Nuevos métodos para clustering basado en algoritmos evolutivos

New methods for clustering based on evolutionary algorithms

Hermes Robles Berumen

Directores: Dr. Sebastián Ventura Soto

Dra. Amelia Zafra Gómez

Computación Avanzada, Energía y Plasma

Universidad de Córdoba

Esta tesis se presenta para optar al grado de

Doctor

Septiembre de 2023

TITULO: *NUEVOS MÉTODOS PARA CLUSTERING BASADO EN ALGORITMOS EVOLUTIVOS*

AUTOR: *Hermes Robles Berumen*

Campus de Rabanales
Ctra. Nacional IV, Km. 396 A
14071 Córdoba

https://www.uco.es/ucopress/index.php/es/
ucopress@uco.es

INFORME RAZONADO DE LAS/LOS DIRECTORAS/ES DE LA TESIS

Este documento se presentará junto con el depósito de la tesis en https://moodle.uco.es/ctp3/

DOCTORANDA/O

Hermes Robles Berumen

TÍTULO DE LA TESIS:

Nuevos métodos para clustering basado en algoritmos evolutivos

INFORME RAZONADO DE LAS/LOS DIRECTORAS/ES DE LA TESIS
(se hará mención a la evolución y desarrollo de la tesis, así como a trabajos y publicaciones derivados de la misma)

La presente tesis doctoral ha tenido una evolución muy favorable en los últimos años, se ha publicado un artículo en una revista del primer cuartil, Knowledge-based systems y dos artículos de congresos, uno nacional y otro internacional. Además, con el trabajo realizado se ha enviado recientemente un artículo a una revista del primer cuartil, Swarm and Evolutionary Computation y se está preparando una versión extendida de la propuesta enviada al congreso internacional debido a las buenas valoraciones recibidas. Es importante destacar que, por motivos personales y laborales del doctorando que se encuentra trabajando y reside en otro país, los primeros años de la tesis no produjeron avances significativos. No obstante, en los últimos años se ha producido un importante avance que ha permitido cubrir todos los objetivos planteados en la misma.

Se considera que el trabajo realizado supone una contribución relevante en el área del problema del agrupamiento resuelto por algoritmos genéticos. Se ha llevado a cabo un exhaustivo estudio que ha permitido realizar una taxonomía específica para estos algoritmos, se ha desarrollado una herramienta totalmente funcional y modular, de código abierto, que contiene todos los algoritmos del estado del arte. Se espera que esta herramienta facilite la tarea de utilizar este tipo de algoritmos tanto para realizar estudios comparativos con ellos, como para el desarrollo de nuevas propuestas. Se ha llevado a cabo un estudio experimental que incluye todos los algoritmos estudiados, con una gran cantidad de conjuntos de datos y medidas de evaluación que nos ha permitido conocer las principales ventajas e inconvenientes de las diferentes propuestas. Finalmente, se han desarrollado dos modelos basados en algoritmos genéticos que muestran un excelente rendimiento frente a las propuestas previas.

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

Córdoba, a 11 de septiembre de 2023

Las/los directoras/es

VENTURA SOTO SEBASTIAN EMILIO - 30510000V
Firmado digitalmente por VENTURA SOTO SEBASTIAN EMILIO - 30510000V
Fecha: 2023.09.11 11:46:15 +02'00'

Fdo.:Sebastián Ventura

ZAFRA GOMEZ AMELIA - 30961378C
Firmado digitalmente por ZAFRA GOMEZ AMELIA - 30961378C
Fecha: 2023.09.11 10:25:22 +02'00'

Amelia Zafra

Dedicatoria

A la memoria de mi padre Luciano y mis hermanas María Guadalupe y Elena Emma

A mi madre Socorro por su apoyo que siempre nos ha brindado para lograr nuestras metas

A mis hermanos Ciro, Ana Isabel, Carlos, Ruth y Susana, a mis sobrinos

A mis hijas Patricia, Hanna y Erika, con dedicación todo se puede lograr

Agradecimientos

Las deudas de gratitud que he incurrido en la preparación de este trabajo son numerosas, son muchas las personas que han ayudado en mi camino.

Agradezco en especial a mis directores de tesis Dra. Amelia Zafra y Dr. Sebastián Ventura Soto por todo el apoyo que me han brindado, por su disposición y por compartir conmigo su conocimiento y experiencia.

A la Universidad Autónoma de Zacatecas por permitirme las condiciones para realizar mis estudios de doctorado.

Por último, agradezco a mi familia su paciencia y apoyo para realizar este trabajo.

Abstract

This doctoral thesis deals with the study of clustering. To solve this problem is necessary to divide a data set into groups, where the elements within each group are similar to each other and different from the elements of other groups. The type of clustering in which this work focuses is partitional clustering where the K-means algorithm appears as one of the precursor algorithms in solving this problem.

Specifically, clustering or group analysis is an unsupervised learning technique used in the field of data mining and artificial intelligence. It has great relevance due to its application in a wide variety of fields of science such as image segmentation, digital voice processing, document retrieval, and Internet applications. Currently, it continues to increase in diverse application domains such as astronomy, geology, geophysics, paleoecology, and medicine.

From the point of view of computational complexity theory, the difficulty of clustering problems is classified as the NP-hard type. It is a non-convex problem that usually has many local optimums, so that, algorithms often find one of these local optimums. The purpose is to obtain an optimal solution that guarantees a quality grouping based on some criteria fixed. For its resolution, many different algorithm approaches have been designed, among which we find the bio-inspired algorithms where are the genetic algorithms. These algorithms are computational models that simulate the process of natural evolution to solve problems in different domains, including clustering.

In this thesis, first, an exhaustive review of the genetic algorithms that solve the clustering problem has been carried out. With the focus on the genetic algorithms that use a single objective, it has been carried out a detailed study with a taxonomy of the different proposals that represent the state of the art. As a complement, a software tool called LEAC has been developed that includes the implementation of all the previous proposals studied and that has been made available to the scientific community. The purpose is to enable access to all these models so that they can be used directly without extensive knowledge within the same framework. In addition, its modular design greatly facilitates the design of new proposals thanks to all the crossover and mutation operators, selection operators, initialization methods, encoding types, and performance measures that are available for use directly in new proposals.

To analyze the different existing proposals, it has been carried out an exhaustive experimental study including all the previous algorithms, a large number of data sets, and performance measures that analyze both compactness and separation, two criteria widely used in these environments. In addition, all the information related to data, algorithms, and execution commands is available to facilitate the reproduction of the results.

Finally, two new genetic algorithm proposals have been developed that solve the clustering problem. They use optimized encodings and specialized operators and are highly competitive with respect to previous proposals. Concretely, they obtain optimized solutions that improve the performance of previous proposals.

Resumen

La presente tesis doctoral aborda el estudio de la solución del problema de agrupamiento. La solución a este problema busca dividir un conjunto de datos en grupos, donde los elementos dentro de cada grupo sean similares entre sí y diferentes con los elementos de otros grupos. El tipo de agrupamiento en el que se enfoca este trabajo es la agrupación basada en particiones, siendo el algoritmo K-means uno de los algoritmos precursores en resolver este problema.

Concretamente, el agrupamiento o análisis de grupos es una técnica de aprendizaje no supervisado utilizada en el campo de la minería de datos y la inteligencia artificial. Tiene una gran relevancia debido a su aplicación en una amplia variedad de campos de la ciencia como la segmentación de imágenes, procesamiento digital de voz, recuperación de documentos, aplicaciones en internet, y actualmente, continúa el incremento en diferentes dominios de aplicación cada vez más diversos tales como la astronomía, geología, geofísica, paleoecología y medicina.

Desde el punto de vista de la teoría de la complejidad computacional, la dificultad del problema de agrupación está al ser considerado del tipo NP-difícil, es un problema no convexo que suele contar con muchos óptimos locales por lo que al ser solucionado por un algoritmo a menudo termina dentro de uno de ellos. La finalidad es obtener una solución óptima que garantice una agrupación de calidad en función de determinados criterios que se fijen. Para su resolución se han diseñado muchos y diferentes enfoques de algoritmos, entre los que se encuentran los algoritmos bio-inspirados y dentro de estos, los algoritmos genéticos, que son modelos computacionales que simulan el fenómeno de evolución natural para resolver problemas en diferentes dominios incluyendo la agrupación.

En esta tesis, en primer lugar, se ha realizado una exhaustiva revisión de los algoritmos genéticos que resuelven el problema de agrupamiento, centrándonos en los algoritmos genéticos que emplean un único objetivo, lo que ha permitido llevar a cabo un estudio pormenorizado con una taxonomía de las diferentes propuestas que representan el estado del arte. Como complemento, se ha desarrollado una herramienta de software llamada LEAC que cuenta con la implementación de todas las propuestas previas estudiadas y que se ha puesto disponible para la comunidad científica. La finalidad es posibilitar el acceso a todos estos modelos y que puedan ser utilizados directamente sin un amplio conocimiento de los mismos dentro de un mismo

marco de trabajo. Además, su diseño modular, facilita en gran medida el diseño de nuevas propuestas gracias a todos los operadores de cruce y mutación, los operadores de selección, los métodos de inicialización, los tipos de codificaciones y las medidas de rendimiento que tiene disponible para su utilización directa en nuevas propuestas.

Con el objetivo de analizar las diferentes propuestas existentes, se ha llevado a cabo un exhaustivo estudio experimental que ha incluido a todas las propuestas previas, una gran cantidad de conjuntos de datos y de índices de desempeño que miden tanto lo compacto que son los grupos que se han formado como la separación que existe entre ellos, esas son dos de las medidas más ampliamente utilizadas en estos entornos. Además, toda la información relativa a datos, algoritmos y comandos utilizados para la ejecución de las propuestas está disponible para facilitar una reproducción de los resultados.

Finalmente, se han desarrollado dos nuevas propuestas de algoritmos genéticos que abordan este problema con codificaciones optimizadas y operadores especializados que han resultado ser altamente competitivas con respecto a las propuestas previas, obteniendo soluciones optimizadas que mejoran el rendimiento de estas propuestas previas.

Índice

Lista de figuras

Lista de tablas

Lista de Algoritmos

Capítulo 1

Introducción

El análisis de grupos o agrupamiento es una técnica de aprendizaje no supervisado utilizada en el campo de la minería de datos y la inteligencia artificial. Su objetivo es dividir un conjunto de datos en grupos[1], donde los elementos dentro de cada grupo sean similares entre sí en comparación con los elementos de otros grupos. La finalidad principal de esta tarea es encontrar estructuras inherentes en los datos sin la necesidad de etiquetas o categorías previas.

En este capítulo se contextualiza el problema que se aborda en esta tesis, se describen los principales objetivos y se comenta la estructura de la tesis.

1.1 Contexto

El problema de agrupación[2] intenta encontrar grupos de conjuntos de objetos[3] en un espacio multidimensional que sean similares entre ellos. Para llevar a cabo esta tarea, se requiere de algún criterio de semejanza o diferencia que nos permita distribuir los patrones en los diferentes grupos, de manera que todos los patrones que estén dentro de un grupo mantengan más similitud entre sí y sean más diferentes con los patrones pertenecientes a otros grupos [63]. Definir el criterio de similitud entre los objetos es, por tanto, una tarea crucial para un análisis de grupos exitoso [63, 132].

Definido el criterio de semejanza y la manera de construir los grupos, la agrupación puede plantearse como un problema de optimización [86, 90] que trata de encontrar el máximo de una función objetivo. Esta función objetivo debe mapear las diferentes particiones de un conjunto de objetos en propiedades deseables de agrupación, tales como *compactibilidad* (qué

[1] En la literatura en inglés sobre este tema a un grupo se le conoce como cluster

[2] En inglés se le conoce como clustering

[3] El término de *objeto*, *patrón*, *instancia*, *ejemplo* o *prototipo* tienen el mismo significado en la literatura de análisis de grupos. En este trabajo se usarán de manera indistinta

tan similares son los objetos dentro del mismo grupo), *separación* (qué tan diferentes son los grupos entre sí) [108] y *conectividad* (qué patrones vecinos comparten el mismo grupo) [54]. Estas propiedades se conocen como la bondad[4] de la solución de agrupamiento y generalmente se mide utilizando una medida de evaluación que se conoce habitualmente como *índice de validez de agrupamiento* (CVI[5]) [90].

La agrupación es una tarea de la minería de datos catalogada como una técnica de análisis exploratoria no supervisada, donde los objetos pueden no estar previamente etiquetados o clasificados por un experto [114, 129]. Este hecho ha convertido al agrupamiento en una actividad relevante debido a que actualmente se producen cantidades masivas de datos, siendo complejo etiquetar los ejemplos por un experto. En este contexto, la agrupación es reconocida ampliamente como una de las tareas fundamentales dentro del aprendizaje automático. Además, tiene aplicación en una amplia variedad de campos de la ciencia como la segmentación de imágenes [125], procesamiento digital de voz, recuperación de documentos [68], aplicaciones en internet [19], y continua el incremento en diferentes dominios de aplicación cada vez más diversos tales como la astronomía [9], geología [45], geofísica [25], paleoecología [47] y medicina [102].

Tradicionalmente, las técnicas de agrupación pueden ser clasificadas por los enfoques que utilizan los algoritmos: *particiones*, *jerárquicas*, *densidad* y *cuadrícula* [63]. Otras taxonomías [34] incluyen a la categoría *basada en modelos*. Esta tesis se centra de manera particular en el estudio de la agrupación basada en particiones. El agrupamiento basado en particiones es una categoría fundamental de los algoritmos de agrupamiento y una de las más ampliamente utilizadas. En esta técnica, busca dividir un conjunto de datos en un número predeterminado de grupos, donde cada dato pertenece exactamente a un grupo. Es decir, cada elemento del conjunto de datos es asignado a uno y solamente un grupo en el proceso de agrupamiento. Esta técnica se puede aplicar a conjuntos de datos con diferentes características. Además, los métodos basados en particiones son computacionalmente más eficientes en comparación de otras técnicas, por lo que son los más utilizados [136].

Otro aspecto importante que motiva el estudio de la agrupación basada en particiones es por la naturaleza del problema. Se trata de un problema no convexo que suele contar con muchos óptimos locales por lo que la solución de los algoritmos a menudo termina dentro de uno de ellos. Así es considerado como un problema de tipo NP-complejo o NP-difícil[6] [36, 77].

En este contexto, una variedad de algoritmos meta-heurísticos se puede encontrar para resolver el problema de optimización. Esta tesis se centra en los Algoritmos Evolutivos (EAs, del inglés Evolutionary Algorithm). Concretamente, en los Algoritmos Genéticos (AGs) que

[4] En inglés, goodness.
[5] Del inglés Cluster Validity Index
[6] En inglés a esta clase de problemas se les conoce como NP-hard

realizan búsquedas estocásticas basadas en el principio de los sistemas genéticos naturales [46, 87]. Durante las últimas décadas y en la actualidad, los AGs se han aplicado a muchos problemas NP-difícil con muy buenos resultados, por lo que se considera relevante seguir realizando avances en esta área [108]. Los AGs realizan una búsqueda multidimensional para proporcionar un valor óptimo para una función objetivo en un problema de optimización. A diferencia de los métodos de búsqueda convencionales, los AGs tratan con múltiples soluciones simultáneamente, se les calcula el valor de aptitud y van evolucionando [92]. Además, los AGs tienen la ventaja de que son clasificados como métodos de búsqueda robustos y adaptativos que realizan una búsqueda global en el espacio de soluciones candidatas [40].

Si bien, el análisis de grupos ha sido ampliamente estudiado y se encuentra una gran cantidad de trabajos donde revisan los diferentes métodos que se han propuesto. Recientemente, está el de Ezugwu et al. [33] donde presentan un estudio descriptivo de algoritmos meta-heurísticos inspirados en la naturaleza, en específico incluyen a los algoritmos de agrupamiento tanto jerárquicos como en particiones. El hecho que incluyan tantas técnicas hace que sea una revisión muy general, que es lo que le ocurre a la mayoría de los trabajos existentes. Por lo tanto, no incluyen una descripción de las particularidades de los AGs ni una taxonomía específica para ellos. Con alguna excepción, como la revisión específica centrada en AGs realizada por Hruschka et al. [59] en 2009 donde hacen una descripción de los EAs para agrupamiento centrado principalmente en los AGs y algunos trabajos sobre el agrupamiento evolutivo multiobjetivo y aprendizaje por conjuntos. Pero no existen revisiones que incluya un análisis experimental para evaluar el rendimiento de las diferentes propuestas. Ni tampoco se encuentran trabajos que lleven a cabo una comparativa de los AGs contra los trabajos previos de manera analítica. Por lo general las implementaciones de los algoritmos no están disponibles y finalmente, todos los estudios se limitan a comparar algunas de las propuestas previas con algunos conjuntos de datos y métricas, así. no se pueden considerar estudios exhaustivos, ni cuantitativos.

Estas características hacen muy difícil tener un estado del arte de las propuestas de AGs que se han aplicado en la actualidad, de forma que se puede tener una taxonomía que permita clasificarlas fácilmente y conocer las propuestas que existen para desarrollar nuevas propuestas. De este modo, debido a los excelentes resultados de los AGs en la resolución del problema de agrupamiento ha dado lugar a la publicación de un gran número de propuestas, éstas no se encuentran categorizadas. Esta tesis para poder llevar a cabo un verdadero estudio exhaustivo se va a centrar en los AGs mono-objetivos, con la finalidad de describir todas las propuestas y evaluar su desempeño utilizando una gran variedad de medidas de evaluación y de conjuntos de datos. Esta información sería esencial para el desarrollo de nuevas propuestas eficientes en esta área y permitir nuevos avances de la comunidad científica. Por ello, la presente tesis

doctoral llevará a cabo un estudio exhaustivo de todas las propuestas, las cuales serán descritas en detalle y se propondrá una taxonomía que permita identificar fácilmente las propuestas existentes, así como clasificar nuevas propuestas, se llevará a cabo una implementación de todas estas propuestas en una misma librería, de forma que puedan ser utilizadas fácilmente en un estudio comparativo, y además se llevará a cabo un análisis exhaustivo de las mismas con diferentes CVIs y conjuntos de datos que permitan identificar las ventajas e inconvenientes de las diferentes propuestas y facilitar el diseño de nuevas propuestas en el área.

Una vez finalizada esta labor, la presente tesis también diseñará e implementará nuevas propuestas que mejoren el estado del arte de los AGs actuales.

1.2 Objetivos

El objetivo principal de esta tesis es el desarrollo de nuevos métodos de agrupación basados en AGs. Para ello, esta tesis se centra en los AGs mono-objetivos y se plantea los siguientes objetivos para lograr su meta final:

- Realizar un estudio exhaustivo de los AGs propuestos hasta la fecha que resuelvan el problema de agrupamiento. Por lo tanto, hasta nuestro conocimiento, se consideran y describen todas las referencias publicadas hasta la fecha que utilizan AGs mono-objetivo para resolver el problema de agrupación basado en particiones. El propósito es describir los principales componentes de cada propuesta, de forma que se permita fácilmente conocer las singularidades de cada algoritmo.

- Proponer una taxonomía específica que considere todas las particularidades de los AGs, para desarrollar la taxonomía se partirá de las propuestas hasta la fecha, la cual debe permitir clasificar fácilmente todos los AGs y además, incluir nuevos algoritmos para agrupamiento.

- Desarrollar una librería con todos los algoritmos propuestos, deberá ser una librería modular para que permita tanto utilizar las propuestas previas, como desarrollar nuevas propuestas fácilmente. Esta librería deberá estar a disposición de la comunidad científica y ser de código abierto para fomentar nuevos avances en esta área.

- Realizar un estudio experimental empírico que evalúe la capacidad de los AGs para resolver problemas de agrupamiento. El estudio considerará bajo el mismo escenario todas los AGs conocidos, con conjuntos de datos representativos (concretamente, 94 conjuntos de datos) y una amplia variedad de métricas (concretamente, 22 CVIs). Este sería un primer intento de evaluar todas las propuestas con un número representativo de

medidas de evaluación y datos. Además, todos los algoritmos, datos y configuraciones estarán disponibles para la comunidad científica de forma que se permita reproducir todos los resultados obtenidos.

- Realizar una nueva propuesta competitiva de un AG que parta de un número de grupos conocidos y que resuelva de forma eficiente el problema de agrupamiento. Cuando se conoce el número de grupos, los AGs se diseñan con unos objetivos diferentes y por tanto, se plantea como objetivo desarrollar una propuesta específica tanto para cuando se conozca este valor, como cuando no es conocido.

- Realizar una nueva propuesta competitiva de un AG que determine el número de grupos óptimo y que resuelva de forma eficiente el problema de agrupamiento. Cuando no se tiene información de los grupos, los AGs deben tener esta característica en cuenta para poder incluir la determinación de este valor dentro de su proceso evolutivo.

1.3 Estructura de la tesis

A continuación se describen brevemente los distintos capítulos en los que está dividida la memoria de esta tesis doctoral:

- El capítulo *Introducción* presenta una visión general de este trabajo. En primer lugar, se contextualiza el problema dando una motivación del trabajo realizado. A continuación, se plantean los principales objetivos a conseguir y finalmente, se comenta la estructura de la tesis.

- El capítulo *El problema de agrupamiento* establece las características de este problema y los principales enfoques que han resuelto el problema basado en particiones. También, incluye una descripción de los diferentes algoritmos bio-inspirados que se han utilizado para resolver este problema.

- El capítulo *Algoritmos genéticos para agrupación* lleva a cabo un estudio exhaustivo de las diferentes propuestas de AGs mono-objetivos que resuelven el problema de agrupamiento. Primero, se incluye una descripción de las características generales de los AGs, clasificando a los diferentes componentes que los conforman. A continuación, se describen los diferentes AGs propuestos diferenciando entre los que necesitan conocer el número de grupos y los que también intentan determinar este valor. Finalmente, incluye una taxonomía para poder clasificar las diferentes propuestas y permitir situar las nuevas propuestas que se desarrollarán.

- El capítulo *Estudio experimental de los algoritmos genéticos para agrupación* proporciona un estudio experimental de los AGs para agrupación permitiendo evaluar la capacidad de los diferentes algoritmos para resolver el problema de agrupación. Se muestra una discusión de los resultados obtenidos donde se determinan las ventajas e inconvenientes de las diferentes propuestas y los componentes que resultan más beneficiosos para resolver el problema. Toda la información de este estudio, algoritmos, conjunto de datos y resultados están disponibles.

- El capítulo *GAMK: Un algoritmo genético para agrupación con número de grupos conocido* propone un nuevo AG para resolver el problema de agrupamiento cuando el número de grupos es conocido. Los principales componentes que forman el algoritmo son descritos y se lleva a cabo un estudio experimental que permite evaluar el rendimiento de esta nueva propuesta y compararlo con las propuestas ya existentes.

- El capítulo *GASGO: Un algoritmo genético para agrupación con número de grupos desconocido* propone un nuevo AG para resolver el problema de agrupamiento cuando el número de grupos es desconocido. Los principales componentes que forman el algoritmo son descritos y se lleva a cabo un estudio experimental que permite evaluar el rendimiento de esta nueva propuesta y compararlo con las propuestas ya existentes. Además, un estudio específico de la estimación del número de grupos en estos algoritmos se lleva a cabo, debido a que se considera un factor muy importante en el diseño de estas propuestas.

- El capítulo *LEAC: Una librería de algoritmos evolutivos para agrupamiento* describe la arquitectura y principal funcionalidad de la librería de AGs desarrollada. Esta librería contiene las 22 propuestas previas, así como las dos nuevas que se han desarrollado en esta tesis. La librería está disponible y es de código abierto para fomentar los avances en esta área. Cuenta con un manual de usuario detallado que facilita su manejo e incluye ejemplos y descripciones de toda la funcionalidad de la librería que ayuda tanto en la ejecución de cualquiera de los algoritmos que incluye, como en la tarea de desarrollar nuevas propuestas partiendo de todos los módulos que tiene disponibles.

- El capítulo *Conclusiones y trabajo futuro* resume los resultados alcanzados en esta memoria de tesis, y se plantean algunas posibilidades de mejoras y trabajos en el futuro sobre nuevas líneas abiertas a partir de este trabajo.

Capítulo 2

El problema de agrupamiento

Análisis de grupos o simplemente agrupación consiste en formar grupos de conjuntos de instancias de tal manera que las instancias dentro del grupo sean similares entre ellas y diferentes de las instancias de otros grupos. La agrupación es una técnica de aprendizaje automático incluida en el aprendizaje no supervisado. Dentro de los diferentes enfoques de agrupación se encuentra el basado en particiones, estudiado inicialmente con el algoritmo de `K-means`. En este capítulo se describen los diferentes tipos de agrupación, centrándose en la agrupación basada en particiones que es el objeto de estudio de este trabajo, además se detallan las revisiones esenciales del estado del arte que se han realizado en esta área.

2.1 Clasificación de los enfoques de agrupación

En la literatura se pueden encontrar diferentes clasificaciones de los enfoques que resuelven el problema de agrupación, la mayoría de ellos están basados en el funcionamiento de los algoritmos [34, 53, 94, 128]. De forma generalizada, la clasificación de estos enfoques se incluye en cinco categorías basados en: *jerarquías*, *densidad*, *marcos*, *probabilidad* y *particiones*. Una descripción más detallada de estos enfoques de agrupación se muestra a continuación:

Basados en jerarquías

La agrupación jerárquica consiste en obtener una representación gráfica o diagrama de datos en forma de un árbol llamado *dendrograma*. Existen dos estrategias para construir un dendrograma: de manera *aglomerativa* o *divisiva*. La aglomerativa forma un grupo con cada instancia del conjunto de datos como miembro único, de manera sucesiva se van uniendo los grupos en términos de similitud formando subgrupos hasta llegar a unir todas las instancias en un grupo, a esta estrategia se le conoce también como estrategia de abajo hacia arriba. Por el contrario, la

estrategia de arriba hacia abajo se le llama divisiva y lleva un procedimiento totalmente a la inversa del descrito. Un algoritmo de esta categoría es BIRCH [133].

Basados en densidad

La agrupación basada en densidad consiste en ubicar las instancias en el espacio euclidiano y construir los grupos de acuerdo con la densidad de las diferentes instancias. Cada grupo tiene una densidad formada por el conjunto de instancias que la forman, esta densidad debe ser considerablemente mayor que fuera del grupo. La idea clave es que, para cada instancia de un grupo, la vecindad de un radio dado debe contener al menos un número mínimo de instancias, es decir, la densidad en la vecindad debe exceder cierto umbral. El algoritmo más común de esta categoría es DBSCAN [32].

Basados en marcos

Los algoritmos basados en marcos o malla son los más eficientes por el dominio de los datos. Su procedimiento principal es dividir el espacio de datos en un número finito de celdas para formar una estructura de cuadrícula, a continuación, se buscan las celdas significativas cuyas densidades superan un umbral predefinido y agrupan las celdas significativas cercanas en grupos. Algunos algoritmos basados en este enfoque son STING [124], WaveCluster [115] y CLIQUE [73].

Basados en probabilidad

Estos métodos optimizan el ajuste entre los datos proporcionados y algún modelo matemático (predefinido). Se basa en la suposición que los datos son generados por una combinación de distribuciones de probabilidad subyacentes. Además, conducen a una forma de determinar automáticamente el número de grupos en función de estadísticas estándar, teniendo en cuenta el ruido (valores atípicos) y, por lo tanto, producen métodos de agrupación robustos. Algunos algoritmos basados en este enfoque son: MCLUST, probablemente el algoritmo más conocido, pero existen otros algoritmos ampliamente utilizados como EM (que utiliza un modelo de densidad de mezcla), agrupación conceptual (como COBWEB) y enfoques de redes neuronales [34].

Basados en particiones

En la agrupación basada en particiones o simplemente agrupación en particiones, los algoritmos suelen ser incrementales, comienzan asignando cada instancia a un grupo obteniéndose una

partición inicial, luego los grupos se van reformulando hasta satisfacer un criterio impuesto por un índice de validez, teniendo en cuenta la homogeneidad, la separación interna y externa, es decir, las instancias en un mismo grupo deben ser similares entre sí y diferentes en otros grupos [129]. Dentro de este enfoque, existen dos maneras de representar el agrupamiento del conjunto de datos: *partición dura* cada instancia pertenece a un solo grupo y *partición difusa* cada instancia es catalogada con un grado de pertenencia en el intervalo $[0,1]$ a un grupo. Ejemplo de algoritmos de esta categoría son `K-means` [83], ISODATA [5], PAM [65] y FCM [13], los cuales son ampliamente mencionados en la literatura. En la siguiente sección se describe con más detalle este enfoque de agrupación, por ser el tema de estudio de este trabajo.

2.2 Definición de la agrupación basada en particiones

El propósito del problema de agrupación es encontrar una partición óptima de un conjunto de datos X, formado por n instancias (patrones, objetos o puntos), en k subconjuntos $C_1, C_2, ..., C_k = \{C_j\}$, donde $k \leq n$ y C_j es un grupo.

La definición del problema de agrupación basado en particiones, en términos de la relación entrada y salida, es la siguiente:

Entrada: Dado un conjunto de datos $X = \{x_1, x_2, ..., x_n\}$ y un número de grupos k, donde cada x_i representa una instancia descrita por un vector o punto $d-$dimensional, $x_i = \langle x_{i1}, x_{i2}, ...x_{id} \rangle$, y x_{il} es una dimensión, la cual representa un atributo de la instancia x_i.

Salida: Una partición óptima en una familia de k conjuntos de X llamados grupos o clústeres, tal que instancias similares están dentro del mismo grupo C_j e instancias diferentes están en diferentes agrupaciones $C_{j'}$. Se cumplen las siguientes condiciones:

$$\begin{aligned} k &\leq n, \\ C_j &\neq \emptyset \quad \forall j = 1,2,...,k; \\ C_j \cap C_{j'} &= \emptyset \quad \forall j, j' = 1,2,...,k \quad \text{y} \quad j \neq j'; \\ \cup_{j=1}^{k} C_j &= X. \end{aligned} \tag{2.1}$$

Para mayor claridad en la notación de este trabajo, los subíndices i, j y l serán utilizados para indicar:

i, la i-ésima instancia x_i de X, donde i: $i \in \{1,2,...,n\}$,

j, el j-ésimo grupo C_j de la partición de X, donde j: $\in \{1,2,...,k\}$, y

l , la l-ésima dimensión o atributo de una instancia, donde $l : l \in \{1,2,...,d\}$.

Instancia (x_i)	**Atributo**	
	x_{i1}	x_{i2}
x_1	0.5	2.6
x_2	0.2	1.6
x_3	3.4	2.6
x_4	7.9	2.4
x_5	7.6	1.4
x_6	3.7	0.2
x_7	7.0	0.9
x_8	4.2	0.7
x_9	1.2	1.9
x_{10}	2.7	3.8
x_{11}	6.8	2.1
x_{12}	4.8	3.3
x_{13}	6.6	1.7
x_{14}	3.6	4.8
x_{15}	4.3	3.5

Tabla 2.1 Un conjunto de datos con fines ilustrativos

Como ejemplo, en la Tabla 2.1 se muestra un conjunto de datos X, compuesto de 15 instancias u objetos ($n = 15$). Cada instancia está representada por dos atributos (2−dimensiones). Este ejemplo será utilizado con fines didácticos en el desarrollo del presente capítulo y del siguiente.

La agrupación en particiones puede ser vista como un problema de optimización [94], definido formalmente por la ecuación (2.2).

$$\underset{C_k}{\overset{\text{Optimize}}{}} \quad [\boldsymbol{f}(X, C_j)] \quad \forall j = 1,2...k \tag{2.2}$$

Donde la función $\boldsymbol{f}$ es el objetivo de la agrupación, la cual denota la similitud y diferencia entre las instancias presentes en el conjunto de datos, llamada *medida de validez de agrupamiento* o también llamado *índice de validez de agrupamiento* (CVI). Así un algoritmo de agrupación va a estar guiado por un CVI.

El problema de agrupación en particiones desde el punto de vista de la combinatoria es no convexo, que tiene muchas soluciones óptimas localmente, por lo que es común que los algoritmos encuentren una de ellas. Se considera como un tipo particular de problema NP-difícil [36, 77]. En el caso que se hiciera una búsqueda exhaustiva, para encontrar todas las

particiones y así satisfacer el óptimo global, el número de agrupaciones válidas se optimice al evaluar $S(n,k)$ [92], por la ecuación (2.3).

$$S(n,k) = \frac{1}{k!}\sum_{j=1}^{k}(-1)^{k-j}\binom{k}{j}j^n, \tag{2.3}$$

2.3 Enfoques de agrupación basados en particiones

En esta sección se describen las características de las tres principales categorías de la agrupación en particiones, basada en: *centroides*, *instancias representativas* y *partición difusa*. Como ejemplo se especifica cómo operan los algoritmos clásicos de cada una de las categorías `K-means`, `PAN` y `FCM`, con el objetivo de dimensionar el problema de agrupación basado en particiones.

2.3.1 Enfoques basados en centroides

Estos enfoques miden la semejanza de los grupos con respecto al valor *medio* de los objetos del grupo, el cual es conocido como el *centroide* o *centro de gravedad* [52].

Los algoritmos que pertenecen a esta categoría buscan encontrar los centroides $\{\mu_j\}$ de los grupos, de tal manera que permitan hacer una partición óptima al conjunto de datos por medio de la ecuación (2.4).

$$x_i \in C_j \leftrightarrow \mu_j \leftarrow \underset{1 \le j \le k}{\arg\min} \|x_i - \mu_j\|, \forall i \in \{1,2,...,n\} \tag{2.4}$$

Donde $\|\cdot\|$ es la distancia euclidiana, pero también puede ser utilizada otra distancia, cuanto más cercana esté la instancia x_i al centroide μ_j, más semejante será a las instancias del grupo C_j.

Otra manera de escribir la ecuación (2.4) es por medio de (2.5), la cual se encuentra en la literatura para definir la pertenencia de x_i al grupo C_j [84].

$$C_j \leftarrow \left\{x_i : \|x_i - \mu_j\| \le \|x_i - \mu_{j'}\|,\ j' = 1,2,...,k \text{ y } j \ne j'\right\} \tag{2.5}$$

Por otro lado, si los miembros de cada grupo son conocidos es posible encontrar los centroides con la ecuación (2.6).

$$\mu_j = \frac{1}{|C_j|}\sum_{x \in C_j} x_i, \forall i \in \{1,2,...,n\} \text{ y } j \in \{1,2,...,k\} \tag{2.6}$$

En la agrupación basada en particiones, tanto basada en centroides como en instancias representativas y partición difusa, el número de grupos puede ser desconocido en el conjunto de datos y tiene que ser encontrado por el algoritmo de manera automática. A estos algoritmos se les conoce como de *k-variable*, a diferencia de los que el usuario tiene que especificar el número de grupos como parámetro de entrada llamados algoritmos de *k-fija*. De esta forma, se hará referencia a cada algoritmo para indicar si necesita conocer la k o no es necesario.

A continuación, se detalla el algoritmo K-means, que es uno de los algoritmos más conocidos dentro de este enfoque. Este algoritmo estaría incluido en los algoritmos de *k-fija* al necesitar que se le pase el valor de k como parámetro.

Algoritmo K-means

`K-means` (1) es un algoritmo clásico mencionado ampliamente en la literatura de agrupación, encontrándose muchas variantes. El algoritmo toma como parámetro de entrada un número natural k y hace una partición de un conjunto de instancias en k grupos, el resultado esperado como en todos los algoritmos de agrupación, es una agrupación donde dentro de cada grupo las instancias sean semejantes entre ellas y diferentes de las instancias que pertenecen a otros grupos. Los pasos iterativos e incrementales del algoritmo `K-means` son los siguientes:

`Paso de inicialización,` primero son seleccionadas k instancias aleatoriamente, las cuales representan los centroides iniciales de cada uno de los grupos (línea 2).

`Paso de asignación,` ahora todas las instancias son asignadas a un grupo con más semejanza tomando como base la distancia entre cada una de instancias y los centroides (línea 4).

`Paso de actualización,` los centroides de cada grupo se vuelven a calcular, tomando la media de todas las instancias asignadas a cada uno de ellos (línea 5).

Los dos últimos pasos se repiten un número fijo de veces, o hasta que no se logre ninguna mejora adicional, así se obtiene una convergencia [39].

Una ventaja de este algoritmo es su complejidad que está cerca de un tiempo lineal $O(gkn)$, donde g es el número de iteraciones, por lo que generalmente es utilizado en otros algoritmos como una función para afinar los resultados de la agrupación.

`K-means` como muchos de los algoritmos de agrupación clásicos tiene la característica de ser un algoritmo de gradiente de descenso rápido [94]. Por una parte, al asignar en cada paso las instancias a su centroide más cercano minimiza la suma de distancias (es un CVI llamado suma de distancias euclidianas (SED)). Minimizar SED es equivalente a encontrar particiones *compactas*. Por otra parte, debido a la naturaleza del problema de agrupación que

es no convexo, y por la estrategia de gradiente de descenso rápido, en ocasiones termina con una partición localmente óptima, teniendo la desventaja de ser sensible a la inicialización de los centroides [22, 93, 100, 126, 135].

Algoritmo 1 Algoritmo de `K-means` [83]

Entrada: X: un conjunto de datos
k: número de grupos
Salida: Una partición de X en $\{C_j\}$ grupos y sus centroides $\{\mu_j\}$, donde $j : j \in \{1,2,...,k\}$

1: t $\leftarrow$ 1
2: Inicializar k centroide $\mu_1^{(t)}, \mu_2^{(t)}, ..., \mu_k^{(t)}$
3: **repeat**
4: Asignar cada instancia x_i a su grupo por:
$C_j^{(t)} \leftarrow \{x_i : \|x_i - \mu_j\| \leq \|x_i - \mu_{j'}\|,\ j' = 1,2,...,k \text{ y } j \neq j'\}$ ecuación (2.5)
5: Actualizar los centroides con las instancias x_i asignadas al grupo:
$\mu_j^{(t+1)} \leftarrow \frac{\Sigma_{x_i \in C_j^{(t)}} x_i}{|C_j^{(t)}|}$ ecuación (2.6)
6: t $\leftarrow$ t + 1
7: **until** cumplir un criterio de convergencia

Por ejemplo, se realizaron dos ejecuciones del algoritmo `K-means` la entrada es el conjunto de datos ilustrativo de la Tabla 2.1 y el parámetro $k = 3$ grupos. Para la primera corrida la inicialización aletoria de los centroides es realizada en la línea 2 con las siguientes instancias:

$$\mu_1^{(1)} \leftarrow x_2$$
$$\mu_2^{(1)} \leftarrow x_{13}$$
$$\mu_3^{(1)} \leftarrow x_3$$

Después de 2 iteraciones el algoritmo converge al no obtenerse ningún cambio en los grupos, los centroides al finalizar el algoritmo serían:

$$\mu_1^{(2)} \leftarrow (0.63, 2.03)$$
$$\mu_2^{(2)} \leftarrow (7.18, 1.7)$$
$$\mu_3^{(2)} \leftarrow (3.81, 2.7)$$

Con estos centroides y al aplicar el paso indicado en la línea (4), los grupos estarán formados por las instancias: $\{x_1, x_2, x_9\}$ al grupo C_1 por estar más cerca a $\mu_1^{(2)}$, las instancias $\{x_4, x_5, x_7, x_{11}, x_{13}\}$ pertenecen al grupo C_2 por estar más cerca a $\mu_2^{(2)}$ y las instancias

$\{x_3, x_6, x_8, x_{10}, x_{12}, x_{14}, x_{15}\}$ son miembros del grupo C_3 por estar más cercas a $\mu_3^{(2)}$, de manera gráfica se muestra la salida de la ejecución en la Figura 2.1a. Con los resultados se puede evaluar la calidad de la agrupación, como se comentó el CVI que minimiza `K-means` es SED (cuanto más pequeño es el valor. mejor es la agrupación):

$$\begin{aligned}\text{SED} &= \sqrt{(0.5-0.63)^2+(2.6-2.03)^2}+\ldots+\sqrt{(7.9-7.18)^2+(2.4-1.7)^2} \\ &+\ldots+\sqrt{(4.3-3.81)^2+(3.5-2.7)^2} \\ &\approx 15.9813561\end{aligned}$$

Una segunda ejecución del algoritmo `K-means` para los mismos datos de entrada (conjunto de datos Tabla 2.1 y parámetro $k = 3$ grupos), pero inicializando los centroides con instancias diferentes (línea 2):

$$\begin{aligned}\mu_1^{(1)} &\leftarrow x_2 \\ \mu_2^{(1)} &\leftarrow x_8 \\ \mu_3^{(1)} &\leftarrow x_{14}\end{aligned}$$

Después de 3 iteraciones converge a los siguientes centroides:

$$\begin{aligned}\mu_1^{(3)} &\leftarrow (0.63, 2.03) \\ \mu_2^{(3)} &\leftarrow (6.26, 1.34) \\ \mu_3^{(3)} &\leftarrow (3.76, 3.6)\end{aligned}$$

De la misma manera, con los centroides obtenidos y al aplicar el paso indicado en la línea (4), los grupos estarán formados por las instancias: $\{x_1, x_2, x_9\}$ en el grupo C_1 por estar más cerca a $\mu_1^{(3)}$, las instancias $\{x_4, x_5, x_6, x_7, x_8, x_{11}, x_{13}\}$ pertenecen al grupo C_2 por estar más cerca a $\mu_2^{(3)}$ y las instancias $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ son miembros del grupo C_3 por estar más cercas a $\mu_3^{(3)}$, de manera gráfica se muestra el resultado en la Figura 2.1b. En este caso, el valor SED es mayor y representa un mínimo local:

$$\begin{aligned}\text{SED} &= \sqrt{(0.5-0.63)^2+(2.6-2.03)^2}+\ldots+\sqrt{(7.9-6.26)^2+(2.4-1.34)^2}\\ &+\ldots+\sqrt{(3.4-3.76)^2+(2.6-3.6)^2}\\ &\approx 17.30623902\end{aligned}$$

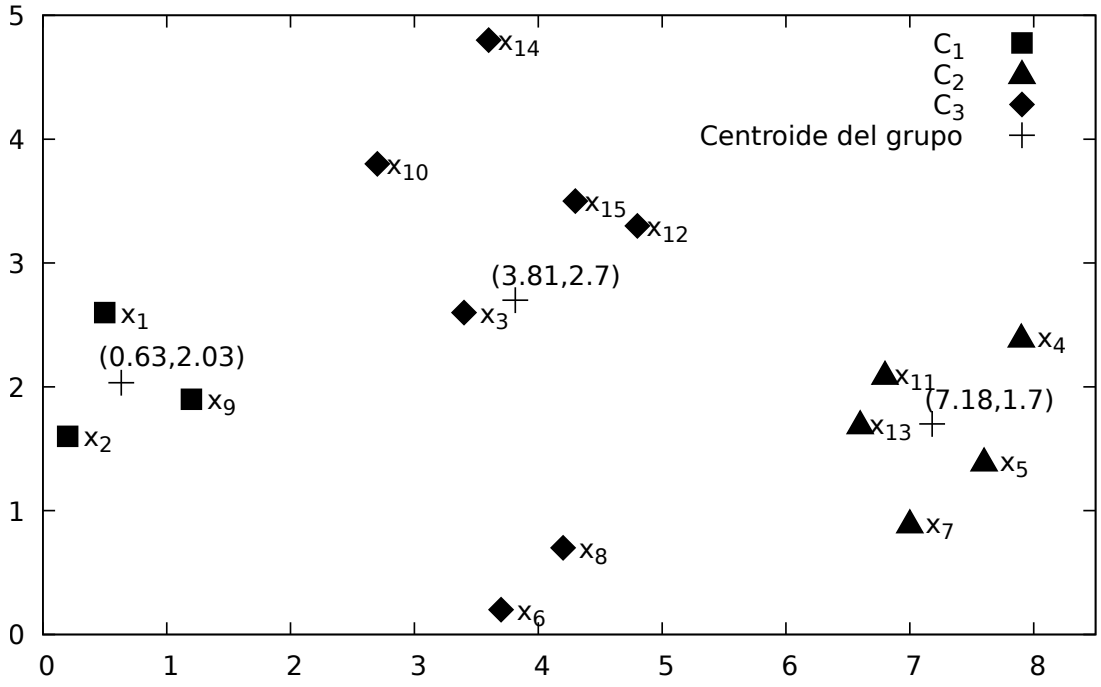
Como se ha comentado, este algoritmo pertenece a los de k-fija siendo necesario que se le proporcione el valor de k para que pueda ejecutarse. Cuando se hace una agrupación de un conjunto de datos y el parámetro k es desconocido, se tendrán dos subproblemas: determinar el número de grupos y encontrar las instancias de cada grupo. Algoritmos como `K-means` solo resuelve el segundo subproblema, al proporcionar el usuario como dato de entrada el número de grupos k. Si k es desconocida se debe recomendar al usuario o este debe ser estimada [108]. Para ser más explícito, se usará k_v en lugar de k cuando se investiga en el conjunto de datos el número de grupos.

En este caso, estos algoritmos que, como se ha indicado, se llamará algoritmos de k-variable, buscarán automáticamente el número de grupos, k_v, con alguna estrategia que optimice tanto la *compactabilidad* como la *separación* de los grupos. Para ello, utilizará un CVI que considere estos dos aspectos, ejemplo de CVI que son utilizados para identificar el número óptimo son Calinski-Harabaz o criterio de relación de varianza (VRC) [14], índice Silhouette [65], índice Davies-Bouldin o índice DB [24] y otros más, también existen otros CVIs estadísticos como *SD* [51], índice S_Dbw index [50] y índice VB [130]. Un estudio comparativo del desempeño de diferentes CVIs, para ser utilizados en la evaluación de algoritmos de agrupación puede ser encontrado en [4], en este trabajo en el capítulo 6 se propondrán CVIs para obtener un valor óptimo de k_v.

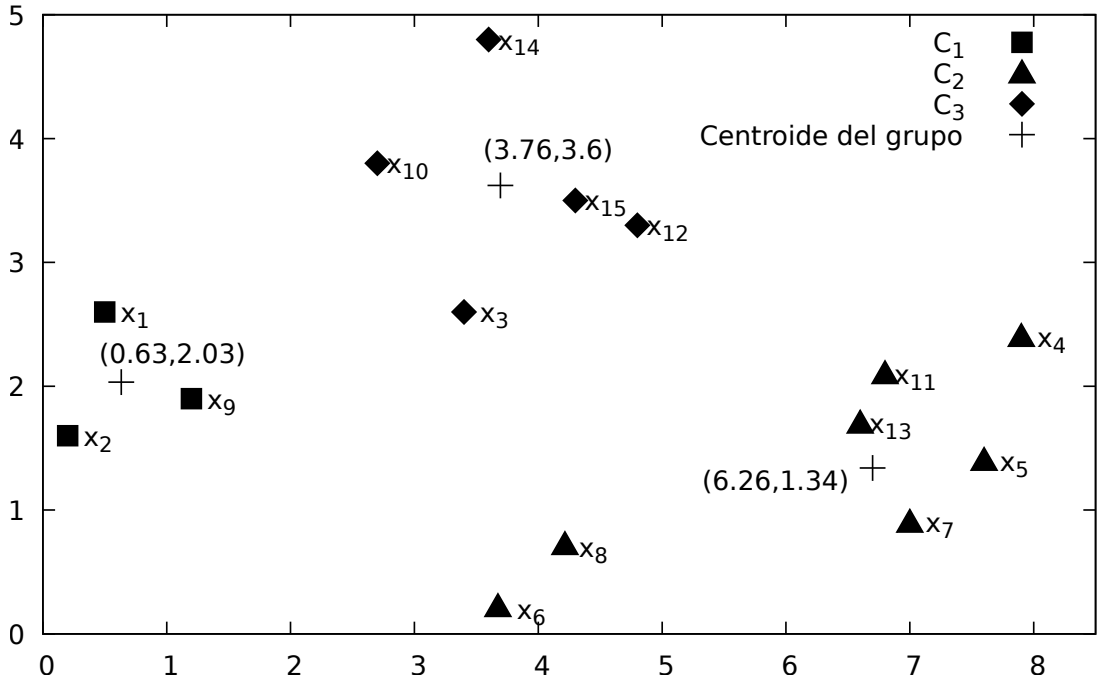
Por ejemplo, el algoritmo X-means [99] basado en estadísticas, determina el mejor número de grupos k_v, por medio de optimizar el criterio de información de Akaide (AIC) o el criterio de información bayesiano (BIC). CNAK [112] es otro algoritmo para determinar el valor óptimo de k_v, basado en el concepto de *estabilidad de la agrupación*.

Una diferencia sutil entre un algoritmo de k-fija y k-variable, el primero hace particiones "equivalentes" independientemente si el conjunto de datos tiene una estructura interna o no, mientras que el segundo obtiene resultados solo cuando el conjuntos de datos tienen una estructura interna, pero deben coincidir ambos cuando el número de grupos es conocido y las instancias están distribuidas con las propiedades de compactibilidad y separación, que son los criterios más aceptados en la formación de grupos [49].

Como ejemplo, un algoritmo de k-variable para el conjunto de datos ilustrativo, posiblemente encontrará que el número óptimo de grupos es $k = 4$ y las particiones son: las instancias



(a) Salida de la primera ejecución



(b) Salida de la segunda ejecución

Figura 2.1 Representación gráfica de dos diferentes ejecuciones del algoritmo `K-means` para conjunto de datos ilustrativo (Tabla 2.1) y $k = 3$

$\{x_6, x_8\}$ que pertenecen al grupo C_1, las instancias $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ son miembros del grupo C_3, las instancias $\{x_1, x_2, x_9\}$ son miembros del grupo C_3, y $\{x_4, x_5, x_7, x_{11}, x_{13}\}$ son miembros del grupo C_4. De manera gráfica se muestra en la Figura 2.2.

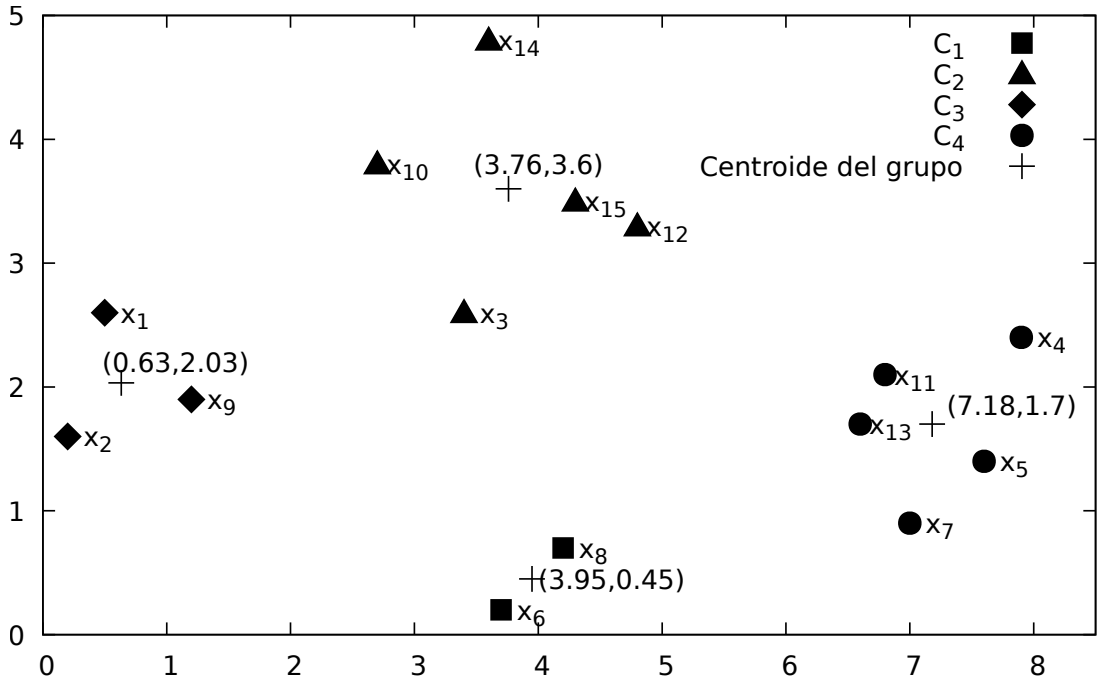


Figura 2.2 Representación gráfica de una agrupación del conjunto de datos ilustrativo (Tabla 2.1) para $k = 4$

2.3.2 Enfoques basados en instancias representativas

Los enfoques basados en instancias representativas o prototipos[1] buscan k instancias del conjunto de datos que representen a cada uno de los grupos, generalmente localizadas más en el centro, y el resto de las instancias son asignadas por la similitud a cada uno de ellos con una ecuación equivalente a (2.4), pero ahora las instancias representativas v_j son utilizadas como referencia, como se muestra en (2.7).

$$x_i \in C_j \leftrightarrow \mu_j \leftarrow \underset{1 \leq j \leq k}{\arg\min} \|x_i - v_j\|, \forall i \in \{1,2,...,n\} \tag{2.7}$$

Los enfoques basados en instancias representativas tienen como ventaja sobre los basados en centroides que no se ven afectados por conjuntos de datos que tengan valores atípicos. En

[1] A una instancia representativa de un grupo o prototipo en inglés se le conoce como *medoids*

los métodos basados en centroides, una instancia con valores extremadamente grandes puede distorsionar la distribución de los datos, este efecto puede aumentar al usar una CVI como la suma del error cuadrático (SSE). En cambio, si se usa una instancia representativa cuando se asigna el resto de las instancias serán más similares a su grupo [52].

Un algoritmo clásico dentro de estos enfoques es `PAM` (2), el cual es iterativo e incremental. Los pasos del algoritmo se detallan a continuación.

Algoritmo PAM

`Paso de inicialización,` primero son seleccionadas k instancias aleatoriamente, los cuales representan las instancias representativas de inicio de cada grupo (línea 2).

`Paso de construcción,` En el proceso iterativo, son remplazadas instancias que fueron seleccionadas como representativas por instancias que no son representativas seleccionadas aleatoriamente del conjunto de datos. El resultado es evaluando para ver si la agrupación es mejorada, con una función de costo S, que mide el promedio de desemejanza entre la instancia seleccionada y la más representativa del grupo (5) y (6)).

`Paso de intercambio,` Si el costo total es negativo, la calidad de la agrupación es mejorada al disminuir SED .

Algoritmo 2 Algoritmo `PAN`. Un algoritmo de agrupamiento basado en instancias representativas [52, 65]

Entrada: X: un conjunto de datos
k: número de grupos
Salida: Una partición de X en $\{C_j\}$ grupos y $\{v_j\}$ instancias representativas, donde $j : j \in \{1,2,...,k\}$

1: t $\leftarrow$ 1
2: Inicializar k instancias representativas $v_1^{(t)}, v_2^{(t)}, ..., v_k^{(t)}$ de X
3: **repeat**
4: Asignar el resto de instancias x_i a su grupo utilizando los v_j:
$C_j^{(t)} = \{x_i : \|x_i - v_j\| \leq \|x_i - v_{j'}\|,\ j' = 1,2,...,k\}$
5: Seleccionar una instancia no representativa x_i de manera aleatoria;
6: Calcular el costo total S, al intercambiar x_i por una $1v_j$ seleccionada de manera aleatoria
7: **if** $S < 0$ **then**
8: Intercambiar x_i por v_j el cual es un mejor instancia representativa de C_j
9: **end if**
10: t $\leftarrow$ t + 1
11: **until** hasta que no cambien v_j

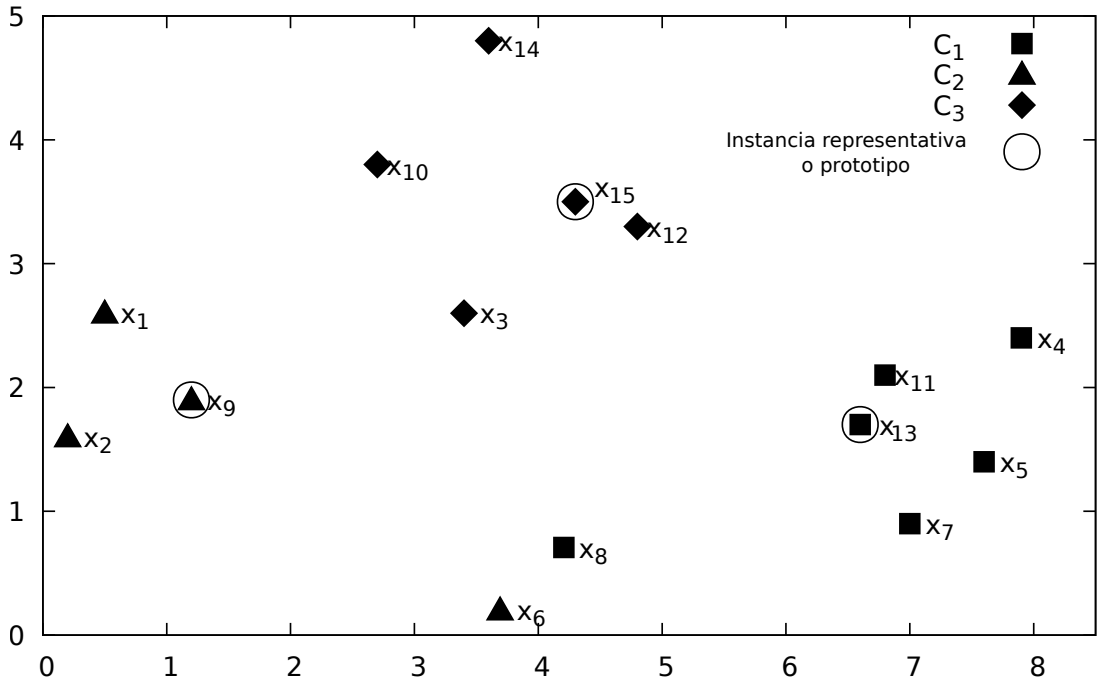


Figura 2.3 Representación gráfica de una agrupación basada en instancias representativas para el conjunto de datos ilustrativo para $k = 3$

Otra ventaja de los algoritmos basados en instancias representativas es evitar la evaluación exhaustiva de las distancias entre las instancias. A diferencia de los enfoques basados en centroides. Las distancias se pueden calcular solamente una vez y almacenarlas en la *matriz de disimilitud* ($D_{n\times n}$) que es una matriz simétrica y por tanto, puede ser manejada como una matriz triangular. De este modo, el costo de obtener una distancia entre un par de instancias sería $O(1)$ independiente de las dimensiones de las instancias. Así, cuantas más dimensiones tenga el conjunto de datos, mayor ventaja computacional se tendrá. Como desventaja, en este algoritmo una n grande se puede convertir en una desventaja por el espacio requerido de $O(n^2)$. Como ejemplo, la matriz de disimilitud del conjunto de datos ilustrativo (Tabla 2.1) se muestra en (2.8), utilizando norma euclidiana y redondeada a 2 decimales.

A continuación, se muestra una posible salida del algoritmo `PAN` para el conjunto de datos ilustrativo (Tabla 2.1) con $k = 3$ y considerando las siguientes instancias representativas:

$$v_1 = x_{13} = (6.6, 1.7)$$
$$v_2 = x_9 = (1.2, 1.9)$$
$$v_3 = x_{15} = (4.3, 3.5)$$

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
x_1	0.00	1.04	2.90	7.40	7.20	4.00	6.72	4.16	0.99	2.51	6.32	4.36	6.17	3.80	3.91
x_2	1.04	0.00	3.35	7.74	7.40	3.77	6.84	4.10	1.04	3.33	6.62	4.90	6.40	4.67	4.52
x_3	2.90	3.35	0.00	4.50	4.37	2.42	3.98	2.06	2.31	1.39	3.44	1.57	3.32	2.21	1.27
x_4	7.40	7.74	4.50	0.00	1.04	4.74	1.75	4.07	6.72	5.39	1.14	3.23	1.48	4.92	3.76
x_5	7.20	7.40	4.37	1.04	0.00	4.08	0.78	3.47	6.42	5.46	1.06	3.38	1.04	5.25	3.91
x_6	4.00	3.77	2.42	4.74	4.08	0.00	3.37	0.71	3.02	3.74	3.64	3.29	3.26	4.60	3.35
x_7	6.72	6.84	3.98	1.75	0.78	3.37	0.00	2.81	5.89	5.19	1.22	3.26	0.89	5.17	3.75
x_8	4.16	4.10	2.06	4.07	3.47	0.71	2.81	0.00	3.23	3.44	2.95	2.67	2.60	4.14	2.80
x_9	0.99	1.04	2.31	6.72	6.42	3.02	5.89	3.23	0.00	2.42	5.60	3.86	5.40	3.76	3.49
x_{10}	2.51	3.33	1.39	5.39	5.46	3.74	5.19	3.44	2.42	0.00	4.44	2.16	4.43	1.35	1.63
x_{11}	6.32	6.62	3.44	1.14	1.06	3.64	1.22	2.95	5.60	4.44	0.00	2.33	0.45	4.19	2.87
x_{12}	4.36	4.90	1.57	3.23	3.38	3.29	3.26	2.67	3.86	2.16	2.33	0.00	2.41	1.92	0.54
x_{13}	6.17	6.40	3.32	1.48	1.04	3.26	0.89	2.60	5.40	4.43	0.45	2.41	0.00	4.31	2.92
x_{14}	3.80	4.67	2.21	4.92	5.25	4.60	5.17	4.14	3.76	1.35	4.19	1.92	4.31	0.00	1.48
x_{15}	3.91	4.52	1.27	3.76	3.91	3.35	3.75	2.80	3.49	1.63	2.87	0.54	2.92	1.48	0.00

(2.8)

En la salida obtenida, v_1 asigna las instancias $\{x_4, x_5, x_7, x_8, x_{11}, x_{13}\}$ al grupo C_1, v_2 asigna $\{x_1, x_2, x_6, x_9\}$ al grupo C_2 y v_3 asigna $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ al grupo C_3. De manera gráfica, la agrupación se muestra en la Figura 2.3.

También se puede evaluar la calidad de la agrupación con SED, con la ayuda de la matriz de disimilitud (2.8):

$$\begin{aligned} \text{SED} &= \|x_4 - v_1\| + \|x_5 - v_1\| + \ldots + \|x_1 - v_2\| + \ldots + \|x_{15} - v_3\| \\ &= 1.48 + 1.04 + \ldots + 0.99 + \ldots + 0.99 \\ &\approx 28.98 \end{aligned}$$

2.3.3 Enfoques basados en una partición difusa

Los enfoques basados en una partición difusa pueden ser adecuados cuando los conjuntos de datos presentan sobre-posición de grupos [94]. Cada instancia tiene un grado de pertenencia a cada grupo especificado en el intervalo de $[0, 1]$, cercano a la unidad significa un alto grado de similitud entre la instancia y el grupo, mientras que cercano a cero implica poca similitud entre la instancia y el grupo [13]. Para almacenar los valores de los grados de pertenencia se usa una matriz U en $\mathbb{R}^{k \times n}$. De manera formal, para denotar el conjunto de todas las particiones difusas de los conjuntos de datos se emplea la ecuación (2.9).

$$Mfc = \{U \in \mathbb{R}^{k\times n} | \sum_{j=1}^{k} u_{ij} = 1, \sum_{i=1}^{n} u_{ij} \geq 1, \text{ y } u_{ji} \in [0,1], \text{ for } 1 \leq j \leq k \text{ y } 1 \leq i \leq n\} \quad (2.9)$$

Las dos condiciones que debe cumplir la matriz U por las restricciones impuestas en el problema de agrupación son:

$$\sum_{j=1}^{k} u_{ij} = 1, \forall i \in \{1,2,...,n\} \quad (2.10)$$

$$\sum_{i=1}^{n} u_{ij} \geq 1, \forall j \in \{1,2,...,k\} \quad (2.11)$$

La ecuación (2.10) indica que cada instancia (x_i) debe pertenecer a un grupo (C_j) y la ecuación (2.11) que cada grupo debe contener al menos una instancia.

De manera similar a la ecuación (2.6), se obtienen los centroides, pero ahora para una partición difusa con la matriz U es posible obtener los centroides por medio de la ecuación (2.12).

$$\mu_j \leftarrow \frac{\sum_{i=1}^{n} (u_{ji})^m x_i}{\sum_{i=1}^{n} (u_{ji})^m}, \forall j \in \{1,2,...,k\}, \quad (2.12)$$

donde m es el exponente de ponderación.

