismos. Intuitivamente, patrones de un mismo cluster son más similares a patrones que se encuentran fuera del mismo. Los humanos resuelven de manera competitiva problemas de clustering en dos dimensiones, pero la mayoría de los problemas reales implican dimensiones más grandes. Además, la distribución de los datos muy difícilmente siga una forma definida. Por ello, encontramos una gran cantidad de algoritmos que se comportan de mejor o peor manera dependiendo de la distribución del conjunto de datos.

La variedad de técnicas que existen difieren en la representación de los datos, en la medida de proximidad (o similitud) entre elementos, y en la manera que agrupan los elementos. El método más simple que resuelve el problema de clustering se denomina *K-means*, en donde debemos definir la

cantidad de grupos (llamados *centroides*) que existen en los datos. Cada centroide define un grupo de datos y se asocia cada dato al centroide más cercano. Luego, iterativamente se recalculan estos centroides de tal forma que en cada iteración se minimiza la función SSE (suma de los errores al cuadrado). Cuando no existe mejora, el algoritmo finaliza su ejecución. Este método posee desventajas muy importantes. Una de las principales es que se debe especificar el número de clusters desde el principio, dato que generalmente no se conoce. Esto significa que para aplicar dicho algoritmo, se presupone un conocimiento previo de la distribución de los datos.

Debido a la gran importancia del problema de clustering en diferentes campos, en la literatura se han propuesto distintos métodos para resolverlo. Recientemente, enfoques bio-inspirados tales como los algoritmos genéticos y metaheurísticas como tabu search, Ant Colony Optimization (ACO) y simulated annealing han sido aplicados exitosamente a este problema. Sin embargo, en la actualidad ha surgido un importante grupo de algoritmos basados en la metáfora del comportamiento de las hormigas reales los cuales son aplicados a clustering. Entre dichos algoritmos podemos citar a *Ant-Class*, *Ant-Tree* y *Ant-Clust*.

En este trabajo investigamos la aplicación de algoritmos basados en el comportamiento de las colonias de hormigas (BCH) a problemas de clustering. En particular, la propuesta se centra en una nueva versión del algoritmo Ant-Tree a la que denominamos *DAntTree*. El algoritmo DAntTree incorpora aspectos dinámicos de las hormigas en el proceso de construcción de los clusters, permitiendo que las hormigas se desconecten del árbol y se posicionen en otro grupo más adecuado. Esto flexibiliza el descubrimiento de clusters dentro de los datos a analizar y permite incrementar el grado de precisión del algoritmo en cuanto al número de clusters descubiertos. El trabajo incluye un estudio comparativo entre DAntTree, el AntTree original y el algoritmo K-means a los efectos de analizar sus potenciales ventajas y desventajas.

El resto del trabajo se organiza de la siguiente manera. En la sección 2 realizamos una breve reseña de la aplicación de técnicas bio-inspiradas al problema de clustering. En la sección 3 describimos el algoritmo AntTree que sirve de base para nuestra propuesta que presentamos luego en la sección 4. En la sección 5 se describe el trabajo experimental realizado. En la sección 6 discutimos fortalezas y limitaciones del enfoque propuesto y posibles extensiones futuras.

2. Clustering mediante técnicas bio-inspiradas

Recientemente se han aplicado al problema de clustering distintos enfoques bio-inspirados tales como los algoritmos genéticos [12, 6] y metaheurísticas como tabu search y simulated annealing [1]. Otra de las metaheurísticas utilizada es el enfoque Ant Colony Optimization (ACO) [5], adaptado al problema de clustering como se describe en [11].

Dentro de estas propuestas bio-inspiradas se destacan un grupo de algoritmos novedosos basados en la metáfora del comportamiento de las colonias de hormigas reales (BCH). Estas técnicas demostraron ser muy efectivas, en comparación con métodos tradicionales, debido a su comportamiento probabilístico (que le permite evitar mínimos locales) y la obtención de buenos resultados sin conocimiento previo del problema.

Como ejemplos de métodos BCH podemos nombrar el algoritmo Ant-Class [9] que utiliza los principios exploratorios y estocásticos del enfoque ACO combinados con los principios determinísticos y heurísticos del K-means. El entorno simulado es una grilla bidimensional en la cual las hormigas recogen o depositan objetos en una parva de acuerdo a su similitud. El algoritmo Ant-Clust [8] inspirado en el reconocimiento químico de las hormigas para formar grupos o clusters diferenciados por sus respectivas propiedades químicas u olores. El algoritmo ACODF [12] un novedoso método que presenta una adaptación del enfoque ACO, para resolver la tarea de agrupación.

