UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO INGENIERÍA INDUSTRIAL

UNA METAHEURÍSTICA PARA EL PROBLEMA DE CLUSTERING AUTOMÁTICO

POR

Víctor Suazo San Martín

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Industrial

Profesor Guía:

Carlos Contreras Bolton

Agradecimientos

En primer lugar, agradezco a Dios por mi vida, los dones que me ha otorgado y el camino que me ha hecho seguir. Agradezco a mis padres Lilian y Víctor, quienes me han formado, han dado amor y siempre han apoyado. Así como mi familia, que estuvo presente para darme un empujoncito para seguir adelante. Mi abuela Magaly que siempre me ha dado todo, mi hermana Bárbara que ha estado ahí desde siempre, mi abuelo Mario, mis tíos y tías, y mi pareja Micaela que me ha acompañado y apoyado estos últimos años.

Gracias a mis amigos que estuvieron ahí cuando los necesité, tanto para estudiar como para la distracción y el relajo. Si bien algunos han llegado hasta esta instancia conmigo, más de alguno se bajó del tren de mi vida por alguna razón, aun así, agradezco la compañía de estos y la de cada persona que al menos una tarde me acompañó.

Agradezco a mi hermano de otra madre, Cristóbal Arcos, siempre soñamos con este momento y mucho más. Primera meta lograda, aun me quedan hartas que alcanzar por ambos. Gracias amigazo por acompañarme un pedazo de mi vida, quizás el más importante.

Muchas gracias a mi profesor guía, Carlos Contreras, su dedicación y experiencia en el área han sido un grandísimo apoyo a mi memoria de título.

Esta memoria de título fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02).

Resumen

El problema de clustering consiste en subdividir un conjunto de datos en una cantidad determinada de grupos e identificar cuales datos tienen la similitud necesaria para pertenecer a un mismo grupo. Para resolver este problema es necesario saber de ante mano la cantidad exacta de grupos a asignar, esto en la práctica, generalmente, no sucede con exactitud sobre todo con conjuntos grandes de datos. Dada esta problemática surge el problema de clustering automático (PCA), que consiste en determinar automáticamente la cantidad de grupos en que se deben subdividir datos de naturaleza desconocida, además de asignar el grupo idóneo para cada uno de estos. Por tanto, el presente trabajo tiene como objetivo resolver el PCA. Para ello, se propone un algoritmo genético multioperador con cuatro operadores de mutación y cuatro operadores de cruzamiento sobre una estructura novedosa de representar la solución, que comprende la aplicación del módulo de la cantidad de grupos sobre la presolución generada. El algoritmo propuesto es validado en un conjunto de diez instancias de la literatura de distintas dimensiones. Además, se utilizan las métricas del índice Davies-Boulding, índice de separación compacta y el índice de la silueta para medir el rendimiento en comparación al estado del arte. Los experimentos computacionales muestran resultados prometedores, puesto que el algoritmo propuesto es competitivo con algoritmos de la literatura en las instancias consideradas y las tres métricas.

Abstract

The clustering problem consists of subdividing a set of data into a certain number of groups and identifying which data have the necessary similarity to belong to the same group. To solve this problem it is necessary to know in advance the exact number of groups to assign, but in practice, this generally does not happen accurately, especially with large data sets. Given this problem arises the automatic clustering problema (ACP), which consists of automatically determining the number of groups in which data of unknown nature should be subdivided, in addition to assigning the ideal group for each of these. Therefore, the present work aims to solve the ACP. For this purpose, a multioperator genetic algorithm with four mutation operators and four crossover operators is proposed on a novel structure of representing the solution, which comprises the application of the modulus of the number of groups on the generated presolution. The proposed algorithm is validated on a set of ten literature instances of different dimensions. In addition, the metrics of Davies-Boulding index, compact separation index and silhouette index are used to measure the performance in comparison to the state-of-the-art. The computational experiments show promising results, since the proposed algortim is competitive with the algorithms from the literature in the considered instances and the three metrics.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Capítulo 1. Introducción

En este capítulo se presenta una introducción al problema de clustering, junto con la motivación para este trabajo. Además, se presentan los objetivos generales y específicos de la investigación, y la estructura del documento.

1.1. Motivación

Cada vez se procesan más datos a medida que avanzan las diversas ciencias y tecnologías. Estos datos pueden llegar a ser realmente significativos para un estudio, si se logran identificar de manera correcta sus características. Por ello, el agrupamiento o clustering de datos ha tomado más importancia con el tiempo, ocupándose cada vez en más campos del conocimiento (Jain, 2010). En términos simples, el problema consiste en tomar una población de datos de cualquier tipo y lograr agruparlos por sus similitudes. De manera de lograr asignar ciertas cualidades a conjuntos de muestras, o en otras palabras ejecutar una minería de datos. Esto es aplicado a diversos campos de estudio, como investigación de operaciones, informática, biología, marketing, entre otros (Jain, 2010).

El problema clásico, llamado problema de clustering (PC) o también llamado clustering supervisado (no automático) generalmente es resuelto por el algoritmo K-Means (MacQueen, 1967), que posee sensibilidad en su iniciación ya que necesita un input de los k grupos que se desean generar. Además, sus salidas tienden a entregar óptimos locales (Alotaibi, 2022). Por tanto, el problema clásico se puede hacer más desafiante si el número de clúster pasa a ser una decisión. Así, el problema pasa a llamarse, problema de clustering automático (PCA) y consiste en identificar automáticamente la cantidad de grupos en un conjunto de datos con naturaleza desconocida. Para lograr esto, se define una distancia entre datos, que puede ser intra-grupo para minimizar y una distancia entre distintos grupos a maximizar. De esta forma, los datos que son parte de un mismo grupo son similares (Deeb et al., 2022). Por ende, la gran diferencia y dificultad que asume el PCA es la exactitud para generar automáticamente el número más adecuado de grupos en un conjunto de datos de naturaleza desconocida y, al mismo tiempo, dividir los objetos de datos en grupos apropiados (Ikotun & Ezugwu, 2022).

En esta memoria de título, se propone una metaheurística basada en un algoritmo genético multioperador que utiliza cuatro operadores de cruzamiento y cuatro operadores de mutación, con el objetivo de trabajar de manera más eficiente y lograr escapar de los óptimos locales. Utilizando una interesante y novedosa representación.

1.2. Objetivo general

Implementar una metaheurística para resolver el PCA.

1.3. Objetivos específicos

- Desarrollar una revisión de la literatura con relación al estado del arte de los métodos propuestos para resolver el PCA.
- Diseñar e implementar una metaheurística para resolver el PCA.
- Diseñar y ejecutar experimentos computacionales para el PCA.
- Analizar los resultados y compararlos con el estado del arte.

1.4. Estructura del documento

El documento es organizado de la siguiente manera. En el Capítulo 2 se realiza una descripción detallada del PCA y la formulación del modelo matemático, una revisión de literatura y las métricas de evaluación. Luego, en el Capítulo 3 se presenta la metaheurística propuesta, explicando la estructura general de esta y sus principales fases. El Capítulo 4 entrega los experimentos computacionales realizados, detallando su implementación, describiendo las instancias utilizadas y la calibración de los parámetros, además de un análisis de los resultados obtenidos. Finalmente, en el Capítulo 5, se presentan las conclusiones obtenidas.

Capítulo 2. Problema de clustering automático

En este capítulo se presenta una descripción del problema de clustering automático. Además, se revisa la literatura más relevante y reciente asociada al PCA, enfocándose en algoritmos del estado del arte que resuelven el problema y sus características.

2.1. Descripción del problema

El PCA es una generalización del PC, ya que ambos poseen el mismo objetivo, agrupar datos similares en conjuntos coherentes. El PC es resuelto popularmente con el algoritmo K-means (Sinaga & Yang, 2020), donde los algoritmos requieren intervención humana para definir el número de *k* clústeres en que se subdividen los datos, el PCA supera la sensibilidad de la cantidad de grupos que existen en la instancia dada, automatizando el proceso de agrupamiento. Esto permite que los algoritmos determinen el número óptimo de clústeres, con el objetivo principal de maximizar la similitud de los datos pertenecientes a un clúster (intra-cluster) y minimizar la similitud entre los diferentes clústeres (inter-cluster) (Ezugwu et al., 2020). De esta forma, la metodología para resolver el PCA permite una exploración más exhaustiva y objetiva de los datos, revelando información importante y facilitando la toma de decisiones en una amplia gama de aplicaciones. Entonces, para ambos problemas existe un valor *k* de clústeres (manual o automático) para un conjunto de datos de naturaleza desconocida, en donde la salida proporciona etiquetas para cada dato que lo asignan a un clúster específico.