También, si los centroides son conocidos, es posible encontrar los grados de pertenencia por medio de la ecuación (2.13).

$$u_{ji} = \left(\sum_{j'=1}^{k} \left(\frac{d_{ji}}{d_{j'i}} \right)^{2/(m-1)} \right)^{-1}, \forall j \in \{1,2,...,k\} \text{ and } \forall i \in \{1,2,...,n\}, \quad (2.13)$$

donde d_{ji}^2 es la A-distancia elevada al cuadrado de la instancia x_i al centroide del grupo C_j que está calculada en la A-norma inducida como la ecuación (2.14).

$$d_{ji}^2 = \|x_i - \mu_j\|_A^2 = (x_i - \mu_j)^T A (x_i - \mu_j), \quad (2.14)$$

donde A es una matriz positiva definida y simétrica, al evaluar la distancia se puede utilizar alguna de las siguientes opciones $A = I$ norma euclidiana, $A = D_x^{-1}$ norma diagonal o $A = cov(x_i, x_j)^{-1}$ norma Mahalonobis que es la inversa de la matriz de covarianza [13].

Estos enfoques tienen como algoritmo representativo el algoritmo FCM (3). Sus pasos son indicados a continuación.

Algoritmo FCM

`Paso de inicialización,` una matriz $U^{(0)}$ es inicializada de manera aleatoria; para facilitar la inicialización los elementos u_{ij} pueden ser inicializados con 0 y 1, solo cumpliendo con las condiciones (2.10) y (2.11) (línea 2).

`Paso de actualización de` $\mu_j^{(t+1)}$, con la matriz actual de pertenencia $U^{(t)}$, los centroides son actualizados obteniendo $\mu_j^{(t+1)}$ con la ecuación (2.12) (línea 5).

`Paso de actualización de` $U^{(t+1)}$, Ahora con los nuevos centroides se obtiene la matriz $U^{(t+1)}$ con (2.13) (línea 6).

Al aplicar de manera iterativa los pasos de actualización, el algoritmo FCM va convergiendo a una solución y el error cuadrado mínimo funcional, J_m (6), se va minimizando hasta lograr grupos compactos.

Algoritmo 3 Algoritmo `FCM`. Un algoritmo basado en particiones difusas [13]

Entrada: X: un conjunto de datos
k: número de grupos
m: peso del exponente
$\|\cdot\|_A$: norma inducida en $\mathbb{R}^l$, $A_{l\times l}$ es una matriz de pesos positiva definida
Salida: U: una partición difusa de X en $\{C_j\}$ grupos y sus centroides $\{\mu_j\}$, donde $j : j \in \{1,2,...,k\}$

1: t $\leftarrow$ 0
2: Inicializar la matriz $U^{(t)}$, donde $U^{(t)} \in Mfc$
3: **repeat**
4: t $\leftarrow$ t + 1
5: Actualizar los centroides $\mu_j^{(t)}$ con (2.12)
6: Actualizar la matriz de pertenencia $U^{(t+1)} = [u_{ji}^{(t)}]$, con (2.13)
7: Compare $U^{(t+1)}$ con $U^{(t)}$ con alguna norma de matrices conveniente
8: **until** hasta que $\|U^{(t+1)} - U^{(t)}\| < \varepsilon$

Para el conjunto de datos ilustrativo (Tabla 2.1), una posible salida de la matriz $U_{3\times 15}$ por algoritmo FCM con los parámetros de $k = 3$, $m = 2$ y la norma euclidiana ($A = I$) se muestra a continuación.

Los centroides serían:

$$\mu_1 = (0.87, 1.97)$$
$$\mu_2 = (3.85, 3.28)$$
$$\mu_3 = (7.02, 1.65)$$

La matriz $U_{3\times 15}$ con los grados de pertenencia:

$$U_{3\times 15} = \begin{array}{c|ccccccccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ \hline C_1 & .95 & .95 & .09 & .02 & .01 & .33 & .02 & .23 & .99 & .18 & .01 & .05 & .01 & .12 & .02 \\ C_2 & .03 & .04 & .87 & .07 & .02 & .39 & .03 & .44 & .01 & .77 & .02 & .85 & .02 & .79 & .96 \\ C_3 & .02 & .01 & .04 & .91 & .97 & .28 & .95 & .33 & .00 & .05 & .97 & .1 & .97 & .09 & .02 \end{array} \tag{2.15}$$

Con los resultados se puede evaluar la calidad de la agrupación utilizando el CVI que minimiza FCM (Jm):

$$\begin{aligned} J_m(U,\{\mu_j\}) &= 0.95 \begin{bmatrix} 0.87-0.5 \\ 1.97-2.6 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.87-0.5 \\ 1.97-2.6 \end{bmatrix} + 0.03 \begin{bmatrix} 3.85-0.5 \\ 3.28-2.6 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3.85-0.5 \\ 3.28-2.6 \end{bmatrix} \\ &\quad + \ldots + 0.2 \begin{bmatrix} 7.02-4.3 \\ 1.65-3.5 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 7.02-4.3 \\ 1.65-3.5 \end{bmatrix} \\ &\approx 15.05333178 \end{aligned}$$

Por cuestiones de espacio los grados de pertenencia de la matriz fueron redondeados a dos decimales pero son suficientes de manera ilustrativa. Como se puede ver la matriz $U_{3\times 15}$ contiene los grados de pertenencia y define la partición del conjunto de datos, así la instancia x_1 por estar más próxima al centroide μ_1 tiene un grado mayor, los grados de pertenecía son 0.95, 0.04, y 0.01 con respecto a los grupos C_1, C_2 y C_3 respectivamente, se puede decir que tiene más semejanza con el grupo C_1. La instancia x_6, es una instancia que tiene los grados de pertenencia con respecto a los grupos de 0.33, 0.39 y 0.28, siendo valores muy similares y manteniendo similitud con los tres grupos.

Un caso particular de una partición difusa es cuando los elementos $u_{ji} \in \{0,1\}$, significa que cualquier instancia está solo en un grupo. La cual es otra manera para representar la relación de pertenencia entre las instancias y los grupos, a estas matrices se les llama *matriz de partición dura*[2]. El conjunto de todas las $k \times n$ matrices de partición dura son denota por Mh (2.16), así las matrices Mh es un subconjunto de Mf.

$$Mh = \{U \in \mathbb{R}^{k\times n} | \sum_{j=1}^{k} u_{ij} = 1, \sum_{i=1}^{n} u_{ij} \geq 1, \text{ y } u_{ji} \in \{0,1\}, \text{ for } 1 \leq j \leq k \text{ y } 1 \leq i \leq n\} \tag{2.16}$$

[2]En inglés se les conoce como *crisp partition matrix*

Una partición difusa puede ser convertida a una partición dura por medio de asignar cada instancia al grupo con el grado de membresía más alto [94], por ejemplo, la matriz (2.15) puede ser transformada a la matriz (2.17).

$$U_{3\times 15} = \begin{array}{c|ccccccccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ \hline C_1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ C_3 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \tag{2.17}$$

La interpretación de la matriz (2.17) es: las instancias $\{x_1, x_2, x_9\}$ son miembros del grupo C_1 por estar asignado 1 en los elementos $U(1,1)$, $U(1,2)$ y $U(1,9)$; las instancias $\{x_4, x_5, x_7, x_{11}, x_{13}\}$ son miembros del grupo C_2 porque $U(2,4)$, $U(2,5)$, $U(2,7)$ y $U(2,11)$ contienen 1 y las instancias $\{x_3, x_6, x_8, x_{10}, x_{12}, x_{14}, x_{15}\}$ son miembros del grupo C_3 porque $U(3,3)$, $U(3,6)$, $U(3,8)$, $U(3,10)$, $U(3,12)$, $U(3,14)$ y $U(3,15)$ contienen 1.

2.4 Algoritmos bio-inspirados para agrupación basada en particiones

Debido a la complejidad del problema de agrupación basado en particiones y que no siempre es posible contar con la solución óptima, pero es deseable al menos obtener una solución que garantice cierta calidad en la agrupación, el diseño de algoritmos de agrupación basados en particiones es un tema abierto de investigación. Dentro de los diferentes enfoques de diseño se encuentran los algoritmos bio-inspirados. Los algoritmos bio-inspirados son técnicas de resolución de problemas que se basan en principios y estrategias que imitan o se inspiran en procesos y comportamientos observados en la naturaleza. Estos algoritmos toman como referencia la evolución biológica, el comportamiento de los seres vivos, y otras analogías con sistemas naturales para abordar desafíos en diversas áreas. Estos algoritmos han demostrado ser útiles para resolver problemas complejos en diversas áreas. Su principal ventaja radica en su capacidad para abordar problemas no lineales y encontrar soluciones cercanas a óptimas en espacios de búsqueda grandes y complejos, que son los encontrados en el problema de agrupamiento [94].

En esta sección se incluye una descripción de los principales algoritmos bio-inspirados que se han utilizado, lo que nos permitirá ubicar el trabajo que se realiza en esta tesis.

Debido al excelente rendimiento de los algoritmos bio-inspirados en la resolución del problema de agrupamiento basado en particiones, se encuentran diferentes propuestas, así

como diferentes revisiones que ven con atención los trabajos relacionados. Una clasificación y taxonomía de los enfoques meta-heurísticos se puede encontrar en [23, 94]. Esta revisión menciona las propuestas más comunes utilizadas por los algoritmos de agrupación basados en particiones. De acuerdo con esta taxonomía se encuentran:

- Algoritmos evolutivos [23]. Son técnicas de búsqueda inspirada en el mecanismo de "evolución por selección natural" formulado por Darwin. Por medio de una población de individuos que tienen un valor de aptitud y son codificados por un cromosoma se representan las soluciones candidatas a resolver el problema planteado.

  - Algoritmos genéticos (GAs, de su abreviatura en inglés) [57]. En este caso, como son los estudiados en este trabajo, se utilizará la abreviatura en español (AGs). Es un modelo computacional que imita el fenómeno de evolución natural para resolver problemas en un dominio amplio, propuesto inicialmente por [57], y estudiado más tarde por [46, 87]. En la evolución natural, cada especie busca adaptaciones beneficiosas a un entorno en constante cambio. A medida que evolucionan las especies, los nuevos atributos se codifican en el cromosoma de cada uno de los individuos de una población. Esta información cambia por mutación aleatoria, pero la verdadera fuerza impulsora detrás del desarrollo evolutivo es la combinación, al intercambiar material genético durante la reproducción [64].
  - Estrategias de evolución (ESs, del inglés Evolutionary Strategy) [11]. Es una técnica basada sobre ideas de adaptación y evolución, usa una representación dependiente del problema, ante todo con operadores de mutación y selección como operadores de búsqueda.

- Algoritmos de enjambre (SI, del inglés Swarm Intelligence) [10]. El comportamiento colectivo y social de los seres vivos motivó a los investigadores a emprender el estudio de lo que hoy se conoce como Inteligencia de enjambre. El término *enjambre* se usa de manera general para referirse a cualquier colección restringida de agentes o individuos que realizan o ejecutan ciertas operaciones. El ejemplo clásico de un enjambre es cuando las abejas pululan alrededor de su colmena, pero la metáfora puede extenderse fácilmente a otros sistemas con características similares. Se puede pensar en una colonia de hormigas como un enjambre y cada una de ellas es un agente de manera individual. Una bandada de pájaros también puede ser visto como un enjambre y cada pájaro como un agente [64].

  - Algoritmos de colonia de hormigas (ACO, del inglés Ant Colony Optimization) [27]. Es una técnica probabilística para resolver problemas computacionales que se centra

en encontrar buenos caminos a través de grafos. Se inspiran en el comportamiento de las hormigas para encontrar caminos desde la colonia hasta la comida. En el mundo real, las hormigas inicialmente deambulan al azar para encontrar comida y regresan a su colonia mientras dejan rastros de feromonas. Si otras hormigas encuentran ese camino, es probable que no sigan viajando al azar, sino que sigan el rastro, regresando y reforzándolo si finalmente encuentran comida [23].

  - Algoritmos de enjambre de partículas (PSO, del inglés Particle Swarm Optimization) [67]. Es una técnica que no requiere ninguna información del gradiente de la función a optimizar utiliza únicamente operadores matemáticos primitivos y es conceptualmente muy simple. En PSO, inicializa una población de "partículas" conceptuales con posiciones aleatorias y velocidades, y se evalúa una función f, usando la posición de la partícula como variable independiente. A través de modelos matemáticos se actualiza la posición y velocidad de las partículas [23].
  - Colonia de abejas artificiales (ABC, del inglés Artificial Bee Colony) [64]. Es una técnica que imita el comportamiento de búsqueda de alimento que conduce al surgimiento de la inteligencia colectiva de los enjambres de abejas. Consta de tres componentes esenciales: fuentes de alimento, recolectores empleados y recolectores desempleados y el modelo define dos modos principales de comportamiento: reclutamiento a través de una fuente de néctar y el abandono de una fuente.

- Sistemas inmunitarios artificiales (AIS, del inglés Artificial Immune Systems). Esta técnica se ha desarrollado modelando computacionalmente procesos inmunes biológicos, abstrayendo esos modelos en algoritmos e implementándolos en el contexto de la ingeniería [101]. Utilizan conceptos inmunológicos, como selección clonal, diversidad y memoria inmunológica para encontrar soluciones adecuadas.

- Optimización por búsqueda bacteriana (BFO, del inglés Bacterial Foraging Optimization). Esta técnica se basa en el comportamiento de las bacterias para resolver problemas de optimización. Fue propuesto por primera vez por Passino [98] y está inspirado en el proceso de búsqueda de alimento de las bacterias. El algoritmo BFO simula el comportamiento de las bacterias individuales y cómo interactúan con su entorno mientras buscan alimentos en un medio ambiente. El algoritmo BFO traduce este proceso de búsqueda de alimentos de las bacterias en un proceso de optimización que se utiliza para encontrar soluciones óptimas o cercanas a óptimas en problemas complejos. El proceso de búsqueda se modela como una población de bacterias virtuales en un espacio de búsqueda multidimensional.

En esta tesis, se centrará en los AGs, los cuales serán descritos con detalle en la siguiente sección.

2.5 Estudios del estado del arte del problema de agrupación

Debido al rápido avance de los estudios realizados para resolver el problema de agrupamiento, se han publicado diferentes revisiones para mostrar un conocimiento actualizado del estado actual de la investigación. En esta sección, con la finalidad de poner en relieve el trabajo de la tesis, se analiza las revisiones anteriores que incluyen entre los métodos de resolución del agrupamiento a los AGs.

El primer trabajo representativo que es encontrado es el de Hruschka et al. (2009)[59]. Hacen una revisión de EAs para el agrupamiento centrándose en los AGs principalmente, también incluyen algunos trabajos de EAs multi-objetivos y modelos de aprendizaje por conjuntos[3]. Realizan un análisis descriptivo de cada AG incluido concluyendo con una taxonomía de los AGs que destaca aspectos relevantes en el contexto de la agrupación. Esta revisión es muy interesante, de hecho, la descripción y taxonomía presentada en esta tesis se basa en los conceptos importantes presentados en esta referencia. Como aporte interesante al trabajo de revisión que se realizó en esta tesis, se pueden encontrar nuevas referencias, se actualiza con información hasta 2022, una taxonomía más detallada y la consideración de un estudio experimental para evaluar el desempeño de las propuestas.

Posteriormente, Nanda y Panda (2014) [94] presentan un estudio descriptivo más general de algoritmos meta-heurísticos inspirados en la naturaleza y nuevas aplicaciones del agrupamiento basado en particiones. La revisión considera diferentes meta-heurísticas, como AG, ES, ACO, PSO, ABC, AIS y BFO. El hecho de que se incluyan tantas técnicas hace que sea una revisión muy general no entrando en profundidad en la descripción de los algoritmos. Así, no existe una descripción específica de las particularidades de los AGs ni una taxonomía específica para ellos. Tampoco se lleva a cabo un análisis experimental para evaluar la capacidad de los AGs para resolver los diferentes problemas de agrupamiento.

Mukhopadhyay et al. (2015) [90] hace una revisión completa de AGs multi-objetivo para resolver el problema de la agrupación. Los autores realizaron una adecuada categorización y descripción de los métodos. La principal diferencia con la revisión realizada en esta tesis es que considera diferentes algoritmos ya que la revisión realizada aquí analiza AG mono-objetivos. Además, se consideran referencias actualizadas a 2022 e incluye un estudio experimental con toda la información disponible para la comunidad científica.

[3]En inglés se le conoce ensemble learning

Posteriormente, Hancer y Karaboga (2017) [53] hacen un análisis descriptivo de los algoritmos de agrupamiento que buscan de manera óptima el número de grupos, k. Así, analizan diferentes técnicas incluyendo AGs. Se centra únicamente en algoritmos que determinan automáticamente el número de grupos considerando tanto enfoques mono-objetivo, como multi-objetivo. Incluyen diferentes algoritmos bio-inspirados como AG, PSO, BFO y AIS, entre otros. Este estudio, como los que incluyen diferentes técnicas, es más general. Por lo tanto, no es posible realizar una categorización y taxonomía específica para los AG. Además, en esta tesis se trabaja también con AG que funcionan con un valor fijo de k y se incluye un estudio experimental para evaluar la calidad de los diferentes enfoques.

Más recientemente, Kayaalp y Erdogmus (2020) [66], presentan otra revisión reciente de AGs, PSO y otros algoritmos meta-heurísticos bio-inspirados no tan comunes como *optimización basada en biogeografía* y *optimización de lobo gris*. El trabajo realiza un estudio experimental, sin embargo, no se realiza una revisión exhaustiva de propuestas para cada enfoque. Se selecciona y analiza experimentalmente una propuesta relevante para cada enfoque. Su estudio experimental considera 12 conjuntos de datos, 5 algoritmos y 4 métricas de rendimiento. De este modo, también hay diferencias significativas con el estudio que se lleva a cabo en esta tesis donde se incluyen 22 propuestas de AGs, 94 conjuntos de datos y 22 medidas de desempeño.

Finalmente, Ezugwu et al. (2022) [33], recientemente presentan un estudio descriptivo más general del problema de agrupación. Incluye agrupación tanto jerárquica como basada en particiones. Además, se consideran tanto algoritmos bio-inspirados como otro tipo de propuestas. El hecho de que se incluyan tantas técnicas hace que sea una revisión muy general. Por lo tanto, no existe una descripción específica de las particularidades de los AGs ni una taxonomía específica para ellos. Tampoco realizan un análisis experimental para evaluar las diferentes propuestas.

Un resumen de la información más relevante de cada revisión se muestra en las Tablas 2.2 y 2.3.

Tabla 2.2 Un resumen de los estudios previos que realizan un estado del arte de la agrupación e incluyen a los AGs para su resolución (I/II)

Referencia	Años	Breve descripción
[59]	1992-2009	• Un estudio descriptivo y con taxonomía basado en las propiedades de los algoritmos evolutivos: representaciones, operadores evolutivos y función de aptitud. • Está enfocado a diferentes tipos de EAs tanto mono-objetivo como multi-objetivo y aprendizaje por conjuntos. En concreto, se consideran 13 AGs mono-objetivos. • No hay un estudio experimental.
[94]	1989-2014	• Un estudio descriptivo sobre algoritmos bio-inspirados. No se da una taxonomía específica para los algoritmos considerados, se agrupan por tipo de propuesta. • Se centra en EAs y otros algoritmos bio-inspirados. Concretamente, se incluyen 2 AGs mono-objetivo. • No hay un estudio experimental.
[90]	1993-2015	• Un estudio descriptivo con taxonomía sobre AEs multi-objetivo considerando: estrategias de codificaciones adoptadas, funciones objetivo, operadores evolutivos, estrategia para mantener soluciones no dominadas y métodos de selección. • Se centra en EAs multi-objetivo. Concretamente, no se considera ningún AG mono-objetivo. • No hay un estudio experimental.
[53]	1991-2017	• Un estudio descriptivo de los enfoques tradicionales y evolutivos que encuentran de manera óptima el número de grupos. No se proporciona una taxonomía específica para los AGs. Se muestra representación y función objetivo. • Se centra en EAs, PSO y otros algoritmos bio-inspirados. Concretamente, se consideran 16 AG mono-objetivos, pero no se especifican, categorizan ni estudian en detalle. • No hay un estudio experimental

Tabla 2.3 Un resumen de los estudios previos que realizan un estado del arte de la agrupación e incluyen a los AGs para su resolución (II/II)

Referencia	Años	Breve descripción
[66]	2011-2018	• Un estudio descriptivo que incluye algoritmos de optimización evolutivos y otros algoritmos bio-inspirados. No se da una taxonomía o categorización específica para los algoritmos considerados, se describen y agrupan por tipo. • Se centra en PSO, AGs, optimización de lobo gris y optimización basada en biogeografía. Concretamente, se considera 1 GA mono-objetivo. • Hay un estudio experimental con 12 conjuntos de datos, 5 propuestas y 4 medidas de rendimiento.
[33]	1991-2021	• Un estudio descriptivo que incluye algoritmos de optimización evolutivos y otros algoritmos bio-inspirados. Además, se aborda el agrupamiento jerárquico y basado en particiones. Se da una taxonomía o categorización específica para todos los algoritmos, es general y no se enfoca en características específicas de los AGs. • Se centra en ACO, AG, PSO y otros algoritmos de optimización no inspirados en la naturaleza. Concretamente, se consideran 10 AG mono-objetivos. • No hay un estudio experimental.
Esta tesis	1991-2022	• Un estudio descriptivo y taxonómico basado en las propiedades de los AG mono-objetivo. Se consideran representaciones, operadores evolutivos, estrategias de búsqueda y funciones de aptitud. • Se consideran 22 AGs mono-objetivo. • Se incluye un estudio experimental con 94 conjuntos de datos, 22 propuestas y 22 medidas de rendimiento. Todos los algoritmos, configuraciones y experimentos están disponibles en LEAC [109] desarrollada en esta tesis.

Capítulo 3

Algoritmos genéticos para agrupación

Los algoritmos genéticos (AGs) [46, 57, 87] son métodos de optimización eficientes de búsqueda que simulan el mecanismo de “evolución por selección natural" formulado por Darwin. Por medio de una población de individuos que varían, los individuos mejor adaptados sobreviven y transmiten sus características exitosas a sus hijos. Los AGs han sido usados ampliamente para encontrar soluciones óptimas o muy cercanas a la solución global en espacios complejos y grandes.

En este capítulo se describen los componentes que conforma un AG y se analizan las características particulares de cada uno de ellos para resolver el problema de agrupación. Primero, se establecen las características de los diferentes componentes de manera individual. Después, estos componentes son analizados formando parte de los algoritmos concretos que hay desarrollados. Finalmente, a partir de toda la información disponible, se define una taxonomía que permita clasificar y comparar los AGs existentes para agrupación.

3.1 Componentes de los algoritmos genéticos para agrupación

Un AG está inspirado en el mecanismo de “evolución por selección natural" y su esquema general es descrito en el Algoritmo (4). En general, las partes principales de un AG son las siguientes:

- Comienza con una población inicial $\mathscr{P}(0)$ de n_P individuos (línea 1), representando un conjunto de soluciones potenciales para el problema a resolver. Una *codificación* es utilizada para representar los individuos, la cual debe permitir representar el espacio completo de posibles soluciones.

- Una *función de aptitud* ($\mathscr{F}(\chi_\iota)$)[1] se usa para evaluar cada individuo y determinar si es apto para sobrevivir, equivale a que sea una solución apta para resolver el problema. La función de aptitud para evaluar a los individuos es usada en las líneas 4 y 9.
- Un mecanismo de *selección* se utiliza para conformar las siguientes generaciones, tanto predecesores como hijos (línea 7).
- Los *operadores genéticos* son los encargados de generar los cambios en los individuos, por ende, estos hacen una explotación y exploración en las soluciones del problema a resolver. Los operadores evolutivos más comunes para generar nuevas soluciones a lo largo de la evolución son *cruce* y *mutación* (línea 8).
- El proceso evolutivo continúa por un número de generaciones hasta que se logre una condición, llamado ciclo evolutivo, establecido en las líneas de 5 a 11.

En la construcción de un AG, se puede ver que está compuesto por un conjunto de componentes. Estos componentes pueden ser escogidos de manera similar que cuando se construye un software, por medio de un catálogo estándar. Un análisis detallado de las diferentes componentes que han sido utilizadas para resolver el problema de agrupación será descrito en las siguientes secciones.

Algoritmo 4 Algoritmo genético general [46, 87]

$\mathscr{P}$(t): *población de individuos*
t $\leftarrow$ 0
inicializa $\mathscr{P}$(t)
evalua $\mathscr{P}$(t)
while no condición de terminación **do**
  t $\leftarrow$ t + 1
  seleccionar $\mathscr{P}'$(t) de $\mathscr{P}$(t)
  $\mathscr{P}''$(t) $\leftarrow$ aplicar operadores genéticos a $\mathscr{P}'$(t)
  evaluar $\mathscr{P}''$(t)
  $\mathscr{P}$(t+1) $\leftarrow$ actualizar la población considerando $\mathscr{P}$(t) and $\mathscr{P}''$(t)
end while

3.1.1 Codificación de individuos

Establecer la codificación de los individuos que forman parte de la población es una de las decisiones importantes en el diseño de un AG, la cual está relacionada al problema específico

[1] en inglés se le conoce como *fitness function*

a resolver. La codificación utilizada debe tener varias propiedades, la principal es permitir a los individuos explorar el espacio completo de soluciones. El resultado obtenido por un AG depende en gran parte del tipo de dato que es utilizado en la codificación para representar a los individuos: binaria, entera o real, y además su fenotipo asociado: basado en centroide, instancia representativa, etiqueta, árbol o grafo.

La codificación más común empleada es un vector de caracteres simulando una cadena de material genético:

α_1	α_2	...	α_n

De acuerdo con el enfoque del AG, cada vector representa un *cromosoma* de un individuo, en otro nivel de abstracción una solución al problema a resolver por el algoritmo. Cada elemento del vector (α_i) es llamado *gen* y puede ser de diferentes tipos de datos, tales como números binarios, enteros o reales o valores nominales.

En la literatura se han propuesto varios esquemas de codificación para la solución del problema de agrupación, la cual ha servido de base para definir diferentes clasificaciones de los AGs [43, 59, 91].

En este trabajo se propone una taxonomía basada en las codificaciones de los AGs que indique la forma de pertenencia de cada objeto x_i del conjunto de datos X a un grupo C_j. Se consideran dos categorías, la primera define la pertenencia de los objetos a un grupo de manera explícita. La segunda utiliza un procedimiento para determinar la pertenencia. Por tanto, una clasificación de acuerdo con su codificación determina dos clases: *explícita* e *implícita*.

Codificación explícita

Para una codificación explícita, la partición de un conjunto de datos X, es determinada por el mismo cromosoma. Cada i-ésimo gen (α_i) determina directamente la pertenencia de la i-ésima instancia (x_i) a un grupo específico, sin necesidad de realizar un procedimiento. La manera común es etiquetar cada x_i con la inscripción del grupo al cual pertenece. Esta codificación puede ser clasificada por el tipo de dato de la *etiqueta* del grupo:

- *Basada en etiquetas de números enteros para los grupos*

  En esta codificación a cada grupo se le asigna una etiqueta de un número entero, así, los cromosomas χ_α son representados por (3.1).

$$\chi_\alpha = \langle \alpha_i \rangle, \text{ donde } \alpha_i : \alpha_i \in \{1,2,...,k\} \text{ y } i : i \in \{1,2,...,n\} \tag{3.1}$$

En (3.1) $\langle \alpha_i \rangle$ es un vector de números enteros, cada elemento del vector (gen) es una etiqueta de alguno de los grupos. La longitud de cada cromosoma es igual al número de instancias n del conjunto de datos. Los valores enteros están comprendidos en el rango desde 1 hasta k (número de grupos). Así, si el i-ésimo gen contiene un valor entero j, entonces la instancia x_i pertenece al grupo C_j.

Como ejemplo ilustrativo, un cromosoma χ_α (3.1) que codifica la partición mostrada en la Figura 2.2 con 15 instancias ($n = 15$) y 4 grupos, las etiquetas serían $\{1,2,3,4\}$ y el vector de genes es el siguiente:

3	3	2	4	4	1	4	1	3	2	4	2	4	2	2
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

El cromosoma representa una solución donde las instancias $\{x_6, x_8\}$ son miembros del grupo C_1, las instancias $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ son miembros del grupo C_2, las instancias $\{x_1, x_2, x_9\}$ son miembros del grupo C_3 y $\{x_4, x_5, x_7, x_{11}, x_{13}\}$ son miembros del grupo C_4.

Una ventaja de esta codificación es que tiene una complejidad computacional de $O(1)$ para encontrar la pertenencia de una instancia x_i a su grupo C_j. En este caso, encontrar los centroides de los grupos tomaría un tiempo de $O(n)$ usando la ecuación (3.2); aunque es un tiempo aceptable, si se realiza muchas veces esta operación dejaría de ser una ventaja significativa. Otra ventaja de esta codificación es permitir definir grupos de formas irregulares a diferencia de la codificación basada en centroides que está encaminada a encontrar grupos encerrados en n-esferas. La codificación también puede ser utilizada para agrupar instancias con atributos de diferentes dominios tales como continuos y nominales. Además, el número de grupos considerados en la partición de X, es independiente del tamaño del cromosoma χ_α.

$$\mu_j \leftarrow \frac{\sum_{\alpha_i=j} x_i}{\sum_{\alpha_i=j} 1}, \forall j \in \{1,2,...,k\} \tag{3.2}$$

La desventaja de esta codificación estaría en los conjuntos de datos grandes. En este caso, el tamaño del individuo será considerablemente grande para su almacenamiento y al aplicar los operadores se requeriría un número de generaciones proporcional a n, a menos que se implemente una búsqueda local. También al cambiar un gen en la mutación puede resultar que no tiene un impacto en la función de aptitud, requiere de operadores genéticos más complejos que les permitan salir de óptimos locales.

Otra desventaja de esta codificación es la redundancia, múltiples cromosomas representan la misma agrupación [18, 103]. Así el espacio de búsqueda a explorar es repetido haciéndolo mucho más grande; es demostrable que una codificación menos redundante resulta en un AG más eficiente en el problema de agrupación [35]. Cuando los cromosomas presentan esta característica, la cual está relacionada al problema, es llamada codificación *una-a-muchas* [59]. Por ejemplo, un cromosoma χ'_α:

3	3	1	4	4	2	4	2	3	1	4	1	4	1	1
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

Resulta que es equivalente al cromosoma del ejemplo anterior, al estar las mismas instancias compartiendo un mismo grupo, $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ en el grupo C'_1, $\{x_6, x_8\}$ en el grupo C'_2, $\{x_1, x_2, x_9\}$ al grupo C'_3 y $\{x_4, x_5, x_7, x_{11}, x_{13}\}$ son miembros del grupo C'_4. Así, los grupos son equivalentes $C'_1 = C_2$, $C'_2 = C_1$, $C'_3 = C_3$ y $C'_4 = C_4$. Una posible solución para reducir el espacio de búsqueda a explorar en esta codificación es por medio de utilizar un procedimiento de remuneración en los cromosomas [36].

- *Basada en etiquetas de números enteros con los k grupos*

  Es una variante de la codificación basada en etiquetas de números enteros. En esta codificación se concatena el número de grupos k de la solución al final del cromosoma, es formulado por (3.3).

$$\chi_{\alpha k} = [\langle \alpha_i \rangle | k] \tag{3.3}$$

Esta codificación es apropiada para ser utilizada en AGs de k-variable. Al incluir k en la codificación es posible saber el número de grupo del cromosoma en $O(1)$.

Por ejemplo, para el caso ilustrativo mostrado en la Figura 2.2 de 15 instancias ($n = 15$), la codificación del cromosoma se representa con las etiquetas $\{1,2,3,4\}$ y el vector es el siguiente:

3	3	2	4	4	1	4	1	3	2	4	2	4	2	2	4
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	

El cromosoma representa una agrupación de 4 grupos, cada uno de ellos con las instancias $\{x_6, x_8\}$, $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$, $\{x_1, x_2, x_9\}$ y $\{x_4, x_5, x_7, x_{11}, x_{13}\}$ para C_1, C_2 C_3 y C_4, respectivamente.

Una variante de esta codificación se obtiene al agregar las etiquetas de cada grupo de manera explícita [2], la notación para especificar a este cromosoma está dada por (3.4).

$$\chi_{\alpha\,1..k} = [\langle \alpha_i \rangle | 1\ 2\ ...\ k] \tag{3.4}$$

.

A la primera parte del cromosoma $\chi_{\alpha\,1..k}$ se le llama *sección de elementos* y a la segunda parte *sección de grupos* [2]. Por ejemplo, un cromosoma que codifica los mismos grupos representados en el ejemplo anterior sería:

3	3	2	4	4	1	4	1	3	2	4	2	4	2	2	1	2	3	4
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}				

Las ventajas y desventajas de esta codificación serían las mismas que la codificación basada en etiquetas con números enteros, solo tendría la desventaja en el espacio de almacenamiento de la segunda parte del cromosoma, para soluciones de k grande. En la primera variante puede ser generada la sección de grupos, cuando es requerida evitando el espacio extra.

- *Basada en una matriz con dígitos binarios*

  La codificación de individuos con matrices de dígitos binarios es el caso particular de una partición difusa descrita en la sección 2.3.3. En este caso, la relación de pertenencia entre instancias y grupos se da usando una matriz $U_{k\times n} = [u_{ji}] \in Mh$ (2.16). Así la matriz $U_{k\times n}$ es una clase de cromosomas que codifican con valores binarios.

  Como ejemplo ilustrativo, un cromosoma $U_{3\times 15}$ que codifica el conjunto de datos de la Tabla 2.1 con 15 instancias ($n = 15$), con $k = 3$ grupos sería:

$$U_{3\times 15} = \begin{array}{c|ccccccccccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \\ \hline C_1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ C_3 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

En este ejemplo, el cromosoma representa exactamente los mismos grupos de la matriz (2.17). Esta codificación fue una de las primeras propuestas en los AGs de agrupación [12].

En cuanto a las ventajas y desventajas de esta codificación, es muy similar a la codificación basada en etiquetas de números enteros para los grupos. La principal diferencia que se puede resaltar es el tiempo computacional para obtener la pertenencia de la instancia x_i a su grupo, en el peor de los casos es necesario comprobar que el elemento que contiene 1 en la columna $U(*,i)$, requiriendo un tiempo de $O(k)$, lo que puede ser una desventaja para un número grande de k, comparado con el $O(1)$ de la codificación Basada en etiquetas de números enteros para los grupos.

Otra ventaja, dado que las $U \in Mh$ (2.16), y estas a su vez son $Mh \subset Mfc$ (2.9), por lo que se tiene el soporte de las ecuaciones de una partición difusa para calcular los centroides (2.12) y al actualizar la pertenencia (2.13) para $m = 1$, se logra minimizar ahora la función J_1.

Codificación implícita

Los cromosomas que definen la pertenencia de las instancias a su grupo de manera indirecta son considerados en la categoría *codificación implícita*. En esta categoría propuesta, la característica que distingue a esta codificación es la utilización de un procedimiento para definir la pertenencia.

Los cromosomas que codifican centroides o instancia representativa son típicos ejemplos de codificación implícita, al utilizar el procedimiento definido por la ecuación (3.5).

$$\upsilon_j \leftarrow \underset{1 \leq j \leq k}{\arg\min} D(x_i, \upsilon_j), \forall i \in \{1,2,...,n\} \tag{3.5}$$

Donde υ_j es centroide (μ_j) o una instancia representativa (ν_j) del grupo C_j dependiendo del objetivo de la investigación y D es una función distancia tal como Minkowski [8], euclidiana [7, 84], Diagonal o Mahalonobis [12]. Así el subíndice j de υ_j es utilizado como etiqueta del grupo ($\alpha_i \leftarrow j$).

Esta categoría de codificación es dividida en tres subcategorías: basadas en *centroides*, *instancia representativa* o *grafos*, las cuales son descritas a continuación.

- *Basada en centroides con números decimales*

  Los cromosomas de esta codificación están constituidos por una cadena de coordenadas en formato de números decimales, las cuales codifican los centroides de los grupos. La longitud de la cadena es de $k \times d$, donde k el número de grupos y d la dimensionalidad del conjunto de datos.

  Dado que d es invariante al aplicar algún operador que intercambia genes, por tanto, si se trabaja con k-variable, los genes deben ser intercambiados donde inicia o termina un

centroide (μ_j) y de esta manera evitar inconsistencias. Así, los centroides de los grupos son considerados indivisibles, es decir, si es aplicado el operador de cruce de un punto solo puede ser aplicado entre dos centroides [6], logrando que los nuevos cromosomas estén conformados con las dimensiones correctas para cada uno de sus centroides.

Para el caso de la agrupación de k-fija, un AG podrá considerar como gen una coordenada x_{jd} o un centroide μ_j. Así, al aplicar los operadores genéticos por ser un tamaño fijo todos los cromosomas al intercambiar genes se mantiene el tamaño $k \times d$.

Por lo comentado anteriormente, se tendrán dos variantes de la codificación basados en centroides:

- La primera se puede ver el cromosoma formado por genes con las coordenadas x_{jd}, denotado por χ_x (3.6).

$$\chi_x = \langle x_{lj} \rangle = \langle x_{11}, x_{12}, ..., x_{1d}, x_{21}, x_{22}, ..., x_{2d}, x_{k1}, x_{k2}, ..., x_{kd} \rangle \tag{3.6}$$

- La segunda variante utiliza genes con los centroides μ_j. La notación para definir un cromosoma de este tipo es por χ_μ (3.7).

$$\chi_\mu = \langle \mu_j \rangle = \langle \mu_1, \mu_2, ..., \mu_k \rangle \tag{3.7}$$

Como ejemplo ilustrativo de un cromosoma χ_x (3.6), para la agrupación mostrada en la Figura 2.2, de 15 instancias ($n = 15$), 4 grupos ($k = 4$) y en el espacio bidimensional ($d = 2$), es el siguiente:

3.95	0.45	3.76	3.60	0.63	2.03	7.18	1.70
μ_1		μ_2		μ_3		μ_4	

Este individuo representa una solución donde el centroide del grupo C_1 es $\mu_1 = (3.95, 0.45)$, el centroide del grupo C_2 es $\mu_2 = (3.76, 3.60)$, el centroide del grupo C_3 es $\mu_3 = (0.63, 2.03)$ y el centroide del grupo C_4 es $\mu_4 = (7.18, 1.70)$. Así, cada centroide atrapa las instancias más cercanas a él, al aplicar (3.5) con la distancia euclidiana, conformándose así los grupos.

Desde el punto de vista de complejidad computacional esta estrategia de codificación es recomendada por [90], debido a la longitud pequeña de los cromosomas. Por tanto, lleva menos tiempo aplicar los operadores genéticos tales como cruce y mutación. También

el uso de valores reales introduce más flexibilidad y variabilidad en la aplicación de diferentes operadores genéticos.

Otra ventaja de esta codificación es la longitud al no depender del número de instancias. Normalmente, el número de dimensiones es mucho menor que el número de instancias. Por lo tanto, es apropiada para grandes conjuntos de datos. Sin embargo, en conjunto de datos con una gran dimensionalidad, la longitud de los cromosomas aumenta sustancialmente y la operación de cruce también puede llevar a un consumo elevado de tiempo.

Como se comentó, esta codificación se puede aplicar en agrupación de k-fija o k-variable. En este contexto, los primeros son de tamaño fijo, mientras que los segundos se inicializan de diferentes tamaños especificando un rango de búsqueda para k desde una $k_{\min}$ hasta $k_{\max}$. Una regla general utilizada ampliamente en varias propuestas [3, 55, 97] para estimación automáticamente del intervalo es $k_{\min} = 2$ y $k_{\max} = \lceil \sqrt{n} \rceil$.

Una desventaja que se presenta en esta codificación es que es sensible a la estructura interna del conjunto de datos, ya que solamente es capaz de detectar grupos en formas esféricas [137]. Otra desventaja de esta codificación es que puede generar grupos vacíos cuando se aplica el operador de cruce de un punto [59]. Además, al igual que la codificación basada en etiquetas también es una codificación de *una-a-muchas*, una misma solución puede ser codificada por varios cromosomas.

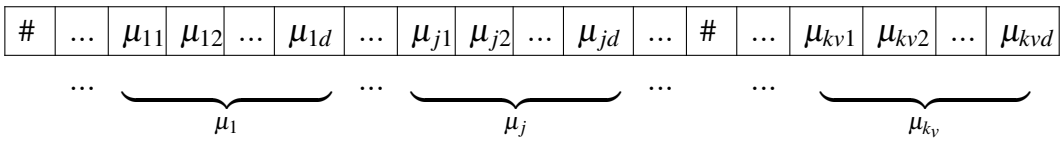
Es importante hacer notar que esta representación es la seleccionada por la mayoría de las propuestas de agrupamiento. Al ser escalable, se puede aplicar a un gran número de problemas en el mundo real.

- *Basada en centroides con símbolos de indiferencia y números decimales*

  La codificación basada en centroides con símbolos de indiferencia es otra variante basada en centroides para algoritmos que determinan automáticamente el valor de k_v, denotados por $\chi_{\mu\#}$.

  Esta codificación consiste en mezclar símbolos de indiferencia (#) entre los genes para representar agrupaciones de diferente número de grupos k_v. Cada gen puede ser un centroide μ_j, o puede no ser un centroide utilizando #. El tamaño es fijo para todos los cromosomas e igual a $k_{max} \times d$. El procedimiento para inicializar un cromosoma consiste en generar un número aleatorio $k_v \in [k_{\min}, k_{\max}]$ y se insertan genes μ_j y # en un orden al azar para completar el tamaño del cromosoma.

  Por ejemplo, la codificación para un individuo donde el número de grupo es k_v y el espacio d-dimensional podría ser la siguiente:



Un aspecto importante por considerar en la evolución de los individuos es tener controlados los límites de búsqueda para el número de grupos en el intervalo establecido $[k_{\min}, k_{\max}]$. La codificación también puede contribuir para que los individuos estén dentro de este límite, de lo contrario, es necesario implementar restricciones en los operadores genéticos para controlar que la codificación no salga del intervalo establecido y nos lleve a soluciones incorrectas y a un gasto más elevado de los recursos computacionales sobre todo cuando el $k_{\max}$ es superado.

En el caso de esta codificación, el límite inferior generalmente se establece a $k_{\min} = 2$, mientras el límite superior $k_{\max}$ debe ser especificado como un parámetro de entrada.

Si $k_v = k_{\max}$, el cromosoma no tiene genes #, por tanto, la misma codificación controla no sobrepasar el límite superior, pudiendo ser una ventaja. Mientras que si $k_v < 2$, el número de grupos codificado es pequeño y el número de genes con valor # es alto. Cuanto más próximo esté el número de # a $k_{\max}$, existe una alta probabilidad de obtener cromosomas nulos al aplicar el operador de cruce de un punto, pudiendo considerar este hecho una desventaja. De este modo, esta codificación puede tener una ventaja parcial en el control de los límites de búsqueda.

Desde el punto de vista de propiedades de esta codificación, es similar a la basada en centroides que se ha comentado anteriormente, por ser parte de esta categoría. Esta codificación es recomendada por [90], debido a que la longitud de los cromosomas es pequeña y, por tanto, la complejidad computacional que implica aplicar los operadores de cruce y mutación es más reducida.

- *Basada en centroides con valores de activación*

  La codificación basada en centroides con valores de activación ($\chi_{a\mu} = [\langle a_j \rangle | \langle \mu_i \rangle]$), es usada en algoritmos que buscan obtener automáticamente el valor k_v. Los cromosomas son de tamaño fijo ($k_{max} + d \times k_{max}$) constituido por dos regiones: un vector de valores de activación $\langle a_j \rangle$, donde $a_i \in [0,1]$ y la otra región es un vector con los centroides de los grupos $\langle \mu_i \rangle$, donde j: $j \in \{2, ..., k_{max}\}$ y k_{max} es el número máximo de grupos especificado como un parámetro de entrada. Si los valores $a_j > 0.5$ hacen que se activen los centroides μ_j y son utilizados para formar parte de la solución [106]. Una variante de

esta codificación usa un 0 o 1 para deshabilitar y habilitar respectivamente la región de activación de centroides [95].

Un ejemplo ilustrativo basado en la Figura 2.2 con 15 instancias ($n = 15$), en el espacio de 2-dimensiones ($d = 2$) y $k_{max} = 4$ (es el rango de búsqueda superior para k_v), podría ser el siguiente:

0.1	**0.8**	**0.6**	0.45	3.95	0.45	**3.76**	**3.60**	**0.63**	**2.03**	7.18	1.70
a_1	a_2	a_3	a_4	$\underbrace{\quad}_{\mu_1}$		$\underbrace{\quad}_{\mu_2}$		$\underbrace{\quad}_{\mu_3}$		$\underbrace{\quad}_{\mu_4}$	

El individuo codifica una solución con dos grupos ($k_v = 2$), al estar activo μ_2 y μ_3, por $a_2 > 0.5$ y $a_3 > 0.5$.

Si se utiliza la regla empírica para determinar el número de grupos ($k_{\max} = \lceil \sqrt{n} \rceil$), para conjuntos de datos con un valor grande de n y alta dimensionalidad, esta codificación tendría la desventaja de requerir más espacio para codificar un cromosoma de $O(\sqrt{n} + d\sqrt{n})$ en lugar del $O(dk)$ que requiere la codificación basada en centroides. Sucede lo mismo para determinar el número de grupos k_v que codifica el cromosoma que requiere un $O(\sqrt{n})$ en lugar del $O(1)$ que emplea la codificación basada en centroides.

- *Basada en la cuantificación vectorial*

  Esta codificación está basada en la técnica de *cuantificación vectorial*[2], la cual consiste en dividir el espacio de entrada en un número de regiones y cada una de ellas es caracterizada por un vector. Así, el conjunto de todos los vectores es mapeado a un conjunto más pequeño llamado *libro de códigos*[3], algunas de las aplicaciones de esta técnica se encuentra en la compresión de datos, clasificación de patrones y reconocimiento de objetos.

  Un libro de códigos puede ser usado para codificar un individuo al utilizar sus k vectores representativos junto con la partición total de vectores, representado por χ_{cb}. Con base en las codificaciones descritas anteriormente, un código de libros es equivalente a la combinación de codificaciones de basada en centroides, más una especificación explícita de pertenencia para el total de vectores, pudiendo ser la codificación basada en etiquetas. Así, la codificación para un individuo donde $\{\mu_j\}$ es el conjunto de centroides y $\{C_j\}$ es la partición de X asociada a los centroides está definida por (3.8).

[2]En inglés Vector Quantization
[3]En inglés codebook

$$\chi_{cb} = [\{\mu_j\}|\{C_j\}] = [\langle \mu_1, \mu_2, ..., \mu_k \rangle | \langle \xi_1, \xi_2, ..., \xi_k | \xi_1, \xi_2, ..., \xi_n \rangle] \tag{3.8}$$

Para facilitar las operaciones de agregar o eliminar instancias a un grupo como parte de los operadores genéticos, es conveniente utilizar un arreglo de pertenecía de índices de instancias, similar a listas enlazadas, el cual se detalla cómo opera en el siguiente ejemplo.

Por ejemplo, basado en la Figura 2.1a con 15 instancias ($n = 15$), en el espacio de 2-dimensiones ($d = 2$) y $k = 3$, el cromosoma χ_{cb} sería el siguiente:

0.63	2.03	7.18	1.7	3.81	2.7
μ_1		μ_2		μ_3	

9	13	15	−1	1	−1	−1	4	3	5	6	2	8	7	10	11	12	14
1	2	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

En el arreglo de pertenecía, la primera parte corresponde con los índices de las primeras instancias de los grupos C_j, la parte segunda (inicia en || del arreglo) está formada por genes ξ_j y ξ_i siendo los índices a las siguientes instancias de cada grupo y un valor centinela (-1) para indicar fin del grupo. Para obtener las instancias del grupo C_1, se obtiene el gen $\xi_1 = 9$ y con el índice se obtiene el siguiente elemento $\xi_9 = 2$, ahora $\xi_2 = 1$ y el $\xi_1 = -1$ indica que el grupo no tiene más instancias. Así, C_1 estaría formado por $C_1 = \{x_9, x_2, x_1\}$. Siguiendo el mismo procedimiento para C_2, $\xi_2 = 13$, $\xi_{13} = 11$, $\xi_{11} = 7$, $\xi_7 = 5$, $\xi_5 = 4$ y $\xi_4 = -1$, por tanto, $C_2 = \{x_{13}, x_{11}, x_7, x_5, x_4\}$. Los elementos de C_3, $\xi_3 = 15$, $\xi_{15} = 14$, $\xi_{14} = 12$, $\xi_{12} = 10$, $\xi_{10} = 8$, $\xi_8 = 6$, $\xi_6 = 3$ y $\xi_3 = -1$. Así, $C_3 = \{x_{15}, x_{14}, x_{12}, x_{10}, x_8, x_6, x_3\}$.

- *Basada en instancias representativas con números enteros*

  En esta codificación el cromosoma de un individuo está codificado por un vector de genes con valores enteros que representan los índices de las instancias del conjunto de datos que representa cada uno de los grupos considerados (k grupos). Así, cada gen representa la instancia más representativa de cada grupo, teniendo un valor que oscila desde 1 a n. Si el j-ésimo gen contiene el valor i corresponde con la instancia x_i y es la instancia considerada más representativa del grupo C_j, la cual es la instancia cuya suma de distancias entre ella y las otras del grupo es la mínima. La notación utilizada para representar está codificación de un individuo es con el cromosoma (3.9).

$$\chi_v = \langle v_j \rangle = \langle v_1, v_2, ..., v_k \rangle \tag{3.9}$$

Un ejemplo ilustrativo, para un individuo que codifica 3 grupos ($k = 3$) del conjunto de datos de la Tabla 2.1, es el siguiente:

13	9	15
v_1	v_2	v_3

Este individuo representa una solución donde la instancia x_{13} es la instancia representativa (v_1) del grupo C_1, x_9 es la instancia representativa (v_2) del grupo C_2 y x_{15} es instancia representativa (v_3) del grupo C_3. Estas instancias representativas se usan como referencia para asignar el resto de las instancias a cada grupo. Utilizando la ecuación (3.5) y la matriz $D_{15\times15}$ (2.8), se obtiene que las instancias $\{x_4, x_5, x_7, x_8, x_{11}, x_{13}\}$ pertenecen al grupo C_1, las instancias $\{x_1, x_2, x_6, x_9\}$ pertenecen al grupo C_2 y las instancias $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ son miembros del grupo C_3. De manera gráfica se muestra en la Figura 2.3.

La ventaja de esta codificación es que es adecuada cuando no es posible encontrar los centros de las agrupaciones, como el caso de agrupación de múltiples instancias [132]. Como se comentó en la sección 2.3.2, esta codificación es susceptible a un mejor manejo de valores atípicos a diferencia de utilizar los centroides de los grupos como referencia. En cuestión de desempeño, los algoritmos se pueden beneficiar al utilizar la matriz de disimilitud $D_{n\times n}$ evitando la evaluación exhaustiva de distancias en conjunto de datos de alta dimensionalidad. Por último, el tamaño de los cromosomas es independiente de los números de instancia.

Una desventaja que se encuentra es que dado que los genes de esta codificación no admiten elementos repetidos no es posible aplicar el operador de cruce tradicional y se requiere de operadores especializados. Por otro lado, debido a cuestiones de combinatoria, al realizar búsquedas para valores grandes de k, la escalabilidad se ve afectada en esta codificación junto con sus operadores.

- *Basada en instancias representativas con dígitos binarios*

  Esta codificación representa otra manera de codificar un cromosoma de agrupación con instancias representativas de cada grupo. En este caso, se usa un vector de dígitos binarios de longitud igual al número de instancias (n), denotado por (3.10).

$$\chi_\beta = \langle \beta_i \rangle = \langle \beta_1, \beta_2, ..., \beta_n \rangle, \text{ donde } \beta_i; \beta_i \in \{0,1\} \text{ y } i : i \in \{1,2,...,n\} \tag{3.10}$$

Si el gen β_i se establece con el dígito 1, la instancia x_i se selecciona como una instancia representativa.

Un ejemplo ilustrativo de un cromosoma χ_β (3.10) utilizando el conjunto de datos ilustrativo con $n = 15$ instancias y $k = 3$, equivalente al ejemplo mostrado anteriormente sería el siguiente:

0	0	0	0	0	0	0	0	1	0	0	0	1	0	1
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}

Si los grupos son numerados de izquierda a derecha, la solución indica las siguientes instancias representativas: $v_1 = x_9$, $v_2 = x_{13}$ y $v_3 = x_{15}$. Al incluir el resto de las instancias en los grupos considerados, se tendría: $\{x_1, x_2, x_6, x_9\}$, $\{x_4, x_5, x_7, x_8, x_{11}\}$ y $\{x_3, x_{10}, x_{12}, x_{14}, x_{15}\}$ para los grupos C_1, C_2 y C_3, respectivamente. A pesar de que los grupos están numerados en otro orden, la solución mostrada en la codificación anteriormente es equivalente a esta.

En cuanto a las ventajas y desventajas, es una codificación adecuada para aplicar operadores genéticos tradicionales siendo una ventaja y puede ser explotada para determinar el número de grupos automáticamente. La desventaja se presenta cuando se utiliza para un conjunto de datos de tamaño grande, siendo el tamaño de la codificación proporcional al tamaño de los datos. Así, para lograr una convergencia es necesario un número grande de generaciones y debe ser configurado para las diferentes ejecuciones de un AG.

- *Basada en grafos*

  Esta representación consiste en construir un grafo $G = (V, E)$ donde las instancias del conjunto de datos forman el conjunto de vértices V y el conjunto de aristas E representa la similitud entre las instancias. De los diferentes trabajos analizados, se encuentran dos codificaciones diferentes basadas en grafos: codificación basada en *bloques vecinos* y codificación *basada en locus*. De esta última codificación se han implementado dos variantes, una para números binarios y otra para números enteros.

  – *Bloques vecinos con dígitos binarios*

    En esta codificación un grafo es construido con el promedio de las distancias al vecino más cercano entre las instancias (d_{av}) y con el parámetro sugerido de $u \in [1.4, 1.8]$ que nos permite establecer qué instancias son adyacentes utilizando la ecuación (3.11).

$$A_{ij} = \begin{cases} 1 & \text{si } \|x_i - x_{i'}\| \leq u \cdot d_{av}, \\ 0, & \text{otro caso.} \end{cases} \tag{3.11}$$

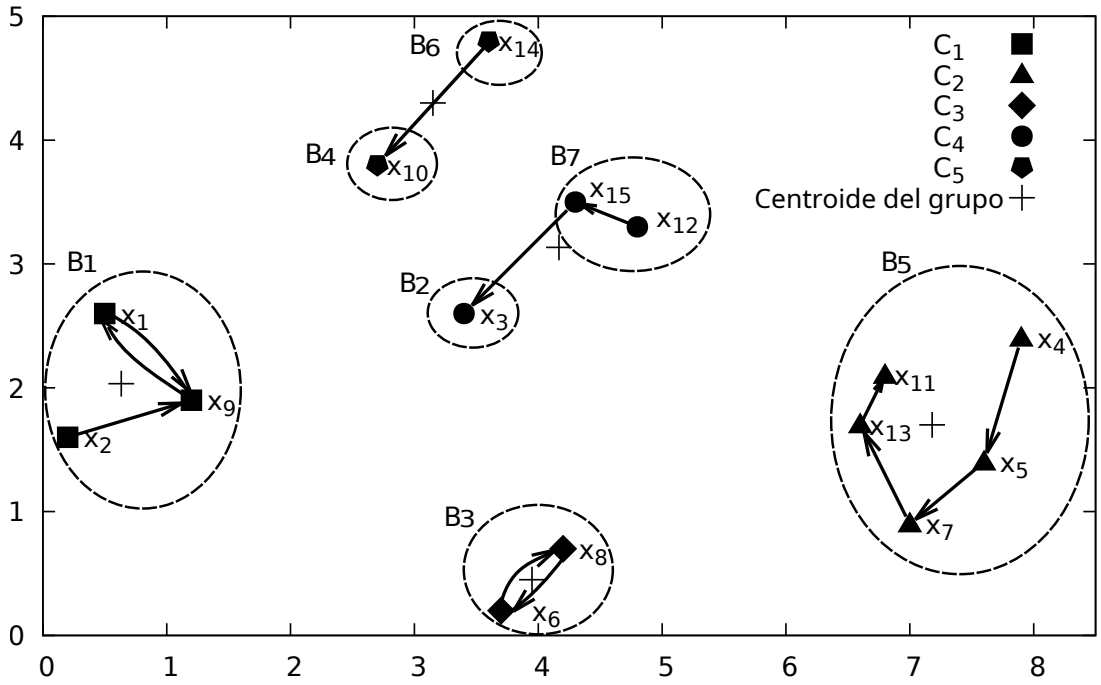


Figura 3.1 Representación gráfica de la codificación basada en grafo con bloques vecinos para $k_v = 5$

Con la matriz de adyacencia $A_{n\times n}$ se calculan los *componentes conexos* del grafo formándose m bloques $(B_1, B_2, ..., B_m)$ con sus m centroides $(\mu_{B_1}, \mu_{B_2}, ..., \mu_{B_m})$. Estos bloques se utilizan en una codificación binaria de longitud m, donde un gen con valor 1 en la posición i-ésima del vector significa que el bloque B_i es usado como semilla para formar un grupo agregando bloques asociados a genes con valor 0. Así, $B_{i'} \subset C_j$ si su centroide $\mu_{B_{i'}}$ es más cercano al centroide μ_j formado por el bloque con gen 1 utilización la ecuación (3.12) [121, 122].

$$\|\mu_{B_i} - \mu_j\| \leq \|\mu_{B_i} - \mu_{j'}\| \text{ para } j, j' : j, j' \in \{1, 2, ..., k\} \text{ y } j \neq j' \tag{3.12}$$

Por ejemplo, para el conjunto de datos de la Tabla 2.1 con $n = 15$ instancias, los pares de instancias vecinas más cercanas y sus distancias son los siguientes: $\|x_1 - x_9\| \approx 0.99$, $\|x_2 - x_9\| \approx 1.044$, $\|x_3 - x_{15}\| \approx 1.273$, $\|x_4 - x_5\| \approx 1.044$, $\|x_5 - x_7\| \approx 0.781$, $\|x_6 - x_8\| \approx 0.707$, $\|x_7 - x_5\| \approx 0.781$, $\|x_8 - x_6\| \approx 0.707$, $\|x_9 - x_1\| \approx 0.99$,

$$A_{15\times 15} = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{array} \begin{array}{c} \begin{array}{ccccccccccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{3.13}$$

$\|x_{10}-x_{14}\| \approx 1.345$, $\|x_{11}-x_{13}\| \approx 0.447$, $\|x_{12}-x_{15}\| \approx 0.539$, $\|x_{13}-x_{11}\| \approx 0.447$, $\|x_{14}-x_{10}\| \approx 1.345$ y $\|x_{15}-x_{12}\| \approx 0.539$. El promedio de las distancias de los pares de instancias vecinas es $d_{av} \approx 0.865$ y con $u = 1.4$, los pares que no forman una arista son (x_3,x_{15}), (x_{10},x_{14}) y (x_{14},x_{10}), por ser la distancia $\geq u \cdot d_{av} \approx 1.2114$. La matriz de adyacencia del grafo es mostrada en la matriz (3.13).

Con la matriz adyacencia como entrada ((3.13)), se obtienen siete componentes conexos (bloques $B_1, B_2, ..., B_7$): $\{x_1,x_2,x_9\}$, $\{x_3\}$, $\{x_6,x_8\}$, $\{x_{10}\}$, $\{x_4,x_5,x_7,x_{13},x_{11}\}$, $\{x_{14}\}$ y $\{x_{15},x_{12}\}$. De manera grafica se muestran en la Figura 3.1, los cromosomas de la población son de tamaños siete, y el cromosoma:

1	1	1	1	1	0	0
B_1	B_2	B_3	B_4	B_5	B_6	B_7

Codifica $k_v = 5$ grupos, a partir de los bloques $\{B_1\} \subset C_1$, $\{B_5\} \subset C_2$, $\{B_3\} \subset C_3$, $\{B_2,B_7\} \subset C_4$ y $\{B_4,B_6\} \subset C_5$, por medio de la ecuación (3.12).

– *Basado en locus de adyacencias con dígitos binarios*

En esta codificación, se siguen los siguientes pasos para construir el grafo. El primer paso, es calcular la matriz de disimilitud $D_{n\times n}$. El siguiente paso, es construir el *árbol de cobertura mínimo*[4] utilizando $D_{n\times n}$ como los pesos de las aristas. Con las aristas del árbol de cobertura mínimo que es un subconjunto del total de aristas,

[4]En inglés se le conoce como Minimum Spanning Tree (MST)

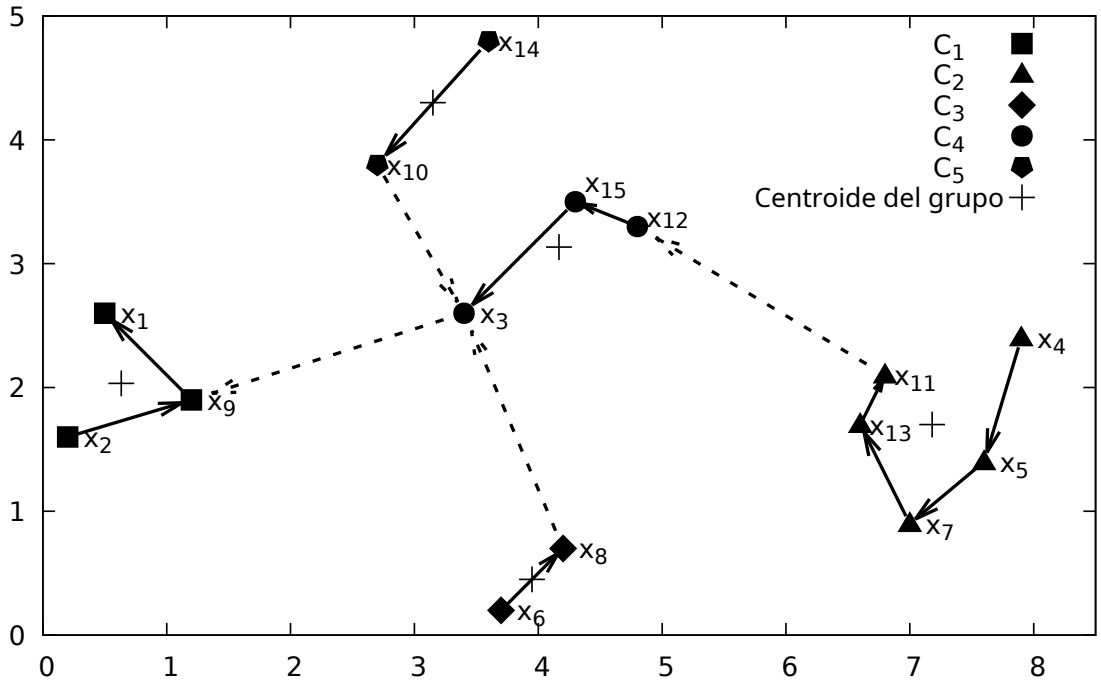


Figura 3.2 Árbol de cobertura mínimo del conjunto de datos de la Tabla 2.1, utilizado para codificación basada en locus de adyacencias con dígitos binarios

se establece una relación uno a uno con los genes de un cromosoma binario χ_b de tamaño $n-1$ igual al número de aristas. Finalmente, para obtener los grupos del conjunto de datos, el cromosoma es decodificado para obtener los componentes conexos asociados al grafo y cada uno de ellos es interpretado como un grupo. Utilizando la estructura de datos de conjuntos disjuntos se logra una complejidad del $O(n)$.

Para obtener los grupos que forman los individuos, las aristas del árbol de cobertura mínimo son mantenidas y eliminadas utilizando la codificación de cada solución que emplea un vector binario con una longitud igual al número de aristas del árbol. La interpretación de la codificación es que un gen con valor 0, significa que la arista se mantiene y un 1 significa que la arista es eliminada del grafo. El número de genes con valor 1 representa k_v+1 grupos [17].

Por ejemplo, para el conjunto de datos de la Tabla 2.1 con $n=15$ instancias, utilizando la matriz de disimilitud (2.8) como pesos del grafo y el árbol de cobertura

mínima que se muestra en la Figura 3.2 que está formado por 14 aristas:

$$E = \{\{x_9,x_1\},\{x_2,x_9\},\{x_3,x_9\},\{x_6,x_8\},\{x_8,x_3\},\{x_4,x_5\},\{x_5,x_7\},$$
$$\{x_7,x_{13}\},\{x_{13},x_{11}\},\{x_{11},x_{12}\},\{x_{12},x_{15}\},\{x_{15},x_3\},\{x_{14},x_{10}\},\{x_{10},x_3\}\}$$

Un cromosoma ejemplo de esta codificación podría ser el siguiente:

0	0	1	0	1	0	0	0	0	1	0	0	0	1
1	2	3	4	5	6	7	8	9	10	11	12	13	14

En la Figura 3.2, las aristas dibujadas con línea punteada son las que corresponden con los genes 1 y las de línea continua con los genes 0, correspondiendo con el cromosoma ejemplo. Para decodificar el cromosoma, los componentes conexos del grafo son: $\{x_1,x_2,x_9\}$, $\{x_4,x_5,x_7,x_{13},x_{11}\}$, $\{x_6,x_8\}$, $\{x_3,x_{12},x_{15}\}$ y $\{x_{10},x_{14}\}$ los cuales son los miembros de los grupos C_1, C_2, C_3, C_4 y C_5 respectivamente, representando una solución de $k_v = 5$ grupos.