Uno de los métodos recientemente desarrollado se denomina AntTree y se basa en el comportamiento de auto-ensamblaje detectado en cierto tipo de hormigas. En este caso las hormigas construyen

una estructura "viviente" con sus cuerpos con el objeto de solucionar distintos problemas que deben afrontar. En el contexto de clustering este método realiza un proceso en donde las hormigas (que representan datos) se conectan entre sí de acuerdo a su similitud. De esta manera se forma una estructura de árbol, que determina una partición del conjunto de datos. Este algoritmo ha sido comparado previamente [2] con el método tradicional K-means y el algoritmo bio-inspirado AntClass, donde demuestra ser un método efectivo y prometedor para su investigación. En la siguiente sección detallaremos este método y posteriormente describiremos nuestra propuesta basada en el algoritmo AntTree y que hemos denominado *DAntTree*.

3. El algoritmo AntTree

El algoritmo AntTree [2] se basa en el comportamiento de auto-ensamblaje encontrado en ciertos tipos de hormigas[1], que se sujetan de manera incremental a un soporte fijo o a otras hormigas. Construye una estructura en forma de árbol cuya organización puede ser interpretada como un agrupamiento jerárquico o como un agrupamiento particionado. Cada hormiga es un dato del conjunto de datos y la forma en que se mueve por la estructura depende de su similitud con otras hormigas (datos). AntTree está fundamentado en una serie de características reales observadas en los comportamientos de auto-ensamblaje de las hormigas como por ejemplo la construcción de una estructura "viviente" a partir de un *soporte* fijo, su capacidad para moverse libremente por la estructura y otras[2].

Para particionar los datos, se construye un árbol cuyos nodos corresponden a hormigas y cada hormiga representa un dato del conjunto de datos a particionar. La tarea de clustering en este contexto se reduce a determinar la disposición de las conexiones (arcos del árbol) que sujetan las hormigas al soporte o a otra hormiga.

El supuesto básico para trabajar con AntTree es que los datos están expresados en algún lenguaje que permite definir una función de similitud Sim entre cualquier par de datos del conjunto bajo consideración. De esta manera, si N es la cantidad total de datos, y (d_i, d_j) es un par de datos arbitrario, $i \in [1, N]$, $j \in [1, N]$, el valor $Sim(i, j) \in [0, 1]$ representará el grado de similitud entre d_i y d_j. Un valor de 0 significa que d_i y d_j son completamente diferentes mientras que 1 significa que los datos son iguales.

Los principios generales del algoritmo son los siguientes: cada hormiga representa un nodo a ser conectado al árbol, es decir, un dato a ser clasificado. Partiendo de un *soporte* ficticio que denotaremos a_0, las hormigas gradualmente se conectarán al mismo y luego se conectarán a otras hormigas ya conectadas y así sucesivamente hasta que no queden hormigas sin conectar al árbol. La organización de la estructura resultante dependerá directamente de la medida de similitud definida por $Sim(i, j)$ y del vecindario local de las hormigas en movimiento.

A cada hormiga a_i asociaremos los siguientes términos:

1. La *conexión saliente* de a_i, que denotaremos $O(a_i)$ y que representa la conexión que a_i mantiene con el soporte u otra hormiga.

2. Las *conexiones entrantes* de a_i, que denotaremos $\mathcal{I}(a_i)$ y que representa el conjunto de conexiones que otras hormigas mantienen hacia a_i (es decir sus hijos).

3. El *dato* d_i representado por a_i.

4. Dos medidas denominadas el *umbral (threshold) de similitud* $T_{Sim}(a_i)$, y el *umbral de diferencia* $T_{Dissim}(a_i)$ que serán actualizadas durante el proceso de construcción del árbol.

[1] Como ejemplos podemos citar a la hormiga argentina *Linepithema humiles* y a la hormiga africana *Oecophylla longinoda*. En éstas últimas, el proceso de auto-ensamblaje se produce generalmente para cruzar espacios vacíos y para construir sus nidos.

[2] Ver [2] para una discusión más detallada de los fundamentos biológicos de AntTree.

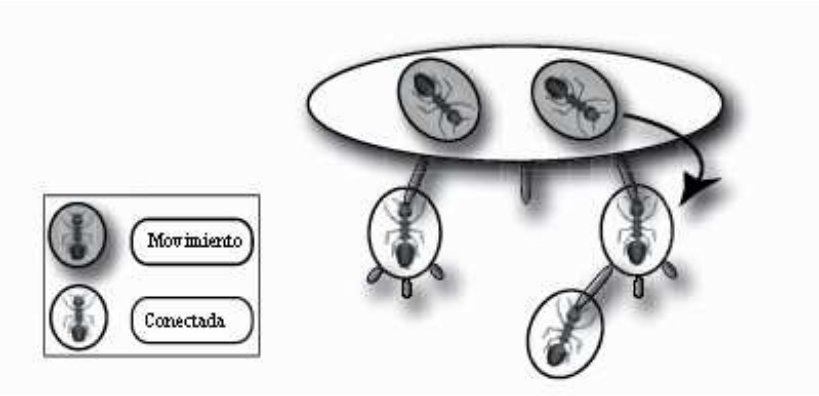

Figura 1: Proceso general de construcción del árbol con hormigas artificiales (adaptado de [2])

En la figura 1 se muestra el proceso general de ensamblaje de las hormigas artificiales. Se puede observar que cada cada hormiga a_i podrá encontrarse en una de las siguientes situaciones:

1. *Moviéndose en el árbol*: una hormiga a_i en movimiento (resaltadas con sombreado en la figura 1) puede encontrarse sobre el soporte del árbol (a_0), o sobre otra hormiga ya conectada (denotada por a_{pos}). En estos casos, a_i no se encuentra conectada a la estructura por lo que será libre de moverse a los vecinos conectados a a_0 o a a_{pos}. En la figura 2 se grafica el vecindario de una hormiga arbitraria a_{pos}.

