Algoritmo de Clustering On-Line Utilizando Metaheurísticas y Técnicas de Muestreo

Arantza Casillas
Dpto. Electricidad y Electrónica,
Universidad del País Vasco
Apto. 664, Bilbao, País Vasco 48940
arantza@we.lc.ehu.es

María Teresa González de Lena
Dpto. Informática, Estadística y Telemática,
Universidad Rey Juan Carlos
C/ Tulipán s/n, Móstoles, Madrid, 28933
m.t.gonzalez@escet.urjc.es

Raquel Martínez
Dpto. Informática, Estadística y Telemática,
Universidad Rey Juan Carlos
C/ Tulipán s/n, Móstoles, Madrid, 28933
r.martinez@escet.urjc.es

Resumen: El clustering de un conjunto de documentos consiste en dividirlo en conjuntos disjuntos de clusters (subconjuntos), tales que los documentos pertenecientes al mismo cluster sean "similares" entre sí y sean menos "similares" a los pertenecientes a los demás clusters. En determinadas condiciones el clustering es una tarea computacionalmente muy costosa, verbigracia: trabajar con una colección extensa de documentos sin conocer a priori el número de clusters en los que se agruparán. Si, además, el contexto en el que se va a realizar el clustering requiere una solución en un tiempo que no supere unos pocos segundos, los métodos convencionales de cálculo de un valor óptimo para el número de clusters resultan inadecuados. En este artículo se propone un algoritmo para realizar el clustering de un conjunto de documentos, sin conocer a priori el número de clusters. El énfasis se ha puesto en la reducción del tiempo de cálculo, por lo que podemos afirmar que nuestro algoritmo es capaz de realizar un clustering on-line. Las técnicas utilizadas combinan el uso de una regla de parada global, algoritmos genéticos, técnicas de muestreo estadístico y un algoritmo de clustering clásico.
Palabras clave: clustering de documentos, algoritmos genéticos, clustering on-line.

Abstract: Document clustering involves dividing a set of documents into separate clusters (subsets), so that the documents are similar to other documents in the same cluster, and less similars or different from documents in other clusters. In certain conditions the clustering is a computational expensive task, for example: working with a huge collection of documents without prior knowlegdge of the appropriate number of clusters. In addition, if it is necessary a solution in few seconds, the conventional methods of calculation of the optimum number of clusters are unacceptable. In this paper we propose an algorithm for clustering a set of documents, without prior knowlegdge of the appropriate number of clusters. The emphasis has been done in the reduction of the calculation time, reason why we be able to say that our algorithm can achieve a clustering on-line. Our algorithm combines the use of a global stopping rule, genetic algorithms, techniques of statistical sampling and one classic algorithm of clustering.
Key words: clustering of documents, genetic algorithms, clustering on-line.

1 Introducción

En determinados contextos no sólo es necesario realizar el clustering de un conjunto de documentos sin conocer a priori el número de clusters, si no que, además, hay que realizarlo en un tiempo que no debe superar unos pocos segundos. El clustering de los documentos que devuelve un motor de búsqueda sería uno de dichos contextos. A la usual lista de relevancia, se podría añadir como salida del motor de búsqueda la información correspondiente al resultado del clustering de los documentos.

Hay un aspecto que resulta crítico en el clustering on-line de documentos, tal y como se está planteando: el desconocimiento a priori del número óptimo de clusters, *k*, en los que se organizará la colección de documentos. Decimos que es crítico porque dicho cálculo lleva asociado un elevado coste computacional.

Aunque existen diversos métodos para calcular el número óptimo de clusters, en este trabajo nos hemos centrado en una regla de detención global (*global stopping rule*). Las reglas de detención globales evaluan la bondad de una medida *C(k)* de una partición de objetos en *k* clusters, y además tratan de hallar el valor de *k* para el cual *C(k)* es óptimo. Esta medida, *C(k)*, se basa normalmente en la semejanza o distancia inter-cluster y entre-clusters. Un inconveniente que poseen estos métodos es la carencia de una definición natural para *C(1)* (Gordon, 1999), ya que se supone que el conjunto inicial siempre se va a dividir. Otro gran inconveniente es el alto coste computacional que conlleva calcular el valor óptimo de *k*, aunque el número de documentos implicados sea moderado. Con nuestro algoritmo damos una solución a ambos problemas.