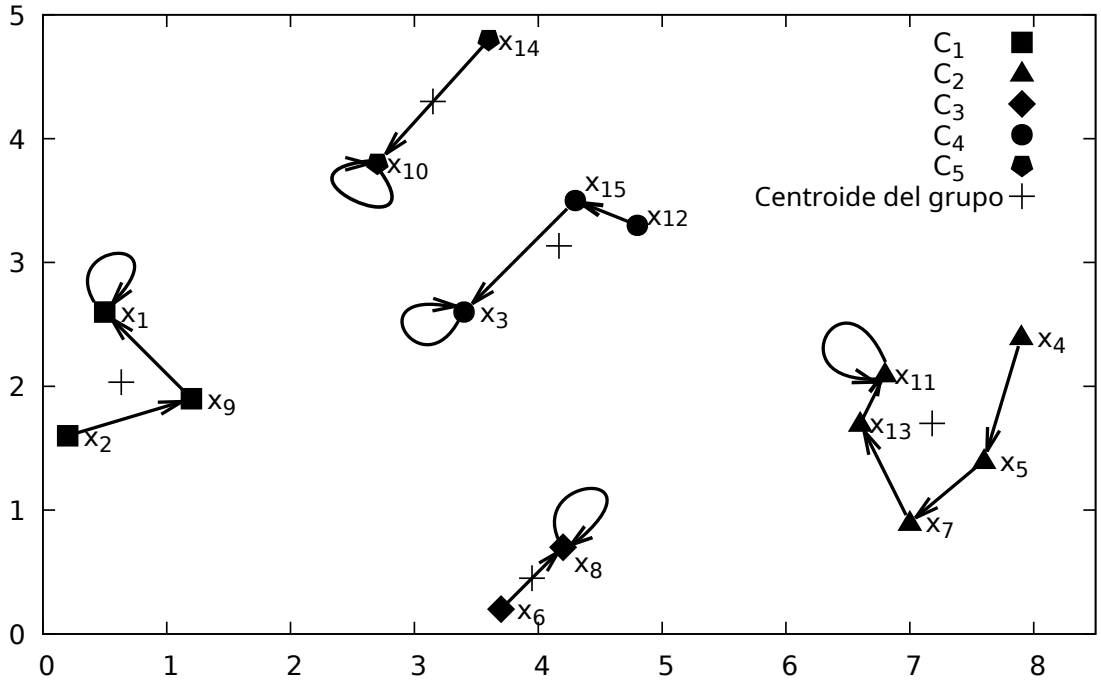


Figura 3.3 Grafo obtenido del árbol de cobertura mínima para codificación basada en locus de adyacencias con números enteros del conjunto de datos de la Tabla 2.1

– *Basado en locus de adyacencias con números enteros*

Esta codificación es una extensión de locus de adyacencias binaria descrita anteriormente. Se emplea en los multi-objective GAs más recientes [44]. Los cromosomas se codifican con índices enteros que representan el enlace entre las instancias. Cada cromosoma es un grafo que se obtiene de un árbol de cobertura mínimo al mantener y eliminar aristas, similar a la codificación anterior.

La longitud de cada cromosoma es igual al número de instancias n del conjunto de datos. Los valores de los genes son enteros en el rango de 1 a n. Así, si el i-ésimo gen contiene un valor entero i', entonces la arista del árbol de cobertura mínimo entre la instancia x_i y $x_{i'}$ ($i \to i'$) se mantiene. Cuando una arista es eliminada en el cromosoma, se indica reemplazando el enlace consigo mismo, es decir $i \to i$.

Por ejemplo, el cromosoma de un individuo de esta codificación sería el siguiente:

1	9	3	5	7	8	13	8	1	10	11	15	11	10	3
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Utilizado el árbol de cobertura mínimo del ejemplo que se puede ver en la Figura 3.3, el cromosoma indica que las aristas $(3,9)$, $(8,3)$, $(10,3)$ y $(11,12)$ mostradas en la Figura 3.3 con línea punteada, son eliminadas. Así, en el cromosoma los valores en los índices 3, 8, 10 y 11 se puede ver que son iguales a su propio índice. Para decodificar el cromosoma, se tiene que obtener los componentes conexos que son equivalentes a los del ejemplo anterior: $\{x_1,x_2,x_9\}$, $\{x_4,x_5,x_7,x_{13},x_{11}\}$, $\{x_6,x_8\}$, $\{x_3,x_{12},x_{15}\}$ y $\{x_{10},x_{14}\}$ los cuales son los miembros de los grupos C_1, C_2, C_3, C_4 y C_5 al obtener $k_v = 5$ grupos.

Las ventajas de esta codificación es la capacidad para codificar particiones de diferente número de grupos k_v sin necesidad de establecer el intervalo de búsqueda $[k_{\min}, k_{\max}]$. Además, permite encontrar formas no convexas e irregulares.

La desventaja de la codificación basada en grafos es la escalabilidad, ya que inicialmente para obtener el grafo asociado al conjunto de datos representado por una matriz de adyacencia, tanto en espacio y tiempo tiene una complejidad computacional de $O(n^2)$. Obtener el árbol de cobertura mínimo con Kruskal's, o Prim's es $O(E \lg V)$ para un grafo disperso de lo contrario es $O(V^2)$ que es el caso en esta codificación donde el número de vértices V es igual a n.

Otra desventaja es cuando los conjuntos de datos presentan superposición de grupos, siendo difícil dividir a estos al utilizar esta codificación.

3.1.2 Función de aptitud

Cada individuo de la población $\mathscr{P}(t)$, representado por el cromosoma χ_l, es evaluado en los pasos 4 y 9 del Algoritmo (4) por una función de aptitud que le asigna un valor de supervivencia que determina su bondad para agrupar las instancias de un conjunto de datos. Este valor se denomina valor de aptitud. La selección adecuada de una función de aptitud $\mathscr{F}(\chi_l)$ es otro aspecto importante en el diseño de un AG [90].

Para el problema de agrupamiento, se utilizan los CVI, que como ya se comentó en el capítulo anterior, son los índices de validación de agrupamiento. Este valor guiará al AG para revelar las estructuras de grupo y determinar la calidad de la partición realizada para un conjunto de datos determinado. Los CVIs se dividen en dos tipos de medidas de validación: índices *internos* y *externos*.

Los índices internos evalúan la bondad de la estructura de agrupamiento sin tener información de la distribución real de los grupos. Se basan en cómo las instancias se distribuyen dentro de un grupo y con otros grupos. Para su cálculo se utilizan los atributos que describen los grupos, a menudo los criterios más aceptados son dos: *separación* y *compactabilidad* de los grupos [50]. Los índices externos utilizan información catalogada por un experto, no presente en los atributos del conjunto de datos, tales como la *clase* a la que pertenece cada instancia. Estos índices verifican la calidad de la agrupación comparándola con la partición utilizando una categorización ya establecida de las instancias [49, 15]. El análisis de grupos, generalmente referido como una técnica de exploración no supervisada, está más asociado con índices internos, aunque algunos AGs proponen usar una función de aptitud basada en índices externos [3].

Mientras que en el caso de AGs con *k-fija* es habitual medir la compactabilidad de los grupos con la *suma de la distancia Euclidiana* (SED) o bien, con alguna variante. Para AGs con *k-variable* muchos CVIs se han desarrollado desde décadas pasadas hasta el presente. El índice específico de cada AG se detalla en la sección 3.2. Además, una descripción de todos los CVIs que se consideran en este trabajo se detallan en la sección 4.1.2 donde en la tabla 4.2 se muestran una especificación de cada CVI.

3.1.3 Operador de selección de padres

En cada generación (ciclo del Algoritmo 4), de la población actual $\mathscr{P}(t)$ se seleccionan los padres que participarán en el proceso de recombinación y mutación. El proceso de selección juega un papel importante para enfocar el esfuerzo de búsqueda en regiones prometedoras en el espacio de soluciones, así como controlar la velocidad de convergencia a una solución [55]. Los

individuos seleccionados pasan a la población $\mathscr{P}'(t)$, donde participarán en las operaciones genéticas posteriores como cruce y mutación.

Para el proceso de selección existen diferentes métodos [29, 46, 88]. La mayoría de los operadores de selección propuestos tiene un componente aleatorio que está condicionado por el valor de aptitud de cada individuo. La diferencia básica entre ellos está determinada por la necesidad de ajustar la velocidad de convergencia, así como la capacidad de eliminar cromosomas con soluciones no válidas.

En el caso de los AGs para agrupación, en el proceso de selección es común encontrar individuos que codifican soluciones no válidas, debido a los resultados en la aplicación de los operadores en la generación anterior. De este modo, la estrategia de cómo tratar las soluciones no válidas en el caso del problema de agrupación es un punto importante por considerar en el diseño de las diferentes componentes del algoritmo.

Los métodos de selección usados en los diferentes algoritmos analizados en este trabajo se describen a continuación.

- *Selección por torneo*

  Consiste en seleccionar un conjunto de individuos al azar, entre ellos compiten obteniendo el mejor individuo para ser colocado en $\mathscr{P}'(t)$. El proceso se repite hasta que la $\mathscr{P}'(t)$ se complete. La selección del torneo puede implementarse con y sin reemplazo de los individuos que compiten. El tamaño del torneo puede variar, por lo general, se utilizan dos individuos conocidos como torneo binario. Sin embargo, se podrían utilizar torneos con tres o más individuos.

- *Seleccion por ruleta*

  Su diseño se asemeja a la ruleta de un casino. Se crea una rueda con tantas casillas como individuos tiene la población. Cada casilla $\imath$ es proporcional a la aptitud del $\imath$-ésimo individuo de la población. Se elige un punto fijo en la circunferencia de la rueda. La selección de un individuo se hace girando la ruleta y observando la posición del punto fijo con la casilla cuando la ruleta se detiene.

  Siguiendo la descripción anterior, se obtiene la función de distribución acumulada $F_{\mathscr{F}_\chi}(\chi_\imath)$ de la aptitud de los individuos de la población por la ecuación (3.14).

$$F_{\mathscr{F}_\chi}(\chi_\imath) = \frac{\sum_{\imath=1}^{\imath} \mathscr{F}(\chi_\imath)}{\sum_{\imath=1}^{n_P} \mathscr{F}(\chi_\imath)} \tag{3.14}$$

  Seleccionar los cromosomas necesarios utilizando el Algoritmo `ruleta` (5) y colocarlos en $\mathscr{P}'(t)$.

Algoritmo 5 Selección por `ruleta`

Entrada: $F_{\mathscr{F}_\chi}$: distribución acumulada -
Salida: χ_ι: Un individuo
1: $r \leftarrow$ número aleatorio $\in [0,1]$
2: $\iota \leftarrow 1$
3: **while** $F_{\mathscr{F}_\chi}(\chi_\iota) <$ r **do**
4: $\quad \iota \leftarrow \iota + 1$
5: **end while**
6: **return** χ_ι

- *Selección por elitismo*

  Los individuos se clasifican por valor de aptitud y se eligen los mejores para reproducirse, los restantes son descartados. Los individuos son ordenados por su valor de aptitud en orden creciente o decreciente si el objetivo es minimizar o maximizar.

Aparte de utilizar un mecanismo de selección de los mencionados anteriormente, generalmente algunos AGs implementan una estrategia en la evolución, las encontradas en la revisión de las propuestas son las siguientes:

- *Selección por rangos*

  Para evitar problemas de convergencia prematura, un procedimiento que se utiliza es normalizar de manera lineal la función de aptitud de los individuos haciendo un balanceo en la selección y evitando una presión en las diferentes generaciones de la evolución.

- *Selección en dos etapas*

  La selección en dos etapas [55] es usada en algoritmos de k-variables, la cual se realiza de acuerdo con la convergencia de la población a un k_v por medio de analizar la consistencia de los individuos con el número de grupos. La variable utilizada es k_{con}, obtenida con la ecuación (3.15).

$$k_{\text{con}} = n_{sk}/n_P, \tag{3.15}$$

  donde n_{sk} es el número de máximo de individuos con el mismo número de grupos en la población actual y $0 < k_{\text{con}} < 1$.

  La variable k_{con} es utilizada para la selección de los individuos y así converger a una k_v óptima con la ecuación (3.16).

$$F_{\mathscr{F}_\chi}(\chi_\iota) = \begin{cases} \frac{\sum_{\iota=1}^{l} \mathscr{F}^\lambda(\chi_\iota)}{\sum_{\iota=1}^{nP} \mathscr{F}^\lambda(\chi_\iota)} & \text{si } 0 < k_{\text{con}} < 1, \text{ primera etapa} \\ \frac{\sum_{\iota=1}^{l} \mathscr{F}(\chi_\iota)}{\sum_{\iota=1}^{nP} \mathscr{F}(\chi_\iota)} & k_{\text{con}} = 1, \text{ segunda etapa} \end{cases} \tag{3.16}$$

Donde λ es un factor de selección dinámico $\lambda = e^{-k_{\text{con}}}$, y $e^{-1} < \lambda < 1$.

En la primera etapa, cuando $0 < k_{\text{con}} < 1$ la búsqueda es enfocada principalmente en encontrar el número óptimo de k_v, individuos con una función de aptitud grande tienen una alta probabilidad de selección. En la segunda con $k_{\text{con}} = 1$, todos los individuos tienen el mismo número de grupos, y la búsqueda se centra en encontrar una partición óptima de las x_i.

3.1.4 Operadores genéticos de reproducción

En cada generación (paso 8 del Algoritmo 4), los individuos seleccionados de la población $\mathscr{P}'(t)$ se modifican para explotar y explorar a través del espacio de búsqueda. Los operadores más comunes utilizados son el *cruce* y la *mutación*.

Los operadores genéticos están relacionados con la codificación de los individuos (descrita en la sección 3.1.1), lo cual debe ser considerada en la selección de los operadores para el diseño de un nuevo AG.

En los AGs para agrupación se han propuesto diferentes operadores genéticos desde tradicionales hasta especializados. En los siguientes apartados se realizará una descripción y clasificación de los que se han utilizado hasta la fecha.

Operador de cruce

El operador de cruce consiste en la transmisión de genes entre dos individuos (padres) para obtener nuevos individuos (hijos). Al fusionarse los genes, los hijos heredan características de los padres, ocurriendo una diversificación y adaptación. Así, los nuevos individuos son potencialmente prometedores y pueden ser mejores que sus padres.

Existen varios operadores genéricos de cruce que han sido utilizado en los AGs de agrupación tales como cruce en un punto, heurístico, basado en un camino y otros operadores específicos que se han diseñado exclusivamente para resolver este problema.

Una propiedad de gran utilidad para caracterizar a los operadores de cruce es *insensibilidad al contexto* [36, 59], establecida de la siguiente manera "los esquemas definidos en los genes

de un solo cromosoma no transmiten información útil que podría ser explotada por el proceso de muestreo implícito llevado a cabo por un AG de agrupación".

En la clasificación propuesta en este trabajo, los operadores de cruce serán clasificados en *convencionales* identificando a los operadores que se pueden aplicar independientemente del problema y *especializados* que se diseñan considerando las particularidades del problema de agrupación. A continuación, se describen los diferentes operadores de cruce que son utilizados en los AG de agrupamiento.

Operadores de cruce convencionales

Los operadores de cruce convencionales intercambian genes independientemente del contexto y restricciones de la agrupación de instancias de un conjunto de datos. Se pueden utilizar con independencia del tipo de dato usado en la codificación. En ocasiones estos operadores producen individuos no válidos para el problema de agrupamiento que deben tenerse en cuenta durante la búsqueda evolutiva, para ello se pueden considerar diferentes estrategias como descartar o sustituir por otros individuos válidos. Sin embargo, recursos computacionales adicionales son necesarios para encontrar y corregir a los individuos no válidos.

- *Cruce en un punto*

  El operador de cruce en un punto es un operador genético clásico, usado en los primeros AG [46] para resolver diferentes problemas de optimización. Consiste en seleccionar aleatoriamente un gen dentro de un par de cromosomas padres, que sirve como punto de cruce para intercambiar genes entre los padres, y así obtener dos hijos.

  Como ejemplo ilustrativo, en la Figura 3.4a se muestra cómo se aplica el operador para cromosomas padres que emplean una codificación basada en etiquetas con números enteros χ_{α_1} y χ_{α_2} (3.1). Como resultado se obtienen dos hijos χ'_{α_1} y χ'_{α_2}. El operador también puede ser aplicado para diferentes codificaciones.

  Por ser un operador de referencia en los AGs, sus propiedades han sido ampliamente estudiadas para el caso del problema de agrupación, y es catalogado como un operador *insensibilidad al contexto* [36, 59], debido a que en algunas situaciones los cromosomas no transmiten material genético útil para ser explotado por el AG de agrupación.

  Un ejemplo ilustrativo donde el operador obtiene una solución no válida se muestra en la Figura 3.4b. Los cromosomas padres χ_{α_3} y χ_{α_4} que utilizan una codificación basada en etiquetas con números enteros codifican soluciones de k-fija con 4 grupos. Al aplicar el operador, el cromosoma hijo χ'_{α_3} obtenido codifica 3 grupos en lugar de 4. Esta situación de insensibilidad al contexto por el operador es también descrita en [59, 90].

No obstante, esta dificultad se puede superar por el hecho de que las funciones de aptitud para algoritmos de k-fija son funciones monótonas crecientes con respecto a k. De este modo, estas soluciones tienen una menor probabilidad de continuar en las próximas generaciones.

A continuación, se muestran otros ejemplos donde también se obtienen soluciones inválidas, tanto para k-fija como k-variable, al aplicar el operador de cruce en un punto para codificación basada en centroides. En la Figura 3.5a, se puede ver que partiendo de los cromosoma padres χ_{x_1} y χ_{x_2}, al aplicar el operador de cruce en un punto, se obtienen los cromosomas hijos χ'_{x_1} y χ'_{x_2} donde χ'_{x_2} tiene dos centroides repetidos. Al repartir las instancias con la ecuación (3.5) se tendrá que el grupo C_4 está vacío. En Figura 3.5b, se puede ver que partiendo de los cromosomas padres χ_{x_3} y χ_{x_4}, al aplicar el operador de cruce en un punto, se obtienen los cromosomas hijos χ'_{x_3} y χ'_{x_4}. A diferencia del caso anterior, se obtienen centroides diferentes y el centroide μ_2 no logra obtener instancias cercanas quedando el centroide C_2 vacío.

El problema de insensibilidad al contexto encontrado en el operador de cruce en un punto, también se encuentra en otros operadores. En estos casos es importante la estrategia que se sigue para manejar estas soluciones incorrectas. Tras varios estudios en este tema, se determina que para los algoritmos de k-variable es más conveniente no forzar la eliminación de cromosoma inválidos y que sea por medio de la función de aptitud la que evite que se seleccionen estos individuos para la siguiente generación. Otra estrategia que se ha utilizado eficazmente es replantear los grupos vacíos.

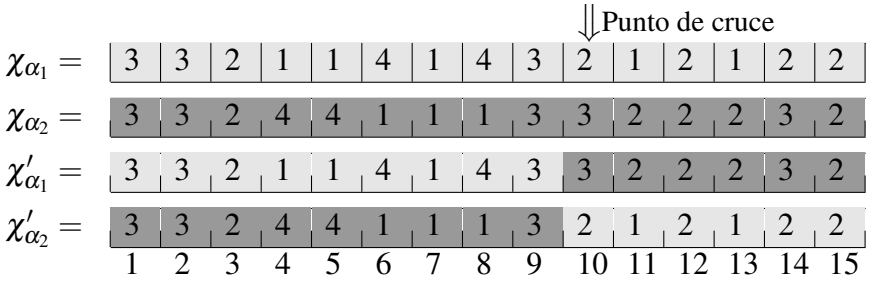
- *Cruce en dos puntos*

  El operador de cruce en dos puntos es una extensión de cruce en un punto. Este operador selecciona aleatoriamente dos puntos de cruce dentro de los cromosomas y luego los dos cromosomas padres intercambian genes a partir de estos puntos para generar los dos nuevos cromosomas hijos. Al igual que el operador de cruce en un punto se puede utilizar para cualquier codificación.

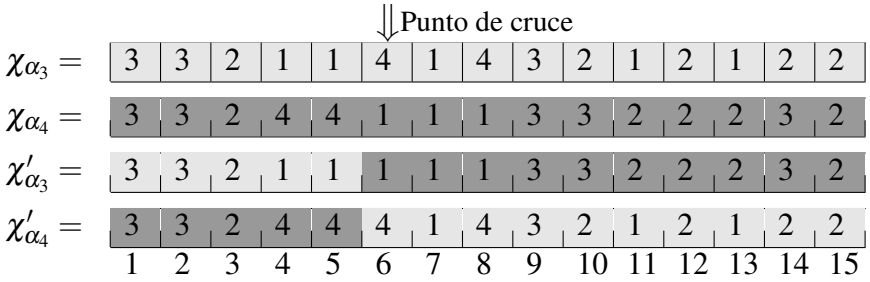
  Una ventaja con respecto al operador de cruce en un punto es que no existe un orden en los grupos codificados para lograr una convergencia. Este operador puede tener un mejor desempeño al combinar cualquier parte del cromosoma, pero también tiene la desventaja de que genera individuos inválidos.

- *Cruce uniforme*

  Se utiliza para generar dos hijos a partir de parejas de padres. Los individuos hijos se crean primero como copias idénticas de los padres. Para el intercambio de genes se utiliza una



(a) Los cromosomas padres χ_{α_1} y χ_{α_2} intercambian genes a partir del gen 10 y como resultado se obtienen los hijos χ'_{α_1} y χ'_{α_2}



(b) El operador obtiene una solución no válida para una agrupación k-fija, al obtener un cromosoma hijo χ'_{α_3} con el grupo C_4 vacío

Figura 3.4 Ejemplos de la aplicación del operador cruce en un punto en cromosomas con codificación basada en etiquetas con números enteros.

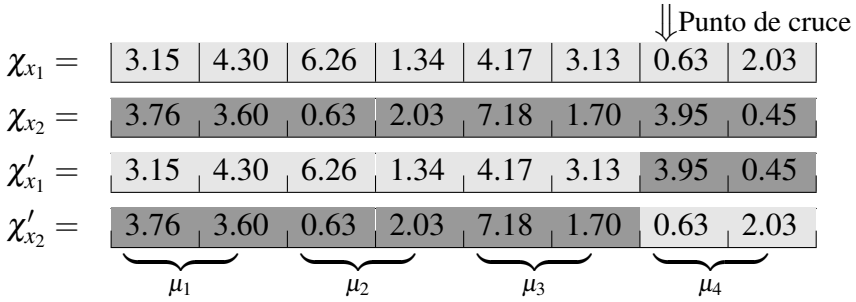
función de probabilidad uniforme en el intervalo $[0,1]$. Cada gen puede ser preservado o intercambiado entre los dos padres de acuerdo con la probabilidad seleccionada. Por lo tanto, este operador tiene la capacidad de explorar cualquier combinación posible del material genético de los padres dados.

Se puede utilizar para cualquier codificación. Sin embargo, se debe ser cauteloso acerca de la generación de individuos inválidos.

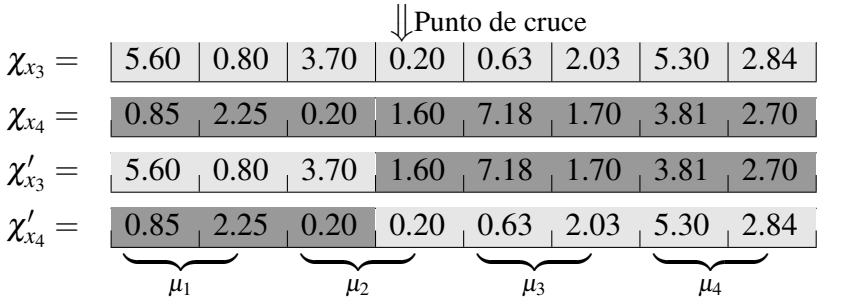
- *Cruce heurístico*

  El operador de cruce heurístico es utilizado como operador de cruce en diferentes AGs para diferentes propósitos. Se distingue por usar la función objetivo para determinar la dirección de búsqueda. La salida del operador es un individuo hijo y en algunos casos no puede obtener ningún individuo [88].

  En el caso de los AGs de agrupación es usado para cromosomas con codificación basada en centroides χ_x (3.6) y χ_μ (3.7). Aunque el operador original produce un solo hijo



(a) Solución no válida, al obtener centroides repetidos μ_2 y μ_4 en el cromosoma hijo χ'_{μ_2}



(b) Solución inválida, debido a que en la asignación de las instancias el centroide μ_2 del cromosoma χ'_{x_4} no atrapa ninguna instancia

Figura 3.5 Insensibilidad al contexto del operador *cruce en un punto* [36, 59], para cromosomas con una codificación basada en centroides

(3.17), los algoritmos lo adaptan para generar dos hijos al complementar la ecuación con (3.18) [18].

Para dos individuos padres χ_{x_1} y χ_{x_2}, el operador de cruce heurístico obtiene dos hijos por medio de las siguientes ecuaciones:

$$\chi'_{x_1} = \chi_{x_1} + r(\chi_{x_1} - \chi_{x_2}) \tag{3.17}$$

$$\chi'_{x_2} = \chi_{x_1} \tag{3.18}$$

Donde $r = U(0,1)$, $U(0,1)$ es una distribución uniforme en el intervalo $[0,1]$ y χ_{x_1} es mejor que χ_{x_2} en términos de la función de objetivo.

Operadores de cruce especializados

Los operadores de cruce especializados están especialmente diseñados para el problema de agrupación. Estos operadores también son llamados orientados a agrupamiento [36]. Estos operadores intercambian genes considerando las diferentes restricciones y condiciones impues-

tas por el problema. Así, los grupos de instancias codificados por cada individuo al aplicar el operador pueden ser copiados, fusionados o eliminados. Estos operadores evitan crear individuos inválidos y también logran una convergencia más acelerada a una solución por parte del proceso evolutivo.

- *Cruce basado en un camino*

  El operador de cruce basado en un camino[5] [18], se utiliza para generar dos cromosomas hijos a partir de una pareja de padres χ_{x_1} y χ_{x_2} (3.6). Es aplicado a cromosomas con codificación basada en centroides con genes de coordenadas, pero puede ser aplicado a otras codificaciones donde los genes tengan relación de orden ($\leq$). El operador recibe el nombre de *camino* porque se construyen diferentes individuos al intercambiar genes partiendo de χ_{x_1} hasta llegar de nuevo a un individuo equivalente a χ_{x_2} por medio de ir iterando individuos con los operadores $* \uparrow *$ y $* \downarrow *$.

  El procedimiento para aplicar el operador es el siguiente:

  (a) Primero, se calcula $\overline{\chi_{x_1}\chi_{x_2}}$ con los genes máximo con:

  $$\overline{\chi_{x_1}\chi_{x_2}}(jd) = \max\{\chi_{x_1}(jd), \chi_{x_2}(jd)\}$$

  (b) Segundo, se inicia aplicando a $\chi_{x_{11}} = \chi_{x_1}$ para obtener el siguiente individuo $\chi_{x_{12}} = \chi_{x_{11}} \uparrow \overline{\chi_{x_1}\chi_{x_2}}$. Se continua de manera iterativa obteniendo los siguientes individuos $\chi_{x_{13}} = \chi_{x_{12}} \uparrow \overline{\chi_{x_1}\chi_{x_2}}$, ..., hasta que $\chi_{x_{1p}} = \chi_{x_{1p-1}} \uparrow \overline{\chi_{x_1}\chi_{x_2}} = \overline{\chi_{x_1}\chi_{x_2}} = \chi_{x_{21}}$.

  (c) Tercero, igual que en el paso anterior, pero con $\chi_{x_{21}}$. Se obtiene $\chi_{x_{22}} = \chi_{x_{21}} \downarrow \chi_{x_2}$, $\chi_{x_{23}} = \chi_{x_{22}} \downarrow \chi_{x_2}$, ..., hasta que $\chi_{x_{2p'}} = \chi_{x_{2p'-1}} \downarrow \chi_{x_2} = \chi_{x_2}$.

  (d) Por último, con los individuos obtenidos $\{\chi_{x_{12}}, \chi_{x_{13}}, ..., \chi_{x_{1p}}, \chi_{x_{21}}, \chi_{x_{22},}, \chi_{x_{2p'}}\}$, dos individuos hijos son seleccionados. El método de selección puede ser aleatorio, ruleta o torneo u otro, los autores de este operador aconsejan seleccionar el mejor individuo y otro de manera aleatoria.

  La ventaja de este operador es con conjuntos de datos con un alto valor de d-dimensiones. Al hacer una exploración más extensa en el espacio de soluciones, la posibilidad de mayor intercambio de genes incrementaría. La desventaja estaría en el costo computacional al construir el camino de individuos y evaluar la función aptitud de cada individuo. Otra desventaja del cruce basado en un camino se presenta cuando los individuos padres son muy similares, por lo que no obtendrá un camino de individuos y el operador no podrá obtener los individuos hijos.

[5] En inglés los autores lo llaman como path-based crossover

Algoritmo 6 Operador de cruce `D_MX` [81]

Entrada: $\chi_{v1} = \langle v_1^1, v_2^1, ..., v_k^1 \rangle$, $\chi_{v2} = \langle v_1^2, v_2^2, ..., v_k^2 \rangle$: dos cromosomas padres (3.9)
Salida: χ'_{v1}, χ'_{v2}: cromosomas hijos

1: Combina a χ_{v1} y χ_{v2}
   $Q \leftarrow \langle v_1^1, v_2^1, ..., v_k^1, v_1^2, v_2^2, ..., v_k^2 \rangle$ concatena una copia de χ_{v1} y de χ_{v2}
   Mezcla aleatoriamente los genes (una permutación)
   $Q \leftarrow \langle v'_1, v'_2, ..., v'_{2k} \rangle$
2: Agrega nuevos genes $\{v_{nj}\}$ con una probabilidad $p_{m,mix}$ al remplazar genes de $\langle Q \rangle$
   $Q \leftarrow \langle v''_1, v''_2, ..., v''_{2k} \rangle$
3: Nuevamente mezcla aleatoriamente los genes de Q
   $Q \leftarrow \langle v'''_1, v'''_2, ..., v'''_{2k} \rangle$
4: Construye χ'_{v1} copiando k genes de Q seleccionados de izquierda a derecha con la restricción que no tenga genes repetidos
   $\chi'_{v1} \leftarrow \langle v'''_1, v'''_2, ... \rangle$
5: Construye χ'_{v2} copiando k genes de Q seleccionados de derecha a izquierda, igualmente con la restricción de no tener genes repetidos
   $\chi'_{v2} \leftarrow \langle v'''_{2k}, v'''_{2k-1}, ... \rangle$
6: **return** (χ'_{v1}, χ'_{v2})

- *Cruce por combinación y mezcla de grupos*

  El operador de cruce por combinación y mezcla de grupos (D_MX)[6] [81] es aplicado a individuos con codificación basada en instancias representativas con números enteros χ_v (3.9). Se utiliza para generar dos hijos a partir de una pareja de padres. D_MX considera la restricción impuesta en los cromosomas χ_v donde una instancia representativa v_j solamente puede ocurrir una vez en el vector de genes. El procedimiento para aplicar el operador D_MX se describe en el Algoritmo (6).

- *Cruce por pares de vecinos más cercanos*

  El operador `Cruce-solución` [41], por sus características es llamado en este trabajo *cruce por pares de vecinos más cercanos*. El componente principal del operador es el algoritmo PNN [7] [31] que está definido para cromosomas de tipo libro de códigos χ_{cb} (3.8), una variante de los individuos con codificación χ_μ (3.7).

  Este operador está especificado en el Algoritmo (7). Tal y como se muestra, inicia haciendo una nueva partición de tamaño $2 \cdot k$ al juntar los centroides $\{\mu_j^1\}$ y $\{\mu_j^2\}$ de los individuos padres en $\{\mu'_j\}$ utilizando `combinar-centroides`. Las particiones $\{C_j\}^1$ y $\{C_j\}^2$ se combinan en $\{C'_j\}$ utilizando el procedimiento `combinar-particiones`.

[6] En inglés se le conoce mix subset recombination
[7] Del inglés pairwise nearest neighbor

Además, ajusta la pertenencia de las instancias con la distancia más cercana a su centroide $\{\mu'_j\}$. Después, los centroides de los grupos son actualizados y eliminados los grupos vacíos utilizando los procedimientos `actualizar-centroides` y `borrar-grupos-vacíos`, respectivamente. Finalmente, el operador de cruce reduce al individuo hijo de una partición de $2 \cdot k$ a una k grupos (línea 5 del algoritmo PNN (8)). En el primer paso de PNN, se encuentran los grupos C_j y C'_j más cercanos con la función `encontrar-vecinos-cercanos` que minimiza la distancia dada por la ecuación (3.19).

$$d_{\mu_j,\mu'_j} = \frac{|C_j||C'_j|}{|C_j|+|C'_j|} \cdot \|\mu_j - \mu'_j\| \tag{3.19}$$

Los pares de índices son almacenados en $Q = q_1, q_2, ..., q_k$, y se selecciona q_i al minimizar la medida dada en la ecuación (3.19). Después, los grupos se unen utilizando la función `unir-grupos` y se actualizan los índices con `actualizar-punteros`. El algoritmo PNN continua hasta que el libro de códigos que codifica el individuo sea reducido a un número de grupos igual a k.

- *Cruce por combinación de grupos*

  El operador de cruce por combinación de grupos fue diseñado y utilizado por [36, 60] para individuos codificados por cromosomas $\chi_{\alpha k}$ (3.3). No obstante, también puede ser utilizado por codificaciones explícitas basadas en etiquetas. Este operador tiene como objetivo tanto converger al número óptimo de grupos como a la identificación de los elementos que componen cada grupo. El procedimiento para aplicar el cruce por combinación de grupos es el siguiente [60]:

  (a) Dos individuos padres son seleccionados ($\chi^1_{\alpha k}$ y $\chi^2_{\alpha k}$).

  (b) Considerando que el individuo padre $\chi^1_{\alpha k}$ codifica k^1_v grupos, un número aleatorio k'_v grupos es seleccionado en el intervalo $1 \leq k'_v \leq k^1_v$,y serán los que se copien en $\chi^2_{\alpha k}$.

  (c) Los grupos que no cambian de $\chi^2_{\alpha k}$ son conservados y los que cambian los objetos son asignados al grupo que tiene el centroide más cercano. De esta manera, se obtiene el individuo hijo $\chi'_{\alpha k}$.

  (d) El mismo procedimiento es empleado para obtener el individuo hijo $\chi''_{\alpha k}$, pero a la inversa. En este caso, se considera que los grupos que cambian de $\chi^2_{\alpha k}$ son copiados en $\chi^1_{\alpha k}$.

- *Cruce de dos puntos para grupos*

Algoritmo 7 `Cruce por pares de vecinos más cercanos` [41]

Entrada: $\chi_{cb1} = [\{\mu_j^1\}|\{C_j^1\}]$, $\chi_{cb2} = [\{\mu_j^2|\{C_j^2\}]$: dos cromosomas padre basados en libro de códigos (3.8)
Salida: $\chi'_{cb} = [\{\mu'_j\}|\{C'_j\}]$: un cromosoma hijo libro de códigos (3.8)

1: $\{\mu'_j\} \leftarrow$ `combinar-centroides`$(\{\mu_j^1\},\{\mu_j^2\})$
2: $\{C'_j\} \leftarrow$ `combinar-particiones`$(\{\mu'_j\},\{C_j^1\},\{C_j^2\})$
3: $\{\mu'_j\} \leftarrow$ `actualizar-centroides`$(\{\mu'_j\},\{C'_j\})$
4: `borrar-grupos-vacios`$(\{\mu'_j\},\{C'_j\})$
5: PNN$(\{\mu'_j\},\{C'_j\})$
6: **return** $\chi'_{cb} = [\{\mu'_j\}|\{C'_j\}]$
▷ //Sub-procedimiento juntar los centroides
7: **procedure** `combinar-centroides`$(\{\mu_j^1\},\{\mu_j^2\}')$
8: **return** $\{\mu_j^1\} \cup \{\mu_j^2\}'$
▷ //Sub-procedimiento crear una nueva partición $\{C'_j\}$
9: **procedure** `combinar-particiones`$(\{\mu'_j\},\{C_j^1\},\{C_j^2\})$
10: **for** $i \leftarrow 1$ to n **do**
11: **if** $\|x_i - \mu_{\alpha_i^1}\|^2 \leq \|x_i - \mu_{\alpha_i^2}\|^2$ **then**
12: $\alpha'_i \leftarrow \alpha_i^1$
13: **else**
14: $\alpha'_i \leftarrow \alpha_i^2$
15: **end if**
16: **end for**
17: **return** $\{C'_j\}$
▷ //Sub-procedimiento actualiza centroides
18: **procedure** `actualizar-centroides`
19: **for** $j'' \leftarrow 1$ to $|\{\mu'_j\}|$ **do**
20: $\mu'_j \leftarrow$ `calcular-centroides` $(\{C'_j\}, j')$
21: **end for**
22: **return** $\{\mu'_j\}$

El operador de cruce de dos puntos para grupos se utiliza en individuos basados en etiquetas con grupo de la forma $\chi_{\alpha\,1..k}$ (3.4) empleado para explorar diferentes número de grupos k_v y particiones óptimas. Se utiliza para generar dos hijos a partir de una pareja de padres.

El procedimiento del operador es descrito en [2] y los principales pasos se detallan a continuación:

(a) Primero, dos individuos son seleccionados aleatoriamente y dos puntos de cruce son establecidos de manera aleatoria en la sección de grupos del cromosoma.

Algoritmo 8 Unión de un par de grupos de vecinos más cercanos (PNN) [31]

Entrada: $\{\mu'_j\}, \{C'_j\}$: una agrupación formada por centroides de los grupos y su partición de tamaño $2 \cdot k$
Salida: $[\{\mu'_j\}|\{C'_j\}]$: una agrupación de tamaño k

```
for j' ← 1 to |{μ'_j}| do
    q_j' ← encontrar-vecinos-cercanos(μ'_j)
end for
while |{μ'_j}| > k'_v do
    a ← encontrar-distancia-minima({q_j'})
    b ← q_a
    unir-grupos(μ_a, P[a], μ_b, P[b])
    actualizar-punteros({q_j'})
end while
return [{μ'_j}|{C'_j}]
```

(b) En el primer hijo, se insertan los genes del primer padre de la sección de elementos que son indicados por los puntos de cruce.

(c) Similar al paso anterior, los genes del segundo padre se insertan con la condición de que no hayan sido asignados por el primer padre. Es decir, se asignan los genes que aún no han sido asignados de manera aleatoria con los grupos actuales.

(d) Se borran los grupos vacíos.

(e) Se modifican las etiquetas del hijo, a fin de numerar los k_v grupos.

Operador de mutación

El operador de mutación tiene como objetivo alterar uno o más genes en un individuo. Esto puede dar como resultado que el individuo tenga genes completamente nuevos que pueden llegar a una mejor solución que no era posible alcanzar anteriormente. Por lo tanto, este operador introduce diversidad permitiendo explorar nuevas regiones del espacio de búsqueda y escapar de los óptimos locales cuando el algoritmo está cerca de la convergencia. Existen diferentes operadores según la codificación que se utiliza en los AGs. Similar a la categorización realizada con los operadores de cruce, los operadores de mutación se van a clasificar como operadores de mutación convencionales que incluyen operadores que solo dependen del tipo de dato usado en la codificación sin considerar las restricciones y limitaciones del problema de agrupación y operadores de mutación especializados que tienen en cuenta las características concretas del problema de agrupación.

Operadores convencionales de mutación

Estos operadores incluyen pequeñas modificaciones en cada individuo de la población con una probabilidad baja independientemente de las conexiones y restricciones del problema de agrupamiento. Se pueden usar según el tipo de datos del cromosoma. Al igual que el operador de cruce, los operadores de mutación también pueden producir individuos inválidos, por lo que es necesario contar con un procedimiento para detectarlos. Se cuenta con dos opciones principalmente, modificar el individuo para convertirlo en válido o desecharlo y reemplazarlo por uno nuevo.

- *Mutación con un reemplazo*

  En la aplicación del operador de mutación con un reemplazo es seleccionado aleatoriamente un individuo para cambiar el valor de un gen por otro al azar obteniéndose un individuo mutado. Se puede utilizar para cualquier codificación, solamente se debe tener en cuenta el rango de valores que pueden ser asignados aleatoriamente para no generar individuos no válidos cuando se fija el número de grupos.

  Por ejemplo, la Figura 3.6 muestra dos ejemplos ilustrativos de la aplicación del operador de mutación con un reemplazo para el cromosoma χ_α (3.1). En el primer caso, se obtiene un individuo válido (Figura 3.6a), mientras que, en el segundo caso se obtiene un individuo que codifica una solución inválida debido a que se está trabajando con un algoritmo de k-fija y no existen instancias para el grupo C_4 (Figura 3.6a). La probabilidad de obtener individuos inválidos al aplicar la mutación con un reemplazo disminuye conforme aumenta el valor n del número de instancias.

- *Mutación con varios reemplazos*

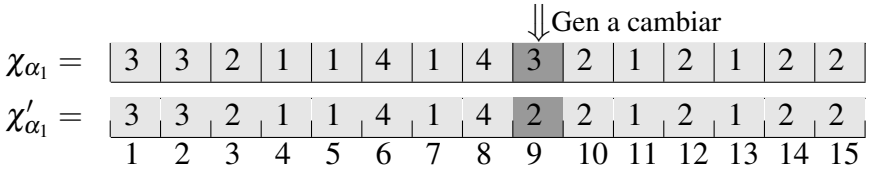
  Similar al operador con un reemplazo, en este operador cada gen del cromosoma seleccionado es modificado con una probabilidad predefinida.

Operadores de mutación especializados

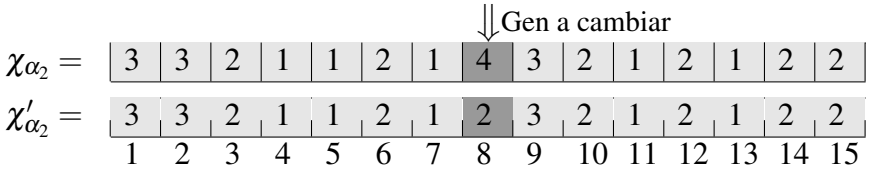
Estos operadores alteran los genes teniendo en cuenta las características particulares de los problemas de agrupamiento. Normalmente, se considera eliminar, modificar o insertar nuevos grupos.

- *Mutación basado en reemplazo por distancia*

  El operador de mutación basado en reemplazo por distancia es especificado en el Algoritmo (9) para individuos con cromosoma χ_α(3.1) basado en etiquetas con números



(a) El cromosoma χ_{α_1} es mutado al cambiar el valor del gen α_9 de 3 a 2 obteniéndose un nuevo individuo χ'_{α_1}



(b) Al mutar el cromosoma χ_{α_2} por cambiar el valor del gen α_8 de 4 a 2 se obtiene el individuo χ'_{α_2} inválido

Figura 3.6 Ejemplos de la aplicación del operador de mutación con un reemplazo en una codificación basada en etiquetas con números enteros.

enteros. Para la aplicación de este operador son seleccionados individuos con una probabilidad p_m.

El operador itera sobre los genes del cromosoma y para cada gen el valor puede ser cambiado dependiendo de la distancia a los centroides de los diferentes grupos con la instancia que representa el gen. Concretamente, la probabilidad de cambiar el valor del gen será más alta, cuanto más cerca esté la instancia de un nuevo centroide y tomará la etiqueta del nuevo grupo como su valor. La función de distribución de probabilidad es la parte central de este operador definida en la línea 10 del Algoritmo 9. Los AGs que utilizan este operador proponen dos diferentes funciones de probabilidad, la mostrada en la ecuación (3.20) [70] y la mostrada en la ecuación (3.21) [79].

$$p_j \leftarrow \frac{c_m d_{\max} - d_j}{\sum_{j'=1}^{k} (c_m d_{\max} - d_{j'})} \tag{3.20}$$

Donde c_m es una constante generalmente ≥ 1, $d_j = d(x_i, \mu_j)$ puede ser la distancia euclidiana entre x_i y μ_j y $d_{\max} = \max_j \{d_j\}$.

$$p_j \leftarrow \frac{1.5 d_{\max} - d_j + 0.5}{\sum_{j'=1}^{k} (1.5 d_{\max} - d_{j'} + 0.5)} \tag{3.21}$$

- *Mutación basada en la distribución uniforme.*

Algoritmo 9 `Mutación basada en reemplazo por distancia` [70]

Entrada: $\chi_\alpha = \langle \alpha_i \rangle$: un cromosoma (3.1)
Salida: χ_α: mutado

for $i \leftarrow 1$ to n **do**
  if $r < P_m$ **then** ▷ $r = U(0,1)$, $U(0,1)$ es una distribución uniforme en $[0,1]$
    Calcular los centroides $\{\mu_j\}$ correspondientes a $\langle \alpha_i \rangle$ con (2.6)
    for $j \leftarrow 1$ to k **do**
      $d_j = d(x_i, \mu_j)$
    end for
    if $d_{\alpha_i} > 0$ **then** ▷ El gen α_i es mutado si $|C_j| > 1$
      $d_{\max} \leftarrow \max\{d_1, d_2, ..., d_k\}$
      for $j \leftarrow 1$ to k **do**
        $p_j \leftarrow (c_m d_{\max} - d_j) / \sum_{j'=1}^{k} (c_m d_{\max} - d_{j'})$ ecuación (3.20)
        $\alpha_i =$ Un número aleatorio seleccionado de $\{1, 2, ...k\}$
          de acuerdo a la distribución $\{p_1, p_2, ...p_k\}$
      end for
    end if
  end if
end for
return $(\chi_\alpha = \langle \alpha_i \rangle)$

El operador de mutación basado en la distribución uniforme es aplicado a individuos con codificación basada en centroides χ_x (3.6). El operador utiliza un número γ obtenido de una distribución uniforme en el rango $[0,1]$ para modificar los genes, x_{lj}, por medio de las ecuaciones (3.22) y (3.23).

$$x_{lj} \pm 2 \cdot \gamma \cdot x_{lj}, \text{ si } x_{lj} \neq 0, \tag{3.22}$$

$$x_{lj} \pm 2 \cdot \gamma, \text{ si } x_{lj} = 0. \tag{3.23}$$

El $+$ o $-$ ocurren con una probabilidad igual, aunque también puede ser implementado por la ecuación (3.24):

$$x_{lj} \pm (\gamma + \varepsilon) \cdot x_{lj} \tag{3.24}$$

Donde $0 < \varepsilon < 1$ dado por [84].

- *Mutación en un punto*

El operador de mutación en un punto (D_PM)[8] [81] es aplicado a individuos con codificación basada en instancias representativas con números enteros χ_v (3.9). Un gen es seleccionado del cromosoma del individuo con una probabilidad predeterminada $P_{m,\text{punto}}$, cuando se tiene éxito es reemplazado por otro gen seleccionado de un subconjunto para obtener un individuo, χ'_v, con la restricción impuesta en los cromosomas, χ_v, donde una instancia representativa, v_j, solamente puede ocurrir una vez en el cromosoma.

Un ejemplo ilustrativo de la aplicación del operador de mutación D_PM se muestra en la Figura 3.7 donde es seleccionado el gen v_2 y se muta su valor de 9 a 3.

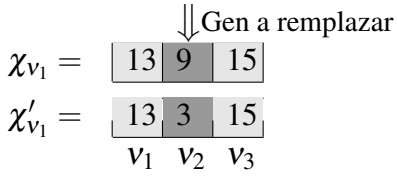


Figura 3.7 Ejemplo de la aplicación del operador de mutación en un punto (D_PM)

- *Mutación con reemplazo max-min*

  El operador de mutación basado en reemplazo máximo-mínimo es aplicado a individuos con codificación basada en centroides χ_x (3.6). Aunque puede ser utilizado como un operador convencional, es propuesto por primera vez en [7] en un AG de agrupación. Este operador también ha sido llamado como mutación bi-direccional en otra propuesta [55]. El operador es guiado por los valores máximo ($\mathscr{M}_{\max}$) y mínimo ($\mathscr{M}_{\max}$) de la métrica de agrupación obtenida por los individuos de la población actual. Para un individuo a mutar con métrica $\mathscr{M}$, un número δ es obtenido con una distribución uniforme en el rango $[-R,+R]$ utilizando la ecuación (3.25). Este operador obtiene un buen desempeño y ha sido utilizado por varios algoritmos.

$$R = \begin{cases} \frac{\mathscr{M}-\mathscr{M}_{\min}}{\mathscr{M}_{\max}-\mathscr{M}_{\min}}, & \text{si } \mathscr{M}_{\max} > \mathscr{M}, \\ 1, & \text{si } \mathscr{M}_{\min} = \mathscr{M}_{\max}. \end{cases} \tag{3.25}$$

  Con los valores $\max(x_{il})$ y $\min(x_{il})$ de cada una de las l-ésimas dimensiones del conjunto de datos X y las coordenadas de los centroides de la codificación, el individuo es mutado utilizando la ecuación (3.26).

[8] En inglés se le conoce a point mutation operator

$$\mu'_{jl} = \begin{cases} \mu_{jl} + \delta(\max(x_{il}) - \mu_{jl}), & \text{si } \delta \geq 0, \\ \mu_{jl} + \delta(\mu_{jl} - \min(x_{il})), & \text{en otro caso.} \end{cases} \tag{3.26}$$

Se puede ver que la ecuación de mutación (3.26) determina si el individuo se muta en función de valor $\mathscr{M}$ (cuanto más pequeño sea el CVI $\mathscr{M}$, mejor será la agrupación). Cuando $\mathscr{M} = \mathscr{M}_{\max}$ o $\mathscr{M} = \mathscr{M}_{\min} = \mathscr{M}_{\max}$, el individuo es mutado con la máxima perturbación. Mientras que cuando $\mathscr{M} = \mathscr{M}_{\min}$, se trata del mejor individuo y no es mutado.

- *Mutación basada en división y fusión*

  El operador de mutación basado en división y fusión de grupos es utilizado para explorar diferentes números de grupos. Es utilizado en varias propuestas de AGs [2, 3, 58, 60]. Los individuos utilizados por este operador utilizan una codifición basada en etiquetas, principalmente cromosomas $\chi_{\alpha k}$ (3.3) o $\chi_{\alpha 1..k}$ (3.4). No obstante, pueden ser aplicados otras codificaciones.

  Para aplicar esta mutación se usan dos operadores:

  `Fusión de grupos`, consiste en unir grupos seleccionados aleatoriamente, como se describe en el Algoritmo (10) `MO-1` , para ello el individuo debe contar con más de dos grupos.

  `División de grupos`, consiste en dividir grupos seleccionados aleatoriamente, como se describe en el Algoritmo (11) `MO-2`, para la división el grupo debe contar por lo menos con dos instancias.

  La manera de aplicar el operador `fusión de grupos` y `división de grupos` depende de la estrategia implementada por el AG.

3.1.5 Operador de búsqueda local

Dentro de la literatura de los AGs, se encuentran los algoritmos que incluyen en su ciclo evolutivo una *búsqueda local*. Estos algoritmos se llaman AGs meméticos, y su característica principal es que los individuos padres e hijos ganan experiencia a través de la búsqueda local incluida en su proceso de evolución [30].

En la resolución del problema de agrupamiento, muchas propuestas de AGs emplean una búsqueda local que les permite lograr que se acelere la convergencia. La búsqueda local permite explotar el espacio localmente para encontrar un óptimo en una vecindad cercana de un

Algoritmo 10 `MO-1` [3], Operador de mutación de fusión de grupos

Entrada: $\chi_\alpha = \langle \alpha_i \rangle$: un cromosoma (3.1), con k_v grupos
Salida: χ_α: mutado con k'_v grupos, donde $k'_v < k_v$

if $k_v > 2$ **then** ▷ Por las restricciones del problema de agrupación al reducir
  $r \leftarrow$ número aleatorio $\in \{1, ..., k_v - 2\}$;
  for $i \leftarrow 1$ to r **do**
    $C_j \leftarrow$ seleccionar un grupo aleatorio codificado en χ_α
    Cambiar las instancias de C_j al grupo más cercano C'_j, donde $C'_j \in \chi_\alpha$ y $j' \neq j$, de acuerdo a la distancia entre el objetos y el centroide del grupo C'_j
  end for
end if
return $(\chi_\alpha = \langle \alpha_i \rangle)$

Algoritmo 11 `MO-2` [3], Operador de mutación de división de grupos

Entrada: $\chi_\alpha = \langle \alpha_i \rangle$: un cromosoma (3.1), con k_v grupos
Salida: χ_α: mutado con k'_v grupos, donde $k'_v < k_v$

$r \leftarrow$ número aleatorio $\in \{1, ..., k_v\}$;
for $i \leftarrow 1$ to r **do**
  $C_j \leftarrow$ seleccionar un grupo aleatorio codificado en χ_α
  if $|C_j| > 2$ **then** ▷ El grupo debe tener más de dos instancias para dividirlo
    Seleccionar $x_i \in C_j$
    Buscar la instancia x'_i más lejos a x_i, donde $x'_i \in C_j$
    Cambiar las instancias de C_j al grupo más cercano C'_j, donde $C'_j \in \chi_\alpha$ y $j' \neq j$,
    Hacer dos nuevos grupos C'_j y C''_j cambiando las instancias de C_j más cercanas a x_i en C'_j y las más cercanas a x'_i en C''_j
    los grupos de la partición son $(\{C_j\} - C_j) \cup C'_j \cup C''_j$
  end if
end for
return $(\chi_\alpha = \langle \alpha_i \rangle)$

individuo. La búsqueda local se puede incluir como un paso más del ciclo evolutivo, o también, puede solamente activarse en determinados momentos como un número de generaciones concretos o el grado de convergencia de los individuos.

Las búsquedas locales que se han incluido en las propuestas de AGs para agrupamiento se detallan a continuación.

- Algoritmo de `K-means`

  A pesar de las limitaciones conocidas del algoritmo clásico `K-means` (1), su simplicidad y su capacidad de ajuste local lo hacen muy efectivo para ser elegido como parte de un AG [39]. Así, son muchas las propuestas que seleccionan este método [3, 58, 70, 79, 80, 93]

y proponen a `K-means` como un operador que en muchos casos reemplaza al operador de cruce. Otras propuestas [41, 55] lo utilizan para hacer un ajuste local en las soluciones representadas por los individuos de forma independiente de los operadores genéticos que utilizan.

La aplicación de `K-means` como operador de búsqueda local se basa en el Algoritmo (1) e incluye los siguientes pasos:

`Selección de individuo,` primero el individuo es seleccionado y decodificado a una solución de centroides $\{\mu_j\}$. Los centroides son inicializados como se establece en la línea 2 del Algoritmo (1).

`Paso de asignación,` todas las instancias son asignadas a un grupo con más semejanza considerando la distancia más cercana entre cada una de las instancias y los centroides (línea 4 del Algoritmo (1)).

`Paso de actualización,` los centroides de cada grupo se vuelven a calcular tomando la media de todas las instancias asignadas a cada uno de ellos (línea 5 del Algoritmo (1).).

`Paso de codificación,` después de realizar un número de iteraciones establecidas en el AG de los pasos `de asignación` y `de actualización`, el individuo es codificado.

Una ventaja de este método es su complejidad cerca de un tiempo lineal $O(gkn)$ donde g es el número de iteraciones.

- Un paso de `K-means`

  Algunos AGs [6–8, 84], aplican solamente una vez los pasos de `asignación` y `actualización` líneas 4 y 5 del Algoritmo `K-means` (1) para mejorar las soluciones. Este paso es aplicado después de utilizar los otros operadores genéticos. En este caso, tiene la ventaja de una complejidad pequeña, $O(kn)$.

- Algoritmo generalizado de Lloyd

  El algoritmo generalizado de Lloyd (GLA) propuesto por [74] se utiliza en cuantificación vectorial. Inicia con un libro de códigos que va iterando para mejorarlo hasta que un mínimo local es alcanzado. En el primer paso, cada vector de entrenamiento x_i (instancia) es mapeado a su vector de código (centroide μ_j) más cercano del libro de código. En el segundo paso, los vectores de código son recalculados como los centroides de la nueva partición. La nueva solución es siempre mejor o igual a la previa. El algoritmo se itera mientras se va obteniendo una solución mejor.

La ventaja de aplicar el método de GLA es que permite realizar un refinamiento progresivo de la solución, así se obtiene una mejor solución global de los vectores de código [41].

El resultado de GLA, al igual que `K-means`, depende de la selección del libro de códigos inicial.

- Algoritmo de búsqueda local heurística

  Este método de búsqueda local intenta encontrar las instancias representativas $\{v_j\}$ de los grupos $\{C_j\}$. El Algoritmo (12) describe la búsqueda local heurística, la cual tiene un diseño iterativo incremental y consta de los siguientes pasos:

  `Paso de inicialización,` selecciona k instancias aleatoriamente, las cuales representan las instancias iniciales representativas de cada grupo $\{C_j\}$ (línea 1 del Algoritmo (12)).

  `Paso de asignación,` asigna cada instancia x_i al grupo más cercano C_j con referencia a la instancia más representativa v_j bajo la métrica de distancia euclidiana (línea 3 del Algoritmo (12)).

  `Paso de actualización,` selecciona para cada grupo C_j un subconjunto de instancias $C_{\text{sub}_j} \subset C_j$ de tamaño p, más cercanas a la instancia representativa v_j y actualiza con una instancia más representativa si esta instancia es diferente a la actual (línea 7 del Algoritmo (12)).

  Los dos últimos pasos son iterados hasta converger a instancias representativas. El objetivo al obtener un subconjunto C_{sub_j} de tamaño pequeño p permite reducir la complejidad a $O(gpn)$ donde g es el número de iteraciones y para ello los autores proponen un valor de $p = 3$ donde otros algoritmos de instancias representativas tienen una complejidad $O(n^2)$ por iteración [116].

- Búsqueda local basada en vecinos

  Este procedimiento de búsqueda local es propuesto por [2]. Las acciones se dirigen a mejorar la función de aptitud del individuo que es seleccionado al realizar una ligera modificación sobre éste y determinar la variación de la función objetivo. Cuando se mueven objetos de un grupo a otro en la solución codificada por el individuo, el cambio se acepta si el resultado de la función objetivo es mejor al anterior.

  La desventaja de este procedimiento es la complejidad computacional, recalcular la función objetivo puede tener una complejidad de $O(n^2)$ para el CVI Silhouette. Esta

Algoritmo 12 Algoritmo de búsqueda local heurística [116]

Entrada: X: un conjunto de datos
k: número de grupos
p: número de vecinos más cercanos a explorar

Salida: Una partición de X en $\{C_j\}$ grupos basada en las instancias más representativas $\{v_j\}$, donde $j : j \in \{1,2,...,k\}$

1: Seleccionar de manera aleatoria k instancias del conjunto de datos X para inicializar: $v_1, v_2, ..., v_k$
2: **repeat**
3: Asignar cada x_i a su grupo C_j por su cercanía a su v_j con la métrica de distancia euclidiana:
$C_j^{(t)} \leftarrow \{x_i : \|x_i - v_j\| \leq \|x_i - v_{j'}\|,\ j' = 1,2,...,k\}$
4: **for** $j \leftarrow 1$ to k **do**
5: **repeat**
6: Con los miembros de C_j, seleccionar un subconjunto C_{sub_j}, formado por las p instancias más cercanas a v_j las cuales no han sido evaluadas antes de la actual iteración
7: Calcular la nueva instancia representativa

$$v_j \leftarrow \underset{x_i' \in C_{\text{sub}_j}}{\arg\min} \sum_{x_i \in C_j} D(x_i', x_i) \tag{3.27}$$

8: **until** no cambie v_j
9: **end for**
10: **until** no cambian $\{v_j\}$

complejidad se multiplica por las veces que cada objeto es cambiado de grupo. Para hacer frente a la complejidad de esta operación, sus autores proponen aplicar esta operación con una probabilidad muy baja.

3.2 Descripción de algoritmos genéticos para agrupación

En la sección 3.1 fueron descritos los componentes por separado de los diferentes AGs que resuelven el problema de agrupamiento basado en particiones, en esta sección, se describen la integración de los componentes que forman los AGs mono-objetivos desarrollados hasta la fecha, considerados el estado del arte en la resolución del problema de agrupamiento.

En esta descripción, se abordará por separado los AGs, los que son llamados de k-fija de antemano se necesita conocer el número de grupos, k, con el que se va a trabajar (sub-sección 3.2.1). Los otros AGs llamados de k-variable, incluyen en su procedimiento el descubrimiento del número de grupos, k, y por tanto, no necesitan conocer este valor con anticipación, estos serán descritos en la sub-sección 3.2.2.

3.2.1 Algoritmos genéticos para k-fija

En la literatura, se encontraron 12 propuestas englobadas en esta categoría. En esta sección se describen como se integran los componentes descritos en la sección 3.1 en uno de los AGs. Un resumen específico de los AGs para k-fija se muestra en la Tabla 3.1, la cual contiene los esquemas de codificación y los componentes principales de cada algoritmo desarrollado.

La descripción de los diferentes AGs de k-fija que se muestra en seguida se hace en división en función de la codificación, que es considerada la parte más representativa en el desarrollo de las diferentes propuestas de desarrolladas.

Codificación explícita basada en etiquetas

Los algoritmos descritos en esta sección obtienen una partición $\{C_j\}$ para una k definida de un conjunto de datos asociada a una codificación explícita basada en etiquetas.

- GA_B (Genetic Algorithm, la B se utiliza para diferenciar el AG de otros con el mismo nombre) [12]. Se le puede considerar el primer AG con un enfoque tradicional por los operadores que utiliza. La codificación que utiliza para los individuos es matriz de dígitos binarios donde cada cromosoma es representado por una $U_{k\times n}$ (2.16). Debido al comportamiento similar de esta representación con respecto a la basada en etiquetas se incluye en esta categoría. Utiliza el CVI de J_1 (7) como función aptitud, con alguna de las normas: euclidiana, diagonal o Mahalonobis. Los operadores genéticos que utiliza son: selección por elitismo, cruce en dos puntos y mutación con un reemplazo.

- GA (Genetic Algorithm) [92], este AG también puede ser catalogado dentro de los tradicionales por los operadores que utiliza. La codificación propuesta está basada en etiquetas con números enteros donde cada cromosoma tiene la forma χ_α (3.1). Su tamaño es de $|\chi_\alpha| = n$, siendo n el número de instancias del conjunto de datos. Esta codificación permite aplicar el operador tradicional de cruce en un punto. El operador de mutación con un reemplazo es aplicado con una probabilidad de inicio de $p_m = 0.5$ y en cada paso se decrementa hasta llegar a un valor de $1/n$ en la última generación. Para lograr una convergencia, se propone una estrategia elitista con una población de tamaño pequeño de 6 individuos, estableciendo que cuando el número de generaciones tiende a infinito, el modelo elitista de un AG debe proporcionar el individuo óptimo para una población de tamaño n_P. La función de aptitud que se emplea es el CVI SED (1).

- GKA (Genetic `K-means` Algorithm) [70]. Se trata de un AG también con un enfoque tradicional, pero incluye un procedimiento de búsqueda local. Utiliza una codificación basada en etiquetas con números enteros cuyo cromosoma es representado por χ_α (3.1).

Los individuos de la población $\mathscr{P}(0)$ son inicializados generando genes aleatorios de una distribución uniforme sobre el conjunto $\{1,...,k\}$. La evaluación se realiza con el CVI de TWCV (4) y la función aptitud la definen utilizando el mecanismo de σ-truncamiento donde $f(\chi_{\alpha_l}) = -\text{TWCV}(\chi_{\alpha_l})$, $g(\chi_{\alpha_l}) = f(\chi_{\alpha_l}) - (\bar{f} - c \cdot \sigma)$, donde $\bar{f}$ y σ denotan el valor promedio y la desviación estandar de $f(\chi_\alpha)$ en la población actual respectivamente y c es una constante entre 1 y 3. Así, el valor aptitud de χ_{α_l}, $\mathscr{F}(\chi_{\alpha_l})$ está dado por la ecuación 3.2.1.

$$\mathscr{F}(\chi_{\alpha_l}) = \begin{cases} g(\chi_{\alpha_l}), & \text{si } g(\chi_{\alpha_l}) \geq 0, \\ 0, & \text{otro caso.} \end{cases}$$

La actualización de la población a la siguiente generación viene dada por la selección por ruleta. Dentro de las innovaciones este AG propone no usar un operador de cruce, en lugar es cambiado por el algoritmo de `K-means`. Para la mutación proponen un operador especializado de agrupación llamado mutación basada en reemplazo por distancia utilizando la función de probabilidad (3.20). El operador de mutación puede tomar tiempo en converger si se utiliza una baja probabilidad de mutación p_m, por el contrario, si se utiliza una alta probabilidad puede hacer que el AG puede conducir a un comportamiento oscilante, por lo que el operador de un paso de `K-means` mejora esta situación. Los operadores genéticos pueden generar individuos ilegales por las restricciones del problema de agrupación. Cuando son detectados, los individuos inválidos son convertidos a válidos.

