Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores
Unidad Morelia

Estudio de metaheurísticas aplicadas a clustering para su comparación en distintos escenarios

T E S I S

para obtener el título de:

Licenciado en Tecnologías para la información en ciencias

presenta:

Saúl Armas Gamiño

Tutora:

Dra. Adriana Menchaca Méndez

Morelia, Michoacán, 2023

ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores
Unidad Morelia

Estudio de metaheurísticas aplicadas a clustering para su comparación en distintos escenarios

T E S I S

para obtener el título de:

Licenciado en Tecnologías para la información en ciencias

presenta:

Saúl Armas Gamiño

Tutora:

Dra. Adriana Menchaca Méndez

Morelia, Michoacán, 2023

ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

MTRA. IVONNE RAMÍREZ WENCE
DIRECTORA
DIRECCIÓN GENERAL DE ADMINISTRACIÓN ESCOLAR
P R E S E N T E

Por medio de la presente me permito informar a usted que en la **sesión ordinaria 07** del **Comité Académico de la Licenciatura en Tecnologías para la Información en Ciencias** de la Escuela Nacional de Estudios Superiores (ENES), Unidad Morelia, celebrada el día **20 de septiembre de 2023**, se acordó poner a su consideración el siguiente jurado para la presentación del Trabajo Profesional del alumno **Saúl Armas Gamiño** de la Licenciatura en **Tecnologías para la Información en Ciencias,** con número de cuenta **41912498-0,** con el trabajo titulado: **"Estudio de metaheurísticas aplicadas a clustering para su comparación en distintos escenarios",** bajo la dirección como tutora de la **Dra. Adriana Menchaca Méndez.**

El jurado queda integrado de la siguiente manera:

Presidente:	Dr. Víctor Hugo de la Luz Rodríguez
Vocal:	Dra. María del Río Francos
Secretario:	Dra. Adriana Menchaca Méndez
Suplente:	Dr. Daniele Colosi
Suplente:	Dra. Elizabeth Montero Ureta

Sin otro particular, quedo de usted.

A t e n t a m e n t e
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Morelia, Michoacán a 05 de marzo de 2024.

DRA. YUNUEN TAPIA TORRES
SECRETARIA GENERAL

Agradecimientos

Institucionales

Estimados maestros de asignatura que conocí durante la carrera, quiero expresar mi sincero agradecimiento a cada uno de ustedes por la invaluable experiencia que me brindaron a lo largo de mi carrera académica. Cada uno de ustedes impartió sus clases con un elevado nivel de profesionalismo, conocimiento y, lo que es aún más destacable, con una apasionada dedicación a su labor docente.

Asimismo, deseo extender un especial reconocimiento a los distinguidos miembros del jurado que evaluaron mi trabajo de tesis. Aprecio profundamente el tiempo y esfuerzo que dedicaron para brindarme su valiosa orientación y apoyo en la realización de este proyecto. Sé que todos ustedes cuentan con una agenda cargada de responsabilidades, por lo que me complace enormemente que hayan decidido dedicar su tiempo a formar parte de este proceso académico.

No puedo pasar por alto la figura fundamental de mi asesora de tesis, la Doctora Adriana Menchaca Méndez. Fue ella quien me propuso el tema de investigación y me acompañó de principio a fin con su inestimable orientación y sabiduría. Sin su guía experta y compromiso, la culminación exitosa de este trabajo no habría sido posible.

Por último, pero no menos importante, deseo expresar mi gratitud hacia la Universidad Nacional Autónoma de México (UNAM) por su apoyo financiero a través del programa de becas UNAM-DGAPA-PAPIME PE106923. Gracias a este programa, pude llevar a cabo mi proyecto de investigación sin preocupaciones económicas que pudieran limitar su desarrollo.

Personales

Deseo expresar mi más sincero agradecimiento a mi familia y a mi querida pareja por su incansable respaldo a lo largo de este desafiante recorrido académico. Su amor incondicional, comprensión constante y aliento incansable fueron el motor que me impulsó a alcanzar esta meta. Este logro no habría sido posible sin su apoyo inquebrantable.

Resumen

La tarea de *clustering* está presente en todos los aspectos de la sociedad pues nos permite clasificar los elementos que nos rodean. Podemos usar la clasificación de dichos elementos para cosas tan cotidianas como saber en qué pasillo del supermercado buscar un determinado producto, hasta cosas tan complejas como segmentar el mercado de una empresa para que dirijan mejor la publicidad, determinar si un paciente con ciertos síntomas padece una enfermedad, incluso determinar si una nueva enfermedad puede ser letal comparándola con otras similares. Si bien existen métodos clásicos de *clustering*, estos no funcionan bien en todos los casos, pues dependen de las características que poseen los datos a clasificar para dar un buen desempeño. Es aquí donde entran las metaheurísticas, las cuales pueden ofrecer buenos resultados en dichos escenarios.

En este trabajo de tesis se estudiaron tres metaheurísticas aplicadas a tareas de *clustering* y se compararon con dos métodos clásicos: *clustering* aglometarivo y *k-means*. El objetivo es identificar ventajas y desventajas de cada método en distintos conjuntos de entrada. Las metaheurísticas estudiadas fueron recocido simulado, evolución diferencial y optimización por cúmulo de partículas. Para cada metaheurística se realizó el mismo procedimiento: estudiar la metaheurística, buscar trabajos donde realicen *clustering* con dicha metaheurística, seleccionar e implementar uno de los trabajos revisados, usar la implementación en tareas de *clustering* tratando de replicar los resultados del trabajo original, comparar los resultados contra los obtenidos por clustering aglometarivo y k-means.

Los resultados experimentales obtenidos corroboran que las metaheurísticas pueden funcionar mejor que los métodos clásicos estudiados en algunos escenarios. Sin embargo, es importante mencionar que no existe un método definitivo para hacer tareas de *clustering* como lo indica el teorema conocido como **No Free Lunch**, el cual establece que cualquier mejora en el desempeño sobre una clase de problemas se compensa con un desempeño inferior en otra clase. Para garantizar una buena clasificación de nuestros datos primero debemos conocerlos y después buscar un método que ofrezca buen desempeño en escenarios similares.

Abstract

The task of *clustering* is present in all aspects of society, as it allows us to classify the surrounding elements. We can use the classification of these elements for everyday things like knowing which aisle in the supermarket to look for a specific product or for more complex tasks like segmenting a company's market for better advertising targeting, determining if a patient with certain symptoms has a disease, or even determining if a new disease can be lethal by comparing it to others. While there are classical methods of *clustering*, they do not work well in all cases because their performance depends on the characteristics of the data being classified. This is where metaheuristics come into play, as they can offer good results in such scenarios.

In this thesis work, three metaheuristics applied to *clustering* tasks were studied and compared with two classical methods: *agglomerative clustering* and *k-means*. The objective is to identify the advantages and disadvantages of each technique for different input datasets. The studied metaheuristics were simulated annealing, differential evolution, and particle swarm optimization. For each metaheuristic, the same procedure was followed: studying the metaheuristic, searching for works that perform *clustering* with that metaheuristic, selecting and implementing one of the reviewed works, using the implementation in *clustering* tasks to replicate the results of the original work, and comparing the results against those obtained by *agglomerative clustering* and *k-means*.

The experimental results corroborate that metaheuristics can perform better than classical methods in some scenarios. However, it is important to mention that there is no definitive method for performing clustering tasks, as indicated by the theorem known as **No Free Lunch**, which states that any improvement in performance on one class of problems is offset by poorer performance on another. To ensure good data classification, we must first understand them and then search for a method that performs well in similar scenarios.

Índice general

Capítulo 1

Antecedentes y justificación

La tarea de agrupamiento, comúnmente conocida como *clustering*, consiste en separar objetos, individuos, muestras, etc., en grupos que se puedan diferenciar entre sí con base en las características de sus componentes. De esta forma, el *clustering* es una herramienta que puede ayudar al entendimiento de las cosas que nos rodean, pues nos permite clasificarlas y relacionarlas. Desde hace varios años, el ser humano ha realizado tareas de *clustering*. Por ejemplo, la separación de las hierbas medicinales de las venenosas y de los minerales de simples piedras. En la actualidad, el campo de la medicina sigue haciendo uso de este tipo de herramientas, por ejemplo, se estudian procesos celulares en cadenas de ADN [16, 23], se identifican con antelación procesos celulares que pueden causar padecimientos [10, 18, 25], lo cual repercute en una mayor esperanza de recuperación ante las enfermedades. Rui Xu y Donald C Wunsch, en el 2010, presentan varias técnicas sofisticadas de *clustering* para investigaciones científicas en biomedicina [28]. Desde un punto de vista empresarial o comercial, el *clustering* puede ayudar a segmentar el mercado, identificar un perfil de los consumidores y dirigir mejor la publicidad.

El problema al que nos enfrentamos en este trabajo de tesis es cómo hacer la clasificación o agrupamiento de elementos para su estudio. Aunque existen varios métodos clásicos, estos métodos no funcionan correctamente en todos los problemas pues dependen de algunos factores, por ejemplo, de las características del conjunto de datos de entrada y de si se cuenta o no con el número de grupos que se desean formar. Para los casos donde los métodos clásicos no funcionan, se pueden utilizar otras técnicas llamadas metaheurísticas. En particular, en esta tesis es de nuestro interés estudiar el comportamiento de tres metaheurísticas y 2 métodos de clustering que describiremos a continuación.

Métodos clásicos para estudiar agrupamientos

Tres métodos clásicos que se pueden considerar para resolver los problemas de esta tesis son los siguientes:

1. **K-means** es un algoritmo al cual se le debe indicar cuántos grupos debe hacer, posteriormente devuelve una forma de agrupar los datos en la cantidad de grupos solicitados. Utiliza la distancia euclidiana, ver Ec. 1.1, para decidir qué elementos son cercanos entre sí y ponerlos en la misma categoría. Su nombre se debe a que

define los centros de las categorías como la *media* de sus elementos. Funciona bien mientras los datos de entrada tengan grupos bien definidos, esto quiere decir que no haya muchos datos que por sus características puedan ser clasificados en dos o más categorías a la vez.

$$D(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{1.1}$$

Siendo k el número de componentes de los datos, x_i y y_i la i-ésima componente de x y y respectivamente.

2. **K-medians** es una variante de *k-means*, la diferencia está en que utiliza la distancia Manhattan, ver Ec. 1.2, para decidir qué tan parecidos son los elementos de cada grupo, además de que los centros de las categorías son la *media* de sus elementos.

$$D(x,y) = \sum_{i=1}^{k}|x_i - y_i| \tag{1.2}$$

Siendo k el número de componentes de los datos, x_i y y_i la i-ésima componente de x y y respectivamente.

3. **Hierarchical clustering**. El *clustering* jerárquico como su nombre lo indica, trata de establecer jerarquías entre los grupos que genera a partir de los datos, sus resultados generalmente se representan con dendrogramas (también conocidos como diagrama de árbol), ver Figura 1.1. Existen dos variantes de *clustering* jerárquico: aglomerativo y divisivo. El **aglomerativo** trabaja formando parejas con los elementos más parecidos entre sí, dicha pareja se vuelve un elemento y se repite el proceso hasta tener un solo elemento, mientras que el **divisivo** comienza considerando todos los elementos en un conjunto y lo divide en dos, cada mitad es considerada un grupo, después vuelve a dividir cada agrupación hasta terminar con la misma cantidad de grupos como de elementos individuales, o cuando cumpla algún criterio de paro dado por el usuario. La distancia euclidiana ha sido utilizada comúnmente para realizar *clustering* jerárquico. Este tipo de *clustering* falla cuando los grupos tienen estructuras complejas y no es un método fácilmente escalable por su complejidad computacional.

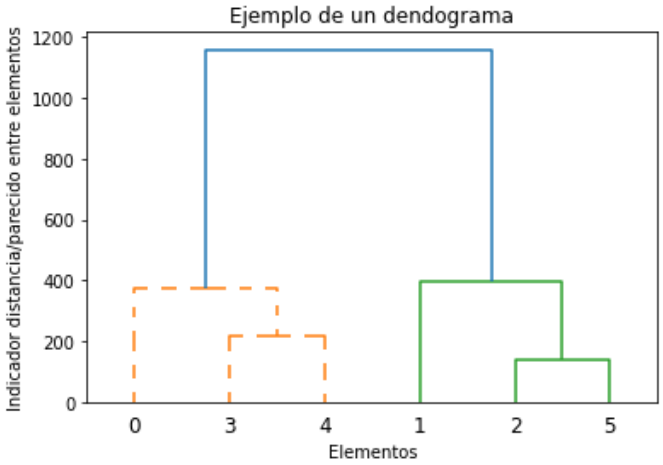


Figura 1.1: Ejemplo de un dendrograma. En este caso en el eje horizontal se encuentran los elementos a agrupar, y en el vertical se mide la distancia entre ellos.

La figura 1.1 se construye en etapas uniendo los dos elementos más similares en uno, cuyo valor será el promedio de sus componentes, hasta agrupar todos los elementos en uno. Si contamos las líneas horizontales de la figura, sabremos cuantas etapas tuvo que hacer para formarse, en este caso cinco, y fueron las siguientes: (1) unir 2 y 5 (2) unir 3 y 4 (3) unir 0 con la media de 3 y 4 (4) unir 1 al grupo de 2 con 5 (5) unir los elementos de línea punteada con los de línea continua.

Existen otro tipo de métodos como los basados en densidad, presentados por Martin Ester (1996) [9] y Mihael Ankerts (1999) [3], los cuales funcionan bien en estructuras complejas siempre y cuando la separación entre un grupo y otro no sea demasiado pequeña. Cuando los límites entre los grupos son ambiguos se recomienda utilizar métodos difusos, propuestos por Bezdek (1981) [4], Krishnapuram (1993) [14] y Dembele (2023) [8]. Otra desventaja que tienen los métodos clásicos en general es que pueden quedar atrapados fácilmente en óptimos locales lo cual impide encontrar el óptimo global, además pueden requerir información extra como el número de grupos que hay en los datos cuando esto último muchas veces no se conoce.

Metaheurísticas para analizar agrupamientos

Una metaheurística se puede ver como un algoritmo de propósito general que permite encontrar buenas soluciones a un problema con un costo computacional razonable. Si bien las metaheurísticas no resolverán todos los problemas de clasificación que puedan existir, si tendrán un mejor desempeño respecto a los métodos clásicos en conjuntos de datos con ciertas características. En esta tesis, estamos interesados en estudiar el comportamiento de tres metaheurísticas:

1. **Recocido simulado** [17]: Como su nombre lo indica, se basa en el proceso de recocido, el cual consiste en llevar a altas temperaturas los metales, para después dejarlos enfriar lentamente. El algoritmo comienza comienza con una solución y después, en cada iteración, genera un vecindario de soluciones cercanas a la actual. Tras generar

las soluciones cercanas, evaluará si cambia su solución actual por una vecina y hará su siguiente iteración. Entonces, el algoritmo se "moverá" entre soluciones hasta llegar a la mejor o al cumplir el máximo de iteraciones establecido. La temperatura en este caso juega un papel fundamental en el algoritmo, pues a mayor temperatura se permite que, con mayor frecuencia, se mueva a otra solución, aunque no sea mejor a la actual. Moverse de una solución a otra ocurre entonces en 2 escenarios: (1) se mueve a una que mejora la actual y (2) la temperatura permite el cambio, aunque no sea favorable.

2. **Evolución diferencial** [13]: Al ser un algoritmo evolutivo intenta imitar el proceso de evolución de los seres vivos, el cual, según el paradigma neo darwinista, está sustentado por cuatro procesos primordiales: reproducción, mutación, competencia y selección. Cada solución del problema se considera un individuo, el cual busca reproducirse y crear descendencia, pero para ello debe competir con los otros, pues solo los más aptos logran reproducirse. Una vez que los individuos logran reproducirse, puede ocurrir una mutación que beneficie o perjudique su descendencia, finalmente la selección se encarga de que solo sobrevivan los más aptos, y pasen a ser los padres de la nueva generación. El algoritmo comienza con una población de individuos aleatorios, que con el paso de las generaciones irá "evolucionando" hasta que el individuo más apto no haya sido desplazado por otro durante una cantidad determinada de generaciones, o al llegar a un número máximo de generaciones determinado por el usuario.

3. **Optimización por cúmulo de partículas** [11]: Es un algoritmo poblacional que se basa en la interacción de las aves dentro de una parvada, busca replicar la forma en que se comparten información entre ellas y permite determinar cuándo es momento de frenar, aterrizar, comer etc. A los individuos se les considera partículas, cada una de ellas es una posible solución al problema que se busca resolver, la forma en que opera es "moviendo" dichas partículas por el espacio de las soluciones al problema hasta encontrar y "aterrizar" en la mejor. Al inicio se generan las partículas en lugares aleatorios dentro del espacio de soluciones. Como se busca una posición óptima para las partículas la comunicación entre ellas es vital, todas saben su posición en el espacio de soluciones y se comparte con las demás, se determina cual partícula está en una mejor zona y las otras se acercan a ella. Durante el camino alguna otra partícula puede encontrar un lugar mejor y se repetiría el proceso hasta que "aterricen" en la mejor zona.

Pruebas que permiten evaluar los métodos de clustering

Para evaluar a los métodos clásicos y las metaheurísticas en tareas de *clustering*, necesitamos conjuntos de datos para que los clasifiquen, y posteriormente calificar dicha clasificación. En este trabajo utilizaremos cuatro conjuntos: Iris, Vino, Vidrio y Cáncer, los cuales se describen con más detalle la sección 2.4.

1.1. Planteamiento del problema

Las técnicas de *clustering* clásicas no pueden ser aplicadas exitosamente en cualquier conjunto de datos, esto motiva hacer uso de diferentes metaheurísticas, las cuales pueden ser aplicadas a una amplia variedad de problemas ya que su diseño está basado en un problema de optimización general. Sin embargo, es bien sabido que un algoritmo no puede resolver eficientemente cualquier tipo de problema de optimización ("no free lunch theorem [27] "). Por lo anterior, es importante identificar las ventajas y desventajas de ciertas metaheurísticas para tener claro en qué tipo de problemas es conveniente utilizarlas. En este trabajo se propone la implementación de al menos tres metaheurísticas para realizar tareas de *clustering*. Dichas implementaciones se utilizarán para realizar un estudio comparativo que nos permita identificar algunas ventajas y desventajas de estos métodos con respecto a las técnicas clásicas, considerando que la principal limitante de dichas técnicas es estar pensadas para problemas más generales, mientras que en problemas específicos pueden no funcionar bien.

1.2. Objetivos del trabajo

Como objetivo general, buscamos estudiar al menos tres metaheurísticas utilizadas en tareas de *clustering* e identificar algunas de sus ventajas y desventajas con respecto a dos técnicas clásicas. En particular, nos proponemos los siguientes objetivos específicos:

1. Estudiar dos métodos clásicos para realizar tareas de *clustering*.
2. Estudiar, comprender e implementar tres metaheurísticas utilizadas en tareas de *clustering*.
3. Realizar un estudio experimental que permita evaluar las metaheurísticas y los métodos clásicos estudiados en los puntos anteriores.
4. Identificar las ventajas y desventajas de cada método considerado en el estudio experimental del punto anterior.

1.3. Metodología

Para lograr el objetivo del trabajo de identificar ventajas y desventajas de las metaheurísticas, fue necesario seguir una seria de pasos, los cuales están enumerados a continuación:

1. Estudiar y comprender dos técnicas clásicas para realizar tareas de *clustering*.
2. Obtener de la librería Sklearn [20] implementaciones de las técnicas estudiadas en el punto anterior.
3. Estudiar al menos tres metaheurísticas utilizadas en tareas de clustering: (1) recocido simulado, (2) evolución diferencial y (3) optimización por cúmulo de partículas. Para cada metaheurística se realizará lo siguiente:

   a) Estudio de la metaheurística.
   b) Búsqueda de propuestas en las que se aplique la metaheurística en alguna tarea de *clustering*.
   c) Selección de una de las propuestas estudiadas.
   d) Implementación de la propuesta seleccionada en el punto anterior.
   e) Recabar conjuntos de datos que permitan evaluar la metaheurística implementada.

4. Realizar un estudio comparativo de las metaheurísticas implementadas en el punto anterior y de los métodos clásicos estudiados en el primer punto.
   a) Elegir al menos tres conjuntos de datos con los que se evaluarán los métodos estudiados.
   b) Elegir al menos dos métricas que se utilizarán para medir el desempeño de las metaheurísticas implementadas.
   c) Realizar un estudio estadístico para conocer el comportamiento de los métodos implementados.
5. Identificar las ventajas y desventajas de los métodos estudiados, para reconocer los escenarios donde le irá bien y donde no, además de conocer su velocidad de ejecución, costo computacional, entre otras.
6. Escritura de la tesis.

1.4. Estructura del documento

En el capitulo 1 hacemos una introducción del clustering, se mencionan algunos métodos para llevarlo a cabo y su importancia en la vida cotidiana, además se describen brevemente las metaheurísticas que se usaran en este trabajo de tesis. Describimos los conjuntos de datos usados en los experimentos, los objetivos de la tesis y la metodología a seguir para lograr dichos objetivos. El capítulo 2 describe conceptos importantes que se usarán a lo largo del proyecto. En los capítulos 3, 4 y 5 abordamos las metaheurísticas **evolución diferencial**, **optimización por cumulo de partículas** y **recocido simulado** respectivamente. En cada capítulo explicamos cómo funciona la metaheurística, replicamos una implementación donde la usan para realizar clustering, tratamos de replicar los resultados de los autores y comparamos los resultados contra las técnicas clásicas **K-means** y **clustering jerárquico**. El capitulo 6 contiene una comparativa de los resultados que ofrecen las 3 metaheurísticas para analizar las ventajas y desventajas que pueden tener en los conjuntos de datos utilizados. Finalmente, el capítulo 7 trata de cómo se podría continuar trabajando el clustering con metaheurísticas.

Capítulo 2

Conceptos Básicos

2.1. ¿Qué es un problema de optimización?

Un problema de optimización, de acuerdo con Andrés Ramos y col. [21] puede verse como "la elección de una solución mejor en algún sentido a las demás soluciones posibles". Para poder definir un problema de optimización son necesarios 3 elementos, los cuales son:

1. Función objetivo
2. Variables
3. Restricciones

2.1.1. Función objetivo

Nos permite evaluar qué tan buena es una solución con respecto al problema que se quiere resolver y el objetivo es encontrar una solución que minimice o maximice el valor de dicha función. Por ejemplo, si se quieren bajar los costos de producción de algún producto, necesitamos encontrar una solución que minimice el valor de la función objetivo. Por el contrario, si se buscara la combinación de productos que debe vender un negocio para aumentar sus ganancias, se tendría que encontrar una solución que maximice el valor de la función objetivo.

2.1.2. Variables de decisión

Cada variable es una componente del problema y todas ellas forman una solución a dicho problema. Por ejemplo, si se busca una combinación de colores para formar otro, cada variable correspondería a la cantidad necesaria de un color para formar el deseado.

