Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Algoritmos de Estimación de Distribuciones para el Clustering Ascendente Jerárquico

Autor: Mario Rubio González
Tutores: Concha Bielza y Juan Antonio Fernández del Pozo

Madrid, Julio 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Algoritmos de Estimación de Distribuciones para el Clustering Ascendente Jerárquico

Julio 2024

Autor: Mario Rubio González
Tutores: Concha Bielza y Juan Antonio Fernández del Pozo
Computational Intelligence Group
ETSI Informáticos
Universidad Politécnica de Madrid

Agradecimientos

Primero de todo quiero dar las gracias a Concha Bielza, Juan Antonio Fernández del Pozo y Pedro Larrañaga por confiar en mí para desarrollar este trabajo fin de máster. Su ayuda ha sido fundamental para superar los momentos más difíciles que han ido surgiendo por el camino.

También quiero agradecer a mi familia y amigos por haberme apoyado en estos últimos meses. Este camino hubiera sido más duro sin vosotros.

Este trabajo ha sido parcialmente apoyado por el Ministerio de Ciencia e Innovación de España a través de los proyectos PID2022-139977NB-I00, PLEC2023-010252 y TED2021-131310B-I00, y por la Comunidad Autónoma de Madrid en el marco de la ELLIS Unit Madrid.

Resumen

El procesamiento de grandes volúmenes de datos es una práctica común en la actualidad, lo que hace que áreas como el aprendizaje automático y la estadística sean sumamente importantes. En estas disciplinas, se utilizan con frecuencia técnicas de *clustering* que agrupan datos en función de su similitud, creando conjuntos donde los objetos de un mismo grupo tienen características similares. Este trabajo fin de máster tratará de presentar un nuevo enfoque para realizar *clustering*.

En este trabajo se presenta un estudio sobre el uso de algoritmos de estimación de distribuciones (EDAs) en el *clustering* jerárquico ascendente. Se presenta un nuevo algoritmo llamado EDAPerm. El objetivo principal es explorar y evaluar cómo los EDAs pueden mejorar los resultados del *clustering* jerárquico a través de una estimación más ajustada de la distribución de los dendrogramas más prometedores. Estos dendrogramas se codifican en un vector compuesto por una permutación que indica el orden de los objetos, y el resto de elementos que representan la estructura del dendrograma. El trabajo se enfoca en la implementación del modelo de Mallows y su variante generalizada, los cuales son modelos probabilísticos basados en permutaciones que han demostrado ser efectivos en la captura de la estructura de las permutaciones en conjuntos de datos complejos.

El estudio incluye una comparación de diferentes distancias utilizadas en el modelo de Mallows, tales como Kendall, Cayley, Ulam y Hamming, evaluando su impacto en la calidad del dendrograma del *clustering* generado. Este estudio se realiza con tres conjuntos de datos con características diferentes, y se compara EDAPerm con el algoritmo estándar de *clustering* jerárquico.

Los resultados obtenidos muestran que los EDAs pueden proporcionar mejoras significativas en términos de calidad en el agrupamiento jerárquico, especialmente cuando se combinan con técnicas de estimación y muestreo de distribuciones (como los EDAs) que se ajustan mejor al problema a resolver.

Se concluye que EDAPerm cumple con los objetivos planteados, superando al algoritmo tradicional de *clustering* jerárquico en calidad de los grupos. Como futuras líneas de investigación se propone mejorar la representación de individuos, paralelizar la ejecución de los EDAs y así disminuir el tiempo de cómputo y, por último, realizar más pruebas con conjuntos de datos con más objetos.

Abstract

Processing large volumes of data is common practice today, making areas such as machine learning and stadistics extremely important. In these disciplines, *clustering* techniques are frequently used that group data based on their similarity, creating sets where objects within the same group have similar characteristics. This master's thesis will try to present a new approach to performing *clustering* is presented.

This work presents a study on the use of estimation of distribution algorithms (EDAs) in agglomerative hierarchical *clustering*. A new algorithm called EDAPerm is presented. The main objective is to explore and evaluate how EDAs can improve hierarchical *clustering* results through an improved estimation of the distribution of the most promising dendrograms. These dendrograms are encoded in a vector composed of a permutation that indicates the order of the objects, and the rest of the elements represents the structure of the dendrogram. The work focuses on the implementation of the Mallows model and its generalized version, which both are probabilistic models based on permutations that have proven to be effective in capturing the structure of permutations in complex datasets.

The study includes a comparison of different distances used in the Mallows model, such as the Kendall, Cayley, Ulam and Hamming distances, evaluating their impact on the quality of the generated *clustering* dendrogram. This study is carried out with three datasets with different characteristics, and EDAPerm is compared with the standard hierarchical *clustering* algorithm.

The results obtained show that EDAs can provide significant improvements in terms of quality in hierarchical *clustering*, especially when combined with distribution estimation and sampling techniques (as EDAs do) that better fit the problem to be solved.

It is concluded that EDAPerm meets the stated objectives, surpassing the traditional hierarchical *clustering* algorithm in terms of the quality of the groups. As future lines of research, it is proposed to search for a better representation of individuals, parallelize the execution of the EDAs and thus reduce the computing time and, finally, perform more tests with datasets with more objects.

Tabla de contenidos

Capítulo 1

Introducción

1.1. *Clustering* con EDAs

El tratamiento y la interpretación de grandes cantidades de datos está a la orden del día. Es por ello que los campos como el aprendizaje automático o la estadística son muy necesarios. En estos campos una tarea importante es el *clustering*. Esta tarea consiste en agrupar objetos de datos por su similitud, de forma que los objetos de un mismo grupo tengan características similares. Los grupos de datos obtenidos por diferentes algoritmos de *clustering* son significativamente distintos en el enfoque, fundamento y objetivos de la tarea de agrupación; y no existe un solución única a un problema de *clustering*, ni un algoritmo general para cualquier conjunto de datos ni dominio de aplicación. A día de hoy sigue siendo una tarea difícil el hecho de predecir qué y cuántos grupos hay en un conjunto de datos. Una de las principales razones es que el *clustering* es una tarea de aprendizaje no supervisado, lo que significa que no conocemos a priori el número de grupos, ni el nombre o las etiquetas de los grupos.

Algunos de las áreas de aplicación del*clustering* son biología, análisis de imágenes, marketing reconocimiento de patrones, búsqueda de información, etc.

Existen diferentes tipos de *clustering*, pero en este trabajo nos vamos a centrar en el *clustering* jerárquico, ya que es uno de los más populares. Especificamente en la estrategia ascendente o aglomerativa, que es la más usada. En esta, se parte de muchos grupos de datos (un grupo por objeto), los cuales se van a ir agrupando en grupos más grandes. El resultado que se obtiene al aplicar un algoritmo de *clustering* jerárquico es un dendrograma, el cual es un árbol binario en el que se puede ver cómo se van uniendo los grupos desde los datos, situados en los nodos terminales u hojas, hacia arriba y a qué altura se unen. De ahí resultan los grupos finales.

En este trabajo se estudia la forma de integrar algoritmos de estimación de distribuciones (EDAs) en el problema de optimizar el *clustering* jerárquico. Los EDAs entran dentro del campo de la computación evolutiva y se aplican para optimización heurística, ya que tratan de encontrar el individuo que mejor se adapte al problema en cuestión. El problema en cuestión que se trata de resolver es ser capaces de dividir un conjunto de datos en diferentes grupos. El objeto de un grupo será similar a otro objeto de ese mismo grupo y distinto a un objeto de fuera de ese grupo. Los EDAs, a diferencia de otros algoritmos evolutivos, son capaces de capturar y explotar las dependencias y relaciones entre las variables de decisión del problema de optimización.

Esto hace que la exploración del espacio de búsqueda sea más eficiente. Además estos algoritmos encuentran un buen balance entre exploración y explotación gracias a la estimación del modelo probabilístico en cada iteración.

A lo largo de este documento se irán detallando los problemas que han ido surgiendo y cómo se han ido solucionando.

1.2. Objetivos

El objetivo principal de este proyecto es desarrollar un enfoque, basándonos en EDAs, con el que se pueda conseguir el dendrograma que represente las agrupaciones con más calidad del conjunto de datos del problema. El algoritmo, que llamaremos EDA-Perm, partirá de una población de individuos completamente aleatoria, y en cada iteración tendrá que seleccionar un conjunto de ellos, estimar la distribución de probabilidad y muestrear nuevos individuos con el fin de crear una nueva población que se ajuste a los objetivos del *clustering*, similitud entre individuos de un grupo y diferencias entre grupos.

Uno de los desafíos más grandes de este trabajo es encontrar una codificación válida de los individuos que sea capaz de representar fielmente dendrogramas de distintos tamaños sin que se consuma una gran cantidad de memoria. Además, la dificultad añadida de esto es conseguir encontrar una codificación de individuos que pueda representar la totalidad del espacio de búsqueda y así no tener que sacrificar la capacidad de exploración.

La metodología propuesta se basa en el trabajo realizado por Lozano y Larrañaga (1999), ya que se toma la misma función objetivo para realizar la selección de los individuos. Además de esto, para obtener el coste de cada individuo se calcula la matriz de distancias ultramétricas siguiendo las mismas aproximaciones que se indican en el artículo mencionado. Por lo tanto, el objetivo fundamental es superar los resultados obtenidos por el algoritmo de *clustering* tradicional.

1.3. Estructura del trabajo fin de máster

En este apartado se muestra la estructura que tendrá este trabajo, y se destacará el contenido principal de cada capítulo.

El Capítulo 2 contiene todo el estado del arte necesario para poner en contexto los enfoques que existen hasta el día de hoy en la literatura. Se verán los enfoques más relevantes de distintos tipos de *clustering* y de EDAs. Además se presentará el modelo Mallows y se verán enfoques evolutivos aplicados a *clustering* particional y jerárquico.

En el Capítulo 3 se puede encontrar el desarrollo completo del algoritmo implementado. En dicho apartado se presentan todos los conceptos necesarios para entender su funcionamiento, la codificación del individuo, el generador de estructuras de dendrogramas, la generación inicial de la población, la selección de individuos, la estimación de la distribución de probabilidad y la generación de nuevos individuos muestreando la distribución estimada.

En el Capítulo 4 se podrán ver los resultados de las pruebas realizadas con el enfoque presentado. Primero se describirán los diferentes conjuntos de datos con los que se

realizan las diferentes pruebas. Las pruebas consisten en una comparación de las distancias usadas en el modelo de Mallows y la comparación de EDAPerm con el algoritmo de *clustering* jerárquico tradicional.

Por último, en el Capítulo 5 se comentarán las conclusiones alcanzadas en la realización de este trabajo y se presentarán las posibles lineas de investigación que se pueden seguir en el futuro.

Capítulo 2

Estado del arte

Este capítulo va a dar una visión más detallada de los componentes más importantes de este trabajo. En cada sección se hará una breve introducción explicando los aspectos más importantes y seguidamente se pondrán en contexto las diferentes metodologías que hay en el estado del arte. De esta forma, se va a conseguir una visión general del estado actual de las líneas de investigación abiertas más similares a este trabajo.

2.1. *Clustering*

Ser capaces de agrupar objetos es muy necesario en diferentes áreas como ingeniería, ciencia y tecnología, humanidades y medicina. Si tomamos unas instancias etiquetadas de un conjunto de datos, normalmente serán usadas para realizar un aprendizaje supervisado, comprobando sí nuestro modelo ha sido capaz de predecir correctamente la clase asignada a cada instancia. El principal objetivo de la clasificación supervisada es desarrollar un algoritmo que pueda predecir las clases de un objeto sin clasificar. Contrariamente a esto, el *clustering*, se corresponde con el aprendizaje no supervisado, el cual no va a necesitar instancias etiquetadas para poder clasificarlas. Según Rokach y Maimon (2010) el *clustering* divide los patrones del conjunto de datos en subconjuntos con patrones similares. El *clustering* se considera más complejo que la clasificación supervisada, ya que no tiene una etiqueta asociada a cada instancia. En Saxena et al. (2017) y Ezugwu et al. (2022) se encuentran dos revisiones actualizadas que pueden servir para contextualizar todas las metodologías más importantes hasta el día de hoy,

Los principales componentes de las tareas del *clustering* contienen los siguientes pasos. El primero, la representación de patrones, que se refiere al número de grupos, y el número, tipo y escala de características disponibles para el algoritmo de *clustering*. Dentro de este paso, está también la selección de características, que es el proceso de identificar el subconjunto más efectivo de las características. Y también, la extracción de características, que se corresponde con la transformación de una o más características para crear otra característica que pueda resultar más interesante. El segundo paso, la proximidad de los patrones, el cual se refiere a la selección de una función de distancias la cual se usa para parejas de objetos. El tercer paso consiste en el *clustering* en sí. En este paso la distancia elegida antes afecta directamente a la formación de los grupos resultantes. Uno de los mayores problemas de este paso

es que no existe un algoritmo de *clustering* que pueda usarse universalmente para resolver todos los problemas. Por lo tanto, es importante investigar cuidadosamente las características del problema en cuestión para seleccionar una estrategia de *clustering* adecuada. El cuarto paso, la abstracción de datos, se realiza solo cuando es necesario, y consiste en una descripción compacta de cada grupo. Y, por último, el quinto paso se corresponde con la evaluación de los grupos creados.

2.1.1. Medidas de similitud

Para poder determinar cómo de similares son dos objetos, se utiliza el cálculo de distancias (Deza et al. (2009)). Diremos que dos objetos son similares si en un espacio de objetos se encuentran más próximos el uno del otro. Y, por el contrario, diremos que dos objetos son distintos si se encuentran a una distancia mayor. En esta sección se van a centrar los esfuerzos en explicar detalladamente las distancias más usadas en la literatura para realizar *clustering*. En el estado del arte se han realizado algunos estudios comparativos sobre diferentes distancias usadas en *clustering*, como Irani et al. (2016) y Pandit et al. (2011).

Distancia Euclídea

La distancia euclídea es la métrica estándar usada en problemas de geometría. Es la distancia de la línea recta entre dos puntos. Así, dados dos puntos $p = (p_1, ..., p_n)$ y $q = (q_1, ..., q_n)$ esta distancia es:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.1}$$

Distancia de Minkowski

La distancia de Minkowski es conocida como una distancia generalizada.

$$d(\mathbf{p}, \mathbf{q}) = \left(\sum_{i=1}^{n} |p_i - q_i|^k \right)^{\frac{1}{k}} \tag{2.2}$$

En la ecuación 2.2, si $k = 1$ se convierte en la distancia de Manhattan. Cuando $k = 2$, la ecuación se convierte en la fórmula de la distancia euclídea. Y, cuando $k = \infty$, la ecuación pasa a ser la distancia de Chebyshev.

Distancia de Manhattan

La distancia de Manhattan es la distancia mínima que hay entre dos puntos. También es conocida como la distancia de la ciudad, porque representa la distancia que un taxista recorrería en una ciudad con calles en cuadrícula:

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} |p_i - q_i| \tag{2.3}$$

Distancia de Chebyshev

La distancia de Chebyshev es el caso donde en la ecuación 2.2 k es igual a ∞:

$$d(\mathbf{p}, \mathbf{q}) = \max_{i=1}^{n} |p_i - q_i| \tag{2.4}$$

Distancia de Hamming

La distancia de Hamming entre dos cadenas de igual longitud es el número de posiciones para las cuales los símbolos correspondientes son diferentes.

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \delta(p_i, q_i) \tag{2.5}$$

donde $\delta(p_i, q_i)$ es una función que devuelve 0 si p_i y q_i son iguales, y 1 si son diferentes. Esta función es identica a usar el operador XOR ($\oplus$).

Distancia de Mahalanobis

La distancia de Mahalanobis es una medida estadística utilizada para determinar la distancia entre dos puntos:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{p} - \mathbf{q})^T \mathbf{S}^{-1} (\mathbf{p} - \mathbf{q})} \tag{2.6}$$

donde S^{-1} es la matriz de covarianza inversa de las variables.