- FGKA (Fast Genetic `K-means` Algorithm) [79], basado en el algoritmo GKA [70] utiliza una codificación basada en etiquetas con números enteros cuyo cromosoma es representado por χ_α. La población $\mathscr{P}(0)$ es inicializada aleatoriamente. Los individuos que son detectados por el AG que son inválidos, a diferencia de GKA [70], les son asignada una función objetivo muy alta, TWCV $+\infty$, con el fin de evitar el consumo de recursos que requiere la eliminación de individuos inválidos.

  La actualización de la población se realizar por la selección por selección por ruleta, llamada *selección proporcional* por los autores, definiendo la función aptitud $\mathscr{F}(\chi_{\alpha_l})$ de acuerdo con la ecuación 3.28.

$$\mathscr{F}(\chi_{\alpha_l}) = \begin{cases} 1.5 \cdot \text{TWCV}_{\max} - \text{TWCV}(\chi_{\alpha_l}), & \text{si } \chi_{\alpha_l} \text{ es legal} \\ e(\chi_{\alpha_l}) \cdot \mathscr{F}_{\min}, & \text{otro caso.} \end{cases} \tag{3.28}$$

  Donde $\text{TWCV}_{\max}$ es el valor máximo que ha sido encontrado hasta la presente generación, $\mathscr{F}_{\min}$ es el valor más bajo de la función aptitud de un individuo válido en la población

actual si existe, en otro caso, $\mathscr{F}_{\min}$ es definido como 1, $e(\chi_{\alpha_\iota})$ es el *radio de la legalidad* de χ_{α_ι} obtenido al dividir el número de grupos no vacíos entre k, si legal $e(\chi_{\alpha_\iota}) = 1$ es válido, en otro caso es ilegal o inválido.

FGKA utiliza el mismo operador de mutación que GKA (mutación basada en reemplazo por distancia) utilizando la función de probabilidad (3.21). También el operador de un paso de `K-means` es aplicado para acelerar el proceso de convergencia.

- IGKA (Incremental Genetich `K-means` Algorithm) [80], está inspirado en FGKA [79] con el objetivo de mejorar el desempeño y la convergencia. Los autores argumentan que, si la probabilidad de la mutación p_m es pequeña, entonces los cambios deben ser mínimos en los genes y el costo de calcular los centroides y TWCV nuevamente puede ser muy costoso, entonces proponen una nueva forma para calcularlos de forma incremental. Los operadores que utiliza son: selección por ruleta, con función aptitud (3.28), el mismo operador de (mutación basada en reemplazo por distancia) con la función de probabilidad (3.21) y el operador de un paso de `K-means`.

Codificación implícita basada en centroides

Los AGs en esta sección buscan ubicar con precisión el centroide de los grupos para una k definida por el usuario y por medio de la ecuación (2.4) es posible obtener la partición del conjunto de datos. Las soluciones son codificadas como un vector consecutivo de coordenadas de los centroides.

- GAs (Genetic Algorithm) [84], los autores lo plantean para determinar los centroides en $\mathbb{R}^n$ para un número fijo de k grupos. GAs utiliza una codificación basada en centroides con números decimales cuyo cromosoma es representado por χ_x (3.6) con un tamaño $|\chi_x| = d \cdot k$. Para inicializar los cromosomas, para cada uno de ellos, son seleccionadas k instancias aleatorias del conjunto de datos y son concatenadas hasta completar el vector. Se le puede considerar un AG tradicional por los operadores que utiliza. La selección se realiza por copias de individuos de la población actual proporcional a la función de aptitud (selección por ruleta). El operador de cruce en un punto es aplicado con una probabilidad p_c, un número aleatorio en el intervalo $[1, |\chi_x| - 1]$ es generado y utilizado como punto de cruce. La mutación distribución uniforme es aplicada con una probabilidad p_m. La función de aptitud es calculada en dos pasos. En el primero son recalculados los centroides reasignando las x_i al centroide más cercano con la ecuación (2.5) y obteniendo los nuevos centroides con la ecuación (2.6) (llamado aquí operador de un paso de `K-means`). En el segundo es calculado el CVI de SED (1) para cada individuo, y la función de aptitud es calculada como $\mathscr{F} = 1/\text{SED}$, porque debe ser maximizada.

- KGA (Genetic Algorithm with `K-means`) [7], es un AG diseñado por los mismos autores de GAs [84]. Este AG incluye características similares a su propuesta inicial tales como codificación basada en centroides con números decimales, selección por ruleta y operador de cruce en un punto. Introducen como novedoso un operador de mutación llamado mutación basada en reemplazo máximo-mínimo, el cual es utilizado posteriormente por otros AGs. El operador de agrupación de un paso de `K-means` es incluido como un paso del AG. Optimiza el CVI de SED (1). En los parámetros de configuración aumenta el número de generaciones haciendo que la complejidad de tiempo aumente, pero se espera obtener mejores resultados en la agrupación.

- GAGR (Genetic Algorithm with Gene Reordenation) [18], debido a que en el problema de agrupación muchas de las codificaciones de los cromosomas son redundantes como se describió en la sección 3.1.1, este algoritmo trata de mejorar su desempeño evitando la redundancia, haciendo que el proceso evolutivo converja más rápido. Con esta finalidad, se incluye un paso de reordenamiento de genes cuando es requerido utilizando el mejor individuo como referencia. En su diseño utiliza una codificación basada en centroides con números decimales cuyo cromosoma es representado por χ_x (3.6). La población de tamaño n_P es inicializada generando k instancias aleatorias escogidas del conjunto de datos con la condición de que no sean idénticas y representan los k centros de los grupos. Después de que la población es inicializada, para cada individuo las instancias son asignadas a su grupo con los centroides que codifica y la ecuación (2.4). Como método de selección utiliza la selección por ruleta para la evolución con los operadores cruce y mutación. En la aplicación de los operadores de cruce y mutación una probabilidad adaptativa es utilizada, por lo que es aplicada cuando es requerida la exploración y explotación y de esta manera se reduce la complejidad de tiempo del AG. Combina dos operadores de cruce, uno de ellos es un nuevo operador que llaman cruce basado en un camino, que combinan con otro operador para lograr producir nuevos individuos hijos, y que llaman cruce heurístico. La mutación basada en reemplazo máximo-mínimo es aplicada de manera individual a los individuos que son seleccionados. El CVI utilizado como función de aptitud es SSE (3), la función aptitud es definida como el reciproco de SSE, $\mathscr{F} = 1/\text{SSE}$.

- CBGA (Code Book Genetic Algorithm) [41], este algoritmo se le puede clasificar como un AG no tradicional por los operados utilizados. Inicialmente aplicado a procesos de cuantificación vectorial está basado en el algoritmo PNN [31] que utiliza para diseñar el operador de cruce-solución. CBGA utiliza la codificación basada en la cuantificación vectorial cuyo cromosoma es representado por χ_{cb} (3.8). Por la manera que opera el operador

de cruce se le puede considerar una extensión de la codificación basada en centroides. Esta codificación presenta como ventaja que es más eficiente computacionalmente en cada paso del algoritmo para mantener tanto la partición $\{C_j\}$, como los centroides $\{\mu_j\}$. Para la aplicación del operador de cruce en la población se utiliza una selección por elitismo, así los pares de individuos son permutados hasta completar la población para la siguiente generación. El operador de mutación es llamado intercambio aleatorio, en el cual un individuo es reemplazado por un individuo construido nuevamente con el propósito de evitar una convergencia rápida. Después de aplicar el operador de cruce, la búsqueda local se utiliza para afinar la solución con el algoritmo generalizado de Lloyd. El CVI utilizado es DD (5).

Codificación implícita basada en instancias representativas

Los AGs en esta sección utilizan las instancias representativas para con una k definida por el usuario, con estas se obtienen las particiones más representativas del conjunto de datos.

- GA-P (Genetic Algorithm for Prototypes) [71], los autores proponen el primer algoritmo para encontrar las instancias representativas o prototipos. Utilizan una codificación basada en una cadena binaria cuyo cromosoma es representado por χ_β (3.10). Con esta codificación es posible usar los operadores tradicionales: cruce uniforme y mutación con varios reemplazos. Debido a que el algoritmo es planteado para k-fija al aplicar los operadores puede suceder que se obtengan individuos con un número de grupos diferente al planteado, por lo que se emplea una función de aptitud que introduce penalizaciones para los individuos inválidos basada en J_1 (7). De este modo, los individuos que obtienen una k diferente son eliminados en la selección, definida por la ecuación 3.29.

$$\mathscr{F}(\chi_\beta) = J_1(U(\chi_\beta), \psi(U(\chi_\beta)) + \alpha(|\chi_\beta| - k)^2 \tag{3.29}$$

  donde el coeficiente $\alpha > 0$ introduce un peso que permite penalizar los individuos inválidos. Tiene el inconveniente de que debe ser ajustado para los diferentes conjuntos de datos.

- GCA (Genetic Algorithm for k-medoid Clustering) [81]. Utiliza una codificación basada en instancias representativas cuyo cromosoma es representado por χ_v (3.9) con tamaño fijo de k genes. Los individuos son inicializados de manera aleatoria. Para la aplicación de los operadores genéticos los individuos son seleccionados por ruleta, utilizando el CVI SED (2) donde la suma de distancias es entre la instancia representativa de cada grupo y las que son miembros del grupo, la cual debe minimizarse. Por tanto, la función de aptitud

propuesta es $\mathscr{F} = -\text{SED}$ o el recíproco $\mathscr{F} = 1/\text{SED}$. Debido a las restricciones de esta codificación no es posible utilizar operadores tradicionales, por tanto, son propuestos los operadores genéticos especializados, el de cruce D_MX y la mutación D_PM. Al aplicar los operadores y obtener la nueva población el número de individuos se mantiene igual en cada generación.

- HKA (A Hybrid Algorithm for K-medoid Clustering) [116], los autores proponen usar una búsqueda local heurística para encontrar instancias representativas que minimicen la diferencia entre cada grupo. Utiliza una codificación basada en instancias representativas cuyo cromosoma es representado por χ_v (3.9) con un tamaño fijo. Inicializa una población $\mathscr{P}(0)$ de tamaño n_P de manera aleatoria sin genes duplicados. La función de aptitud que usa es el CVI de SED, $\mathscr{F} = 1/\text{SED}$ con el objeto de minimizar SED (2). Como método de selección utiliza la selección por torneo entre dos individuos para obtener $n_P/2$ pares de padres. El operador de cruce es el propuesto por [81] D_MX con una probabilidad de p_c. El operador de mutación es el D_PM con una probabilidad p_m.

3.2.2 Algoritmos genéticos para k-variable

En la literatura, se encontraron 11 propuestas englobadas en esta categoría. Estos algoritmos permiten optimizar tanto el número de grupos, como las particiones correspondientes. Siguiendo la clasificación de los componentes realizada en la sección anterior, en esta sección se describen los componentes integrados que utilizan cada una de las propuestas cuyo resumen se puede consultar en la Tabla 3.2.

En las siguientes secciones se hace una división de las propuestas en función de la codificación, que es considerada la parte más representativa en el desarrollo de las diferentes propuestas de AGs desarrolladas.

Codificación implícita basada en etiquetas

En esta categoría se encuentran los AGs que utilizan la codificación basada en etiquetas o alguna variante de esta. Es interesante comentar que la primera propuesta en esta categoría es el algoritmo GGA [36], el cual a evolucionado en una sucesión de diferentes AGs con configuraciones y operadores diversos hasta llegar a una nueva versión del algoritmo GGA [2].

- CGA (Clustering Genetic Algorithm) [60], este algoritmo es un intento de mejora al algoritmo GGA [36]. Utiliza operadores de cruce y mutación especializados al establecer que los operadores tradicionales no son adecuados para el problema de agrupación al igual que [36]. Los operadores son adaptados para trabajar con la codificación de tamaño fijo

basada en etiquetas de números enteros con los k grupos cuyo cromosoma es representado por $\chi_{\alpha k}$ (3.3) en lugar de $\chi_{\alpha 1..k}$ (3.4) utilizado en GGA. El operador de cruce combina información de los individuos el cual es descrito como *cruce por combinación de grupos* y el operador de *mutación basada en división y fusión de grupos*, ambos operadores hacen una exploración de la búsqueda del número de grupos óptimo. La función objetivo empleada por CGA es el promedio de Silhouette (S (13)) desarrollada por [65]. Debido a que la selección por ruleta opera mejor con valores positivos ($-1 \leq S(x_i) \leq 1$), al promedio de S se le suma 1 cambiando el rango de la función de aptitud a $0 \leq S(x_i) \leq 2$. Finalmente, una estrategia elitista es aplicada al copiar el mejor individuo a la siguiente generación.

- EAC (Evolutionary Algorithm for Clustering) [58], es una extensión del algoritmo CGA [60]. Para su diseño realizan tres modificaciones sobre CGA: (i) la primera, es la utilización del Algoritmo `K-means` como una técnica de búsqueda local obteniendo a CGA-II, (ii) la segunda, proponen un nuevo CVI basado en S (13), llamado Simplified Silhouette (SS (14)) el cual utiliza los centroides, obteniendo CGA-III y (iii) la tercera, eliminan el operador de cruce de CGA-III y mejoran el operador de mutación, para finalmente obtener al algoritmo EAC.

- FEAC (Fast Evolutionary Algorithm for Clustering) [3, 93], es un algoritmo que evoluciona a apartir de EAC [58] con la motivación de que sea más eficiente. Para su diseño, primero, evalúan a EAC, y después, hacen cuatro variantes de EAC. De acuerdo con la experimentación, seleccionan la propuesta de FEAC. EAC-I es la primera variante con la estrategia de obtener grupos de mayor calidad al aplicar el operador de mutación de división y fusión de grupos, en el cual son seleccionados los grupos de manera aleatoria, la hipótesis que se plantean es que grupos con menor valor para la función aptitud es más probable que sean mutados, por lo que la función aptitud debe ser calculado de manera parcial a cada grupo. EAC-II normaliza de manera lineal la función de aptitud de cada grupo. Para las variantes anteriores en la mutación el operador `MO-1` (Algoritmo 10) y `MO-2` (Algoritmo 11) cada uno se aplica al 50% de los individuos a mutar, para mejor se propone cambiar la proporción de acuerdo con el desempeño de los operadores, calculando el promedio de la función aptitud antes y después de ser mutados para `MO-1` (Algoritmo 10), con la diferencia de valores es calculada las proporciones de aplicar los operadores, de esta manera se obtiene EAC-III. Finalmente, EAC-IV hace una mejora de desempeño en el operador `MO-1` (Algoritmo 10) al seleccionar un grupo C_j y pasando todas las instancia al grupo más cercano C'_j reduciendo la complejidad de $O(k_v \cdot n)$ a $O(k_v)$. Para la función de aptitud se proponen usar dos CVI el SS (14) y Ω (27) utilizando como

selector, la selección por ruleta incluyendo los rangos. Los resultados indican que los algoritmos EAC-II y EAC-III mostraron una mayor eficiencia estadísticamente en comparación con el EAC original. De acuerdo con lo anterior, combinaron las características de EAC-II y EAC-III en una variante llamada F-EAC.

- GGA (Grouping Genetic Algorithm) [2], proponen un algoritmo que emplea una codificación basada en etiquetas de números enteros con los k grupos cuyo cromosoma es representado por $\chi_{\alpha\,1..k}$ (3.4). Esta codificación también es utilizada por GGA [36]. Para la selección de los individuos se usa la selección por ruleta. Utiliza un operador de cruce novedoso llamado cruce de dos puntos para grupos. El operador de mutación es equivalente a la mutación de división y fusión de grupos, al aplicarlo solo se debe ser consistente en la sección de grupos del cromosoma. En la evolución proponen un modelo de islas no utilizado en los otros AGs con el objeto de paralelizar y mejorar el desempeño de GGA. La manera en que lo aplican es un modelo de migración elitista en el cual el mejor individuo de cada isla migra y sustituye aleatoriamente un individuo de las otras islas con una probabilidad de migración p_e. También, se propone una búsqueda local no aplicada antes llamada búsqueda basada en vecinos. Como función de aptitud se proponen dos CVIs, S (13) y DB (11).

Codificación implícita basada en centroides

En esta categoría se encuentran los AGs que codifican alguna variante de las codificaciones basadas en centroides. Se hallan diversas propuestas con características diferentes entre ellas.

- GCUK (Genetic Clustering for Unknown k) [8], proponen una codificación basada en centroides con símbolos de indiferencia cuyo cromosoma es representado por $(\chi_{\mu\#})$ que permite tener una población de individuos de diferente número de grupos k_v. Además, utiliza la selección por ruleta y en cada generación se aplica una estrategia elitista. El operador de cruce en un punto es utilizado en la explotación y exploración de nuevos individuos y el operador de mutación con distribución uniforme es aplicado sobre genes que no sean símbolos de indiferencia #. Para encontrar el apropiado número de grupos y la adecuada partición la función objetivo a minimizar es el CVI de DB (11), por lo que la función de aptitud es definida $1/\text{DB}$. Antes de calcular la función aptitud, las instancias son asignadas al centroide más cercano, recalculando los centroides, por lo que se realiza un paso de `K-means`.

- VGA (Variable-string-length Genetic Algorithm) [6], es utilizado para determinar de manera automática el número de grupos, así como la apropiada agrupación de datos. Con

una codificación basada en centroides cuyo cromosoma es representado por χ_μ (3.7) y la restricción $k_v \geq 2$. Los diferentes individuos son inicializados con una k_v aleatoria hasta un límite superior $k_{\max}$ dado, los centroides de cada individuo son seleccionados aleatoriamente de las instancias del conjunto de datos. Se le puede considerar un AG tradicional por utilizar operadores clásicos: la selección por ruleta, el cruce en un punto, restringiendo el cruce solamente entre centroides como se describió en la codificación basada en centroides y la mutación con distribución uniforme. Para optimizar los individuos utiliza una búsqueda local de Un paso de `K-means`. En VGA diferentes CVIs son evaluados como función de aptitud: el índice DB (11), el índice Dunn (16) y proponen un nuevo CVI llamado índice I (21) que depende del parámetro p. Aunque no realizan pruebas de manera exhaustiva para la selección del CVI, proponen utilizar el índice I.

- TGCA (A Two-stage Genetic Algorithm for Automatic Clustering) [55], este algoritmo, como todos los incluidos en esta categoría, considera que el número de grupos es desconocido, por tanto, su primera tarea es buscar el número óptimo de grupos k_v. Así, TGCA aplica un proceso de selección en dos etapas utilizando la variable k_{con} (3.15), con una codificación basada en centroides de longitud variable cuyo cromosoma es representado por χ_μ (3.7). Esta codificación permite al algoritmo realizar una búsqueda tanto de la ubicación óptima de los centroides de los grupos, así como el número de grupos óptimo. A diferencia de la mayoría de los AGs donde la inicialización de los individuos es por un método aleatorio de las instancias del conjunto de datos, en TGCA desarrollan un método de partición de rangos de atributos máximo, el cual es dividido en k_v segmentos, al utilizar el resultado del algoritmo de agrupación jerárquico aglomerativo. El operador de cruce que aplica es el tradicional cruce en un punto, el cual es implementado en subpoblaciones, llamándole *cruce paralelo*. En la mutación se aplica una estrategia de *mutación de dos estados* basada en la variable k_{con} (3.15). De la misma manera que en la selección donde el primer estado del algoritmo tiene como objetivo la búsqueda de la k_v, cuando k_{con} está en el intervalo $0 < k_{\text{con}} \leq 0.9$, el operador actúa sobre k_v obteniendo una k'_v aleatoria, en el segundo estado cuando $0.9 < k_{\text{con}} \leq 1$ aplica el operador de mutación con reemplazo max-min. Para obtener los nuevos centroides de los grupos en cada generación antes de ser evaluados los individuos aplican el algoritmo `K-means`. La función de aptitud utilizada está basada en un CVI que identifica el *punto de dobles*[9] relacionado con el número de grupos, VRC (19).

[9] En inglés se le llama knee point o elbow

Codificación implícita basada en grafos

En esta categoría se encuentran los AGs que utilizan una codificación basada en grafos.

- GA_C (Genetic Algorithm, C es para distinguir el algoritmo) [17], es uno de los primeros AGs que utilizan una codificación basada en grafos. Concretamente, las instancias de un conjunto de datos son representadas como los vértices de un grafo. Este algoritmo utiliza la codificación basada en locus de adyacencias con dígitos binarios y para su representación necesita obtener el árbol de cobertura mínimo. GA_C utiliza una selección de individuos proporcional al valor de la función aptitud, la (selección por ruleta) utilizando el CVI de VRC (19) como función aptitud, el cual debe ser maximizado para obtener una mejor agrupación. Debido a que los cromosomas son vectores de dígitos binarios es posible aplicar operadores genéticos tradicionales. Concretamente, se emplea el cruce en un punto y la mutación con un reemplazo. Mientas que la mayoría AGs establece terminar con un número de generaciones fijo, en esta propuesta agregan el criterio detener el AG cuando no cambia el mejor individuo después de un número de generaciones.

- CLUSTERING (Genetic Clustering Algorithm) [122], es un algoritmo dividido en dos etapas. La primera etapa consiste en obtener bloques de instancias vecinas codificados en una cadena por dígitos binarios de longitud m, siendo la base para la representación de los individuos en el AG. Por ello, esta codificación se ha llamado en este trabajo, codificación basada en bloques vecinos con dígitos binarios dentro de las codificaciones basadas en un grafo. La segunda etapa del algoritmo es la aplicación de un AG que se encargará de agrupar los m bloques en k_v grupos. Al utilizar un cromosoma con dígitos binarios, el AG utiliza operadores tradicionales. Inicia construyendo una población de individuos de tamaño m inicializados aleatoriamente, aplicando el operador de cruce en dos puntos y la mutación con un reemplazo. Utiliza una función de aptitud compuesta de dos componentes: D_{intra} que define la distancia entre cada grupo y D_{inter} que define la distancia entre el grupo C_j y el resto de los otros grupos. La función de aptitud estará dada por IC (23), por ser una ecuación que depende de la codificación no puede ser usada como un CVI general.

Tabla 3.1 Descripción de los AGs de *k*-fixed

Algoritmo Genético		Codificación	Función Aptitud	Selección	Cruce	Mutación	Búsqueda Local
GA [92]	Codificación explícita	Etiquetas con números enteros	SED (1)	Selección por ruleta	Cruce en un punto	Mutación con un reemplazo	
GKA [70]		Etiquetas con números enteros	TWCV (4)	Selección por ruleta		Reemplazo por distancia	Algoritmo `K-means`
IGKA [80]		Etiquetas con números enteros	TWCV (4)	Selección por ruleta		Reemplazo por distancia	Algoritmo `K-means`
FGKA [79]		Etiquetas con números enteros	TWCV (4)	Selección por ruleta		Reemplazo por distancia	Algoritmo `K-means`
CBGA [41]		Centroides con números decimales	DD (5)	Elitismo	Cruce vecinos cercanos	Intercambio aleatorio	`Método GLA`
GA_B [12]		Matriz de digitos binarios	J_1 (7)	Elitismo	Cruce en dos puntos	Mutación con un reemplazo	
GA-P [71]	Codificación implicita	Instancias representativas binarias	J_1 (7)	Elitismo	Cruce uniforme	Mutación varios remplazos	
GCA [81]		Instancias representativas enteros	SED (2)	Selección por ruleta	Cruce D_MX	Mutación D_PM	
HKA [116]		Instancias representativas enteros	SED (2)	Selección por torneo	Cruce D_MX	Mutación D_PM	Búsqueda heurística
GAGR [18]		Centroides con números decimales	SSE (3)	Selección por ruleta	Cruce camino y heurístico	Reemplazo con max-min	Un paso de `K-means`
GAs [84]		Centroides con números decimales	SED (1)	Selección por ruleta	Cruce en un punto	Mutación distribución uniforme	Un paso de `K-means`
KGA [7]		Centroides con número decimales	SED (1)	Selección por ruleta	Cruce en un punto	Reemplazo con max-min	Un paso de `K-means`

Tabla 3.2 Descripción de los AGs de agrupación de *k*-variable

Algoritmo Genético		Codificación	Función Aptitud	Selección	Cruce	Mutación	Búsqueda Local
CGA [60]	Codificación explícita	Etiquetas con grupos y números enteros	S (13)	Selección por ruleta con rangos	Combinación de grupos	División y fusión de grupos	
GGA_S [2] GGA_{DB} [2]		Etiquetas con grupos y números enteros	S (13) y DB (11)	Selección por ruleta	Cruce de dos puntos para grupos	División y fusión de grupos	Búsqueda basada en vecinos
EAC [58]		Etiquetas con números enteros	SS (14)	Selección por ruleta con rangos		División y fusión de grupos	Algoritmo `K-means`
$FEAC_{RI}$ [3] $FEAC_{SS}$ [93]		Etiquetas con números enteros	Ω (27) SS (14)	Selección por ruleta con rangos		División y fusión de grupos	Algoritmo `K-means`
GCUK[8]	Codificación implicita	Centroides con símbolos de indiferencia y números decimales	DB (11)	Selección por ruleta	Cruce en un punto	Mutación distribución uniforme	Un paso de `K-means`
VGA [6]		Centroides con número decimales	I (21)	Selección por ruleta	Cruce en un punto	Mutación distribución uniforme	Un paso de `K-means`
TGCA[55]		Centroides con número decimales	VRC (19)	Selección en dos etapas	Cruce en un punto	Reemplazo con max-min	Algoritmo `K-means`
GA_C[17]		Locus de adyacencias con dígitos binarios	VRC (19)		Cruce en un punto	Mutación con un reemplazo	
CLUSTERING[122]		Bloques vecinos con dígitos binarios	IC (23)		Cruce en dos puntos	Mutación con un reemplazo	

3.3 Taxonomía propuesta de algoritmos genéticos

Esta tesis propone una nueva taxonomía que permite clasificar los AGs existentes y las nuevas propuestas. Para llevarla a cabo, se basa en los componentes principales que conforman los AGs y que han sido descritos en la sección 3.1 y que permiten identificar los diferentes AGs. La codificación, la función de aptitud, el uso de búsqueda local y los operadores genéticos constituyen un conjunto de propiedades que caracterizan los diferentes métodos propuestos hasta la fecha. Esta sección presenta una taxonomía basada en las que se han considerado las propiedades más relevantes para identificar a las propuestas. Las Figuras 3.8 y 3.9 ilustran la categorización siguiendo una estructura jerárquica: si es conocido el número de grupos, el tipo de codificación, la utilización de búsqueda local, la función de aptitud y el operador genético. De este modo, se diferencian cuatro categorías principales en la taxonomía propuesta:

- Número de grupos conocidos o sugerido. Esta característica es fundamental para clasificar los algoritmos ya que sus objetivos son diferentes, y por tanto, los AGs tienen diferente representación y desempeño. Hay dos categorías de algoritmos para el problema de agrupación basado en particiones: aquellos en los que el número de agrupamiento k es un parámetro de entrada conocido, ampliamente referenciados en la literatura como algoritmos de *k-fija*, o aquellos que no necesitan este valor y se estiman automáticamente, conocidos como algoritmos de *k-variable*.

  Siempre que sea conocido el número de grupos, es un valor que es conveniente utilizar al hacer la agrupación, permitiendo optimizar los recursos empleados en obtener la solución óptima. No obstante, en los casos en los que no se disponga esta información, se deberá acudir a los otros algoritmos. De las taxonomías ya realizadas en la literatura, el número de grupos es el elemento diferenciador utilizado para caracterizar los diferentes AGs.

- Codificación. La codificación utilizada por los AGs es otro elemento distintivo en su caracterización, por lo que debe ser considerada en la segunda determinación taxonómica. Como se mencionó, la codificación de los individuos permite explorar el espacio de soluciones de diferentes maneras, y también es determinante en el resultado obtenido por los algoritmos. La mayoría de los algoritmos utilizan dos codificaciones basadas en etiquetas y en centroides. Dentro de estas codificaciones se encuentran algunas especializaciones que son utilizadas en algunos AGs. Otras codificaciones que marcan diferencias entre los AGs son basadas en instancias representativas y en grafos.

- Búsqueda local. Debido al amplio uso de este operador en los AGs que resuelven el problema de agrupamiento, ha sido considerado parte de la taxonomía. Algunas propuestas lo utilizan en sustitución del operador de cruce, otras lo usan como complemento a

los operadores genéticos para acelerar la convergencia explotando el espacio localmente para encontrar un óptimo en una vecindad cercana a la solución actual. El operador más común de búsqueda local utilizado es algoritmo de `K-means`. Muchas propuestas solamente aplican una iteración, otras lo aplican hasta que logran una convergencia. Otros algoritmos incluyen otras optimizaciones locales como la búsqueda local heurística y búsqueda basada en los vecinos más cercanos.

- Función aptitud. Como se mencionó en la sección 3.1.2, diferentes CVIs son usados para realizar la optimización de los AGs, por lo que puede ser considerada otra característica que permite diferenciar a los algoritmos. Principalmente, los dos criterios de grupos que intentan analizar los CVIs son la separación y la compactibilidad. La separación está más relacionada con los AGs de *k*-fija, siendo la medida SED la más común entre los AGs. En el caso de los AGs de *k*-variable los CVIs más comunes son VRC y SS y se evalúa tanto la separación como la compactabilidad.

En las Figuras 3.8 y 3.9, se muestra la taxonomía que se ha comentado anteriormente clasificando los distintos algoritmos que se estudian. La primera caracterización divide los AGs entre los que requieren el número de grupos (Figura 3.8) o los que lo determinan de manera automática (Figura 3.9). Los siguientes niveles del árbol taxonómico en ambas figuras determinan las propiedades de la codificación utilizada, el uso de búsqueda local y la función de aptitud de los diferentes AGs. Las hojas de los árboles determinan cada AG específico y nos muestra fácilmente cómo de similares son. Además, los colores con los que se representan los AGs también nos indica la similitud entre ellos en cuestión de los resultados que obtienen, por lo que pueden ser usados indistintamente.

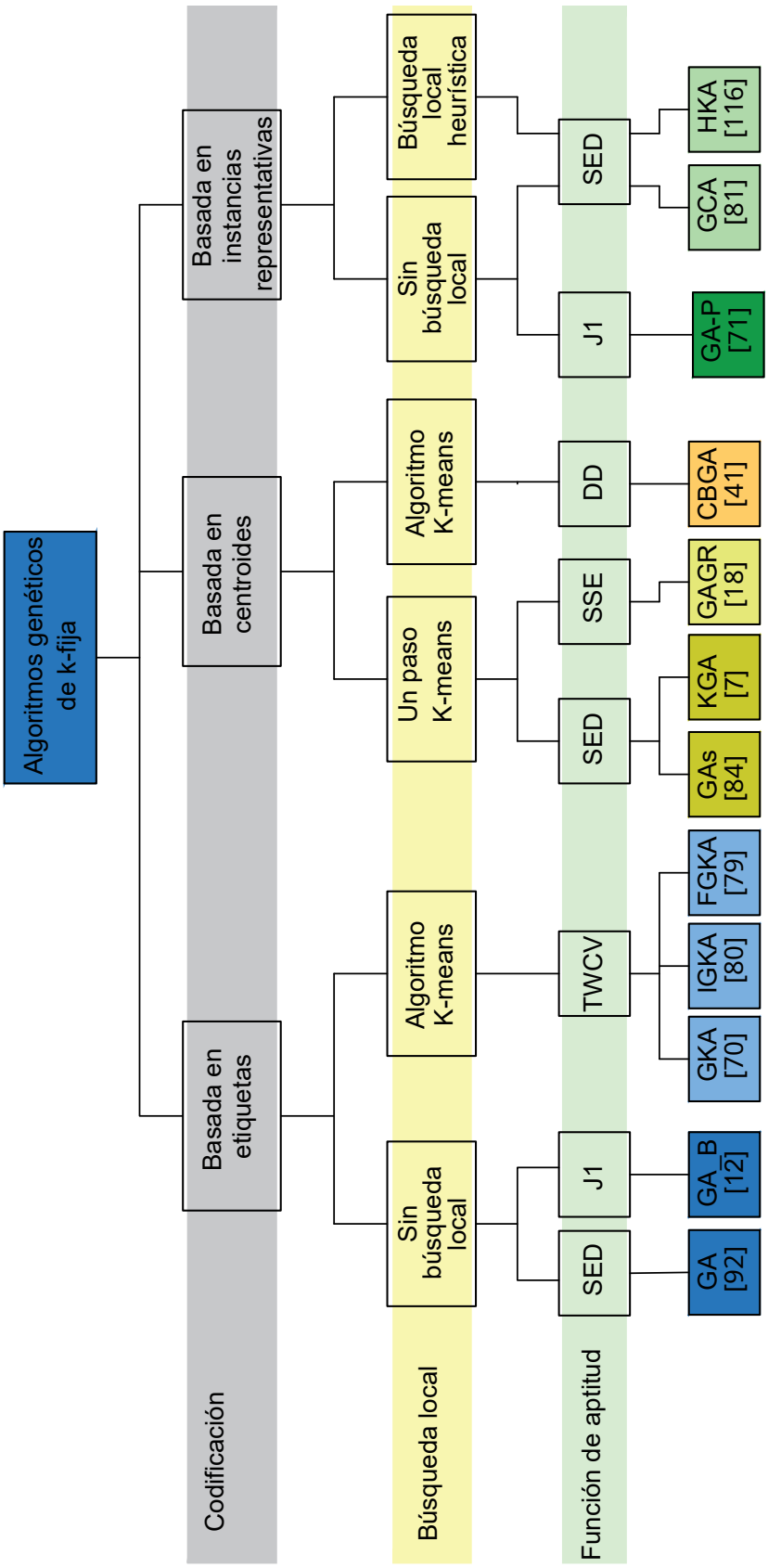


Figura 3.8 Taxonomía para los AGs basados en k-fija

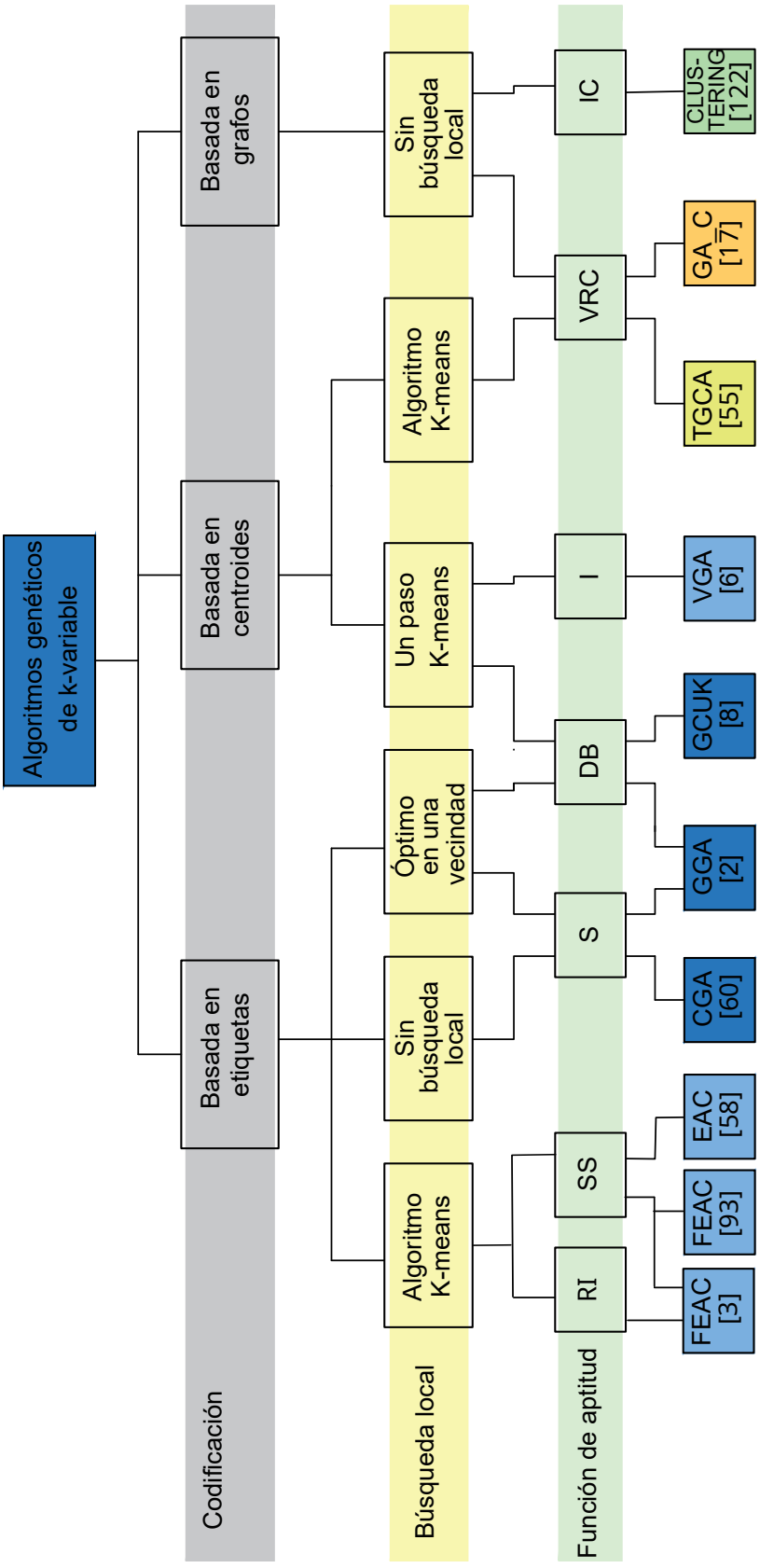


Figura 3.9 Taxonomía para los AGs basados en *k*-variable

Capítulo 4

Estudio experimental de los algoritmos genéticos para agrupación

En este capítulo se presenta un exhaustivo estudio experimental de los AGs que se han propuesto hasta la fecha. La finalidad es analizar el rendimiento de las diferentes propuestas para resolver el problema de agrupamiento que nos permita determinar las ventajas e inconvenientes de las diferentes propuestas. En los diferentes trabajos analizados, cada una de las propuestas es comparada con un número limitado de conjuntos de datos, CVIs y otras propuestas. Este estudio experimental pretende analizar todas las propuestas en un mismo entorno que nos permita obtener una valoración real de las diferentes propuestas para resolver el problema de agrupamiento. En el estudio se consideran las 22 propuestas descritas en el capítulo anterior, 22 CVIs y 94 conjuntos de datos.

En este capítulo se dará en primer lugar una descripción de la configuración y del entorno en el que se realiza la experimentación. A continuación, se presenta un análisis y discusión de los resultados encontrados en la experimentación de los algoritmos de agrupación estudiados. Debido a las diferencias en la especificación de los AGs, los cuales se reflejaron en la taxonomía descrita en la sección 3.3, el estudio experimental se ha dividido en dos partes. Por un lado, se evaluaron y analizaron las propuestas de k-fija y por otro lado, las propuestas de k-variable.

4.1 Especificación de la experimentación

En esta sección se describen los conjuntos de datos de entrada y las configuraciones usadas en el estudio experimental que se ha realizado de los AGs.

4.1.1 Conjunto de datos

En este estudio experimental se han considerado 94 conjuntos de datos, los cuales están disponibles en los repositorios de UCI[1] y KEEL[2].

Los conjuntos de datos considerados tanto los artificiales como reales, sus principales características están descritas en la Tabla 4.1. Los conjuntos de datos artificiales son indicados mediante el símbolo ($\oplus$). La Tabla 4.1 muestra el número de instancias en la *columna #Ex*, el número y el tipo de atributos se indica en la columna *#Att* (R reales, I enteros y N nominales), el número de clases se encuentra en la columna *#Cl* y la información de si el conjunto de datos contiene instancias con valores perdidos está en la columna *#MissVal*. Como se puede ver, se han seleccionado conjuntos de datos teniendo en cuenta varios criterios, de tal manera que permitan evaluar el desempeño de los algoritmos en varios escenarios. Como se sabe el número de instancias n está relacionada con la complejidad de los algoritmos, el rango de n está entre 80 a 100000. El número de atributos l está en los rango de 2 a 292 y el número de clases en los rango de 2 a 100 que va a estar posiblemente relacionada con el número de grupos k y nos va a permitir evaluar los CVIs que parten de esta información. Cuando el conjunto de datos tiene instancias con valores perdidos, estos valores son borradas y el número obtenido por este proceso está indicado en la columna *#Ex*.

En los algoritmos de agrupación se requiere definir una *norma*, la más común es la euclidiana, para asegurar una evaluación apropiada de la distancia entre las instancias. Por tanto, los conjuntos de datos deben ser preparados considerando el tipo de dato de cada atributo y aplicando una transformación para que pueda ser aplicada la medida de distancia, en este caso todos los datos se transformarán a valores reales, $\mathbb{R}$:

- Atributos reales $\mathbb{R}$. Si los atributos de las instancias están en $\mathbb{R}$, no es necesario aplicar ninguna transformación.
- Atributos binarios. Los que constan de solo dos valores, un caso particular de valores nominales (por ejemplo, falso o verdadero) son transformados a dígitos binarios 0 y 1.
- Nominales. Para los atributos nominales de v valores se aplica la transformación a v atributos binarios. Concretamente para las categorías posibles v del atributo se obtendrán v dimensiones. La desventaja es que aumentan las dimensiones de la instancia.

El paso final en la preparación de los conjuntos de datos es normalizar los valores, para que la medida de distancia no se vea alterada por diferentes rangos de valores en los diferentes atributos. De acuerdo con el estudio [89], los métodos que propone son los métodos de

[1] UCI (UC Irvine Machine Learning Repository), https://archive.ics.uci.edu/ml/index.php
[2] KEEL (Knowledge Extraction based on Evolutionary Learning) https://sci2s.ugr.es/keel/datasets.php

normalización Z_4 y Z_5 porque permiten descubrir estructuras subyacentes en los conjuntos de datos. Por lo tanto, para este estudio se ha determinado utilizar la normalización de Z_5.

Con los atributos en el espacio euclidiano $\mathbb{R}^n$, las instancias pueden ser vistas como un punto (n-tupla) y la distancia está definida para dos instancias. El total del número de atributos después de la conversión se indica en la columna *#Att* de la Tabla 4.1.

Tabla 4.1 Descripción de los conjuntos de datos

NOMBRE	#ATT (R/I/N)	#EX	#CL	MISS VAL.	NOMBRE	#ATT (R/I/N)	#EX	#CL	MISS VAL.
A1 ⊕	2 (0/2/0)	3000	20	No	Haberman	3 (0/3/0)	306	2	No
A2 ⊕	2 (0/2/0)	5250	35	No	Hayes-roth	4 (0/4/0)	160	3	No
A3 ⊕	2 (0/2/0)	7500	50	No	Hear	13 (1/12/0)	270	2	No
Aggregation ⊕	2 (2/0/0)	788	7	No	Hepatitis	19 (2/17/0)	155(80)	2	Yes
Birch1 ⊕	2 (0/2/0)	100000	100	No	Housevotes	16(15) (0/0/16)	435(232)	2	Yes
Birch2 ⊕	2 (0/2/0)	100000	100	No	Ionosphere	33 (32/1/0)	351	2	No
Compound ⊕	2 (2/0/0)	399	6	No	Iris	4 (4/0/0)	150	3	No
D31 ⊕	2 (2/0/0)	3100	31	No	Leaves plant margin	64 (64/0/0)	1600	100	No
Dim2 ⊕	2 (0/2/0)	1351	9	No	Leaves plant shape	64 (64/0/0)	1600	100	No
Dim8 ⊕	8 (0/8/0)	5401	9	No	Leaves plant texture	64 (64/0/0)	1599	100	No
Dim32 ⊕	32 (0/32/0)	1024	16	No	LED display domain	7 (0/0/7)	500	10	No
Dim64 ⊕	64 (0/64/0)	1024	16	No	Movement Libras	90 (90/0/0)	360	15	No
Dim128 ⊕	128 (0/128/0)	1024	16	No	Magic	10 (10/0/0)	19020	2	No
Flame ⊕	2 (2/0/0)	240	2	No	Mammographic	5 (0/5/0)	961(830)	2	Yes
G2-8-20 ⊕	8 (0/8/0)	2048	2	No	Marketing	13 (0/13/0)	8993(6876)	9	Yes
G2-8-80 ⊕	8 (0/8/0)	2048	2	No	Monk's	6(17) (0/0/6)	432	2	No
G2-32-20 ⊕	32 (0/32/0)	2048	2	No	Mushroom	22(116) (0/0/22)	8124(8124)	2	Yes
G2-32-80 ⊕	32 (0/32/0)	2048	2	No	New thyroid	5 (4/1/0)	215	3	No
G2-128-20 ⊕	128 (0/128/0)	2048	2	No	Optdigits	64 (0/64/0)	5620	10	No
G2-128-80 ⊕	128 (0/128/0)	2048	2	No	Page blocks	10 (4/6/0)	5473	5	No
Jain ⊕	2 (2/0/0)	373	2	No	Pen-based	16 (0/16/0)	10992	10	No
Madelon ⊕	500 (0/500/0)	2600	2(32)	No	Phoneme	5 (5/0/0)	5404	2	No
Path-based ⊕	2 (2/0/0)	300	3	No	Pima	8 (8/0/0)	768	2	No
R15 ⊕	2 (2/0/0)	600	15	No	Post-operative	8(23) (0/0/8)	90(87)	3	Yes
S1 ⊕	2 (0/2/0)	5000	15	No	Saheart	9(9) (5/3/1)	462	2	No
S2 ⊕	2 (0/2/0)	5000	15	No	SatImage	36 (0/36/0)	6435	7(6)	No
S3 ⊕	2 (0/2/0)	5000	15	No	Segment	19 (19/0/0)	2310	7	No
S4 ⊕	2 (0/2/0)	5000	15	No	Shuttle	9 (0/9/0)	58000	7	No
Spiral ⊕	2 (2/0/0)	312	3	No	Sonar	60 (60/0/0)	208	2	No
Unbalance ⊕	2 (0/2/0)	6500	8	No	Spambase	57 (57/0/0)	4601	2	No
Abalone	8(10) (7/0/1)	4177	28	No	SpectF heart	44 (0/44/0)	267	2	No
Appendicitis	7 (7/0/0)	106	2	No	Spect heart	22 (0/0/22)	267	2	No
Australian	14(38) (3/3/8)	690	2	No	Tae	5(54) (0/1/4)	151	3	No
Automobile	25(77) (15/0/10)	205(159)	5	Yes	Texture	40 (40/0/0)	5500	11	No
Balance	4(20) (0/0/4)	625	3	No	Thyroid	21 (6/15/0)	7200	3	No
Banana	2 (2/0/0)	5300	2	No	Tic-Tac-Toe	9(27) (0/0/9)	958	2	No
Bands	19 (13/6/0)	512(365)	2	Yes	Titanic	3 (3/0/0)	2201	2	No
Breast	9(38) (0/0/9)	286(277)	2	Yes	Twonorm	20 (20/0/0)	7400	2	No
Bupa	6 (1/5/0)	345	2	No	Vehicle	18 (0/18/0)	846	4	No
Cleveland	13 (13/0/0)	303(297)	5	Yes	Vowel	13 (10/3/0)	990	11	No
Coil 2000	85 (0/85/0)	9822	2	No	Wdbc	30 (30/0/0)	569	2	No
Contraceptive	9 (0/9/0)	1473	3	No	Winequality-red	11 (11/0/0)	1599	11(6)	No
Crx	15(42) (3/3/9)	690(653)	2	Yes	Winequality-white	11 (11/0/0)	4898	11(7)	No
Dermatology	34 (0/34/0)	366(358)	6	Yes	Wine	13 (13/0/0)	178	3	No
Ecoli	7 (7/0/0)	336	8	No	Wisconsin	9 (0/9/0)	699(683)	2	Yes
Flare	11(41) (0/0/11)	1389	6	No	Yeast	8 (8/0/0)	1484	10	No
Glass	9 (9/0/0)	214	7(6)	No	Zoo	16(21) (0/0/16)	101	7	No

4.1.2 Métricas de validación de agrupación

En la literatura, se han definido un gran número de índices de validación para caracterizar la "bondad" de la agrupación. Estos índices son conocidos como CVIs, tal y como se comentó en la descripción del componente de función aptitud de los algoritmos genéticos en la sección 3.1.2.

En términos generales, existen dos enfoques para analizar la validez de la agrupación [120]:

- El primer enfoque está basado en *criterios internos*, los cuales son evaluados en términos de características estructurales propias del conjunto de datos y las agrupaciones que se han realizado.
- El segundo enfoque se basa en *criterios externos*, esto implica que se evalúan los resultados del algoritmo de agrupación de acuerdo con la estructura/partición predefinida, la cual es impuesta al conjunto de datos y refleja la agrupación real del conjunto de datos.

En el caso del problema de agrupación no se puede considerar un único CVI como el más relevante, todos ellos aportan cualidades interesantes y se ven afectados por las diferentes características de los conjuntos de datos. Debido a la diversidad de criterios estructurales, validar una agrupación es un problema complejo, ya que para el mismo conjunto de datos existen diferentes particiones dependiendo de los diferentes criterios o el nivel de granularidad que se quiera obtener [123]. Se considera así, la evaluación, un tema de investigación abierto hasta la fecha con muchos estudios que han intentado analizar la [4, 14, 72, 75, 82, 108, 111, 114, 136].

Con respecto a los CVIs internos, se les puede considerar las métricas esenciales del problema de agrupación, ya que en muchas ocasiones se desconoce la agrupación real de las instancias y por tanto, son las únicas medidas que pueden aplicarse. Muchos CVIs han sido desarrollados desde hace décadas hasta el presente. Cada CVI se centra en evaluar alguna propiedad o una combinación de propiedades de los grupos que se han formado, entre las que se pueden mencionar:

- *Compactibilidad*. Evalúan semejanzas entre las instancias de un grupo, es decir, lo cerca que están las instancias dentro de un grupo. Una varianza pequeña indica un grupo compacto, esta propiedad se puede estimar de diferentes maneras como: la distancia máxima o promedio entre pares de instancias, o también la distancia máxima o promedio entre el centro del grupo y las instancias que pertenecen a éste [75].
- *Separación*. Evalúan qué tan distintas son las instancias o lo separado que un grupo está con respecto a los otros grupos. Se puede estimar con las distancias por pares entre los centros de los grupos o las distancias mínimas por pares entre objetos en diferentes grupos.

- *Conectividad*. Basado en la idea de que los elementos de datos vecinos deben compartir el mismo grupo [54].
- *Densidad*. Además de las propiedades anteriores, algunos CVIs se basan en la densidad de las instancias para determinar las que son más semejantes [75].

En este trabajo se propone una clasificación de CVIs internos basada en algunos estudios previos [72, 90]. En este estudio se consideró 15 medidas que serán analizadas en los diferentes AGs. La clasificación propuesta se basa en las diferentes propiedades comentadas:

1. *Compactibilidad*. En esta clase se encuentran los CVIs que miden sólo la *compactibilidad*. En general son funciones monótonas decrecientes dado que al aumentar el número de grupos k [20], la evaluación de compactibilidad disminuye haciéndola mejor. Por tanto, los CVIs de esta clase sólo son apropiadas para los AGs de k-fija, aunque también pueden ser usados en los AGs de k-variable cuando el número de grupos está establecido, como en el caso de TGCA [55] donde se utiliza SSE (3) en una segunda fase del algoritmo. Ejemplos de CVIs de esta categoría son: SED [92, 84, 7], SSE [18], J_1 [12, 71] y DD [38, 69].
2. *Compactibilidad* y *separación*. En esta categoría los CVIs estiman la relación entre *compactibilidad* y *separación*. Por sus características, en este trabajo son clasificadas en *clásicos* y *estadísticos*:
   - *Clásicos*. Son los CVIs más difundidos en la literatura y utilizados en los AGs, los cuales se pueden ver en la Tabla 4.2. Cuando se diseña un algoritmo en la selección de un CVI se debe considerar la codificación de las soluciones porque está relacionada con la complejidad computacional. Esta categoría es dividida en dos subcategorías [90, 61] y actualizada aquí a tres, considerando las características computacionales de los CVIs:
     - De centroide a instancias. Esta subcategoría es dividida en dos:
       - Fijos al centroide de todo el conjunto de datos (M). Los que utilizan M directa o indirectamente en la ecuación del índice. A M también se le conoce como E_1 (22). Ejemplos, índice I (21) y VRC (19).
       - No fijado al centroide del conjunto de datos M. Por ejemplo, el índice DB (11) y el índice XB (18).
     - Entre pares de instancias. La más representativa de este grupo es Silhouette (13), aunque también se encuentra el índice Dunn (16).

  - Híbridos. Aquí se encuentran los índices que mantienen parcialmente las propiedades de las dos categorías anteriores. Por ejemplo, el Simplified Silhouette (14), se define con base al índice Silhouette (13) y se calcula de centroide a instancia.

- *CVIs estadísticos*. Consideran una combinación de medidas, por ejemplo, el índice de validez *SD* [51] y el índice S_Dbw index [50]. Algunos más recientes basados en la teoría del aprendizaje estadístico, sería el índice VB [130].

3. *Estructuras especiales*. Los CVIs descritos arriba son sensibles a formas arbitrarias de grupos, subgrupos y valores atípicos en diferentes grados. Para CVIs de esta clase, la medición de la compactibilidad de los grupos no es obvia, tal como CVNN [76], RTI [110], SVDD [72] y CVDD [61].

Junto a los criterios estructurales principales descritos anteriormente, existen otras propiedades intrínsecas que también pueden estar presentes en los conjuntos de datos, las cuales se llamarán en este trabajo criterios estructurales secundarios, para los cuales los CVIs deben ser diseñados y junto con las estrategias de los AGs resolver de mejor manera la agrupación de las instancias. Por mencionar algunos criterios: *superposición* de grupos [39], *simetría* y *asimetría* que tanto el centroide estará localizado en el centro de la geometría del grupo [111] relacionada con *distribuciones sesgadas* [108], conjuntos de datos con *alta dimensionalidad* esta característica generalmente afecta la complejidad de los algoritmos, *ruido*, *instancias aisladas*[3], particiones *no balanceadas* tanto de diferentes tamaños y de densidades [107] y *formas arbitrarias* de los grupos [72] relacionada a la propiedad de conectividad.

Por otro parte, los CVIs externos nos permiten evaluar la coincidencia entre los grupos generados y los grupos reales (siempre y cuando estos estén disponibles). También, se tiene disponibles una amplia variedad de CVIs externos, en nuestro estudio se consideró 7. Las diferentes medidas se basan en dos propiedades principalmente [107]:

- *Conteo de pares.* En esta categoría los CVIs miden los pares de objetos del conjunto de datos en los que dos particiones diferentes coinciden o no. Las medidas incluidas son: el índice de Rand (27) y el índice de Jaccard (29).

- *Coincidencia de grupos.* En este caso se considera la coincidencia de grupos completos. Se puede realizar teniendo en cuenta las particiones a nivel de puntos: F-measure (28), pureza (25), precisión (31) y recall (30). O bien, teniendo en cuenta las particiones a nivel de grupo, donde se incluye el índice del centroide (26).

[3] En inglés se les llama outlier

Tabla 4.2 Índices de validación de agrupamiento (CVIs), el símbolo de ↑ significa que el CVI debe maximizarse y ↓ minimizarse

Tipo		CVI	Ecuación	
Internos	Compacticidad	Suma de distancias Euclidianas (SED) ↓, usada en [92, 84, 7, 81, 116]	SED $= \sum_{C_j}\sum_{x_i\in C_j}\|x_i-\mu_j\|$ (1), donde μ_j es el centro del j-ésimo grupo C_j. Para el objeto más representativo (v_j) SED $= \sum_{C_j}\sum_{x_i\in C_j}\|x_i-v_j\|$ (2),	
		Suma del error cuadrático (SSE) ↓, también llamado variación total dentro del grupo (TWCV) usada en: [18, 70]	SSE $= \sum_{C_j}\sum_{x_i\in C_j}(x_i-\mu_j)^T(x_i-\mu_j) = \sum_{C_j}\sum_{x_i\in C_j}\|x_i-\mu_j\|^2$ (3) TWCV $= \sum_{j=1}^{k}\sum_{i=1}^{n}u_{ij}\sum_{l=1}^{d}(x_{il}-\mu_{jl})^2$ (4)	
		Distancia distorsionada (DD) ↓, es suavizada (3) con n y d, usada en [38, 69]	DD $= \sum_{C_j}\sum_{x_i\in C_j}\|x_i-\mu_j\|^2/(n\cdot d)$ (5), donde n y d son los números de instancias y la dimensión del conjunto de datos respectivamente	
		Error cuadrado mínimo funcional (J_m) ↓ [13]. J_1 es un caso particular, usado en [12, 71]	$J_m(U,\mu) = \sum_{i=1}^{n}\sum_{j=1}^{k}u_{ji}^m\|(x_i-\mu_i\|_A^2$ (6), donde $U\in Mfc$ agrupamiento difuso (2.9) y $\|\,\|_A$ es la norma inducida A calculada con (2.14). Para J_1 (7), $U\in Mh$ (2.16)	
		Entropía (H_c) ↓ [13], es grado de desorden, definida solo para agrupamiento difuso	$H_c(U) = -\sum_{i=1}^{n}\sum_{j=1}^{k}(u_{ji}\log_a(u_{ji}))/n$ (8), donde U agrupamiento difuso (2.9), puede ser obtenida de una partición dura con (2.13)	
		Partition coefficient (F_c) ↑ [13]	$F_c(U) = \sum_{i=1}^{n}\sum_{j=1}^{k}(u_{ji})^2/n$ (9)	
	Compactibilidad y separación	CS measure (CS)↓ [21], mide la relación entre la suma de dispersión dentro de un grupo y la separación entre grupos	$CS(k) = \frac{\frac{1}{k}\sum_{j=1}^{k}\left\{\frac{1}{\lvert C_j\rvert}\sum_{x_i\in C_j}\max_{x_{i'}\in C_j}\left\{D(x_j,x_{j'})\right\}\right\}}{\frac{1}{k}\sum_{j=1}^{k}\left\{\min_{1\le j'\le k,j\neq j'}\left\{D(\mu_j,\mu_{j'})\right\}\right\}}$ (10), en donde D es la distancia	
		Índice Davies-Bouldin o índice DB ↓ [24], es una relación entre la suma de la dispersión dentro del grupo y la separación entre grupos, usada en [8, 2]	DB $= \frac{1}{k}\sum_j^k R_{C_j,qt}$ (11), en donde $R_{C_j,qt} = \max_{j,j\neq j'}\{(S_{C_j,q}+S_{C'_j,q})/\|\mu_j-\mu'_j\|_t\}$, $\|\|_t$ es la distancia de orden de Minkowski t y la dispersión dentro de C_j es $S_{C_j,q} = \left(\frac{1}{\lvert C_j\rvert}\sum_{x_i\in C_j}\{\|x_i-\mu_j\|_2^q\}\right)^{1/q}$ (12)	
		Índice Silhouette (S) ↑ [65] proporciona un promedio desigualdad entre las instancias dentro de su grupo contra el grupo más cercano, usada en [60]	Por ejemplo $x_i\in C_j$ es $s(x_i) = (b(x_i)-a(x_i))/\max\{a(x_i),b(x_i)\}$, donde $a(x_i) = \sum_{x_{i'}\in C_j}D(x_i,x_{i'})/\lvert C_j\rvert$ es el promedio de disimilitud, y $b(x_i) = \min\{D(x_i,x_{i''})\,\lvert\, x_{i''}\in C_{j''}$ y $C_j\neq C_{j''}\}$. S($\{C_j\}$) $= \sum_i^n s(x_i)/n$ (13) y S $\in[-1,1]$	
		Simplified Silhouette (SS) ↑ [3] es similar a S (13), la diferencia es el uso de centroides para el cálculo de distancias, usada en [58, 3, 93]	Aquí $a(x_i) = \sum_{x_i\in C_j}D(x_i,\mu_j)/\lvert C_j\rvert$ y $b(x_i) = \min\{D(x_i,\mu_{j'})\lvert x_i\notin C_{j'}\}$, SS($\{C_j\}$) $= \sum_i^n s(x_i)/n$. (14).	
		Índice Dunn ↑ (DI) [28] determina la relación mínima entre la distancia inter-grupos y el grupo. También puede calcularse con los centroides llamado aquí índice S Dunn (SD) ↑ (15)	DI($\{C_j\}$) $= \min_{1\le j\le k}\left\{\min_{1\le j'\le k,j'\neq j}\left\{\frac{\delta(C_j,C_{j'})}{\max_{1\le j''\le k}\{\Delta(C_{j''})\}}\right\}\right\}$ (16), en donde $\delta(C_j,C_{j'}) = \min\left\{D(x_i,x_{i'})\,\lvert\, x_i\in C_j,x_{i'}\in C_{j'}\right\}$ y $\Delta(C_{j''}) = \max\left\{D(x_i,x_{i''})\,\lvert\, x_i,x_{i''}\in C_{j''}\right\}$ es el diámetro de $C_{j''}$	
		Score Function (SF) ↑ [113], es una función que combina dos términos, la distancia entre grupos y la distancia dentro del grupo	SF $= 1-1/e^{e^{bcd-wcd}}$ (17), en donde $bcd = (\sum_{j=1}^{k}\|\mu_j-M\|\cdot\lvert C_j\rvert)/(n\cdot k)$ y $wcd = \sum_{j=1}^{k}(1/\lvert C_j\rvert\sum_{x_i\in C_j}\|x_i-\mu_j\|)$, en donde M es el centroide de todos los x_i	
		Índice Xie-Beni (XB) ↓[127], varianza total entre la mínima separación de los grupos	XB $= \frac{\sum_{j=1}^{k}\sum_{x_i}^{n}u_{ji}^2\|x_i-\mu_j\|^2}{n\cdot(d_{min})^2}$ (18), donde $d_{min} = \min_{j,j'=1,j\neq j'}^{k}\left\|\mu_j-\mu_{j'}\right\|$	
		Criterio de relación de varianza (VRC) o índice CH ↑, mide la cohesión interna del grupo y el aislamiento de grupos externos , se usa en [17, 55]	VRC $= \frac{SS_B}{SS_W}\cdot\frac{(n-k)}{(k-1)}$ (19)	en donde $SS_B = \sum_{j=i}^{k}\lvert C_j\rvert\|\mu_j-M\|^2$ es la varianza general entre grupos y $SS_W = \sum_{j=1}^{k}\sum_{x_i\in C_j}\|x_i-\mu_j\|^2$ es la varianza general dentro del grupo
		Índice WB ↓ [135, 134], es una medida similar a VRC (19)	WB$(k) = k\cdot\frac{SS_W}{SS_B}$ (20)	
		Índice I o índice PBM ↑ [85, 96], consiste de tres factores $(1/k)$ decrementa grupos, para crear grupos compactos (E_1/E_k) y separación entre pares de grupos (D_k), usada en [6]	I$(k) = \left(\frac{1}{k}\cdot\frac{E_1}{E_k}\cdot D_k\right)^p$ (21), en donde $E_k = \sum_{j=1}^{k}\sum_{i=1}^{n}u_{ji}\|x_j-\mu_j\|$, $D_k = \max\left\{\|\mu_j,\mu_{j'}\|\,\lvert\, j,j'\in\{1,2,...,k\}\right.$ and $\left.j\neq j'\right\}$ y $E_1 = M$ (22), los autores proponen $p=2$	
		Intra- and inter-cluster distance (IC) ↓, aplicable solo en el AG CLUSTERING [122]	IC $= \sum_{i=1}^{k}D_{inter}(C_j)w - D_{intra}(C_j)$ (23), en donde w es un parámetro	
		Overlap ↓ [39], conteo de posible número de puntos que están más cerca a otro centroide que el propio	Overlap $= \frac{1}{n}\sum ov(d_1,d_2)$ (24), en donde $ov(d_1,d_2) = \begin{cases}1, \text{ if } d_1>d_2\\ 0, \text{ otherwise}\end{cases}$, d_1 distancia de cada punto a su centroide y d_2 distancia del punto al punto más cercano de otro grupo.	
Externos		Pureza ↑, pureza de un grupo dado con respecto a su categoría conocida [105]	Pureza$(\{R_s\},\{C_j\}) = \frac{1}{n}\sum_s\max_j\lvert R_s\cap C_j\rvert$ (25), Pureza $\in[0,1]$	
		Índice de grupos vacíos (CI$_e$) ↓, similar al índice de Centroide (CI) [42], hace una correspondencia entre grupos $\{R_s\}$ y $\{C_j\}$	Si $Cf_{\lvert R_s\rvert\times\lvert C_j\rvert}$ es una matriz de confusión, $m_j = \max_{1\le s\le\lvert R_s\rvert}Cf_{s\,j}$, $s_{1\le s\le\lvert R_s\rvert} = \sum_{1\le j\le\lvert C_j\rvert}m_j$ y Ep$(R_s) = \begin{cases}1, \text{ if } s_s=0\\ 0, \text{ otherwise}\end{cases}$ y por último CI$_e(\{R_s\},\{C_j\}) = \sum$Ep$(R_s)$ (26)	
		Índice Rand (Ω) [104] ↑ mide la similitud entre los grupos C_j y las clases R_s se usa en [3]	$\Omega(\{R_s\},\{C_j\}) = \frac{a+d}{a+b+c+d}$ (27), $\Omega\in[0,1]$	En donde los pares de objetos: a, pertenecen a la misma clase R_s y al mismo grupo C_j; b, pertenecen a la misma clase en R_s y a diferentes grupos C_j; c, en diferentes clases en R_s y al mismo grupo en C_j y d, están en diferentes clases R_s y en diferentes grupos C_j [3]
		F-measure (F$_m$) ↑ [37], mide la precisión y tratar de dar peso a corresponder número de grupos con el número de clases	$F_m(\{R_s\},\{C_j\}) = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$ (28), donde $F_m\in[0,1]$	
		Índice Jaccard (J) ↑, es similar a Ω mide la, similitud y diversidad dentro de los grupos	J$(\{R_s\},\{C_j\}) = \frac{a}{a+b+c}$ (29)	
		Recall ↑	Recall$(\{R_s\},\{C_j\}) = \frac{a}{a+b}$ (30)	
		Precision ↑	Precision$(\{R_s\},\{C_j\}) = \frac{a}{a+c}$ (31)	

Finalmente, para el estudio experimental fueron seleccionados 22 CVIs entre índices internos y externos. Estas medidas fueron seleccionadas por ser las más ampliamente mencionadas en la literatura. En la Tabla 4.2 se enumeran los diferentes CVIs utilizados para medir la validez de la agrupación en el estudios de los diferentes AGs, con excepción de la métrica CI (23), porque es solo aplicable en el algoritmo CLUSTERING [122].