2. *Conectada al árbol*: en este caso a_i ya tiene un valor asignado para $O(a_i)$ y no puede moverse más sobre la estructura. Por otra parte, una hormiga no podrá tener más de L_{max} conexiones entrantes ($|\mathcal{I}(a_i)| \leq L_{max}$). De esta manera podemos garantizar que la máxima apertura del árbol no será mayor que L_{max}.

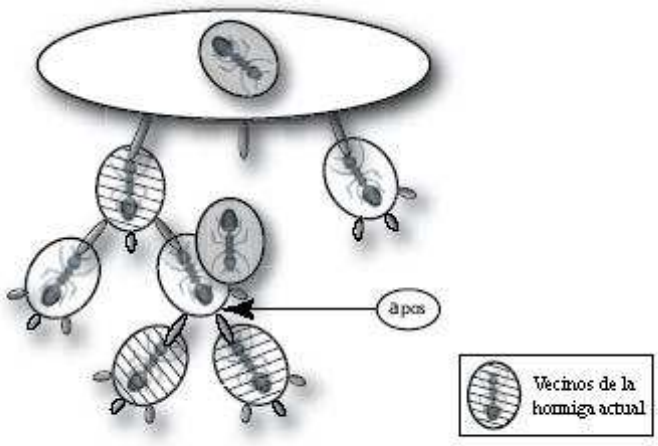

Figura 2: Vecindario de una hormiga arbitraria a_{pos} (adaptado de [2])

3.1. Detalles del algoritmo

El ciclo principal llevado a cabo por el algoritmo AntTree se muestra en la figura 3. Luego de que todas las hormigas son ubicadas sobre el soporte con sus umbrales de similitud y diferencia inicializados, en cada paso del ciclo una hormiga a_i es seleccionada de una lista de hormigas sin conectar $\mathcal{L}$. Más adelante realizaremos algunas consideraciones sobre el ordenamiento de esta lista. La hormiga a_i seleccionada podrá conectarse a otra hormiga (o al soporte) o moverse de acuerdo a su similitud con sus vecinos. Por lo tanto, mientras existan hormigas moviéndose sobre la estructura, se simulará la realización de ciertas acciones por parte de las hormigas que dependerán de su ubicación actual sobre el soporte (figura 4), o sobre otra hormiga (figura 5).

Sea $\mathcal{L}$ una lista (posiblemente ordenada) de hormigas sin conectar
Inicialización: Ubicar todas las hormigas sobre el soporte.
$T_{Sim}(a_j) \leftarrow 1$ y $T_{Dissim}(a_j) \leftarrow 0$, para toda hormiga a_j
Repetir
1. Escoger una hormiga a_i de la lista $\mathcal{L}$
2. Si a_i está sobre el soporte (a_0)
entonces *caso soporte* (ver figura 4)
sino *caso hormiga* (ver figura 5)
Hasta que todas las hormigas estén conectadas al árbol ($\mathcal{L}$ está vacía)

Figura 3: Algoritmo Principal del AntTree

Para el caso en que a_i se encuentra sobre el soporte, la situación más directa es cuando a_i es la primera hormiga, en cuyo caso se conecta directamente al soporte. En otro caso, a_i es comparada con a^+, la hormiga más similar a a_i entre todas las que están conectadas al soporte. Si son lo suficientemente similares (de acuerdo al umbral de similitud de a_i), entonces a_i se moverá por el subárbol correspondiente a a^+. En otro caso, se chequea si a^+ y a_i son lo suficientemente diferentes (de acuerdo al umbral de diferencia), en cuyo caso a_i es conectada al soporte, creando un nuevo subárbol dado que es lo suficientemente diferente de las demás hormigas conectadas al soporte. Finalmente, si a_i no es lo suficientemente similar ni lo suficientemente diferente, se actualizan ambos umbrales (diferencia y similitud) de a_i de la siguiente manera:

$$T_{Sim}(a_i) \longleftarrow T_{Sim}(a_i) * 0.9$$
$$T_{Dissim}(a_i) \longleftarrow T_{Dissim}(a_i) + 0.01$$

con el objeto de que a_i sea más "tolerante" la próxima vez que sea considerada y aumentar así su probabilidad de conectarse (o moverse).