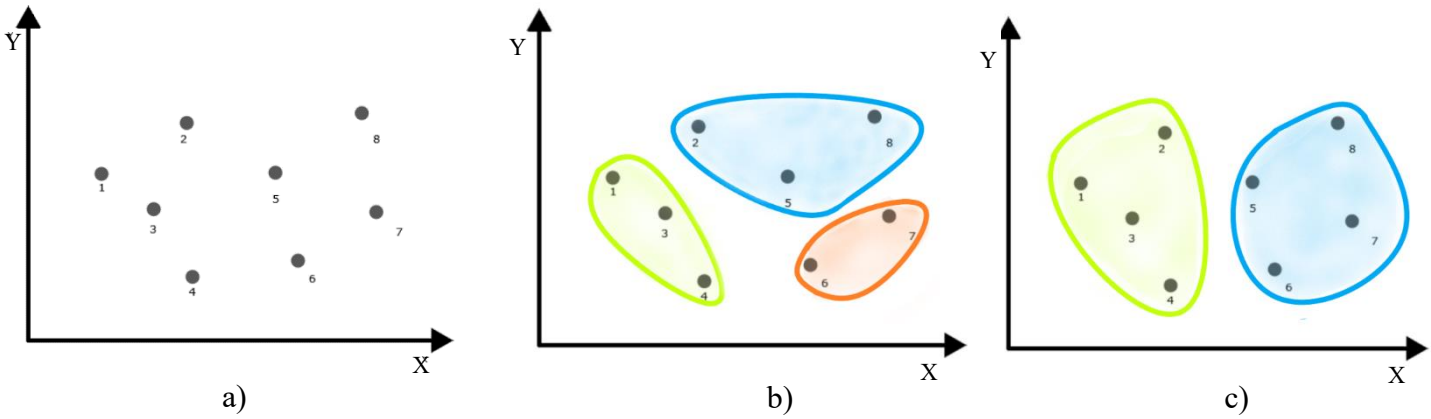

Figura 1: Problema de Clustering.

La Figura 1 muestra un ejemplo de un conjunto de datos que corresponde a ocho elementos con dos características, donde a) presenta estos datos ubicados en un espacio bidimensional. Mientras, en b) es aplicado el algoritmo K-means; con *k*=3, y la solución corresponde a {1, 3, 4}, {2, 5, 8}, {6, 7}. Por otro lado, en c) se aplica un algoritmo para el PCA, con un valor de *k* determinado

automáticamente. Como se puede observar, al tener que agrupar los datos con un valor *k* específico de grupos, se obliga al algoritmo a subdividir los datos. En cambio, cuando se resuelve como PCA, se busca agrupar los datos a partir de un objetivo claro que es dado por una función objetivo o una función fitness, para intentar evaluar cada posible solución con el fin de buscar el valor más cercano al óptimo.

2.2. Problema de clustering

El PC es popularmente resuelto por el algoritmo K-means, que se encuentra entre los diez algoritmos más utilizados en el proceso de minería de datos. Además, es ampliamente aceptado por su implementación sencilla, baja complejidad computacional, flexibilidad y eficiencia (Ikotun & Ezugwu, 2022). Formalmente, se asignan *k* puntos aleatorios en el espacio, como centros de clúster, llamados centroides. Luego, se utiliza el cuadrado de la distancia Euclideana desde cada dato hacia los centroides, asignándolos al más cercano. Con todos los datos asignados, se calcula el promedio de los datos pertenecientes a cada clúster, para redefinir la ubicación de los centroides. Este proceso itera hasta que la función de objetivo sea mínima. Ikotun & Ezugwu (2022) presentan el modelo matemático basado en el algoritmo de K-means en las Ecuaciones (1) – (3).

$$\text{Minimizar} \qquad F(A,C) = \sum_{l=1}^{k}\sum_{i=1}^{n} a_{il}\, d\left(X_i, C_l\right) \tag{1}$$

$$\text{sujeto a:} \qquad \sum_{l=1}^{k} a_{il} = 1, \qquad 1 \le i \le n \tag{2}$$

$$a_{il} \in \{0,1\}, \qquad 1 \le i \le n, \qquad 1 \le l \le k \tag{3}$$

Dado un conjunto de datos X que contiene n objetos numéricos tales que $X_i = (x_1, x_2, \ldots, x_n)$ y un número entero k que representa la cantidad de grupos en que se subdivide X, con $k \le n$. Donde A es una matriz de tamaño $n \times k$. Además, $C = C_1, C_2, \ldots, C_k$ representa un conjunto de objetos en el mismo grupo (centroides) y d representa la distancia cuadrada entre dos objetos. Se busca minimizar la función objetivo de la Ecuación (1), sujeta a las restricciones (2) que restringen que cada dato sea asignado a un clúster. Finalmente, las restricciones (3) restringen los valores que se le pueden asignar a las variables.

2.3. Métricas

La validación de los resultados obtenidos de los algoritmos de clustering es una etapa crucial en el proceso de clustering (Ikotun & Ezugwu, 2022). Para lograr esto, se utilizan métricas que son índices de validación de clústeres que evalúan la calidad de la partición generada. En el estado del arte existen varias métricas, las cuales generalmente se clasifican como internas y externas. Las internas son pensadas para aplicaciones de la vida real, y buscan determinar qué tan bien se ajustan los clústeres generados por los algoritmos a las estructuras subyacentes de los conjuntos de datos, sin contar con información previa sobre las clases presentes en los datos. Mientras, las externas requieren conocimiento previo sobre el conjunto de datos, ya que miden la afinidad de la solución con una estructura predefinida de este conjunto, que en la práctica generalmente se desconoce (Ezugwu et al., 2020). Este trabajo, emplea tres métricas internas para evaluar la calidad de los resultados de la metaheurística propuesta, las métricas son: índice Davies-Bouldin, índice Separación Compacta, e índice de Silueta.

El índice Davies-Boulding (DB) propuesto por Davies & Bouldin (1979) consiste en evaluar la cohesión intra-clúster, es decir, el promedio de las distancias de todos los datos pertenecientes a un mismo clúster, con su respectivo centroide, así como la distancia inter-clúster con los centroides de dos clústeres. Su propósito es determinar la calidad de los resultados de clustering obtenidos mediante un algoritmo de clustering. DB es calculado mediante la Ecuación (4), donde k representa el número de clústeres, mientras que i y j son las etiquetas asignadas a cada clúster. $d(x_i)$ y $d(x_j)$ representan el promedio de las distancias de los puntos de datos respecto a su respectivo centroide dentro de cada clúster. Por su parte, $d(v_i, v_j)$ se refiere a la distancia inter-clúster entre los centroides de los clústeres v_i y v_j. La minimización de este índice conduce a una mejor compacidad y mayor separación entre los clústeres.

$$DB = \frac{1}{k}\sum_{i=1}^{k} \underset{i \neq j}{Max} \left\{ \frac{d\left(x_i\right) + d\left(x_j\right)}{d\left(v_i, v_j\right)} \right\} \tag{4}$$

El índice Separación Compacta (CS) (por sus siglas en inglés, Compact-Separated), utiliza la proporción entre la dispersión dentro del clúster y la separación entre clústeres para evaluar la calidad de un resultado de clustering (Chou et al., 2004). Al igual que en DB, valores más bajos del índice CS indican clústeres bien separados y más compactos. Este índice es considerado más efectivo para manejar clústeres con diferentes densidades, dimensiones y tamaños. CS es calculado mediante la

Ecuación (5), donde existe una partición de un conjunto de datos $X=\{x_j: j=1,2,...,n\}$ en k clústeres, con i de uno a k, tal que Q_i es el conjunto de elementos asignados al i-ésimo clúster, $|Q_i|$ es la cantidad de elementos dentro del clúster, v_i es el centroide del clúster i y d es la distancia entre dos elementos.

$$CS = \frac{\sum_{i=1}^{k}\left[\frac{1}{|Q_i|}\sum_{x_j \in Q_i} \max_{x_c \in Q_i}\left\{d\left(x_j, x_c\right)\right\}\right]}{\sum_{i=1}^{k}\left[\min_{j \in c : j \neq i}\left\{d\left(v_i, v_j\right)\right\}\right]} \quad (5)$$

El índice de Silueta (SI) (por sus siglas en inglés, Silhouette Index) es una medida utilizada para evaluar la compacidad y separación de los clústeres. SI consiste en calcular el ancho de silueta para cada entidad i perteneciente a un clúster X_j (donde j = 1, ..., k). SI es calculado mediante la Ecuación (6). Aquí, $s(i)$ proporciona una medida de la probabilidad de que esa entidad i pertenezca al clúster X_j (Rousseeuw, 1987). Este cálculo se basa en dos medidas: $a(i)$ y $b(i)$. La medida $a(i)$ representa la distancia promedio entre la entidad i y las demás entidades dentro del mismo clúster, mientras que la medida $b(i)$ indica la distancia promedio mínima entre la entidad i y todas las entidades agrupadas en otros clústeres X_k (c = 1, ..., k; $c \neq j$).

$$s(i) = \frac{(b(i) - a(i))}{Max\{a(i), b(i)\}} \quad (6)$$

Por otro lado, el estado del arte utiliza distintas métricas internas además de las mencionadas, algunas se exponen a continuación. El índice PBM (por las iniciales de sus creadores Pakhira, Bandyopadhyay y Maulik) se compone de tres factores que buscan encontrar un equilibrio. El primero busca mantener el número de clústeres lo más bajo posible, controlando así la divisibilidad del sistema de clústeres. El segundo es una división de la suma de las distancias intra-clúster para el conjunto de datos completo tratado como un solo clúster, entre la suma de las distancias intra-clúster para el sistema de k clústeres, controlando así la compacidad de los clústeres individuales. El tercero representa la máxima separación entre un par de clústeres asegurando una gran separación entre al menos dos clústeres (Pakhira et al., 2004).

El índice Xie-Beni (XB) busca minimizar la división entre las distancias intra-cluster e inter-cluster. Mide la distancia inter-cluster como la distancia cuadrática mínima entre los centros de los grupos y la intra-cluster como la distancia cuadrática media entre los objetos de datos en un grupo y su centro de grupo (Xie & Beni, 1991).