4.1.3 Configuración de los AGs

Para este estudio se han considerado 22 AGs para la solución del problema de agrupación, 11 AGs de *k*-fija y 11 AGs de *k*-variable. Estos algoritmos son considerados el estado del arte en esta área. La descripción detallada de las características de estos algoritmos ha sido presentada en el capítulo 3. Concretamente, en las Tablas 3.1 y 3.2 se encuentra un resumen de las características de cada uno de los AGs. Todos los algoritmos han sido implementados en una misma librería, LEAC, la cual ha sido desarrollada en esta tesis [109] y está disponible en github[4] para poder facilitar la tarea del diseño y comparación de nuevas propuestas. La librería es descrita brevemente en el capítulo 7 de este trabajo.

Los parámetros de configuración usados por los AGs en este estudio experimental se muestran en la Tabla 4.3. Para los algoritmos con *k*-fija se ha considerado como valor de entrada *k* el número de clases de cada conjunto de datos. Este parámetro es conocido en todos los conjuntos de datos utilizados en este estudio experimental. Los valores de configuración de los AGs corresponden con los valores de configuración predeterminados por los autores de cada algoritmo, se incluyen en archivos de programación de tareas disponibles en la página web asociada[5] para poder repetir las pruebas de este estudio experimental utilizando la biblioteca LEAC.

4.1.4 Entorno de experimentación

Para la ejecución de los AGs se buscó un entorno lo más homogéneo posible para tener igualdad de condiciones en todos los experimentos realizados. Para el hardware se utilizaron 14 estaciones de trabajo HP Z840 con procesador Intel®Xeon®ES v3, memoria 32 GB DDR4-2133 ECC cada computadora y con el sistema operativo x86_64 GNU/Linux Debian Kernel 3.16.0-4. Los experimentos se realizaron en modo independiente[6], también se usó GNU Parallel[118] una herramienta para la distribución de tareas que permite ejecutar trabajos en paralelo con un archivo de automatización de tareas escritos en el intérprete comandos Bash.

[4] https://github.com/kdis-lab/leac

[5] https://www.uco.es/kdis/a-survey-of-evolutionary-algorithm-for-clustering-taxonomy-and-empirical-analysis/

[6] En inglés se le conoce como standalone

Tabla 4.3 Parámetros de configuración de los AGs

	Algoritmo Genético	Número de Generaciones	Tamaño de la Población	Probabilidad de cruce	Probabilidad de mutación
AGs con k-fija	GA [92]	10000	6	0.8	0.5
	GKA [70]	200	50	∅	0.05
	IGKA [80]	200	50	∅	0.05
	FGKA [79]	200	50	∅	0.05
	GAS [84]	100	100	0.8	0.01
	KGA [7]	1000	50	0.8	0.01
	GAGR [18]	50	50	Adaptativa	Adaptativa
	GA_B [12]	2000	200	1.0	1.0
	CBGA [38]	256	8	1.0 (Elitista)	0.01
	HKA [116]	200	200	0.95	0.05
	GCA [81]	200	200	0.9	0.1
AGs con k-variable	GA_C [17]	Max n	10xn	0.8	0.08
	GGA_S [2]	100	4x20	[0.4-0.8]	0.05
	GGA_{DB}[2]	100	4x20	[0.4-0.8]	0.05
	CGA [60]	200	20	0.5	0.25^a
	EAC [58]	500	20	∅	0.5^a
	$FEAC_{SS}$ [3]	500	20	∅	0.5^a
	$FEAC_{RI}$ [93]	500	20	∅	0.5^a
	VGA [6]	1000	50	0.8	0.05
	GCUK [8]	100	50	0.8	0.01
	TGCA [55]	200	60	0.8	Dos estados
	CLUST [122]	100	50	0.8	0.05

[a]Para cada operador de mutación considerado

Como método de validación de los resultados obtenidos, se empleó la validación cruzada 10-fold siguiendo el trabajo de Tarekegn et al. [119]. Los conjuntos de datos con las particiones específicas están disponibles en la web [7]. Se llevaron a cabo 20 ejecuciones por fold, realizando así un total de 200 ejecuciones por conjunto de datos. Como ya se ha comentado, de acuerdo al estudio de G.W. Milligan and M.C. Cooper [89], los datos fueron normalizados usando el procedimiento Z_5. Este estudio determinaba que tanto el procedimiento de normalizacón Z_4, como Z_5 ofrecían los mejores resultados en el problema de agrupamiento.

Las tablas promedios obtenidas en los resultados de test para cada CVI por cada algoritmo y conjunto de datos considerado, pueden ser consultadas en la página web asociada[7].

[7]https://www.uco.es/kdis/a-survey-of-evolutionary-algorithm-for-clustering-taxonomy-and-empirical-analysis/

4.2 Análisis de resultados experimentales de los AGs de *k*-fija

En este estudio se han analizado 11 AGs considerados el estado del arte para la solución del problema de agrupación basado en particiones para *k*-fija. Las principales características de cada algoritmo están descritas en la sección 3.2. Además, la configuración de cada uno de los AGs es descrita en la sección 4.1.3. La idea es hacer un estudio comparativo de todos los algoritmos para determinar si alguno de ellos tiene diferencias de desempeño significativas con respecto a los demás y obtener conclusiones relevantes acerca de las ventajas e inconvenientes de los diferentes AGs propuestos en la literatura. Con este propósito, los 11 algoritmos son analizados usando **22** CVIs y evaluando **94** conjuntos de datos descritos en la sección 4.1.1.

Para los AGs de *k*-fija el número de grupos debe ser especificado como parámetro de entrada, para ello se ha utilizado el número de clases de los conjuntos de datos. Sin embargo, para algunos conjuntos de datos por su estructura puede resultar que no coincide el número de clases con el número de grupos. No obstante, este hecho no debe afectar el objetivo de la experimentación, los AGs de *k*-fija obtendrán las particiones de tal manera que sean lo más compactas y se podrá medir su desempeño con los diferentes CVIs seleccionados.

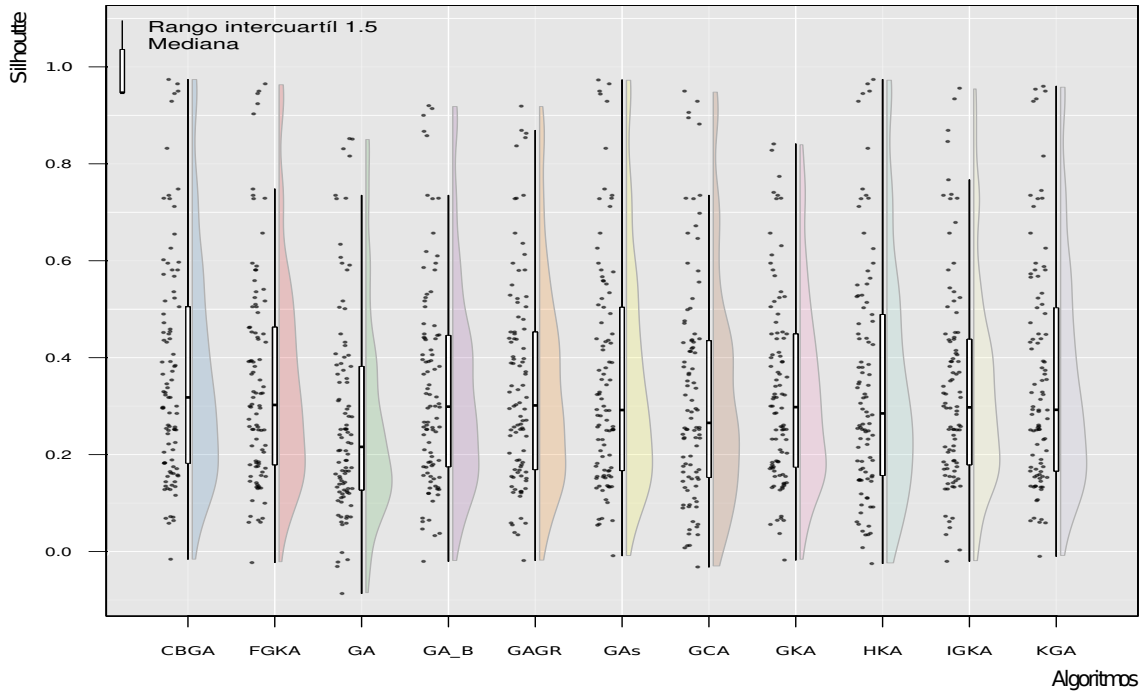
En esta sección se mostrará el estudio realizado y se discutirán los resultados. Concretamente, primero se llevará a cabo un análisis de los CVIs internos y después, se hará un estudio similar considerando los CVIs externos. Se analizá los resultados utilizando test estadísticos no paramétricos [26], para analizar el comportamiento de los AGs.

4.2.1 Análisis de los CVIs internos

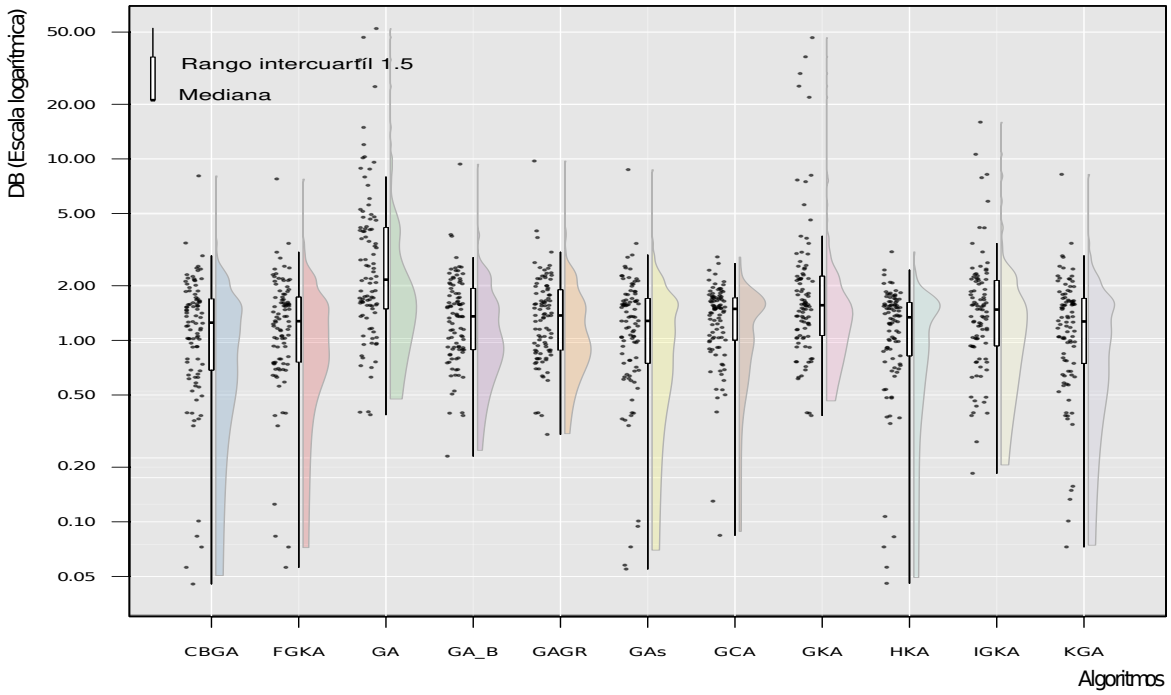
En esta sección, se llevó a cabo un estudio de los CVIs internos, al analizar los valores obtenidos por cada uno de los AGs. Se han considerado un total de 15 CVIs internos. Debido a la extensión de las tablas de resultados para cada CVI que considera 94 conjuntos de datos, se van a analizar dos de los CVIs más representativos: el índice de Silhoutte (13) y el índice de DB (11), los cuales son ampliamente utilizadas en la agrupación. No obstante, como ya se ha comentado, las tablas de promedios obtenidas y los resultados de los test para cada CVI por cada algoritmo y conjunto de datos se pueden consultar en la página web asociada[8].

El índice Silhoutte, es un promedio de disparidad entre los objetos, para un objeto x_i se denota por $s(x_i)$ y puede ser usado para estudiar la desigualdad del objeto x_i dentro de su grupo actual de pertenencia en referencia a su grupo más cercano. Es fácil verificar que el valor del rango de valores para un objeto x_i es $-1 \leq s(x_i) \leq 1$, así cuanto más próximo sea el valor $s(x_i)$ a 1 más adecuada es la pertenencia del objeto al grupo, si $s(x_i)$ es igual a cero, entonces no

[8]Consultar la página web asociada https://www.uco.es/kdis/a-survey-of-evolutionary-algorithm-for-clustering-taxonomy-and-empirical-analysis/

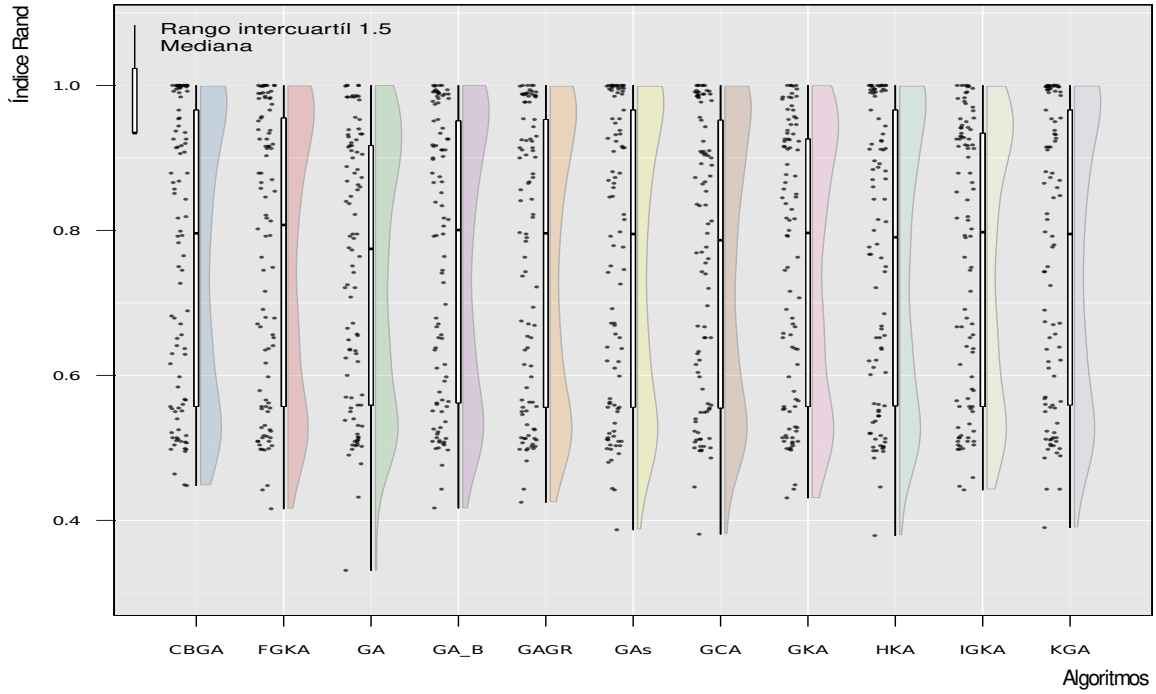


(a) Resultados del índice de Silhoutte (↑) por algoritmo y cojunto de datos

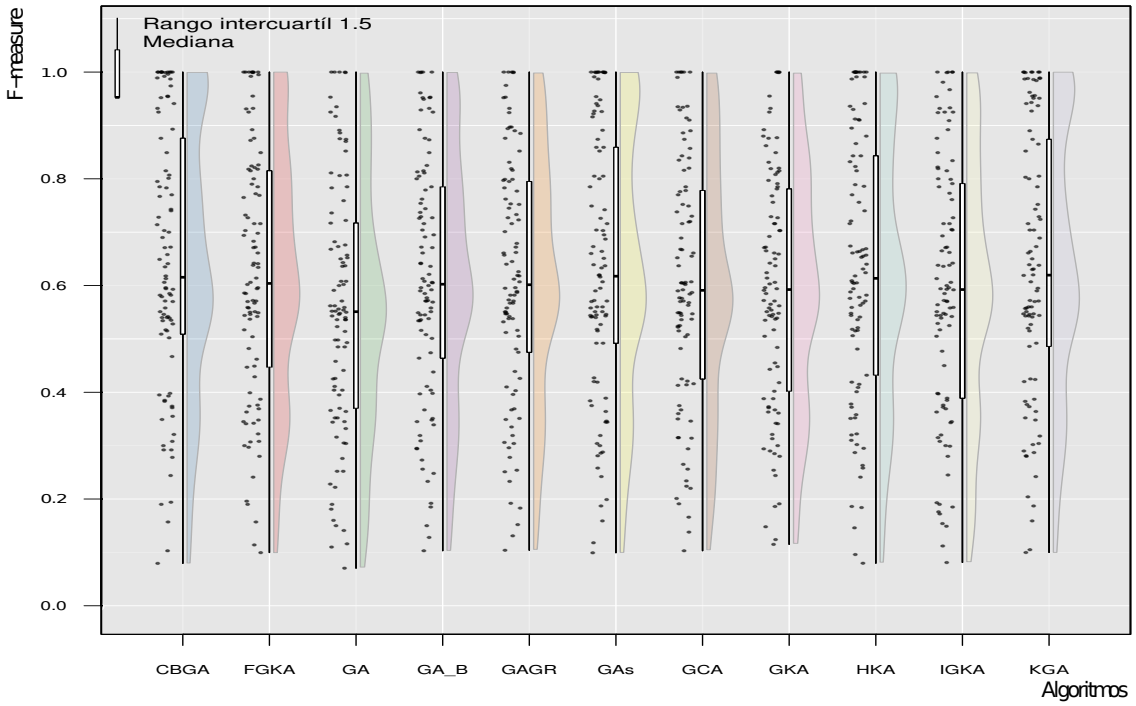


(b) Resultados del índice de DB (↓) por algoritmo y conjunto de datos

Figura 4.1 Resultados promedio de CVIs internos para AGs de *k*-fija



(a) Resultados del Índice de Rand (Ω) (↑) por algoritmo y conjunto de datos



(b) Resultados de F-measure (↑) por algoritmo y conjunto de datos

Figura 4.2 Resultados promedio de CVIs externos para AGs de *k*-fija

existe claridad si el objeto ha sido asignado a su grupo actual o puede pertenecer a un grupo vecino [3].

Los promedios obtenidos del índice Silhoutte experimentalmente se presentan de manera gráfica en la Figura 4.1a. A priori, se puede ver que los mejores valores son obtenidos por el algoritmo GBGA presentando el mejor promedio y rango intercuartílico. Ejemplos de mejores valores de este índice para conjuntos de datos reales son 0.655, 0.595 y 0.581 para Titanic, Wisconsin y Mammographic, respectivamente y los peores valores son 0.069, 0.0639 y 0.0584 para Leaves Plant Texture, Leaves Plant Margin y Post Operative respectivamente. Para conjuntos de datos artificiales, que por lo general son construidos con una estructura definida, los mejores valores son 0.929 a 0.974 para los conjuntos de datos Dim, 0.832 y 0.748 Unbalance y R15 respectivamente, los peores valores son 0.328 para Spiral, de 0.252 a 0.261 para G2 con 80% de sobre-posición entre los grupos y Madelon con 0.0159[8].

El índice DB [24] es la otra medida interna utilizada en este análisis preliminar. Es una función de la relación entre la suma de la dispersión dentro de los grupos y la separación entre grupos, por lo tanto, los grupos que están más separados y menos dispersos darán como resultado una mejor puntuación. Así, esta función debe ser minimizada. Los resultados promedio de los AGs son mostrados de la misma forma en la Figura 4.1b. Se puede ver que el algoritmo que presenta los mejores valores promedio y rango intercuartílico es nuevamente el algoritmo CBGA, con los mejores valores de DB para los conjuntos de datos reales: 0.625, 0.626 y 0.644 para Shuttle, Mammographic y Titanic, respectivamente. Los peores valores son 2.56, 2.925 y 3.439 para Monks, Balance y Tic-Tac-Toe, respectivamente. Si se analizan los conjuntos de datos artificiales los mejores valores son de 0.045 a 0.101 para Dim, 0.338 y 0.361 para R15 y Unbalance y los peores valores son 1.141 para Flame, de 1.483 a 1.589 para los conjuntos G2 con 80% de sobre-posición entre los grupos y 8.063 para Madelon[8].

Debido a la extensión de las tablas de resultados que considera un elevado número de conjuntos de datos, CVIs y propuestas; para realizar el exhaustivo estudio se van a resumir los resultados finales utilizando un análisis de meta-rangos[9] de algoritmos [16], el cual tiene como objetivo evaluar cuál algoritmo tiene un mejor rendimiento en general considerando todos los conjuntos de datos y CVIs.

En el análisis de meta-rangos lo primero es asignar un rango por CVI analizado a cada AG. Estos rangos son obtenidos de forma similar al procedimiento que se lleva a cabo en el test de Friedman [26]. De esta forma, el algoritmo con mejor valor para un CVI para el primer conjunto de datos recibe un rango de 1, el algoritmo con el siguiente mejor valor para ese CVI obtiene un valor de 2, y así sucesivamente, se aplica el mismo procedimiento para el segundo conjunto de datos y así hasta que se tengan todos los rangos de cada algoritmo para

[9]En inglés meta-ranking

cada conjunto de datos. Estos rangos nos permiten saber qué algoritmo obtiene los mejores resultados considerando todos los conjuntos de datos por cada CVI. De esta forma, el algoritmo con el valor más cercano a 1 indica mejor rendimiento en la mayoría de los conjuntos de datos. Si el valor fuese 1, indicaría que ha sido el que ha obtenido el mejor valor para ese CVI en todos los conjuntos de datos. La Tabla 4.4 muestra los rangos medios de cada algoritmo para cada CVI.

Tabla 4.4 Rangos medios de AGs de *k*-fija para CVIs internos utilizando todos los conjuntos de datos

CVI	CBGA	FGKA	GA	GAGR	GA_B	GAs	GCA	GKA	HKA	IGKA	KGA
CS measure (CS)	5.628	5.367	6.426	6.899	6.862	**5.202**	7.117	5.894	5.803	5.553	5.250
Índice Davies-Bouldin (DB)	**3.532**	4.872	9.585	8.223	6.83	5.447	6.250	6.298	4.755	5.287	4.920
Índice Xie Beni (XB)	**3.495**	4.787	9.277	7.941	6.920	5.718	6.473	5.872	4.894	5.255	5.367
Índice S Dunn (SD)	**3.516**	4.505	9.457	7.830	6.495	5.872	6.846	5.367	5.766	4.803	5.543
Índice Dunn index (D)	**4.537**	5.543	7.686	6.191	5.984	5.245	7.633	6.027	6.277	5.612	5.266
Índice I (I)	**4.032**	5.165	9.378	8.074	6.649	5.394	5.941	6.351	4.293	5.569	5.154
Score function (SF)	4.372	5.112	8.191	7.559	6.457	3.793	8.644	6.112	6.798	5.415	**3.548**
Silhouette (S)	**3.793**	4.564	8.617	6.617	6.298	5.122	8.691	5.351	7.053	4.814	5.080
Simplified Silhouette (SS)	**3.920**	4.973	8.899	7.904	6.559	4.596	7.548	6.170	5.697	5.505	4.229
C. de relación de varianza (VRC)	**3.628**	5.239	9.468	8.191	6.750	6.261	5.447	6.048	3.920	5.309	5.739
Índice WB (WB)	3.968	4.723	9.473	8.362	6.697	6.197	5.441	6.543	**3.856**	5.314	5.426
Suma de dist Euclidianas (SED)	4.160	4.564	7.904	7.702	6.160	3.495	9.500	6.181	7.628	5.527	**3.181**
Dist. distorsionada (DD)	**3.287**	4.032	8.282	6.585	5.750	4.665	9.947	5.713	8.420	4.888	4.431
S. del error cuadrático (SSE)	**3.223**	3.676	8.340	6.840	5.830	4.973	9.968	5.468	8.426	4.633	4.622
Error cuadrado mín func (Jm)	5.117	4.941	5.324	5.973	5.596	4.521	9.755	6.309	8.085	5.926	**4.452**
Rango de Friedman	1.800	3.000	9.933	9.200	7.733	4.333	9.200	6.867	6.133	4.600	3.200

A priori, se puede observar que CBGA obtiene los rangos más bajos en la mayoría de las métricas. Después de haber obtenidos los rangos, los cuales determinan el grado de desempeño del AG en cada CVI, sobre estos valores se aplicará el test de Friedman que nos permitirá realizar un análisis de meta-rangos. Esta metodología ha sido aplicada en otros estudios [16], la cual consiste en aplicar el test de Friedman sobre estos valores de rangos promedio. Se construye así un meta-ranking de AGs siguiendo la filosofía del test de Friedman descrita anteriormente. De esta forma, se evalúa qué algoritmos tienen el mejor rendimiento general en la mayoría de las métricas, ya que no existe un método que funcione mejor para todas las métricas. En este caso, el algoritmo con el mejor valor en una métrica tiene un rango de 1 para esa métrica, el algoritmo con el siguiente mejor valor tiene un rango de 2 y así sucesivamente. Sobre estos valores de rangos promedios para las medidas internas, se aplica el test de Friedman. El rango asignado por el test de Friedman se especifica en la última fila de la Tabla 4.4 y muestra que CBGA obtiene el rango más bajo (1.800). Esto indica que es la mejor propuesta para la mayoría de las medidas. Un valor de 1 indicaría que es el mejor para todas las medidas, el tener un valor tan cercano a 1 nos indica que obtiene los mejores valores en la mayoría de las métricas.

Tabla 4.5 Test de Friedman y corrección de Iman Davenport sobre los rangos medios de AGs de *k*-fija para CVIs internos

CVIs	Test estadístico	df	p-valor
Internos	Test de Friedman's χ^2 = 107.65	df = 10	<2.2e-16
	Corrección de Iman Davenport χ^2 = 35.585	df1 = 10, df2 = 140	<2.2e-16

Tabla 4.6 *p*-valores de los rangos promedio de CVIs internos de los AGs de *k*-fija, obtenidos con el test de Friedman post-hoc corrección Shaffer para un intervalo de confianza $> 95\%$

	CBGA	FGKA	GA	GAGR	GA_B	GAs	GCA	GKA	HKA	IGKA	KGA
CBGA	n/a	0.365	**0.000**	**0.000**	**0.000**	0.062	**0.000**	**0.000**	**0.001**	0.037	0.299
FGKA	0.365	n/a	**0.000**	**0.000**	**0.000**	0.315	**0.000**	**0.004**	0.021	0.252	0.874
GA	**0.000**	**0.000**	n/a	0.579	0.101	**0.000**	0.579	0.023	**0.004**	**0.000**	**0.000**
GAGR	**0.000**	**0.000**	0.579	n/a	0.285	**0.000**	1.000	0.086	0.023	**0.000**	**0.000**
GA_B	**0.000**	**0.000**	0.101	0.285	n/a	0.011	0.285	0.514	0.252	0.021	**0.001**
GAs	0.062	0.315	**0.000**	**0.000**	0.011	n/a	**0.000**	0.062	0.192	0.837	0.389
GCA	**0.000**	**0.000**	0.579	1.000	0.285	**0.000**	n/a	0.086	0.023	**0.000**	**0.000**
GKA	**0.000**	**0.004**	0.023	0.086	0.514	0.062	0.086	n/a	0.579	0.092	**0.006**
HKA	**0.001**	0.021	**0.004**	0.023	0.252	0.192	0.023	0.579	n/a	0.265	0.028
IGKA	0.037	0.252	**0.000**	**0.000**	0.021	0.837	**0.000**	0.092	0.265	n/a	0.299
KGA	0.299	0.874	**0.000**	**0.000**	**0.001**	0.389	**0.000**	**0.006**	0.028	0.299	n/a

El estadístico de Friedman y sus *p*-valores se muestran en la tabla 4.5. El test de Friedman rechaza la hipótesis nula y por lo tanto determina que existen diferencias significativas en el rendimiento de los algoritmos al 95% de confianza. A continuación, se realiza el test a posteriori de Shaffer para que nos determine entre qué algoritmos hay diferencias significativas. La Tabla 4.6 muestra los *p*-valores de este test para los AGs de *k*-fija y CVIs internos. Los valores marcados en negrita indican un intervalo de confianza $> 95\%$, con la cual es posible identificar el AG o AGs con diferencias significativas con respecto al resto.

Para una interpretación más representativa de esta información. Se muestra la Figura 4.3 donde se indican los rangos asociados con cada uno de los AGs con un nivel de confianza $> 95\%$. Estos resultados confirman que el algoritmo CBGA funciona significativamente mejor que otras propuestas. Es el el algoritmo de control para todas las métricas consideradas. Las diferentes líneas horizontales indican la equivalencia de algoritmos considerando las diferentes *distancias críticas* que permite establecer entre qué algoritmos, aunque obtengan peores resultados no se aprecian diferencias significas. En este estudio, los algoritmos FGKA, KGA, IGKA y GAs que están unidos con CBGA por la línea de la distancia crítica indica que no existen diferencias significativas en sus resultados. El resto de las propuestas, sí puede considerarse que obtienen peores resultados estadísticamente. Es interesante resaltar que los AGs que utilizan una codificación basada en las instancias más representativa como el caso de HKA y GCA, se encuentran en mitad con respecto a sus valores de rango, siendo HKA superior, seguramente debido a su inclusión del operador de búsqueda local. Además, estas propuestas

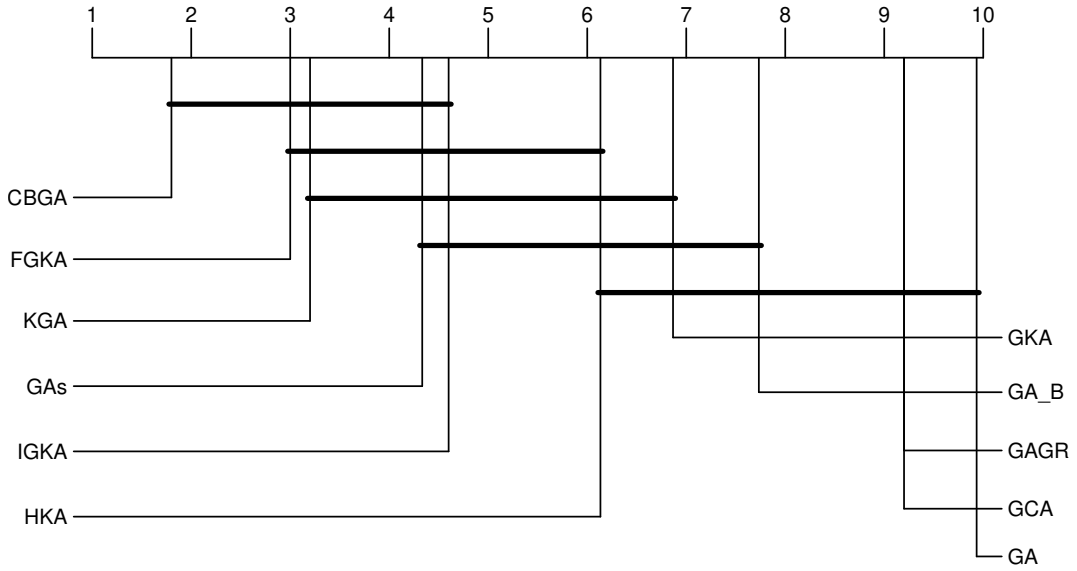


Figura 4.3 Distancia crítica del test a posteriori de Shaffer para los AGs de k-fija considerando los CVIs internos ($\alpha = 0.05$)

obtienen peores resultados que las que emplean una codificación basada en centroides. Los peores resultados son obtenidos por algoritmos que utilizan una codificación binaria. También, los AGs que emplean algún tipo de búsqueda local en su estrategia evolutiva obtienen mejores resultados que los que no lo incluyen. Finalmente, algunas medidas de evaluación de la función de aptitud que funcionan mejor frente a otras son DD, SED o TWCV.

4.2.2 Análisis de los CVIs externos

Las medidas externas evalúan el modelo obtenido por los algoritmos contra clases ya establecidas que son conocidas con anticipación. En este caso, cada objeto del conjunto de datos tiene una etiqueta generalmente obtenida por un experto que cataloga cada objeto perteneciente a una clase concreta, la cual es comparada con la identificación del grupo que pertenece. En esta sección se lleva a cabo un análisis preliminar de los CVIs externos. Debido a la extensión de las tablas de resultados para cada CVI que considera 94 conjuntos de datos, se van a analizar dos de los CVIs más representativos: el índice Rand (27) y F-measure (28), los cuales son ampliamente utilizados en la agrupación. No obstante, como ya se ha comentado, las tablas con los promedios obtenidos para cada CVI por cada algoritmo y conjunto de datos considerados, y los resultados de los tests pueden ser consultadas en la página web asociada [8].

El índice de Rand definido en [62] se representa por la letra Ω. Este índice mide la similitud entre los conjuntos de clases R_s y grupos C_j, por medio de un cociente entre pares de objetos

consistentes que pertenecen a la misma clase y al mismo grupo y pares de objetos en diferentes clases y diferentes grupos. En su cálculo se incluyen todos los pares de ambos conjuntos donde se incluyen los pares inconsistentes de objetos en la misma clase y diferentes grupos y pares de objetos en el mismo grupo y diferentes clases. Proporciona una medida de calidad de una agrupación como un todo [3]. Los valores del índice de Rand $\in [0,1]$ y cuanto más próximo sea el valor a 1, indica que es una partición más consistente. En la Figura 4.2a se muestran los resultados promedio de los AGs del índice de Rand.

Una vez más se puede observar que los mejores valores promedio y rango intercuartílico los obtiene el algoritmo CBGA, aunque en este caso, los resultados de diferentes algoritmos son también muy similares a los obtenidos por CBGA. Como ejemplo de valores obtenidos del índice de Rand, los mejores están presentes en los conjuntos de datos reales, con 0.992, 0.990, y 0.986 en Leaves Plant Margin, Leaves Plant Texture, y Leaves Plant Shape, respectivamente. También, cabe resaltar el conjunto de datos Twonorm con 0.955. Los conjuntos de datos con un valor pequeño son Post Operative, Thyroid, Page-Blocks con valores de 0.486, 0.449 y 0.459, respectivamente. Para los conjuntos de datos artificiales muchos de ellos logran el máximo valor: A3, Birch1, Dim, G2 y Unbalance. Los peores resultados se obtienen en los conjuntos de datos Flame, Spiral y Madelon con valores de 0.728, 0.566 y 0.509, respectivamente[8].

La F-measure (F_m) es definida en [37], es otra medida externa, esta mide la correspondencia entre la clase y el grupo de las instancias, al tratar de unificar el número de grupos con el número de clases. F-measure $\in [0,1]$ y valores más grandes indican una mayor calidad de agrupamiento. En la Figura 4.2b se muestra los resultados de esta métrica obtenidos por los diferentes AGs. Nuevamente, se puede ver que los mejores valores de los promedios y el rango intercuartílico se obtienen con el algoritmo CBGA. Aunque diferentes algoritmos presentan un comportamiento similar. Para un número representativo de conjuntos de datos, cuando un valor cercano a 1 que muestra que los grupos C_j están relacionados con las clases reales R_s. Específicamente, los conjuntos de datos con mejores valores son Twonorm, Wisconsin y Wine con F_m de 0.955, 0.937 y 0.893 respectivamente, y los peores valores son para Marketing, Vowel y Abalone con F_m de 0.222, 0.159 y 0.186. Para los conjuntos de datos artificiales, se logró una $F_m = 1$ Unbalance, G2 y Dim, y los peores valores para Compound, Spiral y Madelon con 0.641, 0.322 y 0.220, respectivamente, de forma particular los diferentes valores pueden ser consultados en la página web asociada [8].

Para realizar un exhaustivo estudio, se van a resumir los resultados finales utilizando un análisis de meta-rangos [16], similar al que se ha comentado en la sección anterior con los CVIs internos.

En el análisis de meta-rangos lo primero es asignar un rango por CVI analizado a cada AG. El procedimiento para obtenerlo es similar al que se ha comentado en la sección anterior

con los CVIs internos. Recordando que estos rangos permiten saber qué algoritmo obtiene los mejores resultados considerando todos los conjuntos de datos por cada CVI. De esta forma, el algoritmo con el valor más cercano a 1 indica mejor rendimiento en la mayoría de los conjuntos de datos. La Tabla 4.7 muestra los rangos medios de cada algoritmo para cada CVI externo que se ha considerado.

Tabla 4.7 Rangos medios de AGs de *k*-fija para CVIs externos utilizando todos los conjuntos de datos

CVI	CBGA	FGKA	GA	GAGR	GA_B	GAs	GCA	GKA	HKA	IGKA	KGA
Purity	4.824	5.511	7.261	6.691	6.617	4.84	6.899	6.787	5.457	6.351	**4.761**
Recall	**4.58**	5.596	7.936	6.128	6.505	5.505	7.154	5.67	5.995	5.585	5.346
Índice Jaccard (J)	**4.266**	5.34	7.957	6.287	6.33	5.223	7.33	6.266	6.021	5.862	5.117
Índice Rand (Ω)	**4.654**	5.16	7.559	6.83	6.468	5.021	7.441	6.564	5.543	5.931	4.83
F-measure (F_m)	**4.367**	5.431	7.957	6.229	6.394	5.181	7.255	6.319	5.989	5.83	5.048
Precision	**4.585**	5.069	7.34	6.787	6.617	5.09	7.303	6.628	5.702	6.037	4.84
Centroid empty index (CI_e)	**4.851**	5.915	7.601	6.362	6.42	5.399	6.356	7.021	4.734	6.34	5
Rango de Friedman	1.286	4.286	11.000	8.143	8.143	3.286	9.571	8.000	4.857	5.429	2.000

Similar a los CVIs internos, se puede observar que CBGA obtiene los rangos más bajos en la mayoría de las métricas, lo que nos indica que este algoritmo optimiza los resultados de estas métricas para la mayoría de los conjuntos de datos analizados. Para estudiar si existen diferencias significativas entre los resultados obtenidos por las diferentes propuestas, sobre estos valores de rangos se aplica el test de Friedman que nos permitirá realizar un análisis de meta-rangos, tal y como se explicado en la sección anterior. El rango asignado por el test de Friedman se especifica en la última fila de la Tabla 4.7 y muestra que CBGA obtiene el rango más bajo (1.286). Esto indica que es la mejor propuesta para la mayoría de las medidas. Un valor de 1 indicaría que es el mejor para todas las medidas, el tener un valor tan cercano a 1 nos indica que obtiene los mejores valores en la mayoría de las métricas.

En el caso de las medidas externas, el test de Friedman cuyos resultados son mostrados en la Tabla 4.8, también rechaza la hipótesis nula, por tanto, determina que existen diferencias significativas en el rendimiento de los algoritmos, por lo que también se realizó el test a posteriori de Shaffer. Las diferencias significativas entre los algoritmos para estas métricas externas con un nivel de confianza $> 95\%$ se muestran en la Tabla 4.9 y de manera gráfica en la Figura 4.4. Una vez más, CBGA trabaja significativamente mejor que las otras propuestas

Tabla 4.8 Test de Friedman y corrección de Iman Davenport sobre los rangos medios de AGs de *k*-fija para CVIs externos

CVIs	Test estadístico	df	p-valor
Externos	Test de Friedman's $\chi^2 = 64.338$	df = 10	= 5.425e-10
	Corrección de Iman Davenport $\chi^2 = 68.174$	df1 = 10, df2 = 60	<2.2e-16

Tabla 4.9 *p*-valores de los rangos promedio de CVIs externos de los AGs de *k*-fija, obtenidos con el test de Friedman post-hoc corrección Shaffer para un intervalo de confianza $> 95\%$

	CBGA	FGKA	GA	GAGR	GA_B	GAs	GCA	GKA	HKA	IGKA	KGA
CBGA	n/a	1.000	**0.000**	**0.005**	**0.005**	1.000	**0.000**	**0.007**	1.000	0.603	1.000
FGKA	1.000	n/a	**0.007**	0.917	0.917	1.000	0.106	1.000	1.000	1.000	1.000
GA	**0.000**	**0.007**	n/a	1.000	1.000	**0.001**	1.000	1.000	**0.024**	0.062	**0.000**
GAGR	**0.005**	0.917	1.000	n/a	1.000	0.227	1.000	1.000	1.000	1.000	**0.024**
GA_B	**0.005**	0.917	1.000	1.000	n/a	0.227	1.000	1.000	1.000	1.000	**0.024**
GAs	1.000	1.000	**0.001**	0.227	0.227	n/a	**0.018**	0.290	1.000	1.000	1.000
GCA	**0.000**	0.106	1.000	1.000	1.000	**0.018**	n/a	1.000	0.290	0.603	**0.001**
GKA	**0.007**	1.000	1.000	1.000	1.000	0.290	1.000	n/a	1.000	1.000	**0.026**
HKA	1.000	1.000	**0.024**	1.000	1.000	1.000	0.290	1.000	n/a	1.000	1.000
IGKA	0.603	1.000	0.062	1.000	1.000	1.000	0.603	1.000	1.000	n/a	1.000
KGA	1.000	1.000	**0.000**	**0.024**	**0.024**	1.000	**0.001**	**0.026**	1.000	1.000	n/a

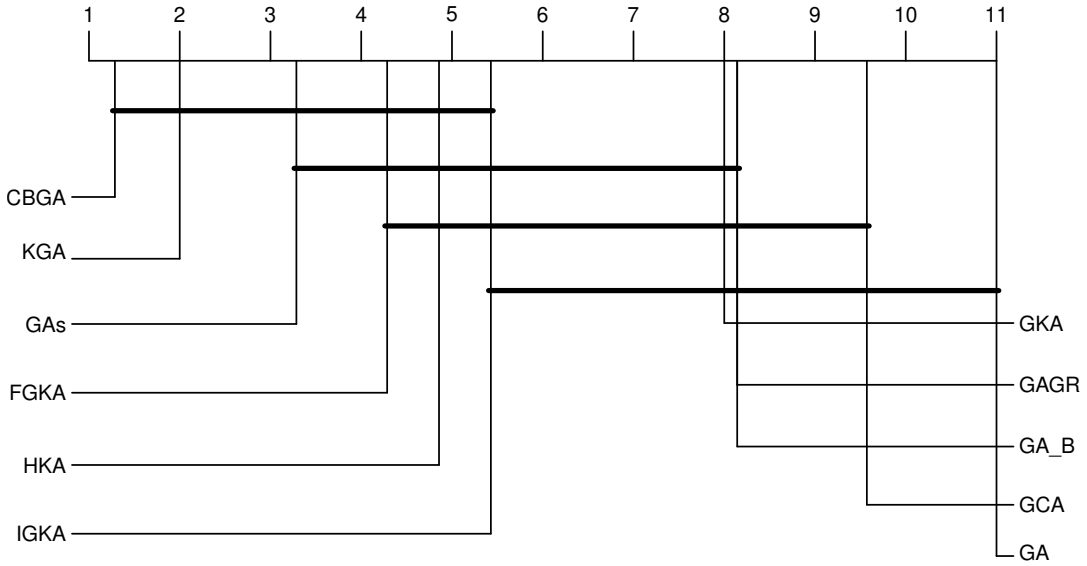


Figura 4.4 Distancia crítica del test a posteriori de Shaffer para los AGs de *k*-fija considerando los CVIs externos ($\alpha = 0.05$)

dado que es el algoritmo con un rango más bajo y se considera el algoritmo de control. Sin embargo, en este caso, las diferencias significativas entre algoritmos se reducen, al encontrar más algoritmos equivalentes. También es importante resaltar que cambia el orden de los mejores algoritmos con respecto a las métricas internas CBGA, KGA, GAs, FGKA, HKA y IGKA, siendo los que utilizan una codificación basada en centroides utilizando tanto operadores especializados como tradicionales para conjuntos de datos sobrepuestos los que obtienen mejores resultados. Los algoritmos considerados estadísticamente peores para estas métricas son GCA y GA, lo común de estos dos algoritmos es que no utilizan búsqueda local. Lo que también se muestra como representativo para mejorar la eficiencia de este tipo de algoritmos. Con respecto a la función de aptitud, la mayoría de los algoritmos se basa en estudiar la compactibilidad.

4.2.3 Conclusiones de los resultados experimentales de los AGs de *k*-fija

Resultados similares son obtenidos tanto en las medidas internas, como en las externas. En ambos casos el algoritmo CBGA se impone como uno de los algoritmos más eficientes para resolver el problema de agrupamiento. A partir de los resultados obtenidos, se puede concluir que los AGs que obtienen mejores resultados son los que usan una codificación basada en centroides con operadores especializados y tradicionales. La siguiente representación que permite también explorar de forma adecuada el espacio de búsqueda sería las codificaciones basadas en etiqueta que usan operadores especializados. En función del número de instancias se podría decantar por unas u otras. Las codificaciones que han obtenido peores resultados son las codificaciones que utilizan una representación binaria y basada en etiquetas que emplean los operadores tradicionales.

Otra interesante característica encontrada es respecto al uso de la búsqueda local, como K-means, el cual es un componente que puede ser utilizado con los AGs para obtener mejores resultados sobre todo para conjuntos de datos de grandes dimensiones. Los AGs que peores resultados logran no incorporan estos métodos en su estrategia evolutiva.

Respecto a los algoritmos de *k*-fija, como el número de grupos está determinado como un parámetro de entrada, hace que los AGs tengan como objetivo principal crear grupos compactos, las métricas usadas por los AGs de *k*-fija trabajan en esta dirección, tales como: DD, SED, TWCV, SSE y J_1, por lo que el estudio de esta sección presenta resultados determinantes de desempeño para los diferentes AGs.

4.3 Análisis de resultados experimentales de los AGs de *k*-variable

En este estudio se han analizado 11 AGs considerados el estado del arte para resolver el problema de agrupación basado en particiones para *k*-variable. Los algoritmos de *k*-variable se diferencian de los de *k*-fija principalmente en que no necesitan especificar el valor de entrada *k*. De este modo, este valor es investigado automáticamente por cada algoritmo, agregando interés desde el punto de la ciencia de los datos por el hecho de descubrir nuevo conocimiento. Estos algoritmos se encargan tanto de determinar el número de grupos, como de encontrar la mejor distribución de los objetos del conjunto de datos en los grupos. Las principales características de cada algoritmo están descritas en la sección 3.2. Además, la configuración de cada uno de los AGs es descrita en la sección 4.1.3. La idea es hacer un estudio comparativo de todos los algoritmos para determinar si alguno de ellos tiene diferencias de desempeño significativa con respecto a los demás y obtener conclusiones relevantes acerca de las ventajas e inconvenientes de los diferentes AGs propuestos en la literatura. Con este propósito, los 11 algoritmos son analizados usando **22** CVIs y evaluando **92** conjuntos de datos de los descritos en la sección 4.1.1. Inicialmente se planteó evaluar los 94 conjuntos de datos descritos en la sección 4.1.1, pero debido a la complejidad del espacio de búsqueda de los AGs basados en una codificación en grafos hacen que la memoria de los equipos utilizados sea insuficiente para los conjuntos de datos Birch1 y Birch2, por tanto, el total de conjuntos de datos fueron 92 para los algoritmos de *k*-variable.

Este tipo de algoritmos suelen incluir como función a optimizar, un CVI interno que permita optimizar la combinación de las propiedades de compactibilidad y separación, que le permitirá guiar la estrategia del AG para obtener el número de grupos óptimo en un conjunto de datos.

El procedimiento utilizado para este estudio es similar al descrito previamente en la sección 4.2. Primero, se llevará a cabo un análisis de los CVIs internos y después, de la misma manera considerando los CVIs externos, al utilizar un test estadístico no paramétricos [26] para determinar el desempeño de los AGs.

4.3.1 Análisis de los CVIs internos

Debido a la gran cantidad de conjuntos de datos, métricas y algoritmos de *k*-variable, de manera similar al análisis de los algoritmos con *k*-fija, se realizó un análisis preliminar de dos índices representativos: Silhouette (13) y DB (11).

Los resultados promedio del índice Silhouette de los AGs para los conjuntos de datos se muestran de manera gráfica en la Figura 4.6a. Se puede resaltar que es una distribución

bimodal y con varianza más pequeña entre los promedios en los algoritmos que no optimizan concretamente esta métrica. Además, a diferencia de los AGs de *k*-fija se obtienen algunos valores de -1 por la dificultad de obtener una partición para algunos conjuntos de datos. También, algunos AGs logran valores mayores a 0 para todos los conjuntos de datos con respecto a los algoritmos de *k*-fija implicando que algunos conjuntos de datos son agrupados de una forma óptima usando una *k* diferente a la dada por el número de clases.

En la Figura 4.6a, se puede ver que los mejores valores del índice Silhouette son obtenidos por los algoritmos CGA y GGA_S los cuales utilizan esta métrica como función de aptitud, seguidos por $FEAC_{SS}$, EAC con métrica Simplified Silhouette y después GCUK que emplea el índice DB.

Los mejores valores específicos del índice Silhouette para conjuntos de datos reales son 0.991, 0.753 y 0.75 para Titanic, Shuttle y Abalone, respectivamente. Para conjuntos de datos artificiales alrededor de 0.94 para los Dim, 0.832 para Unbalance y 0.73 para las G2. Se recuerda que todos los valores medios por métricas y resultados de los diferentes test se pueden consultar en la página web asociada[8].

El índice DB es otra medida mostrada en la Figura 4.6b, se puede ver que los mejores resultados de valores medio e intercuartílicos son alcanzados por los algoritmos $FEAC_{SS}$, EAC, CGA y GCUK. De estos algoritmos solamente el último utiliza el índice DB como función de aptitud. De los algoritmos analizados solamente GGA_{DB} que utiliza esta medida no obtiene unos resultados relevantes, posiblemente debido a la codificación que emplea, la cual está relacionada con los operadores utilizados y el acoplamiento del CVI. También es importante mencionar que el índice de Simplified Silhouette tiene cierta compatibilidad con el índice DB.

Respecto a los valores específicos mejores obtenidos del índice DB, existe coincidencia para los conjuntos de datos artificiales Dim con índice Silhouette, con un valor alrededor de 0.138, para los conjuntos de datos con distribuciones gaussianas se encuentran diferencias. R15 con 0.353, Unbalance con 0.385 y enseguida los G2 alrededor de 0.394. Para los datos reales Titanic, Spambase y Abalone con valores 0.042, 0.294 y 0.381, respectivamente coincidiendo con el índice Silhouette para Titanic.

En seguida se paso a analizar los rangos promedio de cada AG para cada uno de los CVIs internos considerados. En la Tabla 4.10, se muestran los rangos promedio de los CVIs internos. Los AGs con los valores más bajos indican un mejor rendimiento para la mayoría de los conjuntos de datos. Se puede ver que los CVIs que combinan los mejores valores de las medidas de compactibilidad y separación son obtenidos por el algoritmo GCA, seguido por el algoritmo GGA_S, por lo que se les puede considerar las mejores propuestas. Las métricas que miden solamente compactibilidad son optimizadas por el algoritmo CLUSTERING. Esto se debe a la tendencia de este algoritmo a generar un número grande de grupos por su función

de aptitud no "equilibrada", al dar más peso en la ecuación a la compactibilidad y considerar menos relevante la distancia entre grupos. El segundo algoritmo que obtiene mejores valores en este tipo de métricas sería GGA_{DB}, este algoritmo utiliza el índice DB como función de aptitud y el problema puede deberse a los operadores genéticos que utiliza.

Tabla 4.10 Rangos medios de AGs de k-variable para los CVIs internos utilizando todos los conjuntos de datos

CVI	CGA	CLUST	EAC	$FEAC_{RI}$	$FEAC_{SS}$	GA_C	GCUK	GGA_{DB}	GGA_S	TGCA	VGA
CS measure (CS)	**3.397**	9.038	4.598	5.793	5.152	5.261	7.902	7.875	4.723	7.13	5.13
Índice Davies-Bouldin (DB)	4.342	7.299	3.712	6.946	**3.342**	9.217	4.516	7.924	5.125	7.658	5.918
Índice Xie Beni (XB)	**3.989**	4.81	4.56	7.815	4.353	9.283	4.049	8.114	4.87	7.543	6.614
Índice S Dunn (SD)	**2.935**	10.31	4.13	7.185	4.652	8.109	4.5	8.071	4.13	7.38	4.598
Índice Dunn (D)	**3.696**	10.53	5.707	6.413	6.005	5.701	5.44	7.69	3.87	5.527	5.418
Índice I (I)	**2.663**	10.1	5.212	6.87	5.489	6.897	6.158	8.223	4.772	6.929	2.685
Score function (SF)	3.701	10.12	5.049	6.076	5.462	5.125	7.071	7.625	**3.582**	6.826	5.364
Silhouette (S)	**3.038**	9.168	5.158	6.554	4.973	7.245	5.522	7.701	3.12	6.804	6.717
Simplified Silhouette (SS)	3.495	9.924	4.967	7.239	5.228	6.652	5.413	7.951	**3.185**	6.049	5.897
C. de relación de varianza (VRC)	4.19	9.864	6.283	7.402	6.207	5.804	5.283	7.478	**3.402**	5.25	4.837
Índice WB (WB)	5.022	8.908	6.81	7.304	6.685	5.516	4.951	7.478	**3.495**	4.19	5.641
Suma de dist Euclidianas (SED)	8.897	**3.527**	7.56	5.332	6.408	6.326	4.674	4.342	6.348	5.299	7.288
Dist. distorsionada (DD)	8.777	**3.304**	7.641	5.413	6.348	6.283	4.804	4.321	6.391	5.288	7.429
S. del error cuadrático (SSE)	8.832	**3.321**	7.663	5.451	6.446	6.364	4.707	4.38	6.299	5.234	7.304
Error cuadrado mín func (J_m)	9.326	**2.745**	7.554	5.304	6.625	5.804	4.87	3.984	6.929	5.707	7.152
Rangos de Friedman	4.200	7.733	5.833	7.000	5.667	7.267	4.600	7.733	3.767	6.200	6.000

Tabla 4.11 Test de Friedman y corrección de Iman Davenport sobre los rangos medios de AGs de k-variables para CVIs internos

CVIs	Test estadístico	df	p-valor
Internos	Test de Friedman $\chi^2 = 25.882$	df = 10	= 0.003902
	Corrección de Iman Davenport $\chi^2 = 2.9194$	df1 = 10, df2 = 140	= 0.002372

Con el objetivo de evaluar cuál algoritmo de k-variable, tiene en general un mejor rendimiento, al lograr el mejor rango para el mayor número de métricas, se hizo un análisis de meta-rangos [16], de la misma manera que para los algoritmos de k-fija, el cual fue descrito en la sección anterior. Para esto se aplica el test de Friedman sobre los rangos medio de la Tabla 4.10, en la última fila de esta se muestra el rango asignado a cada AG por este test. Se puede ver que el algoritmo que obtiene el valor de rango más bajo es CGA, seguido muy de cerca por el algoritmo GCUK.

Continuando con el análisis de meta-rangos, los resultados del test de Friedman y la corrección de Iman Davenport puede verse en la Tabla 4.11. Con un nivel de confianza de 95%, el test de Friedman rechaza la hipótesis nula, determinando que existen diferencias significativas en el rendimiento de por lo menos uno de los algoritmos. Para confirmar entre qué propuestas existen diferencias significativas, se realizó el test a posteriori de Shaffer y sus

resultados se muestran en la Tabla 4.12 y la gráfica de los rangos con la distancia crítica en la Figura 4.5 con intervalo de confianza $> 95\%$. Aunque no existen diferencias importantes se puede ver que existen dos grupos de algoritmos, encabezados por GGA_S y CGA. Estos algoritmos se caracterizan por utilizar la codificación basada en etiquetas y la métrica Silhouette como función de aptitud. En general, los algoritmos que utilizan una codificación basada en etiquetas obtienen los mejores resultados y la representación basada en grafos obtiene los peores resultados. En los algoritmos de *k*-variable no aparece como operador determinante la búsqueda local, se obtienen resultados similares con métodos diferentes o sin utilizar un método específico. Finalmente, la medida de Silhouette los algoritmos que obtienen mejores resultados son los que utilizan a esta, seguidas por el índice de DB. En estos algoritmos, la combinación de compactibilidad y separación obtiene mejores resultados.

Tabla 4.12 *p*-valores de los rangos promedio de CVIs internos de los AGs de *k*-variable, obtenidos con el test de Friedman post-hoc corrección Shaffer para un intervalo de confianza $> 95\%$

	CGA	CLUST	EAC	FEAC_RI	FEAC_SS	GA_C	GCUK	GGA_DB	GGA_S	TGCA	VGA
CGA	n/a	0.159	1.000	0.935	1.000	0.510	1.000	0.159	1.000	1.000	1.000
CLUST	0.159	n/a	1.000	1.000	1.000	1.000	0.435	1.000	0.058	1.000	1.000
EAC	1.000	1.000	n/a	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
FEAC_RI	0.935	1.000	1.000	n/a	1.000	1.000	1.000	1.000	0.341	1.000	1.000
FEAC_SS	1.000	1.000	1.000	1.000	n/a	1.000	1.000	1.000	1.000	1.000	1.000
GA_C	0.510	1.000	1.000	1.000	1.000	n/a	1.000	1.000	0.173	1.000	1.000
GCUK	1.000	0.435	1.000	1.000	1.000	1.000	n/a	0.435	1.000	1.000	1.000
GGA_DB	0.159	1.000	1.000	1.000	1.000	1.000	0.435	n/a	0.058	1.000	1.000
GGA_S	1.000	0.058	1.000	0.341	1.000	0.173	1.000	0.058	n/a	1.000	1.000
TGCA	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	n/a	1.000
VGA	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	n/a

4.3.2 Análisis de los CVIs externos

Al estudiar los algoritmos de *k*-variable, desde la perspectiva de las métricas externas es de interés por la autonomía del algoritmo para encontrar el número de grupos y analizar la correspondencia entre las clases y grupos por medio de las métricas externas. Se puede comentar, que las medidas externas consideran la estructura preestablecida por un experto en el conjunto de datos.

De los algoritmos estudiados, $FEAC_{RI}$, es el único que utiliza una métrica externa (el índice Rand) como función aptitud. Esta métrica no está basada en los principios de la agrupación de compactibilidad y separación, además se requiere conocer las clases que se relacionan con los grupos para poder aplicarla, lo que no siempre es factible en el problema de agrupamiento, al ser caracterizado por un aprendizaje no supervisado. Es de esperar que los mejores resultados del índice Rand sean obtenidos por $FEAC_{RI}$, tal y como se muestra en la Figura 4.7a con

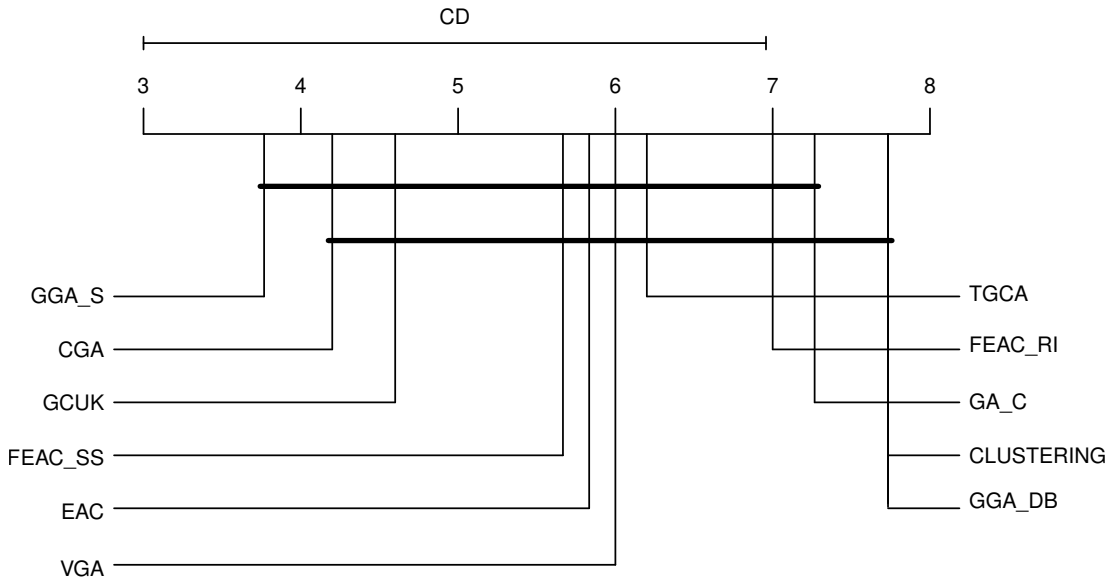
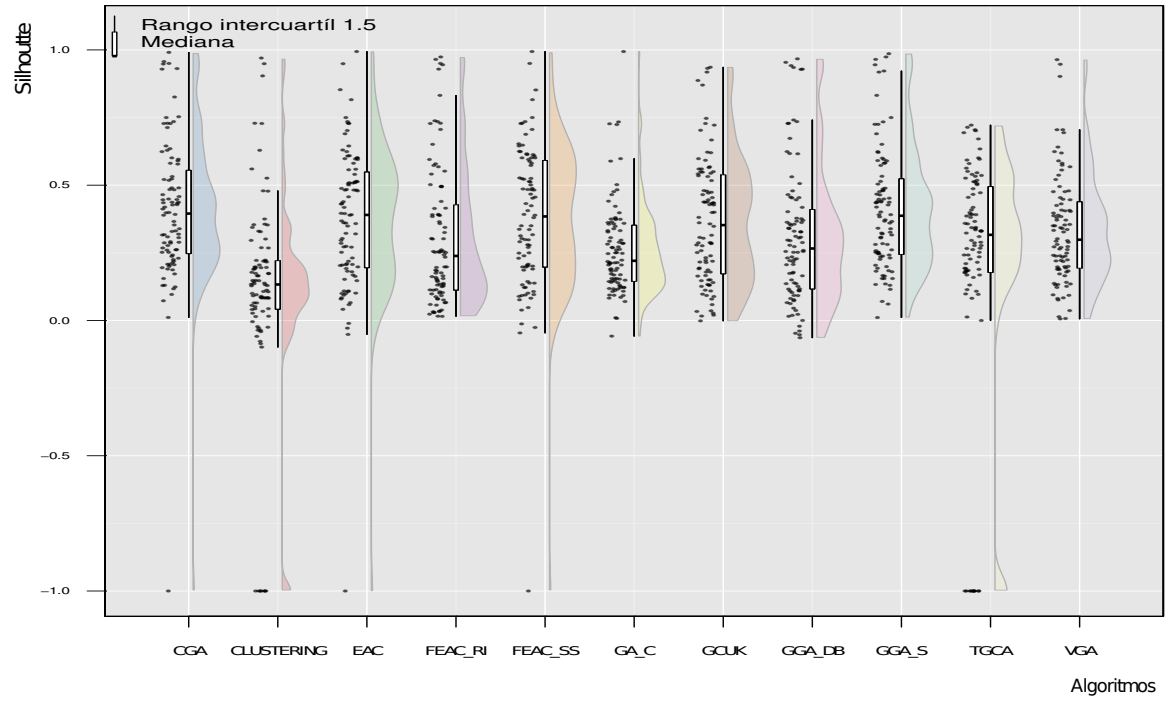


Figura 4.5 Distancia crítica del test a posteriori de Shaffer para los AGs de *k*-variable considerando los CVIs externos ($\alpha = 0.05$)

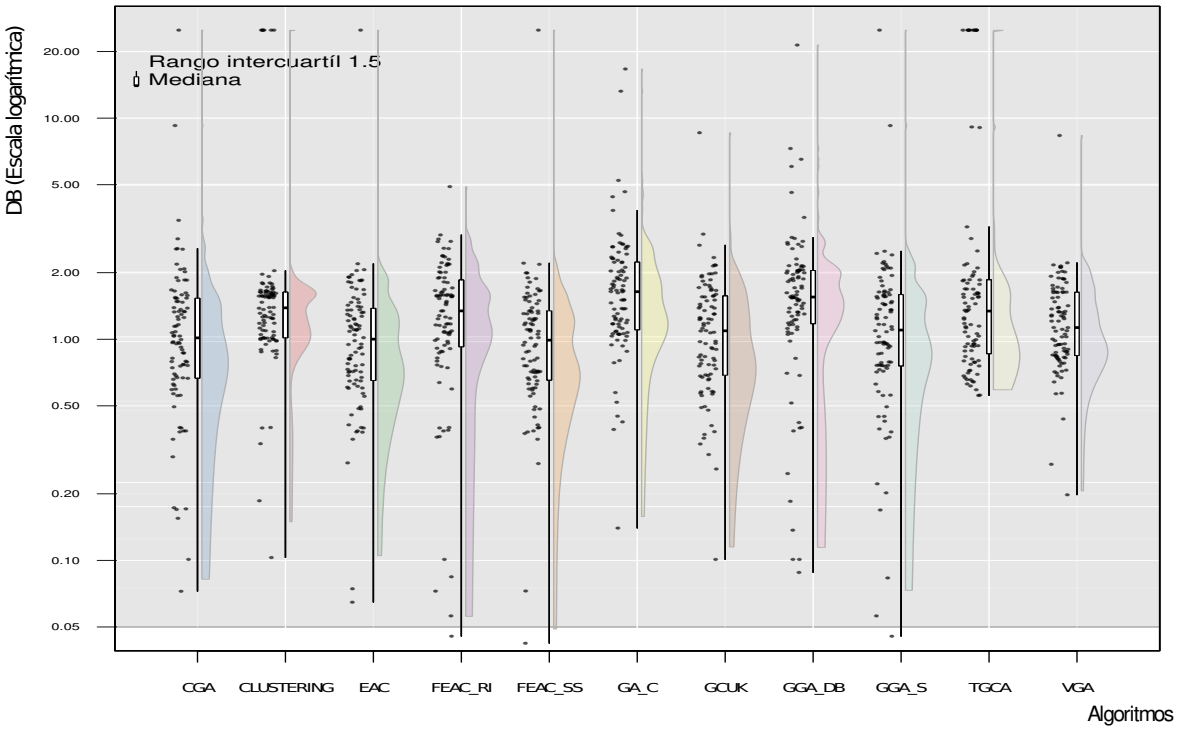
una mediana y rango intercuartílico más cercano a 1 con respecto a todos los otros AGs. Los siguientes algoritmos con mejores valores son GGA_S, GCUK y GGA_{DB}.