El segundo caso es aquel en que a_i se encuentra sobre otra hormiga arbitraria a_{pos} (ver figura 5). Si a_i es *a*) lo suficientemente similar a a_{pos}, *b*) lo suficientemente diferente de las hormigas conectadas a a_{pos} y *c*) existe una conexión entrante disponible ($|\mathcal{I}(a_i)| < L_{max}$) entonces a_i es conectada a a_{pos}. En este caso a_i constituirá la raíz de un nuevo subárbol debajo de a_{pos}. La diferencia que existe con las otras hormigas conectadas a a_{pos} será tal que el nuevo sub-cluster estará bien separado de los demás, pero lo suficientemente parecido a a_{pos}. En caso contrario, a_i es movida de manera aleatoria sobre algún vecino de a_{pos} y sus umbrales son actualizados como en el caso anterior. El algoritmo finaliza cuando todas las hormigas están conectadas.

Antes de concluir la descripción del algoritmo cabe aclarar un punto respecto al orden inicial de las hormigas en la lista $\mathcal{L}$. Este paso influye significativamente en el algoritmo en general ya que define cual será la primer hormiga en conectarse al soporte. La estrategia que ha mostrado mejores

Si ninguna hormiga se encuentra conectada al soporte entonces conectar a_i a a_0
sino
Sea a^+ la hormiga conectada a a_0 más similar a a_i
(a) Si $Sim(a_i, a^+) \geq T_{Sim}(a_i)$ entonces mover a_i sobre a^+
(b) sino
i. Si $Sim(a_i, a^+) < T_{Dissim}(a_i)$ entonces /* a_i *es bastante diferente a* a^+ */
conectar a_i al soporte a_0 (si no quedan más conexiones de entrada en a_0 entonces mover a_i sobre a^+ y decrementar $T_{Sim}(a_i)$)
ii. sino decrementar $T_{Sim}(a_i)$ y aumentar $T_{Dissim}(a_i)$ /* a_i *es mas tolerante* */

Figura 4: Caso Soporte

resultados [2] es ordenar las hormigas de forma creciente en función de su similitud con las demás hormigas (en promedio). De esta forma la primer hormiga en conectarse al soporte es aquella que es más distinta a todas las demás y cercana a su propio grupo.

Debemos tener en cuenta que el árbol obtenido mediante esta técnica no es equivalente a los dendogramas encontrados por otras técnicas tradicionales de clustering. En nuestro caso, cada nodo corresponde a un dato del conjunto de datos, mientras que en los dendogramas sólo las hojas se corresponden a datos del conjunto. En este sentido, la interpretación del árbol obtenido por el algoritmo AntTree (ver figura 6) puede ser la de una partición del conjunto de datos, como cualquier algoritmo de agrupamiento particionado (considerando cada hormiga conectadas a a_0 como un grupo distinto). También puede ser interpretado como un árbol tipo dendograma, dejando que las hormigas que se encuentran en los nodos internos bajen hacia las hojas siguiendo los nodos más similares al mismo y de esta forma ser comparado con cualquier algoritmo de agrupamiento jerárquico.

Debemos resaltar que existen características muy interesantes de AntTree como puede ser el hecho de que evita mínimos locales gracias a su comportamiento probabilístico y produce resultados precisos sin conocimiento previo de la distribución de los datos (no es necesario conocer de antemano la cantidad de grupos existentes).

Sea a_{pos} la hormiga en que se encuentra ubicada a_i, y sea a_k un vecino aleatorio de a_{pos}
1. Si $Sim(a_i, a_{pos}) \geq T_{Sim}(a_i)$ entonces
Sea a^+ la hormiga conectada a a_{pos} más similar a a_i
(a) Si $Sim(a_i, a^+) \leq T_{Dissim}(a_i)$ entonces conectar a_i a a_{pos} /* *Si no quedan más conexiones entrantes entonces mover* a_i *hacia* a_k */
(b) sino decrementar $T_{Dissim}(a_i)$, incrementar $T_{Sim}(a_i)$ y mover a_i hacia a_k
2. sino mover a_i hacia a_k

Figura 5: Caso Hormiga

4. El algoritmo DAntTree

El método *AntTree dinámico* o *DAntTree* que se propone en esta sección puede ser considerado una extensión del algoritmo AntTree, que incluye como aspecto principal la posibilidad de que las hormigas se desconecten del árbol para reposicionarse en otro grupo (ramificación del árbol) más

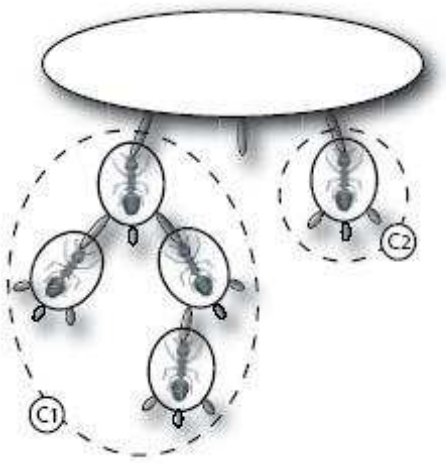

Figura 6: Interpretando el árbol como una partición sin jerarquía (adaptado de [2])

adecuado. Permite además explorar soluciones alternativas y quedarse con aquella que mejor se ajusta al conjunto de datos.

