

Cd. Victoria, Tamaulipas, 03 de diciembre de 2024

INFORME FINAL DE RESULTADOS DEL PROYECTO:

Diseño y ejecución de Pruebas para evaluar el desempeño de la una Herramienta de Segmentación de Imágenes Médicas

Estancia en el consultorio:

SANTA SOFÍA

Elaboró:

Ernesto Gustavo Pérez Estrada

Bajo la dirección de:

Dra. Adriana Mexicano Santoyo

Ciudad Victoria, Tamaulipas

Diciembre 2024

INTEGRANTES DEL PROYECTO:



ING. ERNESTO GUSTAVO PÉREZ ESTRADA
Estudiante de la Maestría en Sistemas
Instituto Tecnológico de Cd. Victoria



DRA. ADRIANA MEXICANO SANTOYO
Directora de Tesis
Instituto Tecnológico de Cd. Victoria



DR. PASCUAL NORADINO MONTES DORANTES
Codirector de tesis
Instituto Tecnológico de Cd. Victoria



DR. JESÚS CARLOS CARMONA FRAUSTO
Asesor de tesis
Instituto Tecnológico de Cd. Victoria

INSTITUTO TECNOLÓGICO
DE CD. VICTORIA
DEPARTAMENTO DE
DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN



DRA. LILIA DEL CARMEN GARCÍA MUNDO
Asesora de tesis
Instituto Tecnológico de Cd. Victoria

Recibe y acepta



DR. EMANUEL ABISAI LUNA NOLASCO
Gerente del Consultorio Médico Santa Sofía



Índice

1. Introducción	4
2. Actividades realizadas durante la estancia	5
2.1. Búsqueda de oportunidades de mejora del algoritmo FCM	5
2.2. Estudio y análisis del comportamiento del algoritmo FCM	6
2.3. Definición la propuesta del nuevo algoritmo: FLCM++	8
2.4. Selección y validación del umbral δ	12
3. Experimentación para la validación de los Resultados	17
3.1. Entorno de pruebas	17
3.2. Realización de pruebas con distintos conjuntos de datos	18
3.3. Pruebas realizadas de segmentación de imágenes	29
4. Resultados obtenidos	35
4.1. Resultados de Conjuntos de Datos Numéricos	35
4.2. Segmentación de Imágenes	36
4.3. Conclusiones Generales	37

Índice de figuras

1.	Diagrama de flujo del algoritmo FLCM++	10
2.	La suma de los cuadrados de las distancias al cuadrado de cada prueba con respecto a los valores correspondientes de δ . 2a) muestra una ampliación de la figura 2 en la cual se puede observar que mientras mayor sea δ existe mayor pérdida en la calidad del agrupamiento	16
3.	Tiempo promedio que tomó cada una de las pruebas, en función de los diferentes valores de δ	16
4.	Resultados de FCM vs FLCM++ en relación con el tiempo de ejecución.	21
5.	Resultados de FCM vs FLCM++ en relación con la suma de las distancias al cuadrado.	22
6.	Resultados de FCM++ vs FLCM++ en relación con el tiempo de ejecución.	25
7.	Resultados de FCM++ vs FLCM++ en relación con la sumatoria de las distancias al cuadrado.	26
8.	Resultados de HOFM vs FLCM++ en relación con el tiempo de ejecución.	27
9.	Resultados de HOFM vs FLCM++ en relación con la sumatoria de las distancias al cuadrado.	29
10.	Imágenes de tomografías computarizadas de un paciente, empleadas en la experimentación de segmentación de imágenes.	30
11.	Resultados al segmentar resonancias magnéticas aplicando el algoritmo FLCM++, empleando un valor de $c = 4$	31

Índice de tablas

1.	Resultados de la ejecución de la instancia ecoli con el algoritmo FCM con los parámetros $\varepsilon = 0.01$, $m = 2$ y $c = 4$	6
2.	Resultados de la ejecución de la instancia WDBC con el algoritmo FCM con los parámetros $\varepsilon = 0.01$, $m = 2$ y $c = 4$	7
3.	Pruebas realizadas para la selección del valor de δ : Tiempo y Sumatoria de Distancias al Cuadrado (Rango $\delta = 0.1, 0.2, \dots, 0.5$)	14
4.	Pruebas realizadas para la selección del valor de δ : Tiempo y Sumatoria de Distancias al Cuadrado (Rango $\delta = 0.6, 0.7, \dots, 1.0$)	15
5.	Características de los conjuntos de datos	18
6.	Comparación entre FCM y FLCM++: Tiempo y Sumatoria de Distancias al Cuadrado	20
7.	Comparación entre FCM++ y FLCM++: Tiempo y Sumatoria de Distancias al Cuadrado	24
8.	Comparación entre HOFCM y FLCM++. Tiempo y Sumatoria de Distancias al Cuadrado	28
9.	Pruebas realizadas con el algoritmo FLCM++ para la segmentación de las imágenes comprendidas entre Imagen0002 e Imagen0008	33
10.	Pruebas realizadas con el algoritmo FLCM++ para la segmentación de las imágenes comprendidas entre Imagen0009 e Imagen0016	34

1. Introducción

El agrupamiento de datos es una técnica esencial en el aprendizaje no supervisado [1], su característica distintiva radica en utilizarlo en conjuntos de datos que no poseen etiquetas, permitiendo descubrir patrones entre ellos para determinar grupos específicos. Esta práctica es ampliamente utilizada en diversas áreas, como el reconocimiento de patrones [2–4], el procesamiento de imágenes [5, 6], los negocios para la segmentación de mercados [7], la minería de datos [8], entre otros. El agrupamiento implica la partición de un conjunto de n objetos en subconjuntos llamados grupos. En este proceso, los objetos dentro de un grupo comparten características similares entre sí y al mismo tiempo son distintos de los objetos en otros grupos. De acuerdo con la literatura especializada, a lo largo del tiempo se han desarrollado varios algoritmos de agrupamiento, categorizándose como basados en centroide [9], jerárquicos [10, 11], basados en densidad [12, 13], entre otros.

El agrupamiento basado en centroides se divide en dos tipos: el agrupamiento duro (clásico) y el agrupamiento suave (o difuso). El agrupamiento duro [2] asigna cada elemento a un único grupo sin permitir solapamientos [14], mientras que el suave asigna grados de pertenencia a múltiples grupos para cada elemento, permitiendo pertenencia parcial a varios grupos [15]. Aunque el agrupamiento duro es directo, el suave es flexible y refleja la incertidumbre en la pertenencia a grupos, siendo útil cuando los límites entre grupos no están definidos claramente. Los datos en el mundo físico rara vez están ordenados en grupos definidos, presentando límites difusos y superposiciones [16]. Esto se debe a que los datos no tienen claridad, son: dudosos, ambiguos, vulnerables al cambio y tienen parámetros poco confiables. En tales situaciones, el agrupamiento suave proporciona una mejor representación de la realidad.

Dentro de los métodos de agrupamiento suave destaca el algoritmo FCM [17], integrante de los algoritmos de lógica difusa. No obstante, enfrenta desafíos como la sensibilidad a la inicialización, lo que puede conducir a converger en mínimos locales no deseados [18]. Además, presenta un alto costo computacional, donde diversos trabajos [19], [20], [21] lo denotan por la Ecuación ??, indicando el orden de crecimiento del tiempo de ejecución en función de la entrada del algoritmo, la complejidad indica cómo el tiempo de ejecución aumenta en función del tamaño de la entrada del algoritmo. Estas dificultades destacan la necesidad de desarrollar mejoras en el algoritmo.

El agrupamiento ha adquirido una relevancia significativa en respuesta al crecimiento exponencial de datos en los últimos años, como lo señalan estudios recientes [22, 23]. Esta tendencia ha posicionado al agrupamiento como un enfoque de estudio fundamental en diversas disciplinas. Sin embargo, enfrenta un desafío considerable al ser clasificado como un problema NP-duro [24]. Esta clasificación implica que no existen algoritmos eficientes conocidos que puedan resolver el problema en un tiempo polinomial, lo que presenta una limitación importante en su aplicación práctica.

Además, la segmentación de imágenes es una aplicación fundamental del agrupamiento y una técnica clave en el aprendizaje no supervisado, debido a que permite agrupar píxeles en regiones con características similares. Sin embargo, el procesamiento de imágenes, especialmente en conjuntos de gran tamaño y alta resolución, conlleva un alto costo computacional. Para abordar este problema, se ha propuesto una mejora basada en el análisis del algoritmo FCM. Este algoritmo asigna a cada elemento un grado de pertenencia a cada grupo, recalculando iterativamente estos grados y los centroides hasta alcanzar la convergencia. No obstante, los resultados obtenidos no justifican el alto costo computacional.

En este trabajo se realiza el diseño y ejecución de pruebas para evaluar el desempeño de una herramienta de segmentación de imágenes médicas basada en el algoritmo FCM (Fuzzy C-Means). Para

optimizar el proceso, se implementa una heurística que reduce el tiempo de ejecución del algoritmo al detener el cálculo de los grados de pertenencia cuando estos no presentan cambios significativos en la asignación de grupos. Adicionalmente, se desarrolla una interfaz gráfica que facilita la aplicación del algoritmo FCM mejorado, permitiendo a los usuarios cargar conjuntos de datos médicos, ejecutar pruebas de segmentación y visualizar los resultados de manera gráfica. Esta propuesta busca no solo mejorar la eficiencia computacional, sino también proporcionar una herramienta accesible y eficaz para la segmentación de imágenes médicas.

2. Actividades realizadas durante la estancia

De acuerdo con la literatura especializada, el algoritmo FCM se considera un problema NP-duro. Por esta razón, es común emplear técnicas heurísticas para encontrar soluciones alternativas, estas técnicas son ampliamente utilizadas para abordar problemas de esta complejidad [25, 26].

Este capítulo está estructurado de la siguiente forma: en la Sección 3.1 se describen las oportunidades de mejora del algoritmo FCM. En la Sección 3.2 se presenta un análisis general del comportamiento del algoritmo FCM. La Sección 3.3 se introduce la propuesta del nuevo algoritmo FLCM++. La Sección 3.4 describe la selección y validación del umbral δ .

2.1. Búsqueda de oportunidades de mejora del algoritmo FCM

El algoritmo de FCM es ampliamente utilizado en tareas de agrupamiento debido a su capacidad para asignar grados de pertenencia a cada dato, lo que permite una representación flexible de las relaciones entre los objetos. Sin embargo, se han identificado varias limitaciones y áreas de mejora en su implementación estándar.

En primer lugar, el tiempo de ejecución del algoritmo FCM es una preocupación significativa, especialmente cuando se trabaja con grandes conjuntos de datos. Este tiempo de ejecución se debe principalmente a la necesidad de recalcularse los grados de pertenencia en cada iteración, lo que puede ser computacionalmente costoso. Además, el algoritmo puede ser sensible a la inicialización de los centroides, lo que puede llevar a una convergencia prematura a óptimos locales, impidiendo una adecuada solución global.

Otra área de mejora identificada es la falta de una estrategia eficiente para manejar los objetos cuya asignación de grupo no cambia entre iteraciones. El tratamiento de estos objetos inalterados produce la sobreutilización de recursos utilizados para la realización de cálculos que no son necesarios ya que no contribuyen a la mejora del sistema y se convierten en un derroche.

En respuesta a estas observaciones, se propone una serie de mejoras que abordan estos desafíos, como la reducción del tiempo de cálculo mediante la identificación y exclusión de objetos inalterados, así como la adopción de un enfoque híbrido para la inicialización de centroides, que mejora la convergencia y mitiga los problemas de óptimos locales.

Tabla 1. Resultados de la ejecución de la instancia ecoli con el algoritmo FCM con los parámetros $\varepsilon = 0.01$, $m = 2$ y $c = 4$

Iteración	Función Objetivo	Criterio de paro ε	Cambios de grupo
1	191.381784	1.584222	0
2	52.218070	1.068415	42
3	47.058314	0.422137	3
4	44.657719	0.263003	1
5	44.143179	0.120846	1
6	44.063088	0.053555	1
7	44.049895	0.027896	2
8	44.047472	0.013306	1
9	44.047000	0.006131	0
10	44.046906	0.002785	0
11	44.046887	0.001258	0
12	44.046883	0.000567	0
13	44.046882	0.000255	0
14	44.046882	0.000115	0
15	44.046882	0.000052	0
16	44.046882	0.000023	0
17	44.046882	0.000011	0
18	44.046882	0.000005	0

2.2. Estudio y análisis del comportamiento del algoritmo FCM

El algoritmo FCM es un método de agrupamiento que asigna a cada elemento un grado de pertenencia hacia cada grupo. Este proceso iterativo implica calcular los grados de pertenencia y los centroides de los grupos en cada iteración hasta alcanzar la convergencia, lo que resulta en una cantidad considerable de cálculos en cada paso. Sin embargo, los resultados obtenidos no justifican los altos y costosos cálculos realizados por el algoritmo FCM. Con la finalidad de analizar al algoritmo FCM se realizaron pruebas con una instancia pequeña Ecoli [27] y una de mayor tamaño WDBC [28] obtenidas de UCI Repositorio de aprendizaje automático [29].

Prueba con la instancia Ecoli

En esta ejecución, se consideraron los siguientes parámetros: $\varepsilon = 0.01$, $m = 2$ y $c = 4$. Los resultados de la prueba se presentan en la Tabla 1, donde se registran el número de iteración, el valor de la función objetivo (Ecuación ??), el criterio de parada y el número de objetos que cambiaron de grupo en cada iteración.

En esta prueba se realizaron dieciocho iteraciones, debido a que se trata de una instancia relativamente pequeña. A partir de la iteración nueve, se observa que ningún objeto cambia de grupo, lo que indica la estabilización en la asignación de los grupos. Asimismo, los valores de la función objetivo muestran que desde la iteración seis, el cambio entre iteraciones es muy pequeño, con una variación insignificante entre iteraciones posteriores.