Distancia del coseno

La distancia de coseno es una medida de similitud entre dos vectores en un espacio vectorial:

$$d(\mathbf{p}, \mathbf{q}) = \arccos \frac{\mathbf{p} \cdot \mathbf{q}}{\|\mathbf{p}\| \|\mathbf{q}\|} \tag{2.7}$$

Coeficiente de correlación de Pearson

El coeficiente de correlación de Pearson es una medida estadística que evalúa la relación lineal entre dos variables continuas. Tiene un rango de valores entre -1 y 1, donde 1 indica una correlación positiva perfecta, 0 indica ausencia de correlación lineal y -1 indica una correlación negativa perfecta. Se define como:

$$r_{pq} = \frac{\sum_{i=1}^{n} (p_i - \bar{p})(q_i - \bar{q})}{\sqrt{\sum_{i=1}^{n} (p_i - \bar{p})^2} \sqrt{\sum_{i=1}^{n} (q_i - \bar{q})^2}} \tag{2.8}$$

donde $\bar{p}$ y $\bar{q}$ son las medias de los valores p y q respectivamente.

Índice de Jaccard

El índice de Jaccard es una medida estadística utilizada para comparar la similitud y diversidad de conjuntos de muestras. Se define como:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.9}$$

siendo A y B dos conjuntos de muestras.

Junto con el problema de incorporar medidas basadas en la distancia para el *clustering*, muchas técnicas presentan diferentes problemas. Alguno de estos problemas son tener que especificar a priori el número de grupos en los que se debe dividir el

conjunto de datos. Que este sea un valor estático dificulta el *clustering* cuando se tienen valores atípicos o datos ruidosos, entre otras cosas. Por lo tanto, no sólo es importante agrupar los datos basándose en patrones similares, sino que también los datos deben agruparse de manera eficiente usando las técnicas o métodos correspondientes en cada problema. A continuación, en los siguientes apartados se van a ver diferentes métodos o algoritmos para cada tipo de *clustering*.

2.1.2. *Clustering* jerárquico

En el *clustering* jerárquico los grupos del conjuntos de datos se representan mediante un dendrograma (Johnson (1967)). Un dendrograma es un árbol binario que organiza los datos en categorías que se van dividiendo en otros hasta llegar a un nivel de detalle deseado. Los grupos de datos se crean, o bien, de forma ascendente o descendente. Comúnmente llamados aglomerativo y divisivo, respectivamente (Saxena et al. (2017)). En el método aglomerativo los dendrogramas son formados de abajo hacía arriba. Esto quiere decir que cada grupo comienza con un solo objeto y se van fusionando iterativamente con otros grupos, para formar grupos más grandes. Esto se realiza hasta que todos los objetos se encuentran en el mismo grupo o se cumple un criterio de parada. En el caso contrario, el método divisivo, comienza con todos los objetos dentro de un mismo grupo. Este grupo se va dividiendo iterativamente hasta que cada objeto se encuentra en un grupo distinto o se cumpla una condición de parada. La unión o la partición de los grupos se realiza basándose en las medidas de similitud vistas en la sección 2.1.1.

En este tipo de *clustering* y especificamente el aglomerativo, la unión se realiza generalizando la distancia que hay entre los objetos individuales a la distancia que hay entre los grupos de objetos. Esto se determina usando diferentes métodos de enlace. Los métodos de enlace más usados aparecen en el Cuadro 2.1.

Método de enlace	**Descripción**	**Complejidad**
Enlace simple	Distancia mínima entre pares de puntos de dos grupos.	$O(n^2)$
Enlace completo	Distancia máxima entre pares de puntos de dos grupos.	$O(n^2 \log n)$
Enlace basado en medias	Distancia promedio entre pares de puntos de dos grupos.	$O(n^2 \log n)$
Enlace basado en centroides	Distancia entre los centroides de dos grupos.	$O(n^2 \log n)$
Enlance de Ward	Minimiza la varianza total dentro de los grupos.	$O(n^2)$

Cuadro 2.1: Métodos de enlace en *clustering* jerárquico

Estos cinco métodos de enlace son los más utilizados en la literatura, pero existen más. En Saraçli et al. (2013) se muestran muchas pruebas de estos métodos con distintas distancias. Ahí se concluye que de estos cinco métodos de enlace, se obtienen mejores resultados utilizando el método basado en medias. Cabe destacar que estas pruebas se han realizado con distintos números de objetos, pero en la mayoría se ha usado la correlación de Cophenetic como criterio de evaluación de los grupos creados.

Las correlación de Cophenetic es una medida que evalúa la fidelidad de un dendrograma con respecto a la matriz de distancias originales entre las muestras. Además, es necesario resaltar que cada método de enlace puede tener un mejor o peor comportamiento dependiendo del conjunto de datos utilizado para realizar las pruebas. Por ello en cada problema habrá que ver cuál es el método con el que mejores resultados se obtienen. En Jarman (2020) se revisaron estos métodos y además se realizó un estudio para determinar cuál de estos métodos era el más eficiente en tiempo de ejecución. Se utilizó el algoritmo tradicional del *clustering* jerárquico para obtener la complejidad temporal de cada método de enlace. El orden de la complejidad de cada método se puede observar en la tabla 2.1.

Una vez explicados los métodos de enlazado, se van a poner en contexto los algoritmos de *clustering* jerárquico más importantes desarrollados. Antes de esto, en el algoritmo 1, se recuerdan los pasos seguidos por los Algoritmos de *clustering* jerárquico aglomerativo. También se puede ver en el Algoritmo 2 los pasos de los algoritmos de *clustering* jerárquico divisivo.

Algoritmo 1 Pasos *clustering* jerárquico aglomerativo

1: Asignar cada objeto a un grupo distinto
2: Encontrar la pareja de grupos más similares (dependiendo del método de enlace y distancia) y unirlos
3: Calcular las distancias entre el nuevo grupo y el resto de grupos
4: Repetir desde el paso 2 hasta que todos los objetos pertenezcan al mismo grupo

Algoritmo 2 Pasos *clustering* jerárquico divisivo

1: Asignar todos los objetos a un mismo grupo
2: Seleccionar el grupo a dividir y evaluar las posibles divisiones del grupo
3: Dividir el grupo en dos subgrupos
4: Repetir desde el paso 2 hasta que todos los objetos pertenezcan a un grupo diferente cada uno

En la literatura existen varias revisiones sobre los algoritmos de *clustering* jerárquico desarrollados en la literatura, como Murtagh y Contreras (2012). Los diferentes algoritmos de *clustering* jerárquico que se van a exponer a continuación presentan pequeñas variaciones con respecto al Algoritmo 1.

CHAMALEON (Karypis et al. (1999)) es un algoritmo de *clustering* jerárquico aglomerativo que aborda las limitaciones de los métodos de *clustering* tradicionales mediante la utilización de un modelo dinámico que considera tanto la interconectividad como la cercanía entre los clusters. Este algoritmo representa los datos mediante un grafo disperso donde los nodos son los objetos y las aristas representan las similitudes entre estos elementos. Utiliza un enfoque de k-vecinos más cercanos (kNN) para construir este grafo, donde cada vértice está conectado a sus k vecinos más similares. Una vez modelados los datos, se divide en dos fases principalmente. La primera fase consiste en utilizar un algoritmo de particionado de grafos para dividir el conjunto de datos en varios grupos más pequeños. En la segunda fase se combinan iterativamente los grupos utilizando un marco de modelado dinámico que evalúa la interconectividad relativa y la cercanía relativa entre los grupos. Una limitación de CHAMALEON es que es conocido por ser aplicable solo a espacios de bajas dimensiones.

En Zhang et al. (1996) presentan el algoritmo llamado BIRCH (*Balanced Iterative*

Reducing and Clustering using Hierarchies), diseñado para manejar grandes bases de datos de manera eficiente. Este busca encontrar patrones útiles en conjuntos de datos grandes, identificando regiones muy pobladas en un espacio multidimensional. BIRCH utiliza una estructura de características llamada *Clustering Feature* (CF) para resumir la información de los grupos. Un CF es un vector que contiene el número de objetos en el grupo, la suma lineal de los objetos y la suma cuadrática de los objetos. Además se crea un árbol balanceado de CFs (CF-Tree) para almacenarlos. Algunas de las ventajas de este algoritmo son la posibilidad de manejo de grandes conjuntos de datos y la capacidad de adaptabilidad dinámica según el conocimiento del conjunto de datos.

En Guha et al. (1998) se presenta un algoritmo de *clustering* jerárquico diseñado también para manejar conjuntos grandes de datos, llamado CURE (*Clustering Using Representatives*). Este se caracteriza por mejorar la robustez frente a valores atípicos y por identificar grupos con formas no esféricas y tamaños variados. En cada paso del algoritmo, se unen los grupos cuyos objetos estén más cercanos. Este proceso se repite hasta obtener el número deseado de grupos. BIRCH y CURE manejan bien los valores atípicos, pero la calidad de los grupos de CURE es mejor que la de BIRCH. Por otro lado, en el aspecto de complejidad temporal BIRCH es mejor que CURE.

ROCK (*Robust Clustering Algorithm for Categorical Attributes*) es un algoritmo presentado en Guha et al. (2000) diseñado para abordar datos categóricos. Este emplea enlaces en lugar de distancias para medir la similitud entre una pareja de puntos de datos. Trata de maximizar el número de enlaces entre puntos dentro del mismo grupo.

2.1.3. *Clustering* particional

El *clustering* particional es un tipo de *clustering* completamente opuesto al *clustering* jerárquico, ya que aquí los datos son asignados a un número especifico de grupos sin tener que crear ninguna estructura jerárquica. Para saber a qué grupo se debe asignar cada objeto se siguen criterios como la distancia euclídea.

En la literatura se pueden observar diferentes tipos de taxonomías para organizar los contenidos del *clustering* no jerárquico (Ezugwu et al. (2022) y Saxena et al. (2017)). En concreto, una división más tradicional del *clustering* particional está formada por enfoques basados en distancias, basados en modelos y basados en densidades. Estos dos últimos tipos corresponden al tipo de *clustering* probabilístico el cual se va a tratar en la Sección 2.1.4. En el *clustering* particional, además de haber enfoques basados en distancias, también se pueden destacar otros tipos como modelos basados en teoría de grafos y basados en subespacios.

Dentro del propio *clustering* particional se pueden hacer aún más divisiones. Una división muy común es separar los modelos en *clustering* duro y *clustering* suave. El *clustering* duro se corresponde con los enfoques que asignan cada objeto exclusivamente a un grupo determinado. Y el *clustering* suave o *fuzzy clustering* contiene aquellos métodos en los cuales cada objeto puede pertenecer a múltiples grupos simultáneamente, con diferentes grados de pertenencia. En este apartado se van a ver diferentes algoritmos de ambos tipos, duro y suave.

La teoría de grafos puede ser usada para representar tanto modelos de *clustering* jerárquico como no jerárquicos. En la Sección 2.1.2 se vió el modelo CHAMALEON

que consiste en un modelo de grafo basado en los k vecinos más cercanos. Un enfoque de *clustering* particional basado en teoría de grafos es el algoritmo de Zahn (1971), el cual genera un árbol de expansión mínimo (MST), identifica aristas inconsistentes y las elimina. De esta forma es capaz de dividir los objetos (nodos) en distintos grupos.

En Sharan y Shamir (2000) se presentó otro algoritmo de *clustering* particional basado en la teoría de grafos, llamado CLICK. Este enfoque consiste en realizar la división de peso mínimo del grafo para generar los grupos.

Los modelos de *clustering* basados en subespacios buscan identificar grupos en subespacios de baja dimensionalidad. En muchos conjuntos de datos de alta dimensión, no todos los atributos son relevantes para todos los grupos. Se trata de encontrar grupos que existan en diferentes subespacios. En Agrawal et al. (1998) se prensenta un enfoque llamado CLIQUE, que es considerado un enfoque basado en rejillas e incluso basado en densidades. Está diseñado para manejar datos de alta dimensionalidad encontrando grupos en diferentes subespacios del espacio de características. Lo primero que hace el algoritmo es dividir el espacio multidimensional en rejillas e identificar las celdas que tengan más objetos en ellas. Lo siguiente es combinar las celdas más densas. Un grupo está compuesto por todas las celdas densas que están conectadas entre sí en un subespacio.

Dentro del *clustering* particional basado en distancias los tipos de enfoques más usados son los de suma de errores cuadrados. Estos asignan todos los objetos en un número específico de grupos en función de la suma de errores al cuadrado:

$$SSE = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \tag{2.10}$$

donde n representa el número de objetos, x_i es el objeto número i de un grupo y $\hat{x}_i$ es el valor central estimado de ese mismo grupo. En los casos en que la suma de los errores al cuadrado de un grupo de objetos es igual a cero, los objetos del grupo son idénticos.

El algoritmo de k-means (MacQueen et al. (1967)) es el algoritmo de *clustering* particional más conocido, basado en la suma de errores al cuadrado. El algoritmo k-means es un método de *clustering* particional que agrupa un conjunto de datos en k grupos, donde k es un número previamente definido por el usuario. El objetivo principal es minimizar la variabilidad dentro de cada grupos y maximizar la variabilidad entre los grupo. Esto se consigue intentando minimizar la suma de las distancias cuadradas de cada objeto al centro de su grupo más cercano. La función objetivo que se basa en la fórmula 2.10 viene dada por: $MinimizarJ = \sum_{i=1}^{k} \sum_{i=1}^{n} \|x - \hat{x}_i\|^2$, donde $\|x - \hat{x}_i\|^2$ representa la distancia al cuadrado elegida entre un objeto y el centro de su grupo. El procedimiento de este algoritmo comienza seleccionando k centroides como puntos iniciales de los grupos. Después se asignan cada objeto al centroide del grupo más cercano. Cada vez que se asigna un nuevo objeto a un grupo se recalculan los centroides. Por último, estos pasos se repiten hasta que se logre la convergencia, las asignaciones y los centroides de los grupos ya no cambien.

A partir de la popularidad de k-means surge el algoritmo de k-medoids (Park y Jun (2009)). Uno de sus objetivos es mejorar al algoritmo k-means en gasto computacional. El procedimiento que sigue este algoritmo es muy similar al de k-means y comienza por calcular la distancia entre cada par de objetos para seleccionar puntos

reales del conjunto de datos como centros de grupos, llamados medoides. A continuación, se actualizan estos medoides de forma que se minimice la distancia total a otros objetos del mismo grupo. Y, por último, se asignan los objetos a los medoides más cercanos. Estos últimos pasos se repiten hasta que se cumpla la condición de parada.

En Kaufman y Rousseeuw (2009) se propuso el algoritmo PAM (*Partitioning Around Medoids*), que es un método de *clustering* que utiliza medoides. Es menos sensible a los valores atípicos en comparación con k-means, ya que utiliza medoides en lugar de centroides. Los pasos que sigue este algoritmo son muy simples. Este comienza seleccionando los medoides iniciales de forma aleatoria. Asigna cada objeto a su modoide más cercano y actualiza los medoides. Esta actualización se realiza calculando el costo total de intercambiar el medoide con un objeto. Este costo es la variación en la suma de las distancias de todos los objetos a su medoide más cercano. Si se encuentra un intercambio que reduce el costo total, se realiza el intercambio. Estos últimos pasos se repiten hasta que no haya mejoras en el costo total.

Otro algoritmo de *clustering* particional presentado en Kaufman y Rousseeuw (2009) y basado en distancias es CLARA (*Clustering Large Applications*). Este algoritmo es una extensión del algoritmo PAM propuesta para manejar grandes conjuntos de datos de manera más eficiente. CLARA mejora la eficiencia de PAM usando muestras aleatorias del conjunto de datos en lugar de procesar todo el conjunto de datos. El procedimiento de CLARA comienza seleccionando muestras aleatorias del conjunto de datos y para cada una de estas muestras se aplica el algoritmo PAM para encontrar k medoides. A continuación, se asignan todos los objetos del conjunto de datos a los medoides más cercanos encontrados en esa muestra. Por último, se evalúa la calidad de los grupos. Estos pasos se repiten con diferentes muestras del conjunto de datos y se selecciona el conjunto de medoides que produce el *clustering* con mayor calidad.

En Ng y Han (2002) se propuso un algoritmo llamado CLARANS (*Clustering Large Applications based on RANdomized Search*) que también surge por la necesidad de manejar grandes conjuntos de datos. CLARANS utiliza una búsqueda aleatoria para identificar estructuras espaciales en los datos, siendo más eficiente que otros métodos existentes como PAM y CLARA. Este algoritmo comienza con una selección aleatoria de k medoides iniciales del conjunto de datos. En cada iteración un medoide se intercambia aleatoriamente con otro objeto que no sea medoide. Si el intercambio reduce el costo total, se acepta el nuevo conjunto de medoides. Este paso se repite iterativamente hasta que no se encuentre un mejor conjunto de medoides en un número prefijado de intercambios. Después de todas las iteraciones el conjunto de medoides que produce el menor coste total se selecciona como el resultado final del *clustering*.