Los mejores valores específicos para los conjuntos de datos artificiales del índice Rand son para G2, Dim y Unbalance con un valor de 1. Por otro lado los conjuntos de datos reales llaman la atención por los diferentes valores para un mismo conjunto de datos por diferentes AGs como Leaves Plant Texture de 0.438 a 0.995, Dermatology de 0.524 a 0.978. Es importante resaltar que, a pesar de obtener un valor mejor, los CVIs internos analizados anteriormente, no siempre están relacionados de manera proporcional a valores altos en métricas externas para todos los conjuntos de datos, esto se puede encontrar principalmente en datos reales como el caso particular de Titanic con 0.582 de índice de Rand; todos estos valores se pueden consultar en la página web asociada[8].

El índice F-measure, es una métrica con propiedades similares a las del índice Rand, pero si lo que se busca es obtener un mejor valor, en esta métrica debe de existir una igualdad entre el número de clases y grupos, determinándose de manera implícita en su ecuación. Los mejores valores son obtenidos por el algoritmo $FEAC_{RI}$ como se muestra en la Figura 4.7b con una mediana y rango intercuartílico más grande con respecto a todos los otros AGs, así con el test de Friedman (ver Tabla 4.14 y Figura 4.8) se puede ver que los algoritmos $FEAC_{RI}$, CGA y GGA_S son equivalentes a pesar de usar métricas con diferentes enfoques, con este resultado, se puede establecer con certeza $> 95\%$ que el índice Silhoutte utilizado de manera individual

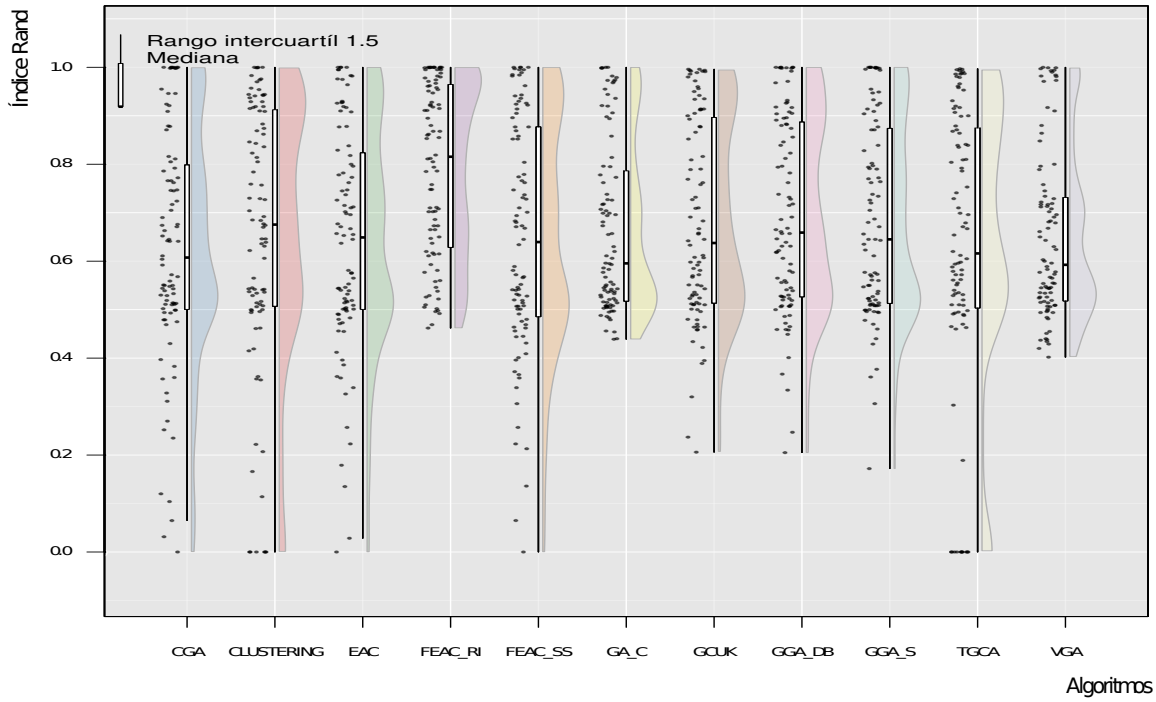


(a) Resultados del índice de Silhoutte (↑) por algoritmo y conjunto de datos

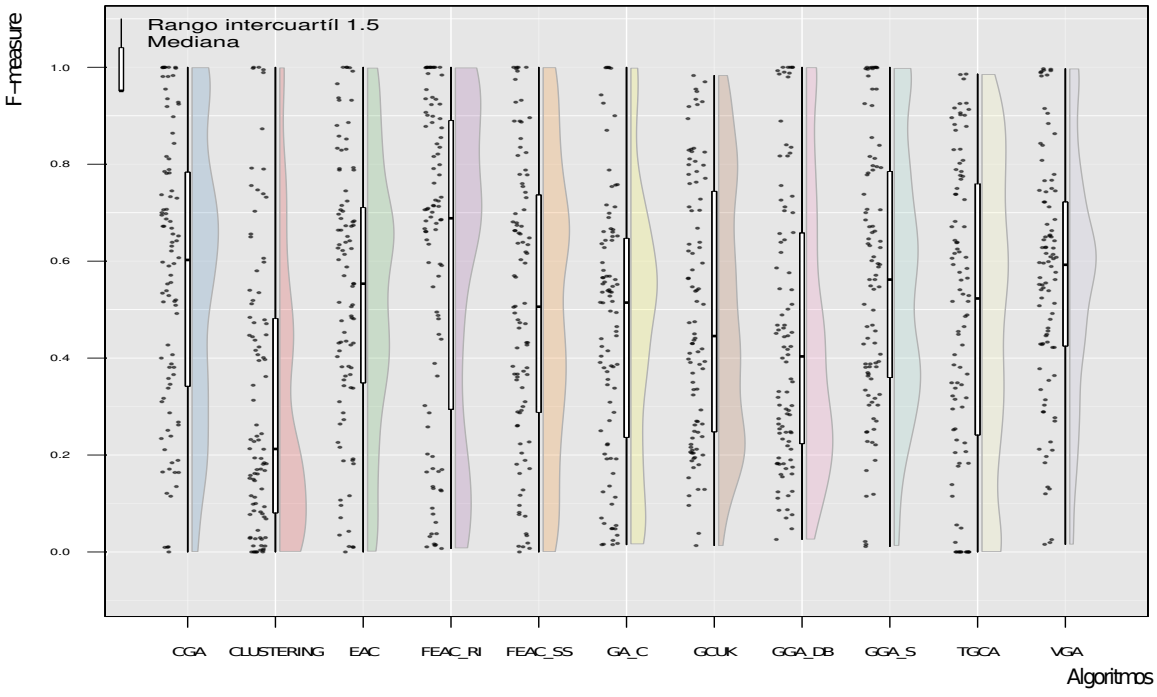


(b) Resultados del índice DB (↓) por algoritmo y conjunto de datos

Figura 4.6 Resultados promedio de CVIs internos para AGs de *k*-variable



(a) Resultados del índice Rand (Ω) ($\uparrow$) por algoritmo y conjunto de datos



(b) Resultados de F-measure ($\uparrow$) por algoritmo y conjunto de datos

Figura 4.7 Resultados promedio de CVIs externos para AGs de *k*-variable

puede ser mejor que los otros CVIs para encontrar el número de grupos con relación a las clases en general para los diferentes conjuntos de datos, aunque el índice Silhoutte tiene la desventaja de tener un costo computacional de $O(n^2)$ que lo hace que no sea escalable para conjuntos de datos actuales.

Los algoritmos que separan los objetos con un número grande de grupos obtendrán peores resultados en el índice F-measure, como es el caso del algoritmo CLUSTERING. También, los algoritmos que no obtienen buenos resultados en esta métrica son TGCA, GCUK y GGA_{DB}, el primero emplea como función de aptitud el índice VRC, y los dos siguientes el índice DB.

Los mejores valores específicos por los AGs de *k*-variable en los conjuntos de datos reales son en Twonorm con un valor de 0.955, lo que lo hace predecible con el modelo de grupos obtenido, Wisconsin con 0.934 y Wdbc con 0.878. Para conjuntos de datos artificiales coinciden varios CVI con los mejores valores al igual que F-measure en Dim y G2 con un valor 1, después siguiendo S1.

Tabla 4.13 Rangos medios de AGs de *k*-variable para CVIs externos utilizando todos los conjuntos de datos

CVI	CGA	CLUST	EAC	$FEAC_{RI}$	$FEAC_{SS}$	GA_C	GCUK	GGA_{DB}	GGA_{S}	TGCA	VGA
Purity	8.451	**3.38**	7.386	4.223	6.217	6.277	5.245	4.538	6.755	6.114	7.413
Recall	**2.603**	9.342	4.212	5.603	5.06	6.647	6.897	7.799	4.739	7.614	5.484
Índice Jaccard (J)	4.853	8.429	5.723	**4.005**	6.06	6.484	6.571	6.842	4.891	6.489	5.652
Índice Rand (Ω)	6.527	6.435	6.511	**2.69**	6.326	6.63	5.978	6.065	5.707	6.332	6.799
F-measure (F_m)	4.826	8.397	5.761	**3.995**	6.038	6.418	6.674	6.826	4.913	6.484	5.668
Precision	8.071	3.967	7.685	**3.967**	6.978	5.875	5.342	4.908	6.212	5.745	7.25
Rangos de Friedman	6.000	7.083	6.500	2.083	5.833	7.333	6.167	6.667	4.333	6.833	7.167

Tabla 4.14 Test de Friedman y corrección de Iman Davenport sobre los rangos medios de AGs de *k*-variables para CVIs externos

CVIs	Test estadístico	df	p-valor
Externos	Test de Friedman $\chi^2 = 13.023$	df = 10	= 0.2224
	Corrección de Iman Davenport $\chi^2 = 1.3861$	df1 = 10, df2 = 50	= 0.2141

A continuación, se analizaron los rangos promedio de cada AG para cada uno de los CVIs externos considerados. Se sigue el mismo procedimiento que se ha comentado para los AGs de *k*-fija y se recuerda que todos los resultados medios por métrica se pueden consultar de forma particular en la página web asociada [8].

En la Tabla 4.13, se muestran los rangos promedio de los CVIs externos. Los AGs con los valores más bajos indican un mejor rendimiento para la mayoría de los conjuntos de datos. El algoritmo con mejores resultados en general es $FEAC_{RI}$, siendo un resultado esperado por

utilizar la función aptitud externa de índice de Rand como se ha descrito al principio de esta sección.

El siguiente paso para realizar el análisis de los meta-rangos es aplicar el test de Friedman sobre los rangos medios de la tabla anterior, para comprobar si hay diferencias significativas en los resultados alcanzados por los diferentes AGs, el resultado se encuentra en la última fila de la Tabla 4.13, donde se muestra el rango asignado a cada AG por este test. Se puede ver que el mejor algoritmo al obtener el valor de rango más bajo es FEAC_{RI}, siendo el siguiente el algoritmo GGA_{S}.

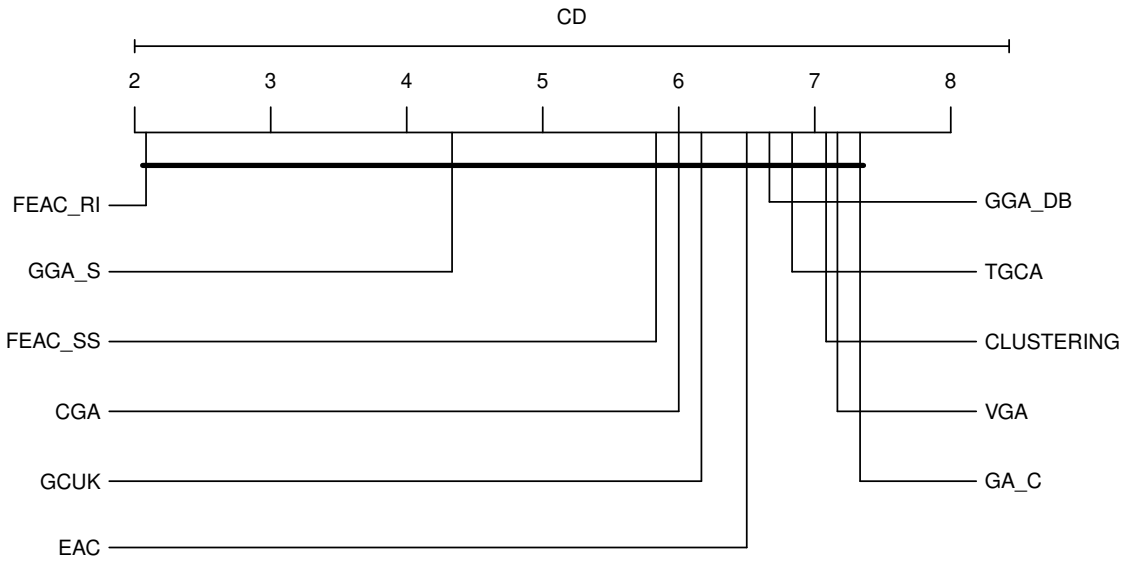


Figura 4.8 Distancia crítica del test a posteriori de Shaffer para los AGs de *k*-variable considerando los CVIs externos ($\alpha = 0.05$)

Los resultados del test de Friedman y la corrección de Iman Davenport puede verse en la Tabla 4.14. Con un nivel de confianza de 95%, el test de Friedman acepta la hipótesis nula, determinando que no existen diferencias significativas en el rendimiento de por lo menos uno de los algoritmos. En las métricas externas, como se ha visto las diferencias entre los algoritmos tiene una tendencia a minimizarse, como se puede ver los resultados en la Tabla 4.14, con el *p*-value en el test de Friedman, el cual no muestra diferencias significativas entre los algoritmos de *k*-variable. Aunque el AG de FEAC_{RI} basado en un CVI externo muestra un buen desempeño para las métricas de índice de Rand y F-measure, pero analizado con el conjunto de métricas externas no llega a mostrar diferencias significativas, el orden de desempeño de los algoritmos en los CVIs externos es mostrado en la Figura 4.8. Se puede ver que GGA_S y CGA se encuentra en una de las posiciones con mejores rangos para los CVIs externos, con lo que se confirma que la codificación basada en etiquetas obtiene mejor rendimiento frente a las otras

representaciones. Es importante resaltar que estas medidas son más complejas en el caso de los AGs con *k*-variable, ya que el número de grupos que puede determinar puede ser muy diferente del número de clases establecido para un conjunto de datos, lo que dificulta la tarea y hace que realmente no haya diferencias representativas.

4.3.3 Conclusiones de los resultados experimentales de los AGs de *k*-variable

Aunque es complejo obtener diferencias significativas entre los algoritmos de *k*-variable, por el hecho de que las métricas miden dos propiedades *compactibilidad* y *separación*, y cada una de ellas trabaja con un número de grupos diferente determinado por el individuo, lo que dificulta la tarea de comparar sus resultados tanto de CVIs internos como externos.

A pesar de esto, se puede establecer cierta tendencia en los resultados analizados, los algoritmos GGA_S y CGA se presentan como las mejores propuestas, obteniendo resultados relevantes en las medidas internas y externas, considerando una amplia variedad de datos y medidas. Estos algoritmos se caracterizan por utilizar la codificación basada en etiquetas y la métrica Silhouette como función de aptitud. En este tipo de AGs se ha introducido una codificación basada en grafos que no utilizaban los AGs de *k*-fija y que no han demostrado un buen resultado en la resolución del problema. Analizando todos los CVIs utilizados por los diferentes AGs, se aprecia que los algoritmos que utilizan la medida de Silhouette obtienen mejores resultados, seguidos por los del índice de DB. Demostrando que los CVIs basados en la combinación adecuada de compactibilidad y separación resultan ser fundamentales para los algoritmos de *k*-variable.

Finalmente, el uso de la búsqueda local en el algoritmo, en este tipo de algoritmos no se considera tan relevante, ya que una de las mejores propuestas (CGA) no realiza ningún tipo de búsqueda.

Capítulo 5

GAMK: Un algoritmo genético para agrupación con número de grupos conocido

En este capítulo se propone un nuevo AG para la solución del problema de agrupación basado en particiones, por sus características de operación es llamado Algoritmo Genético con Mezcla de grupos con `K-means` (GAMK, Genetic Algorithm Mix-groups with `K-means`).

La propuesta de GAMK (pronunciado como "GA-M-K"), comienza considerando la problemática encontrada en CGA [60], al establecer que el problema de agrupación presenta redundancia en la codificación utilizada por un AG, al depender no sólo de su esquema, sino también de los operadores empleados. Por ello, en GAMK se desarrollan operadores especializados para problemas de agrupación que permiten dispensar la codificación redundante, así como solucionar el problema de la insensibilidad al contexto [36, 60].

Para el diseño de GAMK se utilizó el esquema general de un AG descrito en el Algoritmo (4), considerando sus características particulares. Para su implementación, se usó LEAC [109], que es la biblioteca especializada para construir AGs de agrupación basados en particiones que se ha desarrollado en esta tesis, e incluye las implementaciones de todos los AGs propuestos en la literatura.

En las siguientes secciones se describirán los principales componentes de GAMK y se llevará a cabo un estudio experimental que nos indique su rendimiento en comparación con las propuestas previas.

5.1 Codificación de individuos

Los individuos de la población de GAMK utilizan una codificación basada en centroides con cromosomas representados como χ_μ (3.7) de tamaño $k \times d$ de números decimales. Se prefirió esta codificación por ser más adecuada para el diseño de nuevos operadores que tratan de evitar los problemas en los operadores de cruce tradicionales descritos en la sección 3.1.4. A diferencia de los actuales AGs para k-fija que utilizan cromosomas χ_x (3.6) donde un gen es una dimensión de un centroide, aquí un centroide representará un gen.

5.2 Inicialización de la población

El algoritmo de agrupación GAMK inicializa una población de tamaño n_P, que se mantiene fija a lo largo de la evolución. Cada individuo es inicializado seleccionando aleatoriamente k instancias del conjunto de datos con los que se inicializan los centroides de cada grupo. La única restricción que se contempla es que estas instancias sean diferentes para cada individuo.

Algoritmo 13 Operador de cruce `cruce-CI`

Entrada: $\chi_{\mu_1} = \langle \mu_j^1 \rangle$ y $\chi_{\mu_2} = \langle \mu_j^2 \rangle$: dos cromosomas padre basada en centroides (3.7), donde $j : j \in \{1, 2, ..., k\}$

Salida: $\chi'_{\mu_1} = \langle \mu_j^{1\prime} \rangle$ y $\chi'_{\mu_2} = \langle \mu_j^{2\prime} \rangle$: dos cromosomas hijos

1: $\{\mu_j^1\} \leftarrow$ decodificar(χ_{μ_1})
2: $\{\mu_j^2\} \leftarrow$ decodificar(χ_{μ_2})
3: $\{\mu_j^{1\prime}\} \leftarrow$ `función-CI`$(\{\mu_j^1\}, \{\mu_j^2\})$
4: $\{\mu_j^{2\prime}\} \leftarrow$ `función-CI`$(\{\mu_j^2\}, \{\mu_j^1\})$
5: $\chi'_{\mu_1} \leftarrow$ codificar$(\{\mu_j^1\}')$
6: $\chi'_{\mu_2} \leftarrow$ codificar$(\{\mu_j^2\}')$

Algoritmo 14 `función-CI`

Entrada: $\{\mu_j^1\}$, $\{\mu_j^2\}$: dos conjuntos de centroides, donde $j : j \in \{1,2,...,k\}$
Salida: $\{\mu_j'\}$ un nuevo conjunto de centroides

$\{C_{\mu_j}\}$: una partición de $\{\mu_j^2\}$ asociada a los centroides $\{\mu_j^1\}$
$\{C_{\mu_j}\} = \{\mu_j^2 : \|\mu_j^2 - \mu_{j'}^1\| \leq \|\mu_j^2 - \mu_{j'}^1\|,\ j' = 1,2,...,k \text{ y } j \neq j'\}$ basada en (2.5)
`map`$\{(\mu_j^1, \mu_j^2)\} \leftarrow$ `desunionCentroide`$(\{\mu_j^1\}, \{\mu_j^2\}, \{C_{\mu_j}\})$ ▷ Componente aleatorio
for $j \leftarrow 1$ to k **do**
    if $|C_{\mu_j}^2| = 0$ **then** numGenesFaltan $\leftarrow$ numGenesFaltan $+ 1$
    else if $|C_{\mu_j}^2| > 1$ **then** numGenesAgrupar $\leftarrow$ numGenesAgrupar $+ |C_{\mu_j}^2| - 1$
    end if
end for
▷ Ajustar el número de genes para maneter k grupos en la solucion
if numGenesAgrupar $<$ numGenesFaltan **then**
    numGenesCruce $\leftarrow$ numGenesAgrupar
else
    numGenesCruce $\leftarrow$ numGenesFaltan
end if
if numGenesCruce ≥ 1 **then** ▷ Construir el nuevo cromosoma
    ▷ Punto de cruce, el índice donde inicia el la busqueda de genes a intercambiar
    $j \leftarrow$ número aleatorio $\in [1,k]$
    numGenesFaltan $\leftarrow 0$
    numGenesAgrupar $\leftarrow 0$
    bGenACruce$[] = \langle \text{false}_1, \text{false}_2, ..., \text{false}_k \rangle$
    while numGenesFaltan $<$ numGenesCruce$||$numGenesAgrupar $<$ numGenesCruce **do**
        if numGenesFaltan $<$ numGenesCruce$\&\&|C_{\mu_j}^2| = 0$ **then**
            numGenesFaltan$++$
            bGenACruce$[j] =$ true
        else if numGenesAgrupar $<$ numGenesCruce$\&\&|C_{\mu_j}^2| > 1$ **then**
            numGenesAgrupar $\leftarrow$ numGenesAgrupar $+ |C_{\mu_j}^2| - 1$
            bGenACruce$[j] =$ true
        end if
        $j \leftarrow j\%(k+1)$ ▷ La cadena de genes es vista como un arreglo circular
    end while
    for $j \leftarrow 1$, $j' \leftarrow 1$; $j <= k \&\& j' <= k$; $j++$ **do**
        if bGenACruce$[j]$ **then**
            $\mu_{j'}' \leftarrow \mu_j^1$; $j'++$
        else
            for $\mu_j^2 \in C_{\mu_j}^2$ **do** $\mu_{j'}' \leftarrow \mu_j^2$; $j'++$
            end for
        end if
        if $\mu_j^1 \in$ `map`$\{(\mu_j^1, \mu_j^2)\}$ **then** $\mu_{j'}' \leftarrow$ `second(map`$\{(\mu_j^1, \mu_j^2)\})$; $j'++$
        end if
    end for
else
    $\{\mu_j'\} = \{\mu_j^1\}$
end if

5.3 Operadores genéticos

En esta sección se describen los diferentes operadores que fueron diseñados para el algoritmo GAKM tanto de selección, cruce, como mutación.

5.3.1 Operador de selección

Como se comentó en la sección 3.1.3, el proceso de selección es aplicado en cada generación para obtener a los individuos padres.

En GAKM se ha propuesto una selección elitista al aplicar el operador de cruce. Los individuos de la población son iterados para obtener un padre, y se evalúa una probabilidad adaptativa, p_c, si se realiza el cruce el otro padre es el mejor individuo obtenido hasta la generación actual, resultando dos hijos. El mejor de los hijos reemplaza al padre actual pasando a la siguiente generación, si no se realiza el cruce el padre pasa directamente a la siguiente generación.

5.3.2 Operador de cruce-CI

El operador que se utiliza para realizar la reproducción de individuos es el `cruce-CI`, llamado así por las operaciones que lleva a cabo de *capturar* e *intercambiar*. Está especificado en el Algoritmo (13), recibe como entrada dos individuos padres con cromosomas χ_{μ_1} y χ_{μ_2} (3.7). Después de decodificar los individuos, el operador de cruce se centra en la `función-CI` (Algoritmo 14) para generar los dos hijos. El primer paso de esta función es agrupar los centroides representados por el cromosoma padre y con las particiones obtenidas, en un segundo paso, se construyen los individuos hijos con una estrategia de intercambio de genes de acuerdo con la cardinalidad de los subconjuntos de centroides.

Un ejemplo ilustrativo de cómo opera el operador `cruce-CI` para los individuos padres codificados por los cromosomas χ_{μ_1} (5.1) y χ_{μ_2}(5.2), representados de manera esquemática en la Figura 5.1a.

$$\chi_{\mu_1} = \langle \mu_1^1, \mu_2^1, \mu_3^1, \mu_4^1, \mu_5^1, \mu_6^1, \mu_7^1, \mu_8^1 \rangle \tag{5.1}$$

$$\chi_{\mu_2} = \langle \mu_1^2, \mu_2^2, \mu_3^2, \mu_4^2, \mu_5^2, \mu_6^2, \mu_7^2, \mu_8^2 \rangle \tag{5.2}$$

El operador se desempeña de la siguiente manera:

`Paso 1.` El primer individuo hijo se obtiene con la llamada a `función-CI`$(\{\mu_j^1\},\{\mu_j^2\})$ en la línea 3 del Algoritmo 13, la especificación de `función-CI` se muestra en el Algoritmo (14) para obtener la partición $\{C_{\mu_j}\}$.

`Paso 2.` Partiendo de los centroides $\{\mu_j^1\}$ (5.1) para agrupar a $\{\mu_j^2\}$ (5.2) (línea 1 del Algoritmo 13) como si fueran instancias, se obtiene la partición descrita en (5.3), la cual se puede corroborar con la Figura 5.1a.

`Paso 3.` La aleatoriedad del operador es implementada en la función `desunionCentroides` (Algoritmo 15), la cual es utilizada en el Algoritmo 14, de tal manera que permita obtener diversidad de individuos como sucede en la evolución biológica. La función hace un borrado temporal de genes aleatoriamente y se agregan posteriormente al cromosoma hijo donde son seleccionados con una distribución basada en la distancia entre los centroides y una constante de cercanía, tal y como se muestra en las líneas de código 14 a 16 del Algoritmo 13.

$$
\begin{aligned}
C_{\mu_1^1} &= \{\mu_5^2\},\\
C_{\mu_2^1} &= \{\mu_3^2\},\\
C_{\mu_3^1} &= \{\mu_7^2\},\\
C_{\mu_4^1} &= \{\},\\
C_{\mu_5^1} &= \{\},\\
C_{\mu_6^1} &= \{\mu_2^2,\mu_8^2\},\\
C_{\mu_7^1} &= \{\mu_6^2\} \text{ y}\\
C_{\mu_8^1} &= \{\mu_4^2,\mu_1^2\}
\end{aligned}
\tag{5.3}
$$

`Paso 4.` En el ejemplo, los genes de los grupos $C_{\mu_6^1} = \{\mu_2^2,\mu_8^2\}$ y $C_{\mu_8^1} = \{\mu_4^2,\mu_1^2\}$ de (5.3) son candidatos a ser eliminados temporalmente, si el gen μ_2^2 es seleccionado la partición para continuar con el operador de cruce sería (5.4).

$$
\begin{aligned}
C_{\mu_1^1} &= \{\mu_5^2\},\\
C_{\mu_2^1} &= \{\mu_3^2\},\\
C_{\mu_3^1} &= \{\mu_7^2\},\\
C_{\mu_4^1} &= \{\},\\
C_{\mu_5^1} &= \{\},\\
C_{\mu_6^1} &= \{\mu_8^2\},\\
C_{\mu_7^1} &= \{\mu_6^2\} \text{ y}\\
C_{\mu_8^1} &= \{\mu_4^2, \mu_1^2\}
\end{aligned}
\tag{5.4}
$$

`Paso 5.` La estrategia del operador `cruce-CI` para construir los individuos hijos considera tres casos basados en la cardinalidad de los grupos de las particiones. Con esta información se realizará el intercambio de genes, los cuales se describen a continuación:

`Caso 1,` es cuando se tiene una relación de cardinalidad *uno a cero*, no se tiene ningún centroide, los cuales pueden ser candidatos potenciales para formar parte del nuevo individuo hijo para crear un grupo que no existía.

En el ejemplo ilustrativo, en la partición $\{C_{\mu_j}\}$ (5.4) los genes μ_4^1 y μ_5^1 están en este caso.

`Caso 2,` es cuando se tiene más de un centroide, una relación de *uno a muchos*, los cuales pueden ser reducidos por el gen que los agrupa.

En el ejemplo, para (5.4), μ_8^1 puede reemplazar a ambos genes μ_4^2 y μ_1^2.

`Caso 3,` ocurre cuando los genes se agrupan con una cardinalidad de *uno a uno*. A pesar de que los genes tienen diferente índice, son equivalentes. Como se confirma el grupo en ambos cromosomas, pasa a ser conservado como parte del individuo hijo.

En el ejemplo, para (5.4) los genes equivalentes en ambos padres son $\mu_1^1 = \mu_5^2$, $\mu_2^1 = \mu_3^2$, $\mu_3^1 = \mu_7^2$ y $\mu_7^1 = \mu_6^2$.

`Paso 6.` Para mantener la consistencia de los individuos hijos con k grupos, se realiza un conteo de genes que pueden ser intercambiados. Esto se realiza en la `función-CI` dentro del ciclo `for` en las líneas 3 a 5 con las variables `numGenesFaltan` y `numGenesAgrupar`. Si las variables terminan con un valor diferente de genes que van a ser intercambiados al

finalizar el ciclo, el resultado es ajustado con el condicionante `if` en las líneas 9 a 13. En la variable `numGenesCruce` es almacenado el número de genes que se *reciben* y *reducen*.

`Paso 7.` En el ejemplo, al aplicar el procedimiento anterior para (5.4), las variables obtienen los valores de `numGenesFaltan` $= 2$ y `numGenesAgrupar` $= 1$. Por consiguiente, el valor de la variable `numGenesCruce` debe ser ajustado, afirmándose así un valor de 1, la interpretación es que un gen se elimina y otro se agrega.

`Paso 8.` Si `numGenesCruce` ≥ 1, es posible obtener un individuo hijo diferente. Para construirlo primero se genera un número aleatorio con una distribución aleatoria en el intervalo $[1,k]$ que marca el índice en la cadena de genes donde inicia el intercambio de genes, este paso es especificado en la línea 16 del Algoritmo 13. `bGenACruce` es un vector de valores booleanos utilizado para manejar la integridad en el cruce de genes. Cuando `bGenACruce` $=$ `true` se realiza el intercambio del gen.

`Paso 9.` Para finalizar el ejemplo ilustrativo, como `numGenesCruce` es 1 para $\{C_{\mu_j}\}$ (5.4), solamente μ_4^1 o μ_5^1 pasarán a formar parte del nuevo individuo dependiendo del punto de cruce de j. Si $j = 5$, el cromosoma del individuo hijo está dado por (5.5), de manera gráfica se muestra en la Figura 5.1b.

$$\chi'_{\mu_1} = \langle \mu_5^2, \mu_3^2, \mu_7^2, \mu_5^1, \mu_2^2, \mu_8^2, \mu_6^2, \mu_8^1 \rangle \tag{5.5}$$

`Paso 10.` El segundo individuo hijo es obtenido con el mismo procedimiento descrito anteriormente. Con la llamada a la `función-CI`$(\{\mu_j^1\},\{\mu_j^2\})$ línea 4 del Algoritmo 13, se toman los centroides $\{\mu_j^2\}$ para agrupar a $\{\mu_j^1\}$. En el ejemplo, la partición obtenida utilizando los cromosomas de los padres (5.1) y (5.2) está descrita en (5.6).

$$\begin{aligned}
C_{\mu_1^2} &= \{\} \\
C_{\mu_2^2} &= \{\mu_6^1\}, \\
C_{\mu_3^2} &= \{\mu_2^1, \mu_5^1\}, \\
C_{\mu_4^2} &= \{\mu_8^1\} \\
C_{\mu_5^2} &= \{\mu_1^1\}, \\
C_{\mu_6^2} &= \{\mu_7^1\}, \\
C_{\mu_7^2} &= \{\mu_3^1, \mu_4^1\} \text{ y} \\
C_{\mu_8^2} &= \{\}
\end{aligned} \tag{5.6}$$

`Paso 11.` Si el valor de `numGenesCruce` es igual a 2, el segundo hijo estará codificado por cromosoma (5.7). La Figura 5.1b lo muestra de manera gráfica.

$$\chi'_{\mu_2} = \langle \mu_1^2, \mu_6^1, \mu_3^2, \mu_8^1, \mu_1^1, \mu_7^1, \mu_7^2, \mu_8^2 \rangle \tag{5.7}$$

Algoritmo 15 `desunionCentroides`

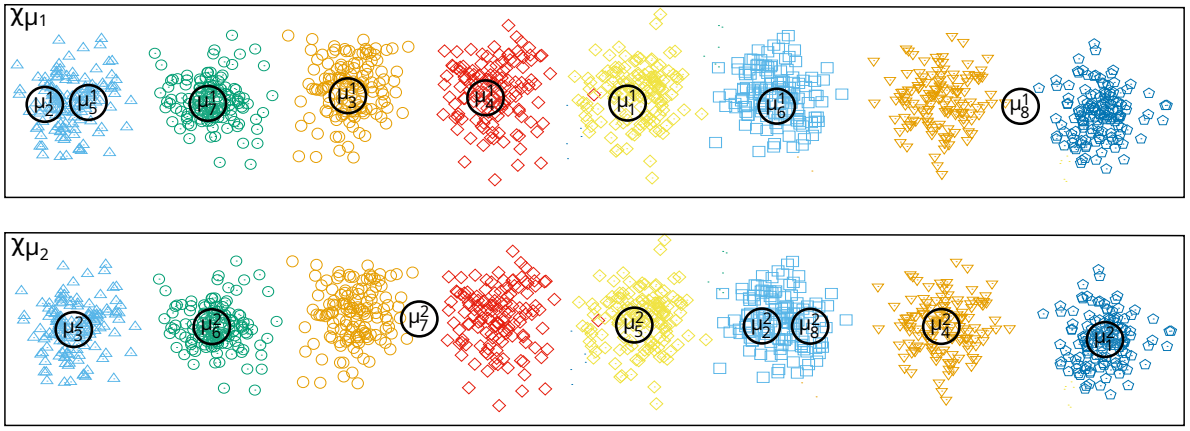
Entrada: $\{\mu_j^1\}, \{\mu_j^2\}$ dos conjuntos de centroides,
$\{C_{\mu_j}^2\}$ una partición de $\{\mu_j^2\}$ asociada a los centroides $\{\mu_j^1\}$ y
constanteCercanía = 0.6

Salida: `map`$\{(\mu_j^1, \mu_j^2)\}$ pares de centroides que son eliminados de la partición de manera aleatoria
$\{C_{\mu_j}^2\}-$ `map`$\{(\mu_j^1, \mu_j^2)\}$ la partición sin un subconjunto de $\{\mu_j^2\}$ seleccionados con una distribución de distancia

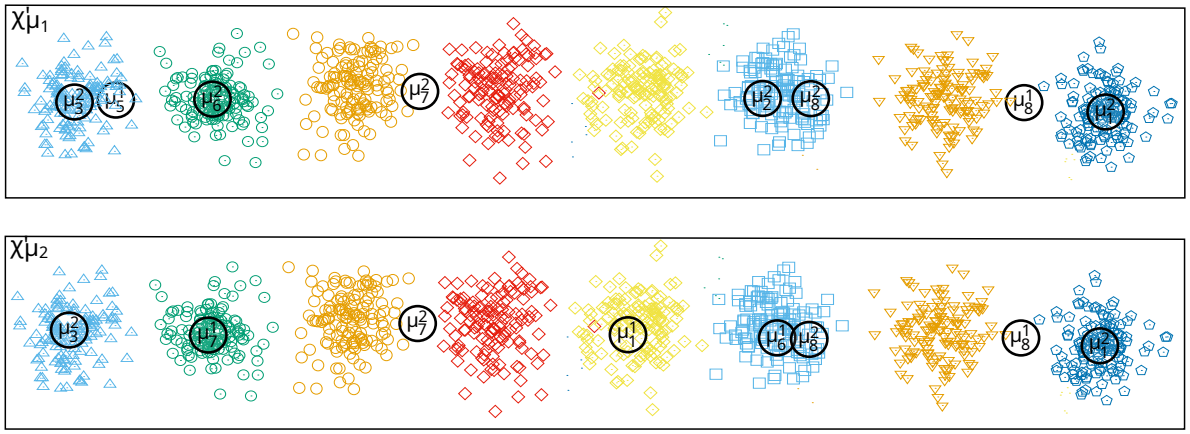
1: **for** $j \leftarrow 1$ to k **do**
2: **if** $|C_{\mu_j}^2| > 1$ **then**
3: minDist $\leftarrow \infty$
4: maxDist $\leftarrow -\infty$
5: **for** $\mu_j^2 \in C_{\mu_j}^2$ **do**
6: dist $\leftarrow \|\mu_j^1 - \mu_j^2\|$
7: **if** dist < minDist **then**
8: minDist $\leftarrow$ dist
9: **end if**
10: **if** maxDist < dist **then**
11: maxDist $\leftarrow$ dist
12: **end if**
13: $r \leftarrow$ número aletorio $\in [0,1]$
14: **if** constanteCercanía (minDist / maxDist) $< r$ **then**
15: `map`$\{\cdot\} \bigcup (\mu_j^1, \mu_j^2)$
16: $C_{\mu_j}^2 \leftarrow C_{\mu_j}^2 - \mu_j^2$
17: **end if**
18: **end for**
19: **end if**
20: **end for**

Después de haber descrito el operador de cruce en los párrafos anteriores, lo siguiente es definir la probabilidad con que se aplicará en cada generación del ciclo evolutivo. En la mayoría de los AGs, se aplica tomando pares de individuos de la población y con una probabilidad definida p_c, se decide si el operador es aplicado o no para cada par de padres.

En GAMK se optó por una probabilidad adaptativa como en [18, 117]. El procedimiento consiste en calcular la probabilidad de realizar el cruce considerando la función aptitud tanto



(a) Cromosomas padre



(b) Cromosomas hijo

Figura 5.1 Ejemplo de la aplicación del operador de cruce GAMK

de la población en conjunto, como la de los individuos que participan. La probabilidad aumenta cuando existen posibilidades de obtener mejores resultados en la reproducción, pero se mantiene la heurística al seguir siendo un proceso probabilístico. La ventaja de usar probabilidad adaptativa es la reducción del tiempo de ejecución del algoritmo, al no cruzar los casos donde posiblemente no lleguen a mejorar las soluciones.

Debido a que se prefiere un cruce elitista, el mejor individuo encontrado hasta la generación actual es utilizado como un padre y el otro es tomado de la población. La ecuación para el cálculo de p_c por [18, 117], aquí se reformuló de la siguiente manera, sea $\mathscr{F}_{\max}$ y $\bar{\mathscr{F}}$ el valor máximo y promedio de la población actual respectivamente y $\mathscr{F}(\chi_t)$ el valor de la función del individuo a cruzar, entoces p_c es calculada con la ecuación (5.8).

$$p_c = \begin{cases} k_1 \times \frac{\mathscr{F}_{\max} - \mathscr{F}(\chi_\iota)}{\mathscr{F}_{\max} - \bar{\mathscr{F}}} & \text{si } \mathscr{F}(\chi_\iota) > \bar{\mathscr{F}} \\ k_3 & \text{si } \mathscr{F}(\chi_\iota) \leq \bar{\mathscr{F}} \end{cases} \tag{5.8}$$

Donde las valores k_1 y k_3 son iguales a 1 propuesto en [18, 117]. Se puede ver que cuando $\mathscr{F}(\chi_\iota)$ está cercano a $\mathscr{F}_{\max}$, los cromosomas son más similares y la posibilidad de poder encontrar una mejora se restringe, por consiguiente, es mejor disminuir p_c.

5.3.3 Operador mutación-RA

El objetivo del operador de mutación es introducir variabilidad en la población. Los cromosomas de los individuos son alterados al modificar uno o más genes, el efecto que se busca es lograr explorar el espacio de todas las posibles soluciones.

El operador de mutación propuesto en GAMK es llamado `mutación-RA`. Su nombre viene de dos operaciones fundamentales que se llevan a cabo, la primera hace referencia a *reducir* el número de grupos y la segunda a *ajustar* a la misma k pero con otra distribución de centroides. De este modo, se logra mover los genes por los posibles valores d-dimensionales en $\mathbb{R}^d$ del conjunto de datos, resultando un individuo mutado que representa una solución diferente.

El operador `mutación-RA` es especificado en el Algoritmo (16), recibe como entrada un cromosoma χ_μ (3.7), pero también puede ser aplicado a otras representaciones basadas en centroides. La descripción del operador es la siguiente:

(a) Inicia decodificando el cromosoma a mutar en la línea 1.

(b) Lo siguiente consiste en seleccionar un centroide para aplicar el proceso de *reducir*, así la solución estará formada de $k-1$ grupos temporalmente.

Para la selección del centroide se analizaron diferentes maneras, siendo la más eficiente una selección guiada por los genes del mejor individuo obtenido hasta la generación actual y con los genes del individuo a mutar. El objetivo es obtener un individuo con características diferentes tanto del mejor como el que se está mutando. El procedimiento es realizado en `suggestedNewGenes`, el cual obtiene un par de centroides próximos μ_{j_1} y μ_{j_2} (línea 3 del Algoritmo 16), representados por los centroides de $\{\mu_j\}_{\text{\_best}}$. Pueden existir varios pares, si ese es el caso uno de ellos es obtenido aleatoriamente, el par final es promediado ($\mu_{\overline{j_1 j_2}}$), para sustituirlo en la línea 6 y obtener $k-1$ grupos.

Cuando no se logra obtener el gen a mutar por el procedimiento `suggestedNewGenes`, uno es seleccionado de manera aleatoria y eliminado en la línea 8 y 9 del Algoritmo 16,

o también, en el caso particular de una agrupación de 2 grupos, en las líneas 12 y 13 del Algoritmo 16 uno es seleccionado y eliminado.

(c) Después de reducir el cromosoma a $k-1$ grupos, sus centroides son ubicados para tener un mejor cubrimiento sobre el conjunto de datos, mediante el algoritmo de `K-means` con re-muestreo (Algoritmo 4) en la línea 16 del Algoritmo 16.

(d) Para completar el operador, el proceso de *ajustar* es realizado al aumentar a k centroides el individuo. El cual consiste en seleccionar un centroide y duplicarlo. Ahora las instancias del grupo son distribuidas entre los dos centroides mediante una actualización de centroides en cada iteración que sea múltiplo de $\sqrt{n}$ con el Algoritmo `updateCentroidsSqrtN` (17), en la línea 21.

Para la selección del centroide a duplicar, se analizaron diferentes estrategias, de todas ellas no se encontró una con diferencias significativas. De este modo, se optó por la basada en ruleta con probabilidad proporcional al número de instancias en cada grupo como se muestra en las líneas 17 y 18 del Algoritmo 16.

Durante el desarrollo del operador, también se analizó *reducir* a un número de grupos menor a $k-1$ de manera recursiva, para buscar una distribución diferente de centroides, pero se concluyó que es suficiente disminuir un gen e incrementar uno en cada mutación.

Similar a la aplicación del operador de cruce, el operador de `mutación-RA` también se aplica con una probabilidad adaptativa como en [18, 117]. La ecuación para obtener la probabilidad de mutación, (p_m), es determinada por la ecuación (5.9).

$$p_m = \begin{cases} k_2 \times \frac{\mathscr{F}_{\max} - \mathscr{F}(\chi_\iota)}{\mathscr{F}_{\max} - \bar{\mathscr{F}}} & \text{si } \mathscr{F}(\chi_\iota) > \bar{\mathscr{F}} \\ k_3 & \text{si } \mathscr{F}(\chi_\iota) \leq \bar{\mathscr{F}} \end{cases} \tag{5.9}$$

Donde las valores k_2 y k_4 son iguales a 0.5 propuesto en [18, 117].

5.3.4 Operador de búsqueda local

El procedimiento de búsqueda local en el AGs permite intensificar la búsqueda. Al ser un procedimiento costoso computacionalmente, la mayoría de los AGs utilizan el algoritmo de `K-means` (1) como operador de búsqueda local aplicando solo una iteración [7, 18, 84], en [41] establece que es suficiente utilizar dos iteraciones.

Es difícil determinar de manera general un número de iteraciones apropiado porque depende de varios factores: número de instancias n, de la estructura del conjunto de datos, del número

Algoritmo 16 Operador `mutación-RA`

Entrada: $\chi_\mu = \langle \mu_j \rangle$: un cromosoma basado en centroides, donde $j : j \in \{1,2,...,k\}$
X: un conjunto de datos
Salida: $\chi'_\mu = \langle \mu'_j \rangle$: un cromosoma mutado, donde $j' : j' \in \{1,2,...,k\}$

1: $\{\mu_j\} \leftarrow$ `decodificar`(χ_μ)
2: **if** $|\{\mu_j\}| > 2$ **then** ▷ Seleccionar un gen reducir a $k-1$ centroides
3: $\{j_1, j_2\} \leftarrow$ `suggestedNewGenes`$(\{\mu_j\}, \{\mu_j\}_{\text{\_best}})$
4: **if** $j_1 \in \{1,2,...,k\} \&\& j_2 \in \{1,2,...,k\}$ **then**
5: $\mu_{\overline{j_1 j_2}} \leftarrow \frac{\mu_{j_1}+\mu_{j_2}}{2}$
6: $\{\mu_{j'}\} \leftarrow \{\mu_j\} - \mu_{j_1} - \mu_{j_2} + \mu_{\overline{j_1 j_2}}$
7: **else**
8: $r_j \leftarrow$ número aleatorio $\in [1,k]$
9: $\{\mu_{j'}\} \leftarrow \{\mu_j\} - \mu_{r_j}$
10: **end if**
11: **else**
12: $r_j \leftarrow$ número aleatorio $\in [1,k]$
13: $\{\mu_{j'}\} \leftarrow \{\mu_j\} - \mu_{r_j}$
14: **end if**
▷ Obtener por la ecuación (5.10) el número de iteraciones para afinar la solución
15: `numIterKmeans` $\leftarrow$ (`numIterKmeans` $-1 > 1$)?`numIterKmeans` -1:1
16: $\{\mu_{j'}\}, \{C_{j'}\} \leftarrow$ `kmeansAlreadyInitResample`$(\{\mu_{j'}\}, X,$`numIterKmeans`$)$
17: $F(\{C_{j'}\}) = \frac{\Sigma_{j'=1}^{j'} |C_{j'}|}{\Sigma_{j'=1}^{|\{C_{j'}\}|} |C'_j|}$
18: $r_{j'} \leftarrow$ `ruleta`$(F(\{C_{j'}\}))$ ▷ Obtener un gen con ruleta proporcional al tamaño del grupos
19: $\mu_{k'} \leftarrow \mu_{r_{j'}}$
20: $\{\mu_{j'}\} \leftarrow \{\mu_{j'}\} \cup \mu_{k'}$ ▷ Aumentar un gen, para tener k centroides
21: $\{\mu_{j'}\} \leftarrow$ `updateCentroidsSqrtN`$(\{\mu_{j'}\}, X, \{r_{j'}, k'\})$
22: $\chi'_\mu \leftarrow$ `codificar`$(\{\mu_{j'}\})$

de grupos a construir y otros. Por otro lado, si es utilizado un número grande de iteraciones puede converger rápidamente y no necesariamente en un óptimo global.

En el algoritmo GAMK, a diferencia de la mayoría de AGs, se ha determinado el número de iteraciones en la búsqueda local por la ecuación (5.10). Esta ecuación se ha obtenido basada en la experimentación, dependiente del valor de k, así se obtiene el parámetro `numIterKmeans` sin forzar la convergencia.

El algoritmo utilizado para la búsqueda local está basado en `K-means` (1) y se ha llamado `kmeansAlreadyInitResample` (ver Algoritmo (18)). La diferencia principal es el proceso de asignación de nuevos grupos cuando se obtienen grupos vacíos. En el ciclo de las líneas 13 a 19 son detectados cuando $|C_j^{(t+1)}| = 0$, entonces un nuevo grupo es creado con un centroide

Algoritmo 17 Algoritmo `updateCentroidsSqrtN` auxiliar de `mutación-RA` (16)

Entrada: $\{\mu_j\}$: centroides de los grupos, donde $j : j \in \{1,2,...,k\}$
X: un conjunto de datos
$\{j_1, j_2\}$: índices de los centroides a actualizar
Salida: $\{\mu_j\}$: centroides actualizados
$\{C_j\}$: una partición X asociada a los centroides $\{\mu_j\}$

1: $sn \leftarrow \sqrt{n}$
2: **for** $i \leftarrow 1$ to n **do**
3: $C_j = \{x_i : \|x_i - \mu_j\| \leq \|x_i - \mu_{j'}\|,\ j' = 1,2,...,k\}$ ▷ //Asignar x_i a su grupo
4: **if** $(i \mod (n/sn)) = 0$ **then**
5: **for** $j \leftarrow 1$ to k **do**
6: **if** $j \in \{j_1, j_2\}$ **then**
7: $\mu_j \leftarrow \frac{\sum_{x_i \in C_j} x_i}{|C_j|}$
8: **end if**
9: **end for**
10: **end if**
11: **end for**
12: $\mu_j = \frac{\sum_{x_i \in C_j} x_i}{|C_j|}$

tomando una x_i aleatoria del conjunto de datos, lo cual resulta mejor que iniciar una nueva solución porque la soluciones están más cerca de los óptimos globales.

$$\texttt{numIterKmeans} = \begin{cases} \lfloor \lg(k)/\sqrt{2} \rfloor & \text{si } \lfloor \lg(k)/\sqrt{2} \rfloor > 1 \\ 1 & \text{en otro caso} \end{cases} \tag{5.10}$$

Algoritmo 18 `kmeansAlreadyInitResample`, basado en K-means [83]

Entrada: $\{\mu_j\}$: centroides de los grupos, donde $j : j \in \{1,2,...,k\}$
X: un conjunto de datos
`numMaxIter`: número máximo de iteraciones
Salida: $\{\mu_j\}$: centroides de los grupos actualizados
$\{C_j\}$: una partición de X asociada a los centroides $\{\mu_j\}$

1: t $\leftarrow$ 1
2: Asignar cada instancia x_i a su grupo por:
$$C_j^{(t)} = \left\{x_i : \|x_i - \mu_j\| \leq \|x_i - \mu_{j'}\|,\ j' = 1,2,...,k\right\}$$
▷ //Verificar los grupos tengan elementos
3: **for** $j \leftarrow 1$ to k **do**
4: **if** $|C_j^{(t)}| = 0$ **then**
5: $i \leftarrow$ número aleatorio $\in [1,n]$
6: $\mu_j^{(t)} \leftarrow x_i$
7: $C_j^{(t)} \leftarrow C_j^{(t)} \cup x_i$
8: **end if**
9: **end for**
10: **repeat**
11: Actualizar los centroides con las instancias x_i asignadas al grupo:
$$\mu_j^{(t+1)} = \frac{\sum_{x_i \in C_j^{(t)}} x_i}{|C_j^{(t)}|}$$
12: Asignar cada instancia x_i a su grupo por:
$$C_j^{(t)} = \left\{x_i : \|x_i - \mu_j^{(t+1)}\| \leq \|x_i - \mu_{j'}^{(t+1)}\|,\ j' = 1,2,...,k\right\}$$
▷ //Verificar los grupos tengan elementos
13: **for** $j \leftarrow 1$ to k **do**
14: **if** $|C_j^{(t+1)}| = 0$ **then**
15: $i \leftarrow$ número aleatorio $\in [1,n]$
16: $\mu_j^{(t+1)} \leftarrow x_i$
17: $C_j^{(t+1)} \leftarrow C_j^{(t+1)} \cup x_i$
18: **end if**
19: **end for**
20: t $\leftarrow$ t + 1
21: **until** t > numMaxIter
22: **return** $(\{\mu_j^{(t+1)}\}, \{C_j^{(t+1)}\})$

5.4 Proceso evolutivo de GAMK

GAMK sigue un proceso evolutivo generacional donde una población de individuos de tamaño fijo es inicializada siguiendo el procedimiento especificado de la Figura 5.2, donde el algoritmo `K-means` es usado como optimizador local. La población inicial pasa a ser la población actual

y de esta los padres serán elegidos por selección de ruleta, los descendientes son obtenidos por la reproducción y mutación. La población descendiente será del mismo tamaño que la actual. Finalmente, la nueva población aplicará el optimizador local, según como se ha especificado. Los descendientes pasarán a la siguiente generación utilizando elitismo de la población actual para que los mejores individuos no se pierdan de una generación a otra y reemplazarán a los individuos menos aptos de la población descendiente. El proceso se repite un número de generaciones establecido.

5.5 Parámetros de configuración

En los AGs es importante establecer los parámetros de tamaño de la población, número de generaciones y probabilidad de aplicación de los operadores genéticos de cruce y mutación, por estar relacionados con los recursos computacionales de espacio y tiempo, así como también en la convergencia a una solución óptima.

Lo complicado en el problema de agrupación para encontrar la configuración apropiada radica en las diferencias de tamaño y dimensiones en los conjuntos de datos, además con una tendencia a incrementar con el paso de los años.

Considerando el tamaño de los conjuntos de datos de la Tabla 4.1, se realizaron pruebas para determinar los parámetros utilizados en GAMK, los recomendados son los siguientes:

- Tamaño de la población $N = 100$,
- Número de generaciones $G = 400$

Y para definir la probabilidad de aplicación de los operadores genéticos se utilizó un procedimiento de probabilidad adaptativa, así para obtener p_c de cruce es determinada por la ecuación (6.3) y p_m de mutación es por la ecuación (5.9).

Figura 5.2 Proceso evolutivo GAMK

5.6 Estudio experimental

Para analizar el desempeño de GAMK, se realizó un estudio experimental que utilizó la misma metodología que se realizó en el capítulo 4, donde se analizaron las propuestas de k-fija en la sección 4.2 y se comparó el desempeño de los diferentes AGs. La configuración de los experimentos en lo relativoa los conjuntos de datos, los CVIs utilizados y la configuración de los algoritmos será similar a la descrita en la sección 4.1. De igual modo, para cada CVI evaluado se obtiene el valor medio del test por *conjunto de datos* $\times$ *algoritmo*. Con este resultado se calculan los rangos promedio por métrica, para finalmente concluir con un análisis de meta-rangos [16] utilizando el total de rangos.

Para realizar el estudio fueron seleccionadas las 8 mejores propuestas encontradas en la sección 4.2, y poder aplicar el estadístico de Friedman con corrección Bergmann y Hommel de los p-valores, que es un método más exhaustivo de comparación de resultados entre pares de algoritmos.

Los rangos promedio para los 15 CVIs internos evaluados se muestran en la Tabla 5.1. Analizando los resultados de manera individual para las diferentes métricas, el mejor resultado por CVI es para el algoritmo que tiene el rango más bajo, siendo GAMK el que obtiene más rangos con valores reducidos para el total de métricas, el que le sigue es la propuesta de CBGA. Como se ha mencionado la función de aptitud de los AGs de k-fija normalmente sólo miden compactibilidad, por tanto, se puede establecer que la aplicación de la metodología es equitativa entre los AGs.

Tabla 5.1 Rangos medios de los CVIs internos obtenidos por GAMK y las ocho mejores propuestas de AGs de $k-$fija, utilizando todos los conjuntos de datos

CVI	CBGA	FGKA	GAMK	GA_B	GAs	GKA	HKA	IGKA	KGA
CS measure (CS)	4.888	4.755	4.878	6.059	**4.622**	5.160	5.106	4.862	4.670
Índice Davies-Bouldin (DB)	**3.505**	5.053	3.527	6.729	5.415	6.085	4.654	5.170	4.862
Índice Xie Beni (XB)	3.580	4.803	**3.388**	6.729	5.670	5.686	4.809	5.027	5.309
Índice S Dunn (SD)	**3.489**	4.569	4.452	6.324	5.628	5.239	5.362	4.638	5.298
Índice Dunn index (D)	**4.202**	5.053	4.691	5.426	4.777	5.394	5.622	5.090	4.745
Índice I (I)	3.920	5.106	**3.819**	6.500	5.197	6.085	4.133	5.319	4.920
Score function (SF)	4.234	5.090	4.340	6.388	3.798	5.883	6.404	5.287	**3.574**
Silhouette (S)	3.888	4.644	**3.356**	6.085	5.160	5.186	6.803	4.723	5.154
Simplified Silhouette (SS)	3.920	5.005	**3.899**	6.431	4.644	5.979	5.543	5.356	4.223
C. de relación de varianza (VRC)	**3.426**	4.973	4.468	6.505	5.899	5.729	3.702	4.878	5.402
Índice WB (WB)	3.846	4.654	**3.649**	6.638	6.059	6.069	3.782	4.989	5.314
Suma de dist Euclidianas (SED)	4.197	4.628	4.356	6.218	3.638	5.963	7.202	5.473	**3.324**
Dist. distorsionada (DD)	3.505	4.229	**3.255**	5.888	4.941	5.612	7.941	4.936	4.691
S. del error cuadrático (SSE)	3.468	3.973	**3.106**	6.053	5.309	5.426	7.968	4.739	4.957
Error cuadrado mín func (Jm)	4.500	4.372	4.846	5.064	4.101	5.431	7.489	5.229	**3.968**
Rango de Friedman	2.333	4.000	**2.267**	8.400	5.067	7.333	6.533	5.267	3.800

Tabla 5.2 Rangos medio de los CVIs externos obtenidos por GAMK y las ocho mejores propuestas de AGs de $k-$fija utilizando todos los conjuntos de datos

CVI	CBGA	FGKA	GAMK	GA_B	GAs	GKA	HKA	IGKA	KGA
Pureza	4.394	4.963	**4.34**	5.872	4.468	5.941	5.016	5.601	4.404
Recall	4.287	5.197	**3.963**	5.835	4.989	5.165	5.527	5.181	4.856
Índice Jaccard (J)	4.074	4.926	**4.005**	5.766	4.856	5.638	5.601	5.367	4.766
Índice Rand (Ω)	**4.362**	4.713	4.500	5.894	4.686	5.840	5.101	5.351	4.553
F-measure (F_m)	4.165	4.995	**4.027**	5.824	4.766	5.660	5.585	5.314	4.665
Precisión	4.293	4.601	**4.234**	5.963	4.766	5.910	5.282	5.431	4.521
Índice de centroide vacío (CI_e)	4.351	5.191	4.473	5.628	4.915	6.080	**4.298**	5.495	4.569
Rango de Friedman	1.857	5.286	**1.429**	8.714	4.286	7.857	5.857	6.571	3.143

Para los CVIs externos los resultados se muestran en la Tabla 5.2. Se puede observar que también GAMK obtiene los mejores resultados, al obtener el rango más bajo en estas medidas. El algoritmo CBGA sería la siguiente mejor propuesta.

Después de haber obtenido los rangos medios, sobre estas medidas, se lleva a cabo el test de Friedman, y su corrección con el test de Iman y Davemport para los CVIs internos y externos, los resultados se muestran en la Tabla 5.3. Se puede observar que el test de Friedman rechaza la hipótesis nula, determinando que existen diferencias significativas en el rendimiento de los algoritmos con un nivel de confianza de 99%. Como la hipótesis nula es rechazada, se realiza el test de Friedman a posteriori con la corrección Bergmann y Hommel para un intervalo de confianza $> 95\%$ que nos permita determinar entre cuáles AGs existen diferencias significativas. Los p-valores de este test se muestran en las Tablas 5.4 y 5.5 para las métricas internas y externas, respectivamente. Los valores en letra negrita indican los algoritmos que tienen diferencias significativas en un intervalo de confianza del $> 95\%$. Para facilitar la interpretación de las Tablas 5.4 y 5.5, se construyen las Figuras 5.3 y 5.4 que nos permiten visualizar más fácilmente entre qué propuestas existen diferencias significativas.

Respecto a los CVIs internos, como se puede ver en la Figura 5.3, se tiene el grupo de AGs con mejor desempeño formado por GAMK, CBGA, KGA, FGKA y GAs, siendo los dos primeros los que destacan, el resto de las propuestas se pueden considerar que obtienen significativamente peores valores. Los algoritmos de FGKA, IGKA y GKA aunque tienen un enfoque de diseño similar, existen diferencias entre ellos de desempeño bien definidas. Las peores propuestas son GA_B, GKA y HKA, siendo GA_B diseñado con operadores tradicionales y sin búsqueda local. HKA basados en formar grupos por las instancias representativas se encuentra en el grupo de peores propuestas, lo que puede significar que no existen punto de comparación entre los basados en centroides e instancias representativas.

Para los CVIs externos, como se muestra en la Figura 5.3, los resultados son similares a las internas. GAMK encabeza la lista, los siguientes AGs ocupan el mismo lugar de desempeño con ligeras variaciones, KGA y GAs muestran igual comportamiento, lo mismo FGKA, IGKA y GKA, lo cual debe ser un resultado esperado por que los tres comparten un diseño similar.