De los resultados obtenidos en la ejecución del algoritmo Fuzzy C-Means (FCM) para la instancia ecoli, se pueden hacer diversas observaciones importantes. A continuación, se describen las deducciones clave derivadas de los datos presentados en la Tabla 1.

En primer lugar, a partir de la iteración nueve, se observa que ningún objeto cambia de grupo. Esto indica que el algoritmo ha alcanzado un punto de estabilidad en el cual, las asignaciones de los objetos a los grupos ya varían. Aunque el proceso de iteración continúa hasta la iteración dieciocho, las asignaciones de grupos permanecen inalteradas, lo que sugiere que la estabilización del algoritmo ocurre a partir de la iteración nueve.

En segundo lugar, el valor de la función objetivo muestra una convergencia rápida. En las primeras iteraciones, se observan cambios significativos en los valores de la función objetivo, especialmente entre las iteraciones uno y dos, donde la función objetivo disminuye de 191.38 a 52.21. Sin embargo, a partir de la iteración seis, los cambios en la función objetivo son mucho más pequeños (44.06 en la iteración seis frente a 44.04 en la iteración siete), lo que indica una estabilización del proceso. Desde la iteración nueve, el valor de la función objetivo prácticamente no cambia, manteniéndose en torno a 44.04.

Además, el criterio de paro ε también muestra una disminución rápida. Al inicio, los valores de ε son considerables, pero después de la iteración seis, los cambios en este criterio son extremadamente pequeños. En la iteración nueve, el valor de ε es tan bajo como 0.006131, y sigue disminuyendo en las iteraciones posteriores hasta alcanzar 0.000005 en la iteración 18. Esto indica que el algoritmo está convergiendo hacia una solución con un refinamiento mínimo en las iteraciones finales.

Finalmente, dado que no se observan cambios significativos en la asignación de grupos ni en los valores de la función objetivo después de la iteración nueve, es posible concluir que el algoritmo podría haberse detenido en esa iteración sin afectar el resultado final. Las iteraciones adicionales (de la 10 a la 18) no aportan cambios relevantes, lo que sugiere que el tiempo computacional podría haberse optimizado interrumpiendo la ejecución desde la iteración 10.

Prueba con la instancia WDBC En esta ejecución, se consideraron los siguientes parámetros: $\varepsilon = 0.01$, $m = 2$ y $c = 4$. Los resultados de la prueba están registrados en la Tabla 2, el análisis de los datos obtenidos muestra el comportamiento de la función objetivo a lo largo de 193 iteraciones. Se puede observar una reducción progresiva en el valor de la función objetivo. Al inicio, en la iteración uno, el valor de la función objetivo es 931,720,587.26, que decrece drásticamente en la segunda iteración a 114,928,415.23, lo cual implica un cambio significativo en el modelo.

Tabla 2. Resultados de la ejecución de la instancia WDBC con el algoritmo FCM con los parámetros $\varepsilon = 0.01$, $m = 2$ y $c = 4$

Iteración	Función Objetivo	Criterio de paro ε	Cambios de grupo
1	931720587.26	1.643709	0
2	114928415.23	1.836642	213
3	86735020.70	0.814777	34
4	79256193.10	0.492933	40
5	75523678.07	0.382413	22
6	72821698.97	0.326913	11
7	70777163.96	0.280027	19
8	69279538.88	0.237931	24
9	68213009.88	0.191650	10
.	.	.	.
.	.	.	.
.	.	.	.
102	63079319.24	0.046016	4

103	63018573.97	0.044855	3
104	62959720.88	0.043302	2
.	.	.	.
.	.	.	.
.	.	.	.
187	62419713.09	0.000018	0
188	62419713.09	0.000016	0
189	62419713.08	0.000015	0
190	62419713.07	0.000013	0
191	62419713.07	0.000012	0
192	62419713.07	0.000011	0
193	62419713.06	0.000010	0

A partir de la segunda iteración, el valor de la función objetivo continúa disminuyendo, aunque a un ritmo menos abrupto, hasta alcanzar 62,419,713.06 en la iteración 193. Este comportamiento refleja que el algoritmo converge de manera efectiva hacia una solución óptima, con una mejora constante pero con decrementos más pequeños conforme se avanza en las iteraciones.

El número de cambios de grupo varía considerablemente a lo largo del proceso. En la primera iteración no hay cambios de grupo, pero en la segunda iteración se observan 213 cambios. Después de esto, los cambios de grupo fluctúan, alcanzando picos de hasta 40 cambios en iteraciones tempranas, pero en general, se observa una tendencia decreciente en el número de cambios conforme se avanza hacia las últimas iteraciones. En las últimas iteraciones, los cambios de grupo se reducen a valores bajos, cercanos a 0, lo que refuerza la idea de que el algoritmo está convergiendo y estabilizando la solución.

Estos resultados obtenidos reflejan que el algoritmo FCM converge de manera efectiva hacia una solución estable, aunque se observa que la mayor parte de los cambios significativos en el valor de la función objetivo ocurren en las primeras iteraciones. A pesar de que el valor de la función objetivo sigue disminuyendo hasta las últimas iteraciones, las variaciones son mínimas, lo cual indica que el modelo alcanza un estado cercano al óptimo. Este comportamiento sugiere que continuar calculando los grados de membresía en cada iteración, aun cuando los cambios son insignificantes, contribuye considerablemente al tiempo de procesamiento total. Por lo tanto, es recomendable utilizar técnicas que reduzcan el número de cálculos necesarios, disminuyendo así el tiempo de procesamiento del algoritmo sin comprometer significativamente la calidad del resultado.

2.3. Definición la propuesta del nuevo algoritmo: FLCM++

La solución propuesta introduce una heurística destinada a reducir el tiempo de ejecución del algoritmo FCM. Esta estrategia se basa en detener el cálculo de los grados de pertenencia cuando no hay cambios en la asignación de grupos, mediante la identificación de los *Objetos Inalterados*, aquellos cuya asignación de grupo permanece constante. Al excluir estos objetos del procesamiento, se reduce la cantidad de cálculos y se incrementa la eficiencia del algoritmo.

Un vector V de n elementos, donde cada elemento es un valor Booleano que indica si un objeto ha experimentado cambios en su asignación de grupo. Si el valor es 0, el objeto ha sido reasignado; si es 1, su asignación de grupo permanece igual. Matemáticamente, el vector V se define como:

$$V = [v_1, v_2, \dots, v_n] \quad (1)$$

donde v_i representa el estado del objeto i : $v_i = 0$ si ha habido cambios en su asignación de grupo, y $v_i = 1$ en caso contrario.

El total de *Objetos Inalterados*, denotado por O , se calcula mediante la siguiente expresión:

$$O = \sum_{i=1}^n v_i \quad (2)$$

A partir de O , se determina el porcentaje γ de objetos cuya asignación se mantiene constante utilizando la fórmula:

$$\gamma = 100 \cdot \left(\frac{O}{n} \right) \quad (3)$$

Cuando γ es menor que un umbral mínimo δ , se aplica la estrategia de descarte de cálculos: los *Objetos Inalterados* mantienen el grado de pertenencia de la iteración anterior, contribuyendo así a reducir el tiempo de cálculo.

Otro desafío abordado en esta propuesta, es evitar la convergencia a óptimos locales debido a una inicialización inadecuada de los centroides. Con la finalidad de resolver este problema, se adopta un enfoque híbrido que incorpora el algoritmo k-Means++ (Algoritmo 1), mejorando la inicialización precisa de los centroides y mitigar la tendencia hacia óptimos locales.

El algoritmo K-Means++ constituye una mejora del tradicional algoritmo K-Means, cuya propuesta se encuentra detallada en [30]. Su función principal radica en la inicialización de los centroides en el contexto del algoritmo K-Means. La esencia de K-Means++ reside en la distribución más uniforme de los centros iniciales dentro del conjunto de datos, lo que, favorece una convergencia más rápida del algoritmo K-Means.

Algorithm 1 K-Means++

Entrada: Datos X , k

Salida: *Centroides*

1: **Inicialización**

2: Datos: $X = \{x_1, \dots, x_n\}$;

3: Asignar el valor de k

4: **Cálculo de centroides**

5: *Centroides* : $V = v_1$ //primer centroide aleatorio

6: **Para** $i = 2 : k$ **hacer**

7: Seleccionar el i -th centroide de X con Ecuación 4

8: *Centroides* : $V = V + v_i$

9: **Fin Para**

10: Retorna *Centroides*;

11: **Fin Algoritmo**

La complejidad computacional de K-Means++ se expresa de manera concisa como $O(\log k)$, donde k representa el número de grupos. Destacando que esta complejidad no guarda una relación directa con el tamaño de los datos, sino más bien con la cantidad de grupos, k , que se pretenden identificar. Este aspecto confiere eficiencia a la inicialización K-means++ incluso en conjuntos de datos extensos, es por lo que resulta óptimo la utilización de este algoritmo.

En la selección de los centroides toma en cuenta la Ecuación 4

$$P(v_i) = \frac{D(x_i, v_j)}{\sum_{x \in X} D(x_i, v_j)} \quad (4)$$

Donde: v_i es el i -ésimo centroide a seleccionar; x_i es el i -ésimo elemento en el conjunto de datos X ; $D(x_i, v_j)$ es la distancia al cuadrado entre el punto x_i y el centroide v_j ; y $\sum_{x \in X} D(x_i, v_j)$ es la suma de las distancias al cuadrado entre el punto x_i y todos los centroides v_j seleccionados.

La Figura 1 muestra un diagrama de flujo del algoritmo propuesto FLCM++.

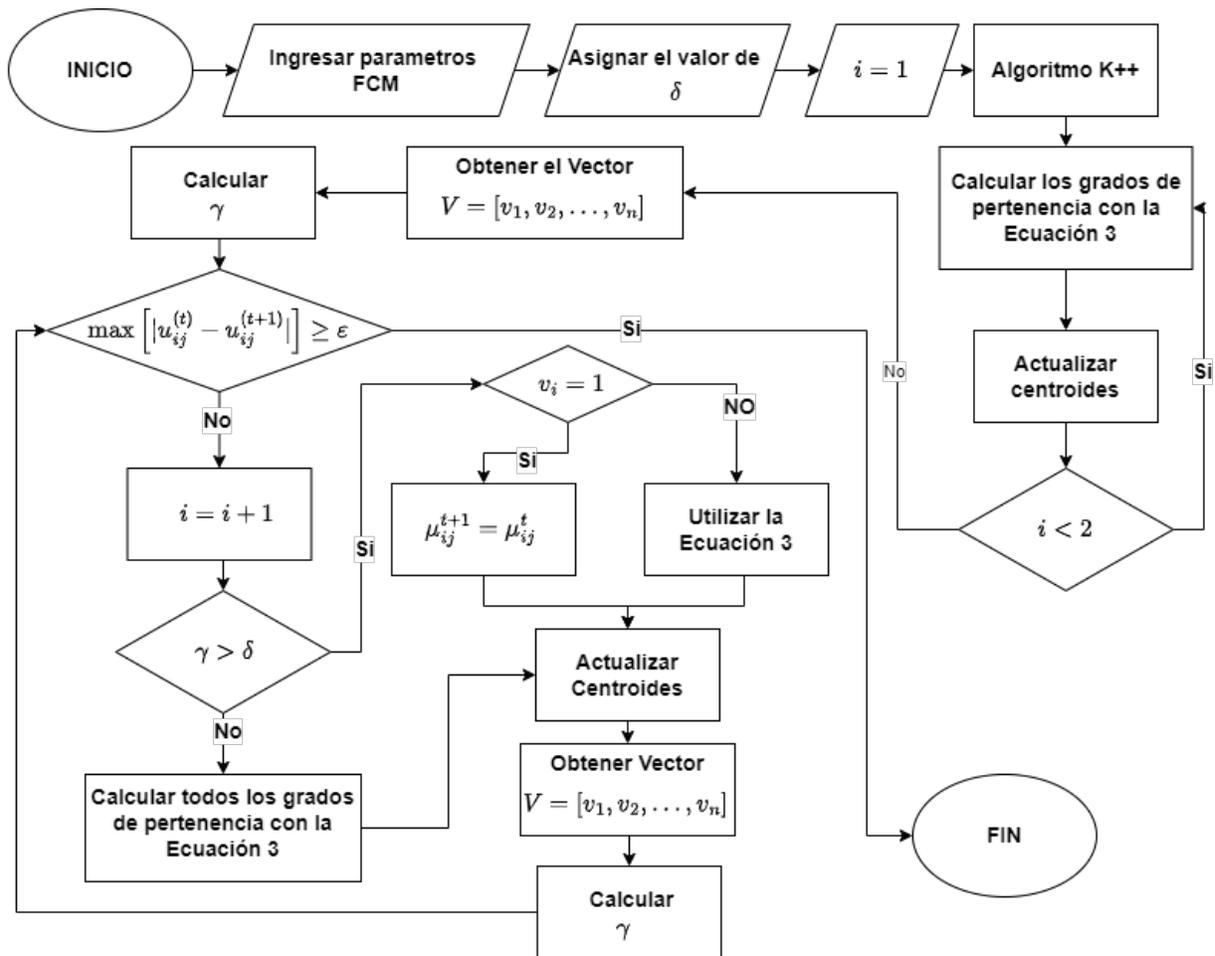


Figura 1. Diagrama de flujo del algoritmo FLCM++

El Algoritmo 4 presenta el funcionamiento detallado de FLCM++. El algoritmo comienza en la Línea 1 con la fase de Inicialización, donde se definen los parámetros de entrada. En la Línea 2, se ingresan estos datos y parámetros necesarios. A continuación, en la Línea 3, se ejecuta el algoritmo k-Means++ para generar una inicialización de los centroides. La Línea 4 abre un bucle que ejecutará dos iteraciones (para $i = 1:2$) para afinar la inicialización de los centroides. Durante cada iteración, la Línea 5 calcula la matriz de pertenencia usando la Ecuación ??, y la Línea 6 recalcula los centroides con la Ecuación ??.