Milligan y Cooper (Milligan y Cooper, 1985) realizaron un estudio comparativo de 30 reglas de detención global, llevando a cabo experimentos con datos artificiales que podían agruparse en conjuntos entre los que no existía solapamiento. Aunque el resultado de los experimentos depende de la estrategia seguida para generar los clusters, el estudio fue muy útil para identificar qué reglas de detención proporcionaban mejores resultados. Una de las reglas que mejor respondió a los experimentos fue la de Calinski y Harabasz. Esta es la regla de detención que nosotros hemos seleccionado como punto de partida.

La regla de Calinski y Harabasz (Calinski y Harabasz, 1974), dado un conjunto de objetos, calcula un indicador informal del mejor número de clusters. Sobre esta regla, hemos diseñado e implementado un algoritmo genético que encuentra un valor próximo al óptimo número de clusters *k* con un coste computacional menor que con la regla de Calinski y Harabasz en la mayoría de los casos. Además, el uso de técnicas de muestreo estadístico permite contemplar el caso *C(1)* (el conjunto inicial no se divide en clusters).

Existen trabajos previos que emplean algoritmos genéticos para realizar el proceso de clustering. En muchos de ellos se asume que el valor de *k* es conocido (Estivill-Castro y Murray, 1998), (Chu, Roddick y Pan, 2002), (Murthy y Chowdhury, 1996), (Merz y Zell, 2002), (Lucasius, Dane y Kateman, 1993). Sin embargo en (Sarkar, Yegnanarayana y Khemani, 1997) y (Imai, Kaimura y Hata, 2000) se proponen algoritmos genéticos para lograr el clustering en un número óptimo de clusters desconocido. El primer trabajo se basa en el conocido algoritmo *K-means*. En el segundo trabajo parten de métodos empleados para el clustering borroso (con solapamiento) y, haciendo suposiciones sobre la distribución de los elementos dentro de cada cluster, consiguen diseñar un método para obtener un número de clusters óptimo. En este segundo trabajo, trasladan el problema del desconocimiento de *k* al problema del desconocimiento de la varianza de la distribución de los elementos dentro del cluster. Además, en ambos trabajos, el tiempo necesario para el cálculo resulta muy elevado para plantear un clustering on-line.

La estructura del artículo es la siguiente: en la sección 2 se describe brevemente la regla de Calinski y Harabasz; la sección 3 presenta el algoritmo genético que se ha diseñado para calcular un número de clusters óptimo; en la sección 4 se expone la técnica de muestreo estadístico empleada; la sección 5 describe el algoritmo que se propone; los resultados experimentales se exponen en la sección 6; por último, en la sección 7 presentamos las conclusiones de nuestro trabajo.

2 Algoritmo de Calinski

El algoritmo de Calinski y Harabasz (Calinski y Harabasz, 1974), proporciona un resultado de clustering a partir de una colección de objetos sin necesidad de que le aportemos el valor de *k*,

es decir, el propio algoritmo decide cuál es el valor de *k* más apropiado y la composición de cada uno de los clusters. Este método parte de una representación de los objetos, es decir, debemos preprocesarlos extrayendo los *n*-rasgos más relevantes y creando un vector *n*-dimensional que represente esta información. Aplicando la distancia euclídea calcularemos las distancias existentes entre los vectores. De este modo se calcula la matriz de distancias, que tendrá un tamaño *num*num*, donde *num,* en nuestro caso, es el número de documentos sobre los que vamos a aplicar el algoritmo. El algoritmo comienza calculando el Árbol de Recubrimiento Mínimo (*Minimal Spannig Tree, MST*). A continuación comienza a explorar todos los posibles clusters que se pueden formar a partir de él. Para ello comenzará a eliminar aristas del MST, consiguiendo así construir los diversos clusters. Si queremos agrupar *num* documentos en *k* clusters será necesario eliminar *k*-1 aristas del MST. Para cada posible solución debe calcular la suma de las distancias entre elementos de un mismo cluster, llamada distacia inter-cluster (*WGSS*). Para obtener la mejor partición posible es necesario realizar una búsqueda exhaustiva en todo el espacio solución, es decir, probará todas las posibilidades para $k = 2$, quedándose con la opción que obtenga el menor *WGSS*, y realizará la misma operación para todos los valores de *k* posibles: 2,3,... *k*-1. De todas las posibles particiones dado un valor de *k se* escogerá la de menor valor de *WGSS* y se evaluará la función criterio, fórmula (1):

$$VRC = \frac{\frac{BGSS}{k-1}}{\frac{WGSS}{n-k}} \qquad (1),$$

donde *BGSS* es la suma total de distancias entre clusters.