Hasta ahora se han revisado los algoritmos de *clustering* particional duro con mayor impacto en la literatura, pero también cabe destacar un algoritmo de *clustering* suave o difuso (Zadeh (1965)), que tiene mucha relevancia. Este algoritmo es Fuzzy k-means (FCM) (Dunn (1973)) y a diferencia del algoritmo k-means, donde cada objeto pertenece a un solo grupo, en FCM cada objeto tiene un grado de pertenencia a cada grupo. El procedimiento de este algoritmo comienza determinando el número de grupos k y elige un valor para el parámetro de difusividad m. A continuación, se calculan los centroides de los grupos dependiendo del grado de pertenencia de cada

objeto en cada grupo, que viene dado por una matriz de pertenencia. Por último, se repite este paso hasta que la matriz de pertenencia deje de cambiar entre iteraciones.

Además de los enfoques vistos, existen muchos más que tuvieron impacto en el estado del arte, como DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) presentado en Ester et al. (1996). Este enfoque es un algoritmo de clustering basado en densidades que identifica grupos en conjuntos de datos espaciales y no espaciales. Algunas de sus variantes son VDBSCAN (Liu et al. (2007)), GRIDBSCAN (Uncu et al. (2006)) o ST-DBSCAN (Birant y Kut (2007)). Otro enfoque es SOM (*Self-Organizing Map*) presentado en Vesanto y Alhoniemi (2000) es utilizado principalmente como una herramienta para la reducción de dimensionalidad y la visualización de datos de alta dimensionalidad. Un algoritmo presentado en Wang et al. (2008) llamado *Affinity Propagation* (AP) se basa en el intercambio de mensajes entre puntos de datos. Por último, se debe comentar otro algoritmo con un gran impacto llamado *Spectral Clustering* (Von Luxburg (2007)). Este método se basa en las propiedades de los valores de las matrices de distancias que representan la similitud entre puntos de datos.

2.1.4. *Clustering* probabilístico

El *clustering* probabilístico es un tipo de *clustering* no jerárquico que asigna probabilidades de pertenencia a cada objeto para cada uno de los grupos en lugar de asignar cada objeto a un único grupo. Este tipo de *clustering* permite que cada objeto pertenezca a varios grupos con diferentes grados de probabilidad.

Fraley y Raftery (1998) presentaron un tipo de *clustering* probabilístico basado en modelos. Este tipo de *clustering* asume que los datos son generados por una distribución de probabilidad subyacente o un modelo. El objetivo principal es optimizar el desempeño del modelo con respecto a los datos.

El algoritmo de *clustering* basado en mixturas (McLachlan y Chang (2004)), es una técnica de *clustering* que asume que los datos son generados a partir de una mezcla de varias distribuciones de probabilidad. Cada distribución en la mixtura corresponde a un grupo diferente de objetos. Cada componente del modelo tiene unos parámetros que describen su forma. En el caso de mixturas de Gaussianas, cada componente tiene el vector de medias y la matriz de covarianzas. En lugar de asignar cada objeto a un solo grupo, el algoritmo proporciona una probabilidad de pertenencia de cada objeto a cada grupo. El algoritmo que se usa para estimar los parámetros de un modelo de mixtura, es el algoritmo EM (Moon (1996)). Este tiene dos grandes pasos, el de esperanza (E) y el de maximización (M). En el paso de esperanza se calculan las probabilidades de que cada objeto pertenezca a cada uno de los grupos, basándose en los parámetros actuales del modelo. Y en el paso de maximización se actualizan los parámetros de los grupos para maximizar la probabilidad de los datos.

A modo de resumen de todos los enfoques presentados en esta sección, se presenta el siguiente cuadro donde se encuentran los nombres de los enfoques y sus referencias.

2.2. Algoritmos de estimación de distribuciones

Los EDAs presentados en Larrañaga y Lozano (2001), son una clase de algoritmos evolutivos (AEs) donde en cada iteración se intenta explorar el espacio de búsqueda

Nombre del enfoque	**Referencia**
CHAMALEON	Karypis et al. (1999)
BIRCH	Zhang et al. (1996)
CURE	Guha et al. (1998)
ROCK	Guha et al. (2000)
CLICK	Sharan y Shamir (2000)
CLIQUE	Agrawal et al. (1998)
k-means	MacQueen et al. (1967)
k-medoids	Park y Jun (2009)
PAM	Kaufman y Rousseeuw (2009)
CLARA	Kaufman y Rousseeuw (2009)
CLARANS	Ng y Han (2002)
FCM	Dunn (1973)
DBSCAN	Ester et al. (1996)
VDBSCAN	Liu et al. (2007)
GRIDBSCAN	Uncu et al. (2006)
ST-DBSCAN	Birant y Kut (2007)
SOM	Vesanto y Alhoniemi (2000)
AP	Wang et al. (2008)
Spectral clustering	Von Luxburg (2007)

Cuadro 2.2: Resumen de los enfoques del estado del arte del *clustering*

gracias a un modelo de probabilidad que ha sido aprendido con los mejores individuos. A diferencia de los AEs tradicionales, como el algoritmo genético, que utiliza operadores de cruce y mutación, los EDAs crean un modelo probabilístico a partir de las soluciones más prometedoras encontradas hasta el momento. Más adelante se generan nuevas soluciones mediante el muestreo de individuos de este modelo.

El procedimiento del EDA se puede observar en el Algoritmo 3. En la línea 1 se obtiene la población inicial, de forma aleatoria o con una población ya generada previamente. Más adelante se entra en un bucle del cual no se sale hasta que no se cumpla un criterio de parada establecido. Por lo general este criterio de parada es el número de iteraciones (línea 2). Dentro del bucle se evalúa cada individuo (línea 3), se seleccionan los mejores individuos (línea 4), se aprende el modelo de probabilidad con los individuos seleccionados (línea 5) y se muestrean nuevos individuos con el modelo aprendido.

Algoritmo 3 Pasos del EDA

Input: Tamaño de la población, función objetivo (coste)
Output: Mejor individuo, coste
1: Población inicial
2: **for** $i = 1, 2, \ldots$ **until** cumplir el criterio de parada **do**
3: Evaluar la población usando una función objetivo
4: Seleccionar mejores individuos
5: Aprender el modelo probabilístico a partir de los mejores individuos
6: Muestrear nuevos individuos con el modelo probabilístico
7: **end for**

Una revisión con diferentes modelos probabilísticos se encuentra en Hauschild y Pelikan (2011). Los EDAs más simples asumen que las variables del problema, $X_1, \ldots, X_n$, son independientes. Matemáticamente, la probabilidad conjunta es el producto de las probabilidades marginales:

$$p(X_1, X_2, \ldots, X_n) = p(X_1)p(X_2)\cdots p(X_n) \tag{2.11}$$

Estos son conocidos como modelos univariados y un ejemplo de EDA univariado es UMDA (Mühlenbein y Paass (1996)). Este modelo es muy simple ya que genera nuevas soluciones modelando y muestreando distribuciones de probabilidad univariadas para cada variable del problema, de manera independiente. Otros ejemplos de EDAs univariados son PBIL (Sebag y Ducoulombier (1998)), KEDA (Luo y Qian (2009)) y cGA (Harik et al. (1999)).

Los EDAs que aprovechan las interacciones entre variables son EDAs multivariados. Estos modelos ya no cuentan con variables independientes entre sí. Las dependencias entre variables se pueden representar a través de una red bayesiana:

$$p(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\pi_i) \tag{2.12}$$

siendo π_i un conjunto de variables desde donde existe un arco en la red bayesiana a X_i y $p(X_i|\pi_i)$ es la probabilidad condicional de X_i dado π_i. Algunos ejemplos de EDAs multivariados son EBNA (Larrañaga y Lozano (2001)), EGNA (Larranaga (2000) y Soloviev et al. (2022)), EMNA (Larrañaga y Lozano (2001)), KEDA (Luo y Qian (2009)), SPEDA (Soloviev et al. (2023)), ecGA (Harik (1997)) y BOA (Cantú-Paz et al. (1999)).

2.3. Modelo de Mallows

Las permutaciones son usadas en un amplio rango de dominios, como conjuntos ordenados de objetos, transposiciones, matrices, gráficos, etc. Dado que las permutaciones aparecen en una amplia variedad de dominios, se vio la necesidad de crear un marco probabilístico para abordar la incertidumbre en los espacios de permutación. El coste de tener un valor de probabilidad para cada elemento de cada permutación es inasumible (Irurozki (2014)). Para evitar esto, como primera aproximación se intentaron adaptar distribuciones de probabilidad conocidas para el dominio de las permutaciones. Más adelante empiezan a surgir modelos de probabilidad más precisos en permutaciones. Algunos de estos son modelos multietapa, que asumen que hay una permutación central σ_0, que captura el consenso de una población de permutaciones (Fligner y Verducci (1988)). Otro reconocido modelo es el de Plackett-Luce (Plackett (1975)) en el cual suponemos que hay una figura que hace de juez para generar un *ranking* donde la permutación preferida está en primera posición, la segunda permutación preferida en la segunda y así sucesivamente. Una revisión sobre diferentes modelos de probabilidad basados en permutaciones se puede encontrar en Critchlow et al. (1991). Además en Ceberio et al. (2012) se puede ver otra revisión con distintos modelos de probabilidad aplicados a los EDAs.

El modelo de Mallows (MM) es un modelo probabilístico de permutaciones basado en distancias. Este modelo se expresa de la siguiente forma:

$$p(\sigma) = \frac{exp(-\theta d(\sigma, \sigma_0))}{\psi(\theta)} \tag{2.13}$$

donde $\psi(\theta) = \sum_\sigma exp(-\theta d(\sigma, \sigma_0))$ es la constante de normalización. La permutación central, denotada por σ_0, es la moda de la distribución si el parámetro de dispersión θ, es mayor que cero. En este caso la probabilidad de otra permutación decae exponencialmente con su distancia a σ_0. Y el parámetro de dispersión controla cómo de rápido decae. Por otro lado, cuando $\theta = 0$, obtenemos la distribución uniforme. El cálculo de $\psi(\theta)$ no es asumible para valores de tamaño mediano de n, siendo n el número de elementos de la permutación. Lo bueno es que se puede factorizar para distancias de τ de Kendall y Cayley. El modelo de Mallows es muy similar a la distribución gaussiana para espacios de permutaciones. Estas similitudes se pueden ver en la Figura 2.1 donde se muestra la probabilidad de las permutaciones de tamaño 5 para cada distancia de Kendall y para diferentes valores de θ.

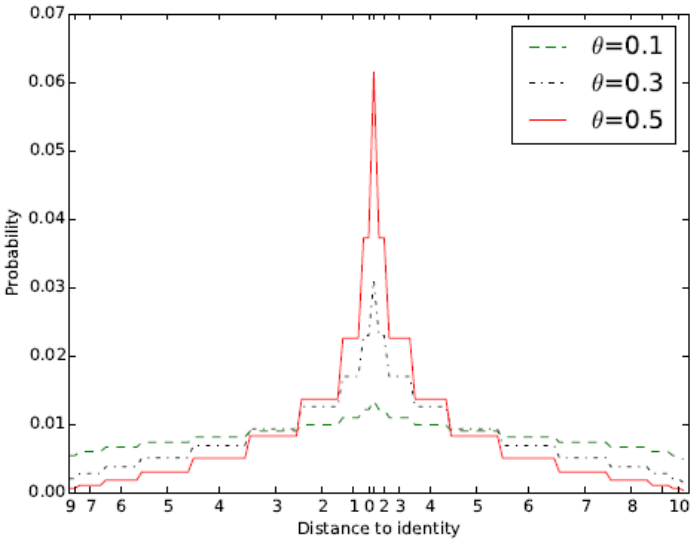


Figura 2.1: Probabilidad de las permutaciones de $n = 5$ para cada distancia y diferentes valores de θ. La permutación identidad es σ_0 (Irurozki (2014))

La función $d(\sigma, \sigma_0)$ es la distancia entre una permutación y la permutación central. En Diaconis (1988) se propuso el uso de diferentes métricas para este modelo. A continuación, se explican las distancias que han sido implementadas. La distancia τ de Kendall cuenta el número de desacuerdos por pares entre dos permutaciones. Se define como el número mínimo de intercambios adyacentes para convertir una permutación en otra. La distancia de Cayley cuenta el número de intercambios (no necesariamente adyacentes) que se han tenido que realizar para transformar una permutación en otra. La distancia de Hamming cuenta el número de posiciones en las cuales dos permutaciones no concuerdan. Y, por último, la distancia de Ulam cuenta la longitud de la subsecuencia más larga y común entre dos permutaciones. Por ello la distancia de Ulam máxima entre permutaciones puede ser $n - 1$.

A partir del modelo de Mallows surge una generalización llamada modelo de Mallows generalizado (GMM). Este trata de eliminar la restricción del modelo de Mallows de tener un mismo valor de dispersión para cada permutación. En vez de esto, se tendrán $n - 1$ parámetros de dispersión por cada permutación. Este enfoque permite destacar el consenso en ciertas posiciones de la permutación mientras se mantiene una mayor

incertidumbre en otras. El modelo se define como sigue a continuación:

$$p(\sigma) = \frac{exp(\sum_{j=1}^{n-1} -\theta_j S_j(\sigma\sigma_0^{-1}))}{\psi(\theta)} \tag{2.14}$$

donde $\psi(\theta) = \sum_\sigma exp(\sum_{j=1}^{n-1} -\theta_j S_j(\sigma\sigma_0^{-1}))$, n es el número de elementos de una permutación, j es el índice de un elemento de una permutación, θ_j es un elemento del vector de parámetros de dispersión. En el caso de la distancia de τ de Kendall, S_j se denota como V_j y se define como $V_j(\pi) = \sum_{i=j+1}^{n} I[\pi(j) > \pi(i)]$, siendo π la permutación, i y j índices de la permutación e $I[condicin]$ un vector booleano que depende de si la condición se cumple o no.

En Ceberio et al. (2011) se presentó un tipo de EDA llamado Mallows EDA. Como dice su nombre, se utilizó el modelo de Mallows como modelo probabilístico en los EDAs. Se realizó un estudio para ver su desempeño en el problema de *Permutation Flowshop Scheduling Problem*, el cual es una adaptación del famoso *Flowshop Scheduling Problem* para permutaciones. Más adelante en Ceberio et al. (2013) se presentó el mismo enfoque pero con el modelo de Mallows generalizado.

2.4. Metaheurísticas basadas en poblaciones para *clustering*

A lo largo de la literatura se han ido viendo enfoques metaheurísticos basados en poblaciones aplicados a *clustering*. Algunos de los enfoques más conocidos son programación evolutiva (EP), algoritmos genéticos (GAs), optimización por enjambre de partículas (PSO), algoritmos de colonias de hormigas (ACO) y EDAs (Saxena et al. (2017)). El algoritmo común a todos estos enfoques metaheurísticos basados en poblaciones para *clustering* es el siguiente:

Algoritmo 4 Pasos de los enfoques evolutivos para *clustering*

1: Elegir una población aleatoria inicial.
2: Asociar un valor de coste a cada individuo de la población. Esto se obtiene a través de una función objetivo que calcula el error cuadrático.
3: Seleccionar los mejores individuos. Un individuo con menor error cuadrático será una mejor solución.
4: Generar nuevos individuos a partir de los individuos seleccionados. Dependiendo del algoritmo elegido se realizará de distinta forma.
5: Evaluar el valor de coste de los individuos generados.
6: Repetir los pasos 4 y 5 hasta que se cumpla una condición de parada.

Estas metaheurísticas inspiradas en la naturaleza y los EDAs son capaces de determinar aspectos como la estructura y el número de grupos de un conjunto de datos. Estos enfoques surgen como alternativa a la necesidad de tener que proporcionar a los algoritmos tradicionales de *clustering* el número de grupos en los que se va a dividir el conjunto de datos y para mejorar la calidad de los grupos creados por estos algoritmos de *clustering* tradicionales. Normalmente solo son los algoritmos de *clustering* particional los que necesitan esta información a priori. Determinar la mejor estimación del número de grupos es llamado problema de *clustering* automático.

Las metaheuristicas basadas en poblaciones se pueden clasificar en EAs y algoritmos de de enjambres de partículas. En Hruschka et al. (2009) y José-García y Gómez-

Flores (2016) se pueden encontrar bastantes enfoques de metaheurísticas aplicadas a *clustering*.