En la Línea 7, el bucle finaliza tras dos iteraciones. En la Línea 8, se obtiene el vector $V = [v_1, v_2, \dots, v_n]$ para almacenar información sobre la pertenencia de los datos, y en la Línea 9 se calcula γ usando la Ecuación 3, lo cual permite evaluar la dispersión de los datos. La Línea 10 marca el inicio de la fase de Agrupamiento. En la Línea 11, se evalúa si $\gamma > \delta$; de ser así, se recalcula la matriz de pertenencia en la Línea 12 usando la Ecuación ??.

Si γ no es mayor que δ , se ingresa en un bloque condicional en la Línea 13, que evalúa si $v_i = 1$; en este caso, la Línea 14 establece que $\mu_{ij}^{t+1} = \mu_{ij}^t$. Si $v_i \neq 1$, en la Línea 16 se calcula el grado de pertenencia utilizando nuevamente la Ecuación ??.

Finalizado este bloque, se recalculan V y γ en las Líneas 17 y 18. La Línea 19 da inicio a la fase de Calcular Centroides, donde en la Línea 20 se recalculan los centroides mediante la Ecuación ??.

Luego, en la Línea 21, comienza la fase de Convergencia. En la Línea 22, si el cambio máximo entre $\mu_{ij}^{(t)}$ y $\mu_{ij}^{(t+1)}$ es menor que ε , el algoritmo se detiene; en caso contrario, en la Línea 24, se actualizan $U^{(t)}$ y t , y se regresa a la fase de Agrupamiento. El proceso continúa hasta alcanzar la convergencia.

Algorithm 2 Fuzzy Less C-Means

Entrada: Datos X , Centroides, m , ε , δ

Salida: U , Centroides

- 1: **Inicialización**
- 2: Ingresar los parámetros de entrada del algoritmo
- 3: Ejecutar el algoritmo k-Means++
- 4: **para i = 1:2**
- 5: Calcular la matriz de pertenencia mediante la Ecuación ??
- 6: Recalcular los centroides con la Ecuación ??
- 7: **Fin Para**
- 8: Obtener el vector $V = [v_1, v_2, \dots, v_n]$
- 9: Calcular γ con la Ecuación 3
- 10: **Agrupamiento**
- 11: **Si $\gamma > \delta$ entonces**
- 12: Calcular la matriz de pertenencia mediante la Ecuación ??
- 13: **Si no**
- 14: **Si $v_i = 1$ entonces**
- 15: $\mu_{ij}^{t+1} = \mu_{ij}^t$
- 16: **Si no**
- 17: Calcular el grado de pertenencia con la Ecuación ??
- 18: **Fin Si**
- 19: **Fin si**
- 20: Obtener el vector $V = [v_1, v_2, \dots, v_n]$
- 21: Calcular el valor de γ
- 22: **Calcular Centroides**
- 23: Calcular los centroides (Ecuación ??)
- 24: **Convergencia**
- 25: **Si $\max[|\mu_{ij}^{(t)} - \mu_{ij}^{(t+1)}|] < \varepsilon$ entonces**
- 26: Detener el algoritmo
- 27: **Si no**
- 28: $U^{(t)} = U^{(t+1)}$ y $t = t + 1$
- 29: Regresar a **Agrupamiento**
- 30: **Fin si**
- 31: **Fin Algoritmo**

2.4. Selección y validación del umbral δ

Al seleccionar el valor de δ , se debe lograr un equilibrio entre el tiempo de ejecución y la precisión, para ello se realizó una prueba con la instancia WDBC en donde se realizaron 50 ejecuciones para las configuraciones de $\delta = 0.1$ a 1.0 y para el número de centroides $c = 4, 6, 8, 10, 14, 18$ y 26. Tomando en cuenta un rango de $\delta = 0.1$ a 1.0.

En este trabajo se eligieron los valores de $c = 4, 6, 8, 10, 14, 18$ y 26 como el número de grupos a considerar, con el propósito de realizar un amplio número de pruebas y evaluar el rendimiento del método propuesto bajo diversas configuraciones. Esta selección permite analizar cómo varía el desempeño del

algoritmo al aumentar la complejidad del agrupamiento. Además, dichos valores fueron seleccionados estratégicamente para establecer una comparativa directa con los resultados reportados por Pérez y colab. en [21] y [31], quienes utilizó configuraciones similares en su estudio previo, lo que facilita validar y contextualizar los hallazgos obtenidos en el presente trabajo.

Las Tablas 3 y 4 muestran los resultados que comparan FCM estándar y FLCM++ en términos de tiempo de ejecución y calidad de agrupación (sumatoria de distancias al cuadrado) tomando en cuenta para δ valores de 0.1 hasta 1.0. La información de las tablas se encuentra distribuida de la siguiente forma: en la primera columna se muestra el valor de δ seleccionado para cada prueba, que varía de 0.1 a 1.0; en la columna 2 se muestra el valor de c . En las siguientes columnas se encuentran los tiempos de ejecución en segundos para los métodos FCM y FLCM++, con el tiempo de diferencia calculado como la resta entre FCM y FLCM++. Además, se presentan las sumatorias de las distancias al cuadrado para ambos métodos, seguidas de la diferencia porcentual entre FCM y FLCM++. La tabla se divide en dos partes: una que incluye los valores de δ de 0.1 a 0.5, y otra que cubre los valores restantes de δ desde 0.6 hasta 1.0.

Las Ecuaciones 5 y 6 muestran como se obtuvieron los porcentajes de reducción del tiempo y la ganancia en la sumatoria de las distancias al cuadrado.

$$\tau \% = 100 \times \left(\frac{T_{iempo_{flcm++}}}{T_{iempo_{fcm}}} \right) \quad (5)$$

$$E \% = 100 \times \left(\frac{Sumatoria_{flcm++}}{Sumatoria_{fcm}} \right) \quad (6)$$

En general, FLCM++ reduce el tiempo significativamente, con la mayor diferencia para $\delta = 0.1$, donde el ahorro llega al 66.83% con $c = 26$. En calidad, FLCM++ mejora para $\delta = 0.1$ y $c = 26$ con un 10.38% de reducción, sin embargo, pierde precisión en $\delta = 1.0$, especialmente con $c = 6$ (-3.32%). Valores de δ más bajos (0.1 a 0.5) presentan un balance entre tiempo y calidad, mientras que δ altos (0.6 a 1.0) muestran mayor variabilidad, con mejoras de tiempo, sin embargo, con mayores pérdidas de calidad, llegando hasta un 7.26%.

Las gráficas de las Figuras 2 y 3 presentan, respectivamente, el tiempo promedio de ejecución y la sumatoria de las distancias al cuadrado, dos indicadores clave para evaluar el desempeño del algoritmo. En la Figura 2, se observa que el tiempo de ejecución se mantiene razonablemente bajo para cualquier valor de $\delta < 1.0$, resultando en tiempos satisfactorios independientemente del número de grupos generados en cada prueba. Por otro lado, la Figura 3 muestra que la precisión, medida por la sumatoria de las distancias al cuadrado, se ve afectada negativamente al incrementar el valor de δ , especialmente en las pruebas con $c = 6, 8$ y 10 . Sin embargo, para valores de $\delta < 0.5$, esta pérdida en calidad es menor o igual que 2.04%, lo que permite obtener resultados comparables a los del algoritmo FCM. Esto sugiere que $\delta = 0.5$ logra un balance adecuado entre eficiencia en el tiempo de ejecución y precisión, evitando un aumento significativo en el error y presentándose como una opción óptima para el caso analizado.

Tabla 3. Pruebas realizadas para la selección del valor de δ : Tiempo y Sumatoria de Distancias al Cuadrado (Rango $\delta= 0.1, 0.2, \dots, 0.5$)

Selección del valor de δ							
δ	c	Tiempo (s)			Sumatoria de Distancias al Cuadrado		
		FCM	FLCM++	τ %	FCM	FLCM++	E %
0.1	4	8.80	5.54	37.00	80,258,122.59	80,243,817.83	0.02
	6	17.04	11.43	32.93	63,277,896.42	63,675,598.96	-0.63
	8	26.98	14.96	44.55	54,413,634.55	54,560,074.09	-0.27
	10	49.67	20.92	57.88	49,192,971.58	48,862,352.92	0.67
	14	96.09	34.04	64.57	43,511,532.63	42,519,267.02	2.28
	18	114.65	47.95	58.18	40,252,502.74	37,839,431.59	5.99
	26	275.99	91.55	66.83	34,981,932.78	31,350,889.44	10.38
0.2	4	8.80	5.04	42.76	80,258,122.59	80,255,170.81	0.00
	6	17.04	7.58	55.50	63,277,896.42	64,547,270.00	-2.01
	8	26.98	12.01	55.51	54,413,634.55	54,774,126.93	-0.66
	10	49.67	17.92	63.91	49,192,971.58	49,257,511.22	-0.13
	14	96.09	25.06	73.92	43,511,532.63	42,756,591.44	1.74
	18	114.65	36.52	68.15	40,252,502.74	37,875,313.08	5.91
	26	275.99	63.02	77.16	34,981,932.78	31,639,294.81	9.56
0.3	4	8.80	4.94	43.89	80,258,122.59	80,307,839.98	-0.06
	6	17.04	7.68	54.91	63,277,896.42	63,655,943.89	-0.60
	8	26.98	12.94	52.04	54,413,634.55	54,711,706.98	-0.55
	10	49.67	18.50	62.76	49,192,971.58	48,876,692.13	0.64
	14	96.09	27.95	70.91	43,511,532.63	42,501,877.80	2.32
	18	114.65	39.29	65.73	40,252,502.74	37,736,848.03	6.25
	26	275.99	67.08	75.70	34,981,932.78	31,561,050.29	9.78
0.4	4	8.80	4.73	46.26	80,258,122.59	80,423,610.05	-0.21
	6	17.04	6.76	60.35	63,277,896.42	64,121,848.07	-1.33
	8	26.98	10.96	59.40	54,413,634.55	54,885,600.95	-0.87
	10	49.67	13.76	72.29	49,192,971.58	49,762,223.49	-1.16
	14	96.09	22.22	76.88	43,511,532.63	42,735,658.72	1.78
	18	114.65	32.99	71.22	40,252,502.74	37,879,410.02	5.90
	26	275.99	53.09	80.76	34,981,932.78	31,923,910.87	8.74
0.5	4	8.80	4.86	44.80	80,258,122.59	80,327,351.17	-0.09
	6	17.04	7.18	57.85	63,277,896.42	64,566,472.70	-2.04
	8	26.98	11.39	57.78	54,413,634.55	54,833,022.14	-0.77
	10	49.67	14.00	71.81	49,192,971.58	49,334,150.63	-0.29
	14	96.09	23.16	75.90	43,511,532.63	42,807,598.34	1.62
	18	114.65	34.10	70.25	40,252,502.74	37,944,607.70	5.73
	26	275.99	47.39	82.83	34,981,932.78	31,674,059.24	9.46

Tabla 4. Pruebas realizadas para la selección del valor de δ : Tiempo y Sumatoria de Distancias al Cuadrado (Rango $\delta= 0.6, 0.7, \dots, 1.0$)

Selección del valor de δ							
δ	c	Tiempo (s)			Sumatoria de Distancias al Cuadrado		
		FCM	FLCM++	τ %	FCM	FLCM++	E %
0.6	4	8.80	4.98	43.41	80,258,122.59	80,632,375.87	-0.47
	6	17.04	6.83	59.95	63,277,896.42	64,932,757.16	-2.62
	8	26.98	10.28	61.89	54,413,634.55	55,420,788.14	-1.85
	10	49.67	11.94	75.95	49,192,971.58	49,702,366.85	-1.04
	14	96.09	21.54	77.59	43,511,532.63	42,769,147.16	1.71
	18	114.65	26.96	76.48	40,252,502.74	38,576,406.91	4.16
	26	275.99	46.87	83.02	34,981,932.78	31,860,423.76	8.92
0.7	4	8.80	4.53	48.58	80,258,122.59	80,716,993.13	-0.57
	6	17.04	6.71	60.63	63,277,896.42	63,769,632.26	-0.78
	8	26.98	10.29	61.87	54,413,634.55	55,455,518.97	-1.91
	10	49.67	14.02	71.77	49,192,971.58	49,564,010.49	-0.75
	14	96.09	19.51	79.70	43,511,532.63	42,683,547.67	1.90
	18	114.65	26.11	77.23	40,252,502.74	38,469,066.48	4.43
	26	275.99	48.74	82.34	34,981,932.78	31,923,408.14	8.74
0.8	4	8.80	4.80	45.45	80,258,122.59	80,636,946.56	-0.47
	6	17.04	6.47	62.06	63,277,896.42	64,958,066.72	-2.66
	8	26.98	9.39	65.18	54,413,634.55	55,143,583.13	-1.34
	10	49.67	11.58	76.67	49,192,971.58	49,970,060.73	-1.58
	14	96.09	17.44	81.85	43,511,532.63	42,944,237.93	1.30
	18	114.65	25.95	77.37	40,252,502.74	38,628,239.67	4.04
	26	275.99	42.18	84.72	34,981,932.78	31,979,906.85	8.58
0.9	4	8.80	4.30	51.15	80,258,122.59	80,642,202.48	-0.48
	6	17.04	5.69	66.59	63,277,896.42	64,283,583.80	-1.59
	8	26.98	8.17	69.71	54,413,634.55	56,181,003.92	-3.25
	10	49.67	9.55	80.76	49,192,971.58	49,923,147.78	-1.48
	14	96.09	16.98	82.32	43,511,532.63	42,993,113.07	1.19
	18	114.65	22.55	80.33	40,252,502.74	38,627,024.39	4.04
	26	275.99	39.50	85.69	34,981,932.78	32,344,335.91	7.54
1.0	4	8.80	3.97	54.84	80,258,122.59	80,939,368.32	-0.85
	6	17.04	5.82	65.82	63,277,896.42	65,380,673.63	-3.32
	8	26.98	8.16	69.74	54,413,634.55	55,919,754.51	-2.77
	10	49.67	10.05	79.75	49,192,971.58	50,193,602.68	-2.03
	14	96.09	17.66	84.60	43,511,532.63	43,174,571.55	-7.26
	18	114.65	21.59	81.16	40,252,502.74	38,658,913.52	3.96
	26	275.99	35.43	87.16	34,981,932.78	32,499,143.53	7.10