El índice de Dunn también es una división, que tiene como objetivo maximizar la separación entre los clústeres buscando la distancia mínima entre elementos de distintos clústeres (numerador), y

minimizar la compacidad de cada clúster tomando la distancia máxima que poseen dos puntos del mismo clúster (denominador) (Dunn,1973).

El Coeficiente de Partición es otra métrica interna, que mide la superposición de clústeres o también llamada compartición de membresía. Esto se aplica a algoritmos difusos, los cuales asignan un valor de pertenencia de los elementos a un clúster, generando así una asignación no binaria de un elemento diferentes clústeres (Bezdek, 1974).

Como es mencionado, las métricas externas son utilizadas cuando se posee información sobre la clasificación de los datos. A continuación, se detallan brevemente algunas. El índice de Rand expresa la similitud de la partición generada con la ya conocida en un rango de 0 a 1, con 1 como una coincidencia perfecta. Este índice es la proporción entre la suma de verdaderos positivos y negativos con el total de casos, estos últimos incluyen además a los falsos positivos y negativos. Los verdaderos son pares de datos bien asignados según la documentación y los falsos son los mal asignados, en ambos casos, se aplica que si el par de elementos analizados están en el mismo clúster, son positivos y en el caso contrario, negativos (Rand, 1971).

La F-medida es una métrica que utiliza dos submétricas, precisión y la recuperación. La precisión entrega una proporción de los elementos de un clúster que están bien asignados en comparación al total de elementos de este clúster. La recuperación entrega una proporción de los elementos de una clase (grupo previamente identificado) asignados a un clúster, entre los elementos totales de esta clase que deberían ir en ese clúster (Van Rijsbergen, 1979).

2.4. Revisión de la literatura

Si bien este estudio trata sobre el PCA, en la actualidad aún hay autores que directamente buscan mejorar o superar el popular algoritmo K-means para el PC, como en Deeb et al. (2022). Los autores abordan el PC usando una metaheurística de Agujero Negro, que es un algoritmo que se basa en una analogía del fenómeno natural de agujeros negros, considerando a distintas soluciones del problema como estrellas. El algoritmo propuesto es comparado con otros algoritmos mediante métricas basadas en la distancia Euclidiana con la mejor solución, la peor, la media, y la desviación estándar, superando entre ellos a K-means. Si bien el algoritmo propuesto supera el agrupamiento de varios otras metaheurísticas, éstas poseen la sensibilidad de interpretación humana de cuantos grupos realmente hay en los datos. Por lo que varios autores han optado por investigar el PCA.

Zhu et al. (2022) proponen una metaheurística para resolver el PCA, que llaman agrupamiento automático basado en un algoritmo de optimización de búsqueda de armonía de parámetros dinámicos.

Este estudio genera una gran exploración de resultados a través de parámetros dinámicos que cambian según el número de iteraciones, dando diversificación a las posibles soluciones, otorgando así la capacidad necesaria a la búsqueda armónica para salir de óptimos locales. No obstante, también hacen recurso de la memoria armónica que posee la versión clásica de este último algoritmo, para explotar los resultados locales (ya obtenidos). Utilizan las métricas PBM, DB y XB en su función fitness y comparan sus resultados con otros estudios mediante la agrupación de datos de la vida real, así como en la segmentación de imágenes en escala de grises.

Por otro lado, Jahan & Hasan (2021) proponen una nueva metaheurística, que se basa en el promedio del SI para identificar la cantidad de grupos óptima de la población de datos, para luego agruparlos con un algoritmo de clustering o con K-means. Esta última forma híbrida, la llaman Meskat-Mahmudul Extended K-Means (MMK). Miden su rendimiento mediante el Coeficiente de Partición, XB, índice de Dunn, entre otros. Concluyen que MMK es un algoritmo robusto y mucho mejor que otros que se usan para problemas de clustering.

Dhas & Gomathi (2019) generan un método híbrido con los algoritmos de medias C difusas jerárquicas generalizadas (GHFCM por sus siglas en inglés: Generalized Hierarchical fuzzy C means) y optimización del lobo gris (GWO por sus siglas en inglés: Grey Wolf Optimization). Primero, utilizan GWO para identificar el punto de agrupamiento de datos y luego GHFCM para lograr agrupar estos. Para evaluar su enfoque miden la F-medida, el índice de Rand y el consumo de tiempo. El algoritmo propuesto supera algoritmos de clustering existentes.

Dado que este estudio utiliza un algoritmo genético. En la literatura se han propuesto diferentes variantes de algoritmos genéticos para mejorar la precisión y automatización del clustering en el análisis de clústeres (Ezugwu et al., 2020). Así, Wang y Wu (2010) desarrollan un algoritmo genético caótico que utiliza la propiedad ergódica de fenómenos caóticos para optimizar la población inicial, acelerando así los procesos de selección, cruce y mutación. Por otro lado, Dutta et al. (2012) implementan un algoritmo genético de características mixtas que aborda la presencia de características continuas y mixtas en los conjuntos de datos, optimizando simultáneamente la distancia intra-clúster y la distancia inter-clúster (Ezugwu et al., 2020). Agustín-Blas et al. (2012) abordan el PCA con un algoritmo genético de agrupamiento, que utiliza una codificación basada en particiones y, las métricas DB y SI como función objetivo. Estos son solo algunos ejemplos de cómo los algoritmos genéticos se aplican al clustering de datos para mejorar la precisión y la automatización del proceso.

Entre los estudios más recientes está el híbrido que proponen Ikotun & Ezugwu (2022), en que mediante el algoritmo de búsqueda de organismos simbióticos generan una búsqueda de la cantidad

de centroides óptima, para entregar el valor de *k* clústeres como input al algoritmo K-Means. Esta metaheurística que llaman SOS-Kmeans (por sus siglas en inglés, Symbiotic Organisms Search) utiliza métodos de búsqueda basados en la naturaleza de las especies para generar individuos. El trabajo utiliza las métricas DB y CS como funciones fitness a minimizar, logran superar al clásico algoritmo de SOS, K-means y algunas otras metaheurísticas.

Capítulo 3. Metaheurística propuesta

Este capítulo presenta la metaheurística para resolver el PCA, realizando una explicación general del algoritmo junto a la representación de la solución. Finalmente, se describe en forma detallada cada fase de la metaheurística.

3.1. Estructura general del algoritmo propuesto

La metaheurística propuesta es un algoritmo genético (AG) adaptado al PCA. La versión clásica de este algoritmo es propuesta por Holland (1975). El AG genera una búsqueda iterativa de soluciones a través de procesos de selección, cruzamiento y mutación de individuos (soluciones), a partir de una población inicial de individuos. Estos procesos llamados también operadores son los que dan la capacidad de explotación y exploración al AG. Típicamente, el AG utiliza operadores únicos para el cruzamiento y la mutación, y estos son los encargados de generar nuevas soluciones y mantener una diversidad en la población. Sin embargo, estos a veces no son suficientes para escapar de óptimos locales o tienen convergencia muy lenta. Así, en la literatura se encuentra que aplicar múltiples operadores tanto para cruzamiento y/o mutación, da pie para generar una potencial mejor sinergia entre estos, dando al algoritmo mayor espacio de soluciones y pueda otorgar mejor calidad a los resultados (Contreras-Bolton et al., 2016).

El Algoritmo 1 presenta el pseudocódigo del AGM propuesto. Este necesita ciertos parámetros para iniciar, como el tamaño de la población (N^p), cantidad máxima de iteraciones (I_{max}) como criterio de terminación de la búsqueda, probabilidad P_c para que ocurra un cruzamiento entre individuos y la probabilidad P_m para que se ejecute el operador de mutación. En las líneas 1-4, se realiza la creación de la población inicial y la evaluación de cada individuo mediante la función fitness. Luego, desde la línea 5 a la 6, se genera un bucle que itera hasta que se cumple el criterio de término establecido. Se aplica el primer operador genético, selección, que realiza un torneo de S individuos aleatorios de la población, donde los individuos ganadores de cada torneo son seleccionados para los siguientes operadores (Eiben & Smith, 2015). En las líneas 9 y 10, se aplica alguno de los cuatro operadores propuestos de cruzamiento, si se cumple la probabilidad P_c. En las líneas 11 y 12, se aplica alguno de los cuatro operadores propuestos de mutaciones si se cumple la probabilidad P_m. Continuando, en la línea 13 se aplica una búsqueda local (LS) al mejor individuo de la población encontrado hasta el momento. En las líneas 14 y 15, se evalúan los individuos de la nueva población. La última línea perteneciente al bucle general del algoritmo (línea 16) posee el operador de Elitismo, que selecciona

a los mejores *E* padres de la población antigua y remplaza a los *E* peores hijos de la nueva población. Finalmente, el algoritmo entrega el mejor individuo obtenido, es decir, la mejor solución encontrada.