Algoritmos evolutivos y de enjambres de partículas

Dentro de los EAs principalmente destacan los GAs, de los cuales se encuentran muchos enfoques para realizar *clustering* sobre conjuntos de datos.

Los algoritmos genéticos simulan el proceso de evolución biológica con individuos que generalmente son llamados cromosomas. El objetivo del algoritmo es encontrar el mejor individuo. El proceso de selección de este individuo se realiza con una función de evaluación (Holand (1975)). El primer paso del algoritmo es elegir la población y se evalúa cada cromosoma de ésta. Después en función de la evaluación en cada iteración se eligen los padres. Estos padres se cruzan entre si para obtener nuevos cromosomas. Existe una baja probabilidad de que estos nuevos cromosomas sufran una mutación (alterar alguna condición hereditaria). Después de esto algunos individuos son eliminados de la población para reducirla al tamaño original.

Con la operación de cruce se intenta aumentar la calidad media de la población y con el operador de mutación se busca explorar nuevas soluciones candidatas y así conseguir que no se caiga en un mínimo local. La elección de estos operadores es muy importante a la hora de encontrar la mejor solución al problema. Algunos operadores de cruce, presentados en Umbarkar y Sheth (2015) son: El cruce parcialmente mapeado (PMX) que transmite información de orden de padres a hijos. El cruce de ciclo (CX) que intenta que cada hijo contenga cada posición igual que la de uno de sus padres. El operador de cruce de orden (OX1) crea un hijo con la cadena de un padre y con el orden de los elementos del otro padre. El operador cruzado de recombinación de votación (VR) que considera que puede haber dos o más padres. El operador de cruce de posiciones alternas (AP) crea un hijo seleccionando alternativamente el siguiente elemento del primer padre y el siguiente elemento del segundo padre. Algunos operadores de mutación son: El operador de mutación por desplazamiento (DM) selecciona una cadena aleatoria y la inserta en otro lugar. El operador de mutación de intercambio (EM) que intercambia dos elementos de la cadena entre sí. El operador de mutación de inserción (ISM) elige un elemento aleatorio de la cadena, lo borra y lo inserta en otra posición. El operador de mutación de inversión simple (SIM) elige dos puntos de la cadena y cambia el orden de los elementos que se encuentran dentro de estos dos puntos.

En He y Tan (2012) se presentó un enfoque llamado TGCA, el cual puede determinar el número de grupos y las particiones exactas del conjunto de datos. Este enfoque combina el algoritmo k-means con el algoritmo genético. El grado de explotación de la optimización local bajo la misma k se logra mediante k-means. Y el grado de exploración de la optimización global con diferentes k se obtiene mediante el algoritmo genético. El procedimiento del TGCA es el siguiente: El primer paso es realizar la inicialización, donde se ingresa el número de objetos, el número de iteraciones, la probabilidad de cruce y la población. Se utiliza el método de *clustering* de para elegir los centros de los grupos iniciales. El segundo paso consiste en la evaluación de los individuos, donde se obtienen nuevos centros de grupos mediante k-means. Se agrupan los objetos de acuerdo con los nuevos centros y se calcula la aptitud de los individuos. Los mejores individuos se mantienen para la siguiente generación. El tercer paso consiste en generar nuevos individuos mediante una selección de indivi-

duos que consta de dos etapas, un operador de cruce y un operador de mutación que también consta de dos etapas. El último paso consiste en la repetición de estos pasos hasta que se cumpla una condición de parada.

En Doval et al. (1999) se presenta un enfoque para la agrupación automática de sistemas de software utilizando un algoritmo genético. Este enfoque busca optimizar la calidad del agrupamiento de gráficos de dependencia de módulos (MDGs, son estructuras complejas representadas como grafos dirigidos) dividiendo grandes sistemas de software en subsistemas relativamente independientes con alta conectividad interna y baja conectividad entre subsistemas. Cada solución potencial es representada por una cadena codificada que indica la asignación de módulos a grupos. A través de iteraciones de selección, cruce y mutación, el GA mejora iterativamente el agrupamiento de los módulos, maximizando la calidad del *clustering* hasta alcanzar una partición óptima del sistema.

GKA es un enfoque presentado en Krishna y Murty (1999) que combina también el algoritmo k-means con el algoritmo genético. Su procedimiento comienza generando una población inicial de individuos. Cada individuo en la población representa una posible solución de *clustering*. Se deben configurar los parámetros necesarios como la probabilidad de mutación, el tamaño de la población y el número máximo de generaciones. El siguiente paso consiste en calcular la aptitud de todos los individuos de la población con la función objetivo (suma de errores cuadrados) y seleccionar los individuos más aptos. Más adelante se aplica un operador de mutación para evitar óptimos locales y mantener diversidad en la población. Una vez llegado a este punto se aplica lo que llaman el operador k-means (KMO), que consiste en aplicar el algoritmo de k-means para acelerar la convergencia. Si una vez aplicado el operador KMO resultan grupos vacíos, se deben reasignar objetos para que se acaben formando el número de grupos predefinido. Finalmente, los pasos de selección, mutación y refinamiento con k-means se deben repetir hasta que se cumpla un criterio de parada.

En Lu et al. (2004) se presentó FGKA que está inspirado en GKA pero con mejoras para que se reduzca el tiempo de cómputo. Estas mejoras se centran principalmente en tres áreas. La primera es la eficiencia en el cálculo de la variación total dentro de los grupos (TWCV). En FGKA respecto GKA se realiza este cálculo utilizando métodos más eficientes que reducen el tiempo requerido para evaluar la TWCV de cada solución. También se mejora en términos de permitir los individuos ilegales (los que tienen grupos vacíos). Por último, se simplifica el operador de mutación, lo que permite acelerar este proceso. Más adelante se presenta IGKA en **?** y llega para mejorar algunos aspectos de FGKA como: La mejora del rendimiento temporal cuando la probabilidad de mutación es baja, una mejor convergencia al óptimo global, una mayor flexibilidad en la adaptación a diferentes configuraciones de datos y parámetros, y una reducción de la sobrecarga computacional. Además se propone una combinación de los algoritmos IGKA y FGKA para obtener un rendimiento aún mejor, llamado HGKA.

En la literatura se encuentran muchos más enfoques orientados a *clustering* particional (k-means) que a *clustering* jerárquico, pero también se deben destacar algunos enfoques donde se han utilizado metaheurísticas basadas en poblaciones para el *clustering* jerárquico.

Lozano y Larrañaga (1999) presentan un enfoque basado en el algoritmo genético para obtener la clasificación jerárquica óptima de un conjunto de datos. Este algoritmo

denominado enfoque basado en órdenes se basa en una biyección entre el conjunto de clasificaciones jerárquicas de un conjunto de datos y el conjunto de distancias ultramétricas. Esta relación permite medir la calidad de una clasificación jerárquica calculando la norma L2 entre la matriz de distancias ultramétricas asociada a la clasificación jerárquica y la matriz de disimilaridad del conjunto de datos.

Una matriz de distancias ultramétrica es una matriz cuadrada y simétrica que se usa para representar las distancias entre objetos y es denotada por $U = [\delta_{i,j}]$. Dado un conjunto de datos Ω, las propiedades que cumple una matriz de distancias ultramétrica son las siguientes:

1. $\forall \omega, \omega' \in \Omega \implies U(\omega, \omega') = U(\omega', \omega)$,
2. $\forall \omega \in \Omega \implies U(\omega, \omega) = 0$,
3. $\forall \omega, \omega', \omega'' \in \Omega \implies U(\omega, \omega'') \leq \max\{U(\omega, \omega'), U(\omega', \omega'')\}$.

El problema planteado es un ejercicio de optimización donde se busca la distancia ultramétrica más cercana a las disimilitudes del conjunto de datos. Esto se realiza minimizando la suma de las diferencias cuadradas entre las disimilitudes del conjunto de datos ($d_{i,j}$) y las distancias ultramétricas de las clasificaciones jerárquicas ($\delta_{i,j}$). La fórmula a minimizar es la siguiente:

$$\sum_{i<j} (d_{i,j} - \delta_{i,j})^2 \tag{2.15}$$

Los individuos se codifican como una permutación de tamaño $n + (n - 2)$, siendo n el número de objetos del *clustering* jerárquico. Los números del 1 al n codifican el orden de los objetos del conjunto de datos y del $n + 1$ al $2n - 2$ codifican la estructura del dendrograma (clasificación jerárquica). Una vez se tienen definidos, tanto la función objetivo, como la codificación de los dendrogramas que se va a utilizar, solo faltaría aplicar el algoritmo genético.

En Shurman et al. (2013) se presentó un enfoque para optimizar la vida útil de redes de sensores inalámbricos mediante el uso de algoritmos genéticos junto con *clustering* jerárquico. Para abordar este problema se debe entender que el consumo de energía de los sensores inalámbricos depende en gran medida de las comunicaciones de larga distancia entre sensores. El objetivo es reducir estas comunicaciones para prolongar la vida de la red. Esto se va a conseguir agrupando los sensores en grupos. Cada individuo codificado es una lista de parámetros donde el valor 1 indica que un nodo es el cabeza de grupo y 0 que es un nodo regular. Los nodos cabeza de grupo son los nodos que contienen una conexión de larga distancia. El operador de selección utilizado es el método de la ruleta, el de cruce se basa en un punto y el de mutación simplemente se alteran bits aleatoriamente. La función objetivo evalúa la aptitud de un individuo en términos de reducción de distancia de comunicación y minimización del número de nodos cabeza de grupo.

Garai y Chaudhuri (2004) presentan un enfoque llamado HCMA que consiste en un proceso iterativo basado en algoritmos genéticos para combinar fragmentos de grupos generados inicialmente. En este método, cada fragmento de grupo se trata como un objeto único. El algoritmo comienza con una población de individuos, donde cada individuo es una cadena de bits que representa la presencia o ausencia de grupos específicos. Se iteran múltiples veces el algoritmo genético para combinar gradualmente

los fragmentos. El proceso se repite hasta obtener el número deseado de grupos. Cada iteración del algoritmo junta algunos de los grupos fragmentados, manteniendo los grupos fusionados en iteraciones posteriores, optimizando así la estructura del dendrograma final.

Los enfoques que se han visto hasta ahora están todos relacionados con algoritmos genéticos. De todos modos en la literatura también se han desarrollado distintos enfoques de *clustering* basados en algoritmos que no sean evolutivos. A continuación se van a ver algunos ejemplos de cómo se han logrado aplicar algoritmos de inteligencia de enjambres a enfoques de *clustering*.

El algoritmo de optimización por enjambres de partículas (PSO) es un algoritmo de inteligencia de enjambre que contiene partículas dentro de un espacio de solución continua (Lai y Chang (2009)). Cada partícula representa una solución candidata al problema y tienen una posición y velocidad asociada. También saben la mejor partícula del enjambre y su mejor posición previa. El algoritmo comienza inicializando aleatoriamente las posiciones y velocidades de una población de partículas. Después estas partículas se mueven, evaluando la aptitud en cada paso de optimización. Cada partícula compara su valor de aptitud actual con el valor de aptitud de su mejor posición anterior. La partícula también compara su valor de aptitud con el valor de aptitud de la mejor posición global, y si es mejor, entonces esta posición global se actualiza con el valor actual y la posición de la partícula actual. La velocidad y posición de cada partícula se actualizan y por último se termina el proceso si se cumple el criterio de parada.

Omran et al. (2004) fueron los primero en introducir la aplicación del algoritmo PSO para la resolución de problemas de *clustering*. El algoritmo utilizaba un número fijo de grupos y PSO buscaba los centroides óptimos de cada grupo para que después se asignaran cada objeto a los grupos más cercanos. Más adelante en Omran et al. (2006) se extendió este enfoque para realizar un enfoque dinámico de *clustering* basado en el algoritmo de enjambre de partículas, llamado DCPSO.

En Cura (2012) se presentó otro enfoque con características muy similares al anteriormente explicado pero con la diferencia de que este método emplea un algoritmo de PSO puro. Este destaca porque es eficiente, robusto, fácil de ajustar y puede aplicarse tanto cuando se conoce el número de grupos como cuando no.

Uno de los algoritmos que más repercusión ha tenido ha sido el denominado DCPG (Kuo et al. (2012)) por su buen desempeño y por ser un enfoque híbrido que integra el algoritmo de enjambre de partículas y el algoritmo genético. El procedimiento se inicia con una generación aleatoria de centroides de grupos, y la posición y velocidad inicial de las partículas. A través de iterar el algoritmo, PSO ajusta las posiciones y velocidades de las partículas basándose en su mejor posición individual y la mejor posición global. El GA introduce operadores y añade diversificación a las soluciones para escapar de óptimos locales. Y, por último, se emplea el algoritmo de k-means para afinar los resultados obtenidos, corrigiendo los centroides de los grupos. Todos estos pasos se realizan hasta cumplir un número predefinido de iteraciones.

Además de estos algoritmos vistos, cabe destacar la revisión de Alam et al. (2015) ya que en ella se encuentran algunos enfoques de *clustering* jerárquico basado en PSO.

Otro algoritmo de inteligencia de enjambre muy usado es el ACO que fue presentado en Dorigo et al. (1996). ACO es capaz de encontrar el camino más corto entre el nido

y la comida de una colonia de hormigas. Cuando estas van caminando de vuelta al nido van soltando feromonas, que servirán para que otras hormigas decidan qué camino escoger en un futuro. Si no hay feromonas en ningún camino, la ruta se elige aleatoriamente. Las feromonas deben reforzarse, si no acaban desapareciendo. Con este método las hormigas acabarán conociendo todas las rutas y se decantarán por la más corta. Cada hormiga representa una posible solución al problema. La actualización de las feromonas y la selección de la ruta son los dos elementos clave de este problema.

En Kanade y Hall (2003) se emplea ACO para crear inicialmente grupos de datos sin necesidad de conocer el número de grupos a priori. El algoritmo de colonia de hormigas mueve los objetos para formar grupos, cuyos centroides se utilizan como centros del grupo inicialmente. Más adelante, se aplica el algoritmo FCM para refinar estos grupos. Los objetos agrupados por FCM mejoran su calidad según el criterio de máxima pertenencia y se crean nuevos grupos.

Handl et al. (2006) presentaron un enfoque llamado ATTA (*Ant-Based Clustering and Topographic Mapping*). Este algoritmo presenta mejorías en las técnicas de *clustering* basadas en ACO, mediante hormigas adaptativas y heterogéneas, y una actividad de transporte dependiente del tiempo. Las hormigas recogen y depositan objetos para formar montones que representan los grupos de objetos, permitiendo identificar automáticamente el número de grupos necesarios.

Algoritmos de estimación de distribuciones

Los EDAs han sido introducidos en la Sección 2.2. En la literatura se han visto enfoques de EDAs aplicados a *clustering*.

En Cagnini et al. (2016) propusieron un algoritmo de *clustering* basado en un EDA univariado (UMDA), llamado Clus-EDA. Este emplea una representación basada en medoides en la que los prototipos del grupos coinciden necesariamente con los objetos del conjunto de datos. La codificación más usada en el *clustering* particional es la codificación de enteros, que consiste en que cada individuo representa el conjunto de datos y se sabe a qué grupo pertenece cada objeto. El algoritmo Clus-EDA es un algoritmo evolutivo que utiliza un tipo de EDA univariado para explorar el espacio de soluciones candidatas y muestrear soluciones prometedoras del modelo probabilístico. Este EDA emplea un vector de probabilidad $\mathbf{p} = (p_1, p_2, ..., p_n)$ como modelo probabilístico, donde p_i denota la probabilidad de que el objeto x_i sea un medoide. Para inicializar este vector de probabilidades se realiza un paso clave y es que se ejecuta el algoritmo de k-means múltiples veces. La función objetivo se utiliza para evaluar la calidad de las particiones del *clustering* y en este caso se eligió una función con complejidad lineal, llamada *Simplified Silhouette Width Criterion*.