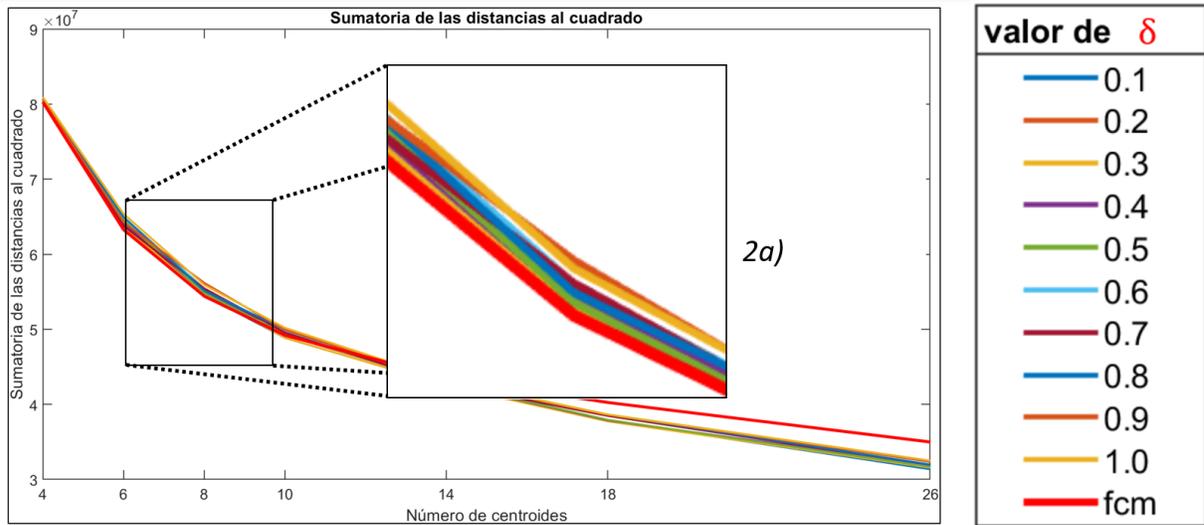


Figura 2. La suma de los cuadrados de las distancias al cuadrado de cada prueba con respecto a los valores correspondientes de δ . 2a) muestra una ampliación de la figura 2 en la cual se puede observar que mientras mayor sea δ existe mayor pérdida en la calidad del agrupamiento

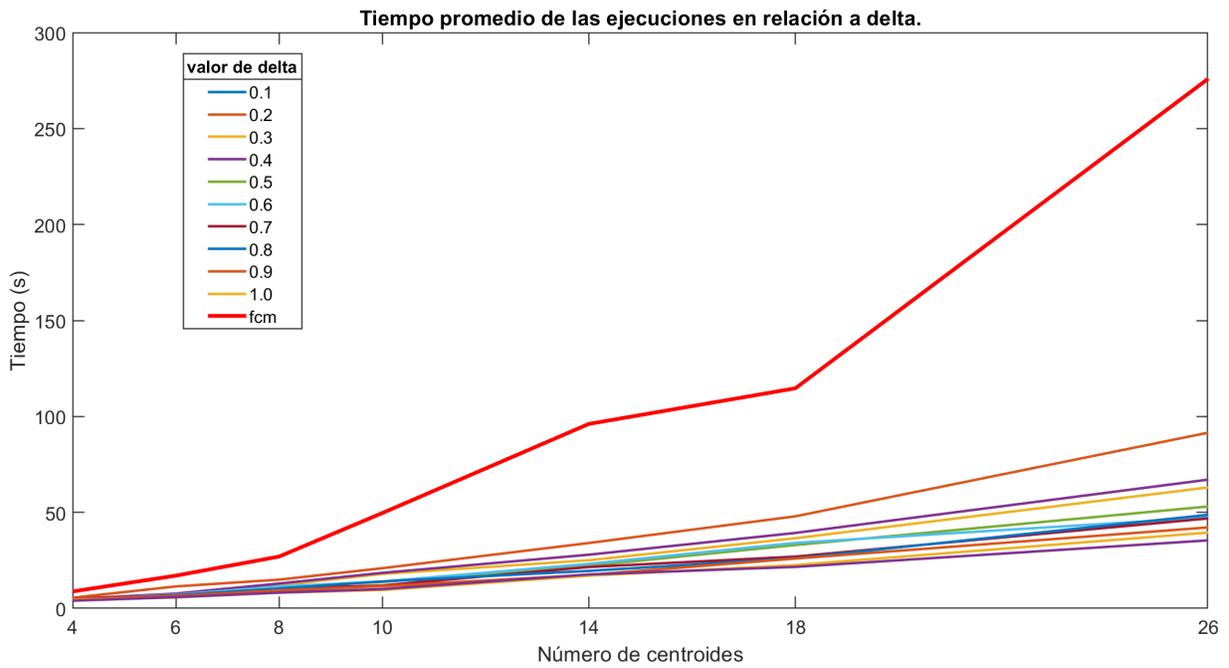


Figura 3. Tiempo promedio que tomó cada una de las pruebas, en función de los diferentes valores de δ

3. Experimentación para la validación de los Resultados

El agrupamiento es una técnica clave en la minería de datos y la inteligencia artificial que permite organizar grandes volúmenes de información en grupos. Hoy en día, esta herramienta es fundamental en diversas aplicaciones, como la segmentación de clientes en marketing, la mejora en la recomendación de productos en comercio electrónico, la optimización de procesos en manufactura, etc. Su capacidad para descubrir patrones en los datos la convierte en una herramienta valiosa para la toma de decisiones.

Este capítulo se encuentra distribuido de la siguiente forma: La Sección 4.1 describe el entorno de pruebas utilizado. La Sección 4.2 muestra las pruebas realizadas con el algoritmo FLCM++ con conjuntos de datos numéricos. La Sección 4.3 muestra el caso de prueba aplicado.

3.1. Entorno de pruebas

Con la finalidad de evaluar el rendimiento del algoritmo FLCM++, se llevó a cabo su implementación en MATLAB® R2022b, utilizando una computadora equipada con un procesador Intel Core i9 de 3.42 GHz, 64 GB de RAM y ejecutando el sistema operativo Windows 11 Pro. Se eligieron los conjuntos de datos reales WDBC [28], ABALONE [32], SPAM [33] y URBAN [34], los cuales están disponibles en un repositorio de aprendizaje automático de la UCI [29].

Con el fin de comparar los resultados del algoritmo FLCM++, se implementaron varios algoritmos, incluyendo FCM estándar, FCM++ y HOFM. Estos algoritmos se configuraron con los parámetros $m = 2$ y $\varepsilon = 0.01$. Para todas las configuraciones de los algoritmos, se llevaron a cabo pruebas variando el valor de c , específicamente $c = 4, 6, 8, 10, 14, 18$ y 26 siguiendo la configuración realizada por Pérez y Colab. en [21] y [31] realizando así una comparativa. Para el algoritmo FLCM++ se asignó un valor de 0.5 a δ .

El número de centroides c fue seleccionado con fines exploratorios. Este enfoque permite evaluar el desempeño del algoritmo FCM bajo distintas configuraciones sin asumir un conocimiento previo sobre la estructura inherente de los datos. Aunque existen técnicas para determinar un número óptimo de grupos, en este caso el objetivo no es optimizar c , sino analizar cómo varía el comportamiento del algoritmo al trabajar con diferentes cantidades de grupos, generando una base de comparación más diversa.

Cada algoritmo se ejecutó en 50 ocasiones para cada configuración de c , proporcionando una base sólida para evaluar su desempeño en diversas condiciones.

En la Tabla 5, se detallan los conjuntos de datos numéricos reales empleados, estructurando la información de la siguiente manera: la primera columna contiene el identificador del conjunto de datos, la segunda columna el nombre, la tercera y cuarta columna presentan el total de objetos y dimensiones del conjunto de datos, respectivamente, mientras que la quinta columna muestra el producto de las columnas tres y cuatro. Al observar la información de la última columna, se destaca que los conjuntos de datos SPAM [33] y URBAN [34] pueden clasificarse como grandes, debido a que el indicador $n * d$ supera los 200,000 objetos.

La selección de los conjuntos de datos presentados en la Tabla 5 se fundamenta en su diversidad respecto a su tamaño n y número de características d . Estos conjuntos incluyen desde bases

pequeñas y con pocas dimensiones, como WDBC, hasta bases grandes y de alta dimensionalidad, como SPAM, y bases masivas como URBAN, que presentan desafíos adicionales por su tamaño. Esta variedad permite evaluar el rendimiento y como se adapta el algoritmo FLCM++ propuesto en diferentes escenarios experimentales, asegurando que los resultados obtenidos sean representativos y generalizables a problemas reales. Además, todos los conjuntos de datos son ampliamente utilizados en la literatura, lo que facilita la comparación directa con trabajos previos.

Con el objetivo de evaluar el rendimiento de los algoritmos implementados y el algoritmo propuesto, se registró el tiempo empleado, expresado en milisegundos. Para evaluar la calidad del agrupamiento, se consideró la función objetivo del algoritmo FCM, que se presenta en la Ecuación ??.

Tabla 5. Características de los conjuntos de datos

Id	Nombre	n	d	Tamaño $n \times d$
1	WDBC [28]	569	30	17,070
2	ABALONE [32]	4,177	7	29,239
3	SPAM [33]	4,601	57	262,257
4	URBAN [34]	360,177	2	720,354

Los conjuntos de datos seleccionados destacan por su variedad en características y dominios, lo que permite evaluar el desempeño del algoritmo FCM en diferentes contextos:

- **WDBC:** Con 569 instancias y 30 características reales, es ideal para identificar agrupaciones relacionadas con características benignas y malignas en imágenes médicas.
- **ABALONE:** Con 4177 instancias y 8 características mixtas (categóricas, enteras y reales), este dataset permite explorar patrones en mediciones físicas para predecir la edad de los abalones.
- **SPAM:** Compuesto por 4601 instancias y 57 características numéricas, es útil para agrupar correos electrónicos según su similitud en clasificación como spam o no spam.
- **URBAN:** Este dataset incluye 360,177 instancias con coordenadas (longitud y latitud) y 2 características reales, siendo ideal para identificar centros urbanos donde ocurren accidentes.

Esta diversidad de datos permite probar la eficacia del FCM en tareas con distintas cantidades de características, tamaños y dominios de aplicación.

3.2. Realización de pruebas con distintos conjuntos de datos

Los experimentos realizados en este estudio están diseñados para evaluar tanto la calidad de las soluciones como la eficiencia del algoritmo FLCM++, para ello se han realizado experimentaciones en las que se compara el algoritmo FCM estándar y FCM++ contra FLCM++.

Se han aplicado los algoritmos sobre conjuntos de datos de gran tamaño mostrados en la Tabla 5, con el objetivo de medir la eficiencia en términos de tiempo de ejecución y la capacidad de FLCM++

para manejar conjuntos de datos extensos de manera eficiente. Estos experimentos permiten observar cómo el algoritmo reduce el tiempo de procesamiento al excluir objetos inalterados de las iteraciones, reduciendo el cómputo innecesario.

Experimento I: FCM vs FLCM++

El propósito de este experimento es contrastar los resultados de los algoritmos FCM estándar y FLCM++.

En la Tabla 6 se muestran los resultados obtenidos en cuanto al tiempo y sumatoria de las distancias al cuadrado de las pruebas realizadas a los algoritmos FCM y FLCM++ para los conjuntos de datos numéricos reales detallados en la Tabla 5.

La estructura de la Tabla 6 se detalla a continuación:

- La **columna uno** contiene el conjunto de datos empleado.
- La **columna dos** presenta el número consecutivo de cada prueba para cada conjunto de datos.
- La **columna tres** indica el número de centroides utilizado en cada prueba.
- Las **columnas cuatro y cinco** muestran los resultados del tiempo promedio para FCM y FLCM++, respectivamente.
- La **columna seis** refleja el porcentaje de reducción del tiempo de FLCM++ en comparación con FCM.
- Las **columnas siete y ocho** contienen los resultados de FCM y FLCM++ respecto a la sumatoria de las distancias al cuadrado.
- Finalmente, la **columna nueve** presenta la ganancia en la sumatoria de las distancias al cuadrado.

Además, se encuentra encerrado en un rectángulo la mayor pérdida de calidad obtenida, y en color gris la mayor reducción de tiempo y la mayor ganancia en la sumatoria de las distancias al cuadrado.

Los resultados demuestran que FLCM++ logra una reducción significativa en el tiempo de ejecución en comparación con FCM, logrando una reducción del tiempo de hasta un 90.38 % y una mejora en la calidad del agrupamiento de hasta un 66.39 %. Sin embargo, en las situaciones más desafiantes, se observó una pérdida de calidad de hasta un 2.27 % en el peor de los casos.