Algoritmo 1: Metaheurística Propuesta

Input: N^p, I_{max}, P_c, P_m
Output: Mejor solución global

```
1  : g←0
2  : for t ← 0 to N^p do
3  :     P_t ← población-inicial()
4  :     evaluar-individuo(P_t)
5  : while g < I_max:
6  :     g ← g+1
7  :     for t ← 0 to N^p do
8  :         (i_1, i_2) ← Torneo (P^(g-1))
9  :         if random(0,1) < P_c:
10 :             P_t^g ← cruzamiento (i_1, i_2)
11 :         if random(0,1) < P_m:
12 :             P_t^g ← mutación (P_t^g)
13 :     LS (P^g_best)
14 :     for t ← 0 to N^p do
15 :         evaluar-individuo (P_t^g)
16 :     P^g ←Elitismo (P^(g-1))
17 : return P^g_best
```

3.2. Representación de la solución

La solución de cada individuo consiste en un cromosoma de n+1 alelos. El primer alelo corresponde al número de clústeres de la solución y los siguientes n alelos corresponden al clúster asignado al i-ésimo dato con $i \in \{1, 2, 3, \ldots n\text{-}1, n\}$. Las soluciones que se generan poseen k clústeres que van desde $K_{min} = 2$, a $K_{max} = \sqrt{n}$, extremos apropiados sugeridos por Ghoneimy et al. (2021).

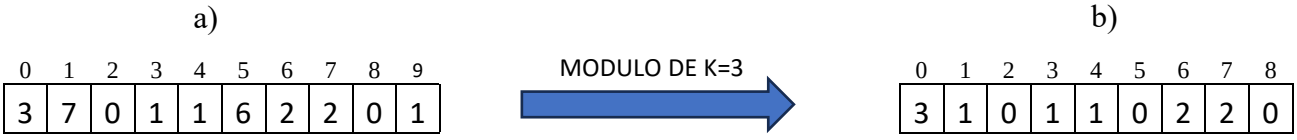

Figura 2: Ejemplo de la representación de la solución.

La Figura 2a) representa la solución de la Figura 1b). El valor del primer alelo indica que hay tres clústeres. Los valores de las posiciones 2, 5 y 8 indican que los datos pertenecen al clúster 0, y análogamente los otros elementos pertenecen al clúster que indica su valor. Los alelos, sin considerar la primera posición, en primera instancia poseen valores entre cero y $K_{max,}$ como se muestra en a). Sin

embargo, si el valor del alelo es más grande que el número de clúster de la solución (primer alelo). Entonces, se debe aplicar el módulo del primer valor (k), de esta forma, cada valor se convierte en el resto que proporciona una división entera entre el valor del alelo y k. Como se observa en la Figura 2a), los valores de los alelos de las posiciones 1(7) y 5(6) son mayor a 3. Entonces, se aplica el módulo de k y se obtiene una solución con valores de cero a k-1, en este caso los valores son 1 y 0, respectivamente.

3.3. Población Inicial

La población inicial es creada por N^p individuos a través de tres métodos. El primer método, consiste en generar cinco individuos con el algoritmo K-means con tamaños de grupo 2, 3, 4, 5 y K_{max}. El segundo, crea dos individuos a partir de K-means, cada uno con un k aleatorio distinto al del otro, tal que $6 \leq k < K_{max}$. El último método crea soluciones aleatorias para la población restante. Además, existe una probabilidad α que se elijan aleatoriamente copias de las soluciones proporcionadas por los dos primeros métodos para ser repetidas en la población inicial. Con la finalidad de balancear la diversidad de la población.

3.4. Función Fitness

La función fitness está encargada de evaluar la calidad de la solución que posee cada individuo. El AGM utiliza los índices DB, CS y SI para evaluar la compacidad y separación de los clústeres. Además, esta fase incluye la aplicación del módulo del primer elemento k a los n elementos restantes de la solución. Así, luego en los operadores de cruzamiento y mutación, los valores de los alelos en los nuevos individuos van acordes al número de clústeres que posee cada solución, es decir, no superan k-1.

3.5. Cruzamiento

Los métodos de cruzamiento permiten explorar nuevas soluciones/descendencias a través de la información genética que poseen los individuos de la población generada anteriormente (padres) (Umbarkar et al., 2015). En este trabajo se utilizan dos cruzamientos de la literatura y un cruzamiento creado exclusivamente para el PCA, que se aplica de dos formas distintas. Cada cruzamiento cuenta con una probabilidad de ocurrencia, siendo las probabilidades c_1, c_2, c_3 y c_4 para 1-Point Crossover (1PC), Partially Matched Crossover (PMX), Cruce PCA al padre 1 (CPCA1) y al padre 2 (CPCA2), respectivamente. Además, se debe cumplir que $c_1 + c_2 + c_3 + c_4 = 1$

- **1PC:** Este operador consiste en seleccionar aleatoriamente un punto de cruce en dos padres de manera de fragmentar la información de estos, obteniendo así la decendencia combinada de dos individuos (Eiben & Smith, 2015). En la Figura 3, se presentan dos individuos padres basados en la Figura 1 y se les aplica un punto de cruce entre cuatro y cinco, generando la descendencia. Como se observa en este caso, el Descendiente 2 posee un *k* igual dos, pero los genes de los puntos siete y ocho indican que estos datos pertenecen al clúster tres. Por tanto, se modifica *k* según la información que poseen el resto de los genes de la solución. Así, el Descendiente 1 y el Descendiente 2 quedan como muestra la figura. Luego, en la fase de evaluación de los individuos al Descendiente 2, se le aplica el módulo debido a que dos de sus alelos (6 y 7) están asignados a grupos fuera del rango de *k-1*. Obteniendo ambos alelos el valor de 0.

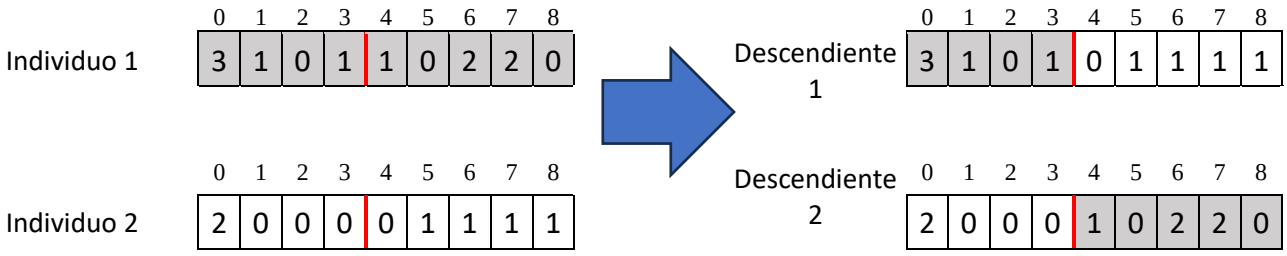

Figura 3: Ejemplo de 1PC.

- **PMX:** Este operador fue propuesto por Goldberg & Linger (1985) y trabaja en representaciones de permutacion. Este operador consiste en crear descendencia utilizando una porción de la información genética de los padres entre dos puntos generados aleatoriamente. Donde los alelos repetidos son remplazados según los mapeos de este operador. Para estos el operador considera dos padres. Genera dos listas auxiliares para ambos padres que contienen la posición que posee cada elemento de la secuencia de cada padre, es decir, si la primera posición la ocupa el valor tres, en la lista auxiliar la posición tres tendrá el valor de uno. Luego, toma dos puntos distintos al azar que son los índices extremos de la porción a intercambiar. Seguido, se recorre entre los dos puntos en ambos padres e intercambia la información entre los individuos. Al mismo tiempo, se utilizan las listas auxiliares para identificar la posición del padre que ya posee el nuevo valor que se asigna al gen correspondiente y asigna a esta posición el valor antiguo del gen. De esta forma, el nuevo valor asignado a un gen no se repite en otro dentro de la misma solución. Como este trabajo, no tiene representación de permutación. Entonces, el operador crea descendencia con el intercambio de las porciones correspondientes,

pero, genera de cero a *k* cambios en elementos aleatorios de la solución, asignándoles valores entre cero y *k*. Además, en este cruzamiento se mantiene el valor *k* original de cada padre en su descendiente directo, ya que para este no se considera el primer valor de la solución. En la Figura 4 se muestra la aplicación entre las soluciones de la Figura 1b) y Figura 1c), Individuo 1 y 2, respectivamente. En este ejemplo se toma la porción entre los puntos 4 y 6 para generar el cruce.

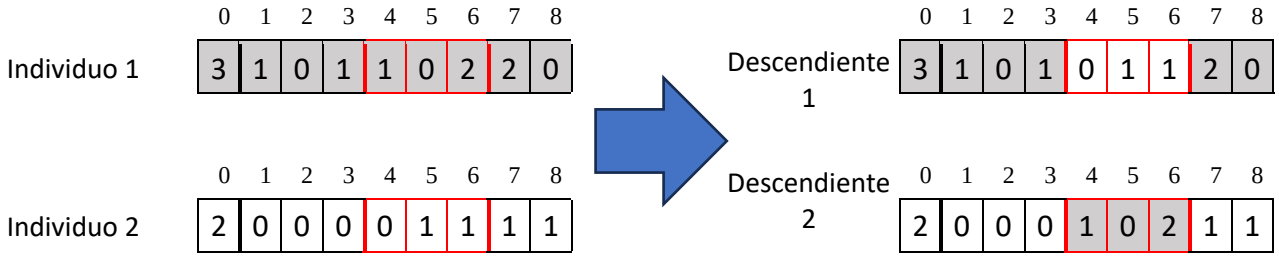

Figura 4: Ejemplo de PMX.

- **CPCA1 y CPCA2:** Estos cruzamientos son creados exclusivamente para este estudio, con la finalidad de alterar el número de clústeres. Así, se modifica su valor de forma aleatoria, entregando un nuevo valor en un rango entre K_{min} y K_{max}. El cruce es utilizado de dos formas, con probabilidad c_4 para el primer padre y c_5 para el segundo padre.