Los autores sugieren la utilización de la fórmula (1) como un indicador informal para la obtención del mejor valor para *k*. Además sugieren una posible mejora computacional del algoritmo, que consiste en ir evaluando sucesivamente valores de *k* crecientes hasta encontrar un máximo local.

Aunque trabajemos con el MST, reduciendo enormemente el espacio solución, las posibilidades que el algoritmo debe explorar, para cada valor de *k*, calculadas mediante la fórmula combinatoria (2):

$$\binom{num-1}{k-1} = \frac{(num-1)!}{(k-1)!*(num-k)!} \qquad (2),$$

son muy elevadas incluso para pequeñas colecciones de documentos. Por ello no es viable utilizar esta regla cuando se espera una respuesta rápida y se trabaja con un número de documentos mayor que unas decenas.

3 Algoritmo Genético

Los algoritmos genéticos son heurísticas que nos ayudan a abordar problemas de búsqueda exhaustiva que por vía analítica son inabordables, debido fundamentalmente al coste computacional que ello conllevaría.

Holland (Holland, 1975) fue el primero en denominar a este tipo de heurísticas algoritmos genéticos, ya que de alguna forma tratan de emular el comportamiento de los seres vivos. A continuación se resume el comportamiento general de un algoritmo genético: se parte de una población inicial aleatoria de soluciones al problema. En los primeros problemas resueltos con este tipo de algortimo se utilizaba un vector binario, compuesto por subvectores binarios o genes, para codificar las soluciones y trabajar de este modo más rápidamente. Actualmente existen otros tipos de codificaciones, de hecho, para solucionar un problema concreto se debe estudiar qué representación es la más adecuada (Michalewicz, 1996). Dicha población es evaluada gracias a una función objetivo que hay que optimizar. A continuación se seleccionan pares de individuos. Aquellos que tengan una mejor adaptación al medio, es decir, un valor mejor de la función de aptitud (función objetivo), tendrán más posibilidades de ser escogidos. A partir de estos pares de individuos se generarán pares de hijos. Una vez generados estos nuevos individuos se verán sometidos, con una probabilidad baja, a una mutación, de tal forma que alguna de sus carácterísticas pueda verse modificada. El motivo principal de emplear un operador de mutación, es poder asegurar que todo el espacio de soluciones tiene una probabilidad distinta de cero de ser evaluado. En algunos casos se emplea la condición de elitismo, que consiste en realizar una copia del mejor individuo de la población anterior y transferirlo directamente a la nueva para evitar que dicha solución se pierda. Llegados a este punto se elimina la población anterior y se evalua la nueva. Este proceso se

repite hasta que se satisface la condición de finalización. Esta condición de finalización en muchos casos suele estar compuesta por dos condiciones: la primera, es que la población converja hacia la solución óptima; y la segunda, es un número máximo de iteraciones o generaciones. Desafortunadamente no existe ninguna condición de terminación que asegure la convergencia del algoritmo (Michalewicz, 1996), (Goldberg, 2002).

A continuación expondremos brevemente las características concretas del algoritmo genético que hemos diseñado.

3.1 Codificación del individuo

La forma elegida para representar a cada individuo se ha visto influenciada directamente por el algoritmo de Calinski y Harabasz. Necesitábamos una codificación que nos permitiera representar un número de clusters variable, y también que nos facilitara información sobre qué aristas serían eliminadas del árbol de recubrimiento mínimo. Por todo ello escogimos la representación clásica de vector binario. La longitud del vector vendrá determinado por el número de aristas del árbol de recubrimiento mínimo. Un bit a 1 nos indica que esa arista debe ser eliminada del árbol, un bit a 0 por el contrario, mantiene esa arista. La suma de los bit a 1 dentro del vector será igual a k-1.