El problema más común de los algoritmos de *clustering* es que requieren que el usuario introduzca el número k de grupos de forma predefinida. Además de otros parámetros como el radio de los grupos, el número mínimo de objetos u otros parámetros similares. También otro problema común es la forma de los grupos. En Cagnini y Barros (2016) se abordaron estos problemas con un enfoque que usa EDAs, llamado *PArameterless Shape-independent clustering ALgorithm* (PASCAL). La codificación de los individuos en este enfoque juega un papel muy importante para resolver estos problemas. Los individuos son árboles de expansión mínimos (MSTs), que son

creados basándose en la idea de que cada objeto esta unido por un arco al resto de objetos. Como es intuitivo, no se necesitan todos los arcos, por lo tanto se prescinden de los arcos más largos (los objetos que están más separados). Así que en vez de construir un grafo completamente conectado se construye un MST donde todos los objetos están conectado por su arcos más pequeños. La función objetivo usada se basa en un criterio de densidad.

En Meiguins et al. (2010) se propuso un enfoque basado en EDAs para la generación artificial de algoritmos de *clustering* basados en densidad. Este enfoque llamado *AutoClustering*, trata de aumentar las posibilidades de obtener un buen algoritmo de *clustering*, así que para ello se definen una serie de procedimientos de *clustering* llamados bloques de construcción. De forma adicional el EDA usa una estructura de datos predefinida que representa todas las secuencias válidas de los bloques de construcción. Esta estructura de datos es un grafo acíclico dirigido, donde los nodos representan los bloques de construcción y las flechas representan una posible secuencia de ejecución. Los individuos usados en el EDA son algoritmos de *clustering* basados en densidad y los algoritmos, como ya se ha explicado, son cualquier secuencia válida de bloques de construcción. Para estimar la calidad de los grupos, en este enfoque se utiliza un método llamado método Clest.

Fan (2019) presenta un enfoque llamado OPE-HCA que junta la estimación probabilística con el *clustering* jerárquico aglomerativo. La estimación óptima probabilística (OPE) se basa en los algoritmos de estimación de distribuciones tratando de estimar el mejor muestreo de soluciones a partir de las formadas mediante la fusión entre de las subsoluciones que tienen máximas probabilidades entre todas las otras subsoluciones. Esta idea se realiza con el fin de obtener el grupo con más probabilidades de ser correcto. Si lo comparamos con los métodos de *clustering* basados en distancias, estos son buenos reconociendo grupos de objetos que están más cercanos. Sin embargo, el enfoque basado en probabilidad es una técnica alternativa para encontrar grupos similares en cualquier posición en el espacio, ya que no necesita calcular distancia sino solo considerar la frecuencia. OPE considera que los grupos con altas probabilidades son los que pertenecen al mismo grupo.

Por lo tanto, el algoritmo OPE-HCA está basado en el procedimiento de OPE. Este algoritmo se basa en cuatro pasos. Lo primero es calcular las distancias entre los datos con la fórmula de la distancia Euclídea. El segundo paso es fusionar dos puntos del conjunto de datos en un mismo grupo si la distancia entre estos dos puntos es menor o igual que θ, siendo θ un parámetro de distancia. De este modo generar k grupos para obtener la vista del grupo de origen $S1$. El tercer paso es asignar la mínima distancia de fusión entre grupos. Y, por último, como resultado se obtendría las vistas del *clustering*.

Para finalizar esta sección se presenta una tabla, a modo de resumen, donde se encuentran los enfoques vistos junto con sus referencias.

Nombre del enfoque	**Referencia**
TGCA	He y Tan (2012)
Clustering automático sistemas software	Doval et al. (1999)
GKA	Krishna y Murty (1999)
FGKA	Lu et al. (2004)
IGKA	**?**
HGKA	**?**
Enfoque GA basado en orden	Lozano y Larrañaga (1999)
HCMA	Garai y Chaudhuri (2004)
DCPSO	Omran et al. (2006)
DCPG	Kuo et al. (2012)
ATTA	Handl et al. (2006)

Cuadro 2.3: Resumen de los enfoques del estado del arte de las metaheurísticas basadas en poblaciones para *clustering*

Capítulo 3

Metodología propuesta

El algoritmo que se presenta en este apartado consiste en un EDA aplicado a *clustering* jerárquico. Este algoritmo se basa en el enfoque realizado por Lozano y Larrañaga (1999) e introducido en la Sección 2.4. El enfoque consistía en un algoritmo genético aplicado a *clustering* jerárquico, por lo que será relativamente sencillo adaptar algún apartado para el uso de EDAs en lugar del uso de un algoritmo genético.

La idea de aplicar EDAs al *clustering* jerárquico surge con la intención de mejorar la calidad de los grupos que se crean con el algoritmo de *clustering* jerárquico tradicional. Además de esto, se debe poder utilizar cualquier conjunto de datos y obtener subconjuntos de buena calidad. Para poder tener una visión general de cómo funciona el algoritmo, hay que entender su flujo de ejecución. El algoritmo propuesto es el siguiente:

Algoritmo 5 Pasos del enfoque propuesto

Input: Tamaño de la población, conjunto de datos
Output: Mejor individuo, coste
1: Lista de estructuras
2: **for** estructura **in** lista de estructuras **do**
3: Definir la función objetivo con la estructura
4: Ejecutar el EDA (Algoritmo 3) con modelo de Mallows
5: **end for**
6: Seleccionar el mejor individuo

Para ser capaces de comprender mejor el algoritmo propuesto hay que entender cada paso, por lo que esta sección se va a dividir en diferentes explicaciones. La primera y una de las más importantes es la representación de los individuos. En este subapartado se va a entender cómo se codifican los dendrogramas del *clustering* jerárquico. Se continuará explicando el generador de estructuras, que resumiendo consiste en un generador aleatorio de dendrogramas. Después se verá en detalle el Algoritmo 5, ya introducido. Se dejarán claros los apartados relacionados con la generación de la población inicial, la selección de los individuos y, por último, el aprendizaje y muestreo de la distribución de probabilidad usada.

3.1. Representación de individuos

La representación de los individuos se basa en ser capaces de codificar un dendrograma que representa las distintas agrupaciones de un conjunto de datos. Un dendrograma es un diagrama en forma de árbol binario que muestra la disposición jerárquica de los objetos agrupados. Este representa las relaciones de similitud entre los objetos mediante ramas que se dividen progresivamente, agrupando los elementos en distintos niveles. El *clustering* jerárquico a diferencia del particional usa los dendrogramas como resultado final de los agrupamientos para representar las uniones/divisiones de objetos o grupos de objetos que se han ido realizando. Se realizan uniones cuando nos encontramos en un algoritmo de *clustering* jerárquico aglomerativo, y divisiones cuando el algoritmo es divisivo (véase Sección 2.1.2). El tipo de algoritmo jerárquico elegido es indiferente en lo que se refiere al resultado ya que siempre se va a obtener un dendrograma.

Para que estos dendrogramas sean más manejables se codifican en vectores, por lo que el objetivo es ser capaces de representar fielmente la estructura de un dendrograma en un vector. Se debe tener algunas cosas en cuenta como que hay que mantener, de forma integra, el orden de los objetos del dendrograma. Además se debe representar la estructura del dendrograma en el mismo vector. La representación de la estructura se va a realizar tomando la distancia que hay entre los objetos ordenados. Se entiende por distancia el número de nodos que hay entre un objeto y el siguiente. En la figura 3.1 se puede ver un ejemplo de dendrograma, el cual ha sido generado aleatoriamente con la función *dendrogram* de la librería *scipy* de python [Virtanen et al. (2020)]. Por ejemplo, la distancia que hay entre los dos primeros objetos, el número 1 y el 0, del dendrograma de la Figura 3.1 es 4. Dicho de otra forma, hay 4 nodos en el camino que une los dos primeros objetos del dendrograma. Así que el vector que representa la estructura del dendrograma es $[4, 2, 4, 2]$.

Por lo tanto, la representación de los individuos será la concatenación del vector de orden de los objetos y el vector de la estructura del dendrograma. El tamaño de los individuos será de $2n-1$, siendo n el número de objetos de un problema en específico. El orden es el de los objetos que van desde x_1 a x_n. Y la estructura del dendrograma con los elementos del x_{n+1} a x_{2n-1}.

$$individuo = [x_1, ..., x_n, x_{n+1}, ..., x_{2n-1}] \tag{3.1}$$

Una vez explicada la representación de los individuos y volviendo al dendrograma de ejemplo de la Figura 3.1, su codificación será $[1, 0, 4, 2, 3, 4, 2, 4, 2]$.

3.2. Generador aleatorio de dendrogramas

El generador se corresponde con el paso previo al EDA, y consiste en generar aleatoriamente estructuras de dendrogramas de un número de objetos determinado. Para este apartado se implementaron dos soluciones.

La primera de ellas consistía en generar matrices de adyacencia de forma aleatoria y válidas, que representen a los dendrogramas. Las matrices de adyacencia de un grafo son matrices cuadradas que se utilizan para representar relaciones binarias, por lo que se pueden usar para representar dendrogramas, que al fin y al cabo son árboles binarios. Una de las características negativas de este generador es el consumo

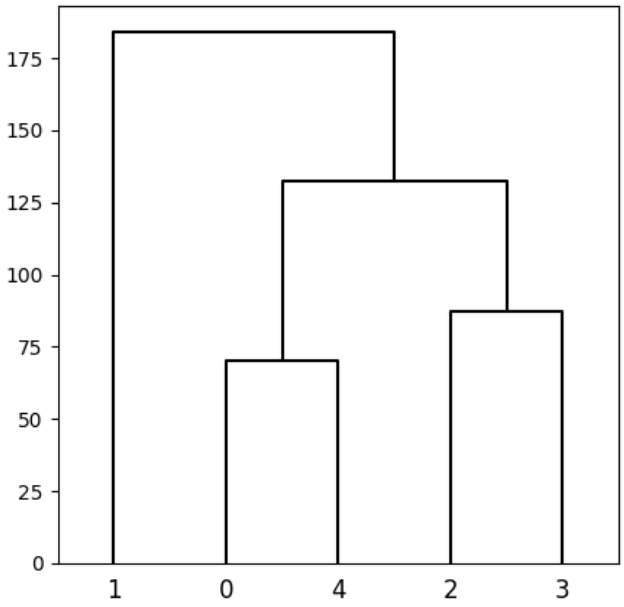


Figura 3.1: Ejemplo de dendrograma

de memoria, debido a que las matrices de adyacencia crecen cuadráticamente en memoria según crece el número de objetos en el problema. Otro punto negativo es la dificultad de obtener matrices de adyacencia de forma aleatoria que sean válidas. Para que una matriz de adyacencia sea válida para representar un dendrograma debe cumplir una serie de restricciones:

1. Es una matriz de dimensiones $(2n-2) \times (n-1)$, siendo n el número de objetos
2. En cada columna tiene que haber dos 1s
3. En cada fila tiene que haber un 1
4. En la última columna tiene que haber al menos un 1 a partir de la fila $n+1$
5. No puede haber ciclos, es decir, un nodo no puede tener conectado una rama que sale de si mismo. Se debe cumplir que sea un árbol.

A continuación se muestra como sería la matriz de adyacencia válida del dendrograma de la Figura 3.1. Se puede comprobar cómo se cumplen las restricciones anteriores. En la la matriz de adyacencia del Cuadro 3.1, los nodos Y se refieren a los nodos internos del dendrograma y los nodos X a los nodos hojas, los cuales son los objetos del problema de *clustering*.

Una vez se han generado m matrices de adyacencia que representen dendrogramas válidos, solo queda codificarlas con la representación de individuos que se ha visto en la Sección 3.1.

La segunda solución para generar aleatoriamente estructuras de dendrogramas mejora tanto en tiempo, como en memoria al método propuesto anteriormente. Este generador también utiliza matrices de adyacencia, pero simplificadas (3.2) para ahorrar

	Nodo Y_1	Nodo Y_2	Nodo Y_3	Nodo Y_4
Nodo X_1	0	0	0	1
Nodo X_2	1	0	0	0
Nodo X_3	1	0	0	0
Nodo X_4	0	1	0	0
Nodo X_5	0	1	0	0
Nodo Y_1	0	0	1	0
Nodo Y_2	0	0	1	0
Nodo Y_3	0	0	0	1

Cuadro 3.1: Matriz de adyacencia del dendrograma de la Figura 3.1

en memoria. El procedimiento se basa en generar y podar matrices de adyacencia extendidas. Estas matrices de adyacencia de tamaño mayor, son generadas a partir de una matriz de adyacencia más pequeña. Este paso trata de replicar lo que sería añadir más nodos hoja a un dendrograma de n objetos y seguidamente podar el mismo número de nodos hoja que se han añadido. Como resultado se obtiene un dendrograma del mismo tamaño inicial. Con estos pasos se consigue generar constantemente dendrogramas válidos y diferentes. Cuanto mayor sea el número de nodos que se extiendan del dendrograma inicial, mayor será el número de dendrogramas diferentes entre sí.

Nodo X_1	Nodo X_2	Nodo X_3	Nodo X_4	Nodo X_5	Nodo Y_1	Nodo Y_2	Nodo Y_3
Y_4	Y_1	Y_1	Y_2	Y_2	Y_3	Y_3	Y_4

Cuadro 3.2: Vector de adyacencia del dendrograma de la Figura 3.1

Una vez generada cada matriz de adyacencia de un dendrograma se codifica con la representación vista en la Sección 3.1. Por último, se aplica un filtro de eliminación de representaciones iguales para poder asegurar que el algoritmo presentado no itera sobre la misma estructura de dendrograma.

3.3. Optimizador (EDAs)

El optimizador es el EDA, cuya función es encontrar el individuo óptimo. Cuando se habla del individuo óptimo en el contexto del *clustering* jerárquico, se hace referencia al dendrograma más cercano en similitud al conjunto de datos.

Para entender mejor la estructura de la metodología propuesta y dónde se sitúa los EDAs en ella, se puede ver la Figura 3.2. En ella se distinguen dos partes que corresponden con las dos partes que tiene la representación de los individuos. Una parte contiene una lista de estructuras de dendrogramas codificadas que han sido generadas previamente tal y como se detalla en la Sección 3.2. Se itera sobre cada estructura, ejecutando un EDA por cada una de ellas, con el fin de cubrir el mayor espacio de soluciones posible. Y la otra parte se corresponde con la primera mitad de los individuos codificados, el orden de los objetos. El EDA se usa para encontrar el orden con el cual el individuo obtiene una mejor aptitud. Se puede ver el pseudocódigo del enfoque presentado en el Algoritmo 5.

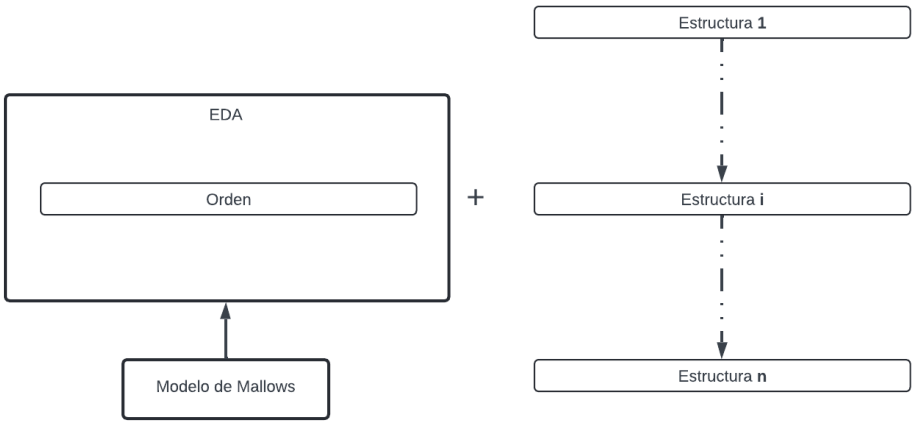


Figura 3.2: Diagrama de la metodología propuesta

El EDA que se propone en este trabajo es un EDA basado en permutaciones. Anteriormente en la literatura se habían visto algunos algoritmos para desarrollar EDAs basados en permutaciones, tanto en dominios discretos como continuos. Estos enfoques tenían muchas limitaciones, así que se animó a la comunidad de investigación de EDAs a buscar nuevas soluciones para problemas basados en permutaciones.

Por este motivo se propone usar un EDA con el modelo de Mallows, introducido en la Sección 2.3. El desarrollo de este nuevo algoritmo se va a realizar con EDAspy [Soloviev et al. (2024)], una libreria de python que contiene muchas implementaciones de EDAs. Este enfoque ha sido llamado EDAPerm y en los siguientes apartados se detallarán los aspectos más destacables.

3.3.1. Selección de individuos

Para asegurar la diversidad poblacional y cubrir una amplia gama del espacio de búsqueda desde el principio, cada individuo de la población inicial es generado como una permutación aleatoria.