En DAntTree la obtención del modelo está dividido en 3 etapas principales: **1)** la **ejecución del AntTree** (con parámetros modificados), **2)** la **reubicación de hormigas** y **3)** la **unión de grupos**. Cada una de estas etapas es resumida en la figura 7.

1. Ejecución del algoritmo AntTree (con parámetros modificados).
2. Iterar hasta que todas las hormigas estén en un grupo adecuado
  a. Selección de las hormigas a desconectar del árbol
  b. Ordenamiento de la lista de hormigas (en forma decreciente)
  c. Reubicar las hormigas (AntTree con pre-asignación de la rama a seguir)
3. Iterar L_{max} veces o hasta que no se puedan realizar uniones
  a. Selección de 2 grupos a combinar
  b. Proceso de unión
  c. Evaluación del modelo

Figura 7: Algoritmo Principal del AntTree Dinámico

4.1. Primera etapa: Ejecución del AntTree

El primer paso del método es la ejecución del algoritmo básico AntTree con sus parámetros modificados de manera tal de obtener un número considerable de grupos pequeños y altamente acoplados (muy similares entre sí) que estén suficientemente distanciados de los demás grupos. Para lograr este efecto, sólo fue necesario modificar la actualización del umbral T_{Dissim} de cada hormiga de la siguiente manera[3]:

$$T_{Dissim}(a_i) \longleftarrow T_{Dissim}(a_i) + 0.2$$

También se incrementó la máxima apertura del arbol, llevándola a $L_{max} = 20$, y de esta manera poder obtener una mayor cantidad de grupos.

[3]La actualización de T_{Sim} se mantuvo como antes.

4.2. Segunda etapa: Reubicación de las hormigas

Este proceso iterativo desconecta las hormigas que no se adecúan al grupo al que pertenecen actualmente, y las reposiciona en otro grupo más adecuado. El proceso se realiza hasta que cada hormiga haya encontrado un grupo adecuado. Como se puede observar en la figura 7, esta etapa involucra los siguientes pasos: a) *selección* de las hormigas a desconectar, b) *ordenamiento* de las hormigas seleccionadas y c) la *reubicación* de las hormigas.

La **selección de las hormigas a desconectar** (paso a) consiste en un recorrido del árbol para determinar cuáles son las hormigas que deberían ser reubicadas en un lugar más adecuado. Para decidir si una hormiga se debía desconectar o no, se utilizó una función basada en el *método de Silhouette* [10] que determina el grado de pertenencia de una hormiga a su grupo actual, y que se define como:

$$s(i) = (b(i) - a(i))/max(a(i), b(i)) \tag{1}$$

En este caso, $a(i)$ es la distancia que existe entre la hormiga a_i con respecto a la media del grupo al que pertenece y $b(i)$ es la distancia mínima entre a_i y las medias de los restantes grupos. Los valores devueltos por esta función se encuentran en el rango $[-1, 1]$. En este caso, se definió un umbral $t = -0{,}2$ para decidir si una cierta hormiga debía separarse de su estructura o no. El objetivo de este umbral es separar a aquellas hormigas que *realmente están en un grupo equivocado*. De esta forma, si $s(i) < t$, podemos decir que hemos encontrado un grupo "mejor"[4] para la hormiga a_i y por lo tanto debe ser seleccionada para su desconexión.

Si $\mathcal{L}$ es la lista de hormigas seleccionadas para desconectar, otro paso importante es el **ordenamiento de las hormigas seleccionadas** $\mathcal{L}$ de acuerdo a su similitud pero en *orden decreciente* (paso b). Observar que en este caso la lista $\mathcal{L}$ resultante, estará ordenada siguiendo el criterio opuesto al adoptado por AntTree, ubicando a la hormiga más similar a las demás *primero*. La razón para ello, es que a consecuencia del proceso de selección previo, algunos grupos se quedarán sin hormigas y desaparecerán. Por lo tanto, las hormigas que han sido asignadas a un grupo desaparecido deberán ser reasignadas a otro grupo. Este proceso de reasignación consiste simplemente en asignar el grupo cuya media se encuentre más cercana a la hormiga a reasignar. Por ello se ordena la lista en forma decreciente con el objeto de que la hormiga más similar de la lista influya en la media del grupo que se le ha asignado y las hormigas similares a ella tomen el mismo grupo. Si hubiéramos ordenado la lista de forma creciente, la hormiga más lejana a las demás se hubiera posicionado en primer lugar, influyendo de manera considerable en la media del grupo asignado y perjudicando a las demás reasignaciones.