2.1.3. Restricciones

Son condiciones a las cuales está limitado nuestro problema. Imaginemos que quisiéramos buscar la forma de minimizar el gasto de nuestra despensa, para lo cual tenemos una lista de productos y un presupuesto. Una restricción en este caso sería que debemos

comprar al menos 1 unidad de cada producto, la otra restricción posible es que no podemos gastar más de nuestro presupuesto.

2.1.4. Definición formal de un problema de optimización

Definición 1 (Problema de optimización). *Buscar la solución* $\vec{x}^* = [x_1, x_2, \ldots, x_n]$ *(vector con los valores de las variables de decisión) que minimice o maximice una función* $f(\vec{x})$ *que está sujeta a una o varias restricciones del tipo* $g(\vec{x}) \leq 0$ *o* $h(\vec{x}) = 0$.

Dada la definición anterior, tenemos que:

- Una **solución factible**: aquella que cumple con todas las restricciones que impone el problema.
- Se considera **región factible**: la zona que delimita las soluciones factibles del problema.
- Un **óptimo local**: aquella solución que sea la mejor en una región determinada de la zona factible.
- El **óptimo global**: la mejor solución que hay en la región factible de soluciones.

La Figura 2.1 muestra gráficamente un ejemplo del comportamiento de las distintas soluciones que puede tener un problema de optimización.

Consideremos que: (1) nuestro problema es de 2 variables: X y Y, (2) las **restricciones** son: $X > -5$ y $X \leq 12$, (3) la función tiene la forma mostrada en la imagen y (4) se busca minimizar el valor de la función. La zona amarilla denota la región factible. Los puntos rojos son soluciones no factibles y los puntos azul claro son soluciones factibles. El punto azul fuerte es un óptimo global y el punto verde es un óptimo local.

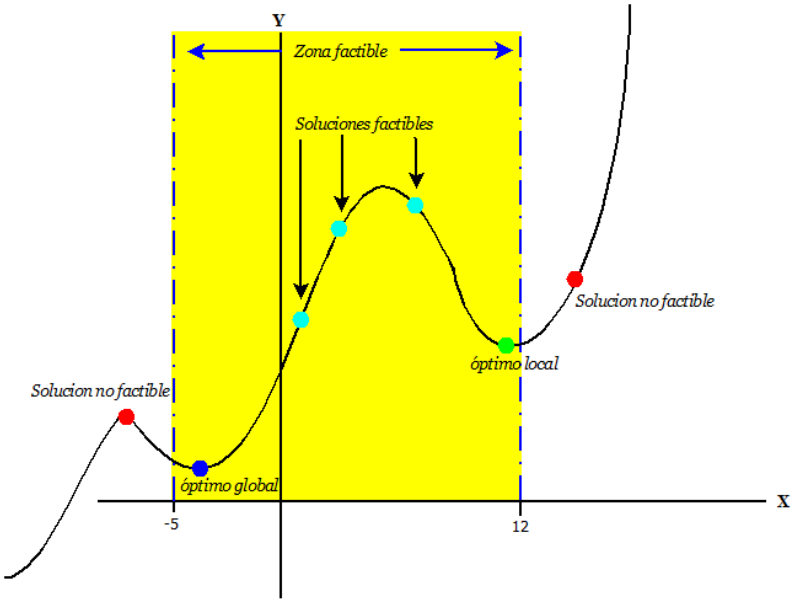


Figura 2.1: Comportamiento de las distintas soluciones que puede tener un problema de optimización como el del Ejemplo.

2.2. ¿Qué es clustering?

Como se describe al inicio de este documento, la tarea de agrupamiento, comúnmente conocida como *clustering*, consiste en separar objetos, individuos, muestras, etc., en grupos que se puedan diferenciar entre sí con base en las características de sus componentes. De esta forma, el *clustering* es una herramienta que puede ayudar al entendimiento de las cosas que nos rodean, pues nos permite clasificarlas y relacionarlas. Para evaluar si se llevó a cabo un buen etiquetado de los elementos, se utilizan herramientas como **índices de validación** y **métricas**. En esta sección primero haremos mención de la función objetivo mas usada por lo métodos clásicos para hacer tareas de *clustering*, después describiremos los **índices de validación** que usaremos: Davies-Bouldin y la medida CS, finalmente explicamos las **métricas** completitud e índice de aleatoriedad ajustado.

Distancia euclidiana al cuadrado

Como mencionan Shokri Z Selim y K1 Alsultan en [24], la función objetivo comúnmente utilizada por el algoritmo k-means y sus variantes para resolver problemas de clustering es la siguiente:

$$J(W, Z) = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} d_{ij}^2 \tag{2.1}$$

Siendo W una matriz que indica a cuál centroide pertenece cada punto y Z la matriz de distancias de cada punto a cada centroide. La variable $w_{ij} \in W$, sirve para asegurarse que solo se tome en cuenta la distancia de un punto a su centroide asignado y no a todos. El valor $d_{ij} \in Z$, es la distancia euclidiana entre el punto i al centroide j. La n es la cantidad de puntos a clasificar, y k el número de grupos que se formarán. La ecuación 2.1 está sujeta a que los pesos w_{ij} suman 1:

$$\sum_{j=1}^{k} w_{ij} = 1, 1 \leq i \leq n$$

$$\text{donde } w_{ij} = \begin{cases} 1 \text{ si el punto i esta asignado al cluster j} \\ 0 \text{ en caso contrario} \end{cases}$$
$$1 \leq i \leq n, 1 \leq j \leq k$$

Si consideramos un conjunto G_j como aquel que contiene a los elementos del grupo j, entonces es posible escribir la ecuación 2.1de la siguiente manera:

$$J(W, Z) = \sum_{j=1}^{k} \sum_{\forall i \in G_j} d_{ij}^2 \tag{2.2}$$

Como podemos observar, en este caso no es necesario el parámetro w_{ij} pues haciendo uso de G_j solo se calcula la distancia de cada elemento al centroide su grupo, y no la distancia de cada punto a cada centroide de todos los grupos.

También es común que los algoritmos de *clustering* usen como función a optimizar la media de la ecuación 2.2, quedando la fórmula de la siguiente manera:

$$J(W, Z) = \frac{\sum_{j=1}^{k} \sum_{\forall i \in G_j} d_{ij}^2}{n} \tag{2.3}$$

Siendo n el número de puntos o elementos a clasificar.

2.2.1. Índices de validación IV

Los índices de validación evalúan la calidad de una partición basándose principalmente en 2 aspectos:

- Cohesión: una mayor puntuación en éste índice se da cuando los elementos que pertenecen a un mismo grupo son muy similares entre sí, lo que se traduce en que la distancia entre ellos sea pequeña.
- Separación: mide la distancia que hay entre los grupos, entre más distancia haya es mejor, ya que se traduce en grupos mejor definidos al no haber grupos muy similares que quizá pudieran convertirse en uno solo.

En general los índices de validación deben cumplir con 2 propósitos: (1) encontrar el número apropiado de grupos que dividen los datos, y (2) repartirlos en la mejor forma en dichos grupos. Algunos ejemplos de los índices más conocidos son: Daivies-Bouldin (DB) [7] y la medida CS [5], las cuales usaremos en este trabajo.

Índice de validación Davies-Bouldin

Tal como indica la documentación `scikit-learn` [20], este índice mide la similitud entre cada cluster C_i con su cluster más similar C_j, para lo cual se define su similitud como R_{ij}, la cual se compone de los valores s_i y d_{ij} descritos a continuación:

- s_i: El diámetro del cluster, que se obtiene calculando el promedio de las distancias entre cada punto y su centroide.
- d_{ij}: La distancia entre los centroides de i y j

Donde R_{ij} se define como:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Y por consiguiente el índice D-B de la siguiente forma:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij} \tag{2.4}$$

Siendo k el número de clusters C que se tienen.

Medida CS

La medida CS fue propuesta en 2024 por C-H Chou, M-C Su y Eugene Lai [5] y surge de la necesidad de detectar grupos de diferentes tamaños y densidades, por lo cual, esta medida se basa en la forma de los grupos, y busca dos cosas en los grupos generados: que sean compactos y que estén bien separados. Un grupo es compacto cuando la distancia entre sus elementos es mínima, y se considera que los grupos están bien separados cuando la distancia entre ellos es la máxima posible. Lo primero que se debe hacer es recalcular los centroides de los grupos de la siguiente forma:

$$\vec{v_i} = \frac{1}{|A_i|} \sum_{X_j \vec{\in} A_i} \vec{X_j}$$

Siendo A_i el conjunto de elementos asignados al i ésimo grupo, $\vec{v_i}$ el centroide de dicho grupo, y $|A_i|$ la cantidad de elementos que tiene A_i. Se define la medida CS de la siguiente forma:

$$CS(c) = \frac{\sum_{i=1}^{c} \left\{ \frac{1}{|A_i|} \sum_{\vec{X_j} \in A_i} \underset{\vec{X_k} \in A_i}{\text{máx}} \{d(\vec{X_j}, \vec{X_k})\} \right\}}{\sum_{i=1}^{c} \left\{ \underset{j \in c, j \neq i}{\text{mín}} d(\vec{v_i}, \vec{v_j}) \right\}} \tag{2.5}$$

Donde d es una función de distancia.

2.2.2. Métricas para medir el desempeño

Al igual que los índices de validación, las métricas nos permiten evaluar la calidad de una partición. A continuación se dan dos ejemplos de métricas: completitud e índice de aleatoriedad ajustado.

Completitud

Con esta métrica se evalúa que todos los datos del mismo grupo hayan sido colocados en la misma clase o categoría. Es decir, se busca que no "falten" datos del grupo en la categoría asignada. Toma valor en el intervalo (0,1), una mayor puntuación indica que es mejor el etiquetado. La documentación de `scikit-learn` [20] define la completitud de la siguiente forma:

$$completitud = 1 - \frac{H(K|C)}{H(K)}$$

donde K es el conjunto de grupos de la partición, C es el conjunto de las clases reales, $H(K|C)$ se refiere a la entropía condicional de C dada la asignación K, y está dado por:

$$H(K|C) = -\sum_{k=1}^{|K|} \sum_{c=1}^{|C|} \frac{n_{c,k}}{n} \cdot \log\left(\frac{n_{c,k}}{n}\right)$$

$H(K)$ es la entropía de los grupos y se define como:

$$H(K) = -\sum_{k=1}^{|K|} \frac{n_k}{n} \cdot \log\left(\frac{n_k}{n}\right)$$

donde n es el número de datos que se están clasificando, n_c y n_k son la cantidad de elementos pertenecientes a la clase c y el grupo k, respectivamente. Finalmente, $n_{c,k}$ representa el número de elementos de la clase c que fueron asignados al grupo k.

Índice de aleatoriedad ajustado

Mide la similitud que hay entre 2 agrupamientos, basándose en el número de datos que fueron agrupados de la misma forma en las observaciones y predicciones. Toma un valor de 0 cuando las predicciones parecen ser aleatorias. Si las predicciones son exactamente igual a las observaciones toma el valor de 1. Para el caso donde son completamente diferentes da el valor de -0.5. En la documentación `scikit-learn` [20] se define el índice aleatorio de la siguiente forma.
Consideremos que C es la clasificación real de los datos y K es el agrupamiento a evaluar. Definimos a y b de la siguiente forma:

- a el número de parejas de elementos clasificados en el mismo grupo tanto en C como en K.
- b el número de parejas de elementos clasificados en distinto grupo en C y en K.

Definido lo anterior, el índice aleatorio está dado por:

$$RI = \frac{a+b}{C_2^{N_{samples}}}$$

Siendo $C_2^{N_{samples}}$ el número de posibles parejas en el conjunto de datos.

La puntuación de RI no está acotada a un rango de valores y depende de la cantidad de elementos en el conjunto de datos, para estandarizarla se agrega el factor de la puntuación esperada para un agrupamiento aleatorio $E(RI)$ y se define el índice aleatorio ajustado de la siguiente forma:

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)}$$

2.3. Clustering como un problema de optimización

La tarea de *clustering* puede verse como un problema de optimización, para ello vamos a definir los 3 componentes del problema de optimización como sigue:

- Función objetivo. El **índice de validación** puede ser convertido en la función objetivo del problema, pues es la forma en que se mide la calidad de un agrupamiento propuesto: a mayor puntuación obtenida en el índice mejor es la solución propuesta. Si el problema de optimización busca maximizar no sería necesario modificar los

índices de validación, si se busca minimizar se debe modificar para que nos dé una menor puntuación con soluciones mejores, esto puede ser regresando el recíproco de la puntuación.

- Variables. Recordemos que una solución al problema de optimización se ve de la forma $\vec{x} = [x_1, x_2, \ldots, x_n]$ siendo n el número de variables del problema. En este caso podemos tener dos representaciones diferentes. La primera considera n como el número de elementos en el conjunto de datos y cada variable x_i corresponde a la etiqueta del elemento i. La segunda considera $n = d \cdot K$, donde d es el número de atributos que tienen los datos del conjunto y K el número de grupos; cada variable x_i sería el valor de uno de los atributos que define un centroide.

- Restricciones. En la primera representación, las restricciones que tenemos son: (1) que todo elemento tenga una etiqueta y (2) que dicha etiqueta pertenezca a uno de los grupos en los que se dividen. Lo que se vería de la forma: $x_i = k_j$ con $x_i \in \bar{X}$ y $k_j \in K$, con K como el conjunto de etiquetas pertenecientes a los grupos en que se dividirán los elementos. Para la segunda representación, la restricción sería que los valores de los atributos de cada centroide estén dentro del rango de valores de los atributos.

2.4. Conjuntos de datos

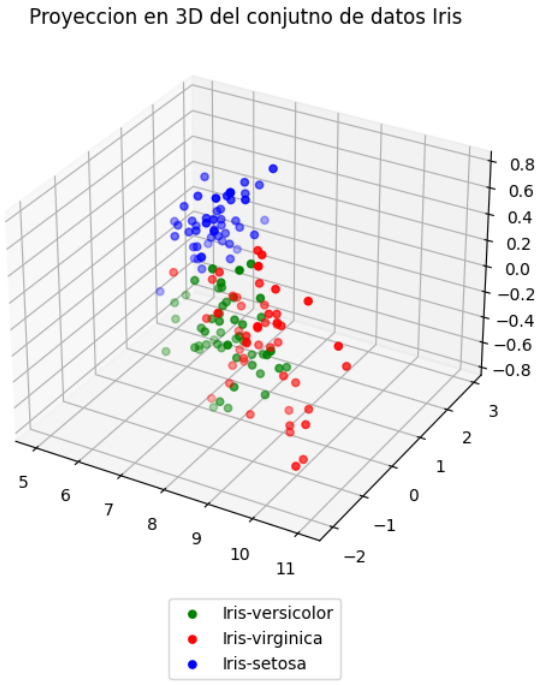
Existen varios conjuntos de datos disponibles que permiten evaluar el desempeño de los algoritmos de *clustering*. En este trabajo de tesis se utilizaron los siguientes:

- **Iris**: tiene 3 grupos, cada uno con 50 registros de flores diferentes haciendo un total de 150 registros. La dimensión de cada registro es 4 y los grupos tienen un pequeño traslape entre ellos.

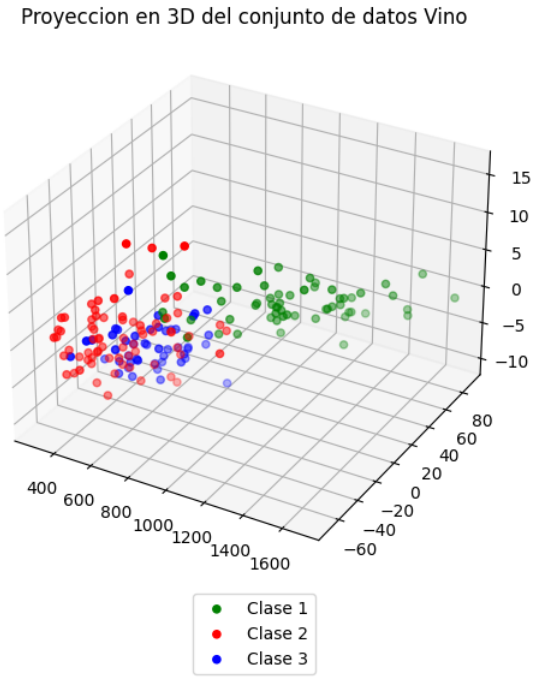
- **Vino**: tiene 3 grupos, cada registro cuenta con 13 atributos, no hay datos faltantes y en total son 178 registros. Según los autores el conjunto de datos tiene un "buen comportamiento".

- **Vidrio**: se tienen 7 categorías, aunque en la descripción del conjunto de datos indican que no hay ningún registro de una de ellas, por lo cual se estaría trabajando con 6. Tiene un total de 214 registros , ninguno tiene atributos faltantes. Cada registro cuenta con 11 atributos, de los cuales quitamos el id y la clase para quedarnos con 9 para clasificarlos.

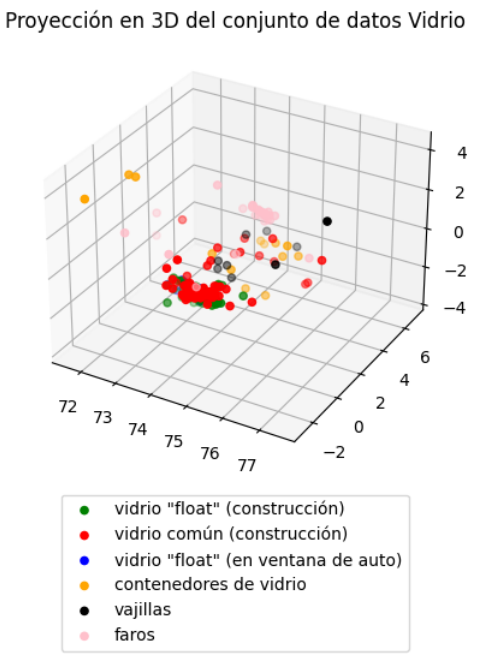
- **Cáncer**: este conjunto cuenta con 2 grupos, ya que sus registros indican si hay un tumor de tipo maligno o benigno. Cada registro cuenta con 10 atributos, pero uno es el id del registro, así que se tienen 9 atributos. En total hay 699 registros, pero hay 16 con datos faltantes, por lo que en este trabajo de tesis usaremos solo 683, de los cuales 444 son de tipo benigno y 239 de tipo maligno. Los grupos tienen un poco de traslape entre ellos.



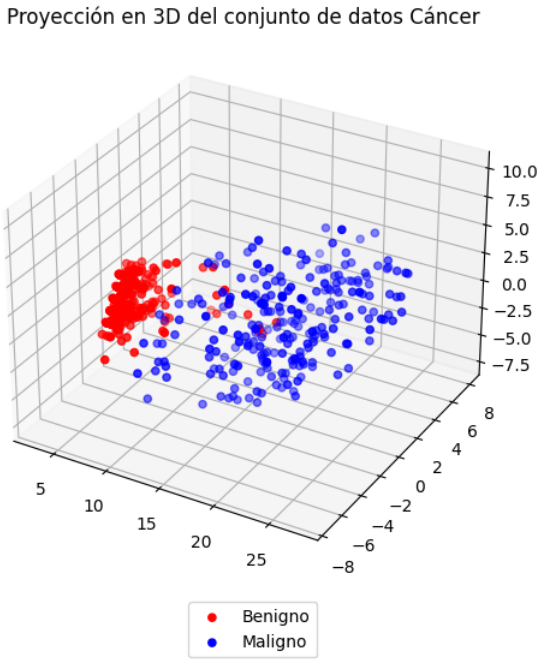
(a) Iris



(b) Vino



(c) Vidrio



(d) Cáncer

Figura 2.2: Representación en 3d de los conjuntos de datos Iris, Vino, Vidrio y Cáncer. La gráfica muestra una reducción de dimensiones de cada conjunto de datos, hechas con la librería **SVD** (descomposición en valores singulares) tomada de Sklearn [20].

Capítulo 3

Evolución diferencial

El algoritmo de Evolución Diferencial (ED) fue propuesto por Storn y Price en 1997 [26] y es un Algoritmo Evolutivo (AE) utilizado, generalmente, para resolver problemas de optimización en los que las variables de decisión son números reales. Un AE intenta imitar el proceso de evolución de los seres vivos, el cual, según el paradigma neodarwinista, está sustentado por cuatro procesos primordiales: reproducción, mutación, competencia y selección.

Al igual que en todos los AE, ED realiza las siguientes operaciones sobre los individuos: cruza, mutación, asignación de aptitud y selección. La **cruza** es cuando a partir de 2 o más individuos "padres", se genera uno nuevo llamado "hijo", haciendo una combinación de los padres. Así, cada componente del hijo será el de alguno de sus padres, o bien, uno generado a partir de los valores de los padres. Por ejemplo, si tenemos a los padres $[0, 1, 2]$ y $[0, 2, 3]$, los posibles hijos, considerando combinaciones, son: $[0, 1, 2]$ $[0, 1, 3]$ $[0, 2, 2]$ $[0, 2, 3]$. El proceso de **mutación** consiste en modificar alguna componente del individuo, lo más común es escalar dicho valor por una constante comúnmente llamada "tamaño de paso" pues determina cuánto cambia dicha componente. Al llegar a la evaluación, los individuos se usan como entrada a una función, la cual dará como resultado un valor que indica la **aptitud** del individuo, a mayor aptitud, el individuo es considerado una mejor respuesta al problema que se enfrenta. La función de aptitud está siempre relacionada con la función objetivo que se está optimizando. Finalmente, la **selección** consiste generalmente en reemplazar un individuo con una baja "aptitud" por otro con una mayor. En ocasiones puede ser que haya un criterio distinto para la selección, como escoger a un individuo sobre otro por probabilidad pero siempre garantizando conservar al menos al mejor individuo de la población actual.

La forma en que trabaja un AE es iterando cierto número de **generaciones**. Al principio se genera una "población inicial", la cual está compuesta por I individuos, cada uno con la dimensión del problema a resolver, y cada parámetro del individuo generado al azar dentro del rango de valores del problema. Por ejemplo, si nuestro problema está en R^3, y la solución solo puede tomar valores en $[-1, 1]$, una población inicial de 3 individuos ($I = 3$) puede ser: $[[0.2, 0.1, -0.3], [-0.2, 0.6, 0.9], [0.5, -0.5, 0.4]]$. Posteriormente, en cada generación se repite el siguiente proceso:

A cada miembro de dicha población se le evalúa para obtener su **aptitud**. Después viene el momento de la cruza donde se generarán los individuos **hijo**, para esto se eligen

individuos de la población actual que actuarán como padres, muchas veces se asignan probabilidades dependiendo su aptitud pero hay ocasiones en las que todos tienen la misma probabilidad de ser padres. Una vez generados los individuos hijos, se les aplica el operador de mutación. Finalmente, se seleccionan a los individuos que pasarán a la siguiente generación.