La selección de los individuos es el paso más importante para poder utilizar un EDA para resolver un problema de *clustering* jerárquico. Este paso se corresponde con el cálculo de la aptitud o coste de cada individuo. Como se comentó anteriormente, EDAPerm se basa en el enfoque presentado por Lozano y Larrañaga (1999). Uno de los puntos en los que se basa es en la función objetivo usada para seleccionar los individuos. Esta es una función de errores al cuadrado con la que se cuantifica la desviación cuadrática entre los valores de disimilitud del conjunto de datos y los valores de la matriz ultramétrica que representa a cada dendrograma. La función es la que se muestra a continuación:

$$\sum_{i<j}(d_{i,j} - \delta_{i,j})^2 \tag{3.2}$$

Siendo $d_{i,j}$ los elementos de la matriz de disimilaridad del conjunto de datos y $\delta_{i,j}$ de la matriz de distancias ultramétricas del dendrograma a evaluar. El objetivo es minimizar esta función.

Cabe destacar que para poder obtener la matriz ultramétrica de un dendrograma, previamente se han tenido que juntar las dos partes que representan cada individuo (orden y estructura) para poder obtener el dendrograma que codifica ese individuo. Los costes obtenidos de cada individuo se utilizan para seleccionar cuáles son los mejores.

3.3.2. Estimación y muestreo de la distribución de probabilidad

La distribución de probabilidad que se usa en este EDA basado en permutaciones es el modelo de Mallows, presentado en la Sección 2.3. El modelo de Mallows es un modelo de probabilidad exponencial basado en distancias el cual puede capturar eficazmente la estructura de las permutaciones. Dada una distancia sobre permutaciones se pueden definir dos parámetros: la permutación central σ_0 y el parámetro de dispersión θ. Estos son los parámetros que se deben estimar a partir de distintas permutaciones, para posteriormente ser capaces de muestrear nuevas permutaciones con una mayor aptitud.

El modelo de Mallows no se ciñe a una sola distancia en específico, puesto que a lo largo de la literatura se han podido ver distintas implementaciones con varias distancias como Kendall, Cayley o Spearman. En Irurozki et al. (2016) se presentó un paquete de R llamado PerMallows el cual implementa el modelo de Mallows, para su aprendizaje y posterior muestreo de permutaciones. Este paquete tiene implementadas las distancias de Kendall, Cayley, Ulam y Hamming para realizar el aprendizaje de los parámetros de forma exacta o aproximada. Además del modelo de Mallows se encuentra la implementación del modelo de Mallows generalizado que presenta algunos cambios vistos en la Sección 2.3 con respecto al modelo principal.

En el siguiente capítulo se van a ver los resultados de las pruebas realizadas con estas distancias mencionadas y ambos modelos de probabilidades vistos.

Capítulo 4

Resultados

4.1. Implementaciones

La implementación de la metodología propuesta, llamada EDAPerm, se ha realizado en python con la librería de EDAspy (Soloviev et al. (2024)) y el paquete PerMallows de R (Irurozki et al. (2016)). Además de esta metodología, a modo de prueba, se implementó un algoritmo de estimación de distribuciones gaussianas (EGNA) con la codificación de los individuos propuesta en Romero et al. (2004) y ha sido llamada EGNAPerm. Esta representación consiste en aproximar una representación continua a un dominio discreto para obtener una permutación. Un ejemplo de esta aproximación de un dominio continuo a uno discreto sería $[0.5, 1.6, 0.2, 0.1]$ siendo el individuo continuo y $[3, 4, 2, 1]$ siendo el individuo discreto (permutación). El código de estas implementaciones se puede encontrar en GitHub[1]. Además, en cada fichero, se encuentra el código de la función objetivo que se utiliza para seleccionar los individuos, junto con los algoritmos de codificación y decodificación de estos.

Las pruebas han sido realizadas en un ordenador con procesador Intel i5-11400F con 8 GB de memoria RAM y paralelamente en otro ordenador con un procesador AMD Ryzen 5 3500U con 8GB de memoria RAM. Además las pruebas realizadas con el EGNA han sido ejecutas en el centro de computación magerit (CeSViMa) de la UPM.

4.2. Conjuntos de datos

Los conjuntos de datos que se han seleccionado para la realización de las diferentes pruebas son tres. Cada uno de estos tres conjuntos de datos tiene características diferentes y han sido diseñados para la realización de tareas distintas y no todos específicamente para *clustering*.

En el Cuadro 4.1 se pueden observar las características de los conjuntos de datos una vez han sido limpiados. El primer conjunto usado es un conjunto diseñado para tareas de clasificación de tipos de coches (Bart (2015)). De la misma forma que el tercer conjunto de datos fue diseñado también para tareas de clasificación (Chaudhury (2020)). Sin embargo, el segundo conjunto de datos que contiene instancias de pacientes con y sin cáncer cervical, fue específicamente creado para realizar *clustering*

[1] https://github.com/marioruub/EDAs-for-hierarchical-clustering

(Kelly et al. (2023)).

Al fin y al cabo si el objetivo principal de este trabajo es evaluar el enfoque propuesto con respecto al algoritmo de *clustering* jerárquico tradicional, no es necesario usar conjuntos de datos específicamente diseñados para la tarea de *clustering*.

Si realizamos una visualización rápida de estos conjuntos, se puede asumir que en el conjunto de datos de cáncer cervical, al tener muchas menos instancias, la aptitud de los individuos va a ser mucho menor que los del conjunto de datos de los coches y de la movilidad geográfica con el covid-19. Por ejemplo en un problema de *clustering* con 20 objetos, si extraemos 20 instancias de un conjunto de datos con pocas instancias de por sí, se obtendrán individuos con menor aptitud.

Conjuntos de datos	**#Instancias**	**#Características**
Coches	5.076	18
Cáncer cervical	72	20
Movilidad covid	23.625	8

Cuadro 4.1: Conjuntos de datos

4.3. Pruebas con distintas distancias

El paquete de PerMallows de Irurozki et al. (2016) se ha usado para implementar el modelo de Mallows y el modelo de Mallows generalizado en EDAPerm. Este paquete además contiene varias distancias implementadas con las cuales se han hecho pruebas para determinar con cuál se obtenían mejores combinaciones.

Se han realizado pruebas con todas las combinaciones de ambos modelos de probabilidad y distancias disponibles en PerMallows. Hay combinaciones que no son posibles usar con algunos modelos ya que todavía no han sido implementadas. Además se han realizado pruebas con una estimación aproximada y exacta de los parámetros del modelo de Mallows. La estimación exacta solo se ha podido probar en problemas con 10 objetos ya que el tiempo de computo ascendía considerablemente al aumentar el número de objetos.

Las características de las pruebas realizadas en este apartado han sido ajustadas para la comparación de ambos modelos de probabilidad y las distancias vistas. Con las características de las pruebas se hace referencia a las características de los EDAs utilizados. Cada EDA se ejecuta como máximo 40 iteraciones y se para de ejecutar si en 20 de estas iteraciones no se ha mejorado el coste. El tamaño de la generación usada en el EDA es de $50n$, siendo n el número de objetos. Y el número de estructuras de dendogramas usadas es de $10n$. Como ya se había explicado previamente en el Capítulo 3 el número de estructuras de dendogramas usadas se corresponde con el número de EDAs que se ejecutan, para poder cubrir el máximo espacio de soluciones posible.

Se han realizado pruebas de 10, 15 y 20 objetos, con los tres conjuntos de datos introducidos en el Cuadro 4.1. También cabe destacar que se han realizado 5 ejecuciones por cada prueba

Pruebas con 10 objetos

En los Cuadros 4.2, 4.3 y 4.4 se pueden observar los resultados de las pruebas que se han realizado con 10 objetos. De primeras llama la atención la diferencia tan grande de costes usando un conjunto de datos u otro. Esto es debido a que al coger solo 10 instancias en conjuntos de datos con muchas más, el agrupamiento en grupos similares es una tarea más compleja. El conjunto de datos de los coches contiene más de 5.000 instancias y el conjunto de datos del cáncer cervical solo 72, por lo que se verifica lo explicado anteriormente.

Se consideran mejores resultados los que tienen un menor coste. Esto significa que hay más similitud entre el conjunto de datos y el dendograma que representa la agrupación de esos datos. Observando los resultados obtenidos con los tres conjuntos de datos se puede ver que no hay un patrón claro que indique que con una distancia y un modelo especifico se obtengan mejores resultados. En negrita se marca la mejor solución obtenida.

Modelo	**Estimación**	**Distancia**	**Coste (Coches)**
MM	Aproximada	Kendall	**114.830,18 ± 2.901,47**
		Cayley	115.749,28 ± 3.493,19
		Ulam	114.918,44 ± 2.407,92
	Exacta	Hamming	118.227,27 ± 4.675,11
		Cayley	119.949,11 ± 4.267,54
GMM	Aproximada	Kendall	117.918,23 ± 4.194,83
		Cayley	117.756,99 ± 2.157,39
		Hamming	115.917,75 ± 3.457,17
	Exacta	Cayley	116.634,6 ± 3.739,32

Cuadro 4.2: Resultados conjunto de datos de coches con 10 objetos

Modelo	**Estimación**	**Distancia**	**Coste (Cáncer cervical)**
MM	Aproximada	Kendall	329,72 ± 3,2
		Cayley	324,01 ± 0,86
		Ulam	340,24 ± 3,85
	Exacta	Hamming	324,51 ± 1,5
		Cayley	327,49 ± 5,41
GMM	Aproximada	Kendall	328,44 ± 5,06
		Cayley	**322,79 ± 0**
		Hamming	324,95 ± 1,92
	Exacta	Cayley	324,34 ± 2,2

Cuadro 4.3: Resultados conjunto de datos de cáncer cervical con 10 objetos

Se pueden observar los diagramas de cajas en la Figura 4.1 para ver los resultados de las pruebas realizadas con 10 objetos de una forma más visual. El eje y de estos diagramas representa el mejor coste o aptitud obtenido con el modelo probabilístico y distancia especificados. Como se había comentado no se ha logrado encontrar un patrón que indique que con una distancia y un modelo en específico se obtienen mejores resultados. Lo que se puede destacar es que en el conjunto de datos de coches, se observa que la mayoría de los algoritmos muestran costes medianos (óptimos) similares, pero con diferencias significativas en la dispersión. Esto es menos notorio en

Modelo	Estimación	Distancia	Coste (Movilidad covid)
MM	Aproximada	Kendall	27.981,99 ± 431,88
		Cayley	29.013,84 ± 709,87
		Ulam	30.317,11 ± 273,9
	Exacta	Hamming	27.653,27 ± 167,27
		Cayley	28.795,86 ± 252,68
GMM	Aproximada	Kendall	28.511,14 ± 776,29
		Cayley	28.309,58 ± 174,2
		Hamming	27.757,27 ± 481,86
	Exacta	Cayley	**27.496,53 ± 113,11**

Cuadro 4.4: Resultados conjunto de datos de movilidad covid con 10 objetos

los otros conjuntos de datos ya que la dispersión de los resultados por lo general es menor.

Pruebas con 15 objetos

Las pruebas realizadas con los mismos conjuntos de datos pero con 15 objetos se pueden observar en los Cuadros 4.5, 4.6 y 4.7. En este caso sí se puede apreciar que con el modelo de Mallows generalizado y la distancia de Hamming se obtienen los mejores resultados.

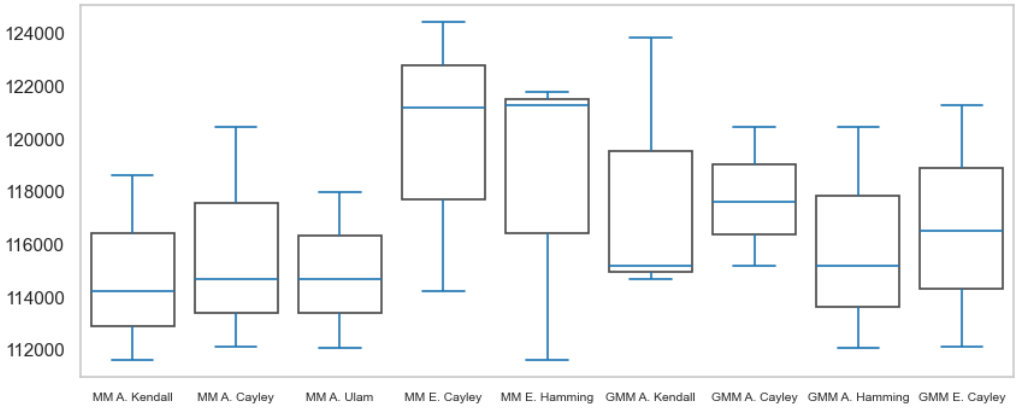
Modelo	Estimación	Distancia	Coste (Coches)
MM	Aproximada	Kendall	433.140,45 ± 19.034.19
		Cayley	441.392,88 ± 2.170,92
		Ulam	421.276,55 ± 10.333,29
GMM	Aproximada	Kendall	435.613,64 ± 6.994,89
		Cayley	438.543,07 ± 14.831,29
		Hamming	**398.767,07 ± 5.556,58**

Cuadro 4.5: Resultados conjunto de datos de coches con 15 objetos

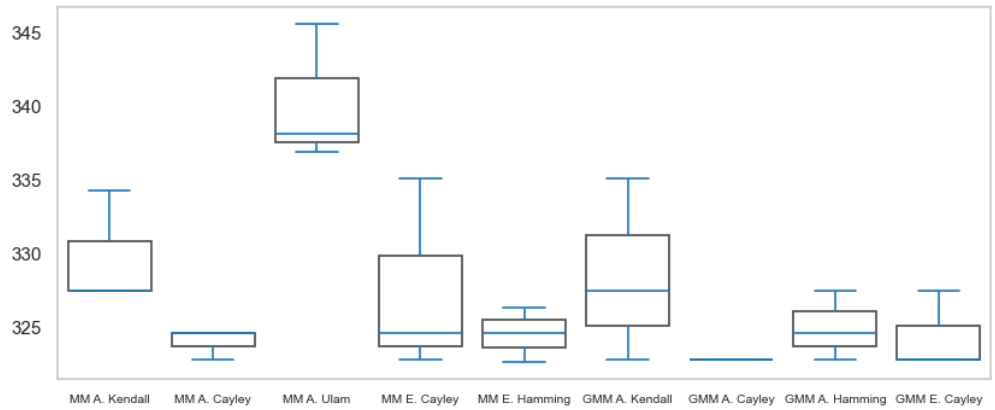
Modelo	Estimación	Distancia	Coste (Cáncer cervical)
MM	Aproximada	Kendall	1.352,62 ± 29,3
		Cayley	1.371,22 ± 37,9
		Ulam	1.449,51 ± 39,01
GMM	Aproximada	Kendall	1.468,22 ± 3,4
		Cayley	1.295,42 ± 36,76
		Hamming	**1.200,05 ± 22,35**

Cuadro 4.6: Resultados conjunto de datos de cáncer cervical con 15 objetos

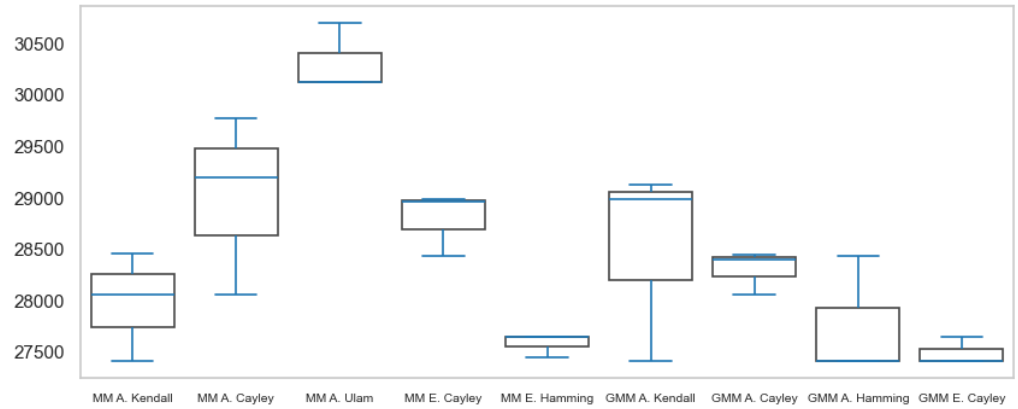
Se puede observar mejor y de una forma más visual en los diagramas de cajas de la Figura 4.2. Se ve que la diferencia de usar GMM con Hamming a no usarlo es bastante grande, ya que se reduce considerablemente el coste calculado. La dispersión de los resultados obtenidos en los tres conjuntos de datos es similar, excepto en el conjunto de cáncer cervical con GMM y Kendall que es notablemente inferior.