Con respecto a la **reubicación de las hormigas** (paso c), este paso involucra dos procesos. El primero es la **verificación de los grupos preasignados**, con el objeto de chequear si alguno de estos grupos ha quedado vacío. Si así fuera, entonces se le asigna un nuevo grupo de los que todavía existen. El segundo paso es el de **reasignar la hormiga** a su nuevo grupo. Esto se realiza moviendo la hormiga a la primera de su grupo conectada al soporte. Luego repetimos los mismos pasos que en el algoritmo básico del AntTree, con el objeto de que se mantengan las distintas interpretaciones de la estructura.

4.3. Tercera etapa: Unión de grupos

En esta última etapa se realiza un proceso de unión entre grupos con el objetivo de encontrar los verdaderos grupos existentes dentro de los datos. La idea general consiste en combinar aquellos grupos muy similares y evaluar el modelo obtenido, retornando aquel que mejor se haya adaptado al conjunto de datos analizados.

[4] Un grupo cuya media se encuentra más cercana a la hormiga a_i que la media del grupo al que pertenece actualmente.

Para seleccionar los 2 grupos a combinar, se toman los más similares de todos los existentes en la estructura, de acuerdo a la similitud de sus medias. El proceso de unión es similar al paso de reacomodar hormigas de la etapa anterior. Se unen las hormigas de uno de los grupos seleccionados al otro grupo seleccionado. Cabe destacar que en este caso no existe un ordenamiento previo de las hormigas, ya que todas han sido asignadas al mismo grupo. Por cada iteración, se evalúa el modelo obtenido con el mejor encontrado hasta ese momento. Si el nuevo modelo es más preciso que el mejor encontrado, entonces se almacena para devolver como resultado. La cantidad de iteraciones depende de la cantidad de grupos encontrados y no supera las L_{max} iteraciones, independientemente del tamaño de nuestro conjunto de datos.

Un aspecto que aún no hemos explicado, es la forma en que se evalúa cada nuevo modelo, para determinar si esta nueva organización de los datos es más adecuada que el mejor modelo obtenido hasta el momento. Para ello utilizamos una función basada en el *índice Davies-Boulding* [7] que se define tomando en cuenta una medida de similitud R_{ij} entre 2 grupos C_i y C_j. La medida R_{ij} se define en base a una medida de dispersión dentro de cada grupo y un factor de similitud entre ambos grupos. R_{ij} debe ser positíva y simétrica [5], y una simple elección que satisface estas condiciones es:

$$R_{ij} = (s_i + s_j)/d_{ij} \tag{2}$$

donde la medida de dispersión s_i se define como la máxima distancia que existe entre la media del grupo C_i con un elemento $d_m \in C_i$ y d_{ij} es la distancia que existe entre la media del grupo C_i con la media del grupo C_j. Finalmente, si n_c es el número de clusters, se define el índice Davies-Boulding (DB) como:

$$DB_{n_c} = \frac{1}{n_c} \sum_{i=1}^{n_c} R_i \tag{3}$$

con $R_i = \underset{j=1...k,\, i \neq j}{\text{máx}} R_{ij}$. Esta función define, intuitivamente, la similitud promedio entre cada grupo con su más parecido. Uno desearía encontrar una agrupación en donde cada grupo sea lo más diferente posible con los demás. Por ello, mientras menor sea el valor devuelto por esta función, mejor es nuestro modelo.

5. Resultados experimentales

Para llevar a cabo los experimentos se implementaron los algoritmos AntTree y DAntTree en el lenguaje JAVA y se utilizó el paquete WEKA [13], que provee utilidades específicas para clustering y permite una fácil realización del estudio comparativo propuesto.

Como función de similitud, se utilizó la *función de similitud de Gower*:

$$Sim(i,j) = \frac{\sum_{k=1}^{p} w_{ijk} s_{ijk}}{\sum_{k=1}^{p} w_{ijk}} \tag{4}$$

donde s_{ijk} es la similitud que existe entre el elemento i y el j para el k-ésimo atributo. Cuando los atributos son categóricos, se comparan por igualdad (s_{ijk} = 1 si son iguales y 0 en caso contrario). Para datos continuos, s_{ijk} se define como $s_{ijk} = 1 - |x_{ik} - x_{jk}|/R_k$ donde R_k es el rango de observaciones del k-ésimo atributo. El término w_{ijk} será 1 o 0, dependiendo de si la comparación se considera válida o no. Por ejemplo, cuando falta el valor de uno de los 2 atributos, w_{ijk} será igual a 0.

Para comparar los algoritmos se utilizaron cuatro bases de datos reales: *breast-cancer-wisconsin* (bcw), *eucalyptus* (euca), *heart-disease* (heart) y *thyroid-disease* (thyroid), obtenidas del repositorio del UCI [6]. También se generaron tres bases de datos artificiales: *art1*, *art2* y *art3* que varían en el

[5] Se debe cumplir que a) $R_{ij} \geq 0$, b) $R_{ij} = R_{ji}$ y c) si $s_i = 0$ y $s_j = 0$ entonces $R_{ij} = 0$

[6] Repositorio de bases de datos para Aprendizaje de Máquina de la Universidad de California en Irvine (UCI) [3].

grado de proximidad que tienen los grupos dentro de los datos: *art1* grupos bien separados, *art2* clara separación pero menor que en *art1* y *art3* con grupos muy cercanos.