Tabla 6. Comparación entre FCM y FLCM++: Tiempo y Sumatoria de Distancias al Cuadrado

FCM VS FLCM++								
Conjunto de datos	p	c	Tiempo (s)			Sumatoria de Distancias al Cuadrado		
			FCM	FLCM++	τ %	FCM	FLCM++	E %
WDBC [28]	1	4	8.80	6.82	22.51	80,258,122.59	80,453,739.70	-0.24
	2	6	17.04	7.45	56.28	63,277,896.42	64,319,869.36	-1.65
	3	8	26.98	10.93	59.50	54,413,634.55	55,141,402.78	-1.34
	4	10	49.67	13.94	71.93	49,192,971.58	49,352,919.43	-0.33
	5	14	96.09	23.04	76.02	43,511,532.63	42,738,518.01	1.78
	6	18	114.65	33.69	70.62	40,252,502.74	37,951,222.12	5.72
	7	26	275.99	49.54	82.05	34,981,932.78	32,056,608.06	8.36
ABALONE [32]	1	4	16.09	14.84	7.77	479.17	479.53	-0.08
	2	6	41.34	29.54	28.54	373.41	374.44	-0.28
	3	8	85.53	54.79	35.94	324.81	325.34	-0.16
	4	10	158.27	79.78	49.59	299.77	301.26	-0.50
	5	14	282.72	151.89	46.28	275.19	277.14	-0.71
	6	18	493.30	237.45	51.86	265.79	267.59	-0.68
	7	26	1,409.02	812.49	42.34	252.48	257.03	-1.80
SPAM [33]	1	4	119.00	53.92	54.69	913,362,052.73	862,177,981.03	5.60
	2	6	467.66	88.46	81.08	842,646,132.33	684,425,841.66	18.78
	3	8	644.40	121.99	81.07	789,949,582.84	524,912,176.51	33.55
	4	10	998.89	193.60	80.62	745,256,630.28	400,430,527.95	46.27
	5	14	2,769.44	315.39	88.61	590,775,731.41	302,429,289.57	48.81
	6	18	4,101.16	406.99	90.08	547,217,835.65	247,783,899.00	54.72
	7	26	9,069.25	872.16	90.38	574,204,251.60	193,009,932.25	66.39
URBAN [34]	1	4	964.36	625.39	35.15	661,335.70	665,012.86	-0.56
	2	6	2,019.83	1,382.16	31.57	554,119.01	562,402.68	-1.49
	3	8	5,485.13	2,147.87	60.84	500,283.52	511,637.93	-2.27
	4	10	7,199.04	3,347.96	53.49	463,378.05	462,288.75	0.24
	5	14	11,414.97	8,574.81	24.88	397,127.56	389,147.71	2.01
	6	18	25,071.58	9,029.19	63.99	370,856.96	346,533.09	6.56
	7	26	48,635.92	17,928.58	63.14	320,985.92	299,858.52	6.58

En las Figuras 4 y 5 se presentan gráficamente los resultados de los algoritmos FCM y FLCM++ en los diferentes conjuntos de datos empleados en las pruebas.

La Figura 4 muestra que, en la mayoría de los casos, FLCM++ presenta una reducción notable en el tiempo promedio de ejecución en comparación con el FCM estándar.

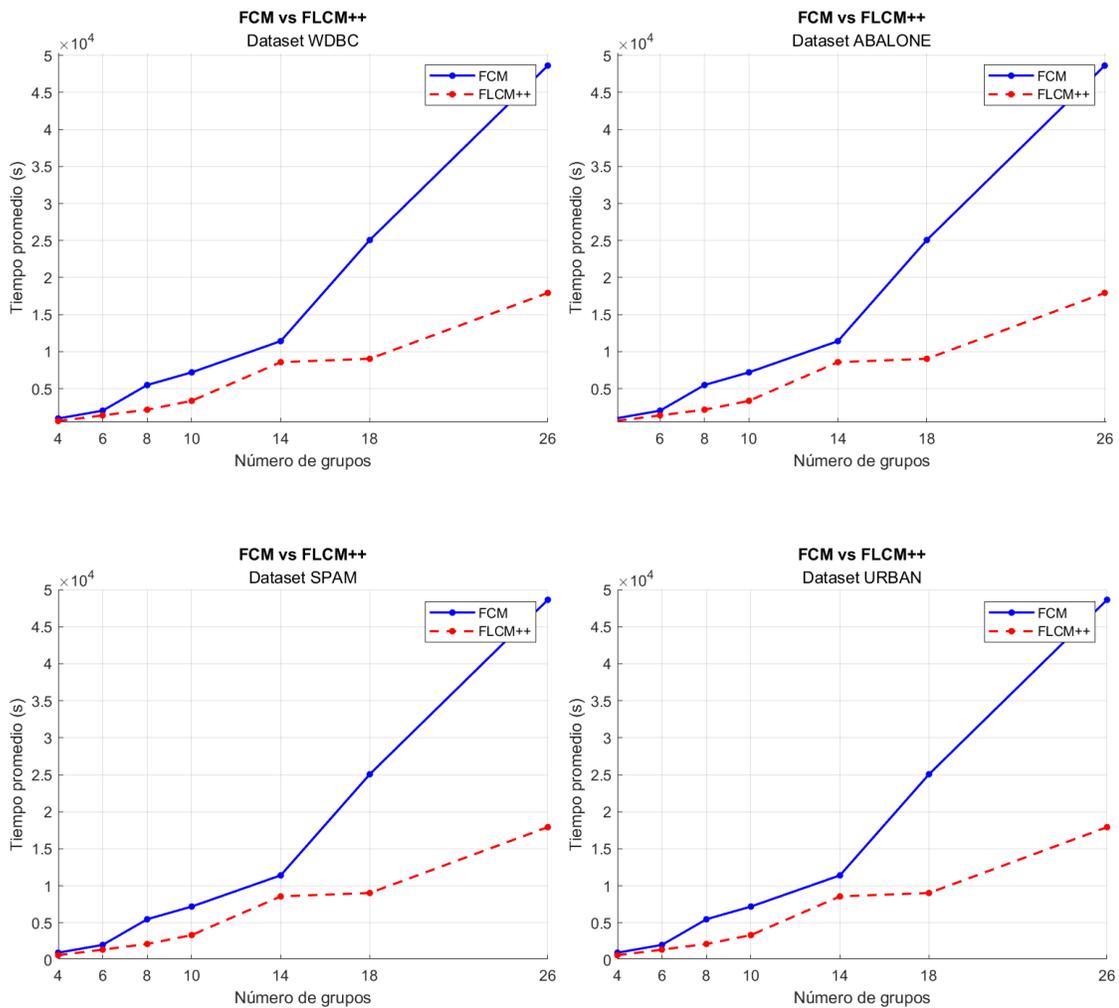


Figura 4. Resultados de FCM vs FLCM++ en relación con el tiempo de ejecución.

Por otro lado, la Figura 5 ilustra la sumatoria de las distancias al cuadrado. Se observa que, en los conjuntos de datos WDBC [28] y Urban [34], FLCM++ muestra una ligera ganancia en cuanto a la calidad del agrupamiento de hasta el 8.36%. En el conjunto de datos ABALONE [32] no se percibe un cambio significativo, mientras que en SPAM [33] el algoritmo FLCM++ ofrece un mejor rendimiento de un máximo de 66.39%, con una ganancia notable en la calidad del agrupamiento en comparación con el FCM estándar. Los resultados para el conjunto de datos SPAM [33] resultan tener un mejor resultado del 66.39% que destaca considerablemente en cuanto a la calidad del agrupamiento, esto es gracias a la utilización del algoritmo K-Means++ que permite una mejor inicialización de los centroides contribuyendo a obtener mejores resultados.

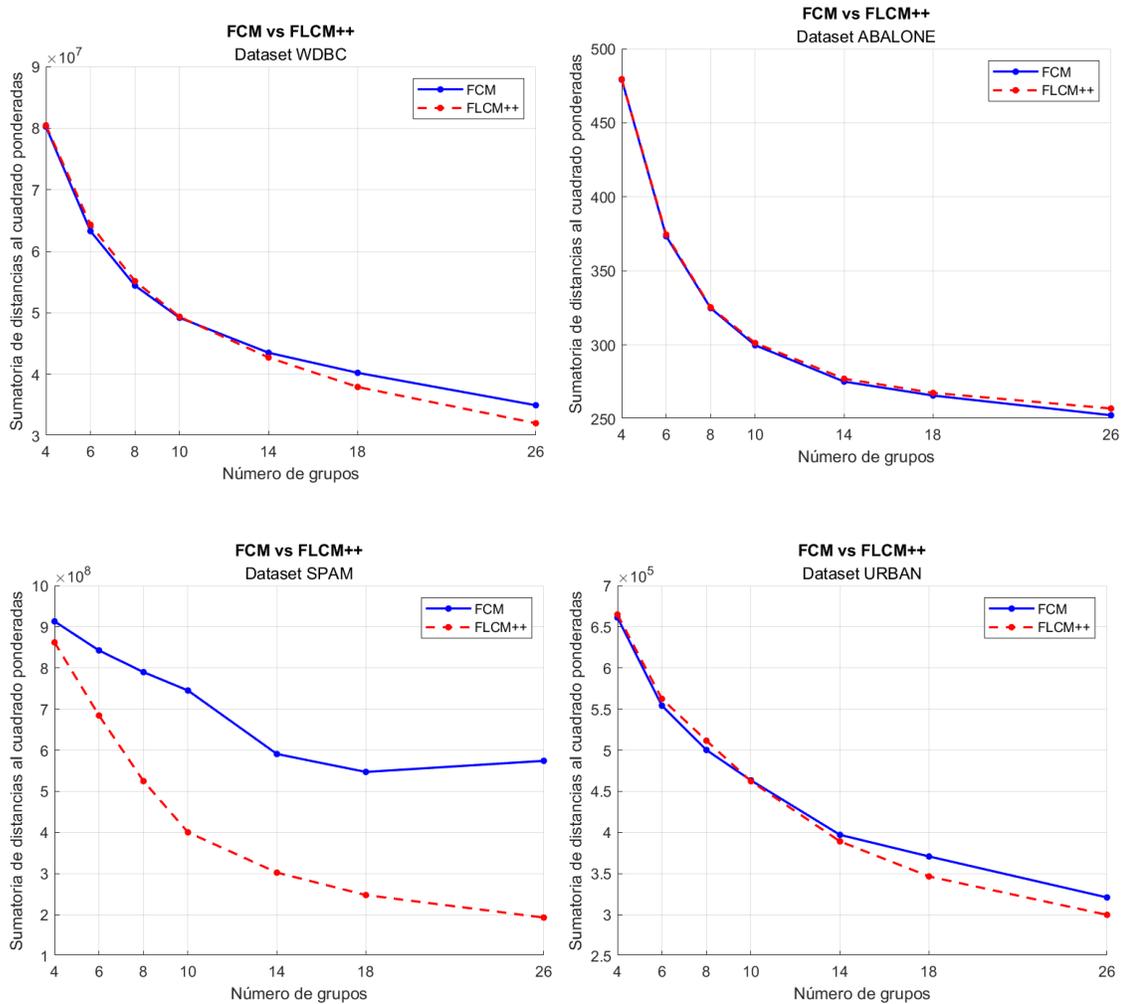


Figura 5. Resultados de FCM vs FLCM++ en relación con la suma de las distancias al cuadrado.

Con el objetivo de determinar si las diferencias observadas entre ambos algoritmos son estadísticamente significativas, se utilizó la prueba de Wilcoxon. La hipótesis nula H_0 establece que no hay diferencias significativas entre los algoritmos, mientras que la hipótesis alternativa H_1 indica la existencia de diferencias significativas. El valor p asociado a la prueba se interpreta en función de un nivel de significancia predefinido $\alpha = 0.05$. Si $p < \alpha$, se rechaza la hipótesis nula, concluyendo que existen diferencias significativas. Si $p \geq \alpha$, no se rechaza la hipótesis nula.

Para el tiempo promedio de ejecución, la prueba de Wilcoxon arrojó un valor $p = 0.0000038$, que es significativamente menor que el nivel de significancia comúnmente utilizado $\alpha = 0.05$. Esto implica que se rechaza la hipótesis nula H_0 , concluyendo que las diferencias observadas en los tiempos de ejecución entre FCM y FLCM++ son estadísticamente significativas. Por lo tanto, se puede afirmar con confianza que FLCM++ es significativamente más rápido que FCM en los conjuntos de datos analizados.

Por otro lado, para la sumatoria de distancias al cuadrado, la prueba de Wilcoxon arrojó un valor $p =$

0.083518, que es mayor que el nivel de significancia de $\alpha = 0.05$. Esto significa que no se rechaza la hipótesis nula, sugiriendo que no hay evidencia estadística suficiente para afirmar que las diferencias en la calidad de las particiones (según esta métrica) entre FCM y FLCM++ son significativas.

Experimento II: FCM++ vs FLCM++

Con el fin de evaluar el desempeño del algoritmo FLCM++, fueron comparados sus resultados con los del algoritmo FCM++. Los datos se presentan en la Tabla 7, organizada de la misma forma que en el Experimento 1. En esta tabla se muestran los porcentajes promedio de reducción del tiempo de ejecución y la mejora en la sumatoria de las distancias al cuadrado para los conjuntos de datos numéricos reales. Los mejores resultados se encuentran resaltados en color gris y encerrados en un rectángulo se encuentran las mayores pérdidas en la calidad del agrupamiento.

La Figura 6 presenta la gráfica de los resultados en términos de tiempo de ejecución. Destacando que el algoritmo FLCM++ logra una reducción significativa en el tiempo de procesamiento frente a FCM++. Por otro lado, la Figura 7 muestra la gráfica de la sumatoria de las distancias al cuadrado, reflejando la calidad del agrupamiento. Esta gráfica confirma que FLCM++ mantiene una calidad de agrupamiento similar a FCM, sin embargo, con tiempos de ejecución menores.