(a) EDAPerm para conjunto de datos de coches

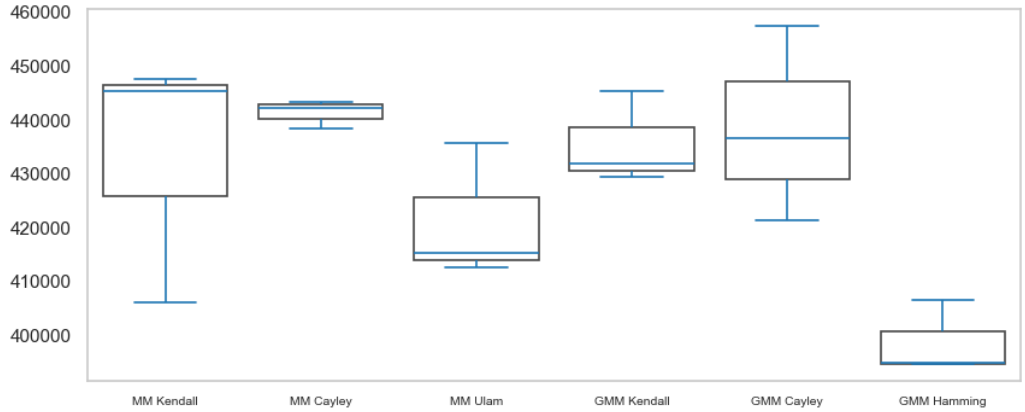


(b) EDAPerm para conjunto de datos de cáncer cervical

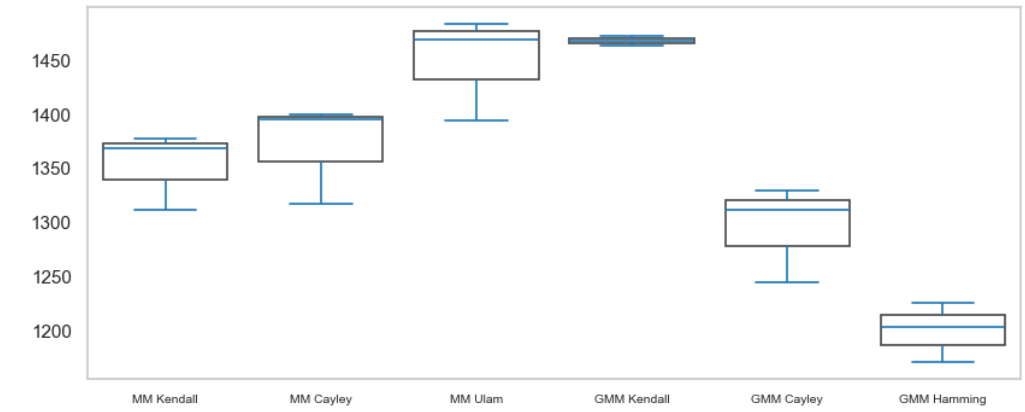


(c) EDAPerm para conjunto de datos de movilidad covid

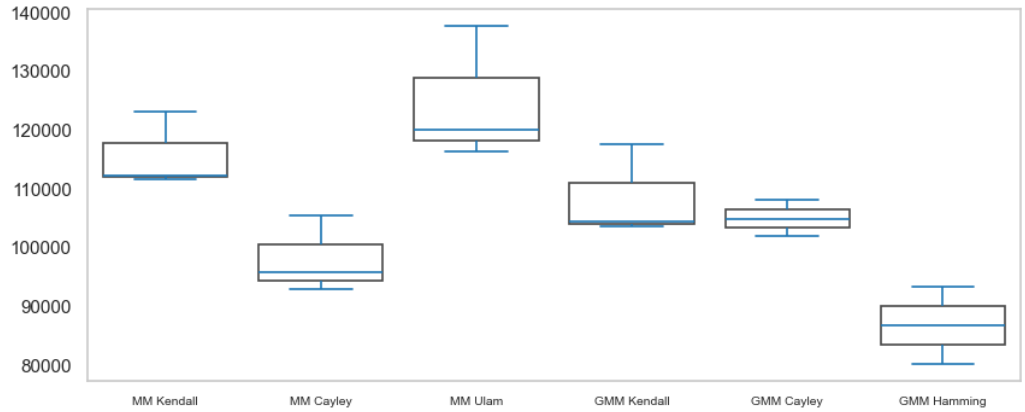
Figura 4.1: Diagramas de caja con conjuntos de datos de 10 objetos



(a) EDAPerm para conjunto de datos de coches



(b) EDAPerm para conjunto de datos de cáncer cervical



(c) EDAPerm para conjunto de datos de movilidad covid

Figura 4.2: Diagramas de caja con conjuntos de datos de 15 objetos

Modelo	**Estimación**	**Distancia**	**Coste (Movilidad covid)**
MM	Aproximada	Kendall	115.757,57 ± 5.234,36
		Cayley	98.091,2 ± 5.391,66
		Ulam	124.693,38 ± 9.360,34
GMM	Aproximada	Kendall	108.585,88 ± 6.380,52
		Cayley	105.044 ± 2.530,29
		Hamming	**86.804,34** ± **5.342,58**

Cuadro 4.7: Resultados conjunto de datos de movilidad covid con 15 objetos

Pruebas con 20 objetos

En los Cuadros 4.8, 4.9 y 4.10 se pueden ver los resultados numéricos de los tres conjuntos de datos con 20 objetos. Al igual que en las pruebas con 15 objetos, el modelo de Mallows generalizado con la distancia de Hamming ha obtenido los mejores resultados.

Modelo	**Estimación**	**Distancia**	**Coste (Coches)**
MM	Aproximada	Kendall	917.853,86 ± 24.652,45
		Cayley	882.113,81 ± 14.619,34
		Ulam	917.799,7 ± 4.027,47
GMM	Aproximada	Kendall	919.812,84 ± 6.331,57
		Cayley	848.890,91 ± 25.011,76
		Hamming	**752.712,28** ± **4.646,66**

Cuadro 4.8: Resultados conjunto de datos de coches con 20 objetos

Modelo	**Estimación**	**Distancia**	**Coste (Cáncer cervical)**
MM	Aproximada	Kendall	2.807,96 ± 58,51
		Cayley	2.775,67 ± 88,17
		Ulam	2.775,56 ± 83,99
GMM	Aproximada	Kendall	2.729,13 ±101,72
		Cayley	2.602,08 ± 120,8
		Hamming	**2.069,16** ± **43,72**

Cuadro 4.9: Resultados conjunto de datos de cáncer cervical con 20 objetos

Modelo	**Estimación**	**Distancia**	**Coste (Movilidad covid)**
MM	Aproximada	Kendall	202.561,51 ± 2.394,64
		Cayley	199.117,6 ± 2.930,71
		Ulam	199.271,53 ± 9.390,9
GMM	Aproximada	Kendall	201.720 ± 8.536,8
		Cayley	182.198,67 ± 8.700,84
		Hamming	**173.136,86** ± **3.916,14**

Cuadro 4.10: Resultados conjunto de datos de movilidad covid con 20 objetos

Si se desea ver en los diagramas de cajas se puede observar la Figura 4.3, donde se aprecia una gran diferencia de costes entre el modelo de Mallows generalizado con Hamming y los demás. También se debe destacar que la dispersión de las pruebas

del conjunto de coches es notablemente menor con MM y Ulam, con GMM y Kendall, y con GMM y Hamming.

4.4. Resultados frente al algoritmo de *clustering* jerárquico tradicional

En este apartado se van a presentar las pruebas realizadas para comparar los enfoques de EDAPerm y EGNAPerm con el algoritmo tradicional de *clustering* jerárquico.

El algoritmo de *clustering* jerárquico usado es el impletado en la libreria de *scipy* de python, en la función *linkage*. Se ha usado el método de enlace de Ward y como métrica la distancia euclídea. Las características de los EDAs usados en los enfoques de EDAPerm y EGNAPerm son las mismas que las usadas en la Sección 4.3 excepto que el número de estructuras de dendogramas usadas en este caso es de $50n$. Este cambio es debido a qué para plantarle cara al algoritmo de *clustering* jerárquico se debe aumentar considerablemente el espacio de búsqueda de soluciones.

En la Sección anterior se compararon el modelo de Mallows y su enfoque generalizado con sus respectivas distancias implementadas en el paquete de PerMallows. LLegamos a la conclusión de que por lo general el modelo que mejor resultados da es el modelo de Mallows generalizado y la distancia es Hamming. Así que en esta sección en las pruebas que se han realizado se han usado esas características.

Se han realizado pruebas con los tres conjuntos de datos presentados y con el número de 10, 15 y 20 objetos. En este caso se han realizado 3 ejecuciones por cada prueba. A continuación se van a ir viendo los diferentes resultados obtenidos por cada conjunto de datos en estas pruebas. En los Cuadros 4.11, 4.12 y 4.13 se puede ver los resultados.

#Objetos	**Coste EDAPerm**	**Coste EGNAPerm**	**Coste *clustering***
10	114.830,18 ± 2.901,47	**112.803,06 ± 1.356,4**	127.962,37
15	**332.269,12 ± 5.237,58**	378.368,96 ± 3.673,12	333.943,02
20	**687.888,94 ± 6.742,16**	863.984,8 ± 6.151,97	737.213,16

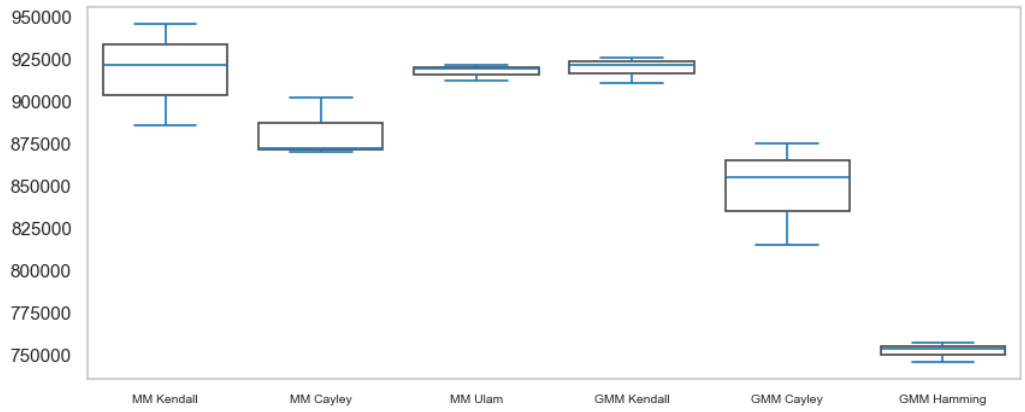
Cuadro 4.11: Comparativa resultados del conjunto de datos de coches

#Objetos	**Coste EDAPerm**	**Coste EGNAPerm**	**Coste *clustering***
10	**322,79 ± 0**	326,51 ± 1,33	322,79
15	**1.149,88 ± 12,43**	1.227,29 ± 8,96	1.151,09
20	**1.961,5 ±61,02**	2.548,68 ± 94,71	2.199,21

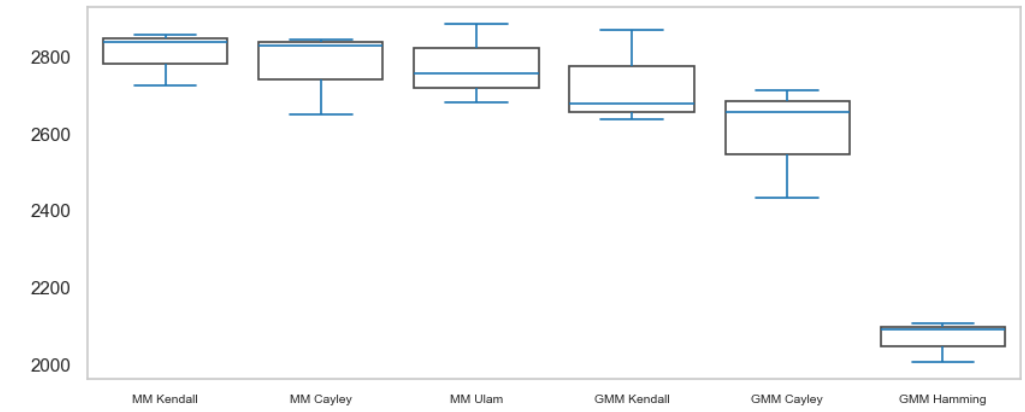
Cuadro 4.12: Comparativa resultados del conjunto de datos de cáncer cervical

#Objetos	**Coste EDAPerm**	**Coste EGNAPerm**	**Coste *clustering***
10	27.496,53 ± 113,11	**27.416,55 ± 0**	39.995,34
15	**73.645,22 ± 5.548,96**	85.180,55 ± 1.819,43	74.017,52
20	**168.279,15 ± 6.763,27**	191.484,69 ± 3.167,11	184.731,58

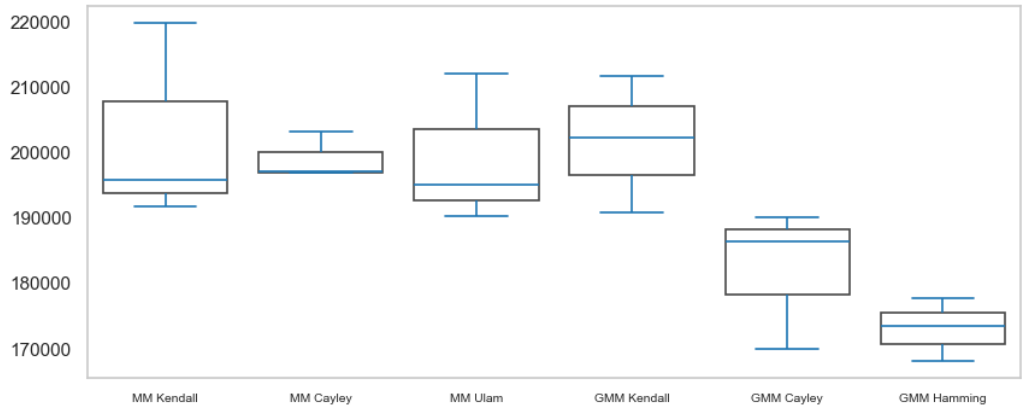
Cuadro 4.13: Comparativa resultados del conjunto de datos de movilidad covid



(a) EDAPerm para conjunto de datos de coches



(b) EDAPerm para conjunto de datos de cáncer cervical

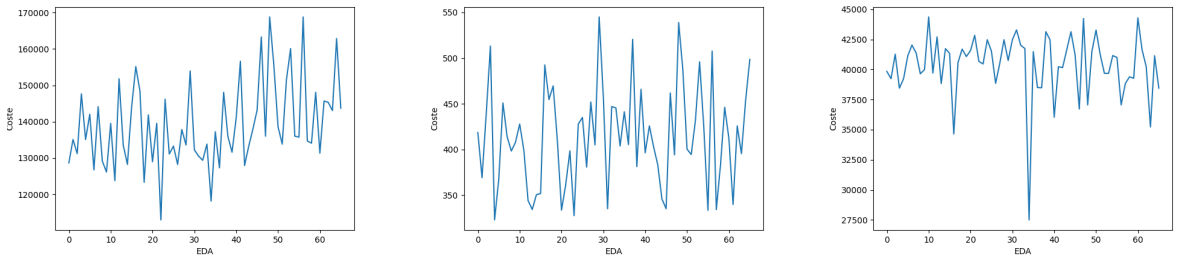


(c) EDAPerm para conjunto de datos de movilidad covid

Figura 4.3: Diagramas de caja con conjuntos de datos de 20 objetos

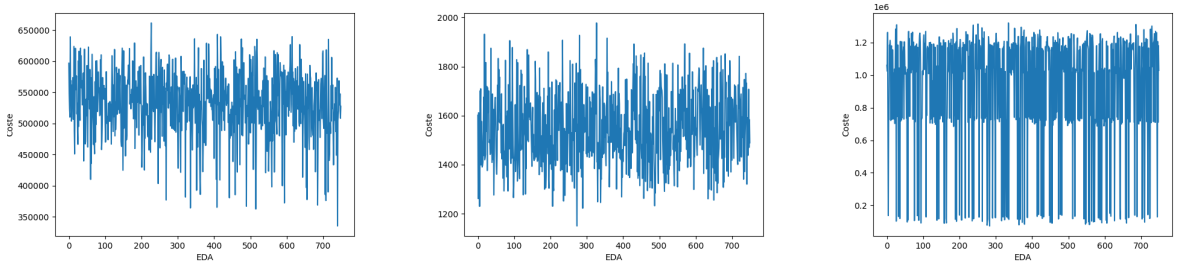
Como se ha podido observar en todas las pruebas realizadas se ha conseguido mejorar el coste del algoritmo de *clustering* jerárquico tradicional. Esto se ha conseguido en su mayoría con el enfoque de EDAPerm, excepto en dos pruebas de 10 objetos que se han obtenido mejores resultados con EGNAPerm.