3.6. Mutación

La mutación es un operador que sirve para escapar de los óptimos locales (diversificación) y en algunos casos también trabaja intensificando como una búsqueda local, de manera que la descendencia pueda poseer información diferente a los padres (Hong et al,. 2000). Este trabajo utiliza cuatro tipos de mutaciones, UniformInt (UI) (Goldberg, 1989), ShuffleIndexes (SFI) (Abdoun et al., 2012), una búsqueda local (LS) y una mutación para el PCA (MPCA). Estas mutaciones tienen probabilidad de aplicación: m_1, m_2, m_3 y m_4, respectivamente. Además, se debe cumplir que $m_1 + m_2 + m_3 + m_4 = 1$. A continuación, se presenta el detalle de las mutaciones propuestas.

- **UI:** Esta mutación remplaza valores del padre de forma uniforme. Con una probabilidad determinada, cada gen cambia su valor por uno generado a través de una distribución uniforme que posee una cota inferior y una superior. Para este trabajo estas cotas están dadas por K_{min} y K_{max}, respectivamente, y la probabilidad independiente de cambiar el valor de cada elemento de la solución es 50%.
- **SFI:** Esta mutación intercambia el valor entre los genes de un padre. Con una probabilidad determinada, cada gen intercambia su información con la de otro gen aleatorio. Generando una

descendencia con la misma cantidad de clústeres del padre, pero, con distintas asignaciones de clúster para los datos. El operador utiliza una probabilidad independiente de 50% para alterar el valor de cada elemento de la solución.

- **LS:** Este operador es una búsqueda local creada específicamente para este problema. Así, LS comienza a partir de la cantidad de *k* clústeres que posee la solución entregada por el individuo, toma un gen aleatorio en el rango [1, *n*] y calcula la distancia de éste hacia el centroide del clúster que tiene asignado. Luego, identifica si este gen está más cerca del centroide de otro clúster. Si se encuentra una distancia menor entre los otros clústeres, entonces, se evalúa la función fitness la potencial solución considerando la modificación del clúster en el gen especificado. Si esta es mejor que la solución actual del individuo, entonces, el gen es modificado. De no cumplirse ambas situaciones, el individuo permanece sin cambios.
- **MPCA**: Esta mutación es la misma que el operador de cruzamiento CPCA. De esta forma, existe mayor probabilidad de alterar los *k* clústeres y así generar un mayor espacio de búsqueda.

Capítulo 4. Experimentos computacionales

Este capítulo presenta los resultados computacionales del algoritmo propuesto, comparándolos con los algoritmos del estado del arte que resuelven el PCA. Además, se presenta la calibración y las instancias utilizadas en los experimentos computacionales para la implementación del algoritmo propuesto.

4.1. Descripción de las instancias

Para validar el AGM, se utilizaron 10 instancias de UCI Machine Learning Repository (Kelly et al., 2023) como conjunto de datos de referencia. En la Tabla 1 se muestra un resumen de las instancias, donde las características indican las dimensiones de cada objeto perteneciente a un conjunto de datos. Además, el número de clústeres es la cantidad de grupos definidos para la instancia en el problema clásico, los cuales no necesariamente son el valor óptimo de agrupaciones para el PCA.

Tabla 1: Resumen de las instancias.

Conjunto de datos	Características	N° de Clústeres	N° de Objetos
Aggregation	2	7	788
Breast	9	2	699
Compound	2	6	399
Flame	2	2	240
Glass	9	7	214
Iris	4	3	150
Leaves	64	100	1600
Thyroid	5	2	215
Wine	13	3	178
Yeast	8	10	1484

4.2. Configuración de la implementación y la calibración

El algoritmo propuesto fue implementado en Python 3.9, los experimentos finales y calibración fueron ejecutados en la infraestructura de super computación del NLHPC. Este consiste en un procesador 2× Intel Xeon Gold 6152 con 2.10 GHz, cada uno con 22 núcleos y 768 GB de RAM. Todos los experimentos fueron ejecutados en un solo hilo utilizando el sistema operativo CentOS Linux 7 (64-bit), a excepción de la calibración que utilizó 40 hilos.

El AGM necesita varios parámetros para trabajar, los cuales deben ser calibrados para un óptimo funcionamiento del algoritmo y así obtener el mejor rendimiento posible. Esto se hizo mediante el

paquete de R, IRACE en su versión 3.4.1 (López-Ibáñez et al., 2016). IRACE ejecuta 5000 iteraciones con las 10 instancias dadas. Finalmente, la calibración de IRACE tomó aproximadamente 50,59 horas en el super computador antes mencionado. La Tabla 2 muestra un desglose de los parámetros, con su descripción, rango, y el valor encontrado por IRACE.

Tabla 2: Parámetros del algoritmo.

Parámetro	Descripción	Rango	Valor
α	Probabilidad de repetir soluciones ya creadas	[0,1]	0,225
E	Porcentaje de Elitismo	[0,1]	0,342
N^p	Población inicial	[50,75,100,125]	125
$Imax$	Máximo número de iteraciones	[20,40,60,80,100,120,140]	120
Pc	Probabilidad de cruzamiento	[0,1]	0,426
Pm	Probabilidad de mutación	[0,1]	0,939
S	Número de individuos del Torneo (Selección)	[2,3,4,5,6]	2
c_1	Probabilidad que el cruzamiento sea 1PC	[0,1]	0,178
c_2	Probabilidad que el cruzamiento sea CPMX	[0,1]	0,436
c_3	Probabilidad que el cruzamiento sea CPCA1	[0,1]	0,089
c_4	Probabilidad que el cruzamiento sea CPCA2	[0,1]	0,297
m_1	Probabilidad que la mutación sea UI	[0,1]	0,101
m_2	Probabilidad que la mutación sea SFI	[0,1]	0,214
m_3	Probabilidad que la mutación sea LS	[0,1]	0,015
m_4	Probabilidad que la mutación sea MPCA	[0,1]	0,670

4.3. Resultados Computacionales

El AGM es evaluado con las métricas DB, CS y SI, cada instancia es ejecutada 10 veces con semillas diferentes. Los resultados son presentados en la Tabla 3 y la primera columna corresponde al nombre de la instancia, la segunda al criterio de evaluación que indica cada fila y las restantes a los resultados de los índices DB, CS y SI. Además, cada instancia muestra el mejor resultado, el peor, el promedio, la desviación estándar y el tiempo promedio de ejecución con cada métrica. Los resultados muestran que la métrica DB obtiene un tiempo más bajo que el algoritmo ejecutando las otras métricas. Esta métrica promedia 31 segundos aproximadamente, frente a los 120 y 129 segundos que promedia el algoritmo con CS y SI, respectivamente.

Tabla 3: Resultados con las tres métricas.

Data set	Criterio	DB	CS	SI
Aggregation	Mejor	0,5701	0,6337	0,5268
	Peor	0,6021	0,7895	0,5256
	Media	0,5929	0,6615	0,5267
	Desv. Std.	0,0099	0,0438	0,0004
	Tiempo Medio	33,9960	57,7915	117,2091
Breast	Mejor	0,7621	0,5770	0,5964
	Peor	0,7621	1,1020	0,5962
	Media	0,7621	0,9456	0,5962
	Desv. Std.	0,0000	0,1741	0,0001
	Tiempo Medio	28,9435	67,6668	93,0612
Compound	Mejor	0,4654	0,7011	0,6383
	Peor	0,5029	0,7792	0,6383
	Media	0,4907	0,7714	0,6383
	Desv. Std.	0,0149	0,0234	0,0000
	Tiempo Medio	24,9483	30,4736	50,8451
Flame	Mejor	0,6321	0,4477	0,4447
	Peor	0,6914	0,9468	0,4437
	Media	0,6817	0,7563	0,4442
	Desv. Std.	0,0168	0,1661	0,0003
	Tiempo Medio	21,0135	25,8873	24,4842
Glass	Mejor	0,5015	0,3934	0,5919
	Peor	0,6578	0,8565	0,5883
	Media	0,5970	0,6790	0,5892
	Desv. Std.	0,0587	0,1341	0,0011
	Tiempo Medio	26,9021	38,1907	34,2511
Iris	Mejor	0,3828	0,6269	0,6867
	Peor	0,3828	0,6269	0,6867
	Media	0,3828	0,6269	0,6867
	Desv. Std.	0,0000	0,0000	0,0000
	Tiempo Medio	21,3052	21,2653	27,3618
Leaves	Mejor	0,2114	0,1570	0,9328
	Peor	0,5491	0,4178	0,9180
	Media	0,4378	0,2622	0,9248
	Desv. Std.	0,1037	0,0822	0,0060
	Tiempo Medio	46,6282	603,6141	503,1244
Thyroid	Mejor	0,5210	0,5579	0,6318
	Peor	0,7771	0,9328	0,5167
	Media	0,7086	0,7811	0,5700
	Desv. Std.	0,0762	0,1067	0,0299
	Tiempo Medio	29,3531	59,5445	33,5882
Wine	Mejor	0,4289	0,5517	0,6601
	Peor	0,4565	0,7632	0,6574
	Media	0,4494	0,6781	0,6592
	Desv. Std.	0,0110	0,0665	0,0009
	Tiempo Medio	22,6067	31,5357	30,1252
Yeast	Mejor	0,9142	0,4963	0,2674
	Peor	1,2973	1,6192	0,2673
	Media	1,1464	0,8742	0,2673
	Desv. Std.	0,1301	0,3724	0,0000
	Tiempo Medio	46,6186	170,2306	275,3369

4.4. Comparación con el estado del arte

En esta sección, se presenta el desempeño del algoritmo con el estado del arte. Para este, se presentan tres comparaciones, donde cada comparacion muestra el resultado del algoritmo con una métrica especifica, frente a otros algoritmos de la literatura que utilizaron la métrica en cuestión. Además, se destaca con negrita los mejores resultados para cada instancia. La Tabla 4 presenta los resultados del AGM propuesto considerando la métrica DB. Cada columna corresponde a los siguientes algoritmos: Optimización de Malezas Invasoras (IWO) (por sus siglas en inglés, Invasive Weed Optimization) (Ezugwu, 2020), Optimizacion basada en enseñanza-aprendizaje, (TLBO) (por sus siglas en inglés, Teaching-Learning-Based Optimization) (Ezugwu, 2020), Búsqueda de Organismos simbióticos (SOS) (por sus siglas en inglés, Symbiotic Organisms Search) (Ezugwu, 2020), SOSKmeans (Ikotun & Ezugwu, 2022) y Optimizacion de enjambre de partículas - búsqueda de organismos simbióticos, (SOSPSO) (por sus siglas en inglés, Symbiotic Organisms Search Particle Swarm Optimization) (Rajah & Ezugwu, 2020).