En el mejor de los casos, FLCM++ logra una reducción del 76.38% en el tiempo de ejecución y una mejora en la calidad del agrupamiento de hasta un 3.57%. Sin embargo, en el peor de los casos, se observa una pérdida en la calidad de agrupamiento de hasta un 2.03%.

Tabla 7. Comparación entre FCM++ y FLCM++: Tiempo y Sumatoria de Distancias al Cuadrado

FCM++ VS FLCM++								
		Tiempo (s)				Sumatoria de Distancias al Cuadrado		
Conjunto de datos	p	c	FCM++	FLCM++	τ %	FCM++	FLCM++	E %
WDBC [28]	1	4	7.76	6.82	12.08	80,255,819.81	80,453,739.70	-0.25
	2	6	18.14	7.45	58.94	63,492,031.23	64,319,869.36	-1.30
	3	8	23.67	10.93	53.84	54,557,102.97	55,141,402.78	-1.07
	4	10	46.31	13.94	69.89	48,810,241.61	49,352,919.43	-1.11
	5	14	63.91	23.04	63.95	42,522,407.61	42,738,518.01	-0.51
	6	18	82.16	33.69	59.00	37,839,574.43	37,951,222.12	-0.30
	7	26	209.71	49.54	76.38	31,479,610.36	32,056,608.06	-1.83
ABALONE [32]	1	4	16.61	14.84	10.69	479.26	479.53	-0.06
	2	6	37.63	29.54	21.50	373.46	374.44	-0.26
	3	8	66.96	54.79	18.18	324.85	325.34	-0.15
	4	10	159.53	79.78	49.99	299.79	301.26	-0.49
	5	14	322.25	151.89	52.87	276.14	277.14	-0.36
	6	18	422.38	237.45	43.78	268.16	267.59	0.21
	7	26	1,668.54	812.49	51.31	255.07	257.03	-0.77
SPAM [33]	1	4	72.59	53.92	25.72	927,301,189.20	862,177,981.03	7.02
	2	6	139.03	88.46	36.37	703,744,690.71	684,425,841.66	2.75
	3	8	282.84	121.99	56.87	544,327,041.05	524,912,176.51	3.57
	4	10	420.38	193.60	53.95	411,904,767.35	400,430,527.95	2.79
	5	14	769.35	315.39	59.01	300,594,794.45	302,429,289.57	-0.61
	6	18	1,210.01	406.99	66.36	247,181,252.90	247,783,899.00	-0.24
	7	26	2,099.46	872.16	58.46	194,730,295.16	193,009,932.25	0.88
URBAN [34]	1	4	783.38	625.39	20.17	661,325.22	665,012.86	-0.56
	2	6	2,553.88	1,382.16	45.88	561,637.35	562,402.68	-0.14
	3	8	4,900.15	2,147.87	56.17	501,437.88	511,637.93	-2.03
	4	10	6,133.41	3,347.96	45.41	463,157.48	462,288.75	0.19
	5	14	11,229.15	8,574.81	23.64	393,225.81	389,147.71	1.04
	6	18	20,405.04	9,029.19	55.75	353,161.68	346,533.09	1.88
	7	26	41,166.96	17,928.58	56.45	301,344.06	299,858.52	0.49

El análisis de los resultados mediante las pruebas de Wilcoxon indica que FLCM++ presenta mejoras significativas en términos de eficiencia temporal respecto a FCM++, con un valor de $p = 0.0000037896$. Por otro lado, para la sumatoria de distancias al cuadrado, con un valor de $p = 0.53867$, no se encontraron diferencias significativas, lo que sugiere que la calidad de las particiones generadas por ambos algoritmos es comparable.

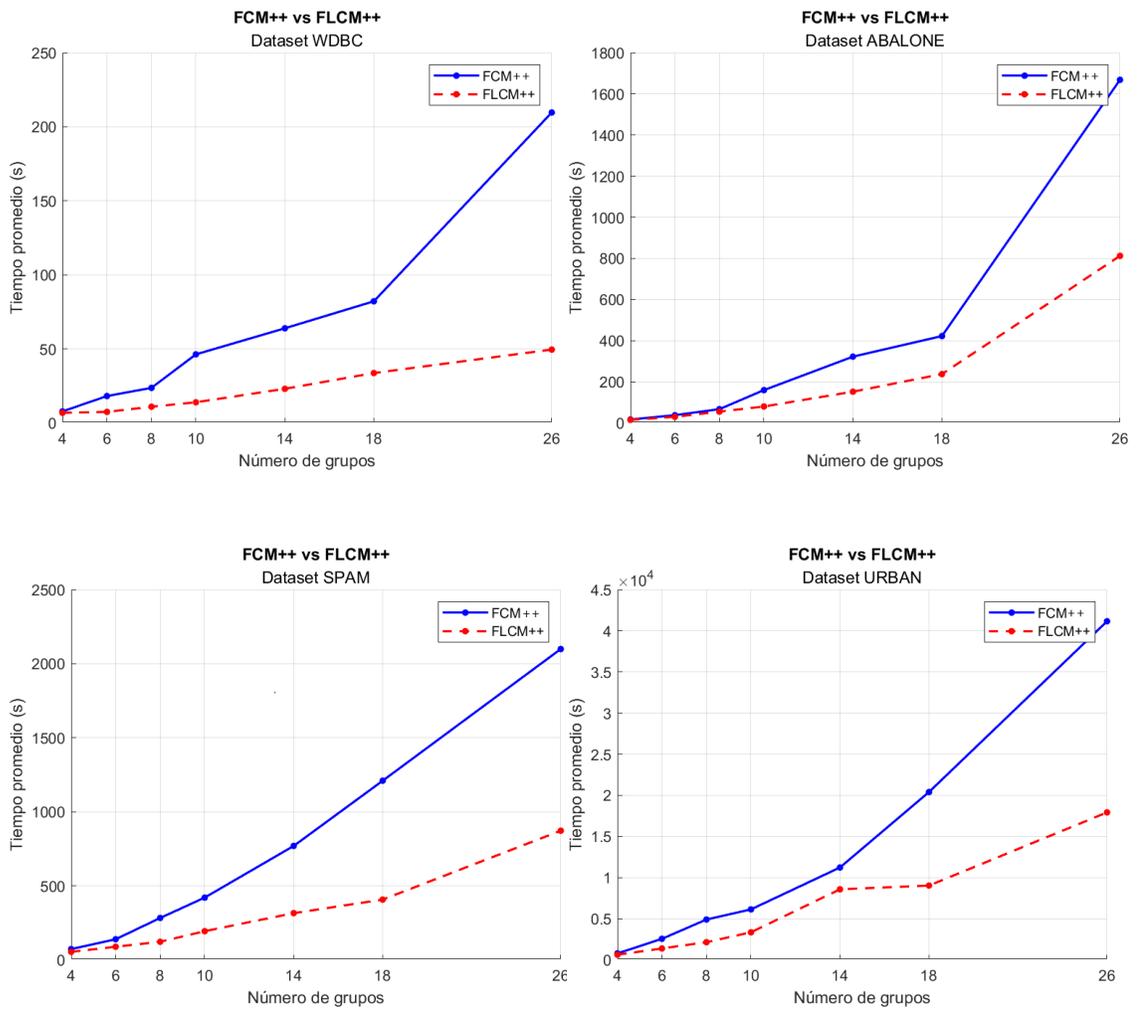


Figura 6. Resultados de FCM++ vs FLCM++ en relación con el tiempo de ejecución.

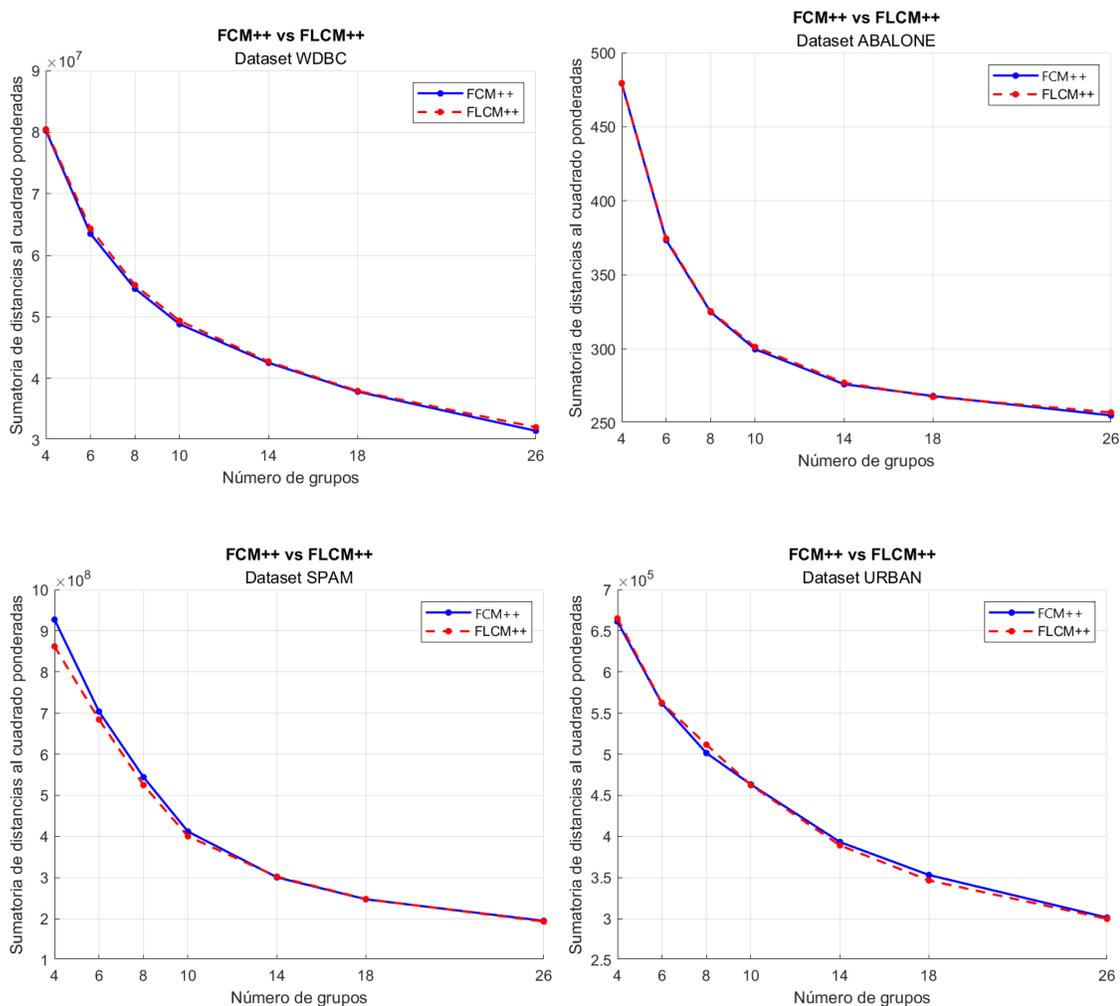


Figura 7. Resultados de FCM++ vs FLCM++ en relación con la sumatoria de las distancias al cuadrado.

Experimentación III: HOFM vs FLCM++

En esta experimentación, los resultados del algoritmo FLCM++ fueron comparados con los del algoritmo HOFM. Los resultados de las pruebas están en la Tabla 8. En términos de tiempo de ejecución, FLCM++ logró una reducción de hasta un 74.21% en el mejor de los casos. En cuanto a la calidad del agrupamiento, se observó una pérdida máxima del 3.73% y en el mejor escenario se alcanzó una mejora de hasta el 62.49%.

La Figura 8 presenta de forma gráfica los resultados de las pruebas realizadas, donde se observa en la mayoría de los casos que FLCM++ logra una reducción en el tiempo de procesamiento. HOFM es una mejora del algoritmo FCM publicada en el año 2022 por Pérez y Colab. [21], sirviendo como referencia al evaluar los resultados obtenidos por FLCM++. Respecto a la calidad del agrupamiento, la Figura 9 muestra resultados favorables, evidenciando un comportamiento similar entre los algoritmos. Los resultados para el conjunto de datos SPAM, muestran una mejora considerable en la calidad del

agrupamiento con FLCM++, destacándose significativamente frente a HOFCM.