3.2 Población inicial

El espacio de soluciones de nuestro problema no es equiprobable ya que existen más casos que estudiar para valores de k tales que:

$$k - 1 = \frac{num - 1}{2} \qquad (3),$$

siendo *num* el número de documentos totales. Por ello decidimos que este hecho se viese refejado en nuestra población inicial. Para obtener el número de posibilidades de cada k debemos emplear la fórmula (2). Cuando el número de documentos sea elevado, se puede producir un error de desbordamiento al intentar calcular dicho valor, sobre todo para k próximos al valor medio, fórmula (3). Así, en lugar de emplear los números combinatorios, fórmula (2), para el cálculo de posibilidades, utilizamos la distribución de Laplace como aproximación. Fórmula (4):

$$f(k) = \frac{\lambda}{2} * e^{-\lambda * |k-\alpha|} \qquad (4),$$
$$-\infty < \alpha < \infty, \lambda > 0$$

3.3 Función de Aptitud

La función de aptitud que hemos empleado para evaluar nuestra población, es la que nos proporciona el algoritmo de Calinski y Harabasz, fórmula (1). De tal forma que un individuo que posea un valor más elevado de *VRC*, es considerado por el algoritmo un individuo mejor adaptado y, por lo tanto, con más posibilidades de reproducirse.

3.4 Operador de selección

El operador de selección será el encargado de escoger los individuos que se reproducirán, es decir, aquellos que generarán los individuos de la próxima población. Para ello, se basará en el valor de la función de aptitud para cada individuo de la población actual; de tal forma que individuos con un mejor valor de aptitud tendrán más posibilidades de reproducirse.

3.5 Operador de Cruce

En los algoritmos genéticos clásicos, el operador de cruce intercambia información de los padres a partir de un único punto de cruce, escogido aleatoriamente (Goldberg, 2002). Nosotros escogimos un número de puntos de cruce aleatorios, ya que al realizar las pruebas experimentales comprobamos que mejoraba la efectividad del algoritmo.

3.6 Operador de mutación

Una vez han sido generados los individuos de la nueva generación, son sometidos a mutación antes de ser evaluados. La posibilidad de que se produzcan mutaciones es, en general, muy baja. Sin embargo, suponemos que al comienzo del algoritmo es beneficioso que la población sea muy variada, lo cual implica una probabilidad de mutación alta. A medida que avance el algoritmo la probabilidad de que se produzca una mutación debe ir disminuyendo, ya que la población debe converger hacia la solución. Por ello introdujimos una probabilidad de mutación variable, aunque para una misma generación se mantenga constante.

3.7 Elitismo

Debido a que la colección de documentos puede ser numerosa, para lograr la convergencia del algoritmo en un tiempo reducido, incluímos la condición de elitismo cuando el número de documentos es superior a 140 (valor determinado experimentalmente).

3.8 Condición de finalización

Hemos implementado tres condiciones de finalización. La primera es el criterio de convergencia: comprueba si el mejor individuo no cambia en tres generaciones sucesivas. La segunda condición limita el número de generaciones del algoritmo, en nuestro caso ha sido limitado a *num*/2. La tercera, comprueba el tiempo de ejecución, de tal forma que no permite continuar con la ejecución si ésta ha rebasado un tiempo límite (este tiempo es del orden de 10 segundos).

4 Muestreo estadístico

El cálculo de un *k* óptimo con todos los documentos puede resultar innecesario si es posible seleccionar un subconjunto de ellos que representen al conjunto.

El muestreo estadístico se emplea para obtener conclusiones de una población sin necesidad de recabar información sobre el total de la población, sino sobre una muestra representativa de ésta. La forma de elegir dicha muestra y el tamaño que debe tener son dos factores primordiales para que las conclusiones que se extraigan de la muestra puedan ser aplicadas al total de la población.

Podemos conocer el tamaño que tendrá la muestra (Pérez, 1999) fijando el error de muestreo. El error de muestreo se define como la raíz cuadrada de la varianza, fórmula (5):

$$e(x) = \sqrt{\sigma(x)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2} \qquad (5),$$

N = número total de individuos que componen la población.

x_i = valor de la variable x para el elemento i.

$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$, la media.

Una vez fijado el error de muestreo podemos conocer el tamaño que tendrá la muestra aplicando la fórmula (6):

$$n = \frac{N * S^2}{N * e^2 + S^2} \qquad (6),$$

n = tamaño de la muestra.

N = número total de individuos que componen la población.

e = error de muestreo (calculado con la fórmula 5).