En la Tabla 4 se puede destacar que el algoritmo propuesto supera en media y mejor resultado a las instancias Compound, Flame, Iris, Leaves y Wine. Por otro lado, AGM es superado por el algoritmo TLBO en la instancia Agreggation. En Glass, AGM es superado por cuatro de los cinco algoritmos con los que se compara. Además, AGM es superado en las instancias Breast, Thyroid y Yeast por los algoritmos TLBO y SOSPSO. Finalmente, el AGM destaca con el mejor promedio general (0,5389) y el promedio general (0,6245) en comparación a los otros algoritmos.

Se aplica el test de Wilcoxon (Wilcoxon, 1945) a los mejores resultados de los algoritmos con la métrica DB, este test tiene por finalidad indicar si existe una diferencia significativa entre el par de muestras emparejadas. En la Tabla 5 son presentados los resultados del test de Wilcoxon. Donde, en la primera columna se encuentran los algoritmos a comparar, en la segunda la estadística de prueba que proporciona el valor-p de la tercera columna. Sin embargo, los valores-p indican que el AGM solo posee una diferencia significativa con el algoritmo SOS, ya que es el único que logra rechazar la hipótesis nula, que indica que no hay diferencia significativa entre ambas muestras (valor-$p < 0.05$).

Notar que no se realiza una comparación de los tiempos computacionales debido a que sería una comparación injusta, ya que cada algoritmo fue ejecutado en computadores con diferentes características y condiciones de término.

Tabla 4: Comparación con la literatura en la métrica DB.

Data set	Criterio	AGM	IWO	TLBO	SOSK-means	SOS	SOS-PSO
Aggregation	Mejor	0,5701	0,7199	**0,5688**	-	0,6284	-
	Peor	**0,6021**	0,7565	0,6270	-	0,7222	-
	Media	**0,5929**	0,7245	0,6107	-	0,6809	-
	Desv. Std.	0,0099	0,0122	0,0091	-	0,0232	-
Breast	Mejor	0,7621	**0,6519**	**0,6519**	0,8121	0,9151	0,6522
	Peor	0,7621	0,6521	0,6669	0,8121	1,2527	1,1410
	Media	0,7621	**0,6520**	0,6538	0,8121	1,0611	0,7128
	Desv. Std.	0,0000	0,0000	0,0025	0,0000	0,1096	0,1458
Compound	Mejor	**0,4654**	0,4932	0,4932	0,4974	0,5068	-
	Peor	0,5029	**0,4932**	**0,4932**	0,5158	0,5831	-
	Media	**0,4907**	0,4932	0,4932	0,5046	0,5392	-
	Desv. Std.	0,0149	0,0000	0,0000	0,0044	0,0173	-
Flame	Mejor	**0,6321**	0,7732	0,7737	0,7755	0,7325	-
	Peor	**0,6914**	0,7788	0,7788	0,7787	0,7708	-
	Media	**0,6817**	0,7754	0,7740	0,7770	0,7520	-
	Desv. Std.	0,0168	0,0027	0,0010	0,0008	0,0109	-
Glass	Mejor	0,5015	**0,3336**	**0,3336**	**0,3633**	**0,3633**	0,6092
	Peor	0,6578	**0,6092**	0,6105	0,8159	0,8159	0,7268
	Media	**0,5970**	**0,5954**	0,6026	0,7113	0,7113	0,6318
	Desv. Std.	0,0587	0,0608	0,0436	0,1217	0,1217	0,0418
Iris	Mejor	**0,3828**	0,8167	0,5700	0,5937	0,6205	0,5700
	Peor	**0,3828**	1,1598	0,5700	0,6744	0,7820	0,5850
	Media	**0,3828**	0,9755	0,5700	0,6346	0,6854	0,5714
	Desv. Std.	0,0000	0,0747	0,0000	0,0188	0,0330	0,0038
Leaves	Mejor	**0,2114**	1,4059	0,6600	-	1,3210	0,6619
	Peor	**0,5491**	1,6272	1,1371	-	1,4654	1,1274
	Media	**0,4378**	1,5484	0,8139	-	1,3918	0,7598
	Desv. Std.	0,1037	0,0572	0,1191	-	0,0381	0,1235
Thyroid	Mejor	0,5210	0,9001	**0,4814**	0,5754	0,5754	0,**4820**
	Peor	0,7771	1,2864	**0,5897**	0,6934	0,6934	0,6898
	Media	0,7086	1,0558	**0,4968**	0,6321	0,6321	0,5021
	Desv. Std.	0,0762	0,1066	0,0175	0,0316	0,0316	0,0483
Wine	Mejor	**0,4289**	0,4381	0,4382	1,0045	0,9003	0,8017
	Peor	**0,4565**	0,5573	0,8276	1,0896	1,0457	1,0729
	Media	**0,4494**	0,5478	0,6231	1,0460	0,9866	0,8489
	Desv. Std.	0,0110	0,0316	0,1313	0,0207	0,0367	0,0741
Yeast	Mejor	0,9142	1,0660	0,8002	**0,4460**	1,0481	0,6743
	Peor	1,2973	1,4619	**0,9681**	1,0819	1,1538	0,8756
	Media	1,1464	1,2830	0,8330	0,8496	1,1250	**0,7599**
	Desv. Std.	0,1301	0,0728	0,0383	0,1588	0,0243	0,0666
Promedio general	Mejor	**0,5389**	0,7599	0,5771	-	0,7611	-
	Pero	**0,6679**	0,9382	0,7269	-	0,9285	-
	Media	**0,6249**	0,8651	0,6471	-	0,8565	-
	Desv.Std.	0,0421	0,0419	0,0362	-	0,0446	-

Tabla 5: Prueba de Wilcoxon en métrica DB.

Comparación	**Estadística**	**Valor-*p***
AGM vs IWO	10	0,0840
AGM vs TLBO	24	0,7695
AGM vs SOSK-means	11	0,3828
AGM vs SOS	6	0,0273
AGM vs SOS-PSO	9	0,4688

En la Tabla 4 se puede destacar que el algoritmo propuesto supera en media y mejor resultado a las instancias Compound, Flame, Iris, Leaves y Wine. Por otro lado, AGM es superado por el algoritmo TLBO en la instancia Agreggation. En Glass, AGM es superado por cuatro de los cinco algoritmos con los que se compara. Además, AGM es superado en las instancias Breast, Thyroid y Yeast por los algoritmos TLBO y SOSPSO. Finalmente, el AGM destaca con el mejor promedio general (0,5389) y el promedio general (0,6245) en comparación a los otros algoritmos.

Se aplica el test de Wilcoxon (Wilcoxon, 1945) a los mejores resultados de los algoritmos con la métrica DB, este test tiene por finalidad indicar si existe una diferencia significativa entre el par de muestras emparejadas. En la Tabla 5 son presentados los resultados del test de Wilcoxon. Donde, en la primera columna se encuentran los algoritmos a comparar, en la segunda la estadística de prueba que proporciona el valor-*p* de la tercera columna. Sin embargo, los valores-*p* nos indican que el AGM solo posee una diferencia significativa con el algoritmo SOS, ya que es el único que logra rechazar la hipótesis nula, que indica que no hay diferencia significativa entre ambas muestras (valor-$p < 0.05$).

Notar que no se realiza una comparación de los tiempos computacionales debido a que sería una comparación injusta, ya que cada algoritmo fue ejecutado en computadores con diferentes características y condiciones de término.

En la Tabla 6 se presentan los resultados considerando la métrica CS de AGM y cuatro algoritmos de la literatura, IWO, TLBO, SOSK-means y SOS. Esta comparación considera un algoritmo menos, ya que el trabajo que proporciona el algoritmo faltante no utiliza esta métrica. Por otro lado, solo se consideran los mejores algoritmos de la literatura.

Tabla 6: Comparación con la literatura en la métrica CS.