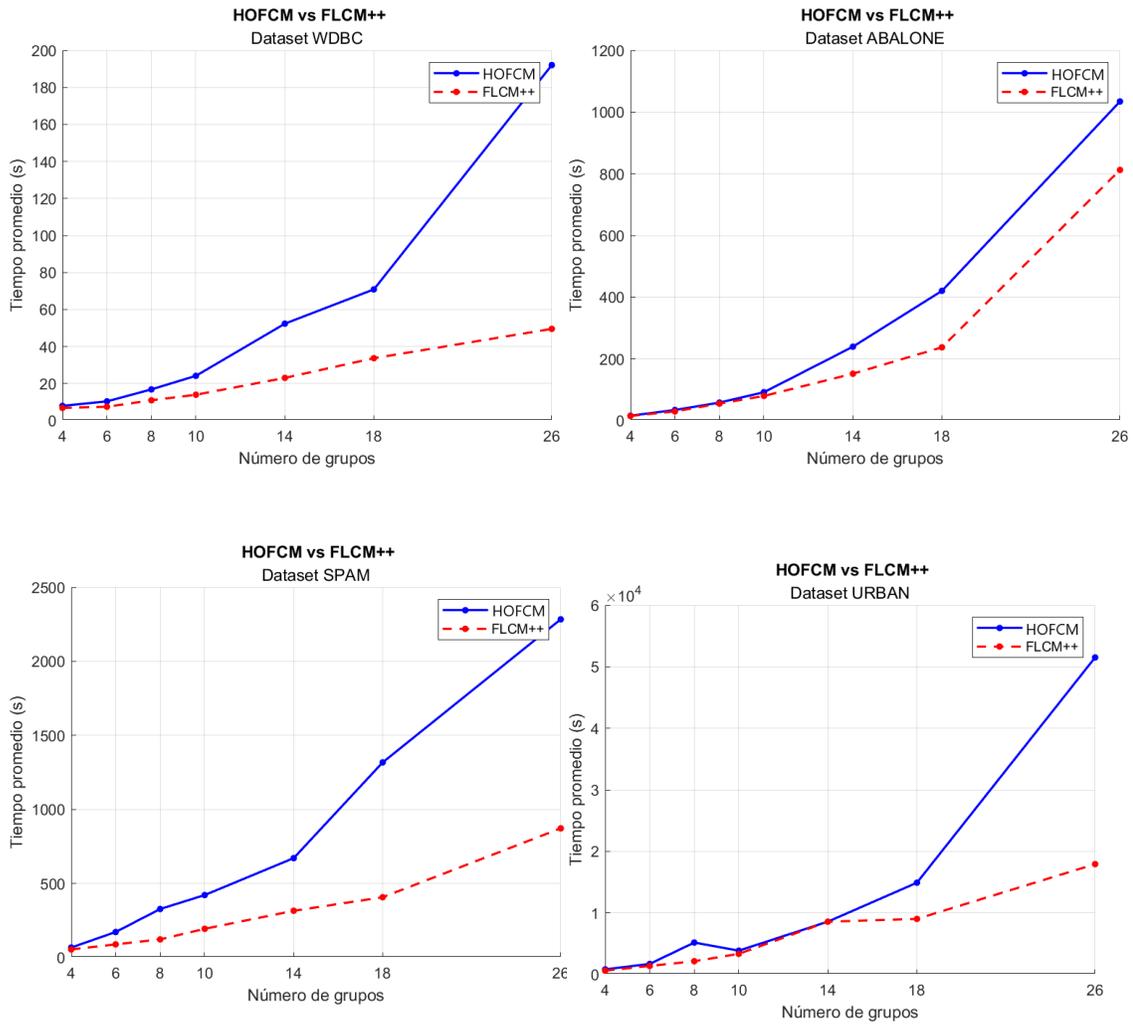


Figura 8. Resultados de HOFCM vs FLCM++ en relación con el tiempo de ejecución.

Tabla 8. Comparación entre HOFCM y FLCM++. Tiempo y Sumatoria de Distancias al Cuadrado

HOFCM VS FLCM++								
Conjunto de datos	p	c	Tiempo (s)			Sumatoria de Distancias al Cuadrado		
			HOFCM	FLCM++	τ %	HOFCM	FLCM++	E %
WDBC [28]	1	4	7.93	6.82	14.03	80,250,054.07	80,453,739.70	-0.25
	2	6	10.35	7.45	28.02	63,474,605.78	64,319,869.36	-1.33
	3	8	16.84	10.93	35.11	54,341,108.97	55,141,402.78	-1.47
	4	10	24.16	13.94	42.29	49,129,854.20	49,352,919.43	-0.45
	5	14	52.34	23.04	55.98	42,864,738.55	42,738,518.01	0.29
	6	18	70.87	33.69	52.46	39,569,569.41	37,951,222.12	4.09
	7	26	192.09	49.54	74.21	36,690,329.56	32,056,608.06	12.63
ABALONE [32]	1	4	15.63	14.84	5.06	479.10	479.53	-0.09
	2	6	34.19	29.54	13.59	373.30	374.44	-0.31
	3	8	58.18	54.79	5.83	324.81	325.34	-0.16
	4	10	91.83	79.78	13.12	299.76	301.26	-0.50
	5	14	239.46	151.89	36.57	275.60	277.14	-0.56
	6	18	420.48	237.45	43.53	265.79	267.59	-0.68
	7	26	1,034.50	812.49	21.46	252.59	257.03	-1.76
SPAM [33]	1	4	66.69	53.92	19.15	855,877,644.85	862,177,981.03	-0.74
	2	6	171.95	88.46	48.56	664,699,673.47	684,425,841.66	-2.97
	3	8	327.40	121.99	62.74	598,993,314.74	524,912,176.51	12.37
	4	10	421.51	193.60	54.07	586,779,307.95	400,430,527.95	31.76
	5	14	671.08	315.39	53.00	553,278,457.25	302,429,289.57	45.34
	6	18	1,317.76	406.99	69.12	535,392,355.33	247,783,899.00	53.72
	7	26	2,284.16	872.16	61.82	514,528,074.53	193,009,932.25	62.49
URBAN [34]	1	4	827.60	625.39	24.43	661,265.50	665,012.86	-0.57
	2	6	1,716.65	1,382.16	19.48	551,218.69	562,402.68	-2.03
	3	8	5,182.38	2,147.87	58.55	493,258.02	511,637.93	-3.73
	4	10	3,868.26	3,347.96	13.45	451,503.15	462,288.75	-2.39
	5	14	8,580.78	8,574.81	0.07	385,191.63	389,147.71	-1.03
	6	18	14,910.52	9,029.19	39.44	339,818.98	346,533.09	-1.98
	7	26	51,502.89	17,928.58	65.19	296,710.58	299,858.52	-1.06

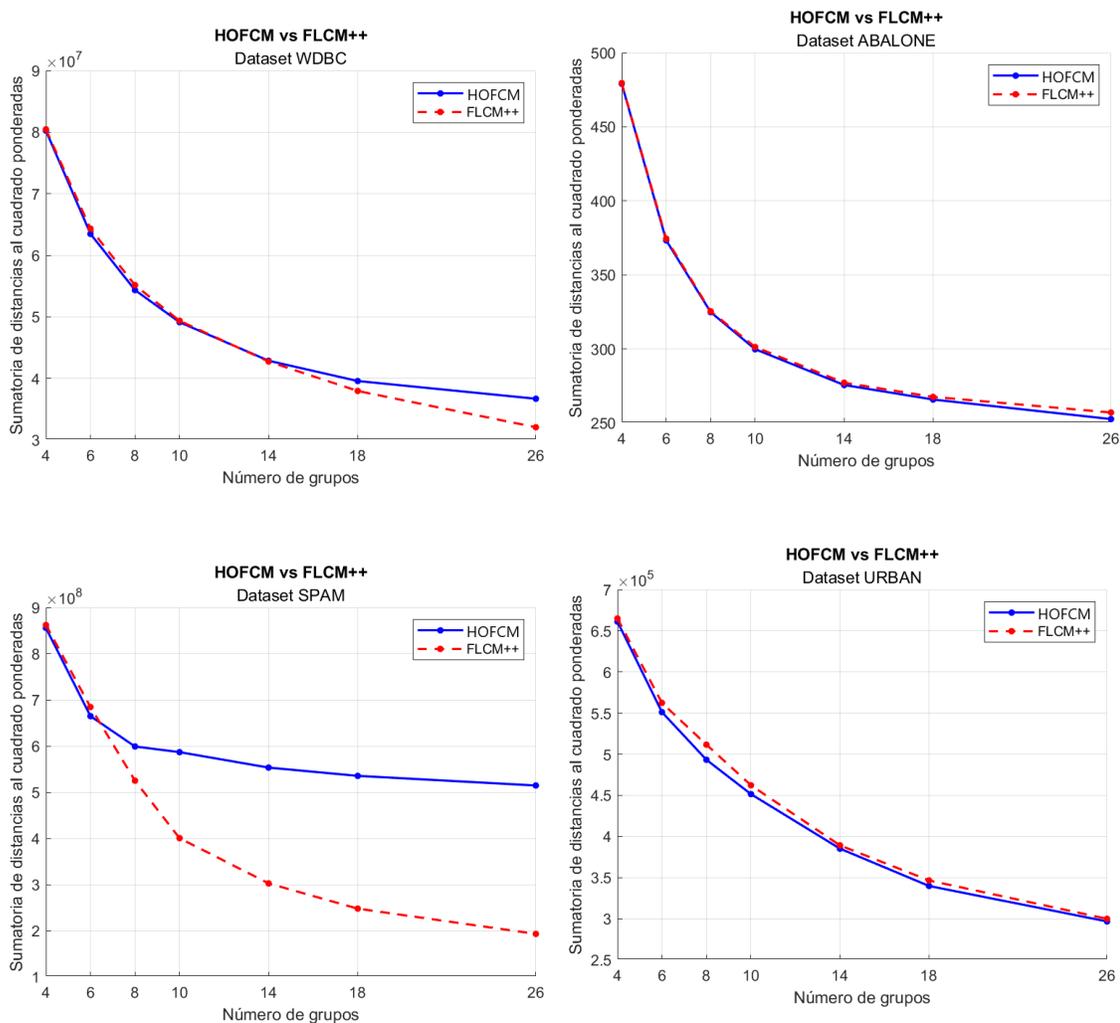


Figura 9. Resultados de HOFCM vs FLCM++ en relación con la sumatoria de las distancias al cuadrado.

Los resultados de las pruebas de Wilcoxon entre HOFM++ y FLCM++ muestran que, para el tiempo de ejecución, con un valor $p = 0.0000037896$, existen diferencias estadísticamente significativas que favorecen a FLCM++ en términos de eficiencia temporal. Sin embargo, para la sumatoria de distancias al cuadrado, con un valor $p = 0.69867$, no se encontraron diferencias significativas, lo que indica que ambos algoritmos ofrecen una calidad de partición comparable.

3.3. Pruebas realizadas de segmentación de imágenes

Este caso de prueba, realizó ejecuciones del algoritmo FLCM++ y los resultados se compararon con los del algoritmo FCM estándar. Los experimentos se llevaron a cabo utilizando valores de $c = 3, 4, 5$ y 6 , mientras que el parámetro δ se fijó en $\delta = 0.5$. Adicionalmente, para los parámetros del algoritmo FCM, se emplearon un valor de $m = 2$ y un criterio de convergencia $\varepsilon = 0.01$.

En estas pruebas, se llevaron a cabo 50 ejecuciones por cada una de las configuraciones de c .

Las imágenes utilizadas se muestran en la Figura 10, corresponden a tomografías computarizadas de la región lumbar, capturadas mediante un escáner especializado en una clínica privada de Cd. Victoria, Tamaulipas. Este tipo de tomografía se emplea principalmente para obtener imágenes detalladas de estructuras internas del cuerpo, como huesos, tejidos y órganos, siendo fundamental en el diagnóstico y tratamiento de afecciones médicas. En este caso, se seleccionaron 15 imágenes representativas para evaluar el rendimiento del algoritmo FLCM++ en la segmentación de imágenes médicas.

Las imágenes utilizadas en este estudio corresponden a 15 tomografías computarizadas de tipo T1. Las tomografías T1 [35] son imágenes médicas obtenidas mediante resonancia magnética que destacan por su capacidad de diferenciar entre tejidos blandos, mostrando un alto contraste entre grasa, líquidos y otras estructuras. Este tipo de imágenes es especialmente útil para analizar la anatomía detallada de la columna vertebral y detectar posibles anomalías. En el contexto de este estudio, estas tomografías fueron seleccionadas como muestra representativa para evaluar la eficacia del algoritmo FLCM++ en la segmentación de imágenes médicas, centrando el análisis en la calidad del agrupamiento y el tiempo de procesamiento requerido.

El algoritmo de agrupamiento, diseñado para la segmentación de imágenes, se enfoca en la separación de los objetos según su afinidad, basándose en propiedades como la similitud de los niveles de intensidad. El objetivo es agrupar píxeles que comparten características similares, permitiendo distinguir claramente las diferentes regiones o estructuras presentes en la imagen.

Un ejemplo de los resultados de segmentar imágenes empleando agrupamiento se muestra en la Figura 11, tomando en cuenta un valor de $c = 4$.

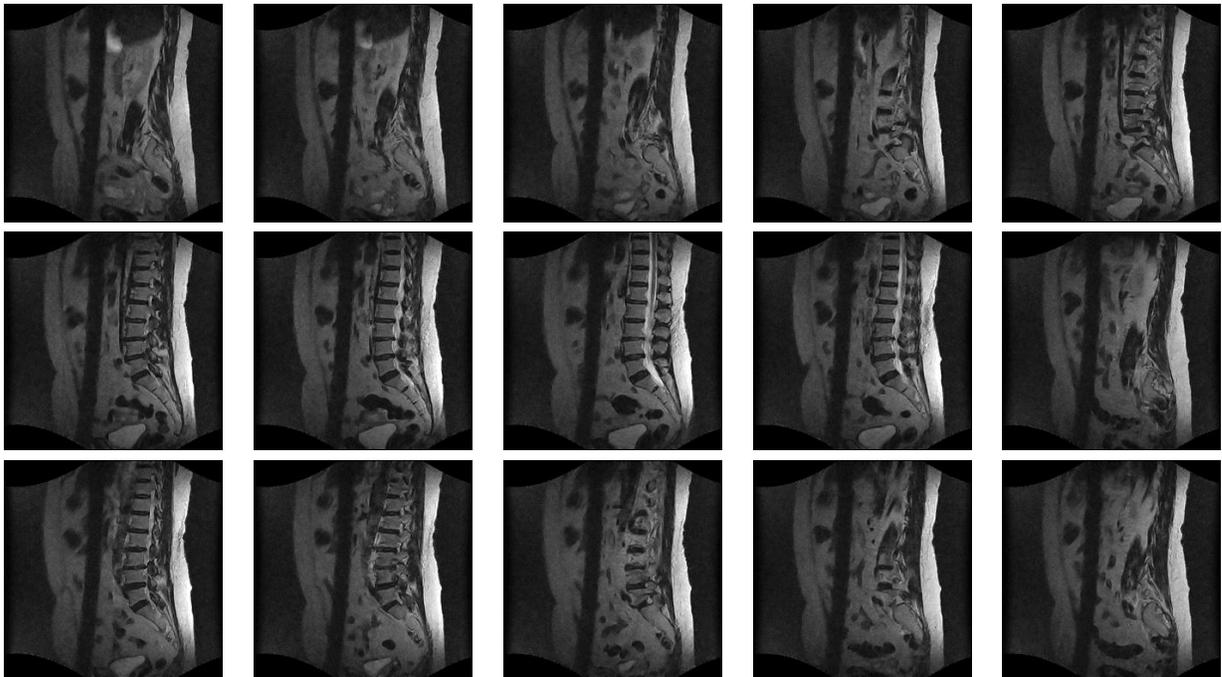


Figura 10. Imágenes de tomografías computarizadas de un paciente, empleadas en la experimentación de segmentación de imágenes.