S = cuasivarianza = $\sqrt{\frac{1}{(N-1)}\sum_{i=1}^{N}\left(x_i - \bar{x}\right)^2}$

En nuestro problema la población total es una matriz de rasgos en la que cada fila representa a un documento en forma vectorial. Como cada uno de los rasgos de los documentos es relevante, realizamos el cálculo del error de muestreo sobre cada uno de estos rasgos, seleccionando el de menor valor de todos ellos, ya que como puede apreciarse en la fórmula (6), el tamaño de la muestra es inversamente proporcional al cuadrado del error de muestreo, es decir, para cometer un menor error de muestreo es necesario aumentar el tamaño de la muestra. De esta forma aseguramos que la muestra escogida sea representativa para todos los rasgos de los documentos.

Una vez conocido el tamaño que debe tener la muestra para que pueda representar a toda la población, escogemos de forma aleatoria y sin reposición (no se permiten repeticiones de documentos) una muestra de ese tamaño sobre la población.

Como el muestreo estadístico nos indica el número de documentos que son representativos de toda la población, si el tamaño de la muestra es 1, es decir, con un único documento se podría representar a toda la población, consideramos que no se debe partir la colección de documentos.

Una vez seleccionada la muestra, el cálculo de un valor para *k* se realiza solamente teniendo en cuenta dicha muestra, reduciéndose considerablemente el coste computacional asociado. Despues, ya conocido el valor de *k*, se puede calcular el clustering en *k* clusters por cualquier método clasico.

5 Algoritmo de Clustering On-line

Las primeras pruebas experimentales corroboraron que el algoritmo de Calinski y Harabasz sólo podía dar una respuesta adecuada

al clustering on-line, en cuanto al tiempo se refiere, cuando el conjunto de documentos era pequeño, del orden de 15 documentos; o cuando el número de clusters final, *k*, era pequeño (dos o tres), con independencia del número de documentos (Casillas, González de Lena y Martínez, 2003). También pudimos comprobar que si no se limitaba el tiempo de ejecución del algoritmo genético, tardaba mucho tiempo en alcanzar la convergencia cuando el número de documentos era elevado. Por ello decidimos emplear el muestreo estadístico, de tal forma que el cálculo de un valor óptimo para *k* se realiza sobre una muestra representativa de la población. El algoritmo final es el siguiente:

1. Cálculo del tamaño de la muestra *n*.
2. Si $n = 1 \Rightarrow$ no hay división del conjunto inicial en clusters.
3. Si $n \neq 1$
   - 3.1. Selección aleatoria de la muestra.
   - 3.2. Cálculo de *k*:
     - 3.2.1. Si $n < 15 \Rightarrow$ Aplicar algoritmo de Calinski y Harabasz.
     - 3.2.2. Sino $\Rightarrow$ Aplicar algoritmo genético.
   - 3.3. Obtención de los *k*-clusters empleando un algoritmo de partición simple (en esta versión se ha empleado la librería de algoritmos de clustering, conocido el valor de *k* , de CLUTO (Karypis, 2002)).

6 Pruebas experimentales

Las colecciones de documentos con las que hemos realizado las pruebas experimentales corresponden a la salida de un motor de búsqueda ante consultas de un usuario. El corpus del que parten las colecciones está formado por noticias de la agencia EFE del año 2000. Cada noticia constituye un documento y cada documento se ha representado mediante los lemas de las palabras que contiene su primer párrafo.

Las pruebas experimentales se han llevado a cabo en un PC Pentium III a 933 MHz.

En la Tabla 1 reflejamos los resultados obtenidos para diversas colecciones de documentos. En la primera columna se indica el número de documentos sobre los que se va a realizar el clustering. La segunda columna refleja el tamaño de la muestra seleccionada. La tercera columna muestra el número de clusters obtenidos con la muestra. La cuarta indica el tiempo medio de procesamiento del algoritmo de clustering on-line para cada valor de *k*. El experimento se llevo a cabo 10 veces con cada colección de documentos; aspecto que se ve reflejado en los porcentajes que aparecen en la tercera columna. En ella queda patente cómo influye la aletoriedad en la elección de los documentos de la muestra en el resultado del *k* obtenido.

Nº documentos	**Nº muestras**	**K**	**Tiempo medio**
71	5	2 (40%) 3 (60%)	0,18 s 0,27 s
114	4	2 (40%) 3 (60%)	1,51 s 1,25 s
116	5	2 (10%) 3 (90%)	1,48 s 1,10 s
117	3	2 (100%)	0,42 s
119	3	2 (100%)	0,45 s
122	4	2 (10%) 3 (90%)	1,50 s 1,12 s
350	30	2 (20%) 3 (30%) 4 (30%) 5 (20%)	3,91 s 6,41 s 7,98 s 8,80 s

Tabla 1 : Resultados experimentales

Los tamaños de las muestras son reducidos, lo que indica que hay cierto grado de relación entre los documentos de partida. Esto es razonable ya que recordemos que todos ellos son documentos relevantes para una determinada consulta. Los tiempos obtenidos no llegan a los diez segundos incluso para la colección de 350 documentos. Consideramos que estos resultados son muy satisfactorios para un clustering, sin conocer a priori el valor de *k*.