Tabla 5.3 Test de Friedman y la corrección de Iman Davenport sobre los rangos medios de GAMK y AGs de *k*-fija

CVI	Test estadístico	df	p-valor
Internas	Test de Friedman χ^2 = 72.907	df = 8	= 1.294e-12
	Corrección de Iman Davenport χ^2 = 21.674	df1 = 8, df2 = 112	<2.2e-16
Externas	Test de Friedman χ^2 = 48.381	df = 8	= 8.354e-08
	Corrección de Iman Davenport χ^2 = 38.1	df1 = 8, df2 = 48	<2.2e-16

En relación a un buen desempeño de un operador de cruce, lo que se busca al aplicar el operador es la capacidad de receptividad de material genético útil en los hijos sea transmitada. De tal manera que se tenga una explotación en la solución del problema de agrupación. De acuerdo con los resultado experimentales, es evidente que la heurística usada por los dos operador genético especializados trabajan en favor de una métrica de agrupación en particular. Por ejemplo, el operador `Cruce-solución` utilizado por CBGA obtiene mejores resultados para el CVI de Criterio de Relación de Varianza o índice CH (19), por otro lado, el operador `cruce-CI` de GAMK le ocurre igual con el índice Silhouette (13). Con esto se puede decir que ambos operadores de cruce contribuye particularmente en dirección a una métrica debido a su insensibilidad al contexto se vuelve más específico. En el caso de los operadores `cruce-solución` y `cruce-CI` por sus resultados se puede establecer que no son insensibles al contexto y ofrecen buenos resultados en la resolución de este problema.

Otra conclusión referente al diseño de un operador genético para agrupación para *k*-fija, aunque el objetivo principal que se busca es optimizar la compactibilidad de los grupos. Por los resultados obtenidos, como se puede ver en la Tabla 5.1, obtener mejores valores en compactibilidad, no significa obtenerlos en otras métricas que tienen este término en la ecuación también resulte mejor el desempeño del operador. Así, es importante considerar en su diseño el componente de separación y que exista un balanceo entre ambos, también, para el caso de operadores para *k*-fija.

Tabla 5.4 p-valores de los rangos promedio de CVIs internos de GAMK y AGs de k-fija obtenidos con el test de Friedman post-hoc corrección Bergmann y Hommel para un intervalo de confianza $> 95\%$

	CBGA	FGKA	GAMK	GA_B	GAs	GKA	HKA	IGKA	KGA
CBGA	n/a	0.830	1.000	**0.000**	0.066	**0.000**	**0.000**	**0.049**	1.000
FGKA	0.830	n/a	0.830	**0.000**	1.000	**0.011**	0.113	1.000	1.000
GAMK	1.000	0.830	n/a	**0.000**	0.066	**0.000**	**0.000**	**0.049**	0.876
GA_B	**0.000**	**0.000**	**0.000**	n/a	**0.014**	1.000	0.619	**0.028**	**0.000**
GAs	0.066	1.000	0.066	**0.014**	n/a	0.281	0.876	1.000	1.000
GKA	**0.000**	**0.011**	**0.000**	1.000	0.281	n/a	1.000	0.349	**0.007**
HKA	**0.000**	0.113	**0.000**	0.619	0.876	1.000	n/a	1.000	0.075
IGKA	**0.049**	1.000	**0.049**	**0.028**	1.000	0.349	1.000	n/a	1.000
KGA	1.000	1.000	0.876	**0.000**	1.000	**0.007**	0.075	1.000	n/a

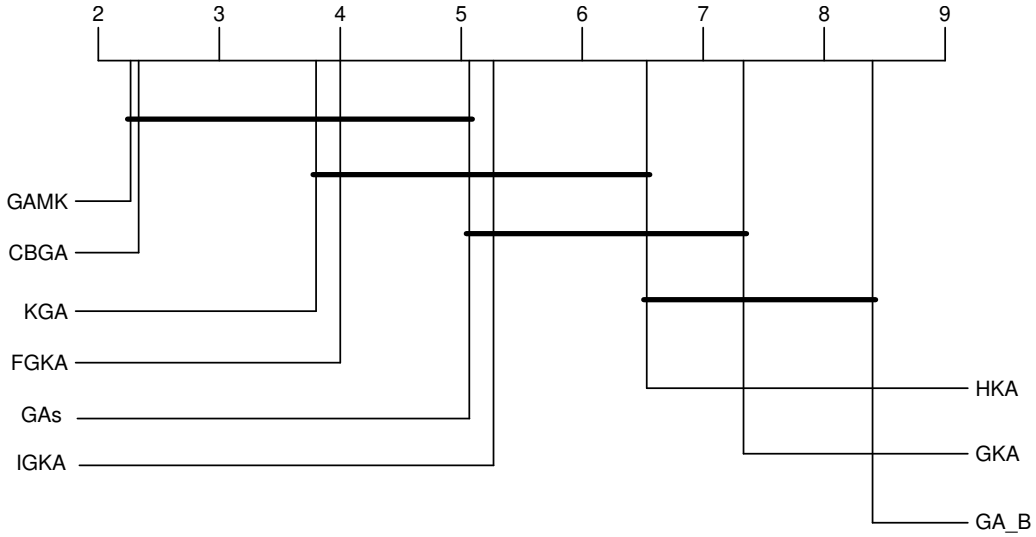


Figura 5.3 Distancia crítica GAMK contra los otros AGs de k-fija obtenida a partir de Tabla 5.4 de p-valores de los CVIs internos

Tabla 5.5 p-valores de los rangos promedio de CVIs externos de GAMK y AGs de k-fija obtenidos con el test de Friedman post-hoc corrección Bergmann y Hommel para un intervalo de confianza $> 95\%$

	CBGA	FGKA	GAMK	GA_B	GAs	GKA	HKA	IGKA	KGA
CBGA	n/a	0.230	1.000	**0.000**	0.711	**0.001**	0.082	**0.020**	1.000
FGKA	0.230	n/a	0.135	0.307	1.000	0.711	1.000	1.000	1.000
GAMK	1.000	0.135	n/a	**0.000**	0.612	**0.000**	**0.045**	**0.010**	1.000
GA_B	**0.000**	0.307	**0.000**	n/a	**0.045**	1.000	0.612	1.000	**0.003**
GAs	0.711	1.000	0.612	**0.045**	n/a	0.191	1.000	1.000	1.000
GKA	**0.001**	0.711	**0.000**	1.000	0.191	n/a	1.000	1.000	**0.020**
HKA	0.082	1.000	**0.045**	0.612	1.000	1.000	n/a	1.000	0.637
IGKA	**0.020**	1.000	**0.010**	1.000	1.000	1.000	1.000	n/a	0.307
KGA	1.000	1.000	1.000	**0.003**	1.000	**0.020**	0.637	0.307	n/a

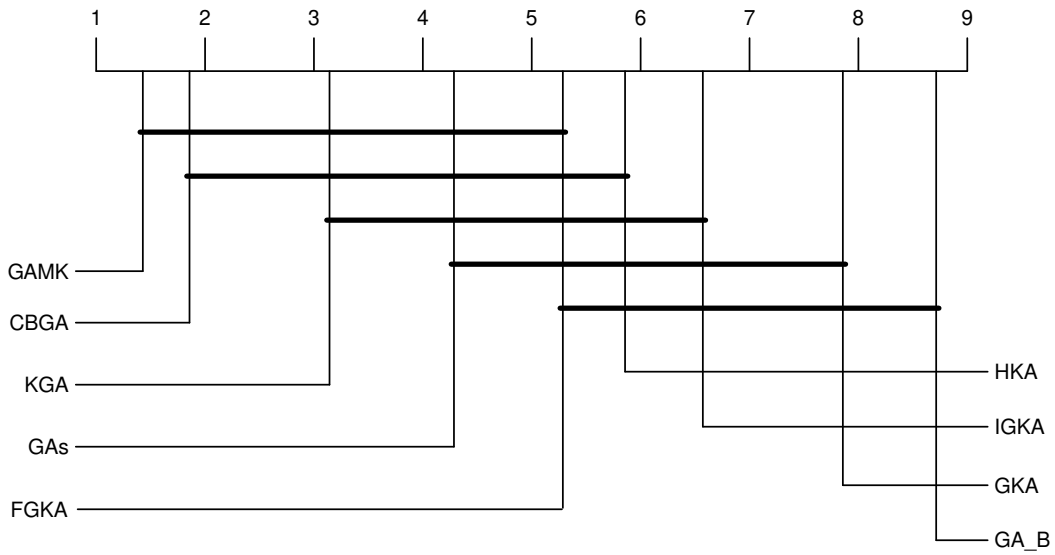


Figura 5.4 Distancia crítica GAMK contra los otros AGs de k-fija obtenida a partir de la Tabla 5.5 de p-valores de los CVIs externos con una confianza $> 95\%$

Capítulo 6

GASGO: Un algoritmo genético para agrupación que estima el número de grupos

En este capítulo se propone un nuevo AG para resolver el problema de agrupación basado en particiones cuando no se conoce el número de grupos que se quiere obtener. Por sus características implementadas se llama Algoritmo Genético Con Operadores Genéticos Especializados (GASGO, Genetic Algorithm with Specialized Genetic Operators).

La propuesta GASGO desarrolla operadores especializados que les permita reducir la codificación redundante y solucionar el problema de la insensibilidad al contexto.

Para el diseño de GASGO utilizó el esquema general de un AG descrito en el Algoritmo (4), considerando sus características particulares. Para su implementación, se usó LEAC [109], que es la biblioteca especializada para construir AGs basados en agrupación en particiones que se ha desarrollado en esta tesis y ya incluye todas las propuestas previas encontradas en la literatura.

En las siguientes secciones se describirán los principales componentes de GASGO y se llevará a cabo un estudio experimental que nos permita analizar el rendimiento de la propuesta realizada comparada con las propuestas previas.

6.1 Codificación de individuos

Los individuos de la población de GASGO utilizan una codificación basada en la *cuantificación vectorial*. Como se comentó en la sección 3.1.1, esta codificación consiste en dividir el espacio de entrada en un número de regiones y cada una de ellas es caracterizada por un vector. Así, un

conjunto de vectores es mapeado a un conjunto más pequeño llamado *libro de código*. Cada libro de código está representado por una combinación de k vectores representativos junto con la partición total de vectores que puede ser especificada por un vector de etiquetas.

Desde el punto de vista del problema de agrupación basado en particiones, un libro de código se puede usar para representar la agrupación de las instancias de un conjunto de datos. Concretamente, un libro de código es utilizado para codificar el cromosoma de cada individuo. Con base a las codificaciones descritas en la 3.1.1, es equivalente a la combinación de dos codificaciones estudiadas, la codificación basada en centroides y la codificación basada en etiquetas con números enteros. De manera formal, el cromosoma se define por χ_{cb} (3.8).

A diferencia de CBGA [41] basado en cuantificación vectorial, en GASGO los individuos codifican soluciones con un número de grupo diferentes, por tanto, el algoritmo debe obtener el óptimo valor de k_v de cada conjunto de datos. De este modo, se diseñan nuevos operadores genéticos que permitan optimizar este valor.

6.2 Inicialización de la población

El algoritmo de agrupación GASGO inicializa una población de tamaño n_p, que se mantiene fija a lo largo de la evolución. Se considera una inicialización aleatoria buscando una mayor diversidad. Este proceso se realiza en dos pasos: inicialización del número de grupos e inicialización de los centroides. A continuación, se describe cada uno de ellos.

Inicialización del número de grupos

La población de individuos con cromosomas $\chi_{cb} = [\{\mu_j\}|\{C_j\}]$ son creados, el tamaño de la parte de los centroides $\{\mu_j\}$ es $k_v \times d$, el valor inicial de número de grupos k_v para cada uno de los individuos es obtenido de manera aleatoria utilizando una distribución uniforme en el intervalo recomendado por la mayoría de los AGs actuales:

$$k_v \in [2, \lceil \sqrt{n} \rceil]$$

La parte de la partición $\{C_j\}$ es de tamaño $k_v + n$ (para detalles ver cromosoma basado en la cuantificación vectorial).

Inicialización de los centroides

Con el espacio asignado para los individuos de la población, se pasa a la inicialización de los centroides de cada cromosoma, seleccionando aleatoriamente tantas instancias del conjunto

de datos como grupos se haya determinado en dicha solución que serán las que se usen para inicializar los centroides de cada grupo.

Una vez inicializado los centroides $\{\mu_j\}$ de cada individuo, la partición $\{C_j\}$ se construye al asignar las instancias al centroide más cercano, obteniéndose la partición óptima asociada a los centroides $\{\mu_j\}$.

6.3 Operadores genéticos

En esta sección se describen los diferentes operadores genéticos que fueron diseñados para el algoritmo GASGO. Para establecer el diseño final del algoritmo se probaron diferentes combinaciones de operadores y funciones objetivo, el desempeño de las diferentes versiones será presentado en la sección 6.6 y con estos resultados se establecerá la versión final del algoritmo.

6.3.1 Operador de selección

Como se comentó en la sección 3.1.3, el proceso de selección es aplicado en cada generación para obtener a los individuos padres, los cuales participan en la aplicación de los operadores genéticos posteriores.

En GASGO la selección se realiza proporcional a la función aptitud utilizando el método clásico de ruleta (3.14).

6.3.2 Operador de cruce-PNN+KV

El operador de cruce de GASGO está basado en el algoritmo de PNN [31], el cual fue utilizado por primera vez como operador de cruce en el algoritmo CBGA de k-fija [38, 41]. En este trabajo las propiedades del algoritmo PNN son estudiadas y personalizadas para ser utilizado también para encontrar el número de grupos óptimo.

El operador de `cruce-PNN+KV` utilizado en GASGO se describe en el Algoritmo (19) y está basado en `CrossSolution` [41], el cual recibe dos cromosomas de tipo libro de códigos $[\{\mu_j\}|\{C_j\}]$ y $[\{\mu_{j'}\}|\{C_{j'}\}]$, cada uno codifica k_v y $k_{v'}$ grupos respectivamente, como resultado se obtiene un nuevo cromosoma descendiente $[\{\mu_{j''}\}|\{C_{j''}\}]$ de $k_{v''}$ grupos.

El operador inicia uniendo los libros de códigos de dos individuos padres. La partición nueva $\{C_{j''}\}$ es construida de acuerdo con las particiones existentes $\{C_j\}$ y $\{C_{j'}\}$ de los padres por `combinarParticiones` línea 10 del Algoritmo (19) al ir asignando los objetos x_i a su grupo por la distancia más pequeña. Después, los centroides son actualizados con la función `actualizarCentroides`. En el operador original, `CrossSolution`, crea inicialmente un

Algoritmo 19 Operador `cruce-PNN+KV`, basado en `CrossSolution` [41]

Entrada: $[\{\mu_j\}|\{C_j\}]$, $[\{\mu_{j'}|\{C_{j'}\}]$: dos cromosomas de libro de códigos
Salida: $[\{\mu_{j''}\}|\{C_{j''}\}]$: un nuevo cromosoma de libro de códigos

1: $\{\mu_{j''}\} \leftarrow$ `combinarCentroides`$(\{\mu_j\},\{\mu_{j'}\})$
2: $\{C_{j''}\} \leftarrow$ `combinarParticiones`$(\{\mu_{j''}\},\{C_j\},\{C_{j'}\})$
3: $\{\mu_{j''}\} \leftarrow$ `actualizarCentroides`$(\{\mu_{j''}\},\{C_{j''}\})$
4: `borrarClustersVacios`$(\{\mu_{j''}\},\{C_{j''}\})$
5: $k_v'' \leftarrow$ número aleatorio $\in [Li,Ls]$ utilizando (6.2)
6: PNN$(\{\mu_{j''}\},\{C_{j''}\},k_v'')$
7: **return** $[\{\mu_{j''}\}|\{C_{j''}\}]$

▷ //Sub-procedimiento juntar los centroides

8: **procedure** `combinarCentroides`$(\{\mu_j\},\{\mu_{j'}\}')$
9: **return** $\{\mu_j\} \cup \{\mu_{j'}\}'$

▷ //Sub-procedimiento crear una nueva partición $\{C_{j''}\}$

10: **procedure** `combinarParticiones`$(\{\mu_{j''}\},\{C_j\},\{C_{j'}\})$
11: **for** $i \leftarrow 1$ to n **do**
12: **if** $\|x_i - \mu_{\alpha_i}\| \leq \|x_i - \mu_{\alpha_i'}\|^2$ **then**
13: $\alpha_i'' \leftarrow \alpha_i$
14: **else**
15: $\alpha_i'' \leftarrow \alpha_i'$
16: **end if**
17: **end for**
18: **return** $\{C_{j''}\}$

▷ //Sub-procedimiento actualiza centroides

19: **procedure** `actualizarCentroides`
20: **for** $j'' \leftarrow 1$ to $|\{\mu_{j''}\}|$ **do**
21: $\mu_{j''} \leftarrow$ `calculaCentroide` $(\{C_{j''}\}, j'')$
22: **end for**
23: **return** $\{\mu_{j''}\}$

▷ //procedimiento PNN adaptado para reducir a k_v'' grupos

24: **procedure** PNN$(\mu_{j''}\},\{C_{j''}\},k_v'')$
25: **for** $j'' \leftarrow 1$ to $|\{\mu_{j''}\}|$ **do**
26: $q_{j''} \leftarrow$ `encontrarVecinosCercanos`$(\mu_{j''})$
27: **end for**
28: **while** $|\{\mu_{j''}\}| > k_v''$ **do**
29: $a \leftarrow$ `encontrarDistanciaMinima`$(\{q_{j''}\})$
30: $b \leftarrow q_a$
31: `unirClusteres`$(\mu_a, P[a], \mu_b, P[b])$
32: `actualizarPunteros`$(\{q_{j''}\})$
33: **end while**
34: **return** $[\{\mu_{j''}\}|\{C_{j''}\}]$

cromosoma de tamaño $2 \cdot k$ y es reducido a k por el procedimiento PNN, donde k es fija. Aquí se actualiza para trabajar con k variable.

En cualquier operador de cruce se busca hacer una *exploración* y *explotación*, en el caso de los algoritmos de agrupación basados en particiones de k-variable consistirá en reducir o aumentar el número de grupos k_v de un cromosoma para ir encontrando el valor óptimo. En el desarrollo de este operador se determinó que la combinación de los dos cromosomas obtendrá un descendiente con un número de grupos más grande que los padres y en el paso siguiente se reducirá a un valor específico de k_v. Para llevar a cabo esta reducción se utilizará el algoritmo de PNN que nos permitirá lograr la exploración y explotación en los individuos. A continuación, se describe el proceso con más detalle.

Sean dos cromosomas padres seleccionados χ_{ce} y χ'_{ce}, cada uno de ellos codifica k_v y k'_v número de grupos, el operador BLX-α [56] se utilizará para obtener el valor k''_v del cromosoma descendiente. El cual es un número seleccionado aleatoriamente de manera uniforme en el intervalo:

$$[c_{\min} - I \cdot \alpha, c_{\max} + I \cdot \alpha] \tag{6.1}$$

Donde $c_{\min} = \min(k_v, k'_v)$, $c_{\max} = \max(k_v, k'_v)$, $I = c_{\max} - c_{\min}$ y $\alpha = 0.5$ recomendado por [56]. Debido a que este operador es usado en el dominio de los números reales, aquí debe ser limitado a números naturales, y además debe cumplirse que $k''_v \in [k_{\min}, k_{\max}]$, $k_{\min} = 2$ y $k_{\max} = \lceil \sqrt{n} \rceil$. Por tanto, el intervalo (6.1) de BLX-α es personalizado utilizando la ecuación 6.2.

$$[Li, Ls] \tag{6.2}$$

Donde:

$$L_i = \begin{cases} 2 & \text{si } c_{\min} - \lceil I \cdot \alpha \rceil < 2 \\ c_{\min} - \lceil I \cdot \alpha \rceil & \text{en otro caso} \end{cases}$$

$$L_s = \begin{cases} c_{\max} + \lceil I \cdot \alpha \rceil & \text{si } c_{\max} + \lceil I \cdot \alpha \rceil < \lceil \sqrt{n} \rceil \text{ y } c_{\min} + c_{\max} \geq c_{\max} + \lceil I \cdot \alpha \rceil \\ \lceil \sqrt{n} \rceil & \text{si } c_{\max} + \lceil I \cdot \alpha \rceil \geq \lceil \sqrt{n} \rceil \text{ y } c_{\min} + c_{\max} \geq \lceil \sqrt{n} \rceil \\ c_{\min} + c_{\max} & \text{en otro caso} \end{cases}$$

Finalmente, para completar la descripción del operador `cruce-PNN+KV`, en la línea (5) es seleccionado un número aleatorio k_v'' y con el procedimiento `PNN` se finaliza el cruce al obtener k_v'' número de grupos para el individuo descendiente.

Para determinar la probabilidad de cruce, fueron probadas dos opciones, fija y adaptativa. La probabilidad adaptativa es la utilizada en [18, 117] para calcular la probabilidad de cruce (p_c), sea $\mathscr{F}_{\max}$ y $\bar{\mathscr{F}}$ el valor máximo y promedio de la población actual respectivamente y $\mathscr{F}'$ el valor aptitud más grande de los dos individuos a cruzar, p_c está dada por la ecuación (6.3).

$$p_c = \begin{cases} k_1 \times \frac{\mathscr{F}_{\max} - \mathscr{F}'}{\mathscr{F}_{\max} - \bar{\mathscr{F}}} & \text{si } \mathscr{F}' > \bar{\mathscr{F}} \\ k_3 & \text{si } \mathscr{F}' \leq \bar{\mathscr{F}} \end{cases} \tag{6.3}$$

Donde las constantes k_1 y k_3 son iguales a 1 propuestas en [18, 117] y establecen que cuando $\mathscr{F}'$ está cercano a $\mathscr{F}_{\max}$, los cromosomas son más similares y la posibilidad de poder encontrar una mejora disminuye por tanto es mejor disminuir p_c.

6.3.3 Operador de mutación

El objetivo del operador de mutación es introducir variabilidad en la población permitiendo explorar otros espacios de soluciones y haciendo que los individuos puedan escapar de máximos locales. En el diseño de GASGO se proponen dos operadores de mutación con este objetivo como prioridad:

- `mutación-R`: hace el reemplazo de un individuo de la población por uno nuevo.
- `mutación-SJ+RA`: consiste en partir y unir grupos. A diferencia de los operadores existentes se incluye una nueva opción que es realizar una nueva distribución manteniendo el número de grupos.

A continuación, se describe cada uno de estos operadores.

Operador de mutación-R

Este operador hace una variación completa del individuo a mutar en la población para el problema de agrupación de k-variable, reemplazando al individuo por un nuevo con un número de grupos seleccionado aleatoriamente. Es equivalente a reiniciar un individuo.

Para analizar el desempeño de este operador se utilizó una probabilidad de mutación fija.

Operador de mutación-SJ+RA

Este operador selecciona de forma aleatoria uno de los grupos representados por el individuo y plantea tres posibles opciones: partirlo (aumentar k_v), juntarlo con otro (disminuir k_v) o mantener el número de grupo, pero cambiar su distribución.

La resolución del problema de agrupación cuando se desconoce el valor de k tiene dos objetivos que resolver: encontrar la k_v óptima y obtener la partición óptima para esa k_v. En [2, 3, 58, 60, 78], los operadores de mutación a lo largo de la evolución enfatizan una búsqueda para una k_v diferente. En el diseño de este operador se intenta cubrir al mismo tiempo los dos objetivos del problema de agrupación de k-variable mientras transcurre la evolución. De tal manera que el operador de mutación proporcione el suficiente material genético para que el de cruce tenga la posibilidad de encontrar la partición óptima buscando una conjunción entre los dos operadores.

La especificación del operador `mutación-SJ+RA` se muestra en el Algoritmo (20), el cual recibe un cromosoma basado en libro de código y regresa el cromosoma mutado. En la línea 2, la función `Mühlenbein` obtiene el valor aleatorio de k'_v a mutar con el operador Mühlenbein [56], mejorando el resultado con la variante propuesta en [48], el cual es definido por la ecuación (6.4).

$$\gamma = \sum_{i=a+1}^{b} \alpha_i 2^{-i} \tag{6.4}$$

Donde,

$$a = \xi_1 \times \frac{\text{Generación actual}}{\text{Máximo de generaciones}}$$

$$b = a + \xi_2$$

y $\alpha_i \in \{0,1\}$ son generados aleatoriamente con $P(\alpha_i = 1) = \frac{i-a}{2\times\xi_2}$. Los parámetros ξ_1 y ξ_2 deben ser determinados por el usuario [48], en la implementación de este trabajo se usó $\xi_1 = 2$ y $\xi_2 = 8$.

Con el valor obtenido de (6.4), el gen, k_v, número de grupos del cromosoma actual es mutado de acuerdo con la ecuación 6.5.

$$k'_v = k_v \pm \text{range}_i \cdot \gamma \tag{6.5}$$

Donde, $\text{range}_i = k_{\max} - k_{\min}$. El signo $\pm$ es seleccionado con una probabilidad de 0.5. Aquí se personaliza el resultado convirtiendo al dominio de los números naturales: si el signo es positivo, $\lceil k'_v \rceil$; en otro caso, $\lfloor k'_v \rfloor$, y finalmente el valor de k'_v es restringido a $k'_v \in [k_{\min}, k_{\max}]$.

Después de haber descrito la heurística para obtener el valor k'_v por la función `Mühlenbein`, dependiendo del valor actual de k_v, se tendrán tres casos de mutación:

- Si $k'_v < k_v$, se aplica una reducción de grupos utilizando de manera sucesiva la función `borrarCluster` (línea 5) hasta obtener un cromosoma con $k'_v = |\{\mu_{j'}\}|$. La descripción de `borrarCluster` está descrita en el Algoritmo (21). Para la selección de grupos a reducir existen diferentes estrategias, una manera puramente aleatoria EAC [58], con ruleta con la proporción de la función objetivo que aporta cada grupo FEAC [3] o proporcional a alguna propiedad de los grupos como la distancia entre grupos cercanos GCUK[78] $\min_{1 \leq j' \leq k_v, j \neq j'} |\mu_j, -\mu_{j'}|$. Como se puede ver en la Figura 4.5 no existen diferencias significativas entre EAC y FEAC, por tanto, no debe ser un factor determinante. Para la selección de los grupos a eliminar en GASGO se usó la ruleta inversamente proporcional al radio promedio de los grupos.

- Si $k_v < k'_v$, se aplica un aumento de grupos utilizando la función `añadirCluster` (línea 9) hasta obtener un cromosoma con $k'_v = |\{\mu_{j'}\}|$. Como en el caso anterior para seleccionar los grupos a dividir y así aumentar k_v existen varias estrategias, aletoria usada en EAC [58] o ruleta proporcional a $S_{C_j,q}$ (12) usada en GCUK [78]. En este operador se ha usado la ruleta, pero al contrario del caso anterior, proporcional al radio promedio del grupo con el Algoritmo (22).

- Si $k'_v = k_v$, (linea 12), se hace una nueva distribución de las instancias en los grupos manteniendo la misma k_v. Se utiliza la función `reducirAjuste`, descrita en el Algoritmo (23). En esta función se selecciona un grupo y se elimina su centroide. A continuación, se aplica `K-means` para $k'_v - 1$ y se selecciona un nuevo centroide de los $k'_v - 1$ utilizando la función `actualizarCentroideSqrtN` (Algoritmo (17)) para obtener nuevos k'_v grupos. Cuando $k_v = 2$, es un caso especial, en este caso una instancia es seleccionada aleatoriamente para ser utilizada como un nuevo centroide (como se muestra en las líneas 15 a 18).

Para la aplicación de este operador se analizó utilizar tanto una probabilidad fija como una adaptativa utilizando la ecuación (5.9) para obtener la probabilidad de p_m.

Algoritmo 20 Operador `mutación-SJ+RA`

Entrada: $[\{\mu_j\}|\{C_j\}]$: cromosoma basado en libro de códigos
Salida: $[\{\mu_{j'}|\{C_{j'}\}]$: cromosoma mutado

$[\{\mu_{j'}|\{C_{j'}\}] \leftarrow [\{\mu_j\}|\{C_j\}]$
$k'_v \leftarrow$ `Mühlenbein()` ▷ Obtener un número aleatorio basado en (6.5)
if $k'_v < |\{\mu_{j'}\}|$ **then**
    while $k'_v < |\{\mu_{j'}\}|$ **do**
        $\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `borrarCluster`$(\{\mu_{j'}\},\{C_{j'}\})$
    end while
else if $k_v < k'_v$ **then**
    while $k_v < k'_v$ **do**
        $\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `añadirCluster`$(\{\mu_{j'}\},\{C_{j'}\})$
    end while
else ▷ $k_v = k'_v$
    $\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `reducirAjuste`$(\{\mu_{j'}\},\{C_{j'}\})$
end if
return $[\{\mu_{j'}\}|\{C_{j'}\}]$

Algoritmo 21 Algoritmo `borrarCluster` auxiliar de `mutación-SJ+RA` (20)

Entrada: $\{\mu_j\}$: centroides de los grupos, donde $j : j \in \{1,2,...,k_v\}$
$\{C_j\}$: una partición de X asociada a los centroides $\{\mu_j\}$
X: un conjunto de datos
Salida: $\{\mu_{j'}\}$: una nueva distribución de centroide con un menos $j' : j' \in \{1,2,...,k_v-1\}$
$\{C_{j'}\}$: una partición de X asociada a los centroides $\{\mu_{j'}\}$

$\{\mu_{j'}\} \leftarrow \{\mu_j\}$
$\{C_{j'}\} \leftarrow \{C_j\}$
$F(\{C_{j'}\}) = \frac{\sum_{j'=1}^{j'} 1/\text{avgRadius}(C_{j'})}{\sum_{j'=1}^{|\{C_{j'}\}|} 1/\text{avgRadius}(C_{j'})}$ ▷ Hacer una distribución inversamente proporcional al radio promedio de la partición
$r_{j'} \leftarrow$ `ruleta`$(F(\{C_{j'}\}))$ ▷ Seleccionar un centroide a eliminar
$\{\mu_{j'}\} \leftarrow \{\mu_{j'}\} - \mu_{r_{j'}}$
`numIter` $\leftarrow 1$
$\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `kmeansAlreadyInitResample`$(\{\mu_{j'}\},X,$`numIter`$)$
return $[\{\mu_{j'}\}|\{C_{j'}\}]$

Algoritmo 22 Algoritmo `addCluster` auxiliar de `mutación-SJ+RA` (20)

Entrada: $\{\mu_j\}$: centroides de los grupos, donde $j : j \in \{1,2,...,k_v\}$
$\{C_j\}$: una partición de X asociada a los centroides $\{\mu_j\}$
X: un conjunto de datos
Salida: $\{\mu_{j'}\}$: una nueva distribución de los centroides con uno más $j' : j' \in \{1,2,...,k_v+1\}$
$\{C_{j'}\}$: una partición de X asociada a los centroides $\{\mu_{j'}\}$

1: $\{\mu_{j'}\} \leftarrow \{\mu_j\}$
2: $\{C_{j'}\} \leftarrow \{C_j\}$
3: $F(\{C_{j'}\}) = \frac{\sum_{j'=1}^{j'} \text{avgRadius}(C_{j'})}{\sum_{j'=1}^{|\{C_{j'}\}|} \text{avgRadius}(C_{j'})}$
▷ Hacer una distribución proporcional al tamaño de los grupos
4: $r_{j'} \leftarrow$ `ruleta`$(F(\{C_{j'}\}))$ ▷ Seleccionar un centroide para aumentar $k'_v + 1$
5: $\{\mu_{j'}\} \leftarrow \{\mu_{j'}\} \cup \mu_{r_{j'}}$
6: $\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `actualizarCentroidsSqrtN`$(\{\mu_{j'}\}, X, \{r_{j'}, k'\})$
7: **return** $[\{\mu_{j'}\}|\{C_{j'}\}]$

Algoritmo 23 Algoritmo `reducirAjuste` auxiliar de `mutación-SJ+RA` (20)

Entrada: $\{\mu_j\}$: centroides de los grupos, donde $j : j \in \{1,2,...,k_v\}$
$\{C_j\}$: una partición de X asociada a los centroides $\{\mu_j\}$
X: un conjunto de datos
Salida: $\{\mu_{j'}\}$: una nueva distribución de los centroides $j' : j' \in \{1,2,...,k_v\}$
$\{C_{j'}\}$: una partición de X asociada a los centroides $\{\mu_{j'}\}$

1: $\{\mu_{j'}\} \leftarrow \{\mu_j\}$
2: $\{C_{j'}\} \leftarrow \{C_j\}$
3: $F(\{C_{j'}\}) = \frac{\sum_{j'=1}^{j'} 1/|C_{j'}|}{\sum_{j'=1}^{|\{C_{j'}\}|} 1/|C_{j'}|}$ ▷ Hacer una distribución inversamente proporcional al tamaño de los grupos
4: $r_{j'} \leftarrow$ `ruleta`$(F(\{C_{j'}\}))$ ▷ Seleccionar un centroide para reducir a $k'_v - 1$
5: $\{\mu_{j'}\} \leftarrow \{\mu_{j'}\} - \mu_{r_{j'}}$
6: **if** $|\{\mu_j\}| > 2$ **then**
7: `numIter` $\leftarrow$ (`numIter` $-1 > 1$)?`numIter` -1:1 ▷ Obtener por ecuación (5.10)
8: $\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `kmeansAlreadyInitResample`$(\{\mu_{j'}\},X,$`numIter`$)$
9: $F(\{C_{j'}\}) = \frac{\sum_{j'=1}^{j'} |C_{j'}|}{\sum_{j'=1}^{|\{C_{j'}\}|} |C_{j'}|}$
10: $r_{j'} \leftarrow$ `ruleta`$(F(\{|C_{j'}|\}))$
11: $\mu_{k'_v} \leftarrow \mu_{r_{j'}}$
12: $\{\mu_{j'}\} \leftarrow \{\mu_{j'}\} \cup \mu_{k'_v}$ ▷ //Aumentar un gen, para tener k centroides
13: $\{\mu_{j'}\} \leftarrow$ `actualizarCentroidsSqrtN`$(\{\mu_{j'}\},X,\{r_{j'},k'\})$
14: **else**
15: $r_i \leftarrow$ número aleatorio $\in [1,n]$
16: $\mu_{k'_v} \leftarrow x_{r_i}$
17: $\{\mu_{j'}\} \leftarrow \{\mu_{j'}\} \cup \mu_{k'_v}$ ▷ //Aumentar un gen, para tener k centroides
18: $\{\{\mu_{j'}\},\{C_{j'}\}\} \leftarrow$ `kmeansAlreadyInitResample`$(\{\mu_{j'}\},X,$`numIter`$)$
19: **end if**
20: **return** $[\{\mu_{j'}\}|\{C_{j'}\}]$

6.4 Proceso evolutivo de GASGO

GASGO sigue un proceso evolutivo generacional donde una población de individuos de tamaño fijo es inicializada siguiendo el procedimiento especificado en esta sección donde el algoritmo `K-means` es usado como optimizador local. La población inicial pasa a ser la población actual y de ella serán elegidos los padres por selector de ruleta que genera los descendientes utilizando cruce y mutación. En este caso se usa también un operador de búsqueda local (`K-means`) que es incluido en una de las mutaciones que se han desarrollado. Los descendientes pasarán a la siguiente generación utilizando elitismo de la población actual para que los mejores individuos no se pierdan de una generación a otra y reemplazarán a los individuos menos aptos de la población descendiente. El proceso se repite un número de generaciones establecido. El proceso concreto es especificado en la Figura 6.1.

6.5 Parámetros de configuración

En los AGs es importante establecer los parámetros de tamaño de la población, número de generaciones y probabilidad de aplicación de los operadores genéticos de cruce y mutación, por estar relacionados con los recursos computacionales de espacio y tiempo, así como también en la convergencia a una solución óptima.

Lo complicado en el problema de agrupación para encontrar la configuración apropiada radica en las diferencias de tamaño y dimensiones en los conjuntos de datos, además con una tendencia a incrementar con el paso de los años.

Considerando el tamaño de los conjuntos de datos de la Tabla 4.1, los parámetros utilizados en GASGO fueron los siguientes:

- Tamaño de la población $N = 50$,
- Número de generaciones $G = 300$

Y para la probabilidad de aplicación de operadores se utilizaron dos esquemas:

- Probabilidad fija:
  - Probabilidad de cruce $p_c = 0.8$,
  - Probabilidad de mutación $p_m = 0.05$
- Probabilidad adaptativa:
  - Probabilidad de cruce p_c determinada por la ecuación (6.3),
  - Probabilidad de mutación p_m por la ecuación (5.9)

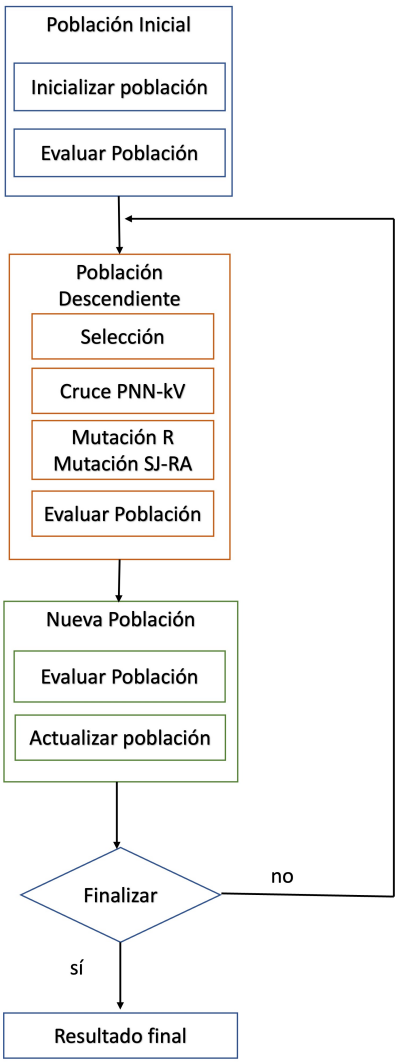


Figura 6.1 Proceso evolutivo GASGO

6.6 Estudio experimental

Es este estudio experimental se va a seguir la misma metodología que se realizó en los estudios anteriores. Concretamente, similar al estudio de las propuestas de *k*-variable en la sección 4.3 donde se comparaba el desempeño de los diferentes AGs. La configuración de los experimentos en lo relativo al conjunto de datos, CVIs, configuración de los algoritmos será similar a la descrita en la sección 4.1. De igual modo que en los estudios anteriores, para cada CVI evaluado se obtiene el valor medio del test por *conjunto de datos* $\times$ *algoritmo*. Con este resultado se calculan los rangos promedio por métrica, para finalmente concluir con un análisis de meta-rangos [16] utilizando el total de rangos.

Tabla 6.1 Rangos medios de los CVIs internos obtenidos por GASGO y las ocho mejores propuestas de AGs de *k*-variable, utilizando todos los conjuntos de datos

CVI	CGA	EAC	$FEAC_{RI}$	$FEAC_{SS}$	GASGO	GCUK	GGA_S	TGCA	VGA
CS measure (CS)	**3.418**	4.391	5.473	4.832	4.283	7.299	4.576	5.630	5.098
Índice Davies-Bouldin (DB)	4.228	3.674	6.505	**3.301**	4.630	4.413	5.060	7.522	5.658
Índice Xie Beni (XB)	**3.674**	4.174	6.788	4.027	4.609	3.505	4.500	7.717	6.005
Índice S Dunn (SD)	**3.082**	4.293	6.815	4.815	4.880	4.717	4.462	7.043	4.891
Índice Dunn index (D)	3.864	5.527	6.076	5.739	**3.359**	5.451	4.011	5.418	5.554
Índice I (I)	2.891	5.353	6.473	5.560	3.875	6.370	5.272	6.446	**2.761**
Score function (SF)	3.739	4.929	5.810	5.283	**3.582**	6.766	3.679	5.929	5.283
Silhouette (S)	**3.092**	5.060	6.158	4.951	3.685	5.527	3.402	6.500	6.625
Simplified Silhouette (SS)	3.636	4.946	6.598	5.141	**3.168**	5.440	3.484	6.592	5.995
C. de relación de varianza (VRC)	4.234	5.864	6.652	5.766	**1.918**	5.185	3.766	6.527	5.087
Índice WB (WB)	4.717	6.163	6.543	5.995	**1.647**	4.783	3.685	6.043	5.424
Suma de dist Euclidianas (SED)	7.098	5.891	4.163	4.880	4.587	**3.207**	4.658	4.978	5.538
Dist. distorsionada (DD)	6.951	5.924	4.207	4.772	4.375	**3.315**	4.712	5.065	5.679
S. del error cuadrático (SSE)	7.016	5.957	4.272	4.875	4.505	**3.196**	4.636	4.967	5.576
Error cuadrado mín func (Jm)	7.250	5.788	3.935	4.946	4.962	**3.163**	4.875	4.875	5.207
Rango de Friedman	4.000	5.467	6.533	4.967	**3.000**	4.267	3.367	7.167	6.233

Para realizar el estudio fueron seleccionadas las 8 mejores propuestas encontradas en la sección 4.3. Se aplica el estadístico de Friedman con corrección Bergmann y Hommel de los *p*-valores, que es un método más exhaustivo de comparación de resultados entre pares de algoritmos.

Los rangos promedio para los 15 CVIs internos evaluados se muestran en la Tabla 6.1. Analizando los resultados de manera individual para las diferentes métricas, cabe esperar que el rango promedio obtenido por un AG de *k*-variable con un CVI específico como función objetivo, sea el mejor clasificado para ese CVI. En el caso de AGs utilizan diferentes CVIs, los rangos estarán repartidos y la hipótesis nula podría ser confirmada. No obstante, nos interesa que en general, en todas las métricas los resultados sean competitivos. En el estudio de rangos, se puede ver que los mejores rangos, por orden, están repartidos entre los algoritmos GASGO, GGA y CGA. El valor de ranking de Friedman aplicado sobre los rankings de cada algoritmo para cada uno de los CVIs analizados es mostrado en la última fila de la Taba 6.1. Recordando este valor nos indicaba cuanto más bajo era, que mejor era el desempeño del AG en las diferentes métricas consideradas.

Una importante caracterización de los CVIs para *k*-variable, son los que se inclinan más por la compactibilidad que por la separación, como se comentó anteriormente por la definición de agrupación debe existir un equilibrio en estas dos propiedades. En este contexto, se puede ver que GCUK que emplea el índice DB (11) como función de aptitud, obtiene los grupos más compactos, pero con menos separación. También, la tabla 6.1 nos da información de las medidas que son optimizadas por los diferentes AGs estando ellas representadas en negrita.

Los rangos promedios para las métricas externas se muestran en la Tabla 6.2. Los mejores rangos son obtenidos por $FEAC_{RI}$, el cual es basado en un CVI externo, este es un resultado que

Tabla 6.2 Rangos medios de los CVIs internos obtenidos por GASGO y las ocho mejores propuestas de AGs de k-variable utilizando todos los conjuntos de datos

CVI	CGA	EAC	$FEAC_{RI}$	$FEAC_{SS}$	GASGO	GCUK	GGA_S	TGCA	VGA
Pureza	6.674	5.707	**3.239**	4.707	4.739	3.614	5.103	5.679	5.538
Recall	**2.696**	4.158	5.212	4.832	4.418	6.745	4.924	6.478	5.538
Índice Jaccard (J)	4.446	5.158	**3.690**	5.424	4.011	5.984	4.668	6.310	5.310
Índice Rand (Ω)	5.516	5.527	**2.478**	5.397	4.255	5.179	4.875	6.005	5.766
F-measure (F_m)	4.413	5.190	**3.685**	5.391	3.967	6.049	4.679	6.272	5.353
Precisión	6.370	6.060	**3.147**	5.402	4.293	3.886	4.799	5.440	5.603
Índice de centroide vacío (CI_e)	5.766	5.174	**2.951**	4.636	5.310	4.005	5.701	6.114	5.342
Rango de Friedman	5.571	5.571	1.714	4.857	3.000	5.000	4.571	8.143	6.571

Tabla 6.3 Test de Friedman y la corrección de Iman Davenport sobre los rangos medios de GASGO y AGs de k-variable

CVI	Test estadístico	df	p-valor
Internas	Test de Friedman $\chi^2 = 33.982$	df = 8	= 4.093e-05
	Corrección de Iman Davenport $\chi^2 = 5.5308$	df1 = 8, df2 = 112	= 6.76e-06
Externas	Test de Friedman $\chi^2 = 26.133$	df = 8	= 0.0009965
	Corrección de Iman Davenport $\chi^2 = 5.25$	df1 = 8, df2 = 48	= 9.587e-05

ya pudimos ver en el análisis de la sección 4.3.2. GASGO no gana ningún índice, pero siempre se encuentra en el segundo lugar de los AGs, en los CVIs externos, lo que es representativo porque es competitivo también en este tipo de CVIs.

Después de haber obtenido los rangos medios, sobre estas medidas, se lleva a cabo el test de Friedman, y su corrección con el test de Iman y Davemport para los CVIs internos y externos y sus resultados se muestran en la Tabla 6.3. Se puede observar que el test de Friedman rechaza la hipótesis nula, determinando que existen diferencias significativas en el rendimiento de los algoritmos con un nivel de confianza de 99%. Como la hipótesis nula es rechazada, se realiza el test de Friedman a posteriori con la corrección Bergmann y Hommel para un intervalo de confianza $> 95\%$ que nos permita determinar entre cuáles AGs existen diferencias significativas. Los p-valores de este test se muestran en las Tablas 6.4 y 6.5 para las métricas internas y externas, respectivamente. Los valores en letra negrita indican los algoritmos que tienen diferencias significativas en un intervalo de confianza del $> 95\%$. Para facilitar la interpretación de estas Tablas, se muestran las Figuras 6.2 y 6.3 que nos permiten visualizar más fácilmente entre qué propuestas existen diferencias significativas.

En la figura 6.2, se puede apreciar como el algoritmo GASGO se impone como la propuesta con mejor ranking, siendo el algoritmo de control. Además, diferencias importantes se aprecian entre los diferentes AGs. Concretamente, se puede apreciar dos grupos de algoritmos entre los que existen diferencias significativas. Los mejores desempeños encabezado por GASGO, son

Tabla 6.4 p-valores de los rangos promedio de CVIs internos de GASGO y AGs de k-variable obtenidos con el test de Friedman post-hoc corrección Bergmann y Hommel para un intervalo de confianza $> 95\%$

	CGA	EAC	$FEAC_{RI}$	$FEAC_{SS}$	GASGO	GCUK	GGA_S	TGCA	VGA
CGA	n/a	1.000	0.181	1.000	1.000	1.000	1.000	**0.034**	0.306
EAC	1.000	n/a	1.000	1.000	0.245	1.000	0.464	1.000	1.000
$FEAC_{RI}$	0.181	1.000	n/a	1.000	**0.011**	0.304	**0.034**	1.000	1.000
$FEAC_{SS}$	1.000	1.000	1.000	n/a	0.788	1.000	1.000	0.445	1.000
GASGO	1.000	0.245	**0.011**	0.788	n/a	1.000	1.000	**0.001**	**0.027**
GCUK	1.000	1.000	0.304	1.000	1.000	n/a	1.000	0.067	0.788
GGA_S	1.000	0.464	**0.034**	1.000	1.000	1.000	n/a	**0.004**	0.067
TGCA	**0.034**	1.000	1.000	0.445	**0.001**	0.067	**0.004**	n/a	1.000
VGA	0.306	1.000	1.000	1.000	**0.027**	0.788	0.067	1.000	n/a

Tabla 6.5 p-valores de los rangos promedio de CVIs externos de GASGO y AGs de k-variable obtenidos con el test de Friedman post-hoc corrección Bergmann y Hommel para un intervalo de confianza $> 95\%$

	CGA	EAC	$FEAC_{RI}$	$FEAC_{SS}$	GASGO	GCUK	GGA_S	TGCA	VGA
CGA	n/a	1.000	0.185	1.000	1.000	1.000	1.000	1.000	1.000
EAC	1.000	n/a	0.185	1.000	1.000	1.000	1.000	1.000	1.000
$FEAC_{RI}$	0.185	0.185	n/a	0.509	1.000	0.397	0.663	**0.000**	**0.025**
$FEAC_{SS}$	1.000	1.000	0.509	n/a	1.000	1.000	1.000	0.446	1.000
GASGO	1.000	1.000	1.000	1.000	n/a	1.000	1.000	**0.012**	0.309
GCUK	1.000	1.000	0.397	1.000	1.000	n/a	1.000	0.509	1.000
GGA_S	1.000	1.000	0.663	1.000	1.000	1.000	n/a	0.323	1.000
TGCA	1.000	1.000	**0.000**	0.446	**0.012**	0.509	0.323	n/a	1.000
VGA	1.000	1.000	**0.025**	1.000	0.309	1.000	1.000	1.000	n/a

seguidos por GGA_S, CGA, GCUK y $FEAC_{SS}$. Es interesante resaltar que la función objetivo utilizada por los AGs de este grupo son diferentes entre ellos. El segundo grupo, entre los que hay diferencias significativas con respecto al rendimiento del grupo comentado anteriormente, por orden de desempeño estaría EAC, VGA, $FEAC_{RI}$ y TGCA. Es interesante resaltar, que este último utiliza la misma función de aptitud, VRC, al igual que GASGO.

Con respecto a los CVIs externos, también se pueden ver diferencias significativas atribuidas a la relación guardada con las métricas internas cuando son utilizadas como función aptitud. Como se puede ver en la Figura 6.3, en el primer grupo en orden de desempeño se encuentra $FEAC_{RI}$ que era un resultado esperado por utilizar el índice Rand, como se comentó en la sección 4.3.2. A continuación, GASGO obtiene una posición representativa, separándose del grupo GGA_S, $FEAC_{SS}$ y GCUK. Los algoritmos entre los que existen diferencias significativas por obtener un peor desempeño que los anteriores, por orden, serían VGA y TGCA. Es importante resaltar que CGA, pierde desempeño cuando se consideran los CVIs externos.

En el caso de GASGO, se va a incluir un estudio del tiempo de ejecución del algoritmo. Los AGs adolecen de necesitar un elevado tiempo de cómputo para lograr optimizar las soluciones. Por ello, en este estudio, se desea resaltar que durante el diseño de este algoritmo siempre se ha diseñado y seleccionado componentes que permitiesen optimizar el tiempo con la finalidad de

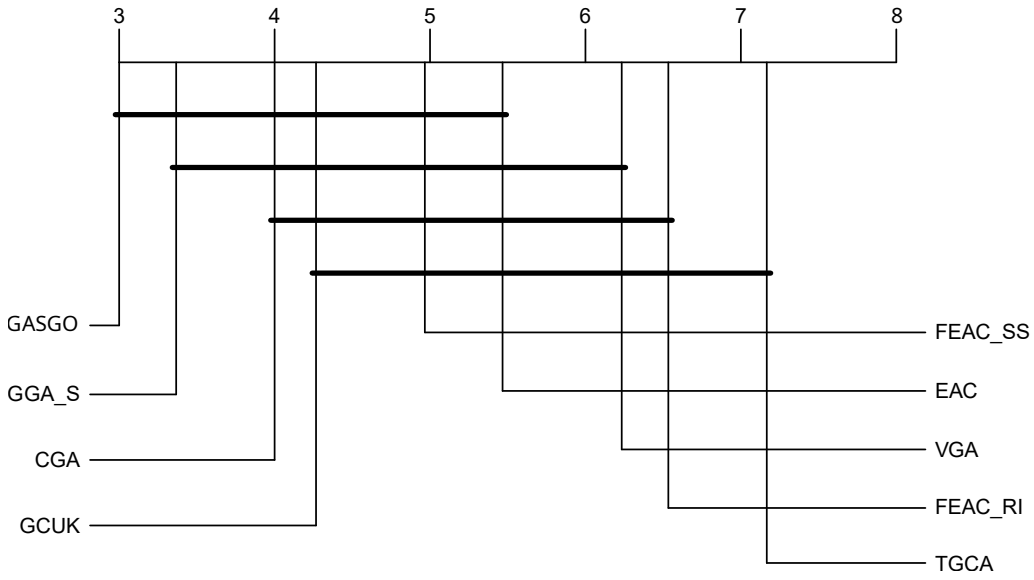


Figura 6.2 Distancia crítica de GASGO contra los otros AGs de k-variable obtenida a partir de la Tabla 6.4 de p-valores de los CVIs internos con una confianza $> 95\%$

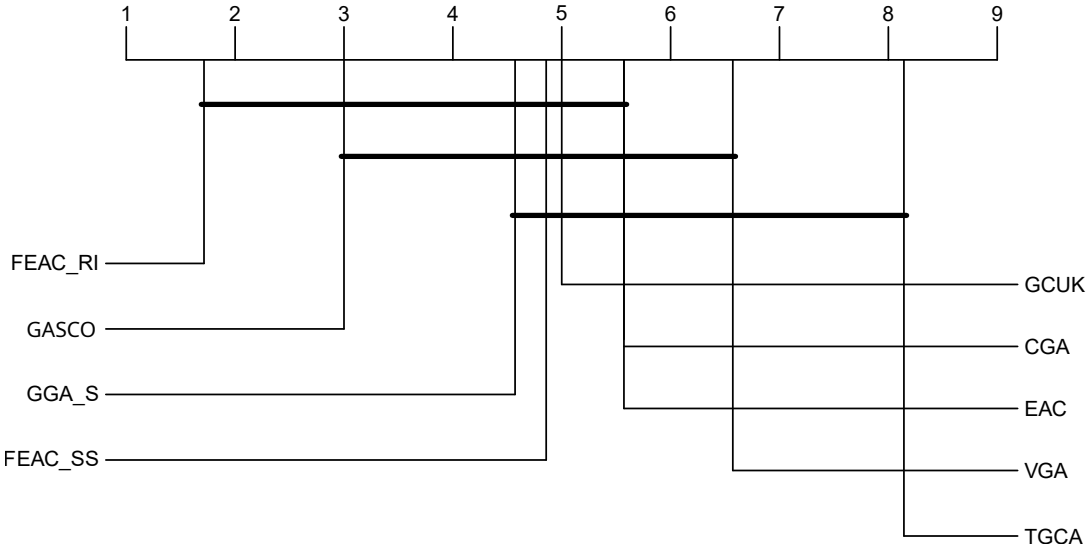


Figura 6.3 Distancia crítica de GASGO contra los otros AGs de k-variable obtenida a partir de la Tabla 6.5 de p-valores de los CVIs externos con una confianza $> 95\%$

poder reducirlo lo máximo posible. Para ello, a modo de resumen, se muestran directamente los resultados del test a posteriori de Bergmann y Hommel para un intervalo de confianza $> 95\%$, después de que el test de Friedman determinase diferencias significativas entre los algoritmos. En la Figura 6.4, se pueden observar diferencias significativas entre los AGs. El tiempo de ejecución está relacionado en gran medida a la complejidad computacional de la función objetivo utilizada por el algoritmo, así como a su estrategia de búsqueda de k_v. El mejor algoritmo en cuanto a tiempo de ejecución es VGA, pero no es el mejor en cuanto a calidad en los diferentes CVIs analizados, con lo que no tendría el equilibrio que se busca. Seguramente, su problema sea debido a que el índice I (21) está compuesto por un término de $1/k$ que debe maximizarse, lo que limita la posibilidad de buscar k_v grandes, así la complejidad disminuye al orientar una búsqueda para valores pequeños de la variable k_v. A continuación, por orden de rango, se encuentra GASGO con unos valores muy similares a GCUK. Por tanto, si se busca el equilibro entre tiempo de ejecución y rendimiento en los diferentes CVIs analizados, GASGO tendría una posición relevante. Formando otro grupo con diferencias significativas en cuanto al tiempo de cómputo estarían los algoritmos CGA Y TGCA. Finalmente, con un peor tiempo de cómputo estarían los algoritmos EAC, FEAC_{RI}, FEAC_{SS} y GGA_{S}, en gran parte debida a la heurística de estos AGs. Es interesante resaltar que los AGs que utilizan índice Silhouette (13) como GGA_{S} y CGA, presentan un reducido desempeño en lo relativo al tiempo de ejecución como se muestra en la Figura 6.4. El motivo es que este índice requiere calcular todas las distancias entre las instancias conduciendo a un costo computacional de $O(n^2)$ [3], aunque sí que se ha detectado que el índice de Silhouette (13) es un CVI apropiado para la estimación de k_v. Para reducir el tiempo de cómputo del índice de Silhouette y utilizar sus buenos resultados para estimar la k_v, el FEAC_{SS} introdujo el índice Silhouette simplificado, que reduce la complejidad de este índice a $O(n)$. No obstante, no se logra el resultado esperado por la estrategia que sigue su operador de mutación para dividir grupos, aumentando el valor de k_v y, por tanto, su complejidad.

Para finalizar, se puede concluir que el algoritmo GASGO que se propone en este trabajo, obtiene el mejor balance entre tiempo de ejecución y rendimiento en los diferentes CVIs. En su diseño, tanto la representación como los operadores se han establecido para lograr estos objetivos. El operador de `cruce-PNN+KV` (ver sección 6.3.2) funciona con el mismo objetivo de otros operadores de cruce para un problema tradicional, al aplicarlo los individuos son mejorados constantemente a lo largo de la evolución, no se trata de un solo intercambio de genes. Además, este operador trabaja también en la dirección de mejorar el número de grupos. En este algoritmo, tanto el operador como la codificación y la función aptitud son diseñadas para que se relacionen de forma adecuada en la resolución del problema de agrupación.

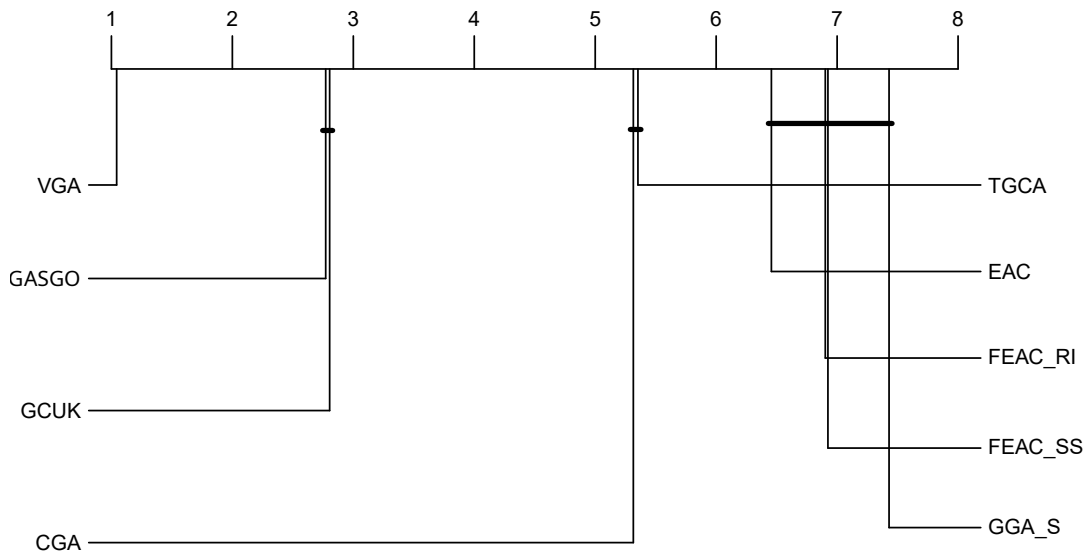


Figura 6.4 Distancia crítica de GASGO contra los otros AGs de k-variable obtenida a partir de los p-valores obtenidos para el tiempo de ejecución con una confianza $> 95\%$

6.6.1 Estudio del número de grupos estimado por los algoritmos genéticos

La estimación del número de grupos se plantea en la actualidad como uno de los problemas abiertos en el problema de agrupamiento [33]. En esta sección, se estudia si el número de grupos obtenido por algunas de las propuestas que se han estudiado y que estiman el número de grupos es confiable con respecto a la estructura interna de cada uno de los conjuntos de datos. Además, se plantean algunas ideas que pueden servir de base para realizar avances en la estimación confiable del número de grupos.