3.1. Algoritmo

Sea D el número de variables de decisión del problema. En la versión original de ED cada individuo es un vector de números flotantes de tamaño D, donde cada posición pertenece a una de las variables:

$$X = x_0, x_1, ..., x_D$$

El objetivo de ED es encontrar un individuo. El cual, al ser evaluado en la función objetivo, nos deberá arrojar el valor mínimo de la función, o al menos un valor mínimo local. Los parámetros requeridos por ED son: tamaño de la población N, dimensión de los individuos D, constante de escalamiento F, numero de generaciones G y probabilidad de cruza CR. En ED, el operador de mutación puede ser considerado como uno de los operadores principales y existen varias versiones. La propuesta original de Storn y Price en 1997 [26] lo define con la fórmula 3.1.

$$V = X_{r_1} + F \cdot (X_{r_2} - X_{r_3}) \tag{3.1}$$

donde r_1, r_2 y r_3 son índices de individuos de la población actual escogidos al azar, tal que $r_1 \neq r_2 \neq r_3$.

Para generar la población de hijos se hace la cruza, la cual consiste en los siguiente: para cada individuo i en la población actual, se genera un vector mutado V_i, posteriormente, para cada componente j del individuo hijo se genera un número aleatorio entre 0 y 1. Si el valor aleatorio es menor o igual a CR, se toma la componente j del vector mutado. De lo contrario, se toma la componente j del individuo i de la población actual. Cada que se genera un hijo, se evalúa en la función objetivo, si es menor a la del individuo actual con el que se generó, lo reemplaza en la población. El Algoritmo 1 muestra el procedimiento completo.

Nota: las ecuaciones 3.1 y 3.2 no garantizan que el vector mutado V no salga de los valores que pueden adquirir, esto debe ser corregido en el código, en nuestro caso, describimos como resolvieron este problema los autores en el segundo punto de la nota en la sección 3.3.4.

Algoritmo 1 Propuesta original ED

Require: N, D, G, F, CR
1: Iniciar la población inicial X
2: Determinar la mejor solución X_B de la población X
3: **for** $g := 1$ to G **do**
4: **for** $i := 0$ to N **do**
5: Seleccionar 3 individuos al azar X_{r1}, X_{r2} y X_{r3} tal que $X_{r1} \neq X_{r2} \neq X_{r3}$
6: $V_i = X_{r_1} + F \cdot (X_{r2} - X_{r3})$ ▷ Creamos el vector mutado
7: **for** $j := 0$ to D ▷ Creamos el individuo hijo H de dimensión D
8: **if** Rand() $\leq CR$ **then**
9: $H_j = V_{ij}$
10: **else**
11: $H_j = X_{ij}$
12: **end if**
13: **end for**
14: Verificamos si H es mejor solución que X_i, de ser el caso lo reemplaza
15: Si H resulta mejor que B lo reemplaza
16: **end for do**
17: **end for**
18: Regresar X_B

Otra variante para generar al vector mutado es utilizando al mejor individuo conocido hasta el momento X_B y a cuatro individuos aleatorios de la población actual X_{r_1}, X_{r_2} X_{r_3} y X_{r_4} tales que $r_1 \neq r_2 \neq r_3 \neq r_4$, definimos un vector diferencia vd como $vd = (X_{r_1} - X_{r_2}) + (X_{r_3} - X_{r_4})$, el vector mutado V_i lo definimos como: $V_i = X_B + F * vd$, ver Ec. 3.2 y usamos éste vector para generar la cruza. Ver Algoritmo 2.

$$\begin{aligned} vd &= (X_{r1} - X_{r2}) + (X_{r3} - X_{r4}) \\ V &= X_B + F \cdot vd \end{aligned} \tag{3.2}$$

Siendo X_{r_1}, X_{r_2} X_{r_3} y X_{r_4} individuos escogidos al azar tal que $r_1 \neq r_2 \neq r_3 \neq r_4$, X_B la mejor solución actual y F el factor de escalamiento.

La forma cómo funcionan los algoritmos 1 y 2 es la misma, comienzan creando una población inicial de soluciones (X) y guardando la mejor solución inicial (X_B), después iteran un número de generaciones G que determina el usuario, en cada una de ellas deben crear un hijo H para cada solución X_i de la población, para lo cual primero ocurre la **mutación** al calcular el vector mutado V_i, posteriormente viene la **cruza** al construir el hijo ya sea con las componentes de su padre X_i o con las del vector mutado V_i, después se hace la **selección** cuando se decide si el hijo H tiene una mayor aptitud que su padre X_i y por lo tanto toma su lugar en la población, también se verifica si el hijo H reemplazará a la mejor solución actual X_B. La diferencia entre los algoritmos está en la **mutación**, pues en la propuesta original ver Algoritmo 1, se crea el vector mutado con 3 individuos tomados al azar X_{r_1}, X_{r_2} y X_{r_3}, usando la Ec . 3.1, mientras que en el Algoritmo 2 se

utiliza la mejor solución actual X_B y 4 individuos escogidos al azar X_{r_1}, X_{r_2} X_{r_3} y X_{r_4} con la formula ver Ec. 3.2.

Algoritmo 2 ED usando al mejor individuo conocido al momento

Require: N, D, G, F, CR
1: Iniciar la población inicial X
2: Determinar la mejor solución X_B de la población X
3: **for** $g := 1$ to G **do**
4: **for** $i := 0$ to N **do**
5: Tomar 4 individuos al azar X_{r1}, X_{r2}, X_{r3} y X_{r4} tal que $X_{r1} \neq X_{r2} \neq X_{r3} \neq X_{r4}$
6: $vd = (X_{r1} - X_{r2}) + (X_{r3} - X_{r4})$
7: $V_i = X_B + F \cdot vd$ ▷ Creamos el vector mutado
8: **for** $j := 0$ to D **do** ▷ Creamos el individuo hijo H de dimensión D
9: **if** Rand() $\leq CR$ **then**
10: $H_j = V_{ij}$
11: **else**
12: $H_j = X_{ij}$
13: **end if**
14: **end for**
15: Verificamos si H es mejor solución que X_i, de ser el caso lo reemplaza
16: Si H resulta mejor que X_B lo reemplaza
17: **end for**
18: **end for**
19: Regresar X_B

3.2. Representación

Se pueden usar los AE para tareas de *clustering* con dos enfoques diferentes.

En el primero, se trabajaría con una lista de tamaño N que contendría la etiqueta del grupo al que pertenece cada elemento. Por ejemplo, un arreglo de tamaño N con elementos divididos en 3 grupos se vería de la siguiente manera:

$$\underbrace{0, 1, 2, 0, 1, \ldots, 0, 1, 2}_{N}$$

En este caso los operadores de cruza, mutación y selección se aplicarían sobre dicha cadena de enteros, posteriormente se evaluaría mediante alguna función para determinar su valor como respuesta, y después de algunas iteraciones se obtendría la cadena de etiquetas más adecuada al problema en cuestión. Dado que el algoritmo de ED fue creado pensando en problemas de optimización continua, esta representación no sería adecuada.

El otro enfoque trabaja sobre los centroides de cada grupo, definidos como un arreglo de tamaño $C \cdot D$, siendo C el número máximo de grupos en el que se van a repartir los datos y D la dimensión de los datos con los que se trabaja. En este enfoque, cada número del arreglo debe ir en el rango (x_{min}, x_{max}) siendo éstos el valor mínimo y máximo dentro

de los datos. Un arreglo de esta implementación, considerando a lo más 3 grupos, que viven en el rango (0,1) se ve de la siguiente forma:

$$(\underbrace{[0.5, 0.2, 0.3]}_{\text{centroide grupo 1}}, \underbrace{[0.4, 0.3, 0.1]}_{\text{centroide grupo 2}}, \underbrace{[0.1, 0.8, 0.7]}_{\text{centroide grupo 3}})$$

Con este enfoque los operadores de cruza, mutación y selección se aplican directamente a los centroides, lo que se traduce a moverlos de lugar hasta que se encuentra el lugar más apropiado en el plano que generan los datos, para que al momento de clasificarlos y evaluar la clasificación se obtenga la mayor puntuación en la función de aptitud del problema.

3.3. Clustering con evolución diferencial

Existen varias propuestas para hacer tareas de agrupamiento utilizando evolución diferencial, ver por ejemplo [19], [29] y [30]. En esta tesis estudiamos el algoritmo llamado "Automatic Clustering using an Improved Differential Evolution Algorithm (ACDE)" [6].

En ACDE los individuos están compuestos por los centroides de la solución que se va refinando durante el proceso de búsqueda. Algo interesante de esta propuesta es que puede definir automáticamente el número de grupos en el cual dividir los datos, por lo que no es necesario que el usuario conozca de antemano cuántos grupos tiene su conjunto de datos. Otro aspecto que resaltan los autores es que su función de aptitud, la cual nos sirve para decidir si un individuo es más apto que otro, puede utilizar cualquier índice de validación (**IV**) de la siguiente forma:

$$aptitud(X) = \frac{1}{IV(X) + eps} \tag{3.3}$$

Siendo *eps* un valor positivo muy pequeño usado para evitar divisiones entre cero.

3.3.1. Parámetros y notación utilizada

ACDE recibe los siguientes parámetros:

- Dimensión (D): tamaño de los datos con los que trabaja, si son datos en $\mathbb{R}^2$ la dimensión es 2.
- Número máximo de agrupamientos (k_{max}): cantidad máxima de grupos que debe considerar el algoritmo para dividir los datos.
- Tamaño de la población (N): se refiere a cuántos individuos se van a tener en cada generación.
- x_{min} es el valor más pequeño en el conjunto de datos.
- x_{max} es el valor más grande en el conjunto de datos.
- Número máximo de evaluaciones permitidas en la función objetivo (*eval*).

- Función objetivo a optimizar ($F_{objetivo}$).
- Valor máximo que puede tomar la probabilidad de cruza (cr_{max}).
- Valor mínimo que puede tomar la probabilidad de cruza (cr_{min}).
- Número de iteraciones del algoritmo (NI).

3.3.2. Individuos

En ACDE cada individuo tiene dos componentes básicos:

- Umbrales: Se define como un arreglo de flotantes con el umbral de cada centroide. Un umbral mayor o igual a 0.5 significa que el centroide está activo, por lo que incrementa en 1 el número de grupos que buscará formar el algoritmo.
- Centros: centroides generados.

Considerando un problema de dimensión D, con K_{max} como la cantidad máxima de grupos en la que se van a dividir los datos. Un individuo es un arreglo de tamaño $k_{max} + (D \cdot k_{max})$ y se vería de la siguiente forma:

$$\underbrace{([U_1, U_2, ..., U_{k_{max}}]}_{\text{Umbrales}} \underbrace{[C_1, C_2, ..., C_{k_{max}}])}_{\text{centroides de los grupos}}$$

donde cada centroide C_i es un arreglo de flotantes con longitud D.

Además de los atributos mínimos anteriores para formar un individuo, nos interesa saber la siguiente información asociada a cada individuo:

- Activos: Se define como un arreglo con los centroides cuyo umbral es mayor o igual a 0.5.
- Aptitud: número flotante que representa la "calidad" de la agrupación.
- Objetivo: valor que toma el individuo al ser evaluado en la función objetivo.
- Partición: vector que contiene las etiquetas que se asignan a los datos.

A continuación, daremos un ejemplo de cómo se ve un individuo y que significarían los resultados que nos arroje al ser evaluado. Consideremos tener un conjunto de datos con 6 elementos, sean $k_{max} = 3$, $k_{min} = 2$, con $x_{min} = 0$ y $x_{max} = 9$, un posible individuo en la población seria:

$$\underbrace{([0.7, 0.5, 0.3],}_{\text{Umbrales}} \underbrace{[(0, 5), (2, 3), (7, 4)])}_{\text{centroides de los grupos}}$$

Suponiendo que al ser evaluado arroja los siguientes resultados: partición $= [0, 1, 1, 0, 1, 0]$, aptitud $= 0.6$ y objetivo $= 1.66$.
Analizando al individuo y sus resultados podemos obtener la siguiente información: es una

solución que tiene dos centroides activos $(0, 5)$ y $(2, 3)$, pues sus respectivos umbrales son mayores o iguales a 0.5. Basándonos en su partición, contando desde 0, los elementos 0, 3 y 5 fueron asignados al grupo con el centroide 0: $(0, 5)$ y al grupo con el centroide 1: $(2, 3)$ se le asignaron el 1, 2 y 4. Al ser evaluada la solución en la función objetivo (CS) obtuvo un valor de 1.66 y su aptitud correspondiente es de 0.6.

3.3.3. Operadores genéticos

Los operadores genéticos actúan sobre los umbrales y centros de los individuos. Primero, se crea un individuo **mutado** V, utilizando la propuesta original descrita en la sección 3.1, mediante la fórmula: $V = X_{r_1} + F \cdot (X_{r_2} - X_{r_3})$. Supongamos que tenemos $F = 0.5$, y se realiza una mutación con los siguientes individuos:

$X_{r_1} = ([0.2, 0.3, 0.4], [(1, 2), (2, 2), (4, 1)])$
$X_{r_2} = ([0.4, 0.6, 0.7], [(2, 0), (0, 5), (3, 1)])$
$X_{r_3} = ([0.1, 0.3, 0.5], [(2, 3), (1, 5), (2, 4)])$

Las operaciones se realizan elemento a elemento de cada individuo, por lo que tendríamos el vector mutado V de la siguiente forma:

$$V = ([0.2, 0.3, 0.4], [(1, 2), (2, 2), (4, 1)]) + 0.5 \times \begin{array}{cccccc} 0.4 & 0.6 & 0.7 & (2, 0) & (0, 5) & (3, 1) \\ - \; 0.1 & 0.3 & 0.5 & (2, 3) & (1, 5) & (2, 4) \\ \hline 0.3 & 0.3 & 0.2 & (0, -3) & (-1, 0) & (1, -3) \end{array}$$

Después de la multiplicación tenemos:

$$V = \begin{array}{cccccc} 0.2 & 0.3 & 0.4 & (1, 2) & (2, 2) & (4, 1) \\ + \; 0.15 & 0.15 & 0.01 & (0, -15) & (-5, 0) & (5, -15) \\ \hline 0.35 & 0.45 & 0.41 & (1, -13) & (-3, 2) & (9, -14) \end{array}$$

Por lo tanto, el vector mutado queda de la siguiente forma:

$$V = ([0.35, 0.45, 0.41], [(1, -13), (-3, 2), (9, -14)])$$

El operador de **cruza** genera un **hijo** a partir un individuo mutado V y un individuo **padre**. Este operador requiere de una probabilidad de cruza CR, se genera un número aleatorio en el rango $(0, 1)$ para decidir cada componente del hijo, si el valor generado es menor a CR el hijo tendrá la componente del vector mutado, en caso contrario tendrá la componente de su padre. Consideremos el siguiente vector padre:

$$P = ([0.5, 0.6, 0.7][(0, 1), (2, 1), (3, 2)])$$

una probabilidad de cruza $CR = 0.45$, y que la cruza se realizará con el vector mutado que acabamos de calcular:

$$V = ([0.35, 0.45, 0.41], [(1, -13), (-3, 2), (9, -14)])$$

dado que hay en total 9 componentes, (3 de umbrales [0.35, 0.45, 0.41] y 6 de 3 centros en $\mathbb{R}^2$ [(1,-13),(-3,2),(9,-14)]) se generaran 9 números aleatorios. Sean: [0.21, 0.13, 0.65,

0.47, 0.52, 0.31, 0.88, 0.92, 0.49] los números aleatorios generados, el hijo quedaría de la siguiente forma:

$$H = ([0.35, 0.45, 0.7], [(0, 1), (-3, 1), (3, 2)])$$

3.3.4. Operador de selección

Este operador se aplica después de haber creado un hijo a partir de un padre y el vector mutado. Recordemos que para evaluar un individuo es necesario hacer lo siguiente:

1. Generar la partición a partir de los centros activos. Cada dato se asigna al centro activo más cercano.

2. Evaluar la partición con el índice de validación que se esté usando (CS en nuestro caso).

3. Finalmente, se calcula la aptitud utilizando la ecuacion 3.5.

Una vez que se conoce la aptitud del individuo **padre** y del individuo **hijo**, se da preferencia al individuo cuya aptitud sea mayor y éste será parte de la población de la siguiente generación, mientras que el otro individuo se desecha.

Nota: Los autores de ACDE Swagatam Das, Ajith Abraham y Amit Konar [6], mencionan 2 puntos importantes que deben cumplir los individuos, con la finalidad de evitar errores al momento de querer calcular la aptitud del individuo:

- Tendrán al menos 2 centroides activos: después de hacer la cruza se verifica cuantos están activos, si son menos de 2 se escogen y activan al azar 2 de ellos.

- Cada centroide deberá tener al menos 2 elementos (datos) en él, de lo contrario significa que el centroide está fuera del rango de los datos. Para solucionar esto, se dividen los n datos entre los k centroides activos del individuo, de forma que cada grupo tenga n/k datos y asignando a cada dato su centroide más cercano.

En este trabajo de investigación se observó una implicación del segundo punto antes mencionado: se debe revisar no sólo que los grupos tengan al menos 2 elementos, sino también que la cantidad de grupos formados sea igual a la de los grupos **activos**. De lo contrario, si únicamente se revisa que los grupos formados contengan 2 o más puntos, al momento de calcular la aptitud del individuo nos dará error.

3.4. Resultados obtenidos

A continuación se muestran los resultados que se obtuvieron al replicar los experimentos que reportan Swagatam Das, Ajith Abraham y Amit Konar en [6]. Para ello se ejecutó ACDE con los parámetros igual que los autores: $k_{max} = 20$, $Cr_{max} = 1$, $Cr_{min} = 0.5$, $N = 10$, $MaxEval =$ 100,000 y NI=31. x_{min}, x_{max} y D dependen del conjunto de datos de entrada. Los experimentos consideran los cuatro conjuntos de prueba descritos en la sección 2.4.

3.4.1. Basado en la medida CS

Ésta es la función objetivo que utilizan originalmente los autores de ACDE en su función aptitud.

$$aptitud(X) = \frac{1}{CS(X) + eps} \tag{3.4}$$

La Tabla 3.1 contiene los resultados reportados por los autores que buscamos replicar, en su trabajo nos brindan las puntuaciones de su algoritmo en 4 aspectos: el número de grupos encontrados, la puntuación que obtuvo en CS, la distancia *intra clusters* promedio y la distancia promedio entre los *clusters*, las cuales corresponde al numerador y denominador, respectivamente, de la medida CS (ver ecuación 2.5.)

Como se puede observar, nuestra implementación replicó el número de grupos encontrados en tres de los cuatro conjuntos de datos donde se llevaron a cabo las pruebas. En el conjunto de datos donde hay mayor diferencia es en Vidrio, donde se encontraron tres grupos cuando deberían ser seis. Por el contrario, en el conjunto de datos Iris, se encontraron exactamente 3 grupos, superando a lo reportado por Swagatam Das, Ajith Abraham y Amit Konar.

Considerando los valores obtenidos en la *medida CS*, podemos decir que nuestra implementación obtiene mejores resultados en los conjuntos de Iris y Vino, mientras que en Cáncer y Vidrio los resultados fueron peores. Con respecto a la distancia *intra clusters* se desea tener un valor pequeño (los elementos en el grupo son más parecidos) y vemos que nuestra implementación tiene distancias más grandes en los conjuntos Iris Vino y Cáncer. Mientras que en Vidrio logra valores mucho más pequeños.

Finalmente, vemos que en los conjuntos Iris, Vino y Cáncer nuestra implementación logra valores mucho más grandes en la distancia entre clusters, lo cual es deseable. Mientras que en Vidrio obtiene un valor mucho más pequeño. Por lo anterior, podemos decir que en estos conjuntos de datos tener una distancia grande entre grupos no implica que se obtenga una distancia pequeña entre los elementos de un grupo, por lo cual se debe tener cuidado al hacer conclusiones considerando estas distancias por separado.

Conjunto de datos	Trabajo	Grupos encontrados	Puntuación CS	Distancia intra clusters	Distancia entre clusters
Iris	Original	3.25	0.6643	**3.1164**	2.5931
(3 grupos)	Nuestro	**3.0**	**0.5956**	5.3467	**8.9764**
Vino	Original	**3.25**	0.9249	**4.046**	3.1483
(3 grupos)	Nuestro	3.26	**0.5953**	1034.89	**1740.96**
Vidrio	Original	**6.05**	**0.3324**	563.247	**853.62**
(6 grupos)	Nuestro	3.0	0.45	**9.6108**	21.1903
Cáncer	Original	**2.0**	**0.4532**	**4.2439**	3.2577
(2 grupos)	Nuestro	2.0333	0.7057	26.7095	**37.8280**

Tabla 3.1: ACDE usando la medida CS en su función de aptitud. Se reporta la media de 31 ejecuciones de nuestra implementación y la media reportada por los autores. Los mejores resultados están resaltados en negritas.

En la Tabla 3.2 presentamos el análisis estadístico de las 31 ejecuciones del algoritmo

y es resaltable que ninguna desviación estándar es mayor a 0.1, lo que nos indica que las respuestas que nos da el algoritmo son muy consistentes y no dependen del azar.

Conjunto de datos	Promedio	Desviación estándar	Mejor	Peor	Mediana
Iris	0.5956	1.11e-16	0.5956	0.5956	0.5956
Vino	0.5954	0.0163	0.5899	0.6581	0.5899
Vidrio	0.4533	0.0232	0.4444	0.5250	0.4444
Cáncer	0.7057	0.0325	0.6989	0.8802	0.6989

Tabla 3.2: Análisis estadístico de los resultados obtenidos por ACDE usando la medida CS en su función de aptitud. Se reporta la media, la desviación estándar, la mejor y peor solución, y la solución que corresponde a la mediana, todas ellas considerando la medida CS.

ACDE vs. métodos clásicos

En este apartado, comparamos ACDE con respecto a dos métodos de clustering clásicos: K-Means y clustering aglomerativo. Los métodos fueron evaluados usando tres indicadores: el índice de validación CS y las métricas para medir el desempeño ARI y Completitud, descritas en la sección 2.2.2. Tanto los métodos de K-Means y clustering aglomerativo como los indicadores ARI y completitud fueron tomados de la librería **sklearn**. Es importante mencionar que los dos métodos clásicos requieren que se indique el número de grupos que deben generar. En este caso se ejecutaron indicando el número de grupos que tiene cada conjunto de prueba.

La Tabla 3.3 muestra los resultados obtenidos tras ejecutar 31 veces cada uno de los métodos en los cuatro conjuntos de prueba. Dado que clustering aglomerativo es un método determinista, solo se reporta el valor que obtuvo en cada indicador. De la tabla podemos resaltar lo siguiente:

- En la medida CS, como era de esperarse, el algoritmo ACDE obtuvo siempre la mayor puntuación dado que CS es utilizada en la definición de la función objetivo.
- En la métrica índice de aleatoriedad ajustado (ARI), ACDE obtuvo una puntuación menor a K-Means y a clustering aglomerativo, con excepción del conjunto de datos **Cáncer**, donde incluso supera a K-Means.
- Con respecto al indicador de completitud ACDE se desempeña mejor que las técnicas clásicas en los conjuntos Iris y Cáncer, mientras que en Vino y Vidrio obtiene peores resultados que éstas.