A continuación se observan en las Figuras 4.4, 4.5 y 4.6 el coste obtenido en cada iteración del algoritmo. Se ha elegido la mejor ejecución para realizar las gráficas.



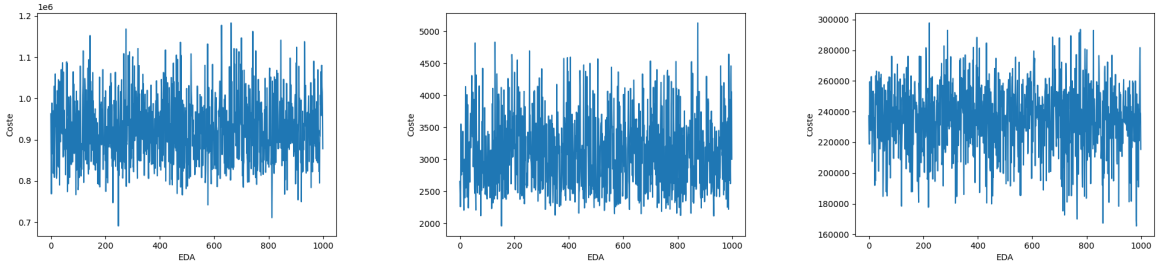
(a) Coches (b) Cáncer cervical (c) Movilidad covid

Figura 4.4: Ejecución prueba con 10 objetos



(a) Coches (b) Cáncer cervical (c) Movilidad covid

Figura 4.5: Ejecución prueba con 15 objetos



(a) Coches (b) Cáncer cervical (c) Movilidad covid

Figura 4.6: Ejecución prueba con 20 objetos

Cabe recordar que cada iteración del enfoque presentado corresponde con la ejecución de un EDA con una estructura de dendograma determinada. Por ello lo que se visualiza en las gráficas es el resultado que se obtiene en cada EDA ejecutado hasta llegar a $50n$ iteraciones, siendo n el número de objetos del problema.

Observando detalladamente las gráficas no se puede extraer ningún patrón significativo, por la sencilla razón de que los resultados obtenidos en cada iteración dependen exclusivamente de cómo de buen desempeño tenga el EDA. Y esto, a su vez, en gran parte está influenciado por la estructura de dendograma que se ha utilizado en dicha iteración.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

El uso de algoritmos de *clustering* ha sido siempre muy recurrente en diferentes tareas como reconocimiento de patrones, análisis de datos, etiquetado de conjuntos de datos, etc. En este trabajo se ha desarrollado un enfoque que utiliza un EDA para aplicarlo a *clustering* jerárquico. Este EDA utiliza una distribución de probabilidad basada en permutaciones, llamada modelo de Mallows. El calculo de la aptitud de los individuos se realiza mediante una función de errores al cuadrado basada en una matriz de distancias ultramétricas.

A partir de los resultados obtenidos, se puede afirmar que se han conseguido cumplir los objetivos propuestos al principio de este trabajo, ya que el enfoque EDAPerm ha sido capaz de superar al algoritmo tradicional de *clustering* jerárquico. Además, se han obtenido otros resultados que indican que el modelo probabilístico que mejores resultados da es el modelo de Mallows generalizado usando la distancia de Hamming.

En las pruebas que se han realizado se han usado tres conjuntos de datos diferentes con distintas características. Una vez vistos estos resultados se puede concluir que realizando pruebas con un número mayor de objetos se puede superar al algoritmo de *clustering* jerárquico de igual manera, siempre que se explore de forma adecuada el espacio de soluciones.

5.2. Líneas futuras

Durante el desarrollo de este trabajo han ido surgiendo diferentes ideas para la mejora del enfoque presentado. El objetivo de estas ideas es poder mejorar, aún más, varios aspectos vistos a lo largo de este documento.

La primera idea es buscar y probar diferentes representaciones de individuos que resulten en una codificación con la cual se pueda explorar la totalidad del espacio de soluciones simplemente con ejecutar un EDA. La dificultad de esto pasa por encontrar una codificación que sea capaz de representar un dendrograma único. Un buen ejemplo de representación de dendrogramas sería una permutación, sin exceder el tamaño del individuo en exceso.

La segunda cuestión que surge al realizar el trabajo fue la posibilidad de paralelizar

el enfoque de EDAPerm. Este enfoque ejecuta muchos EDAs por cada prueba, por lo que si fuera posible ejecutarlos, por ejemplo, en lotes de cuatro algoritmos a la vez, se conseguiría reducir mucho el tiempo de computo. Otra línea futura de investigación relacionada con la reducción del tiempo de cómputo es el desarrollo de una aproximación de la función objetivo.

Y por último, la tercera idea que surge es la realización de pruebas con conjuntos de datos que contengan un número significativamente mayor de objetos que los utilizados en el presente trabajo. Se quiere hacer esto para comprobar la escalabilidad del algoritmo propuesto. Para ello se debe evaluar el enfoque en contextos más exigentes.

Bibliografía

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

Douglas E Critchlow, Michael A Fligner, y Joseph S Verducci. Probability models on rankings. *Journal of Mathematical Psychology*, 35(3):294–318, 1991.

Marco Dorigo, Vittorio Maniezzo, y Alberto Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cbernetics, Part b (cybernetics)*, 26(1):29–41, 1996.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

Tian Zhang, Raghu Ramakrishnan, y Miron Livny. BIRCH: An efficient data clustering method for very large databases. *ACM Sigmod Record*, 25(2):103–114, 1996.

Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, y Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

Sudipto Guha, Rajeev Rastogi, y Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. *ACM Sigmod Record*, 27(2):73–84, 1998.

Erick Cantú-Paz, Martin Pelikan, Martin Pelikan, David E Goldberg, y David E Goldberg. Boa: The bayesian optimization algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, volume 1, 1999.

Diego Doval, Spiros Mancoridis, y Brian S Mitchell. Automatic clustering of software systems using a genetic algorithm. In *Proceedings of the Ninth International Workshop Software Technology and Engineering Practice*, pages 73–81. IEEE, 1999.

Georges R Harik, Fernando G Lobo, y David E Goldberg. The compact genetic algorithm. *IEEE transactions on evolutionary computation*, 3(4):287–297, 1999.

George Karypis, Eui-Hong Han, y Vipin Kumar. CHAMALEON: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

Sudipto Guha, Rajeev Rastogi, y Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, y Susan J Brown. FGKA: A fast genetic k-means clustering algorithm. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 622–623, 2004.

Mahamed G Omran, Andries P Engelbrecht, y Ayed Salman. Image classification using particle swarm optimization. In *Recent Advances in Simulated Evolution and Learning*, pages 347–365. World Scientific, 2004.

Txomin Romero, Pedro Larrañaga, y Basilio Sierra. Learning Bayesian networks in the space of orderings with estimation of distribution algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(04):607–625, 2004.

Julia Handl, Joshua Knowles, y Marco Dorigo. Ant-based clustering and topographic mapping. *Artificial Life*, 12(1):35–62, 2006.

Mahamed GH Omran, Ayed Salman, y Andries P Engelbrecht. Dynamic clustering using particle swarm optimization with application in image segmentation. *Pattern Analysis and Applications*, 8:332–344, 2006.

Ozge Uncu, William A Gruver, Dilip B Kotak, Dorian Sabaz, Zafeer Alibhai, y Colin Ng. Gridbscan: Grid density-based spatial clustering of applications with noise. In *2006 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 2976–2981. IEEE, 2006.

Peng Liu, Dong Zhou, y Naijun Wu. Vdbscan: varied density based spatial clustering of applications with noise. In *2007 International conference on service systems and service management*, pages 1–4. IEEE, 2007.

Kaijun Wang, Junying Zhang, Dan Li, Xinna Zhang, y Tao Guo. Adaptive affinity propagation clustering. *arXiv preprint arXiv:0805.1096*, 2008.

Elena Deza, Michel Marie Deza, Michel Marie Deza, y Elena Deza. *Encyclopedia of distances*. Springer, 2009.

Eduardo Raul Hruschka, Ricardo JGB Campello, Alex A Freitas, et al. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(2):133–155, 2009.

Aruanda SG Meiguins, Alex A Freitas, Robert C Limão, Samuel FS Junior, y Bianchi S Meiguins. An estimation of distribution algorithm for the automatic generation of clustering algorithms. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 1069–1070, 2010.

Josu Ceberio, Alexander Mendiburu, y Jose A Lozano. Introducing the Mallows model on estimation of distribution algorithms. In *In Proceedings of the 18th International Conference on Neural Information Processing, ICONIP 2011, Part II 18*, pages 461–470. Springer, 2011.

Shraddha Pandit, Suchita Gupta, et al. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science*, 2 (1):29–31, 2011.

Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, y Jose A Lozano. A review on estimation of distribution algorithms in permutation-based combinatorial optimization problems. *Progress in Artificial Intelligence*, 1:103–117, 2012.

RJ Kuo, YJ Syu, Zhen-Yao Chen, y Fang-Chih Tien. Integration of particle swarm optimization and genetic algorithm for dynamic clustering. *Information Sciences*, 195:124–140, 2012.

Josu Ceberio, Ekhine Irurozki, Alexander Mendiburu, y Jose A Lozano. A distance-based ranking model estimation of distribution algorithm for the flowshop scheduling problem. *IEEE Transactions on Evolutionary Computation*, 18(2):286–300, 2013.

Sinan Saraçli, Nurhan Doğan, y İsmet Doğan. Comparison of hierarchical cluster analysis methods by Cophenetic correlation. *Journal of Inequalities and Applications*, 2013:1–8, 2013.

Mohammad M Shurman, Mamoun F Al-Mistarihi, Amr N Mohammad, Khalid A Darabkh, y Ahmad A Ababnah. Hierarchical clustering using genetic algorithm in wireless sensor networks. In *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 479–483. IEEE, 2013.

Shafiq Alam, Gillian Dobbie, y Saeed Ur Rehman. Analysis of particle swarm optimization based hierarchical data clustering approaches. *Swarm and Evolutionary Computation*, 25:36–51, 2015.

Henry EL Cagnini, Rodrigo C Barros, Christian V Quevedo, y Márcio P Basgalupp. Medoid-based data clustering with estimation of distribution algorithms. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 112–115, 2016.

Jasmine Irani, Nitin Pise, y Madhura Phatak. Clustering techniques and the similarity measures used in clustering: A survey. *International Journal of Computer Applications*, 134(7):9–14, 2016.

Ekhine Irurozki, Borja Calvo, y Jose A Lozano. PerMallows: An R package for mallows and generalized mallows models. *Journal of Statistical Software*, 71:1–30, 2016.

Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, y Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, y Andronicus A Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

Vicente P Soloviev, Pedro Larrañaga, y Concha Bielza. Estimation of distribution algorithms using gaussian bayesian networks to solve industrial optimization problems constrained by environment variables. *Journal of Combinatorial Optimization*, 44(2): 1077–1098, 2022.

Markelle Kelly, Rachel Longjohn, y Kolby Nottingham. The UCI machine learning repository. *URL https://archive.ics.uci.edu*, 2023.

Vicente P Soloviev, Concha Bielza, y Pedro Larrañaga. Semiparametric estimation of distribution algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation*, 2023.

Vicente P Soloviev, Pedro Larrañaga, y Concha Bielza. EDAspy: An extensible python package for estimation of distribution algorithms. *Neurocomputing*, page 128043, 2024.

Michael A Fligner y Joseph S Verducci. Multistage ranking models. *Journal of the American Statistical Association*, 83(403):892–901, 1988.

Heinz Mühlenbein y Gerhard Paass. From recombination of genes to the estimation of distributions i. binary parameters. In *International Conference on Parallel Problem Solving from Nature*, pages 178–187. Springer, 1996.

Michèle Sebag y Antoine Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *International Conference on Parallel Problem Solving from Nature*, pages 418–427. Springer, 1998.

Chris Fraley y Adrian E Raftery. How many clusters? Which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

José Antonio Lozano y Pedro Larrañaga. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20(9):911–918, 1999.

K Krishna y M Narasimha Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(3):433–439, 1999.

Juha Vesanto y Esa Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.

Roded Sharan y Ron Shamir. CLICK: A clustering algorithm with applications to gene expression analysis. In *Proc Int Conf Intell Syst Mol Biol*, volume 8, page 16. Maryland, MD, 2000.

Pedro Larrañaga y Jose A Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, volume 2. Springer Science & Business Media, 2001.

Raymond T. Ng y Jiawei Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

Parag M Kanade y Lawrence O Hall. Fuzzy ants as a clustering concept. In *22nd International Conference of the North American Fuzzy Information Processing Society, (NAFIPS 2003)*, pages 227–232. IEEE, 2003.

GJ McLachlan y SU Chang. Mixture modelling for cluster analysis. *Statistical methods in medical research*, 13(5):347–361, 2004.

Gautam Garai y BB Chaudhuri. A novel genetic algorithm for automatic clustering. *Pattern Recognition Letters*, 25(2):173–187, 2004.

Derya Birant y Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1):208–221, 2007.

Chih-Chin Lai y Chuan-Yu Chang. A hierarchical evolutionary algorithm for automatic medical image segmentation. *Expert Systems with Applications*, 36(1):248–259, 2009.

Hae-Sang Park y Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009.

Na Luo y Feng Qian. Evolutionary algorithm using kernel density estimation model in continuous domain. In *2009 7th Asian Control Conference*, pages 1526–1531. IEEE, 2009.

Leonard Kaufman y Peter J Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 2009.

Lior Rokach y Oded Maimon. *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.

Mark Hauschild y Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

Fionn Murtagh y Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

Hong He y Yonghong Tan. A two-stage genetic algorithm for automatic clustering. *Neurocomputing*, 81:49–59, 2012.

Anant J Umbarkar y Pranali D Sheth. Crossover operators in genetic algorithms: A review. *ICTACT Journal on Ssoft Computing*, 6(1), 2015.

Henry EL Cagnini y Rodrigo C Barros. PASCAL: An EDA for parameterless shape-independent clustering. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3433–3440. IEEE, 2016.

Adán José-García y Wilfrido Gómez-Flores. Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing*, 41:192–213, 2016.

Bart. Corgis dataset project, 2015. URL `https://corgis-edu.github.io/corgis/csv/cars/`.

Chaudhury. Covid-19 community mobility dataset, 2020. URL `https://www.kaggle.com/datasets/arghadeep/covid19-community-mobility-dataset`.

Tunchan Cura. A particle swarm optimization approach to clustering. *Expert Systems with Applications*, 39(1):1582–1588, 2012.

Persi Diaconis. Group representations in probability and statistics. *Lecture Notes-monograph series*, 11:i–192, 1988.

Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

Jiancong Fan. OPE-HCA: An optimal probabilistic estimation approach for hierarchical clustering algorithm. *Neural Computing and Applications*, 31:2095–2105, 2019.

Georges Raif Harik. *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. University of Michigan, 1997.

Jon H Holand. Adaptation in natural and artificial systems. university of michigan press. *Ann Arbor*, 1(975):1, 1975.

Ekhine Irurozki. *Sampling and Learning Distance-Based Probability Models for Permutation Spaces*. PhD thesis, Universidad del País Vasco-Euskal Herriko Unibertsitatea, 2014.

Angur Mahmud Jarman. Hierarchical cluster analysis: Comparison of single linkage, complete linkage, average linkage and centroid linkage method. *Georgia Southern University*, 29, 2020.

Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Pedro Larranaga. Optimization in continuous domains by learning and simulation of gaussian networks. In *Proc. of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, 2000.

Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17: 395–416, 2007.

Lotfi A Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

Charles T Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 100(1):68–86, 1971.