Tabla 6.6 Estimación de número de grupos con los AGs

DATASET	K	CGA	FEAC_{SS}	GCUK	GGA_{S}	GOPM_SS	GOPM_DB	GASGO
A1	20	2.00±0.00	13.9±2.85	17.50±1.98	4.50±5.15	17.24±0.47	18.40±1.00	**20.20±1.99**
A2	35	4.10±0.36	28.50±4.08	30.30±2.76	8.10±2.36	**35.00±0.07**	34.30±0.44	**35.00±0.00**
A3	50	4.00±0.00	41.10±4.01	42.60±3.54	6.10±7.60	49.99±0.12	48.90±0.64	**50.00±0.00**
Aggregation	7	4.00±0.00	**5.00±0.00**	4.70±1.50	4.00±0.00	4.39±0.49	4.00±0.00	22.50±2.82
Birch1	100	3.00±0.52	3.80±0.52	95.40±6.59	5.90±6.33	**100.00±0.00**	**100.00±0.00**	**100.00±0.00**
Birch2	100	3.00±0.96	62.90±11.81	119.60±22.10	56.50±42.06	70.66±2.77	**83.50±1.24**	299.90±0.34
Compound	6	3.00±0.00	3.00±0.00	**3.10±0.70**	3.00±0.10	3.00±0.00	3.00±0.00	3.00±0.00
D31	31	3.40±0.50	26.00±2.44	26.50±2.58	7.90±6.22	**31.00±0.00**	30.30±0.78	**31.00±0.00**
Dim2	9	**9.00±0.00**	**9.00±0.00**	8.90±0.40	11.10±2.97	**9.00±0.00**	**9.00±0.00**	**9.00±0.00**
Dim8	9	**9.00±0.00**	8.10±1.15	8.40±1.17	11.10±7.87	**9.00±0.00**	**9.00±0.00**	**9.00±0.00**
Dim32	16	15.80±0.50	10.90±2.54	15.00±0.91	**16.00±0.00**	**16.00±0.00**	**16.00±0.00**	**16.00±0.00**
Dim64	16	15.70±0.55	10.70±1.98	15.30±0.83	**16.00±0.00**	**16.00±0.00**	**16.00±0.00**	**16.00±0.00**
Dim128	16	15.70±0.55	11.00±1.92	15.40±0.80	**16.00±0.00**	**16.00±0.00**	**16.00±0.00**	**16.00±0.00**
Flame	2	4.00±0.10	**3.80±0.66**	4.10±0.43	4.00±0.00	4.04±0.23	4.00±0.00	8.10±2.19
G2-8-20	2	**2.00±0.00**	**2.00±0.00**	10.60±14.78	2.00±0.28	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**
G2-8-80	2	**2.00±0.00**	7.40±28.41	16.70±16.89	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**
G2-32-20	2	**2.00±0.00**	**2.00±0.00**	10.10±14.97	2.20±2.21	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**
G2-32-80	2	2.00±0.00	2.10±0.22	16.20±17.21	**2.00±0.00**	**2.00±0.00**	1.70±0.45	**2.00±0.00**
G2-128-20	2	2.00±0.00	2.00±0.00	6.90±9.54	2.10±1.45	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**
G2-128-80	2	2.00±0.00	2.00±0.00	8.50±10.44	**2.00±0.00**	1.98±0.14	1.10±0.26	**2.00±0.00**
Jain	2	2.20±0.41	3.30±1.36	4.00±2.13	**2.00±0.12**	3.00±0.00	3.10±0.30	15.70±1.46
Madelon	32	1.00±0.12	1.00±0.00	8.40±2.58	2.00±0.00	1.71±2.78	**13.30±10.43**	2.00±0.00

Tabla 6.6 – continuación de la página previa

DATASET	K	CGA	$FEAC_{SS}$	GCUK	GGA_S	GOPM_SS	GOPM_DB	GASGO
Path-based	3	2.90±0.31	**3.00±0.91**	4.70±2.57	2.60±0.49	2.51±0.50	3.40±0.62	12.70±2.82
R15	15	8.00±0.00	8.00±0.00	10.50±2.92	8.50±1.83	8.80±0.40	**15.00±0.00**	**15.00±0.0**
S1	15	10.40±2.80	14.80±0.39	14.80±0.68	16.20±3.00	**15.00±0.00**	**15.00±0.00**	**15.00±0.00**
S2	15	4.10±0.87	14.30±0.85	14.90±0.76	13.40±6.20	**15.00±0.00**	**15.00±0.00**	**15.00±0.00**
S3	15	2.00±0.14	12.80±2.48	14.40±3.21	6.70±5.16	14.96±0.20	14.40±0.48	**15.00±0.00**
S4	15	2.80±0.40	13.20±2.92	15.40±4.53	10.80±6.60	14.12±0.49	14.10±0.62	**15.00±0.00**
Spiral	3	**3.00±0.10**	25.90±1.75	12.50±1.86	4.00±3.15	13.18±1.65	14.40±0.93	14.40±1.17
Unbalance	8	5.90±1.34	4.00±0.00	8.10±7.97	**8.00±0.00**	4.00±0.00	8.00±0.21	19.20±0.47
Abalone	28	3.00±0.00	3.00±0.00	3.50±1.63	3.00±0.00	3.00±0.00	3.00±0.00	**14.40±1.26**
Appendicitis	2	**2.00±0.07**	2.00±0.43	2.10±0.99	2.00±0.12	1.97±0.18	1.90±0.30	1.90±0.30
Australian	2	1.00±0.16	42.10±10.18	15.90±2.83	7.10±5.99	15.55±7.24	18.60±3.84	**2.00±0.00**
Automobile	5	1.90±0.32	**5.20±3.34**	5.60±1.62	6.90±3.92	6.20±2.65	6.40±2.84	2.00±0.00
Balance	3	**3.80±0.82**	43.10±3.31	18.30±2.72	5.00±0.00	21.73±1.24	21.60±1.43	2.00±0.00
Banana	2	**2.00±0.10**	3.70±1.45	19.10±13.77	2.60±0.97	2.91±0.29	8.90±4.20	11.80±0.79
Bands	2	1.60±0.48	2.00±0.45	5.20±3.24	2.00±0.26	2.04±0.88	3.20±3.37	**2.00±0.00**
Breast	2	**2.00±0.00**	15.10±7.51	11.80±1.44	**2.00±0.00**	9.44±2.65	10.60±2.53	**2.00±0.00**
Bupa	2	1.50±0.50	1.90±0.54	3.90±2.66	**2.00±0.00**	2.00±0.52	3.10±2.61	1.90±0.29
Cleveland	5	**2.00±0.00**	14.20±2.35	10.80±1.36	10.70±3.74	10.71±1.51	11.40±1.26	2.00±0.00
Coil 2000	2	1.90±0.34	316.20±328.93	45.80±27.28	2.00±0.10	2.03±0.19	4.40±13.37	**2.00±0.00**
Contraceptive	3	**2.00±0.33**	11.70±23.03	8.20±4.33	4.60±1.00	4.71±1.19	6.10±1.82	4.00±0.16
Crx	2	1.10±0.30	17±19.33	8.60±6.69	2.20±1.57	2.13±1.27	2.70±3.14	**2.00±0.00**
Dermatology	6	**3.00±0.00**	2.10±0.39	2.10±0.54	3.00±0.17	2.00±0.00	2.30±0.66	**3.00±0.00**
Ecoli	8	1.30±0.46	1.70±0.64	2.00±1.07	**3.70±0.58**	1.60±0.49	2.90±0.98	3.60±0.49
Flare	6	2.00±0.1	67.90±5.05	17.00±3.63	35.10±15.85	31.81±2.10	31.00±1.95	**2.00±0.00**
Glass	6	1.20±0.42	1.10±0.30	2.00±1.23	**2.70±0.60**	2.16±0.94	1.50±1.05	2.00±0.00
Haberman	2	1.5±0.5	2.70±1.64	7.7±1.77	**2.00±0.00**	2.07±0.26	9.40±2.34	**2.00±0.00**
Hayes-Roth	3	2.00±0.00.1	13.2±1.51	8.5±1.09	**2.60±2.09**	8.69±1.17	8.6±1.05	2.00±0.00
Heart	2	**2.00±0.00**	13.6±2.2	10.9±1.49	10.8±3.44	10.8±1.54	11.4±1.33	**2.00±0.00**
Hepatitis	2	**2.00±0.00**	5.3±2.09	4.6±0.98	2.00±0.00.3	3.34±1.51	3.5±1.56	**2.00±0.00**
Housevotes	2	**2.00±0.00**	3.2±4.15	2.10±0.83	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**
Ionosphere	2	1.00±0.10	1.30±0.46	2.40±1.64	4.1±0.75	3.33±1.4	1.3±1.04	**2.00±0.00**
Iris	3	2.00±0.00	2.00±0.00	2.00±0.00.07	2.00±0.00	2.00±0.00	2.00±0.00	**3.00±0.22**
Leaves Plant-M	100	2.00±0.00	2.5±0.81	**7.60±7.17**	2.00±0.00.3	2.47±0.68	2.30±0.46	2.00±0.00
Leaves Plant-S	100	2.00±0.00	2.00±0.00	2.40±0.99	2.00±0.00.28	2.00±0.00	2.00±0.00	**3.00±0.00**
Leaves Plant-T	100	1.8±0.37	4.6±13.92	26.7±5.27	4.8±5.53	24.35±13.66	**33.4±7.27**	2.00±0.00
LED displayn	10	2.30±1.73	24.70±2.52	**10.70±1.18**	21.20±2.27	15.40±1.8	14.9±1.76	2.00±0.00
Movement	15	2.00±0.00	22.40±3.09	**12.5±1.71**	9.40±4.59	13.14±1.51	13.70±1.45	2.00±0.00
Magic	2	**2.00±0.00**	2.00±0.00.21	9.2±11.48	2.5±0.5	**2.00±0.00**	3.70±0.58	**2.00±0.00**
Mammographic	2	**2.00±0.00**	**2.00±0.00**	4.8±6.07	**2.00±0.00**	**2.00±0.00**	1.90±0.24	**2.00±0.00**
Marketing	9	2.00±0.00	477.8±50.59	33.80±23.48	2.00±0.00	**5.74±0.61**	48±17.64	2.00±0.00
Monks	2	2.00±0.00	32.30±2.57	15.8±1.6	4.4±2.7	13.75±2.14	15.30±1.74	**2.00±0.00**
Mushroom	2	7.00±1.43	3.00±0.81	19.5±17.97	13.80±1.09	4.79±4.29	4.00±3.8	**3.00±0.00**
New thyroid	3	1.80±0.43	1.90±0.59	2.2.00±0.00.87	1.80±0.37	1.87±0.37	1.90±0.39	**2.80±0.38**
Optdigits	10	2.00±0.00.19	9.40±42.8	17.8±8.41	**10±1.37**	7.36±2.87	9±0.91	2.00±0.00
Page-Blocks	5	1.60±0.5	2.80±0.87	14±13.03	2.00±0.00	1.87±0.42	**3.70±8.36**	3.30±0.97
Penbased	10	2.00±0.00.21	7±3.54	20.10±10.11	**10.60±1.67**	7.87±1.43	7.50±1.14	3.00±0.18
Phoneme	2	**3.70±1.72**	4.30±1.44	19.10±14.21	6.2.00±0.00.68	4.39±0.52	5.00±0.36	5.20±0.40
Pima	2	1.80±0.43	3.40±1.32	7.30±4.47	2.00±0.00.22	2.18±0.38	4.5±4.88	**2.00±0.00**
Post Operative	3	1.30±0.45	5.6±2.75	5.10±1.37	4.20±2.25	2.43±1.86	**3.30±2.01**	2.00±0.00
Saheart	2	**2.00±0.00**	2.40±0.48	2.10±0.47	**2.00±0.00**	**2.00±0.00**	2.10±0.31	**2.00±0.00**
Satimage	6	2.00±0.00	2.00±0.00	**3.4±2.29**	2.00±0.00	2.00±0.00	2.00±0.00	3.00±0.00
Segment	7	2.00±0.00	2.00±0.00.07	**4.70±5.72**	2.10±0.42	2.00±0.00	2.00±0.00.07	2.10±0.51
Shuttle	7	1.60±0.50	18.40±132.34	20.9±14.19	**6.8±5.94**	1.91±0.42	2.5±2.69	169.7±11.79
Sonar	2	1.7±0.47	1.5±0.5	4±1.95	**2.00±0.00.39**	3.87±3.2	2.70±2.56	1.90±0.30
Spambase	2	1.90±0.3	2.00±0.00.14	21±11.52	**2.00±0.00.12**	2.04±0.27	7.6±15.26	2.00±0.00.22
SpectF heart	2	1.7±0.47	1.50±0.5	2.10±0.42	**2.00±0.00**	1.98±0.16	1.60±0.49	**2.00±0.00**
Spect heart	2	**2.00±0.00**	8.9±4.88	7.90±1.86	**2.00±0.00**	2.96±0.32	7.20±3.41	1.90±0.30
Tae	3	8.50±1.59	6.10±2.3	4±1.65	10±1.44	3.44±2.83	**2.6±2.51**	2.00±0.00
Texture	11	2.00±0.00	2.00±0.00	**3.30±2.90**	2.00±0.00.28	2.00±0.00	2.00±0.00	2.00±0.00
Thyroid	3	**2.90±2.88**	34.4±7.11	14±4.4	23.70±4.66	20.98±9.95	31±12.2	2.00±0.00
Tic-Tac-Toe	2	**2.00±0.00**	50.20±15.41	17.2±2.88	12.10±5.79	18.59±4.27	16.20±1.08	**2.00±0.00**
Titanic	2	10.40±0.87	11.7±0.9	**4.70±2.11**	10.40±0.93	10.79±2.05	11.70±0.9	11.70±0.90
Twonorm	2	2.00±0.00	34.00±127.11	57.90±27.96	**2.00±0.00**	**2.00±0.00**	4.70±14.21	**2.00±0.00**
Vehicle	4	1.4±0.49	2.00±0.00.72	**2.60±1.77**	2.00±0.00.14	2.03±0.16	2.00±0.00.16	2.00±0.00
Vowel	11	4.00±0.00	21.50±20.81	**4.30±1.42**	4.00±0.00	4.05±0.21	4.1±0.34	4±0
Wdbc	2	1.80±0.40	1.80±0.54	2.10±0.74	**2.00±0.00**	**2.00±0.00**	1.90±0.26	**2.00±0.00**
Winequality-R	6	1.5±0.5	2.10±0.8	8.3±4.98	2.10±0.42	2.06±0.31	**7.30±8.68**	2.00±0.00
Winequality-W	7	1.00±0.00	36.90±115.49	18.70±15.42	2.00±0.00	2.00±0.00	**2.00±0.00.07**	2.00±0.00
Wine	3	2.10±0.22	**2.30±0.79**	2.2.00±0.00.81	**2.30±0.47**	2.04±0.33	1.60±0.92	**2.30±0.44**
Wisconsin	2	**2.00±0.00**	**2.00±0.00**	2.10±0.72	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**	**2.00±0.00**
Yeast	10	1.70±0.46	1.3±0.46	**6.80±4.84**	1.90±0.34	1.87±0.75	2.80±2.57	2.00±0.00
Zoo	7	4.70±0.53	**6.70±1.59**	5.00±0.89	4.90±0.66	6.05±1.18	4.00±2.49	3.40±0.92

En la Tabla 6.6 se muestran los diferentes conjuntos de datos con los que se ha trabajado en los estudios experimentales junto con su valor de k. Este valor representa el número de clases o grupos que está definido para ese conjunto de datos y que se considera que sería el valor confiable que debería estimar los diferentes algoritmos. Se han considerado algunas de las propuestas previas que obtenían mejor rendimiento en el estudio que se llevó a cabo y tres versiones del algoritmo GASGO, uno de los algoritmos propuestos en esta tesis. Concretamente, GASGO es la versión tal cual se ha definido en esta tesis y las otras dos extensiones varían en el CVI que utilizan como función de aptitud. GASGO_{SS} utiliza el índice SS y GASGO_{DB} el índice DB. Estas versiones se han diseñado para poder plantear algunos casos de estudio que se han encontrado en la estimación de grupos y que serán explicadas en esta sección.

En principio, la Tabla 6.6 nos muestra que el algoritmo GASGO obtiene unas estimaciones muy próximas al número de grupos real, coincidiendo en muchos casos con el valor exacto. CGA también obtiene unos valores similares. Concretamente, este estudio nos ha determinado que, a priori, el uso de diferentes CVIs empleados por los algoritmos nos permite ajustarnos mejor a unos determinados conjuntos de datos que a otros. Concretamente, se ha comprobado que una combinación de CVIs nos permitiría realizar una estimación más precisa de k_v. Analizando los resultados de la experimentación mostrados en la Tabla 6.6 y los conjuntos de datos artificiales, los cuales en su mayoría fueron diseñados haciendo coincidir las clases de las instancias con los grupos, se puede determinar que la combinación de los índices SS, DB y VRC son los más apropiados para obtener el número de grupos para un conjunto de datos. Estos son los CVIs que se han empleado en las diferentes versiones que se han diseñado de GASGO y sus resultados se han mostrado en la tabla 6.6.

De acuerdo con los resultados obtenidos, se ha estudiado si fuese posible obtener un intervalo de k_v que se considere confiable con respecto al número de grupos de los diferentes conjuntos de datos. Para ello, partiendo de los tres CVIs que se han comentado anteriormente y de los resultados de las tres versiones de GASGO que lo utilizan, se han encontrado cuatro casos que se comentan a continuación y que podrían ser un comienzo para ayudar a la estimación de un valor de k_v confiable. Para describir estos casos se considera que las particiones óptimas de X obtenidas por los CVIs son denotadas por: $\{C_i\}_{\text{SS}}$, $\{C_i\}_{\text{DB}}$ y $\{C_i\}_{\text{VRC}}$; el número de grupos para cada partición es $|\{C_i\}_{\text{SS}}|$, $|\{C_i\}_{\text{DB}}|$ y $|\{C_i\}_{\text{VRC}}|$ respectivamente.

`Caso 1.` Cuando los índices coinciden en la estimación del mismo número de grupos, como se establece en la ecuación (6.6), el valor sería confiable.

$$\lfloor |\{C_i\}_{\text{SS}}| \pm s + \frac{1}{2} \rfloor = \lfloor |\{C_i\}_{\text{DB}}| \pm s + \frac{1}{2} \rfloor = \lfloor |\{C_i\}_{\text{VRC}}| \pm s + \frac{1}{2} \rfloor = k_v \tag{6.6}$$

donde s es la desviación estándar, obtenida en las diferentes ejecuciones, puede ser sumada o restada para compensar el valor de k_v.

Por ejemplo, en la Tabla 6.6, algunos conjuntos de datos artificiales donde coinciden son A2 al obtener $\lfloor 35+0.07+1/2\rfloor = \lfloor 34.3+0.44+1/2\rfloor = \lfloor 35+0+1/2\rfloor \implies k_v = 35$, $A3$ en $k_v = 50$, Birch1 en $k_v = 100$, Dim2 y Dim8 en $k_v = 9$; S1, S2, S3 y S4 en $k_v = 15$ y los G2 en $k_v = 2$ variando cuando están sobrepuestos los grupos, al disminuir el resultado de $|\{C_i\}_{\mathrm{DB}}|$.

En los conjuntos de datos reales Wisconsin con $k_v = 2$, Housevotes con $k_v = 2$ y Texture; aunque existe coincidencia en $k = 2$ por los índices, el número de clases de las instancias es $|\{R_s\}| = 11$ implicando que no existe coincidencia entre $\{C_i\} \neq \{R_s\}$ para este conjunto de datos.

`Caso 2` Este caso se presenta cuando los índices definen un intervalo como el que se indica en la ecuación (6.7). Lo que se puede apreciar en los resultados es que los valores de k_v se encuentran en el siguiente orden: el menor es obtenido por SS, a continuación, DB y el mayor sería el obtenido por VRC.

$$\begin{aligned} \lfloor |\{C_i\}_{\mathrm{SS}}| \pm s + \frac{1}{2}\rfloor \leq \lfloor |\{C_i\}_{\mathrm{DB}}| \pm s + \frac{1}{2}\rfloor \leq \lfloor |\{C_i\}_{\mathrm{VRC}}| \pm s + \frac{1}{2}\rfloor \wedge \\ \lfloor |\{C_i\}_{\mathrm{SS}}| \pm s + \frac{1}{2}\rfloor < \lfloor |\{C_i\}_{\mathrm{VRC}}| \pm s + \frac{1}{2}\rfloor \\ \implies \lfloor |\{C_i\}_{\mathrm{SS}}| + \frac{1}{2}\rfloor \leq k_v \leq \lfloor |\{C_i\}_{\mathrm{VRC}}| + \frac{1}{2}\rfloor \end{aligned} \tag{6.7}$$

Para la determinación de k_v, este caso se han visto los casos 2.1 y 2.2

`Caso 2.1` A parte de cumplir con la ecuación (6.7), si dos índices coinciden con el número de grupos, como se establece en la ecuación (6.8) y uno de los tres es diferente. Existe una alta posibilidad que el número de grupos esté determinado por la coincidencia de los dos índices, pero con cierta incertidumbre.

$$\begin{aligned} (6.7) \wedge (\lfloor |\{C_i\}_{\mathrm{SS}}| \pm s + \frac{1}{2}\rfloor = \lfloor |\{C_i\}_{\mathrm{DB}}| \pm s + \frac{1}{2}\rfloor \vee \\ \lfloor |\{C_i\}_{\mathrm{DB}}| \pm s + \frac{1}{2}\rfloor = \lfloor |\{C_i\}_{\mathrm{VRC}}| \pm s + \frac{1}{2}\rfloor) \end{aligned} \tag{6.8}$$

Por ejemplo, en la Tabla 6.6, para el conjunto de datos R15 el número de grupos cumple con la condición $\lfloor 8.8+0.4+1/2\rfloor \leq \lfloor 15+1/2\rfloor \leq 15+1/2\rfloor$ y $\lfloor 8.8+$

$0.4+1/2\rfloor < \lfloor 15+1/2\rfloor$, por tanto, $k_v \in [9,15]$, y por coincidir dos índices existe una alta probabilidad que sea $k_v = 15$. Iris es otro conjunto de datos donde se cumple este caso, al cumplir con las condiciones (6.7) y (6.8) $\implies k_v \in [2,3]$ y solo $|\{C_i\}_{\text{VRC}}|$ obtiene la respuesta correcta, por estar cerca del Caso 3.2, es posible tomarlo como solución.

`Caso 2.2` Este caso solamente se cumple con la condición (6.7), que define un intervalo para k_v (6.9).

$$(6.7) \wedge \neg(6.8) \implies k_v \in [\lfloor |\{C_i\}_{\text{SS}}| + \frac{1}{2}\rfloor, \lfloor |\{C_i\}_{\text{VRC}}| + \frac{1}{2}\rfloor] \tag{6.9}$$

Si es necesario especificar una solución, lo más adecuado es el índice que obtiene el valor intermedio, $k_v = \lfloor |\{C_i\}_{\text{DB}}| + \frac{1}{2}\rfloor$.

Por ejemplo, en la Tabla 6.6, en el conjunto de datos Unbalance, los índices definen un intervalo $4 \leq 8 \pm 0.21 \leq 19.2 \pm 0.47$, así $k_v \in [4,19]$ y el valor apropiado sería $k_v = 8$, que es el valor considerado confiable. En el conjunto de datos Banana, el número de grupos esta mejor caracterizado en $k_v \in [3,12]$, específicamente un valor es $k_v = 9$, el número de clases de las instancias es $|\{R_s\}| = 2$, por su distribución en el espacio de las instancias es imposible corresponder clases y grupos, por lo tanto, $k_v = 9$ hace una buena agrupación de las instancias.

`Caso 3.` Este caso también lo dividimos en dos sub-casos dependiendo del valor de $|\{C_i\}_{\text{BD}}|$, que puede ser mucho más grande que los otros índices (Caso 3.1) o menor (Caso 3.2).

`Caso 3.1` En este caso se encuentran conjuntos de datos con una estructura compleja, pero es posible hacer una agrupacion, al poder determinarse por la ecuación (6.10).

$$\begin{gathered} \lfloor |\{C_i\}_{\text{SS}}| \pm s + \frac{1}{2}\rfloor \approx \lfloor |\{C_i\}_{\text{VRC}}| \pm s + \frac{1}{2}\rfloor \ll |\{C_i\}_{\text{BD}}| \pm s \\ \implies k_v = \lfloor |\{C_i\}_{\text{DB}}| \pm s + \frac{1}{2}\rfloor \end{gathered} \tag{6.10}$$

Por ejemplo, en la Tabla 6.6, el conjunto de datos Madelon tiene el valor $|\{C_i\}_{\text{BD}}| = 13.3 \pm 10.43$, que es más alto que los otros índices ($2 \approx 2 \ll 24$). Según los resultados, se estima que para encontrar un valor más apropiado se podría extender con una exploración de los resultados de los AGs de k-fija alrededor del valor k_v que permita encontrar una propiedad desea.

`Caso 3.2` En este caso se encuentran los conjuntos de datos con grupos sobrepuestos que podrían determinarse por la ecuación (6.11).

$$|\{C_i\}_{\mathrm{SS}}| \pm s \leq |\{C_i\}_{\mathrm{VRC}}| \pm s > |\{C_i\}_{\mathrm{BD}}| \pm s \implies k_v = \lfloor |\{C_i\}_{\mathrm{VRC}}| \pm s + \frac{1}{2} \rfloor \quad (6.11)$$

Por ejemplo, en la Tabla 6.6, el conjunto de datos G2-128-80 con valores de $1.98 + 0.14 \leq 2 > 1.1 + 0.26$ cumpliría con este caso.

`Caso 4.` En este caso se encuentran los conjuntos de datos donde es difícil descubrir una estructura de agrupación, la condición está dada por la ecuación (6.12).

$$|\{C_i\}_{\mathrm{VRC}}| \leq 2 \text{ y } |\{C_i\}_{\mathrm{SS}}| \gg 2 \quad (6.12)$$

Por ejemplo, en la Tabla 6.6, el conjunto de datos Tic-Tac-Toe con $|\{C_i\}_{\mathrm{VRC}}| = 2$ y $|\{C_i\}_{\mathrm{SS}}| = 18.59 \pm 4.27$. Leaves Plant Texture con $|\{C_i\}_{\mathrm{VRC}}| = 2$ y $|\{C_i\}_{\mathrm{SS}}| = 24.35 \pm 13.66$, por tanto, en estos casos se vuelve complejo poder estimar una k_v.

Capítulo 7

LEAC: Una librería de algoritmos evolutivos para agrupamiento

Library Evolutionary Algorithms for Clustering (LEAC) [109] es una nueva biblioteca basada en EAs escrita en C++, que sirve para resolver el problema de agrupación. LEAC proporciona una gran cantidad de componentes (esquemas de codificación de individuos, operadores genéticos y métricas de evaluación, entre otros) que permiten un fácil y rápido desarrollo de nuevos AGs. Además, incluye 24 AGs que representan el estado del arte en AEs para la agrupación en particiones.

El código de la biblioteca LEAC se ha desarrollado bajo GNU General Public License v3 (GPLv3) siguiendo las guías dictadas de la ingeniería de software. Su documentación (API y manual del usuario) está disponible para ayudar al usuario a hacer un uso eficaz de la herramienta *https://github.com/kdis-lab/leac* e incluye ejemplos de uso, un tutorial sobre la ejecución de algoritmos implementados, así como la inclusión de nuevos algoritmos y una descripción detallada de la arquitectura del software.

En este capítulo se describen las características principales y los principios de diseño del software, así como la principal funcionalidad disponible en LEAC que permite realizar una comparación entre diferentes propuestas y extenderla fácilmente mediante la inclusión de nuevos algoritmos. Para una descripción más detallada, se recomienda consultar el manual de usuario[1].

[1] https://github.com/kdis-lab/LEAC/blob/master/leac-userManual.pdf

7.1 Introducción

Como se ha comentado, los AEs han ganado gran popularidad debido a que son capaces de obtener soluciones casi óptimas a dicho problema considerado NP-difícil, en un tiempo razonable permitiendo realizar tareas de agrupación basadas en particiones [59]. Por lo tanto, se puede encontrar una gran cantidad de AGs para resolver problemas de agrupamiento [59, 53, 94, 90].

Con la realización de este trabajo, la principal deficiencia que se ha encontrado en el diseño de nuevos EA para resolver la tarea de agrupamiento es la gran cantidad de tiempo que se requiere para codificar nuevas propuestas, así como para explorar sus propiedades y evaluar su rendimiento en comparación con otros algoritmos anteriores. Para reducir los esfuerzos de investigación en esta área, se ha desarrollado LEAC. Esta librería está modularizada y contiene un número considerable de CVIs, operadores genéticos y funciones para poder llevar a cabo implementaciones de nuevas propuestas de AGs con relativamente poco esfuerzo. Además, incluye 24 de las propuestas de AGs mono-objetivo más representativas en el tema hasta la fecha que se pueden ejecutar fácilmente a través de experimentos configurados por los usuarios.

Hasta donde se conoce, aunque existen librerías para facilitar la implementación de AG de propósito general, no existe una biblioteca de código abierto que enfoque sus componentes para resolver el problema de agrupación basado en particiones. Así, las principales aportaciones de la librería LEAC se pueden resumir en dos puntos. Por un lado, proporcionar el estado del arte de los AGs para resolver la agrupación en particiones que se puede ejecutar sin esfuerzo, lo que permite a los investigadores realizar un estudio experimental completo, y por otro lado, hacer más natural y eficiente la implementación de nuevos EAs para agrupación basada en particiones aplicando paquetes que contienen varias medidas de desempeño, operadores genéticos, representaciones de los individuos y funciones específicas para resolver la tarea de agrupación. En este contexto, se considera que el diseño de software libre es hoy en día una herramienta fundamental para facilitar la difusión y el avance en las diferentes áreas [131].

En este capítulo se describirán los aspectos más relevantes de esta librería la cual está disponible en github, junto con los 96 conjuntos de datos que se han utilizado en los estudios llevados a cabo, y cuenta con un manual de usuario donde toda la información relevante puede ser consultada.

7.2 Arquitectura de LEAC

LEAC se implementa en C++11, C++14 y STL. El uso de STL proporciona a LEAC una biblioteca numérica de generación de números pseudoaleatorios. Además, LEAC tiene una

capa de software de bajo nivel basada en Streaming SIMD Extensions (SSE) y en la biblioteca OpenBLAS para optimizar su rendimiento.

LEAC se ha diseñado en una arquitectura basada en cuatro capas o módulos. Cada una de ellas consiste en un conjunto de paquetes relacionados como se muestra en la Figura 7.1. A continuación, se describe brevemente cada una de las capas:

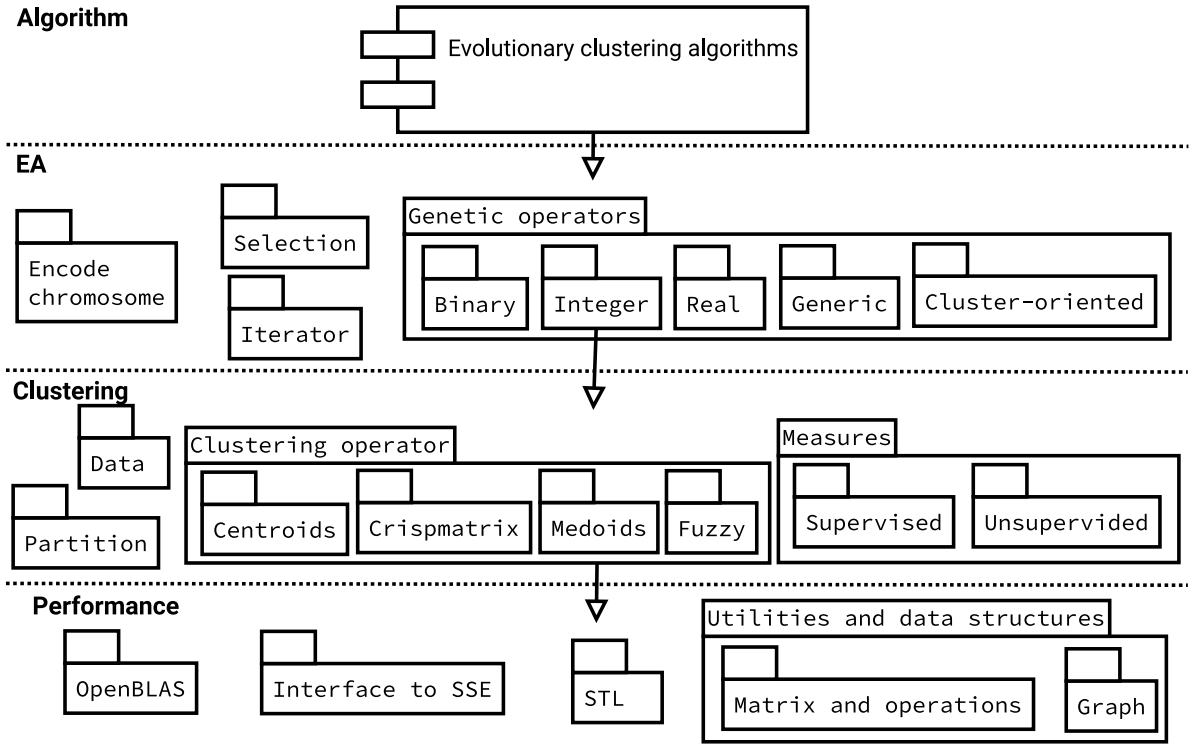


Figura 7.1 Arquitectura de capas de la biblioteca LEAC

`Algorithm`. Consta de los AGs de agrupación mono-objetivos que se han propuesto en la literatura hasta la fecha, los cuales son descritos en la sección 3.2. Para la implementación de los algoritmos se usa la capa de componentes y estrategias (**EA**). Actualmente, contiene 24 AGs que se consideran el estado del arte de los AGs para resolver el agrupamiento. La información detallada sobre los algoritmos disponibles se encuentra en el manual de usuario.

`EA`. Contiene varios paquetes de componentes y estrategias que pueden ser utilizados para la implementación de nuevos AGs, tales como criterios de codificación, métodos de inicialización, métodos de selección, operadores de cruce y mutación, y estrategias de reemplazo y actualización de la población. Una descripción de diferentes componentes y estrategias se incluye en la sección 3.1. Actualmente cuenta con 4 tipos de codificación

de las soluciones, 4 procedimientos para inicialización de la población, 3 criterios para seleccionando descendientes, 18 operadores de cruce y 14 operadores de mutación. Esta capa se basa en el apoyo de las capas inferiores para lograr un esquema de evolución. Una información detallada sobre los criterios, operadores y funciones contenidos en la biblioteca se pueden encontrar en el manual del usuario.

`Clustering.` Esta capa está formada por varios paquetes con operadores específicos para agrupación independiente de los operadores genéticos, tales como: operadores basados en los centroides, particiones duras, instancias representativas y métricas de desempeño de agrupación supervisadas y no supervisadas (en total implementa 28 CVIs).

`Performance.` Esta capa está formada por las funciones programadas de bajo-nivel aprovechando las arquitecturas actuales de unidad central de procesamiento (CPU). Por ejemplo, el alineamiento de datos y las extensiones de transmisión (SSE) [1]. Esta capa permite a las capas superiores trabajar con un mayor desempeño .

Para trabajar en el dominio de los números enteros y reales, LEAC se basa en plantillas que se utilizan para especificar las interfaces de sus funciones. Considera el estándar impuesto por la librería STL Algorithms y, en consecuencia, opera sobre rangos de elementos definidos como [primero; último) donde último se refiere al elemento más allá del último elemento para inspeccionar o modificar. También, incorpora nuevas características de C++ como el uso de funciones lambda para permitir flexibilidad en la implementación de los algoritmos.

Finalmente, se comenta que está orientada a una implementación compacta para evitar duplicidad de código y, de esta manera, todos los AGs sean implementados en igualdad de circunstancias para permitir hacer una comparación y análisis de desempeño entre ellos.

7.3 Funcionalidades de LEAC

LEAC es fácilmente extensible y configurable, y permite realizar un estudio experimental completo. En LEAC, se puede diferenciar dos funcionalidades principales:

- *Modo básico*. Este modo permite ejecutar cualquiera de los AGs implementados en LEAC para resolver problemas de agrupamiento. LEAC también proporciona la funcionalidad para llevar a cabo un estudio experimental completo que admite la validación cruzada de k-fold para la partición de datos, la configuración de los parámetros del algoritmo y los informes de salida. Por lo tanto, esta biblioteca es una opción interesante para comparar el rendimiento de nuevas propuestas con los métodos conocidos disponibles en LEAC.

- *Modo Avanzado*. Esta modalidad permite reducir el trabajo de programación para desarrollar nuevos algoritmos utilizando los diferentes módulos proporcionados por LEAC. Dado que el código fuente está disponible y está dividido por módulos (ver Figura 7.1), es posible diseñar nuevas propuestas utilizando o ampliando la funcionalidad disponible o desarrollando la propuesta desde cero. Los principales componentes disponibles son:

  - Representación de los individuos. Se encuentran clases disponibles para implementar cualquiera de los esquemas de codificación que se han visto en las diferentes propuestas estudiadas. Esto incluye codificación binaria y con números enteros o reales. Cada tipo de codificación se puede asociar con un fenotipo específico: basada en centroides, basada en instancia representativas, basadas en etiquetas y basada en árboles. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.1 del manual de usuario.
  - Métodos de inicialización. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. Desde el procedimiento más simple que emplea una inicialización de la población aleatorio, los objetos se asignan aleatoriamente a un grupo, sin embargo, es una de las más ampliamente utilizadas, hasta otros métodos más elaborados. Existen diferentes métodos de inicialización, en función de la codificación seleccionada. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.2 del manual de usuario.
  - Función de aptitud. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. De este modo, cualquiera de los 28 CVIs implementados, puede ser usado en el algoritmo que se diseñe. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.3 del manual de usuario.
  - Criterio de parada. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. No existe un criterio de parada que asegure la convergencia en el proceso evolutivo. Por lo general, se utilizan dos criterios de parada. En el primero, el proceso se ejecuta para un número fijo de iteraciones y la mejor solución/individuo que es obtenida se considera la óptima. En el otro criterio, el algoritmo se termina si no se observa una mejora adicional en el valor de aptitud del mejor individuo para un número fijo de iteraciones, y se toma como óptimo el mejor individuo obtenido. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.4 del manual de usuario.

- Método de selección. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. En este paso, los individuos de la población se seleccionan para aplicarles los operadores de cruce y mutación. Los principales esquemas utilizados son dos: selección de ruleta que está basado en el concepto de supervivencia utilizado en los sistemas evolutivos naturales y selección por torneos que realiza varios torneos entre un pequeño número de individuos especificados que son elegidos al azar de la población. El individuo con la mejor función de aptitud es seleccionado. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.5 del manual de usuario.
- Operadores de cruce. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. Estos operadores están clasificados en especializados o no y según la codificación en la que se basan: codificación binaria, entera o real. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.6 del manual de usuario.
- Operadores de mutación. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. Estos operadores están clasificados siguiendo la misma distribución que se ha comentado con los operadores de cruce. Esto sería, si se trata de operadores especializados o no y según la codificación en la que se basan: codificación binaria, entera o real. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.7 del manual de usuario.
- Métodos de actualización de la población. Se encuentran clases disponibles para implementar cualquiera de los métodos que emplean las diferentes propuestas estudiadas. En este paso, se determinan los nuevos individuos que pasarán a la siguiente población entre los nuevos generados a través de los operadores genéticos y los que ya existían en la población. La mayoría de las propuestas, utilizan un procedimiento donde los descendientes reemplazan automáticamente a sus padres utilizando un procedimiento de elitismo que garantiza que los mejores individuos se mantienen de una población a otra. Para una descripción más detallada, se recomienda la consulta de la sección 3.1.8 del manual de usuario.

Por supuesto, cualquiera de estos métodos puede ser extendido para desarrollar nuevas propuestas a partir de ellos.

```
gga_vklabelsilhouette -i ../data/ionosphere.data  -a "1-34"  -c 35
   -M m_gga_ionosphere.data -T stdout --table-format yes
```

Figura 7.2 Ejecución del algoritmo GGA [2]

7.3.1 Caso de estudio

En esta sección, se muestra un ejemplo de los dos modos de funcionamiento. La documentación detallada está disponible en https://github.com/kdis-lab/leac que describe ampliamente todas las funciones y estructuras de la biblioteca.

Ejecutar un EA en la biblioteca (modo básico)

Cualquiera de los AG se puede ejecutar desde una terminal con parámetros en línea. Como ejemplo, la Figura 7.2 muestra la ejecución del algoritmo GGA [2] para el conjunto de datos ionosfera.

La nomenclatura utilizada para el nombre de los AGs se basa en los tres aspectos principales de estos algoritmos: nombre del algoritmo en el que se basa el programa, número de grupos fijo o variable k y la codificación utilizada:

```
nombre_[fk|vk]codificación
```

El parámetro `-help`, nos va a permitir obtener una descripción de las diferentes opciones. Hay dos tipos de opciones, las comunes a todos los programas y las particulares de cada uno de los algoritmos. Las opciones particulares se muestran de forma explícita.

Para poder ejecutar un algoritmo, el parámetro obligatorio es `-i, -instances=FILE or DIRECTORY` que siempre se requiere para especificar el nombre del conjunto de datos. El resto de los parámetros tomará por defecto los valores predeterminados que hayan especificado cada uno de los autores.

El parámetro `-a, -select-attributes[=ARG]` se emplea para seleccionar los atributos y el parámetro `-c, -class-column[=NÚMERO]` se usa para especificar el atributo de clase.

Para todos los AGs estudiados en este trabajo, los parámetros compartidos son:

`-generaciones`

`-tamaño de la población`

Solo varían en el número de generaciones y en el tamaño de la población propuesta por los diferentes autores.

De nuevo, para una descripción más detallada y un ejemplo de cada uno de los AGs de la librería, se recomienda la consulta de la sección 3.2 del manual de usuario.

Desarrollo de un nuevo AE en la biblioteca (modo avanzado)

Para resolver el problema de agrupación basado en particiones por medio de un AE, se necesita configurar los siguientes elementos: *criterio de codificación*, *inicialización de población*, *función de aptitud*, *criterio de parada*, *esquema de evolución*, *criterio para seleccionar padres*, *operador de cruce* y *operador de mutación*. En la guía del usuario se presentan ejemplos específicos del uso y la configuración de los EA. Aquí se incluye el código del ejemplo que se ha desarrollado para la implementación del algoritmo KGA.

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo se incluyen las conclusiones del trabajo realizado, se enumeran las publicaciones tanto presentadas en congresos como en revistas internacionales, así como la descripción de las futuras líneas de investigación que se proponen como continuación del trabajo presentado en esta memoria de tesis.

8.1 Conclusiones

Se han cubierto satisfactoriamente todos los objetivos que se plantearon con las tesis. A continuación, se detallan las principales conclusiones:

1. Se ha llevado a cabo una revisión exhaustiva de los AGs mono-objetivo, al realizar una descripción detallada de las principales características de los algoritmos más relevantes publicados hasta la fecha para resolver el problema del agrupamiento basado en particiones. Este estudio permitió analizar codificaciones, operadores genéticos, operadores selectores y otras características principales de todas las propuestas desarrolladas.

2. Se ha propuesto una taxonomía específica que considera todas las particularidades de los AGs. Esta taxonomía está basada en las características más relevantes, la cual permite una fácil clasificación de todas las propuestas desarrolladas hasta la fecha y además incluir cualquier nueva propuesta que se desarrolle.

3. Se ha llevado a cabo un estudio experimental exhaustivo que analiza la capacidad de los AGs para resolver problemas de agrupamiento. El estudio ha considerado bajo el mismo escenario, 22 AGs, 22 CVIs y 94 conjuntos de datos utilizando. Se trataría de un primer intento de evaluar todas las propuestas con un número representativo de mediciones y

datos. Además, todos los algoritmos, datos y configuraciones se han puesto disponibles para la comunidad científica[1].

4. Se ha diseñado y desarrollado una nueva propuesta de un AG que trabaja en un entorno en el que se conoce el número de grupos (GAMK). Esta propuesta emplea nuevos operadores genéticos que se han desarrollado específicamente para lograr una búsqueda más eficiente. En un exhaustivo estudio con todas las propuestas previas muestra un excelente rendimiento logrando superar a los métodos previos. Las principales contribuciones de este método son:

   - Los clústeres son representados como centroides, una de las representaciones que han resultado ser más flexibles en este tipo de problemas y presenta como principales novedades unos operadores genéticos más eficientes que permiten realizar una búsqueda más eficiente adaptándose al problema de agrupamiento.
   - Introduce una adaptación del número de iteraciones que se llevan a cabo en la búsqueda local que se realiza tanto en la inicialización como en la actualización de la nueva población. El proceso de búsqueda local ha resultado muy beneficioso en la optimización de las soluciones, pero es costoso computacionalmente. En este trabajo se determina este valor en función del número de grupos y de instancias del problema.

5. Se ha diseñado y desarrollado una nueva propuesta de un AG que trabaja en un entorno en el que no se conoce el número de grupos (GASGO). Esta propuesta emplea nuevos operadores genéticos que se han desarrollado específicamente para lograr una búsqueda más eficiente junto con la representación seleccionada. En un exhaustivo estudio con todas las propuestas previas muestra un excelente rendimiento logrando superar a los métodos previo. Las principales contribuciones de este método son:

   - Utiliza una representación basada en libros de códigos que permite reducir el tiempo de cómputo.
   - Se emplean unos operadores de cruce y mutación especializados que se combinan a la perfección con la codificación utilizada lo que permite una búsqueda más eficiente y, en consecuencia, encontrar mejores agrupamientos.
   - Estima automáticamente el número óptimo de grupos encontrando los valores más cercanos a los valores reales.

[1] Consultar la página web asociada http://www.uco.es/kdis/a-survey-of-evolutionary-algorithm-for-clustering-taxonomy-and-empirical-analysis/

6. Se ha desarrollado una librería con todos los AGs especificados en este trabajo, incluidas las propuestas desarrolladas. La biblioteca, LEAC, se ha puesto a disposición de la comunidad científica. Incluye además, todos los datos y configuraciones de los experimentos realizados. Aunque existen bibliotecas para facilitar la implementación de EAs de propósito general, no existe una biblioteca de código abierto que enfoque sus componentes para resolver el agrupamiento basado en particiones. La librería permite por un lado, proporcionar el estado del arte de los AGs para que se puedan ejecutar y analizar sin esfuerzo, lo que permite a los investigadores realizar un estudio experimental completo, y por otro lado, hacer más natural y eficiente la implementación de nuevos AGs utilizando los paquetes que contienen CVIs, operadores genéticos, representaciones individuales y funciones específicas para resolver el problema de agrupamiento. En este contexto, el diseño de software libre se considera hoy en día una herramienta fundamental para facilitar la difusión y el avance en las diferentes áreas.

8.2 Trabajo futuro

Como líneas de trabajo futuro que sirven de continuación y mejora de esta tesis, se plantean los siguientes puntos:

- El problema de agrupamiento ha sido ampliamente estudiado, pero todavía se requieren importantes estudios para resolver el problema en determinados ámbitos que tienen una mayor complejidad. En este contexto, incluir nuevas propuestas que incluyan representaciones más flexibles, como puede ser el aprendizaje con múltiples instancias, podría conseguir mejores resultados. El aprendizaje con múltiples instancias permite representar problemas complejos más fácilmente, mejorando así el espacio de búsqueda para encontrar las soluciones. No existen apenas propuestas evolutivas que consideren este aprendizaje, con lo que sería interesante desarrollar nuevas propuestas y evaluar su funcionamiento.

- Continuando con el desarrollo de propuestas en entornos más complejos, se plantea también como esencial el desarrollo de nuevas medidas de similitud que permitan adaptarse mejor al espacio de búsqueda con el que se esté trabajando.

- En el ámbito de la evaluación de los diferentes métodos, sigue habiendo una necesidad de nuevos estudios en el desarrollo de nuevos CVIs que permitan desarrollar índices más específicos que sean eficientes y permitan valorar diferentes características simultáneamente, así como incluir características específicas que no se han considerado actualmente.

- Finalmente, el estudio de la determinación del número de grupos resulta crucial para poder hacer un agrupamiento óptimo. En este estudio, solamente se han analizado algunos casos relativos a los resultados obtenidos, sería necesario realizar estudios empíricos en esta línea que permitan desarrollar un procedimiento para obtener valores confiables del número de grupos.

8.3 Publicaciones asociadas a la tesis

Con la finalidad de promover los resultados obtenidos de la investigación realizada en esta tesis, se han realizado las siguientes publicaciones, las cuales cubren los diferentes objetivos que se plantearon:

- Robles-Berumen, H., Zafra, A., Ventura, S. A Survey of Genetic Algorithms for Clustering: Taxonomy and Empirical Analysis enviada a Swarm and Evolutionary Computation. Cubriría el trabajo desarrollado en los tres primeros puntos que se han planteado en las conclusiones.

- Robles-Berumen, H., Zafra, A., Ventura, S. Una nueva propuesta basada en Algoritmos Genéticos para resolver problemas de Agrupamiento. XI Congreso científico de personal investigador en formación. Universidad de Córdoba, Mayo, 2023. Cubriría el trabajo desarrollado en el cuarto punto que se ha planteado en las conclusiones.

- Robles-Berumen, H., Zafra, A., Ventura, S. A Novel Genetic Algorithm with Specialized Genetic Operators for Clustering. The 18th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2023), Salamanca, Septiembre, 2023. Cubriría el trabajo desarrollado en el quinto punto que se ha planteado en las conclusiones.

- Robles-Berumen, H., Zafra, A., Fardoun, H., Ventura, S. LEAC: An efficient library for clustering with evolutionary algorithms. Knowl. Based Syst. 179: 117-119 (2019). Cubriría el trabajo desarrollado en el sexto punto que se ha planteado en las conclusiones.

Bibliografía

[1] (2010). *Intel®64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*. Intel, Santa Clara, CA, USA.

[2] Agustín-Blas, L. E., Salcedo-Sanz, S., Jiménez-Fernández, S., Carro-Calvo, L., Del Ser, J., and Portilla-Figueras, J. A. (2012). A new grouping genetic algorithm for clustering problems. *Expert Syst. Appl.*, 39(10):9695–9703.

[3] Alves, V. S., Campello, R. J. G. B., and Hruschka, E. R. (2006). Towards a fast evolutionary algorithm for clustering. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 1776–1783. IEEE.

[4] Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J. M., and nigo Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243 – 256.

[5] Ball, G. H. and Hall, D. J. (1967). A clustering technique for summarizing multivariate data. *Behaviorial Sciences*, 12(2):153–155.

[6] Bandyopadhyay, S. and Maulik, U. (2001). Nonparametric genetic clustering: Comparison of validity indices. *Trans. Sys. Man Cyber Part C*, 31(1):120–125.

[7] Bandyopadhyay, S. and Maulik, U. (2002a). An evolutionary technique based on k-means algorithm for optimal clustering in rn. *Inf. Sci. Appl.*, 146(1-4):221–237.

[8] Bandyopadhyay, S. and Maulik, U. (2002b). Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35(6):1197 – 1208.

[9] Banerjee, P., Chattopadhyay, T., and Chattopadhyay, A. K. (2023). Comparison among different clustering and classification techniques: Astronomical data-dependent study. *New Astronomy*, 100:101973.

[10] Beni, G. (2005). From swarm intelligence to swarm robotics. In Şahin, E. and Spears, W. M., editors, *Swarm Robotics*, pages 1–9, Berlin, Heidelberg. Springer Berlin Heidelberg.

[11] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52.

[12] Bezdek, J. C., Boggavarapu, S., Hall, L. O., and Bensaid, A. (1994). Genetic algorithm guided clustering. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 34–39 vol.1.

[13] Bezdek, J. C., Ehrlich, R., and Full, W. (1984). Fcm: The fuzzy c-means clustering algorithm. *Computers and Geosciences*, 10(2):191–203.

[14] Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

[15] Campo, D., Stegmayer, G., and Milone, D. (2016). A new index for clustering validation with overlapped clusters. *Expert Systems with Applications*, 64:549 – 556.

[16] Cano, A., Luna, J. M., Galindo, E. L. G., and Ventura, S. (2016). LAIM discretization for multi-label data. *Inf. Sci.*, 330:370–384.

[17] Casillas, A., de Lena, M. T. G., and Martínez, R. (2003). Document clustering into an unknown number of clusters using a genetic algorithm. In Matoušek, V. and Mautner, P., editors, *Text, Speech and Dialogue*, volume 2807 of *Lecture Notes in Computer Science*, pages 43–49. Springer Berlin Heidelberg.

[18] Chang, D.-X., Zhang, X.-D., and Zheng, C.-W. (2009). A genetic algorithm with gene rearrangement for k-means clustering. *Pattern Recogn.*, 42(7):1210–1222.

[19] Chawla, S. (2016). Application of genetic algorithm and back propagation neural network for effective personalize web search-based on clustered query sessions. *Int. J. Appl. Evol. Comput.*, 7(1):33–49.

[20] Chiang, M. M.-T. and Mirkin, B. (2010). Intelligent choice of the number of clusters in k-means clustering: An experimental study with different cluster spreads. *Journal of Classification*, 27(1):3–40.

[21] Chou, C.-H., Su, M.-C., and Lai, E. (2004). A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, 7(2):205–220.

[22] Dai, W., Jiao, C., and He, T. (2007). Research of k-means clustering method based on parallel genetic algorithm. In *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, volume 2, pages 158–161.

[23] Das, A. A. and Konar, A. (2009). *Metaheuristic Clustering*. Studies in computational intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg.

[24] Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(2):224–227.

[25] Delforge, D., Watlet, A., Kaufmann, O., Van Camp, M., and Vanclooster, M. (2021). Time-series clustering approaches for subsurface zonation and hydrofacies detection using a real time-lapse electrical resistivity dataset. *Journal of Applied Geophysics*, 184:104203.

[26] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

[27] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy.

[28] Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1):95–104.

[29] Eiben, A. E. and Smith, J. E. (2015). *Fitness, Selection, and Population Management*, pages 79–98. Springer Berlin Heidelberg, Berlin, Heidelberg.

[30] Elbeltagi, E., Hegazy, T., and Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, 19(1):43 – 53.

[31] Equitz, W. H. (1989). A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(10):1568–1575.

[32] Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996). A Density-Based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., and Fayyad, U., editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon. AAAI Press.

[33] Ezugwu, A. E., Ikotun, A. M., Oyelade, O. O., Abualigah, L., Agushaka, J. O., Eke, C. I., and Akinyelu, A. A. (2022). 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.

[34] Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S., and Bouras, A. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279.

[35] Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1):5–30.

[36] Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., New York, NY, USA.

[37] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874.

[38] Fränti, P., Kivijärvi, J., Kaukoranta, T., and Nevalainen, O. (1997). Genetic algorithms for large-scale clustering problems. *The Computer Journal*, 40(9):547–554.

[39] Fränti, P. and Sieranoja, S. (2018). K-means properties on six clustering benchmark datasets. *Applied Intelligence*, 48(12):4743–4759.

[40] Freitas, A. A. (2008). A review of evolutionary algorithms for data mining. In Maimon, O. and Rokach, L., editors, *Soft Computing for Knowledge Discovery and Data Mining*, pages 79–111. Springer US, Boston, MA.

[41] Fränti, P. (2000). Genetic algorithm with deterministic crossover for vector quantization. *Pattern Recognition Letters*, 21(1):61–68.

[42] Fränti, P., Rezaei, M., and Zhao, Q. (2014). Centroid index: Cluster level similarity measure. *Pattern Recognition*, 47(9):3034 – 3045.

[43] Garcia-Piquer, A., Fornells, A., Bacardit, J., Orriols-Puig, A., and Golobardes, E. (2014). Large-scale experimental evaluation of cluster representations for multiobjective evolutionary clustering. *IEEE Transactions on Evolutionary Computation*, 18(1):36–53.

[44] Garza-Fabre, M., Handl, J., and Knowles, J. (2018). An improved and more scalable evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 22(4):515–535.

[45] Ghezelbash, R., Maghsoudi, A., and Carranza, E. J. M. (2020). Optimization of geochemical anomaly detection using a novel genetic k-means clustering (gkmc) algorithm. *Computers and Geosciences*, 134:104335.

[46] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[47] Göhring, A., Mauder, M., Vohberger, M., Nehlich, O., von Carnap-Bornheim, C., Hilberg, V., Kröger, P., and Grupe, G. (2018). Palaeobiodiversity research based on stable isotopes: Correction of the sea spray effect on bone carbonate 13c and 18o by gaussian mixture model clustering. *Palaeogeography, Palaeoclimatology, Palaeoecology*, 490:673 – 686.

[48] Haghrah, A., Nazari-Heris, M., and Mohammadi-ivatloo, B. (2016). Solving combined heat and power economic dispatch problem using real coded genetic algorithm with improved mühlenbein mutation. *Applied Thermal Engineering*, 99:465–475.

[49] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145.

[50] Halkidi, M. and Vazirgiannis, M. (2001). Clustering validity assessment: finding the optimal partitioning of a data set. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 187–194.

[51] Halkidi, M., Vazirgiannis, M., and Batistakis, Y. (2000). Quality scheme assessment in the clustering process. In Zighed, D. A., Komorowski, J., and Żytkow, J., editors, *Principles of Data Mining and Knowledge Discovery*, pages 265–276, Berlin, Heidelberg. Springer Berlin Heidelberg.

[52] Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

[53] Hancer, E. and Karaboga, D. (2017). A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number. *Swarm and Evolutionary Computation*, 32:49 – 67.

[54] Handl, J. and Knowles, J. (2007). An evolutionary approach to multiobjective clustering. *IEEE Transactions on Evolutionary Computation*, 11(1):56–76.

[55] He, H. and Tan, Y. (2012). A two-stage genetic algorithm for automatic clustering. *Neurocomput.*, 81:49–59.

[56] Herrera, F., Lozano, M., and Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artif. Intell. Rev.*, 12(4):265–319.

[57] Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

[58] Hruschka, E. R., Campello, R. J. G. B., and de Castro, L. N. (2006). Evolving clusters in gene-expression data. *Inf. Sci.*, 176(13):1898–1927.

[59] Hruschka, E. R., Campello, R. J. G. B., Freitas, A. A., and de Carvalho, A. C. P. L. F. (2009). A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(2):133–155.

[60] Hruschka, E. R. and Ebecken, N. F. F. (2003). A genetic algorithm for cluster analysis. *Intell. Data Anal.*, 7(1):15–25.

[61] Hu, L. and Zhong, C. (2019). An internal validity index based on density-involved distance. *IEEE Access*, 7:40038–40051.

[62] Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

[63] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

[64] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471.

[65] Kaufman, L. and Rousseeuw, P. J. (1990). *Finding groups in data: an introduction to cluster analysis*. John Wiley and Sons, New York.

[66] Kayaalp, F. and Erdogmus, P. (2020). Benchmarking the clustering performances of evolutionary algorithms: A case study on varying data size. *IRBM*, 41(5):267 – 275.

[67] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4.

[68] Kim, H., Kim, H. K., and Cho, S. (2020). Improving spherical k-means for document clustering: Fast initialization, sparse centroid projection, and efficient cluster labeling. *Expert Systems with Applications*, 150:113288.

[69] Kivijärvi, J., Fränti, P., and Nevalainen, O. (2003). Self-adaptive genetic algorithm for clustering. *Journal of Heuristics*, 9(2):113–129.

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

[71] Kuncheva, L. I.and Bezdek, J. C. (1997). Selection of cluster prototypes from data by a genetic algorithm. In *in Proc. 5th Eur. Congr. Intell. Tech. Soft Comput.*, pages 1683–1688.

[72] Lee, S. H., Jeong, Y. S., Kim, J. Y., and Jeong, M. K. (2018). A new clustering validity index for arbitrary shape of clusters. *Pattern Recognition Letters*, 112:263 – 269.

[73] Lin, N. P., Chang, C.-I., Chueh, H.-E., Chen, H.-J., and Hao, W.-H. (2008). A deflected grid-based algorithm for clustering analysis. *W. Trans. on Comp.*, 7(4):125–132.

[74] Linde, Y., Buzo, A., and Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95.

[75] Liu, Y., Li, Z., Xiong, H., Gao, X., and Wu, J. (2010). Understanding of internal clustering validation measures. In *2010 IEEE International Conference on Data Mining*, pages 911–916.

[76] Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., and Wu, S. (2013). Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics*, 43(3):982–994.

[77] Liu, Y., Peng, J., Chen, K., and Zhang, Y. (2006). An improved hybrid genetic clustering algorithm. In Antoniou, G., Potamias, G., Spyropoulos, C., and Plexousakis, D., editors, *Advances in Artificial Intelligence*, pages 192–202, Berlin, Heidelberg. Springer Berlin Heidelberg.

[78] Liu, Y., Wu, X., and Shen, Y. (2011). Automatic clustering using genetic algorithms. *Applied Mathematics and Computation*, 218(4):1267 – 1279.

[79] Lu, Y., Lu, S., Fotouhi, F., Deng, Y., and Brown, S. J. (2004a). Fgka: a fast genetic k-means clustering algorithm. In *Proceedings of the 2004 ACM symposium on Applied computing*, SAC '04, pages 622–623, New York, NY, USA. ACM.

[80] Lu, Y., Lu, S., Fotouhi, F., Deng, Y., and Brown, S. J. (2004b). Incremental genetic k-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics*, 5:172.

[81] Lucasius, C., Dane, A., and Kateman, G. (1993). On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasiblity and comparison. *Analytica Chimica Acta*, 282(3):647 – 669.

[82] Luna-Romera, J. M., Martínez-Ballesteros, M., García-Gutiérrez, J., and Riquelme, J. C. (2019). External clustering validity index based on chi-squared statistical test. *Information Sciences*, 487:1–17.

[83] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

[84] Maulik, U. and Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465.

[85] Maulik, U. and Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:1650–1654.

[86] Maulik, U., Bandyopadhyay, S., and Mukhopadhyay, A. (2011). *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*. Springer Publishing Company, Incorporated.

[87] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin.

[88] Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag, London, UK, UK.

[89] Milligan, G. and Cooper, M. (1988). A study of standardization of variables in cluster analysis. *J. Classification*, 5:181–204.

[90] Mukhopadhyay, A., Maulik, U., and Bandyopadhyay, S. (2015). A survey of multiobjective evolutionary clustering. *ACM Comput. Surv.*, 47(4):61:1–61:46.

[91] Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., and Coello, C. A. C. (2014). Survey of multiobjective evolutionary algorithms for data mining: Part ii. *IEEE Transactions on Evolutionary Computation*, 18(1):20–35.

[92] Murthy, C. A. and Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms. *Pattern Recogn. Lett.*, 17(8):825–832.

[93] Naldi, M. C., Campello, R. J. G. B., Hruschka, E. R., and de Carvalho, A. C. P. L. F. (2011). Efficiency issues of evolutionary k-means. *Appl. Soft Comput.*, 11:1938–1952.

[94] Nanda, S. J. and Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation*, 16:1–18.

[95] Nguyen, H., Louis, S. J., and Nguyen, T. (2019). Mgka: A genetic algorithm-based clustering technique for genomic data. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 103–110.

[96] Pakhira, M. K., Bandyopadhyay, S., and Maulik, U. (2004). Validity index for crisp and fuzzy clusters. *Pattern Recognition*, 37(3):487 – 501.

[97] Pal, N. R. and Bezdek, J. C. (1995). On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy Systems*, 3(3):370–379.

[98] Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67.

[99] Pelleg, D. and Moore, A. W. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 727–734, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[100] Perim, G. T., Wandekokem, E. D., and Varejão, F. M. (2008). K-means initialization methods for improving clustering by simulated annealing. In *Proceedings of the 11th Ibero-American conference on AI: Advances in Artificial Intelligence*, IBERAMIA '08, pages 133–142, Berlin, Heidelberg. Springer-Verlag.

[101] Pinto, R. and Gonçalves, G. (2022). Application of artificial immune systems in advanced manufacturing. *Array*, 15:100238.

[102] Poczeta, K., Kubuś, L., and Yastrebov, A. (2020). Multidimensional medical data modeling based on fuzzy cognitive maps and k-means clustering. *Procedia Computer Science*, 176:118 – 127. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020.

[103] Radcliffe, N. J. and Surry, P. D. (1995). Fitness variance of formae and performance prediction. volume 3 of *Foundations of Genetic Algorithms*, pages 51–72. Elsevier.

[104] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

[105] Rao, K. R. and Josephine, B. M. (2018). Exploring the impact of optimal clusters on cluster purity. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 754–757.

[106] Raposo, C., Antunes, C. H., and Barreto, J. P. (2014). Automatic clustering using a genetic algorithm with new solution encoding and operators. In Murgante, B., Misra, S., Rocha, A. M. A. C., Torre, C., Rocha, J. G., Falcão, M. I., Taniar, D., Apduhan, B. O., and Gervasi, O., editors, *Computational Science and Its Applications – ICCSA 2014*, pages 92–103, Cham. Springer International Publishing.

[107] Rezaei, M. and Fränti, P. (2016). Set matching measures for external cluster validity. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2173–2186.

[108] Rezaei, M. and Fränti, P. (2020). Can the number of clusters be determined by external indices? *IEEE Access*, 8:89239–89257.

[109] Robles-Berumen, H., Zafra, A., Fardoun, H. M., and Ventura, S. (2019). Leac: An efficient library for clustering with evolutionary algorithms. *Knowledge-Based Systems*, 179:117–119.

[110] Rojas-Thomas, J., Santos, M., and Mora, M. (2017). New internal index for clustering validation based on graphs. *Expert Systems with Applications*, 86:334 – 349.

[111] Rojas-Thomas, J., Santos, M., Mora, M., and Duro, N. (2019). Performance analysis of clustering internal validation indexes with asymmetric clusters. *IEEE Latin America Transactions*, 17(05):807–814.

[112] Saha, J. and Mukherjee, J. (2021). Cnak: Cluster number assisted k-means. *Pattern Recognition*, 110:107625.

[113] Saitta, S., Raphael, B., and Smith, I. F. C. (2007). A bounded index for cluster validity. In Perner, P., editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 4571 of *Lecture Notes in Computer Science*, pages 174–187. Springer Berlin Heidelberg.

[114] Saitta, S., Raphael, B., and Smith, I. F. C. (2008). A comprehensive validity index for clustering. *Intell. Data Anal.*, 12(6):529–548.

[115] Sheikholeslami, G., Chatterjee, S., and Zhang, A. (2000). Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 8(3):289–304.

[116] Sheng, W. and Liu, X. (2004). A hybrid algorithm for k-medoid clustering of large data sets. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 1, pages 77–82 Vol.1.

[117] Srinivas, M. and Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667.

[118] Tange, O. (2011). Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47.

[119] Tarekegn, A. N., Michalak, K., and Giacobini, M. (2020). Cross-validation approach to evaluate clustering algorithms: An experimental study using multi-label datasets. *SN Computer Science*, 1(5):263.

[120] Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition, Third Edition*. Academic Press, Inc., Orlando, FL, USA.

[121] Tseng, L. Y. and Bien Yang, S. (2000). A genetic clustering algorithm for data with non-spherical-shape clusters. *Pattern Recognition*, 33(7):1251–1259.

[122] Tseng, L. Y. and Bien Yang, S. (2001). A genetic approach to the automatic clustering problem. *Pattern Recognition*, 34(2):415 – 424.

[123] Žalik, K. R. and Žalik, B. (2011). Validity index for clusters of different sizes and densities. *Pattern Recognition Letters*, 32(2):221 – 234.

[124] Wang, W., Yang, J., and Muntz, R. R. (1997). Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, page 186–195, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[125] Wu, C. and Kang, Z. (2020). Robust entropy-based symmetric regularized picture fuzzy clustering for image segmentation. *Digital Signal Processing*, page 102905.

[126] Xiao, J., Yan, Y., Zhang, J., and Tang, Y. (2010). A quantum-inspired genetic algorithm for k-means clustering. *Expert Syst. Appl.*, 37:4966–4973.

[127] Xie, X. L. and Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(8):841–847.

[128] Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.

[129] Xu, R. and Wunsch, D., I. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.

[130] Yang, X., Cao, A., and Song, Q. (2006). A new cluster validity for data clustering. *Neural Processing Letters*, 23(3):325–344.

[131] Zhang, C., Bi, J., Xu, S., Ramentol, E., Fan, G., Qiao, B., and Fujita, H. (2019). Multi-imbalance: An open-source software for multi-class imbalance learning. *Knowledge Based Systems. In press, 2019.*

[132] Zhang, M.-L. and Zhou, Z.-H. (2009). Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, 31(1):47–68.

[133] Zhang, T., Ramakrishnan, R., and Livny, M. (1997). Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182.

[134] Zhao, Q. and Fränti, P. (2014). Wb-index: A sum-of-squares based index for cluster validity. *Data & Knowledge Engineering*, 92:77 – 89.

[135] Zhao, Q., Xu, M., and Fränti, P. (2009). Sum-of-squares based cluster validity index and significance analysis. In Kolehmainen, M., Toivanen, P., and Beliczynski, B., editors, *Adaptive and Natural Computing Algorithms*, pages 313–322, Berlin, Heidelberg. Springer Berlin Heidelberg.

[136] Zhu, E. and Ma, R. (2018). An effective partitional clustering algorithm based on new clustering validity index. *Applied Soft Computing*, 71:608 – 621.

[137] Zhu, S., Xu, L., and Goodman, E. D. (2020). Evolutionary multi-objective automatic clustering enhanced with quality metrics and ensemble strategy. *Knowledge-Based Systems*, 188:105018.