Algoritmo		CS			ARI			Completitud		
Iris	ACDE	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	**Mejor**	**Peor**	**Media**
		0.5956	**0.5956**	**0.5956**	0.5637	0.5637	0.5637	**0.9202**	**0.9202**	**0.9202**
		Desviación Estándar			Desviación Estándar			**Desviación Estándar**		
		+	**1.11e-16**		*	0.0		+	**0.0**	
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		0.8108	0.8129	0.8108	0.7302	0.7163	0.7302	0.7649	0.7474	0.7649
		Desviación Estándar			Desviación Estándar			Desviación Estándar		

		* 0.0003	0.0024	* 0.0030
	Clustering Aglomerativo	Resultado obtenido 0.8003 Desviación Estándar 2.22e-16	**Resultado obtenido 0.7311 Desviación Estándar + 0.0**	Resultado obtenido 0.7795 Desviación Estándar 0.0
Vino	ACDE	**Mejor 0.5899 Peor 0.6581 Media 0.5953 Desviación Estándar + 0.0163**	Mejor 0.2843 Peor -0.0030 Media 0.016 Desviación Estándar * 0.0691	Mejor 0.3338 Peor 0.1854 Media 0.2043 Desviación Estándar * 0.0370
	K-Means	Mejor 0.8108 Peor 0.8129 Media 0.8108 Desviación Estándar 0.0003	**Mejor 0.7302 Peor 0.7163 Media 0.7302 Desviación Estándar + 0.0024**	**Mejor 0.7649 Peor 0.7474 Media 0.7649 Desviación Estándar + 0.0030**
	Clustering Aglomerativo	Resultado obtenido 1.0339 Desviación Estándar * 0.0	Resultado obtenido 0.3684 Desviación Estándar 2.22e-16	Resultado obtenido 0.4162 Desviación Estándar 1.11e-16
Vidrio	ACDE	**Mejor 0.4444 Peor 0.5250 Media 0.4533 Desviación Estándar + 0.0232**	Mejor 0.0059 Peor 0.0039 Media 0.0059 Desviación Estándar * 0.0003	Mejor 0.3453 Peor 0.3274 Media 0.3447 Desviación Estándar * 0.0032
	K-Means	Mejor 1.0997 Peor 1.4327 Media 1.4422 Desviación Estándar * 0.1110	**Mejor 0.2470 Peor 0.2701 Media 0.2625 Desviación Estándar + 0.0088**	**Mejor 0.4459 Peor 0.4680 Media 0.4614 Desviación Estándar + 0.0072**
	Clustering Aglomerativo	Resultado obtenido 1.4233 Desviación Estándar 4.44e-16	Resultado obtenido 0.2620 Desviación Estándar 5.55e-17	Resultado obtenido 0.4410 Desviación Estándar 0.0
Cáncer	ACDE	**Mejor 0.5899 Peor 0.6581 Media 0.5953 Desviación Estándar + 0.0325**	Mejor 0.8483 Peor 0.0050 Media 0.0331 Desviación Estándar * 0.1513	Mejor 0.7397 Peor 0.1540 Media 0.1736 Desviación Estándar * 0.1051
	K-Means	Mejor 1.0970 Peor 1.0970 Media 1.0970 Desviación Estándar * 2.22e-16	Mejor 0.8464 Peor 0.8464 Media 0.8464 Desviación Estándar 0.0	Mejor 0.7528 Peor 0.7528 Media 0.7528 Desviación Estándar 1.11e-16
	Clustering Aglomerativo	Resultado obtenido 1.0780 Desviación Estándar 0.0	**Resultado obtenido 0.8689 Desviación Estándar + 0.0**	**Resultado obtenido 0.7927 Desviación Estándar + 2.22e-16**

Tabla 3.3: ACDE usando CS vs. K-Means y Clustering aglomerativo. Se reportan los resultados obtenidos utilizando los indicadores CS, ARI y completitud, tras 31 ejecuciones de cada método. Se han coloreado los resultados para destacarlos, en negritas y con un símbolo + en la esquina están los mejores resultados, los peores tienen un asterisco y están de rojo, esto basándonos en la puntuación *Media*.

Análisis de los resultados obtenidos

Es importante que recordemos que el algoritmo ACDE verifica que cada grupo activo tenga al menos dos elementos en él para evitar que los centroides se salgan del rango del problema, además de que al menos debe haber dos grupos activos para evitar divisiones entre 0.

Al revisar el etiquetado asignado por el algoritmo ACDE basándose en la función CS, se observó que tiende a hacer un "gran grupo" en el cual se concentra la mayoría de los datos, y los grupos restantes tendrán aquellos datos que estén más alejados al centroide del

"gran grupo". Esto toma sentido al recordar la definición de la medida CS, pues se divide la distancia interna de los grupos entre la distancia que hay de un grupo a otro, al crecer el denominador el valor de la función debería disminuir. Por lo cual, el algoritmo separa lo más posible los centroides entre sí, dejando alguno "más céntrico" a los datos asignándole la mayoría de estos, y los centroides restantes tendrían solo los datos más "marginados" en el espacio del problema. Esto pasa en todos los conjuntos de datos excepto en Iris donde se crean dos "grandes grupos".

Lo anterior puede ser una explicación de los resultados tan bajos en los indicadores ARI y completitud en todos los conjuntos de datos, con la excepción en el conjunto Iris donde obtiene buena puntuación en la métrica ARI, pues en dicho conjunto de datos hay dos "grandes grupos" y no solo uno.

3.4.2. Basado en el índice de validación DB

Para ésta sección reemplazamos el índice de validación CS por DB, quedando así la función objetivo:

$$aptitud(X) = \frac{1}{DB(X) + eps} \tag{3.5}$$

Aunque los autores Swagatam Das, Ajith Abraham y Amit Konar no reportan resultados en el índice DB, mencionan que es posible usarlo, así que decidimos ver cuál de los dos (CS y DB) nos daba mejores puntuaciones, para poder comparar nuestros resultados basados en DB con los reportados en CS calculamos las distancias *intra* y *entre* clusters las soluciones encontradas, así como su puntuación ontenida en CS. En la Tabla 3.4 se reportan los resultados.

Los resultados más cercanos al trabajo original respecto a grupos encontrados son los obtenidos en *Cáncer* y *Vino*. Si consideramos el índice DB, nuestra implementación obtuvo mejores resultados que la original en todos los conjuntos de datos excepto en Cáncer, donde obtuvo un resultado muy parecido. Con respecto a la distancia *intra cluster* vemos que en Vidrio nuestra implementación obtiene mejores resultados. Mientras que la distancia entre clusters indica que nuestra implementación logró mejores resultados en Iris, Vino y Cáncer. Por lo anterior, podemos concluir nuevamente que tener grupos muy separados no necesariamente indica que la distancia entre elementos de un grupo sea pequeña.

Es importante notar, que la desviación estándar de las 31 corridas es cercana a cero, lo que significa que ACDE obtuvo resultados muy similares en todas las ejecuciones, ver Tabla 3.5.

Conjunto de datos	Trabajo	Grupos encontrados	Puntuación CS	Distancia intra clusters	Distancia entre clusters
Iris	Original	**3.05**	0.4645	**3.1633**	2.8387
(3 grupos)	Nuestro	2.0	**0.3835**	4.9862	**7.9432**
Vino	Original	3.25	3.0432	**4.4212**	3.1029
(3 grupos)	Nuestro	**3.0**	**0.3499**	959.2433	**1103.3526**

Vidrio (6 grupos)	Original	**6.05**	1.0092	501.757	**893.46**
	Nuestro	2.0	**0.2952**	**9.0355**	14.3956
Cáncer (2 grupos)	Original	2.05	**0.5203**	**4.4212**	3.1029
	Nuestro	**2.0**	0.5246	26.8482	**37.0603**

Tabla 3.4: ACDE usando el índice de validación BD en su función de aptitud. Se reporta la media de 31 ejecuciones de nuestra implementación y la media reportada por los autores. Se resaltan en negritas los mejores resultados.

Conjunto de datos	Promedio	Desviación estándar	Mejor	Peor	Mediana
Iris	0.3835	1.11e-16	0.3835	0.3835	0.3835
Vino	0.3499	1.66e-16	0.3499	0.3499	0.3499
Vidrio	0.2952	5.55e-16	0.2952	0.2952	0.2952
Cáncer	0.5246	0.0670	0.5017	0.7569	0.5017

Tabla 3.5: Análisis estadístico de los resultados obtenidos por ACDE usando el índice de validación DB en su función de aptitud. Se reporta la media, la desviación estándar, la mejor y peor solución, y la solución que corresponde a la mediana, todas ellas considerando el índice DB.

ACDE vs. métodos clásicos

Al igual que en la Sección 3.4.1, en este apartado, se compara ACDE con respecto a K-Means y clustering aglomerativo, utilizando los indicadores DB, ARI y Completitud. La Tabla 3.6 muestra los resultados obtenidos tras ejecutar 31 veces cada uno de los métodos en los cuatro conjuntos de prueba, de dicha tabla se resalta lo siguiente:

- ACDE obtuvo mejores puntuaciones en DB como era de esperarse.
- Con respecto a ARI, ACDE tiene un desempeño pobre en comparación con las técnicas clásicas, excepto en el conjunto *Cáncer* donde su mejor respuesta obtuvo una mejor puntuación que K-Means y estuvo muy cerca del clustering aglomerativo.
- En completitud, ACDE supera a los métodos clásicos en los conjuntos de datos Iris y Vidrio.

	Algoritmo	DB			ARI			Completitud		
Iris	ACDE	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	**Mejor**	**Peor**	**Media**
		0.3835	**0.3835**	**0.3835**	0.5681	0.5681	0.5681	**0.9999**	**0.9999**	**0.9999**
		Desviación Estándar			Desviación Estándar			**Desviación Estándar**		
		+	**1.11e-16**		*	1.11e-16		+	**1.11e-16**	
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		0.6623	0.6663	0.6623	0.7302	0.7163	0.7302	0.7649	0.7474	0.7649
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.0007			0.0024			*	0.0030	
	Clustering Aglomerativo	Resultado obtenido			**Resultado obtenido**			Resultado obtenido		
		0.6566			**0.7311**			0.7795		
		Desviación Estándar			**Desviación Estándar**			Desviación Estándar		

		* 1.11e-16	+ **0.0**	0.0
Vino	ACDE	**Mejor Peor Media** **0.3499 0.3499 0.3499** **Desviación Estándar** + **1.66e-16**	Mejor Peor Media 0.0495 0.0495 0.0495 Desviación Estándar * 1.38e-17	Mejor Peor Media 0.3313 0.3313 0.3313 Desviación Estándar * 1.11e-16
	K-Means	Mejor Peor Media 0.5342 0.5342 0.5342 Desviación Estándar 1.11e-16	**Mejor Peor Media** **0.7302 0.7163 0.7302** **Desviación Estándar** + **0.0024**	**Mejor Peor Media** **0.7649 0.7474 0.7649** **Desviación Estándar** + **0.0030**
	Clustering Aglomerativo	Resultado obtenido 0.5357 Desviación Estándar * 0.0	Resultado obtenido 0.3684 Desviación Estándar 2.22e-16	Resultado obtenido 0.4162 Desviación Estándar 1.11e-16
Vidrio	ACDE	**Mejor Peor Media** **0.2952 0.2952 0.2952** **Desviación Estándar** + **5.55e-17**	Mejor Peor Media 0.0105 0.0105 0.0105 Desviación Estándar * 3.46e-18	**Mejor Peor Media** **0.5076 0.5076 0.5076** **Desviación Estándar** + **1.11e-16**
	K-Means	Mejor Peor Media 0.8769 0.9649 0.9558 Desviación Estándar 0.0295	**Mejor Peor Media** **0.2470 0.2701 0.2625** **Desviación Estándar** + **0.0088**	Mejor Peor Media 0.4459 0.4680 0.4614 Desviación Estándar 0.0072
	Clustering Aglomerativo	Resultado obtenido 0.9817 Desviación Estándar * 4.44e-16	Resultado obtenido 0.2620 Desviación Estándar7 5.55e-17	Resultado obtenido 0.4410 Desviación Estándar * 0.0
Cáncer	ACDE	**Mejor Peor Media** **0.5017 0.7569 0.5017** **Desviación Estándar** + **0.0670**	Mejor Peor Media 0.8574 0.0050 0.0617 Desviación Estándar * 0.2112	Mejor Peor Media 0.7653 0.1540 0.1952 Desviación Estándar * 0.1507
	K-Means	Mejor Peor Media 0.7572 0.7572 0.7572 Desviación Estándar 3.33e-16	Mejor Peor Media 0.8464 0.8464 0.8464 Desviación Estándar 0.0	Mejor Peor Media 0.7528 0.7528 0.7528 Desviación Estándar 1.11e-16
	Clustering Aglomerativo	Resultado obtenido 0.7897 Desviación Estándar * 4.44e-16	**Resultado obtenido** **0.8689** **Desviación Estándar** + **0.0**	**Resultado obtenido** **0.7927** **Desviación Estándar** + **2.22e-16**

Tabla 3.6: ACDE usando DB vs. K-Means y Clustering aglomerativo. Se reportan los resultados obtenidos utilizando los indicadores DB, ARI y completitud, tras 31 ejecuciones de cada método. Se han coloreado los resultados para destacarlos, en negritas y con un símbolo + en la esquina están los mejores resultados, los peores tienen un asterisco y están de rojo, esto basándonos en la puntuación *Media*.

Análisis de los resultados obtenidos

De nueva cuenta, el promedio de las desviaciones estándar indica que el algoritmo es estable y sus resultados no dependen del azar. Sin embargo, resalta que en **Cáncer** la mejor solución y la solución media tengan una notable diferencia en ARI y completitud.

Al revisar las etiquetas asignadas, el algoritmo vuelve a crear un "gran grupo", esto se puede deber a la definición de DB, pues como se menciona en [5], DB asigna puntuaciones muy similares a una partición de dos grupos y a una de tres, dado que el tercer grupo tiene pocos elementos. Lo anterior sugiere que el índice de validación DB en la función objetivo desprecia grupos con pocos elementos y tiende a unirlos para formar uno más grande.

En la sección apéndice se encuentra la figura 7.2, donde se muestran los tiempos de ejecución de esta implementación, concretamente en las gráficas a y b. También contiene información del entorno donde se ejecutaron las pruebas y la referencia a la liga del repositorio con los códigos implementados.

Capítulo 4

Optimización por cúmulo de partículas

La idea del algoritmo por cúmulo de partículas nació en 1995 [12], cuando Kennedy y Eberhart decidieron hacer una simulación por computadora que imitara el vuelo de las aves en busca de comida. Les resultó interesante como las aves parecían sincronizadas en el vuelo y realizaban movimientos que aparentan ser una coreografía, en la cual cada individuo actuaba en respuesta a su situación, pero considerando también la de los demás. La optimización por cúmulo de partículas (PSO por sus siglas en inglés) tiene relación con la computación evolutiva y con la inteligencia artificial, el concepto detrás es muy simple, requiere de pocos conocimientos matemáticos para que pueda ser implementado, y no requiere de muchos recursos computacionales en cuanto a memoria y cantidad de operaciones a ejecutar. Este algoritmo define que cada partícula es una posible solución al problema que se enfrenta. Lo que hace, es colocar un determinado número de partículas en una posición aleatoria dentro del espacio de soluciones e ir moviéndolas, buscando la mejor posición en él. Encontrar la mejor posición significa que encontró la mejor configuración de los valores en los parámetros del problema. Durante el movimiento por el espacio de soluciones, cada partícula va a recordar la mejor posición que ha visitado P_{best}, basándose en la puntuación que obtuvo al ser evaluada en la función objetivo a optimizar. Además conocerá la mejor posición que haya sido visitada por cualquier partícula del grupo G_{best} y se moverá torno a dichas posiciones. El proceso termina cuando la mejor posición global G_{best} no sufra cambios en un determinado número de iteraciones o cuando se alcance el límite establecido para éstas. Generalmente son los autores quienes experimentan y establecen la cantidad de iteraciones para ambos casos.

4.1. Algoritmo

Sea d el número de variables de decisión del problema a optimizar, se define a la partícula X y su velocidad V como vectores de flotantes de tamaño d. El vector X guarda la posición de la partícula mientras que el vector V representa su velocidad.

$$\vec{X} = x_1, x_2, ..., x_d$$

$$\vec{V} = v_1, v_2, ..., v_d$$

La parte fundamental del algoritmo es el movimiento de las partículas, que consiste en sumar una velocidad a la posición actual de éstas, y la modificación de dicha velocidad para que pueda "guiarlas" a la mejor zona posible del espacio de soluciones. Para llevar a cabo la modificación de la velocidad, se necesitan los elementos P_{best} y G_{best}, que se refieren a la mejor posición alcanzada por la partícula y a la mejor posición encontrada hasta el momento, respectivamente. Con estos dos elementos podemos definir cómo varía la velocidad en cada iteración del algoritmo:

$$\vec{V_{i+1}} = \vec{V_i} + c_1 \times r_1 \times (P_{best} - \vec{X}) + c_2 \times r_2 \times (G_{best} - \vec{X}) \tag{4.1}$$

Siendo $\vec{X}$ la posición actual de la partícula, c_1 y c_2 constantes positivas que controlan el tamaño de paso entre iteraciones, y r_1 junto a r_2 números aleatorios en el rango $(0, 1)$. Una vez modificada la velocidad de la partícula, se puede llevar a cabo el movimiento en el espacio de soluciones: $\vec{X} = \vec{X} + \vec{V}$. El Algoritmo 3 muestra el procedimiento completo.

Algoritmo 3 Optimización por Cúmulo de Partículas

1: Inicializar la población de partículas de tamaño P con posiciones $\vec{X}$ y velocidades $\vec{V}$ aleatorias.
2: Calcular el valor de la mejor posición personal P_{best} y la general G_{best}, así como el valor que adquieren en la función objetivo $F(P_{best})$ y $F(G_{best})$.
3: **while** Criterio de paro no alcanzado **do**
4: **for** $i := 0$ to P **do**
5: **if** $F(\vec{X_i}) < F(P_{best})$ **then**
6: $P_{best} = \vec{X_i}$
7: $F(P_{best}) = F(\vec{X_i})$
8: **end if**
9: **if** $F(\vec{X_i}) < F(G_{best})$ **then**
10: $G_{best} = \vec{x_i}$
11: $F(G_{best}) = F(\vec{x_i})$
12: **end if**
13: $\vec{V_i} = \vec{V_i} + r_1 \times (P_{best} - \vec{X_i}) + r_2 \times (G_{best} - \vec{X_i})$
14: $\vec{X_i} = \vec{X_i} + \vec{V_i}$
15: **end for**
16: **end while**

El Algoritmo 3 comienza creando una población de partículas de tamaño P, donde cada partícula tiene una posición $\vec{X}$ y una velocidad $\vec{V}$, después guarda la mejor posición de la población G_{best} y cada partícula guarda la mejor posición que ha visitado P_{best}. Cuando se tienen dichos valores comienza a iterar el ciclo principal, el cual se compone de 3 partes: (1) verifica si la posición actual de la partícula es la mejor que ha visitado, de ser el caso actualiza el valor de P_{best}, (2) verifica si la posición actual de la partícula es mejor a la que ha visitado cualquier otra partícula, de ser el caso actualiza el valor de G_{best}, (3) mueve la partícula actual a una nueva posición, para esto primero actualiza su velocidad con la ecuación 4.1 y posteriormente suma esa velocidad a la posición actual. Como se explicó al inicio del capítulo, el ciclo principal itera hasta cumplirse un criterio de paro generalmente definido los autores del algoritmo.

4.2. Representación

Como se explicó en la sección anterior, cada partícula y su velocidad son vectores de dimensión d siendo éste el número de variables de decisión que tiene el problema que se aborda. Para hacer **clustering** con este método, necesitamos identificar tres elementos primordiales:

- La dimensión d del problema en cuestión: definiremos d como $k \times R$ siendo k el número de grupos en los cuales se dividirán los datos a etiquetar: Y R la dimensión de dichos datos.
- El rango en que deben estar las componentes de la partícula. Este rango estará conformado por el mínimo y máximo valor que adquieren los datos a clasificar.
- La velocidad mínima y máxima que tendrán las partículas.

De forma general, una partícula y su velocidad tienen la siguiente forma:

$$\vec{X} = \overbrace{(\underbrace{[x_1, x_2, ..., x_R]}_{\text{grupo 1}}, \underbrace{[x_1, x_2, ..., x_R]}_{\text{grupo 2}}, ..., \underbrace{[x_1, x_2, ..., x_R]}_{\text{grupo k}})}^{\text{Longitud } k \times R} \tag{4.2}$$

$$\vec{V} = \overbrace{(\underbrace{[v_1, v_2, ..., v_R]}_{\text{grupo 1}}, \underbrace{[v_1, v_2, ..., v_R]}_{\text{grupo 2}}, ..., \underbrace{[v_1, v_2, ..., v_R]}_{\text{grupo k}})}^{\text{Longitud } k \times R} \tag{4.3}$$

Supongamos que se necesita dividir en 3 grupos un conjunto de datos que viven en $\mathbb{R}^4$, los mínimos de las componentes son $[0.1, 0.1, 0.2, 1]$ y sus máximos $[2, 3, 1, 5]$. Además, las partículas deben tener una velocidad en el intervalo $(-1, 1)$. Por lo tanto, tenemos que $k = 3$, $r = 4$, $v_{min} = -1$ y $v_{max} = 1$, la componente 1 debe estar en el rango $(0.1, 2)$, la componente 2 en $(0.1, 3)$, etc. Por lo cual una partícula se vería de la siguiente forma:

$$\vec{X} = (\underbrace{[0.5, 1.0, 0.7, 1.5]}_{grupo1}, \underbrace{[0.8, 2.0, 0.6, 3.4]}_{grupo2}, \underbrace{[1.1, 2.5, 0.8, 2.5]}_{grupo3})$$

Su velocidad tiene las mismas dimensiones, con la diferencia que todos sus componentes deben estar en el rango $(-1, 1)$, por ejemplo:

$$\vec{V} = (\underbrace{[-0.2, 0.5, 0.7, 0.6]}_{grupo1}, \underbrace{[-0.5, -0.2, 0.6, 0.4]}_{grupo2}, \underbrace{[0.9, -0.7, 0.8, 0.1]}_{grupo3})$$

4.3. Clustering con optimización por cúmulo de partículas

Para aplicar PSO en problemas de *clustering*, nos basaremos en el trabajo de Alireza Ahmadyfard y Hamidreza Modares [2], el cual combina el algoritmo de PSO con K-Means para lograr un mejor rendimiento que el de ambas técnicas por separado.