Data set	Criterio	AGM	IWO	TLBO	SOSK-means	SOS
Aggregation	Mejor	0,6337	0,7352	**0,5231**	-	0,6593
	Peor	0,7895	1,1415	**0,6368**	-	0,6821
	Media	0,6615	0,8251	**0,5930**	-	0,6727
	Desv. Std.	0,0438	0,1172	0,0245	-	0,0098
Breast	Mejor	**0,5770**	0,5996	0,6862	0,5996	1,0570
	Peor	1,1020	1,1171	1,0352	**0,9574**	1,1930
	Media	0,9456	0,8055	0,9821	**0,7606**	1,1510
	Desv. Std.	0,1741	0,2171	0,1060	0,1217	0,1060
Compound	Mejor	0,7011	0,7180	**0,5032**	**0,5032**	0,5113
	Peor	0,7792	0,7732	0,7732	**0,5918**	0,6507
	Media	0,7714	0,7673	0,5368	**0,5072**	0,5527
	Desv. Std.	0,0234	0,0173	0,0593	0,0155	0,0576
Flame	Mejor	0,4477	**0,3846**	**0,3846**	**0,3846**	0,4116
	Peor	0,9468	0,7142	0,3948	**0,3846**	0,5803
	Media	0,7563	0,4228	0,3868	**0,3846**	0,5153
	Desv. Std.	0,1661	0,0986	0,0036	0,0000	0,0740
Glass	Mejor	0,3934	**0,0608**	**0,0608**	**0,0608**	**0,0608**
	Peor	0,8565	**0,0608**	**0,0608**	**0,0608**	**0,0608**
	Media	0,6790	**0,0608**	**0,0608**	**0,0608**	**0,0608**
	Desv. Std.	0,1341	0,0000	0,0000	0,0000	0,0000
Iris	Mejor	0,6269	0,6599	**0,5311**	0,5367	0,6037
	Peor	**0,6269**	1,0670	0,7191	0,6444	0,6573
	Media	0,6269	0,8582	**0,5366**	0,5743	0,6321
	Desv. Std.	0,0000	0,1032	0,0297	0,0237	0,0185
Leaves	Mejor	**0,1570**	0,5286	0,4920	-	0,5103
	Peor	**0,4178**	0,9884	0,4920	-	0,5103
	Media	**0,2622**	0,8234	0,4920	-	0,5103
	Desv. Std.	0,0822	0,1078	0,0000	-	0,0000
Thyroid	Mejor	0,5579	0,6409	0,6409	0,6409	**0,4271**
	Peor	0,9328	0,9872	0,6409	0,6409	**0,4423**
	Media	0,7811	0,6572	0,6409	0,6409	**0,4375**
	Desv. Std.	0,1067	0,0634	0,0000	0,0000	0,0175
Wine	Mejor	0,5517	**0,3074**	0,3105	0,6570	0,4061
	Peor	0,7632	**0,4117**	0,4558	0,8829	0,5078
	Media	0,6781	**0,3748**	0,3859	0,8422	0,4648
	Desv. Std.	0,0665	0,0254	0,0398	0,0527	0,0373
Yeast	Mejor	0,4963	0,9337	0,6570	**0,3897**	0,6864
	Peor	1,6192	1,6193	0,8829	**0,6303**	1,0184
	Media	0,8742	1,2314	0,8474	**0,5242**	0,9121
	Desv. Std.	0,3724	0,1900	0,0656	0,0437	0,1082
Promedio general	Mejor	0,5143	0,5569	**0,4789**	-	0,5334
	Peor	0,8834	0,8880	**0,6092**	-	0,6303
	Media	0,7036	0,6827	0,5462	-	**0,5296**
	Desv. Std.	0,1169	0,0940	0,0329	-	0,1043

Tabla 7: Prueba de Wilcoxon en métrica CS.

Comparación	Estadística	Valor-*p*
AGM vs IWO	19	0,4316
AGM vs TLBO	22	0,6250
AGM vs SOSK-means	9	0,2500
AGM vs SOS	27	1

Al analizar los resultados de la Tabla 6 se puede comprender que el AGM no logra obtener un buen rendimiento general, como con la métrica de DB. El AGM solo supera al resto únicamente en la instancia Leaves y solo en promedio para Breast. No obstante, se observa que compite constantemente con los mejores resultados, siendo el peor solo en las instancias Flame y Glass. El AGM obtiene el segundo mejor valor en Agreggation, Thyroid y Yeast. En el promedio general no logra superar al mejor de los algoritmos ni al mejor promedio, pero si logra competir posicionando el mejor resultado obtenido como el segundo mejor entre los algoritmos comparados.

En la Tabla 7 se muestra la prueba de Wilcoxon aplicada a los mejores resultados de los algoritmos con la métrica CS. En la primera columna se muestran los algoritmos a comparar, en la segunda la estadística de la prueba y en la tercera el valor-*p* que proporciona dicha estadística. En este caso, los valores-*p* nos indican que no hay evidencia suficiente para rechazar la hipótesis nula de que ningún par de muestras (valor-p>0.05) es distinto, en otras palabras, no existe diferencia significativa entre los mejores resultados obtenidos por AGM y la literatura.

Además, como el AGM se ve superado por TLBO en varias de las instancias, se ejecuta la prueba de Mann-Whitney U en dirección unilateral (Mann & Whitney, 1947) para probar la hipótesis nula que consiste en que la muestra de los mejores resultados de TLBO no es significativamente mayor que la muestra de los mejores resultados de AGM. En esta prueba se obtiene: una estadística de prueba = 44 y un valor-p = 0.3388 (>0.05). Por tanto, no hay evidencia suficiente para rechazar la hipótesis nula, que en otras palabras significa que TLBO no es significativamente menor (o mejor en este caso) que AGM.

Finalmente, la Tabla 8 muestra los resultados considerando la métrica SI y compara el algoritmo propuesto con otros cuatro algoritmos de la literatura, considerando la media y desviación estándar. Los algoritmos a considerar son los siguientes: algoritmo seno coseno- difuso posibilista medias c-ordenadas (SCA-FPCOM) (por sus siglas en inglés, Sine Cosine Algorithm - Fuzzy Possibilistic C-

Ordered Means)(Kuo et al., 2020), Evolución diferencial multi-objetivo (MODE) (por sus siglas en inglés, Multi-Objective Differential Evolution) (Suresh et al., 2009), Metaheurística hibrida de optimizacion de búsqueda atómica y algoritmo de seno-coseno (ASOSCA) (por sus siglas en inglés Atom Search Optimization y Sine-cosine Algorithm) (Elaziz et al., 2019), Colonia de Hormigas modificada llamada *Anthill* por sus creadores, Pacheco et al. (2018).

Tabla 8: Comparación con la literatura en la métrica SI.

Data set	**Criterio**	**AGM**	**SCA-FPCOM**	**MODE**	**ASOSCA**	**Anthill**
Aggregation	Media	**0,5267**	0,4730	-	-	-
	Desv. Std.	0,0004	0,0000	-	-	-
Compound	Media	**0,6383**	0,4150	-	-	-
	Desv. Std.	0,0000	0,0010	-	-	-
Glass	Media	0,5892	0,4160	**0,7435**	0,3779	-
	Desv. Std.	0,0011	0,0700	0,0114	-	-
Iris	Media	**0,6867**	0,4860	0,6064	0,5601	**0,6870**
	Desv. Std.	0,0000	0,0000	0,0348	-	0,0000
Wine	Media	**0,6592**	0,3010	0,5822	0,5236	0,5390
	Desv. Std.	0,0009	0,0000	0,0043	-	0,0001
Promedio general	Media	**0,6200**	0,4182	-	-	-
	Desv. Std.	0,0005	0,0142	-	-	-

Tabla 9: Prueba de Wilcoxon en métrica SI.

Comparación	**Estadística**	**Valor-*p***
AGM vs SCA-FPCOM	0	0,0625
AGM vs MODE	3	1
AGM vs ASOSCA	0	0,2500
AGM vs ANTHILL	1	1

La Tabla 8 contiene menos instancias debido a la diversidad de éstas que poseen los estudios que utilizan la métrica SI. Se compara solo la media y la desviación estándar debido a que la literatura de esta métrica otorga (y compara) solo con estos criterios. Aun así, se puede destacar que el algoritmo propuesto supera a la literatura en cuatro de las siete instancias, además de igualar el mejor resultado para Iris y superar en promedio general al SCA-FPCOM.

Al igual que para las otras dos métricas, se ejecuta el test de Wilcoxon para analizar si los valores obtenidos son significativamente diferentes (valor-p<0.05), lo cual no se observa para ningún par de muestras. Por lo que no hay evidencia suficiente para rechazar la hipótesis nula de la prueba. No

obstante, los bajos tamaños de las últimas tres muestras de comparaciones pueden afectar el resultado de la prueba.

Para la comparación de esta métrica tampoco se considera los tiempos computacionales debido a que sería una comparación injusta, ya que cada algoritmo fue ejecutado en computadores con diferentes características y condiciones de termino.

Capítulo 5. Conclusiones

Este trabajo de título presenta una metaheurística basada en un AGM para resolver el PCA. El AGM propuesto utiliza cuatro tipos de cruzamientos y cuatro tipos de mutación diferentes, los cuales trabajan con una representación novedosa del PCA. Además, utiliza el algoritmo K-means y soluciones aleatorias en la población inicial. El algoritmo propuesto es validado en instancias de la literatura y considerando tres métricas, DB, CS y SI. El AGM obtiene el mejor desempeño con la métrica DB, logrando superar al estado del arte en la mayoría de las instancias y en el promedio general de estas. Además, en la métrica CS se observa que aun cuando el AGM es superado por otros algoritmos, logra competir constantemente con los mejores resultados. Por otro lado, en la métrica SI, si bien no hay una muy alta cantidad de algoritmos para comparar debido a la gran variabilidad de instancias que existen en los estudios que abordan este índice. Se puede observar una superioridad del algoritmo propuesto por sobre el resto, en donde a pesar de ser superado e igualado en dos instancias, este demuestra que puede competir de manera importante con otros estudios.