Las principales características de estas bases de datos se resumen en la tabla 1, especificándose para cada una de ellas el número de instancias (*NI*), número de atributos (*NA*), tipo de los atributos (*TA*, categóricos o numéricos), número de registros con valores faltantes (*VF*), número de clases (K) y distribución de las instancias para cada una de las clases (*DI*).

Base de Datos	*NI*	*NA*	*TA*	*VF*	K	*DI*
bcw	699	9	cat.	16	2	$\langle 458, 241 \rangle$
euca	736	19	cat. y num.	141	5	$\langle 180, 107, 130, 214, 105 \rangle$
heart	303	13	cat. y num.	0	5	$\langle 164, 55, 36, 35, 13 \rangle$
thyroid	1505	5	num.	0	3	$\langle 1050, 245, 210 \rangle$
art_i, $i = 1, 2, 3$	1002	31	num.	0	3	$\langle 334, 334, 334 \rangle$

Tabla 1: Características de las bases de datos utilizadas en la experimentación

Para todas las bases de datos presentadas previamente, se conoce el cluster al que pertenece cada instancia [7]. Por lo tanto, se pueden evaluar los distintos algoritmos respecto a sus clases verdaderas. Una medida de este tipo, es el **error de clasificación** Ec utilizado en [2]. En este caso, asumamos que k_i es la clase verdadera del objeto d_i, k'_i es la clase encontrada por los métodos para el objeto d_i. K es la cantidad de grupos verdaderos y K' es la cantidad de grupos encontrados por los métodos. Podemos entonces definir a Ec como:

$$Ec = \frac{2}{N(N-1)} \sum_{(i,j) \in \{1,.....N\}^2,\ i<j} \epsilon_{ij} \tag{5}$$

donde:

$$\epsilon_{ij} = \begin{cases} 0 & \text{si } (k_i = k_j \wedge k'_i = k'_j) \vee (k_i \neq k_j \wedge k'_i \neq k'_j), \\ 1 & \text{en otro caso.} \end{cases}$$

Debemos notar que el valor de K (número real de clusters) debe ser especificado como parámetro para el caso del método K-means utilizado en los experimentos. Los restantes algoritmos desconocen este valor.

En la tabla 2 se muestran los resultados comparativos obtenidos con K-means, AntTree y DAntTree. Se incluye en la misma el error de clasificación Ec definido de acuerdo a la ecuación 5, el número de clusters encontrados K' y la función de Davies-Boulding (DB) correspondiente a la ecuación 3. También se muestra la función *Global Silhouette promedio* definida en base a la ecuación 1 como: $GS = (\sum_{i=1}^{N} s(i))/N$. Debemos destacar que en todos los casos, el valor K' del método K-means corresponde al valor de K que es especificado como parámetro.

Del análisis de estos resultados podemos destacar los siguientes puntos:

- Respecto al error de clasificación Ec, DAntTree produce mejores resultados que AntTree para todas las bases de datos. En particular, DAntTree no es afectado tan severamente como AntTree cuando los límites de los clusters no están claramente delimitados (observar las bases de datos euca, heart, $art2$ y $art3$). En general, los resultados obtenidos con DAntTree son comparables a los de K-means y en algunos casos mejor como es el caso de las bases de datos euca, $art1$ y $art3$.

- Para distribuciones de datos en donde existe un limite demarcado entre grupos (como es el caso de las bases de datos artificiales), el método DAntTree encuentra soluciones óptimas.

[7] Obviamente esta información no es utilizada por los algoritmos.

Base de Datos	Algoritmo	Ec	K'	DB	GS
bcw	K-mean	0.04	2	1.89	0.58
	AntTree	0.21	10	4.85	0.4
	DAntTree	0.11	2	1.56	0.47
euca	K-mean	0.31	5	2.38	0.27
	AntTree	0.78	1	0	0
	DAntTree	0.27	13	1.02	0.64
heart	K-mean	0.33	5	2.38	0.24
	AntTree	0.65	1	0	0
	DAntTree	0.37	2	1.87	0.29
thyroid	K-mean	0	3	1.85	0.62
	AntTree	0.47	1	0	0
	DAntTree	0.31	2	1.63	0.68
art_1	K-mean	0. 28	3	3.37	0.41
	AntTree	0. 22	2	0.48	0.72
	DAntTree	0	3	0. 31	0.87
art_2	K-mean	0	3	0.42	0.83
	AntTree	0. 67	1	0	0
	DAntTree	0	3	0.42	0.83
art_3	K-mean	0.28	3	4.07	0.1
	AntTree	0. 67	1	0	0
	DAntTree	0	3	0. 38	0.83

Tabla 2: Resultados obtenidos con las distintas bases de datos

- DAntTree tiende a encontrar un número de clusters K' más cercano al número real de clusters K que en el caso del AntTree. En particular, para 4 de las 7 bases de datos, el valor de K' para DAntTree coincidió con el de K.