Alireza Ahmadyfard y Hamidreza Modares mencionan que PSO es bueno para encontrar óptimos globales, pero su convergencia al acercarse a la solución es baja, por lo cual utilizan PSO para encontrar la mejor zona del espacio de búsqueda, y después proceden a utilizar K-Means, el cual converge a la solución más rápido pues es bueno encontrando óptimos locales.

La forma en que opera el algoritmo propuesto en [2] es la siguiente: se crea una población inicial de partículas en lugares aleatorios dentro del espacio de búsqueda, mediante PSO se mueven dichas partículas hasta encontrar la zona donde esté la solución óptima, una vez ahí se cambia a K-Means para que continúe con la búsqueda y de con la solución óptima. Ver Algoritmo 4.

4.3.1. Parámetros y notación utilizada

El algoritmo PSO-KM requiere los siguientes parámetros:

- x_{min} es un vector con el valor mínimo que se puede adquirir para cada componente.
- x_{max} es un vector con el valor máximo que se puede adquirir para cada componente.
- v_{min} velocidad mínima que puede tener una partícula en cada componente.
- v_{max} velocidad máxima que puede tener una partícula en cada componente.
- k número de grupos a formar.
- n_{part} número de partículas que componen el cúmulo.
- $iter_{pso}$ número máximo de iteraciones para PSO.
- $iter_{km}$ número máximo de iteraciones para K-Means.
- c_1 y c_2 constantes de aceleración, los creadores de PSO [11] sugieren darles el valor 2 basados en sus pruebas.
- *GenSinCambios* si no cambia la mejor solución general durante esta cantidad de generaciones termina la parte de PSO.

4.3.2. Partículas

Las partículas son un vector que contiene los centroides de los k grupos a formar, y tienen la forma mostrada en la ecuación 4.2, su velocidad se ve como en la ecuación 4.3, pero con unos cambios a la ecuación 4.1 quedando de la siguiente forma:

$$\vec{V_{i+1}} = \omega \times \vec{V_i} + c_1 \times r_1(P_{best} - \vec{X}) + c_2 \times r_2(G_{best} - \vec{X})$$

Como se puede observar, se han introducido nuevos elementos, definimos cada componente de la fórmula anterior de la siguiente manera:

- r_1 y r_2 son valores aleatorios en el intervalo $(0, 1)$.

- ω es una variable que controla el impacto de velocidades anteriores en la actual.
- P_{best} es la mejor posición que ha visitado la partícula.
- G_{best} es la mejor posición de todas las que han sido visitadas por el grupo.
- c_1 y c_2 son constantes positivas que controlan el tamaño de paso entre iteraciones.
- $\vec{X}$ es la posición actual de la partícula.

4.3.3. Combinando PSO con K-Means

El Algoritmo 4 muestra cómo combinan los autores de [2] PSO con K-Means. En el Algoritmo 5 se puede consultar el funcionamiento clásico de K-Means. La función objetivo que utiliza el Algoritmo 4 se describe en la ecuación 2.3. Como el algoritmo maximiza una función de aptitud, tomaremos el negativo de la función objetivo, por lo que nos queda la aptitud como en la ecuación 4.4

$$F(G) = -\frac{\sum_{\forall G_j \in G} \sum_{\forall i \in G_j} d_{ij}^2}{n} \tag{4.4}$$

Donde: G son los grupos formados, G_j es el conjunto de elementos asignados al grupo j, d_{ij} la distancia del punto i al centro del grupo j y n el número de elementos.

Algoritmo 4 PSO-KMeans

Require: $iter_{pso}$, $iter_{Km}$, $GenSinCambios$, x_{min}, x_{max}, v_{min}, v_{max}

```
Obtener la población inicial X⃗ de tamaño P
Calcular el valor que obtienen las partículas en la función de aptitud
Para cada partícula X⃗_i guardar su mejor posición visitada P_best−i
Obtener la mejor posición global G_best
while iter <iter_pso do
    for i := 0 to P do
        Actualizar la posición y velocidad de la i-ésima partícula X⃗_i
        Si la nueva posición se sale de su rango [x_min, x_max], se reinicializan
        como x_min ó x_max. Lo mismo pasa con la velocidad y v_min, v_max
        if Apt(X⃗_i) >Apt(P_best−i) then
            P_best−i = X⃗_i
        end if
        if Apt(X⃗_i) >G_best then
            G_best = X⃗_i
        end if
    end for
    iter = iter+1
    if G_best no cambia then
        SinCambios += 1
```

else
    SinCambios := 0
end if
if SinCambios == GenSincambios **then**
    Salir del ciclo while
end if
end while
Usar K-Means partiendo de la mejor posición encontrada G_{best}
Regresar etiquetado obtenido.

El Algoritmo 4 comienza utilizando PSO con 2 criterios de paro: (1) llegar a un máximo de iteraciones $iter_{pso}$, (2) la mejor solución G_{best} no cambia en un numero de iteraciones igual a $GenSinCambios$. Primero se crea una población de soluciones $\vec{X}$ de tamaño P, guarda la mejor posición que ha visitado cada partícula P_{best} y la mejor posición de la población G_{best}, después itera sobre las partículas, inicia actualizando la posición de cada una de ellas $\vec{X}_i$ y verificando se encuentren en el espacio de soluciones, en caso contrario se les asigna una posición aleatoria dentro de él, después verifica si la posición donde están es mejor a las que has visitado ($Apt(\vec{X}_i) > Apt(P_{best-i})$), de ser así se actualiza la mejor posición visitada a la actual ($P_{best} = \vec{X}_i$), también se revisa si la posición actual supera a la mejor de la población ($Apt(\vec{X}_i) > Apt(G_{best})$) y si es el caso se actualiza ($G_{best} = \vec{X}_i$). Finalmente se comprueba si el algoritmo alcanzó el número máximo de iteraciones sin cambios permitidas. Una vez terminada la parte de PSO se utiliza K-Means partiendo de la mejor solución encontrada al momento G_{best} y se limitan sus iteraciones a $iter_{Km}$, el resultado obtenido por K-Means es el que regresa el algoritmo.

Algoritmo 5 K-Means

Require: k,$Iteraciones$,$Tolerancia$,$datos$
$\vec{C} := [c_1, c_2, ..., c_k]$ siendo k el número de grupos a formar.
i :=0
$\vec{L} = [l_1, l_2, .., l_n]$ donde n es el número de datos que hay
while i <$Iteraciones$ **do**
    for j := 0 to d **do**
        Encontrar el centroide c_m más cercano al dato $datos_j$
        $L_j = m$ ▷ m es el índice del centroide más cercano al dato
    end for
    Recalcular cada centroide como la media de sus elementos asignados
    Si los centroides tienen un cambio menor a la *tolerancia*, salir del ciclo
    Si L se mantiene igual al final de la iteración, salir del ciclo
end while
Regresar $\vec{L}$

El Algoritmo 5 es la versión clásica de K-Means, requiere que el usuario introduzca la cantidad de grupos a formar k, y los datos a clasificar *datos*, los parámetros *iteraciones*

y *Tolerancia* son opcionales en la mayoría de implementaciones pues tienen valores por defecto. La forma como trabaja es la siguiente: comienza con un conjunto de centroides $\vec{C}$ de tamaño k, el cual puede ser introducido por el usuario, en caso contrario lo crea aleatoriamente, después hace una lista de etiquetas L de tamaño n, siendo n la longitud de *datos*, donde cada etiqueta l_j representa el grupo al que pertenece el j-ésimo dato. En cada iteración se asocia cada dato a su centroide más cercano, posteriormente los centroides son recalculados como la media de todos los datos que se le asociaron, el proceso termina en cualquiera de los siguientes casos: (1) se alcanza el número máximo de *iteraciones*, (2) los centroides tienen un cambio menor a la *Tolerancia* por lo que se considera no cambia la respuesta, (3) cuando la lista de etiquetas $\vec{L}$ no cambia tras una iteración.

4.4. Resultados obtenidos

Se llevaron a cabo dos tipos de experimentos, el primero utiliza la función de aptitud presentada en la ecuación 4.4, y el segundo toma como función de aptitud el negativo del índice DB [7]. En ambos casos se usaron los siguientes parámetros: $GenSincambios = 10$ imitando al trabajo original, c_1, $c_2 = 2$, esto lo proponen los autores del algoritmo PSO original [11], $v_{min} = 0$, $v_{max} = 1$, $n_{part} = 10$, $iterPso = 100$, $iterKmeans = 50$, x_{min}, x_{max} y k dependen del conjunto de datos de entrada.

Con el fin de compararnos con respecto a los autores Alireza Ahmadyfard y Hamidreza Modares, creamos tres conjuntos de datos sintéticos con la librería `scikit-learn` [20], los parámetros utilizados son los que ellos reportan y se pueden consultar en la Tabla 4.1.

Conjunto de datos	Elementos	Grupos	Centroides	Dimension	Desviación estándar de los puntos al centroide
Sintético 1	210 70 por grupo	3	(1) [-10,-10] (2) [-6,-6] (3) [-3,-3]	2	1.0
Sintético 2	210 70 por grupo	3	(1) [-10,-10] (2) [-8.5.-8.5] (3) [-3,-3]	2	1.0
Sintético 3	250 50 por grupo	5	(1) [-20,-20,-20] (2) [-10,-10,-10] (3) [-5,-5,-5] (4) [0,0,0] (5) [19,19,19]	3	4.0

Tabla 4.1: Parámetros utilizados en la creación de los conjuntos de datos sintéticos para las pruebas que buscan replicar los resultados reportados en el trabajo original.

Para medir el desempeño del algoritmo, los autores del trabajo PSO-KM utilizaron dos indicadores: proporción de error (ER, por sus siglas en inglés) y suma cuadrada de distancias (MSE, por sus siglas en inglés). Por lo anterior, adoptamos estos mismos indicadores.

ER consiste en contar la cantidad de datos mal clasificados y dividir dicha cantidad entre el total de elementos a clasificar.

MSE no está pensado para ser una medida en la similitud de etiquetados, por lo cual debemos hacer unas aclaraciones para poder utilizarlo y que tengan más "sentido" los resultados que arroja tomando dos etiquetados como entrada.

- MSE calcula el cuadrado de la diferencia entre un valor predicho y el valor real, por lo cual, castigará en mayor medida el etiquetar un elemento de la clase 0 con clase 2 que clasificándolo como 1. MSE(0,2) = 4 >MSE(0,1) = 1.

- El orden de las etiquetas importa. Como se calcula la diferencia elemento a elemento, aunque nuestro programa dividiera de forma correcta los datos, sí le pone una etiqueta distinta a la que tienen los datos en la realidad. No importará que sean el mismo grupo con diferente nombre, contará todo como error. Para sortear este inconveniente, se hicieron todas las posibles formas de etiquetar los resultados, y se evaluó MSE con cada forma de etiquetado, quedándonos con el valor más pequeño, pues corresponde a cuando el programa le da el nombre que corresponde a los grupos que creó.

- Medir el MSE entre etiquetados no nos dice realmente qué tan cerca están los resultados entre sí, por lo explicado en el primer punto, por lo cual calculamos la MSE que hay entre los centroides de los grupos formados a los centroides reales de los datos. Si la distancia fuera 0 significa que son el mismo punto, por lo cual concluimos que ambos grupos tienen los mismos elementos. Como no sabemos de antemano qué centroides corresponden, hicimos la combinación de todas las posibles parejas de centroides (reales y encontrados por el algoritmo) y nos quedamos con la puntuación más baja.

4.4.1. Basado en la función objetivo propuesta por los autores

El objetivo de esta sección es replicar los resultados publicados en el trabajo original propuesto por Alireza Ahmadyfard y Hamidreza Modares, se utiliza como función de aptitud la ecuación 4.4, en adelante llamada MDEC por ser la media de la distancia cuadrada.

En la Tabla 4.2 reportamos la media de los resultados tras 31 ejecuciones del algoritmo, y lo comparamos con los resultados que reportan los autores. En todos los conjuntos de datos excepto "sintético 2", el indicador ER es muy similar al reportado por los autores. Esto se debe a que no tenemos los mismos conjuntos pues carecemos de una semilla para generarlos, y depende completamente de la distribución de los datos en el conjunto. En la medida MSE obtenemos mejores resultados en todos los conjuntos de datos, pero esto se puede atribuir al segundo punto de las consideraciones hechas en la Sección 4.4.

Conjunto de datos	Medida	PSO-KM autores	PSO-KM nuestro
Iris	ER	**10.5 %**	11.1 %
	MSE	0.52	**0.11**
Cáncer	ER	**3.7 %**	3.9 %
	MSE	26.3	**0.039**
Sintético 1	ER	**0 %**	1 %
	MSE	1.11	**0.01**
Sintético 2	ER	**7.2 %**	12.4 %
	MSE	1.15	**0.12**

Sintético 3	ER	**12.1 %**	13.4 %
	MSE	38.43	**0.13**

Tabla 4.2: PSO-KM utilizando MDEC, ver ecuación 4.4, como función de aptitud. Se reporta la media tras 31 iteraciones y el valor proporcionado por los autores Alireza Ahmadyfard y Hamidreza Modares. Se resaltan en negritas los mejores resultados.

La Tabla 4.3 contiene el análisis estadístico de los resultados considerando las 31 corridas del algoritmo. Como se observa en la desviación estándar, los resultados son consistentes y no dependen de la suerte. Sin embargo, resalta que en el conjunto de datos Sintético 3 usando la medida "MSE a los centroides" se encuentra la mayor variación entre el mejor y peor resultado obtenido.

Conjunto de datos	Métrica de error	Promedio	Desviación estándar	Mejor	Peor	Mediana
Iris	ER	11.11 %	0.003	10.70 %	11.33 %	11.33 %
	MSE (etiquetados)	0.113	0.003	0.107	0.113	0.111
	MSE (centroides)	0.046	0.001	0.047	0.049	0.048
Cáncer	ER	3.9 %	0.001	3.80 %	4.0 %	3.95 %
	MSE (etiquetados)	0.039	0.001	0.038	0.040	0.039
	MSE (centroides)	0.137	0 006	0.128	0.141	0.141
Sintético 1	ER	1 %	0.0005	0.95 %	1 %	1 %
	MSE (etiquetados)	0.010	0.0005	0.009	0.01	0.009
	MSE (centroides)	0.003	0.0001	0.003	0.003	0.003
Sintético 2	ER	12.4 %	0.001	12.36 %	12.9 %	12.38 %
	MSE (etiquetados)	0.124	0.001	0.123	0.129	0.124
	MSE (centroides)	0.023	0.003	0.02	0.032	0.022
Sintético 3	ER	13.4 %	0.046	12.8 %	30.4 %	17.2 %
	MSE (etiquetados)	0.198	0.072	0.128	0.424	0.184
	MSE (centroides)	17.958	47.307	1.185	196.794	3.368

Tabla 4.3: Análisis estadístico de los resultados obtenidos usando como función de aptitud MDEC, ver ecuación 4.4 y las medidas de desempeño propuestas por Alireza Ahmadyfard y Hamidreza Modares, se reporta la mejor, peor y mediana solución encontradas tras ejecutar 31 veces el algoritmo.

PSO-KMeans vs. métodos clásicos

En esta sección utilizamos los cuatro conjuntos de datos reales descritos en la Sección 2.4 y comparamos PSO-KMeans (PSO-KM) contra las técnicas clásicas K-Means y clustering aglomerativo. Para esto se hicieron 31 corridas de cada algoritmo en cada uno de los conjuntos de datos y se reportan los resultados obtenidos en los índices de validación ARI y completitud descritas dentro de la Sección 2.2.2. La Tabla 4.4 muestra que PSO-KM y K-Means obtienen los mejores resultados en la mayoría de los escenarios, PSO-KM supera a K-Means casi siempre y cuando no lo hace queda muy cerca de sus resultados. El método

clásico clustering aglomerativo solo resalta en los conjuntos Iris y Cáncer, donde saca las mejores puntuaciones en *ARI* y *completitud*.

	Algoritmo	MDEC			ARI			Completitud		
Iris	PSO-KM	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	Mejor	Peor	Media
		0.526	**0.526**	**0.526**	0.730	0.716	0.726	0.765	0.747	0.759
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		+ **0.0**			0.006			0.008		
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		0.526	0.956	0.554	0.730	0.419	0.699	0.764	0.656	0.744
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		* 0.105			* 0.073			* 0.023		
	Clustering Aglomerativo	Resultado obtenido			**Resultado obtenido**			**Resultado obtenido**		
		0.529			**0.731**			**0.779**		
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		0.0			+ **0.0**			+ **0.0**		
Vino	PSO-KM	**Mejor**	**Peor**	**Media**	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media
		13318.48	**14771.43**	**13365.36**	**0.371**	**0.335**	**0.370**	0.433	0.429	0.429
		Desviación Estándar			**Desviación Estándar**			Desviación Estándar		
		+ **256.71**			+ **0.006**			0.001		
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	**Mejor**	**Peor**	**Media**
		13318.48	14913.80	13844.04	0.371	0.313	0.362	**0.451**	**0.419**	**0.434**
		Desviación Estándar			Desviación Estándar			**Desviación Estándar**		
		* 709.04			* 0.014			+ **0.009**		
	Clustering Aglomerativo	Resultado obtenido			Resultado obtenido			Resultado obtenido		
		13504.91			0.368			0.416		
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.0			0.0			* 0.0		
Vidrio	PSO-KM	Mejor	Peor	Media	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media
		1.571	1.985	1.670	**0.281**	**0.2**	**0.262**	0.485	0.344	0.444
		Desviación Estándar			**Desviación Estándar**			Desviación Estándar		
		* 0.127			+ **0.017**			0.030		
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	**Mejor**	**Peor**	**Media**
		1.571	1.965	1.641	0.298	0.209	0.259	**0.618**	**0.349**	**0.462**
		Desviación Estándar			Desviación Estándar			**Desviación Estándar**		
		0.101			* 0.015			+ **0.053**		
	Clustering Aglomerativo	**Resultado obtenido**			**Resultado obtenido**			Resultado obtenido		
		1.628			**0.262**			0.441		
		Desviación Estándar			**Desviación Estándar**			Desviación Estándar		
		+ **0.0**			+ **0.0**			* 0.0		
Cáncer	PSO-KM	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		28.292	28.292	28.292	0.852	0.846	0.848	0.759	0.753	0.755
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.0			0.003			0.003		
	K-Means	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	Mejor	Peor	Media
		28.291	**28.291**	**28.291**	0.851	0.846	0.846	0.759	0.752	0.753
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		+ **0.0**			* 0.001			* 0.001		
	Clustering Aglomerativo	Resultado obtenido			**Resultado obtenido**			**Resultado obtenido**		
		30.41			**0.869**			**0.792**		
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		* 0.0			+ **0.0**			+ **0.0**		

Tabla 4.4: PSO-KMeans vs. K-Means y Clustering aglomerativo. Se reportan los resultados obtenidos en los indicadores ARI y completitud, utilizando como aptitud MDEC, ver ecuación 4.4.Se han coloreado los resultados para destacarlos, en negritas y con un símbolo + en la esquina están los mejores resultados, los peores tienen un asterisco y están de rojo, esto basándonos en la puntuación *Media*.

Análisis de los resultados obtenidos

En la teoría PSO-KM debería ser mejor en la tarea de clustering que K-Means, pues se basa en este último y tiene el agregado de usar PSO, cosa que se ve en los resultados ya que lo supera en la mayoría de los escenarios, además, es importante observar que con PSO se gana estabilidad en los resultados, pues tienen una menor variación. Esto indica que PSO logra acercar a K-Means a la zona donde se encuentra el óptimo global para que posteriormente K-Means mejoré la precisión de la ubicación de los centroides.

Cuando comparamos PSO-KM con clustering aglomerativo observamos que en la función objetivo (MDEC) PSO-KM obtiene mejor puntuación en Iris, Vino y Cáncer, siendo Vidrio la única excepción. En *ARI* y *Completitud* PSO-KM tiene mejor puntuación en los conjuntos Vino y Vidrio, mientras que en Iris y Cáncer pierde por muy poco en ambas métricas, pues la mayor diferencia entre resultados ocurre en el conjunto Cáncer en la métrica *Completitud* donde PSO-KM pierde por 0.037.

4.4.2. Basado en el índice de validación DB

En esta sección se hicieron los mismos experimentos que en la anterior, pero en este caso se utiliza de función de aptitud el negativo de la métrica DB [7]. Se reportan los resultados tras 31 ejecuciones del algoritmo, se compara nuestra implementación con la original y más adelante se compara contra las dos técnicas clásicas K-Means y clustering aglomerativo. La Tabla 4.5 muestra la comparativa de los resultados obtenidos y se puede observar que en ER, DB da peores resultados. Contrario a lo que sucede con MSE, donde supera por mucho, pero, esto se podría atribuir a las consideraciones que se hicieron a la hora de utilizar MSE para calificar etiquetados.

Conjunto de datos	Medida	PSO-KM autores	PSO-KM nuestro
Iris	ER	**10.5 %**	18.4 %
	MSE	0.52	**0.109**
Cáncer	ER	**3.7 %**	3.9 %
	MSE	26.3	**0.039**
Sintético 1	ER	**0 %**	0.5 %
	MSE	1.11	**0.005**
Sintético 2	ER	**7.2 %**	38.6 %
	MSE	1.15	**0.386**
Sintético 3	ER	**12.1 %**	19.9 %
	MSE	38.43	**0.219**

Tabla 4.5: PSO-KM utilizando el negativo de DB [7], ver ecuación 2.4, como función aptitud. Se reporta la media tras 31 corridas y el valor proporcionado por los autores. Se resaltan en negritas los mejores resultados.

En la Tabla 4.6 vemos el análisis estadístico de los resultados, la desviación estándar nos dice que hay estabilidad en los resultados reportados. Al menos entre el mejor y la mediana, pero cabe resaltar que la diferencia entre la mejor y peor solución es muy amplia

en el conjunto Iris y los sintéticos 1 y 2. En general, se encuentra un resultado promedio muy cercano a la mediana en todos los conjuntos de datos.