Este algoritmo aborda de una forma distinta el PCA, tanto en la representación de la solución como en la forma de abordarlo con un AG con múltiples operadores. Esto podría dar espacio para generar nuevas metaheurísticas, como que utilice un operador de búsqueda local con otros criterios de evaluación, alguna que genere un tipo de memoria histórica para generar descendencia a partir de una población de los mejores resultados obtenidos durante la evolución. Esta sería como una especie de AG híbrido con Tabu Search, o incluso tomar desde la literatura, la idea de hacer una metaheurística totalmente basada en K-means que calcule todas las posibles distribuciones considerando todos los valores de clústeres factibles, luego, tome los mejores y genere cruces únicamente entre ellos. Por otro lado, también se podría entrenar un modelo de machine learning con las mejores agrupaciones generadas en distintas instancias. Con la finalidad de que logre determinar y entender la agrupación de datos, de manera que logre entregar una cantidad de clústeres óptima. Así, el AGM podría resolver el subproblema, solo el PC, y así considerar un k sin el error humano.

Referencias

Abdoun, O., Abouchabaka, J., & Tajani, C. (2012). Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem. International Journal of Emerging Sciences, 2(1), 61-77

Agustín-Blas, L. E., Salcedo-Sanz, S., Jiménez-Fernández, S., Carro-Calvo, L., Kasabov, N., & Portilla-Figueras, J. A. (2012). A new grouping genetic algorithm for clustering problems. Expert Systems With Applications, 39(10), 9695-9703.

Alotaibi, Y. (2022). A new meta-heuristics data clustering algorithm based on Tabu Search and Adaptive Search Memory. Symmetry, 14(3), 623.

Bezdek, J. C. (1974). Numerical taxonomy with fuzzy sets. Journal of Mathematical Biology, 1(1), 57-71.

Chou, C. P., Su, M., & Lai, E. M. (2004). A new cluster validity measure and its application to image compression. Pattern Analysis and Applications, 7, 205–220.

Contreras-Bolton, C., Gatica, G., Barra, C., & Parada, V. (2016). A multi-operator genetic algorithm for the generalized minimum spanning tree problem. Expert Systems With Applications, 50, 1-8.

Davies, D. L., & Bouldin, D. W. (1979). A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2), 224-227.

Deeb, H., Sarangi, A., Mishra, D., & Sarangi, S. K. (2022). Improved Black Hole optimization algorithm for data clustering. Journal of King Saud University - Computer and Information Sciences, 34(8), 5020–5029.

Dhas, P. E., & Gomathi, B. S. (2019). A novel clustering algorithm by clubbing GHFCM and GWO for microarray gene data. The Journal of Supercomputing, 76(8), 5679-5693.

Dunn, J. (1973). A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. Journal of Cybernetics, 3(3), 32-57.

Dutta, D., Dutta, P., & Sil, J. (2012). Data clustering with mixed features by multi objective genetic algorithm. 2012 12th International Conference on Hybrid Intelligent Systems.

Eiben, A. E., & Smith, J. E. (2015). Introduction to evolutionary computing. (2nd ed.). Springer Berlin Heidelberg.

Elaziz, M. A., Nabil, N., Ewees, A. A., & Lu, S. (2019). Automatic data clustering based on hybrid atom search optimization and sine-cosine algorithm. 2019 IEEE Congress on Evolutionary Computation (CEC).

Ezugwu, A. E. (2020). Nature-inspired metaheuristic techniques for automatic clustering: a survey and performance study. SN applied sciences, 2(2).

Ezugwu, A. E., Shukla, A., Agbaje, M. B., Oyelade, O. N., José-García, A., & Agushaka, J. O. (2020). Automatic clustering algorithms: a systematic review and bibliometric analysis of relevant literature. Neural Computing and Applications, 33(11), 6247-6306.

Ghoneimy, M., Hassan, H., & Nabil, E. (2021). A New Hybrid Clustering Method of Binary Differential Evolution and Marine Predators Algorithm for Multi-omics Datasets. International Journal of Intelligent Engineering and Systems, 14, 421-431.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional.

Goldberg, D. E., & Lingle Jr., R. (1985). Alleles, loci and the traveling salesman problem. En Proceedings of the 1st International Conference on Genetic Algorithms, pp. 154-159.

Holland J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology control and artificial intelligence. University of Michigan Press.

Hong, T., Wang, H., & Chen, W. (2000). Simultaneously Applying Multiple Mutation Operators in Genetic Algorithms. Journal of Heuristics, 6(4), 439-455.

Ikotun, A. M., & Ezugwu, A. E. (2022). Boosting k-means clustering with symbiotic organisms search for automatic clustering problems. PLOS ONE, 17(8), e0272861

Jahan, M., & Hasan, M. M. (2021). A robust fuzzy approach for gene expression data clustering. Soft Computing, 25(23), 14583-14596.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. Pattern Recognition Letters, 31(8), 651-666.

Kelly M., Longjohn R., Nottingham K. (2023) The UCI Machine Learning Repository (2023). https://archive.ics.uci.edu, última vez visitado el 07 de agosto de 2023.

Kuo, R., Lin, J., & Nguyen, T. H. O. (2020). An application of sine cosine algorithm-based fuzzy possibilistic c-ordered means algorithm to cluster analysis. Soft Computing, 25(5), 3469-3484.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3:43–58.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, 1, 281-297.

Pacheco, T. M., Goncalves, L. B., Stroele, V., & Soares, S. S. R. F. (2018). An ant colony optimization for automatic data clustering problem. 2018 IEEE Congress on Evolutionary Computation (CEC).

Pakhira, M. K., Bandyopadhyay, S., & Maulik, U. (2004). Validity index for crisp and fuzzy clusters. Pattern Recognition, 37(3), 487-501.

Rajah, V., & Ezugwu, A. E. (2020). Hybrid symbiotic organism search algorithms for automatic data clustering. 2020 Conference on Information Communications Technology and Society (ICTAS).

Rand, W. M. (1971). Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association, 66(336), 846–850.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53-65.

Sinaga, K. P., & Yang, M.-S. (2020). Unsupervised K-means clustering algorithm. IEEE Access. 8, 80716–80727.

Suresh, K., Kundu, D., Ghosh, S., Das, S., & Abraham, A. (2009). Data Clustering Using Multi-objective Differential Evolution Algorithms. Fundamenta Informaticae, 97(4), 381-403.

Umbarkar, A. J., & Sheth, P. D. (2015). Crossover Operators in Genetic Algorithms: A Review. ICTACT journal on soft computing, 06(01), 1083-1092.

Van Rijsbergen, C. J. (1979). Information Retrieval (2a ed.). Butterworth-Heinemann.

Wang, S., & Wu, Y. (2010). Clustering analysis based on Chaos Genetic Algorithm. 2010 Chinese Control and Decision Conference.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. Biometrics Bulletin, 1(6), 80–83.

Xie, X., & Beni, G. (1991). A validity measure for fuzzy clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(8), 841-847.

Zhu, Q., Tang, X., & Elahi, A. (2022). Automatic clustering based on dynamic parameters harmony search optimization algorithm. Pattern Analysis and Applications, 25(4), 693-709.

UNIVERSIDAD DE CONCEPCION – FACULTAD DE INGENIERIA

RESUMEN DE MEMORIA DE TITULO

Departamento: Departamento de Ingeniería Civil Industrial

Carrera: Ingeniería Civil Industrial

Nombre del memorista: Víctor Alfonso Suazo San Martín

Título de la memoria: Una metaheurística para el problema de clustering automático

Fecha de la presentación oral: 07 de septiembre de 2023

Profesor(es) Guía: Carlos Contreras Bolton

Profesor(es) Revisor(es): Carlos Herrera

Concepto:

Calificación:

Resumen

El problema de clustering consiste en subdividir un conjunto de datos en una cantidad determinada de grupos e identificar cuales datos tienen la similitud necesaria para pertenecer a un mismo grupo. Para resolver este problema es necesario saber de ante mano la cantidad exacta de grupos a asignar, esto en la práctica, generalmente, no sucede con exactitud sobre todo con conjuntos grandes de datos. Dada esta problemática surge el problema de clustering automático (PCA), que consiste en determinar automáticamente la cantidad de grupos en que se deben subdividir datos de naturaleza desconocida, además de asignar el grupo idóneo para cada uno de estos. Por tanto, el presente trabajo tiene como objetivo resolver el PCA. Para ello, se propone un algoritmo genético multioperador con cuatro operadores de mutación y cuatro operadores de cruzamiento sobre una estructura novedosa de representar la solución, que comprende la aplicación del módulo de la cantidad de grupos sobre la presolución generada. El algoritmo propuesto es validado en un conjunto de diez instancias de la literatura de distintas dimensiones. Además, se utilizan las métricas del índice Davies-Boulding, índice de separación compacta y el índice de la silueta para medir el rendimiento en comparación al estado del arte. Los experimentos computacionales muestran resultados prometedores, puesto que el algoritmo propuesto es competitivo con algoritmos de la literatura en las instancias consideradas y las tres métricas.