7 Conclusiones

En este trabajo presentamos un algoritmo de clustering de documentos que obtiene un valor aproximado del óptimo número de clusters, *k*, y resuelve la forma de agruparlos dentro de esos *k* clusters. El énfasis se ha puesto en la reducción del tiempo de cálculo, por lo que podemos afirmar que nuestro algoritmo es capaz de realizar un clustering on-line.

El algoritmo combina el uso de la regla de detención global de Calinski y Harabasz, con un algoritmo genético diseñado sobre dicha

regla. El cálculo de un valor óptimo (o una aproximación al óptimo) para *k* se realiza sobre una muestra representativa de los documentos, aspecto que reduce considerablemente el tiempo de cálculo, a la vez que proporciona una condición de no partición del conjunto inicial.

Los resultados experimentales permiten concluir que el algoritmo diseñado aborda satisfactoriamente el clustering on-line sin conocer a priori el número de clusters.

Agradecimientos
Este trabajo de investigación forma parte del proyecto HERMES (TIC2000-0335-C03-03) financiado por el Centro de Investigación Científica y Tecnológica (CICYT) de España.

Bibliografía

Calinski, T., Harabasz, J.: "A Dendrite Method for Cluster Analysis". *Communications in Statistics*, 3(1), (1974) 1–27.

Casillas, A., González de Lena, M.T., Martínez, R.: "Document Clustering into an unknown number of clusters using a Genetic Algorithm", *International Conference on Text Speech and Dialogue TSD 2003*, September 8-11, Ceske Budejovice, Czech Republic (2003). Aceptado y pendiente de publicación.

Chu S.C., Roddick J.F., Pan J.S.: "An Incremental Multi-Centroid, Multi-Run Sampling Scheme for k-medoids-based Algortihms-Extended Report". *Proceedings of the Third International Conference on Data Mining Methods and Databases, Data Mining III*, (2002), 553–562.

Estivill-Castro V., Murray A.T.: "Spatial Clustering for Data Mining with Genetic Algorithms". *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, EIS-98*, (1998).

Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley Longman, Inc. (2002).

Gordon, A. D.: *Classification*, Chapman & Hall/CRC, (1999).

Holland, J. H.: *Adaptation in natural and artificial system*, Ann Arbor: The University of Michigan Press. (1975).

Imai, K., Kaimura, N., Hata, Y.:"A New Clustering with Estimation of Cluster Number Based on Genetic Algorithms". *Pattern Recognition in Soft Computing Paradigm*, World Scientific Publishing Co., Inc. (2000).

Karypis G.,*"CLUTO: A Clustering Toolkit"*, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, Technical Report: #02-017, 2002.

Lucasius C.B., Dane A.D., Kateman G.: "On k-medoid clustering of large data sets with the aid of Genetic Algorithm: background, feasibility and comparison". *Analytica Chimica Acta*, Elsevier Science Publishers B.V. 283(3), (1993) 647–669.

Merz P., Zell A.: "Clustering Gene Expresion Pro.les with Memetic Algorithms". *Lecture Notes in Computer Science 2439*, Springer-Verlag Berlin (2002) 811–820.

Michalewicz, Z.: *Genetic algorithms + data structures = evolution programs*, Springer Comp. (1996).

Milligan, G. W., Cooper, M.C.: "An Examination of Procedures for Determining the Number of Clusters in a Data Set". *Psychometrik*, 58(2), (1985) 159–179.

Murthy C.A., Chowdhury N.: "In search of Optimal Clusters Using Genetic Algorithms". *Pattern Recognition Letters*, 17(8), (1996), 825–832.

Pérez C., "*Técnicas de Muestreo Estadístico. Teoría, práctica y aplicaciones informáticas*", Ed. RA-MA, 1999.

Sarkar, M., Yegnanarayana, B., Khemani, D.: "A clustering algorithm using an evolutionary programming-based approach". *Pattern Recognition Letters*, 18, (1997) 975–986.