Conjunto de datos	Métrica de error	Promedio	Desviación estándar	Mejor	Peor	Mediana
Iris	ER	27 %	0.184	10.70 %	48.7 %	11.33 %
	MSE (etiquetados)	0.286	0.204	0.107	0.527	0.1133
	MSE (centroides)	1.45	1.651	0.047	3.392	0.0469
Cáncer	ER	3.9 %	0.001	3.80 %	4.0 %	3.95 %
	MSE (etiquetados)	0.039	0.001	0.038	0.040	0.039
	MSE (centroides)	0.135	0.006	0.128	0.141	0.141
Sintético 1	ER	0.5 %	0.0	0.5 %	0.5 %	0.5 %
	MSE (etiquetados)	0.005 %	0.0	0.005 %	0.005 %	0.005 %
	MSE (centroides)	0.001 %	0.001	0.001 %	0.001 %	0.001 %
Sintético 2	ER	38.6 %	0.119	11.0 %	46.7 %	43.3 %
	MSE (etiquetados)	0.386	0.119	0.11	0.467	0.433
	MSE (centroides)	14.118	6.312	0.023	21.33	16.22
Sintético 3	ER	19.9 %	0.071	12.4 %	32.4 %	21.2 %
	MSE (etiquetados)	0.219	0.103	0.124	0.452	0.224
	MSE (centroides)	27.857	54.57	1.503	191.263	12.29

Tabla 4.6: Análisis estadístico de los resultados obtenidos usando como función de aptitud el negativo de DB [7], ver ecuación 2.4, y las medidas de desempeño propuestas por Alireza Ahmadyfard y Hamidreza Modares, se reporta la mejor, peor y mediana solución encontradas tras ejecutar 31 veces el algoritmo.

PSO-KMeans vs. métodos clásicos

La Tabla 4.7 contiene la comparativa de PSO-KM contra las técnicas clásicas K-Means y clustering aglomerativo. Resalta que en el indicador *ARI* el método clásico *clustering aglomerativo* obtuvo la mejor puntuación en todos los conjuntos de datos, además en *completitud* obtiene los mejores resultados en Iris y Cáncer.En este caso PSO-KM queda muy cerca de los resultados de K-Means tanto en *ARI* como *completitud* en todos los conjuntos de datos, incluso superándolo en ocasiones, la diferencia entre más grande entre PSO-KM y K-Means está en *ARI* del conjunto Iris.

	Algoritmo	DB			ARI			Completitud		
Iris	PSO-KM	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		0.662	0.879	0.756	0.730	0.420	0.592	0.765	0.657	0.710
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		*	0.097		*	0.147		*	0.045	
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		0.662	0.666	0.664	0.730	0.716	0.722	0.764	0.656	0.744
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.002			0.073			0.023		
	Clustering Aglomerativo	**Resultado obtenido**			**Resultado obtenido**			**Resultado obtenido**		
		0.656			**0.731**			**0.779**		
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		+	**0.0**		+	**0.0**		+	**0.0**	

Vino	PSO-KM	Mejor Peor Media 0.550 0.550 0.550 Desviación Estándar * 0.0	Mejor Peor Media 0.352 0.352 0.352 Desviación Estándar * 0.0	**Mejor Peor Media** **0.451 0.451 0.451** **Desviación Estándar** \+ **0.001**
	K-Means	Mejor Peor Media 0.534 0.558 0.539 Desviación Estándar 0.008	Mejor Peor Media 0.371 0.338 0.363 Desviación Estándar 0.011	Mejor Peor Media 0.451 0.428 0.434 Desviación Estándar 0.009
	Clustering Aglomerativo	**Resultado obtenido** **0.535** **Desviación Estándar** \+ **0.0**	**Resultado obtenido** **0.368** **Desviación Estándar** \+ **0.0**	Resultado obtenido 0.416 Desviación Estándar * 0.0
Vidrio	PSO-KM	**Mejor Peor Media** **0.688 1.180 0.871** **Desviación Estándar** \+ **0.098**	Mejor Peor Media 0.289 0.218 0.257 Desviación Estándar * 0.015	Mejor Peor Media 0.589 0.345 0.442 Desviación Estándar 0.051
	K-Means	Mejor Peor Media 0.751 1.235 0.905 Desviación Estándar 0.093	Mejor Peor Media 0.299 0.231 0.259 Desviación Estándar 0.015	**Mejor Peor Media** **0.607 0.362 0.452** **Desviación Estándar** \+ **0.048**
	Clustering Aglomerativo	Resultado obtenido 0.981 Desviación Estándar 0.0 * 0.0	**Resultado obtenido** **0.262** **Desviación Estándar 0.0** \+ **0.0**	Resultado obtenido 0.441 Desviación Estándar 0.0 * 0.0
Cáncer	PSO-KM	**Mejor Peor Media** **0.757 0.757 0.757** **Desviación Estándar** \+ **0.0**	Mejor Peor Media 0.852 0.846 0.849 Desviación Estándar 0.003	Mejor Peor Media 0.759 0.753 0.756 Desviación Estándar 0.003
	K-Means	**Mejor Peor Media** **0.757 0.757 0.757** **Desviación Estándar** \+ **0.0**	Mejor Peor Media 0.851 0.846 0.846 Desviación Estándar * 0.001	Mejor Peor Media 0.759 0.752 0.753 Desviación Estándar * 0.001
	Clustering Aglomerativo	Resultado obtenido 0.789 Desviación Estándar * 0.0	**Resultado obtenido** **0.869** **Desviación Estándar** \+ **0.0**	**Resultado obtenido** **0.792** **Desviación Estándar** \+ **0.0**

Tabla 4.7: PSO-KM vs. K-Means y Clustering aglomerativo. Se reportan los resultados obtenidos en los indicadores ARI y completitud, utilizando el negativo de DB [7], ver ecuación 2.4, como aptitud. Se han coloreado los resultados para destacarlos, en negritas y con un símbolo + en la esquina están los mejores resultados, los peores tienen un asterisco y están de rojo, esto basándonos en la puntuación *Media*.

Análisis de los resultados obtenidos

En general, le fue peor a PSO-KM usando DB como función de aptitud en comparación con la propuesta original. Aun así, PSO-KM siempre estuvo cerca de los resultados que obtiene K-Means en los cuatro conjuntos de pruebas.

En la sección apéndice se encuentra la figura 7.2, donde se muestran los tiempos de ejecución de esta implementación, concretamente en las gráficas c y d. También contiene información del entorno donde se ejecutaron las pruebas y la referencia a la liga del repositorio con los códigos implementados.

Capítulo 5

Recocido simulado

Los orígenes del recocido simulado (RS) se remontan al trabajo de Metropolis en 1953 [17]. RS es un método probabilístico que se utiliza en problemas de optimización para buscar el mínimo global dentro de varios mínimos locales. Un trabajo icónico es el realizado por Kirkpatrick, Gelatt y Vecchi en 1983 [13], pues por primera vez se utilizó el algoritmo para resolver el problema del viajero considerando miles de ciudades a la vez.

RS debe su nombre a que trata de emular el cambio de estado de los metales, los cuales al ser fundidos sufren un cambio en la estructura de sus átomos, pasan de una configuración con alta energía a una de baja energía. Como se menciona en el artículo de EHL Aarts y PJM Van Laarhoven [1], para implementar un algoritmo de RS se requieren definir 4 aspectos:

- La forma que tendrán las soluciones: es la representación de cómo se ve una solución al problema en cuestión.
- Un método de perturbación: es la forma en que se genera una solución vecina mediante cambios (perturbaciones) en la solución principal.
- La función de costo: nos dice qué solución es mejor a otra al ser evaluadas.
- Forma de enfriamiento: en esta parte se requiere especificar la temperatura inicial y final del sistema, además de la función que se usará para disminuir dicha temperatura.

RS inicia con una solución **principal** aleatoria, la cual mediante una **perturbación** genera una solución **vecina**. Posteriormente, evalúa la puntuación obtenida en la función de costo por ambas soluciones, si la solución **vecina** es al menos tan buena como la que se tenía pasa a ser la solución **principal**, también se acepta una solución **vecina** peor a la **principal** con una probabilidad que depende de la temperatura actual del algoritmo y de qué tan mala es la nueva solución (ver ecuación 5.1), esto con el objetivo de tratar de no quedar atorado en un óptimo local. Al inicio del algoritmo se tiene una temperatura alta, que permite hacer cambios entre soluciones con más frecuencia, tras cada iteración disminuirá la temperatura y se irán aceptando solo los cambios que mejoren la solución principal, el algoritmo itera hasta llegar a una temperatura final establecida por el usuario. El Algoritmo 6 muestra el procedimiento completo.

$$p = \exp\left(\frac{f(principal) - f(vecina)}{T_i}\right) \tag{5.1}$$

Algoritmo 6 Recocido Simulado

Require: temperatura inicial ($T_{inicial}$), temperatura final (T_{final}), función de cambio en la temperatura ($T(t)$), función de perturbación ($P(X)$), $f(X)$

1: Inicializar temperatura: $T_j = T_{inicial}$
2: Generar la solución principal X_i
3: **while** $T_j \geq T_{final}$ **do**
4: $X_v = P(X_i)$
5: **if** $f(X_v) \leq f(X_i)$ **then**
6: $X_i = X_v$
7: **else if** random(0,1) $\leq \exp\left(\frac{f(principal)-f(vecina)}{T_j}\right)$ **then**
8: $X_i = X_v$
9: **end if**
10: $T_j = T(T_j)$
11: **end while**

El Algoritmo 6 muestra la forma clásica de recocido simulado, comienza estableciendo la temperatura actual T_j igual a la temperatura inicial $T_{inicial}$, después genera la solución principal X_i aleatoriamente y comienza a disminuir la temperatura, en cada iteración crea una solución prueba X_v mediante una perturbación a X_i, después evalúa si X_v reemplazará a X_i, lo cual sucede en los siguientes escenarios: (1) X_v tiene menor costo que X_i ($f(X_v) < f(X_i)$) o (2) se genera un número aleatorio entre (0,1) y resulta menor al valor que adquiere la ecuación 5.1, finalmente se actualiza la temperatura actual ($T_j = T(T_j)$) y el proceso finaliza cuando adquiere un valor igual o menor a la temperatura final T_{final}.

5.1. Clustering con recocido simulado

Existen varios trabajos donde se utiliza RS para llevar a cabo la tarea de *clustering* y generalmente la diferencia principal que tienen es la forma en que generan las soluciones vecinas. En este trabajo de tesis, se decidió utilizar el algoritmo SAGMDE propuesto por Julian Lee y David Perkins en [15] pues destaca al utilizar dos métodos para generar las soluciones vecinas y no solo uno como la mayoría. SAGMDE al ser un algoritmo de RS requiere los cuatro aspectos definidos en la introducción de este capítulo, a continuación detallaremos cada uno de ellos:

- **Forma de las soluciones**. Cada solución será un vector de tamaño $k \times R$, siendo k el número de grupos en los cuales repartir los datos y a su vez el número de centroides que contiene la solución, y R la dimensión en que se encuentran dichos datos, por lo tanto, el número de componentes de cada centroide. La forma general en que se ve

una solución es la siguiente:

$$\vec{S} = \overbrace{(\underbrace{[x_1, x_2, ..., x_R]}_{\text{Centroide 1}}, \underbrace{[x_1, x_2, ..., x_R]}_{\text{Centroide 2}}, ..., \underbrace{[x_1, x_2, ..., x_R]}_{\text{Centroide k}})}^{\text{Longitud } k \times R} \tag{5.2}$$

- **Método de perturbación**. Utiliza *mutación gaussiana* y *perturbación por utilidad*. La mutación gaussiana se utiliza en cada iteración moviendo un poco la solución por su vecindario actual. La perturbación por utilidad entra en juego cada 20 iteraciones, permitiendo un cambio significativo para explorar nuevos vecindarios en la zona de búsqueda.
- **Función de costo**. Se utiliza la distancia euclidiana al cuadrado de cada elemento a su centroide, ver ecuación 2.2, también se le conoce como SSE.
- **Forma de enfriamiento**. Utiliza un cambio lineal: $T(t_{i+1}) = t_i \cdot \alpha$. Es importante mencionar que SAGMDE opera en dos partes, para cada una de ellas utiliza la misma forma de enfriamiento pero establece valores independientes para las temperaturas iniciales y finales de cada etapa.

5.1.1. Parámetros y notación utilizada

El algoritmo SAGMDE utiliza los siguientes parámetros:

- k: número de grupos a formar.
- $T_{inicial}$: temperatura inicial para el método de mutación gaussiana.
- T_{final}: temperatura mínima para el método de mutación gaussiana.
- α: factor de enfriamiento para la temperatura usada en mutación gaussiana.
- $T_{distorsion}$: temperatura inicial para el método de distorsión.
- $Tf_{distorsion}$: temperatura mínima para el método de distorsión.
- $\alpha_{distorsion}$: factor de enfriamiento de la temperatura para la distorsión.
- $Max_{iteraciones}$: número máximo de iteraciones por temperatura.

5.1.2. Mutación gaussiana

Esta mutación añade un valor δ multiplicado por un aleatorio gaussiano g a cada componente de un centroide escogido al azar. El Algoritmo 7 muestra el procedimiento de la mutación gaussiana: primero se escoge al azar un centroide c_i de la solución que va a mutar, después se genera un número aleatorio g, posteriormente, a cada componente j del centroide c_i se le sumará δ veces el numero aleatorio g, finalmente se reasignan los elementos a clasificar con su centroide más cercano y se recalcula el costo de la solución.

Algoritmo 7 Mutación gaussiana

Require: *solucion*,δ
1: Seleccionar de *solucion* un centroide c_i al azar
2: generar un número aleatorio gaussiano g
3: **for** c_{ij} in c_i **do**
4: $\quad c_{ij} = c_{ij} + \delta g$
5: **end for**
6: Reasignar los elementos a su centroide más cercano
7: Calcular el costo de la nueva solución

Dado que δg afecta a todas las componentes de la misma forma, se tiene que escalar δ de acuerdo al rango de los datos o bien normalizar las componentes de la solución. Si se decide por escalar el factor δg, a cada componente de la solución se le agrega el factor $d_j \delta g$ siendo d_j la diferencia del máximo y mínimo valor que adquieren los datos de la componente j. En caso de normalizar los datos, cada componente de los datos se tiene que trasladar al rango (0,1), para esto utilizamos la ecuación 5.3. De esta forma no importa que las componentes estén en distintos rangos, al final todas tendrán un valor entre 0 y 1.

$$MinMax(x_i) = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{5.3}$$

5.1.3. Perturbación por distorsión y utilidad de los grupos

Esta forma de perturbación consta de dos partes como su nombre lo indica, la distorsión y utilidad de los grupos que ofrece la solución en cuestión. Definiremos la distorsión de un grupo como la sumatoria de las distancias entre cada elemento del grupo a su centroide, y su utilidad será dividir su distorsión entre la media de las distorsiones obtenidas en cada grupo. La distorsión de un grupo G_i se calcula entonces con la siguiente fórmula:

$$D(G_i) = \sum_{x \in G_i} d(x, C_i)$$

donde G_i el i-ésimo grupo y $d(x, C_i)$ la distancia euclidiana del punto x al centroide del grupo i. Una vez obtenidas las distorsiones de todos los grupos en la solución actual, se puede calcular la utilidad de cada uno de la siguiente forma:

$$U(G_i) = \frac{D_i}{D_{promedio}}$$

siendo D_i la distorsión del i-ésimo grupo, y $D_{promedio}$ el promedio de las distorsiones de los grupos.

La distorsión de un grupo nos indica qué tan separados están sus elementos entre sí. Una mayor distorsión indica que los elementos del grupo están más separados. Por otro lado, su utilidad nos dice si los elementos tienen una distorsión por debajo o por arriba del promedio. Si la utilidad es menor a 1 significa que la distorsión está por debajo del promedio, y al grupo se le considera de *baja utilidad*. Mientras que si es mayor a 1 estará por

encima del promedio y se considera de *alta utilidad*. Lo que busca este método es equiparar las utilidades existentes moviendo el centroide de un grupo de *baja utilidad* cerca de uno de *alta utilidad*, esto se hace de una forma específica detallada en la Sección 5.1.3. A grandes rasgos lo que se consigue es dividir un grupo con elementos muy dispersos en 2 grupos más pequeños con menor distorsión.

Movimiento de los centroides: el concepto de la hipercaja

Para mover el centroide de un grupo con baja utilidad cerca de un centroide con alta utilidad se realiza lo siguiente:

1. Escoger al azar un grupo con baja utilidad G_{baja} y definimos su centroide como c_{baja}.
2. Escoger al azar un grupo con alta utilidad G_{alta} y definimos su centroide como c_{alta}.
3. Creamos la hipercaja de volumen mínimo que contiene todos los elementos del grupo G_{alta}.
4. Definimos la *diagonal principal* de la hipercaja como la que va del vértice formado por los mínimos valores al vértice que contiene los máximos valores de los elementos en el grupo.
5. Ubicamos los centroides c_{baja} y c_{alta} de forma que dividan la diagonal principal en tres.

Una vez ubicados los centroides sobre la diagonal de la hipercaja, se reasignan los puntos a sus centroides más cercanos y se recalculan los centroides para que sean la media de sus elementos. En el Algoritmo 8 podemos ver el proceso completo y la Figura 5.1 muestra un ejemplo de este tipo de perturbación.

Algoritmo 8 Pseudocódigo perturbación por distorsión y utilidad

Require: *solucion*

Calcular la *distorsion* y *utilidad* de todos los grupos de la *solucion*.
Seleccionar al azar el centroide de un grupo con baja utilidad. C_{baja}
Seleccionar al azar un grupo con alta utilidad G_{alta}.
Tomamos de los elementos en G_{alta} el valor mínimo de cada una de sus componentes V_{min}.
Tomamos de los elementos en G_{alta} el valor máximo de cada una de sus componentes V_{max}.
$Diagonal = V_{max}$ - V_{min}
$V_{min} = V_{min} + 1/3 \cdot Diagonal$
$V_{max} = V_{max}$ - $1/3 \cdot Diagonal$
Reasignamos los elementos a su centroide más cercano.
Calcular el costo de la nueva solución.

Como podemos observar, el Algoritmo 8 sigue los 5 pasos enumerados al inicio de esta sección para mover los centroides. Primero calcula la *distorsión* y *utilidad* de cada grupo en

la solución, procede a seleccionar al azar un grupo con baja *utilidad* G_{baja} y su centroide C_{baja} (paso 1), y un grupo de alta *utilidad* G_{alta} junto con su centroide C_{alta} (paso 2). Después realizamos a la vez la hipercaja (paso 3) y le definimos su diagonal principal (paso 4), para esto, buscamos los valores mínimos V_{min} y máximos V_{max} que adquieren los elementos de G_{alta} en cada una de sus componentes, posteriormente la diagonal principal se define como: $Diagonal = V_{max} - V_{min}$. Finalmente, movemos los centroides C_{baja} y C_{alta} de forma que dividan la $Diagonal$ de la hipercaja en tercios (paso 5), lo anterior se hace con las siguientes formulas: $C_{baja} = V_{min} + 1/3 \cdot Diagonal$ y $C_{alta} = V_{max} - 1/3 \cdot Diagonal$. Ya que se movieron los centroides de lugar se reasignan los elementos a su centroide mas cercano y se calcula el costo de la nueva solución.

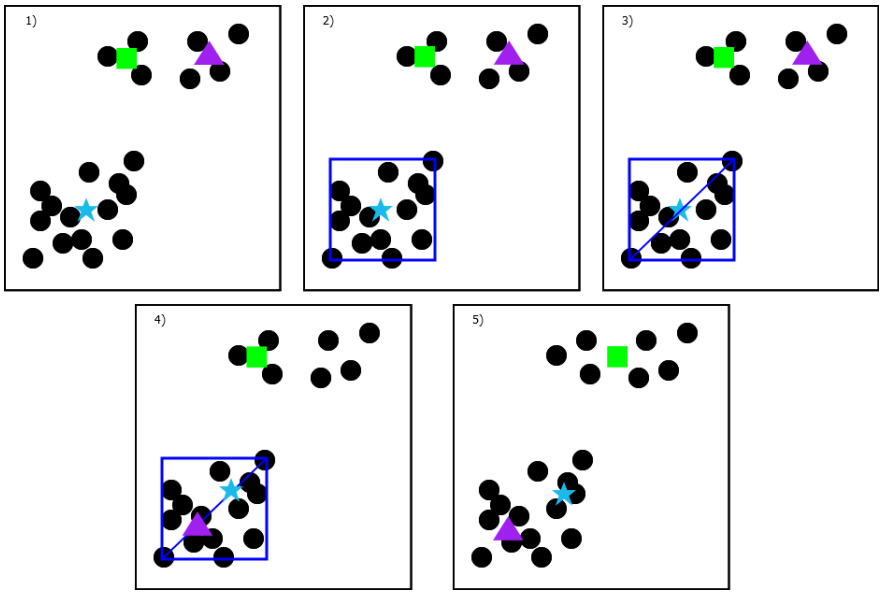


Figura 5.1: **Perturbación por distorsión y utilidad**. En el paso (1) se determinan los centroides de los grupos de alta y baja utilidad, en este caso solo el de azul (estrella) tiene alta utilidad, después se escoge al azar un centroide tanto de alta como de baja utilidad, para el ejemplo se tomó el azul (estrella) y el morado (triangulo) respectivamente. En el paso (2) se crea la hipercaja que contiene a los puntos del grupo de alta utilidad. El paso (3) consiste en determinar la diagonal principal de la hipercaja, la cual va desde los mínimos valores de cada componente en los datos hasta sus máximos valores. En (4) se ubican los centroides de baja y alta utilidad de forma que dividen la diagonal en tercios. Finalmente, en (5) reasignamos cada elemento a su centroide más cercano y recalculamos el centroide como la media de sus elementos.

5.2. Resultados obtenidos

El objetivo de esta sección es replicar los resultados que reportan los autores del algoritmo SAGMDE [15]. Por lo cual usaremos los mismos parámetros que ellos: $Max_{iteraciones} = 2n$ siendo n el número de datos del conjunto a clasificar, $T_{inicial} = 0.0015$, $T_{final} = 0.000001$, $\alpha = 0.98$, $T_{distorsion} = 6$, $Tf_{distorsin} = 0.025$ y $\alpha_{distorsion} = 0.985$.

Los conjuntos de datos son los descritos en la Sección 2.4 pero los valores de sus atributos han sido normalizados al intervalo (0,1) utilizando el método $MinMax$ descrito

en la Sección 5.1.2.

5.2.1. Basados en SSE

Utilizaremos SSE como la función objetivo a minimizar. En cada conjunto de datos se hicieron 31 corridas independientes del algoritmo, los parámetros usados son los descritos con anterioridad, los únicos que cambian son k y $Max_{iteraciones}$ pues dependen del conjunto de datos.

La Tabla 5.1 muestra las puntuaciones en SSE reportadas por los autores y las obtenidas por nuestra implementación. Se puede observar que en todos los conjuntos de datos los resultados son muy similares, llegando incluso a mejorar a los autores en el conjunto de datos Cáncer.

Iris

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	**6.998**	7.004
Promedio	**6.998**	7.004
Máximo	**6.998**	7.004
Desviación estándar	0	0.0

Vino

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	**48.954**	50.502
Promedio	**48.954**	50.823
Máximo	**48.954**	50.835
Desviación estándar	0	0.058

Vidrio

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	**18.241**	18.302
Promedio	**18.298**	18.499
Máximo	**18.382**	18.637
Desviación estándar	0.056	0.067

Cáncer

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	253.397	**242.314**
Promedio	253.397	**243.737**
Máximo	253.397	**243.790**
Desviación estándar	0	0.261

Tabla 5.1: Tabla comparativa de los resultados reportados por los autores tras 20 corridas del algoritmo, contra los obtenidos en nuestra implementación tras 31 ejecuciones. Se muestran las puntuaciones obtenidas usando la función objetivo SSE en los conjuntos de datos: Iris, Vino, Vidrio y Cáncer. Se resaltan en negritas los mejores resultados.