- Con respecto a las medidas DB y GS que intentan estimar la calidad del clustering, debemos recordar que valores pequeños de DB y altos de GS son deseables. En este sentido, el método propuesto ha encontrado modelos con valores mínimos de DB y máximos de GS en relación a los demás métodos en comparación (AntTree y K-means), para la mayoría de las bases de datos consideradas. Si consideramos la medida DB por ejemplo, DAntTree arroja mejores resultados que K-means en todos los casos y que AntTree cuando este último arroja resultados no nulos. Con respecto a la medida GS, DAntTree es superior a AntTree en todos los casos, y supera a K-means en todas las bases de datos excepto bcw.

- Los resultados obtenidos con DAntTree son comparables e incluso mejores para algunas de las métricas utilizadas en comparación a los obtenidos por K-means. Este aspecto no deja de ser relevante si se considera que K-means se ve beneficiado de conocer a priori el número exacto de clusters reales y que tanto AntTree como DAntTree no requieren de esta información.

Debemos notar que los tiempos de ejecución que pudimos observar del método propuesto no difieren en general en forma muy significativa de los tiempos de ejecución de su predecesor, el algoritmo AntTree. Sin embargo, este aspecto requiere de un estudio más detallado utilizando otras bases de datos con grandes volúmenes de información.

6. Conclusiones y trabajo futuro

En este trabajo se presentó un nuevo algoritmo de clustering basado en el método AntTree, un enfoque bio-inspirado cuyos fundamentos se encuentran en el comportamiento de auto-ensamblaje observado en ciertas clases de hormigas. El algoritmo propuesto utiliza al AntTree básico con sus parámetros de tolerancia por similitud y diferencia modificados, posteriormente reacomoda las hormigas y finalmente une los grupos resultantes seleccionando el mejor modelo obtenido.

La principal conclusión obtenida mediante este nuevo método es que, dada la capacidad de desconectarse de las hormigas de la estructura, hemos obtenido resultados más precisos que el algoritmo AntTree y comparables con otros métodos que requieren de mayor información previa.

También se ha mantenido una característica fundamental del AntTree ya que podemos interpretar de distintas formas el modelo obtenido, y de esta manera visualizarlo como un agrupamiento particionado o un agrupamiento jerárquico. Este nuevo método tiene la propiedad fundamental de adaptarse a distintas distribuciones de los datos, dado que partimos de pequeños grupos y realizamos un proceso de unión en base a la similitud entre los mismos.

Posibles extensiones a este trabajo incluyen la experimentación del algoritmo DAntTree con bases de datos con mayores volúmenes de información, incluyendo bases de datos documentales.

Referencias

[1] Hussein Abbass, Charles Newton, and Ruhul Sarker. *Data Mining: A Heuristic Approach*. Idea Group Publishing, Hershey, PA, USA, 2002.

[2] H. Azzag, N. Monmarche, M. Slimane, G. Venturini, and C. Guinot. Anttree: A new model for clustering with artificial ants. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 2642–2647, Canberra, 8-12 December 2003. IEEE Press.

[3] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. *University of California, Irvine, Dept. of Information and Computer Sciences*, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[4] Morven Leese Brian Everitt, Sabine Landau. *Cluster analysis*. Halsted Press, Institute of Psychiatry, Kings College, 2001.

[5] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolutionary Computation*, 1(1):53–66, 1997.

[6] Alex A. Freitas. A survey of evolutionary algorithms for data mining and knowledge discovery. In A Ghosh and S Tsutsui, editors, *Advances in Evolutionary Computation*, pages 819–845. Springer-Verlag, August 2002.

[7] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. Clustering validity checking methods: Part II. *SIGMOD Record*, 31(3):19–27, 2002.

[8] N. Labroche, N. Monmarché, and G. Venturini. AntClust: Ant Clustering and Web Usage Mining. In *Genetic and Evolutionary Computation Conference*, pages 25–36, Chicago, 2003.

[9] N. "M. Slimane, N. Monmarché, and G."Venturini. Antclass: discovery of clusters in numeric data by an hybridization of an ant colony with kmeans algorithm. Rapport interne 213, Laboratoire d Ínformatique de l Úniversit e de Tours, E3i Tours,, 1999.

[10] F. Azuaje N. Bolshakova. Improving expression data mining through cluster validation. 2003.

[11] V.K. Jayaraman P.S. Shelokar and B.D. Kulkarni. An ant colony approach for clustering. Technical report, Chemical Engineering and Process Division, National Chemical Laboratory, India, 2003.

[12] Cheng-Fa Tsai, Chun-Wei Tsai, Han-Chang Wu, and Tzer Yang. Acodf: a novel data clustering approach for data mining in large databases. *J. Syst. Softw.*, 73(1):133–145, 2004.

[13] Ian H. Witten and Eibe Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.