SAGMDE vs. Métodos clásicos

A continuación se presentan los resultados obtenidos en la comparativa del algoritmo SAGMDE contra los métodos clásicos K-Means y *clustering aglomerativo*, usando SSE, ARI y completitud.

La Tabla 5.2 contiene al análisis estadístico de los resultados obtenidos en cada conjunto de datos. Se puede observar que en SSE SAGMDE obtiene una mejor puntuación comparado a *clustering aglomerativo* en todos los conjuntos de datos excepto en Vino donde queda muy cerca a éste, comparado a K-Means siempre queda muy cerca de sus resultados, superándolo incluso en el conjunto Iris. En el indicador ARI, *clustering aglomerativo* se lleva el primer puesto en los conjuntos Iris, Cáncer y Vino, mientras que en Vidrio la mejor puntuación la obtiene K-Means, en este indicador *SAGMDE* supera a K-Means en el conjunto Iris y en los demás siempre está muy cerca de sus resultados. En *completitud* se dan puntuaciones muy similares a los obtenidos en *ARI*, teniendo a *clustering aglomerativo* con los mejores resultados en la mayoría de conjuntos, siendo la excepción en Vidrio donde *SAGMDE* logra superarlo aunque queda por debajo de K-Means.

	Algoritmo	SSE			ARI			Completitud		
Iris	SAGMDE	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	Mejor	Peor	Media
		7.004	**7.004**	**7.004**	0.716	0.716	0.716	0.747	0.747	0.747
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		+ **0.0**			0.0			0.0		
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		6.998	10.908	7.169	0.716	0.428	0.702	0.747	0.653	0.734
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.685			* 0.050			* 0.020		
	Clustering Aglomerativo	Resultado obtenido			**Resultado obtenido**			**Resultado obtenido**		
		7.173			**0.731**			**0.779**		
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		* 0.0			+ **0.0**			+ **0.0**		
Vino	SAGMDE	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		50.502	50.835	50.835	0.836	0.791	0.791	0.831	0.774	0.774
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		* 0.058			0.015			0.012		
	K-Means	**Mejor**	**Peor**	**Media**	**Mejor**	**Peor**	**Media**	**Mejor**	**Peor**	**Media**
		48.954	**49.015**	**48.975**	**0.899**	**0.802**	**0.846**	**0.874**	**0.791**	**0.828**
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		+ **0.015**			+ **0.017**			+ **0.016**		
	Clustering Aglomerativo	Resultado obtenido			Resultado obtenido			Resultado obtenido		
		49.547			0.368			0.416		
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.0			* 0.0			* 0.0		
Vidrio	SAGMDE	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	Mejor	Peor	Media
		18.302	**18.637**	**18.499**	0.175	0.159	0.166	0.340	0.321	0.332
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		+ **0.067**			* 0.003			* 0.004		
	K-Means	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		18.403	23.077	19.822	0.278	0.142	0.180	0.484	0.294	0.360
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		1.264			0.031			0.044		
	Clustering Aglomerativo	Resultado obtenido			**Resultado obtenido**			**Resultado obtenido**		
		19.837			**0.262**			**0.441**		
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		* |c|		0.0	+ **0.0**			+ **0.0**		
Cáncer	SAGMDE	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
		242.314	243.790	243.737	0.824	0.819	0.819	0.729	0.723	0.723
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		0.261			* 0.0009			* 0.001		
	K-Means	**Mejor**	**Peor**	**Media**	Mejor	Peor	Media	Mejor	Peor	Media
		238.557	**238.558**	**238.557**	0.851	0.846	0.847	0.759	0.752	0.753
		Desviación Estándar			Desviación Estándar			Desviación Estándar		
		+ **0.0001**			0.002			0.002		
	Clustering Aglomerativo	Resultado obtenido			**Resultado obtenido**			**Resultado obtenido**		
		255.426			**0.868**			**0.792**		
		Desviación Estándar			**Desviación Estándar**			**Desviación Estándar**		
		* 0.0			+ **0.0**			+ **0.0**		

Tabla 5.2: SAGMDE vs métodos clásicos. Se reportan los resultados obtenidos en los indicadores SSE, ARI y completitud. Se han coloreado los resultados para destacarlos, en negritas y con un símbolo + en la esquina están los mejores resultados, los peores tienen un asterisco y están de rojo, esto basándonos en la puntuación *Media*.

Análisis de los resultados obtenidos

Aunque *SAGMDE* no resalta en *ARI* ni *completitud* en ningún conjunto de datos, sus resultados no son malos, pues en Iris y Vino obtiene el segundo lugar en ambos indicadores, donde falla es en Vidrio y Cáncer al obtener los peores resultados, pero quedando muy

cercanos a los de K-Means. En este caso el método clásico *clustering aglomerativo* obtiene las mejores puntuaciones en *ARI* y *completitud* en casi todos los conjuntos de datos excepto en Vino, por lo cual resulta ser el mejor algoritmo. Después está K-Means el cual tanto en *ARI* como *completitud* obtiene los mejores resultados en Vino, los segundos mejores resultados en Vidrio y Cáncer, siendo Iris el único conjunto donde tiene los peores resultados. El tercer puesto es para *SAGMDE* ya que en general obtiene puntuaciones más bajas que K-Means aunque por muy poco, siendo la excepción en Iris donde llega a superarlo en ambos indicadores.

5.2.2. Basado en el índice de validación DB

En este caso, la función objetivo a minimizar será el índice de validación DB. Nuevamente se hicieron 31 ejecuciones independientes del algoritmo en cada conjunto de datos con los valores de los parámetros declarados con anterioridad, teniendo cambios solamente en k y $Max_{iteraciones}$ al ser dependientes del conjunto de datos.

La información en la Tabla 5.3 compara los resultados reportados por los autores contra los obtenidos por nuestra implementación, se puede observar que el mínimo obtenido es muy similar al del los autores en todos los conjuntos de datos, incluso superando su resultado en Cáncer, pero el promedio y máximo son mucho más grandes y por lo tanto peores, lo que provoca una desviación estándar más alta.

Iris

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	**6.998**	7.599
Promedio	**6.998**	16.443
Máximo	**6.998**	60.767
Desviación estándar	0	10.029

Vino

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	**48.954**	51.697
Promedio	**48.954**	54.889
Máximo	**48.954**	96.038
Desviación estándar	0	7.612

Vidrio

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	**18.241**	22.336
Promedio	**18.298**	29.453
Máximo	**18.382**	60.530
Desviación estándar	0.056	7.359

Cáncer

Puntuación SSE	SAGMDE autores	SAGMDE implementado
Mínimo	253.397	**246.132**
Promedio	**253.397**	309.133
Máximo	**253.397**	436.938
Desviación estándar	0	76.213

Tabla 5.3: Tabla comparativa de los resultados reportados por los autores tras 20 ejecuciones del algoritmo, contra los obtenidos en nuestra implementación tras 31 ejecuciones. Se muestran las puntuaciones obtenidas usando como función objetivo el índice DB en los conjuntos de datos: Iris, Vino, Vidrio y Cáncer. Se resaltan en negritas los mejores resultados.

SAGMDE vs. Métodos clásicos

Nuevamente se compara la implementación de SAGMDE con las técnicas clásicas K-Means y *clustering aglomerativo*, con la diferencia de que SAGMDE buscó minimizar el índice DB y no la función costo SSE.

En la Tabla 5.4 se presenta el análisis estadístico de los resultados obtenidos. De ella se puede observar que *SAGMDE* obtiene mejor puntuación en DB, lo cual es de esperarse pues es el único método que buscó minimizarlo, los métodos clásicos funcionan minimizando

SSE. Tanto en *ARI* como en *completitud* el método clásico *clustering aglomerativo* se lleva el primer lugar en todos los conjuntos de datos a excepción de Vidrio, donde *SAGMDE* lo supera. En esta ocasión *SAGMDE* supera en *ARI* y *completitud* a K-Means en 2 conjuntos de datos: Vidrio y Cáncer, en los restantes nuevamente obtiene puntuaciones más bajas pero muy cercanas a lo que reporta K-Means.

	Algoritmo	DB	ARI	Completitud
Iris	SAGMDE	**Mejor Peor Media** **0.426 0.740 0.715** **Desviación Estándar** + **0.076**	Mejor Peor Media 0.684 0.449 0.642 Desviación Estándar * 0.060	**Mejor Peor Media** **0.951 0.745 0.789** **Desviación Estándar** + **0.043**
	K-Means	Mejor Peor Media 0.760 0.861 0.772 Desviación Estándar * 0.020	Mejor Peor Media 0.716 0.428 0.702 Desviación Estándar 0.050	Mejor Peor Media 0.747 0.653 0.734 Desviación Estándar * 0.020
	Clustering Aglomerativo	Resultado obtenido 0.748 Desviación Estándar 0.0	**Resultado obtenido** **0.731** **Desviación Estándar** + **0.0**	Resultado obtenido 0.779 Desviación Estándar 0.0
Vino	SAGMDE	**Mejor Peor Media** **1.248 1.313 1.130** **Desviación Estándar** + **0.011**	Mejor Peor Media 0.853 0.463 0.820 Desviación Estándar 0.066	Mejor Peor Media 0.837 0.643 0.816 Desviación Estándar 0.032
	K-Means	Mejor Peor Media 1.305 1.321 1.313 Desviación Estándar 0.006	**Mejor Peor Media** **0.899 0.326 0.847** **Desviación Estándar** + **0.126**	**Mejor Peor Media** **0.874 0.478 0.831** **Desviación Estándar** + **0.077**
	Clustering Aglomerativo	Resultado obtenido 1.318 Desviación Estándar * 0.0	Resultado obtenido 0.368 Desviación Estándar * 0.0	Resultado obtenido 0.416 Desviación Estándar * 0.0
Vidrio	SAGMDE	**Mejor Peor Media** **0.629 0.815 0.713** **Desviación Estándar** + **0.055**	Mejor Peor Media 0.250 0.112 0.199 Desviación Estándar 0.036	**Mejor Peor Media** **0.577 0.249 0.468** **Desviación Estándar** + **0.103**
	K-Means	Mejor Peor Media 0.852 1.480 1.136 Desviación Estándar 0.119	Mejor Peor Media 0.278 0.142 0.180 Desviación Estándar * 0.031	Mejor Peor Media 0.484 0.294 0.360 Desviación Estándar * 0.044
	Clustering Aglomerativo	Resultado obtenido 1.157 Desviación Estándar * 0.0	**Resultado obtenido** **0.262** **Desviación Estándar** + **0.0**	Resultado obtenido 0.441 Desviación Estándar 0.0
Cáncer	SAGMDE	**Mejor Peor Media** **0.756 0.757 0.757** **Desviación Estándar** + **0.00007**	Mejor Peor Media 0.857 0.835 0.850 Desviación Estándar 0.010	Mejor Peor Media 0.765 0.740 0.757 Desviación Estándar 0.011
	K-Means	**Mejor Peor Media** **0.757 0.757 0.757** **Desviación Estándar** + **0.00008**	Mejor Peor Media 0.851 0.846 0.847 Desviación Estándar * 0.002	Mejor Peor Media 0.759 0.752 0.753 Desviación Estándar * 0.002
	Clustering Aglomerativo	Resultado obtenido 0.787 Desviación Estándar * 0.0	**Resultado obtenido** **0.869** **Desviación Estándar** + **0.0**	**Resultado obtenido** **0.792** **Desviación Estándar** + **0.0**

Tabla 5.4: SAGMDE vs métodos clásicos. Se reportan los resultados obtenidos en los indicadores ARI y completitud además de la puntuación en la función costo DB. Se han coloreado los resultados para destacarlos, en negritas y con un símbolo + en la esquina están los mejores resultados, los peores tienen un asterisco y están de rojo, esto basándonos en la puntuación *Media*.

Análisis de los resultados obtenidos

Como era de esperarse, *SAGMDE* logra mejores puntuaciones en DB al usarla como función objetivo, en ARI *clustering aglomerativo* supera por poco a *SAGMDE* y K-Means en los conjuntos Iris, Cáncer y Vidrio pero en Vino pierde por mucho ante ambos, siendo aquí el vencedor K-Means. En *completitud* se ven más divididos los resultados, pues *SAGMDE* tiene el mejor desempeño en los conjuntos Iris y Vidrio mientras que *clustering aglomerativo* es mejor en Vino y K-Means en Cáncer.

En general a *SAGMDE* le va bien en *completitud* pues es el mejor en Iris y Vidrio, mientras que en Vino y Cáncer queda en segundo lugar muy de cerca a los mejores resultados. En el indicador *ARI* tiene la peor puntuación en Iris, pero obtiene el segundo lugar en los conjuntos restantes, quedando muy cerca de las mejores puntuaciones, así que en general no le va mal.

De los resultados podemos concluir que *SAGMDE* queda cerca de una solución óptima global, pues cuando no obtiene la mejor puntuación queda muy cerca de ella, además que su desviación estándar es baja, lo cual indica que las soluciones que encuentra en cada corrida no difieren mucho entre sí.

En la sección apéndice se encuentra la figura 7.2, donde se muestran los tiempos de ejecución de esta implementación, concretamente en las gráficas e y f. También contiene información del entorno donde se ejecutaron las pruebas y la referencia a la liga del repositorio con los códigos implementados.

Capítulo 6

Comparación entre métodos

En este capítulo se llevará a cabo una comparación entre las metaheurísticas estudiadas en este trabajo: evolución diferencial (DE, por sus siglas en inglés), optimización por cúmulo de partículas (PSO, por sus siglas en inglés) y recocido simulado (SA, por sus siglas en inglés). Para tener una comparación justa, definiremos el costo computacional de los algoritmos como la cantidad de veces que llaman a la función que buscan optimizar, luego limitaremos cada método para que hagan la misma cantidad de llamadas a su función por optimizar. De esta forma, todos harán el mismo número de evaluaciones de la función objetivo sin importar su tamaño de población o número de iteraciones. Finalmente, se le indica a evolución diferencial cuántos grupos hay, para que se limite a optimizar su función y no a encontrar el número de grupos.

Los experimentos se realizaron utilizando los cuatro conjuntos de datos expuestos en la Sección 2.4 y consistieron en 15 ejecuciones independientes de cada algoritmo, donde cada algoritmo realizó a lo más 50 mil llamadas a su función objetivo. Las métricas utilizadas para medir su desempeño son *ARI* y *completitud*. La configuración de los algoritmos se describe a continuación.

ACDE

La función que optimiza es el índice CS, ver ecuación 2.5, y los parámetros usados son: $k_{max} = 20$, $Cr_{max} = 1$, $Cr_{min} = 0.5$, $N = 10$, $MaxEval = 50{,}000$. x_{min}, x_{max} y D dependen del conjunto de datos. En este caso, la variable $MaxEval$ limita la cantidad de llamadas a la función objetivo.

PSO-Kmeans

Optimiza el negativo de la función MSSE, ver ecuación 2.3, utilizando los siguientes parámetros: $GenSincambios = 10$, c_1 y $c_2 = 2$, $v_{min} = 0$, $v_{max} = 1$, $n_{part} = 50$, $iterPso = 800$, $iterKmeans = 200$, x_{min}. x_{max} y k dependen del conjunto de datos de entrada.

Para que este algoritmo haga a lo más 50 mil llamadas a su función objetivo, debemos calcular cuántas iteraciones tiene que hacer. En cada iteración realiza una llamada a la función objetivo por cada partícula. Dado que $n_{part} = 50$, necesitamos que haga 1000 iteraciones en total. Por lo anterior, establecemos $iter_{Pso} = 800$ y $iter_{Kmeans} = 200$, dando en total las 1000 iteraciones, tambien sería posible usar otras cantidades para las variables

siempre que sumen 1000, pero se decidió usar éstas ya que PSO tendrá más iteraciones para buscar la mejor zona global.

SAGMDE

Esta metaheurística minimiza la función SSE, ver ecuación 2.2, utilizando los siguientes parámetros: $Max_{iteraciones} = 2n$, siendo n el número de datos del conjunto a clasificar, $T_{inicial} = 0.0015$, $T_{final} = 0.000001$, $\alpha = 0.98$, $T_{distorsion} = 6$, $Tf_{distorsin} = 0.025$ y $\alpha_{distorsion} = 0.985$.

En este algoritmo la cantidad de llamadas a la función que minimiza depende del tamaño del conjunto de datos, pues en cada temperatura realiza $2n$ iteraciones. La forma en que controlamos cuántas llamadas realiza fue modificar el código y agregar un contador. Como realiza dos procesos de enfriamiento, le asignamos el 80 % de las llamadas al primer proceso y el 20 % al segundo, ya que en la primera etapa hace una búsqueda general y en la segunda un refinamiento.

6.1. Resultados obtenidos

La Tabla 6.1 muestra las puntuaciones de cada metaheurística en el conjunto de datos **Iris**. De la tabla podemos observar que PSO-Kmeans y SAGMDE obtienen la mejor puntuación en el indicador *ARI*, en todos los casos. Siendo mejor SAGMDE solo por tener una desviación estándar menor que PSO-Kmeans, pero no hay una diferencia significativa en los resultados. ACDE por su parte saca la mejor *completitud* a cambio de quedar más bajo en ARI respecto a las otras dos metaheurísticas.

	ACDE			**PSO-Kmeans**			**SAGMDE**		
ARI	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.563	0.558	0.563	0.730	0.716	0.722	0.730	0.716	0.717
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.001			0.006			0.003		
Completitud	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.920	0.854	0.915	0.764	0.747	0.755	0.764	0.747	0.748
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.016			0.008			0.004		

Tabla 6.1: **Conjunto de datos IRIS**. Análisis estadístico de los resultados obtenidos en los indicadores ARI y completitud, después de 15 ejecuciones de cada algoritmo. Se reporta la media, mejor y peor solución así como la desviación estándar en las puntuaciones.

La Tabla 6.2 muestra los resultados obtenidos en el conjunto de datos **Vino**. Aquí podemos observar que a ACDE no le va muy bien en *ARI*, pues llega incluso a puntuaciones negativas, en *completitud* se quedó bastante atrás de PSO-Kmeans y SAGMDE, los cuales nuevamente empatan y obtienen las mejores puntuaciones en todos los casos.

	ACDE			PSO-Kmeans			SAGMDE		
ARI	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.001	-0.003	-0.0003	0.371	0.371	0.371	0.371	0.371	0.371
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.002			0.0			0.0		
Completitud	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.232	0.185	0.209	0.428	0.428	0.428	0.428	0.428	0.428
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.020			0.0			0.0		

Tabla 6.2: **Conjunto de datos VINO**. Análisis estadístico de los resultados obtenidos en los indicadores ARI y completitud, después de 15 ejecuciones de cada algoritmo. Se reporta la media, mejor y peor solución así como la desviación estándar en las puntuaciones.

Las cosas cambian un poco en el conjunto de datos **Vidrio**, pues como podemos apreciar en la Tabla 6.3, SAGMDE supera a PSO-Kmeans en *ARI*, su mejor solución obtiene el mismo valor pero lo supera en los otros casos. En *completitud*, SAGMDE supera a PSO-Kmeans en todos los escenarios. ACDE muestra nuevamente debilidad en el indicador *ARI* y en *completitud* logra obtener la puntuación más alta en el mejor resultado, pero su media está bastante debajo de lo que obtienen PSO-Kmeans y SAGMDE.

	ACDE			PSO-Kmeans			SAGMDE		
ARI	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.249	0.005	0.005	0.273	0.205	0.266	0.273	0.261	0.269
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.060			0.016			0.0		
Completitud	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.574	0.345	0.345	0.468	0.332	0.434	0.470	0.436	0.463
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.057			0.032			0.010		

Tabla 6.3: **Conjunto de datos VIDRIO**. Análisis estadístico de los resultados obtenidos en los indicadores ARI y completitud, después de 15 ejecuciones de cada algoritmo. Se reporta la media, mejor y peor solución así como la desviación estándar en las puntuaciones.

Finalmente, veamos los resultados del conjunto de datos Cáncer en la Tabla 6.4. Aquí ACDE obtiene la puntuación más alta en el mejor resultado, pero su peor resultado es bastante malo, esto ocurre en ambos indicadores *ARI* y *completitud*. PSO-Kmeans pasó de empatar con SAGMDE en dos conjuntos de datos y perder en uno a superarlo en todos los escenarios en ambos indicadores.

Conjunto de datos cáncer

	ACDE			PSO-Kmeans			SAGMDE		
ARI	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.902	0.005	0.716	0.851	0.847	0.846	0.824	0.819	0.820
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.355			0.002			0.002		
Completitud	Mejor	Peor	Media	Mejor	Peor	Media	Mejor	Peor	Media
	0.825	0.154	0.684	0.759	0.752	0.754	0.729	0.723	0.724
	Desviación estándar			Desviación estándar			Desviación estándar		
	0.265			0.002			0.002		

Tabla 6.4: **Conjunto de datos CÁNCER**. Análisis estadístico de los resultados obtenidos en los indicadores ARI y completitud, después de 15 ejecuciones de cada algoritmo. Se reporta la media, mejor y peor solución así como la desviación estándar en las puntuaciones.

En la figura 6.1, podemos observar de forma gráfica la puntuación en *ARI* que obtiene cada metaheurística en los cuatro conjuntos de datos. La gráfica se construyó tomando los mejores resultados en *ARI* de las tablas 6.1,6.2,6.3 y 6.4.

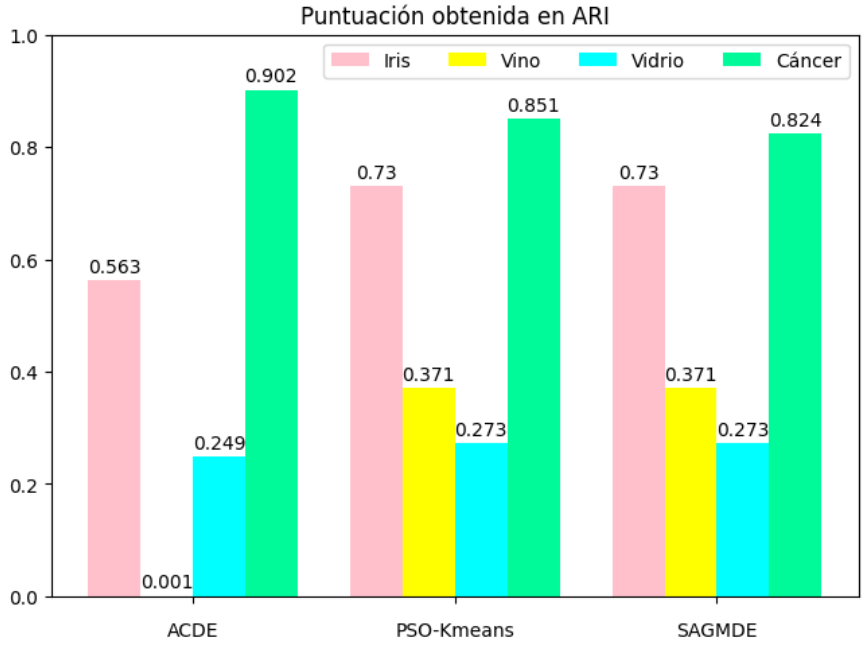


Figura 6.1: Gráfica con los resultados obtenidos en la puntuación ARI, considerando el mejor resultado de los algoritmos.

Por su parte, la figura 6.2 muestra la puntuación en *Completitud* que obtiene cada metaheurística en los cuatro conjuntos de datos. De igual manera se construyó tomando los mejores resultados en *ARI* de las tablas 6.1,6.2,6.3 y 6.4.

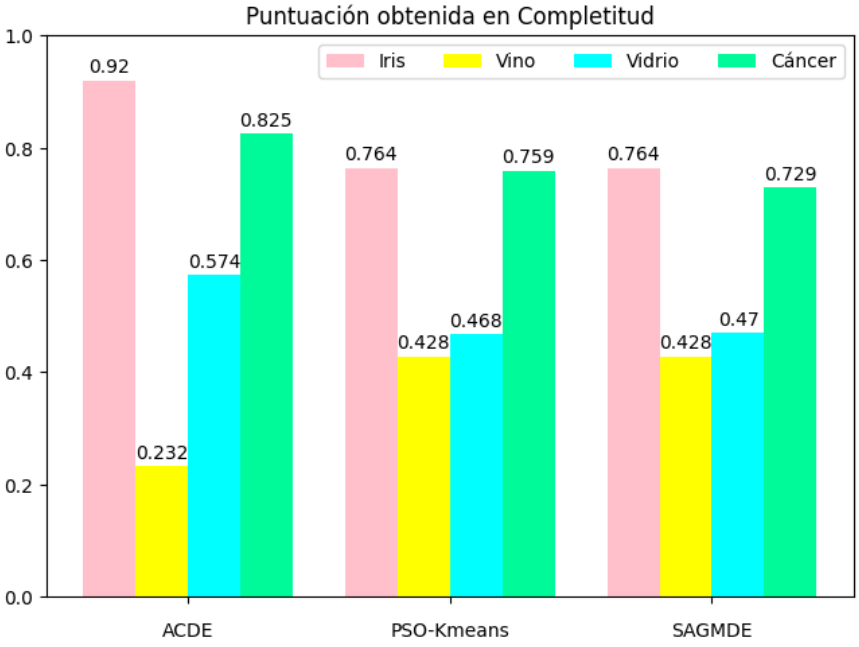


Figura 6.2: Gráfica con los resultados obtenidos en la puntuación Completitud, considerando el mejor resultado de los algoritmos.

Capítulo 7

Conclusiones y trabajo futuro

En este trabajo de tesis estudiamos tres metaheurísticas que se han utilizado en tareas de *clustering*. La primera de ellas está basada en el algoritmo evolutivo llamado *Evolución Diferencial* (DE, por sus siglas en inglés), la segunda en *optimización por cúmulo de partículas* (PSO, por sus siglas en inglés) y la tercera en *Recocido Simulado* (SA, por sus siglas en inglés). Todas ellas se implementaron buscando replicar los resultados reportados por sus autores y se probaron en cuatro conjuntos de datos de prueba: Iris, Vino, Vidrio y Cáncer. A partir de los resultados experimentales podemos concluir lo siguiente.

En los cuatro conjuntos de prueba, las tres metaheurísticas estudiadas superaron a los métodos clásicos conocidos como *K-Means* y *clustering aglomerativo*, cuando se evaluaron utilizando las métricas que optimizan durante su proceso de búsqueda. Esto tiene sentido porque se especializan en obtener buenos resultados en la función objetivo que están optimizando. Sin embargo, no pasa lo mismo cuando se considera el correcto etiquetado de los datos. Esto se debe a las características de los conjuntos de datos. Por ejemplo, si la métrica optimizada busca generar grupos bien definidos y separados pero el conjunto de prueba tiene datos pertenecientes a diferentes grupos que comparten áreas del espacio, los resultados al considerar el etiquetado correcto no van a ser los mejores. Para lidiar con este tipo de conjuntos de datos, es necesario cambiar la función objetivo de tal forma que considere dichas intersecciones en las áreas.

A continuación se describen de manera detallada nuestras observaciones en cada metaheurística estudiada.

ACDE

ACDE es el único algoritmo de los 3 que busca la cantidad de grupos existentes en los conjuntos de datos, a la vez que minimiza una función objetivo, nuestra implementación logró esta tarea en los conjuntos de prueba. Ofrece resultados estables en todas las métricas utilizadas y queda muy cerca de los valores que reportan sus autores.

Sin importar si utilizamos *CS* o *DB* como función objetivo no da buenos resultados en el indicador *ARI*. Solamente en el conjunto de datos Cáncer obtiene una puntuación sobresaliente, en Iris es regular y en los otros dos conjuntos obtiene malos resultados. Pero el escenario cambia en el indicador *completitud*, pues, aunque nuevamente en Iris y Cáncer es donde sobresale, en los otros dos se desempeña mucho mejor que en *ARI*.

Basándonos en los resultados podemos decir que a esta metaheurística no le va muy bien en conjuntos de datos con una dimensión muy alta, pues en el conjunto Vino de dimensión 13 obtiene las peores puntuaciones tanto en *ARI* como en *completitud*, pero maneja bien conjuntos de datos de hasta 9 dimensiones. Aunque otra posible causa del bajo desempeño que tiene este conjunto de datos puede ser el traslape entre grupos, pues si observamos la gráfica de su representación en 3d ver fig. 2.2 notaremos que el grupo de azul y el de rojo están muy cerca entre sí, mientras que el grupo de verde esta más separado de ambos.

Por lo anterior, es recomendable usar esta metaheurística si se busca la mayor *completitud* en el etiquetado de datos con una dimensión igual o menor a 9, ya que es donde presenta su mejor rendimiento, y supera considerablemente a las otras dos metaheurísticas.

PSO-Kmeans

Este algoritmo es bastante sencillo tanto de comprender como de implementar. Los resultados obtenidos por nuestra implementación en ocasiones son incluso mejores a los reportados por los autores Alireza Ahmadyfard y Hamidreza Modares. Los resultados son estables en todas las métricas utilizadas y se desempeña bastante bien en cualquier escenario. El cambiar su función objetivo entre la que proponen Alireza Ahmadyfard y Hamidreza Modares y el índice DB no representa un gran cambio en las puntuaciones obtenidas en *ARI* y *completitud*. En la mayoría de los conjuntos de datos tiene las mejores puntuaciones en *ARI* empatando con *SAGMDE* en Vino, siendo Vidrio el único escenario donde obtiene peores resultados. En la métrica de *completitud* tiene el segundo lugar en desempeño, pues supera a *SAGMDE* en los conjuntos Iris y Cáncer, empatan en Vino y solo pierde en Vidrio.

Basándonos en nuestros experimentos podemos concluir que esta metaheurística tiene un desempeño aceptable en *completitud* en cualquier conjunto de datos, y en *ARI* ofrece de los mejores resultados, solo podría presentar algunas complicaciones en conjuntos de datos con una dimensión alta, pues en Vidrio su peor solución y mediana son levemente inferiores a los de *SA*.

Por lo anterior, podemos recomendar PSO-Kmeans cuando se busque la mejor puntuación de *ARI*, solo hay que considerar que en dimensiones iguales o superiores a 13 podría no presentar los mejores resultados. Si se busca *completitud* ofrecerá resultados no muy buenos, por lo que no sería muy apropiado.

SAGMDE

Nuestra implementación queda muy cerca de los resultados reportados por los autores Julian Lee y David Perkins, incluso superándolos en el conjunto Cáncer. Siempre obtiene una desviación estándar menor que los métodos clásicos en cualquier métrica y conjunto de datos por lo que podemos decir que es muy estable.

En esta metaheurística se observa mayor cambio en los resultados al utilizar diferentes funciones objetivo. Usando DB obtiene mejores resultados que con SSE pues se mejoran las puntuaciones en *ARI* en los conjuntos Vino, Vidrio y Cáncer, además obtiene mejor

completitud en todos los conjuntos de datos, esto a costa de una desviación estándar mayor respecto a la obtenida usando SSE, pero en ningún caso superior a 0.1.

Observando sus resultados destaca que junto a *PSO-Kmeans* ofrecen los mejores resultados en *ARI* en cualquier escenario, pues aunque queda por detrás en Iris y Cáncer solo en éste último conjunto hay una diferencia significativa en las puntuaciones, ya que en Iris únicamente pierde por tener una *media* ligeramente inferior a la de *PSO-Kmeans*. En *completitud* nuevamente pierde ante *PSO-Kmeans* en dos conjuntos de datos, en Iris por tener una *media* un poco menor, y en Cáncer por que le fue peor en todos los resultados. Este algoritmo parece ser el que menos sufre en escenarios de datos con dimensión alta, pues en el conjunto Vidrio superó a *PSO-Kmeans* tanto en *ARI* como *completitud*, siendo que estas dos metaheurísticas son las de mejor desempeño en general en *ARI*.

SAGMDE es recomendable si se busca obtener buenos resultados en *ARI* y si se tienen conjuntos de datos de dimensión alta.

7.1. Trabajo futuro

Un posible trabajo futuro sería estudiar diferentes aplicaciones de la vida real relacionadas con el problema de *clustering* e identificar escenarios donde las metaheurísticas estudiadas puedan ser aplicadas de manera exitosa. Por ejemplo, para hacer más eficiente el transporte público de una ciudad a partir de la minimización de las distancias que deben recorrer los ciudadanos para abordar algún transporte público y del agrupamiento de las rutas que siguen diariamente para trasladarse a sus casas, trabajos, escuelas, centros comerciales, etc.

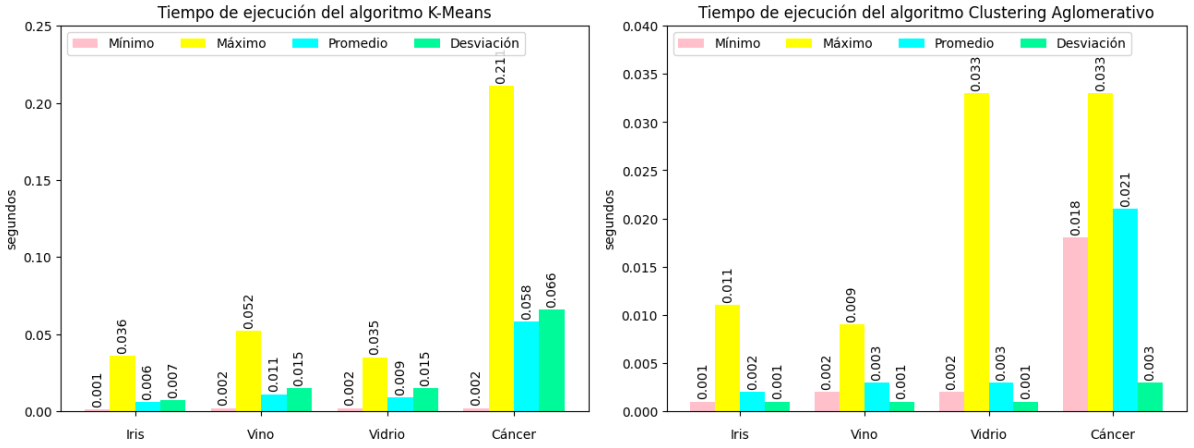
En este trabajo se buscó replicar los resultados de los trabajos originales, por lo que no se analizó el comportamiento de las implementaciones con diferentes valores en sus parámetros. Sería entonces interesante comprobar la importancia que tienen los parámetros utilizados en las implementaciones, con el fin de determinar su impacto en los resultados obtenidos. También se pueden realizar más pruebas en diferentes conjuntos de datos, con la finalidad de obtener un cuadro más general de su desempeño.

Apéndice

Las librerías tomadas de `scikit-learn` [20] fueron utilizadas principalmente para el cálculo de valores como: las medidas de validación ARI y Completitud, distancias euclidianas y el valor que adquieren las soluciones en el índice DB.

Todos los códigos fueron programados en Python 3, por lo cual no son tan rápidos como las implementaciones replicadas, pues éstas venían en C. Se utilizó la página de Google Colab en su versión gratuita para el desarrollo del trabajo, por lo que se puede ejecutar desde cualquier dispositivo capaz de abrir dicha pagina web. La máquina virtual de Google Colab sólo nos proporciona las siguientes características: 12.7 GB de RAM y 107.7 GB de disco duro, del cual tenemos libres 80.8 GB, lamentablemente no se proporciona información sobre el procesador, pero si sabemos que no se necesita una GPU para correr las implementaciones.

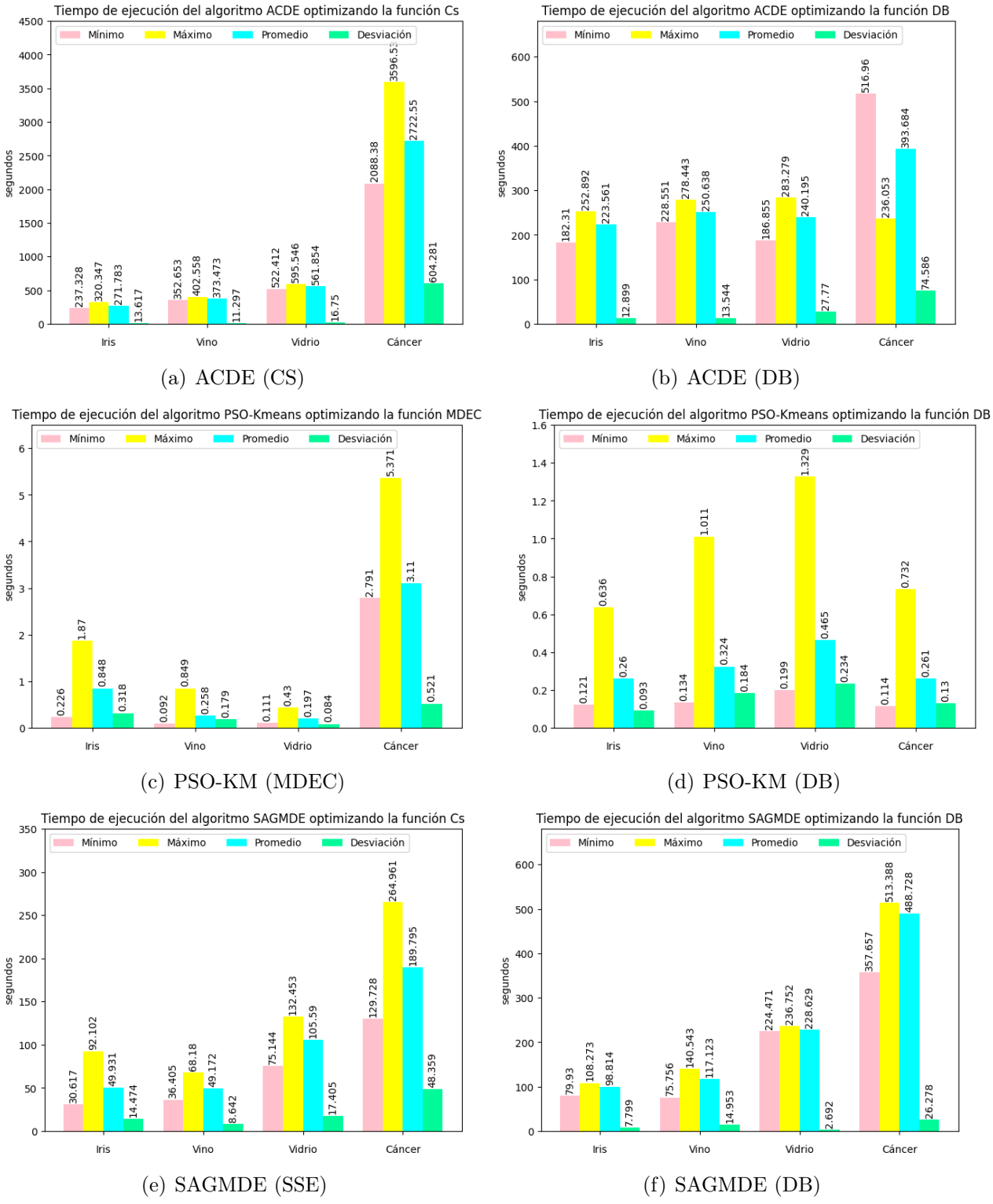
Adjuntamos 6 gráficas con el tiempo de ejecución aproximado de las implementaciones, y otras 2 con el tiempo de los métodos clásicos. En el repositorio [22] se encuentran los códigos implementados y las instrucciones para replicar los resultados reportados en este trabajo de tesis.



(a) K-Means

(b) Clustering Aglomerativo

Figura 7.1: Análisis estadístico del tiempo de ejecución de los métodos clásicos K-Means y clustering glomerativo, se muestra el tiempo mínimo, máximo y promedio, así como su desviación estándar. Se reporta el tiempo de los métodos en cada uno de los conjuntos de datos utilizados para medir su desempeño.



(a) ACDE (CS)

(b) ACDE (DB)

(c) PSO-KM (MDEC)

(d) PSO-KM (DB)

(e) SAGMDE (SSE)

(f) SAGMDE (DB)

Figura 7.2: Análisis estadístico del tiempo de ejecución de las implementaciones hechas en este trabajo, se muestra el tiempo mínimo, máximo y promedio, así como su desviación estándar. Cada gráfica lleva el nombre del algoritmo evaluado y entre paréntesis la función que está optimizando. Se reporta el tiempo de las implementaciones en cada uno de los conjuntos de datos utilizados para medir su desempeño.

Referencias

[1] EHL Aarts y PJM Van Laarhoven. «Simulated annealing: an introduction». En: *Statistica Neerlandica* 43.1 (1989), págs. 31-52.

[2] Alireza Ahmadyfard y Hamidreza Modares. «Combining PSO and k-means to enhance data clustering». En: *2008 International Symposium on Telecommunications*. 2008, págs. 688-691. DOI: 10.1109/ISTEL.2008.4651388.

[3] Mihael Ankerst y col. «OPTICS: Ordering Points to Identify the Clustering Structure». En: *SIGMOD Rec.* 28.2 (jun. de 1999), págs. 49-60. ISSN: 0163-5808. DOI: 10.1145/304181.304187. URL: https://doi.org/10.1145/304181.304187.

[4] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer, Boston, MA, 1981.

[5] C-H Chou, M-C Su y Eugene Lai. «A new cluster validity measure and its application to image compression». En: *Pattern Analysis and Applications* 7.2 (2004), págs. 205-220.

[6] Swagatam Das, Ajith Abraham y Amit Konar. «Automatic clustering using an improved differential evolution algorithm». En: *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans* 38.1 (2007), págs. 218-237.

[7] David L Davies y Donald W Bouldin. «A cluster separation measure». En: *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), págs. 224-227.

[8] Doulaye Dembélé y Philippe Kastner. «Fuzzy C-means method for clustering microarray data». En: *Bioinformatics* 19.8 (mayo de 2003), 973-980, 3. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg119. URL: https://doi.org/10.1093/bioinformatics/btg119.

[9] Martin Ester y col. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». En: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, págs. 226-231.

[10] Vishwanath R Iyer y col. «The transcriptional program in the response of human fibroblasts to serum». En: *science* 283.5398 (1999), págs. 83-87.

[11] J. Kennedy y R. Eberhart. «Particle swarm optimization». En: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942-1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[12] James Kennedy y Russell Eberhart. «Particle swarm optimization». En: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, págs. 1942-1948.

[13] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi. «Optimization by Simulated Annealing». En: *Science* 220.4598 (1983), págs. 671-680. ISSN: 0036-8075. DOI: 10.1126/science.220.4598.671.

[14] R. Krishnapuram y J. M. Keller. «A possibilistic approach to clustering». En: *IEEE Transactions on Fuzzy Systems* 1.2 (1993), págs. 98-110. DOI: 10.1109/91.227387.

[15] Julian Lee y David Perkins. «A simulated annealing algorithm with a dual perturbation method for clustering». En: *Pattern Recognition* 112 (2021), pág. 107713.

[16] Robert J Lipshutz y col. «High density synthetic oligonucleotide arrays». En: *Nature genetics* 21.1 (1999), págs. 20-24.

[17] Nicholas Metropolis y col. «Equation of State Calculations by Fast Computing Machines». En: *Chemical Physics* 21.6 (1953), págs. 1087-1092. DOI: https://doi.org/10.1063/1.1699114.

[18] Yves Moreau y col. «Functional bioinformatics of microarray data: from expression to regulation». En: *Proceedings of the IEEE* 90.11 (2002), págs. 1722-1743.

[19] Hossam MJ Mustafa y col. «Multi-objective memetic differential evolution optimization algorithm for text clustering problems». En: *Neural Computing and Applications* (2022), págs. 1-21.

[20] F. Pedregosa y col. «Scikit-learn: Machine Learning in Python». En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[21] Andrés Ramos y col. «Modelos matemáticos de optimización». En: *Publicación Técnica* 1 (2010).

[22] Saúl. *ImplementacionesTesis*. Ver. 1.0. 2023. URL: https://github.com/luasikirf/ImplementacionesTesis.

[23] Mark Schena y col. «Quantitative monitoring of gene expression patterns with a complementary DNA microarray». En: *Science* 270.5235 (1995), págs. 467-470.

[24] Shokri Z Selim y K1 Alsultan. «A simulated annealing algorithm for the clustering problem». En: *Pattern recognition* 24.10 (1991), págs. 1003-1008.

[25] Paul T Spellman y col. «Comprehensive identification of cell cycle–regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization». En: *Molecular biology of the cell* 9.12 (1998), págs. 3273-3297.

[26] Rainer Storn y Kenneth Price. «Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces». En: *Journal of Global Optimization* 11.4 (1997), págs. 341-359. DOI: 10.1023/A:1008202821328.

[27] D.H. Wolpert y W.G. Macready. «No free lunch theorems for optimization». En: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), págs. 67-82. DOI: 10.1109/4235.585893.

[28] Rui Xu y Donald C Wunsch. «Clustering algorithms in biomedical research: a review». En: *IEEE reviews in biomedical engineering* 3 (2010), págs. 120-154.

[29] Zhan Yu. «Fe Cluster Structure Optimization Based on Small Habitat Differential Evolution Algorithm». En: *International Conference on Innovative Computing*. Springer. 2022, págs. 783-789.

[30] Xiaoling Zhu y col. «Multiobjective Differential Evolution Clustering Algorithm Based on Multimedia Information in Working Mechanism Innovation for Traditional Chinese Medicine». En: *Advances in Multimedia* 2022 (2022).