Las Tablas 9 y 10 presentan los resultados de las pruebas realizadas con el algoritmo FLCM++ aplicado a la segmentación de imágenes, comparando su rendimiento frente al algoritmo FCM estándar. Analizando dos aspectos clave: el tiempo de procesamiento (en segundos) y la sumatoria de distancias al cuadrado, reflejando la calidad de la segmentación. Adicionalmente, se calculan los porcentajes de mejora o reducción para cada caso, con las siguientes consideraciones:

- En cuanto al tiempo, un porcentaje positivo indica una reducción en el tiempo promedio, mientras que un porcentaje negativo implica un aumento.
- En cuanto a la sumatoria de distancias al cuadrado, un porcentaje positivo indica una ganancia en calidad, mientras que un porcentaje negativo implica una pérdida en calidad.

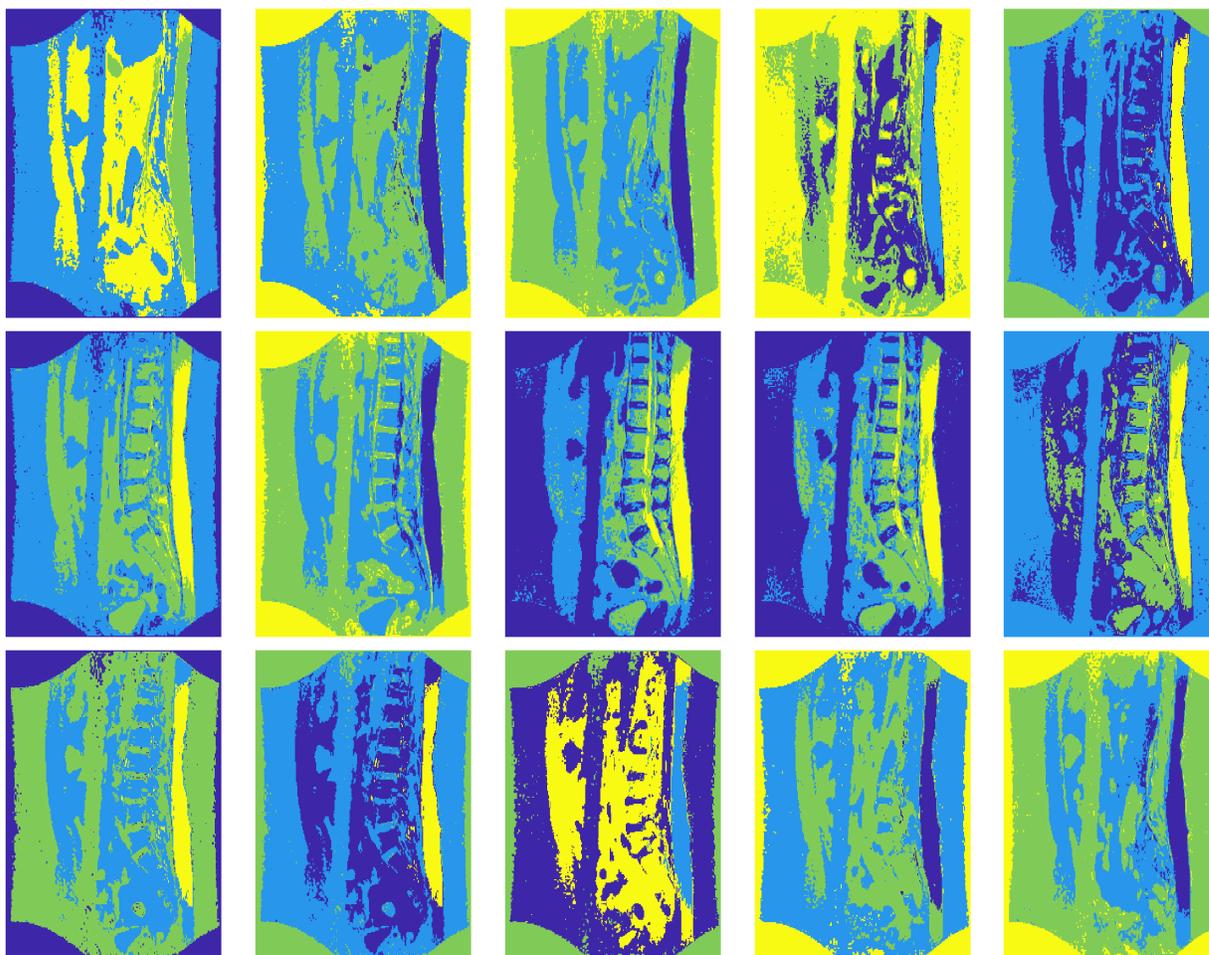


Figura 11. Resultados al segmentar resonancias magnéticas aplicando el algoritmo FLCM++, empleando un valor de $c = 4$

En la Tabla 9 se muestran los resultados para las imágenes *Imgs0002* a *Imgs0008* destacan una reducción significativa en el tiempo de procesamiento al usar FLCM++ en cuanto a la configuración de $c = 6$, con valores de mejora superiores al 95%. Sin embargo, en algunos casos, la calidad de segmentación

medida por la sumatoria de distancias no siempre mejora, mostrando pérdidas en calidad, como en la imagen *Imgs0002*, donde se registra una mejora en tiempo del 95.75 % pero una pérdida en calidad del 7.06 %, sin embargo, en otros casos como en la *Imgs0006* se obtiene un reducción del tiempo del 95.86 % y una ganancia en la calidad del agrupamiento del 6.22 %.

En la Tabla 10 se incluyen los resultados de las pruebas de las imágenes *Imgs0009* a *Imgs0016* bajo condiciones similares. Nuevamente, el algoritmo FLCM++ logra reducir los tiempos de procesamiento en más del 80 % en la mayoría de los casos. En términos de la sumatoria de distancias al cuadrado, existe una ganancia de la calidad del agrupamiento de hasta el 3.84 %. Sin embargo, persisten configuraciones donde la mejora en tiempo no viene acompañada de una mejor segmentación, como en *Imgs0014* con $c = 6$, donde la mejora en tiempo es del 93.48 %, pero una pérdida en la calidad del 4.59 %.

■ Mejoras en Tiempo:

- La configuración de $c = 6$ en la Tabla 9 y 10 reduce el tiempo en más del 90 %, subrayando la eficiencia del FLCM++ para grandes números de clústeres.
- La imagen *Imgs0002* logra la mayor mejora en tiempo, con una reducción del 95.75 %.

■ Calidad de Segmentación:

- La mayoría de las pruebas muestran que el FLCM++ no afecta significativamente la calidad de segmentación en términos de la sumatoria de distancias.
- Las imágenes *Imgs0006* y *Imgs0008* presentan las mayores ganancias en calidad, con 6.22 % y 4.07 %, respectivamente.
- Algunos casos muestran pérdidas en la calidad de segmentación, como en *Imgs0002* y *Imgs0004*, mostrando pérdidas del 7.06 % y 6.15 %, respectivamente.

Tabla 9. Pruebas realizadas con el algoritmo FLCM++ para la segmentación de las imágenes comprendidas entre Imagen0002 e Imagen0008

Pruebas de FLCM++ aplicado a segmentación de imágenes								
Imagen	c	p	Tiempo (s)			Sumatoria de Distancias al Cuadrado		
			FCM	FLCM++	τ %	FCM	FLCM++	E %
Imgs0002	3	4	106.68	89.59	16.02	11,414,822.08	11,459,591.70	-0.39
	4	6	336.03	98.07	70.82	14,458,919.56	14,469,735.45	-0.07
	5	8	612.86	112.11	81.71	17,812,711.60	17,905,581.98	-0.52
	6	10	2,943.43	125.06	95.75	20,771,864.07	22,238,912.96	-7.06
Imgs0003	3	4	106.35	89.26	16.07	12,137,687.28	12,075,619.92	0.51
	4	6	534.86	99.61	81.38	15,254,790.02	15,395,934.52	-0.93
	5	8	540.67	112.27	79.24	18,810,660.54	18,864,146.65	-0.28
	6	10	2,944.89	129.11	95.62	22,017,515.73	22,834,084.66	-3.71
Imgs0004	3	4	89.85	88.67	1.31	12,692,470.34	12,715,348.33	-0.18
	4	6	344.44	99.25	71.18	15,719,299.92	15,748,724.18	-0.19
	5	8	645.65	116.72	81.92	19,390,855.65	19,281,888.20	0.56
	6	10	1,880.50	130.29	93.07	22,192,859.25	23,558,678.52	-6.15
Imgs0005	3	4	97.71	89.16	8.75	13,342,750.03	13,301,392.57	0.31
	4	6	500.15	99.84	80.04	16,441,864.97	16,553,943.01	-0.68
	5	8	594.57	118.84	80.01	19,883,104.68	20,069,134.46	-0.94
	6	10	2,961.51	132.91	95.51	22,832,980.51	24,277,182.36	-6.33
Imgs0006	3	4	93.18	89.11	4.37	14,045,666.75	14,091,033.44	-0.32
	4	6	411.11	99.07	75.90	17,070,668.29	17,152,794.08	-0.48
	5	8	807.66	113.57	85.94	20,543,312.70	20,728,992.82	-0.90
	6	10	3,018.76	125.12	95.86	28,134,415.06	26,384,185.79	6.22
Imgs0007	3	4	90.46	89.31	1.27	14,278,316.80	14,311,749.33	-0.23
	4	6	386.80	98.87	74.44	17,322,218.78	17,581,047.76	-1.49
	5	8	894.30	110.33	87.66	21,039,242.48	22,372,489.68	-6.34
	6	10	2,207.99	130.12	94.11	26,908,747.31	27,641,967.68	-2.72
'Imgs0008'	3	4	94.32	88.89	5.76	14,099,740.78	14,175,398.92	-0.54
	4	6	637.44	98.41	84.56	17,272,025.10	18,486,845.24	-7.03
	5	8	844.86	115.91	86.28	23,888,809.49	22,916,232.65	4.07
	6	10	1,259.36	137.32	89.10	28,269,734.45	28,735,079.57	-1.65

Tabla 10. Pruebas realizadas con el algoritmo FLCM++ para la segmentación de las imágenes comprendidas entre Imagen0009 e Imagen0016

Pruebas de FLCM++ aplicado a segmentación de imágenes								
Imagen	c	p	Tiempo (s)			Sumatoria de Distancias al Cuadrado		
			FCM	FLCM++	τ %	FCM	FLCM++	E %
'Imgs0009'	3	4	90.79	89.23	1.72	14,340,719.06	14,366,468.33	-0.18
	4	6	739.86	98.60	86.67	17,825,633.51	18,702,549.60	-4.92
	5	8	1,168.80	114.45	90.21	22,260,024.50	23,322,693.89	-4.77
	6	10	1,502.65	128.53	91.45	29,221,816.55	28,098,965.52	3.84
'Imgs0010'	3	4	93.24	89.04	4.51	14,261,704.76	14,374,497.56	-0.79
	4	6	971.00	98.34	89.87	17,646,817.85	18,774,506.41	-6.39
	5	8	1,793.46	111.59	93.78	22,713,234.77	23,702,436.44	-4.36
	6	10	1,195.32	129.88	89.13	28,654,868.93	29,125,770.26	-1.64
'Imgs0011'	3	4	99.23	88.85	10.46	14,361,691.83	14,427,245.06	-0.46
	4	6	735.20	99.23	86.50	17,535,876.87	17,798,499.35	-1.50
	5	8	695.10	114.09	83.59	21,434,574.69	21,700,794.28	-1.24
	6	10	1,800.84	127.95	92.90	27,974,195.03	27,186,951.22	2.81
'Imgs0012'	3	4	86.28	88.72	-2.83	14,575,612.25	14,548,129.57	0.19
	4	6	429.69	100.46	76.62	17,672,644.57	18,099,630.31	-2.42
	5	8	764.27	112.64	85.26	21,362,584.68	21,822,467.11	-2.15
	6	10	2,265.20	125.60	94.46	25,837,689.66	26,871,839.09	-4.00
'Imgs0013'	3	4	88.74	88.79	-0.06	14,065,950.48	14,146,297.14	-0.57
	4	6	519.05	99.98	80.74	17,154,822.32	17,270,834.02	-0.68
	5	8	626.87	116.00	81.49	20,778,136.12	20,822,101.80	-0.21
	6	10	2,113.54	127.64	93.96	23,940,603.93	24,925,146.51	-4.11
'Imgs0014'	3	4	92.77	88.78	4.29	13,747,654.73	13,752,696.58	-0.04
	4	6	390.16	98.67	74.71	16,774,466.72	16,961,426.25	-1.11
	5	8	587.12	111.42	81.02	20,378,078.32	20,376,596.96	0.01
	6	10	1,939.74	126.41	93.48	23,327,602.46	24,397,593.35	-4.59
'Imgs0015'	3	4	115.25	89.09	22.70	12,534,358.30	12,669,648.28	-1.08
	4	6	393.21	100.50	74.44	15,636,519.35	15,803,424.72	-1.07
	5	8	615.64	115.67	81.21	19,228,204.61	19,393,125.10	-0.86
	6	10	2,007.28	129.54	93.55	22,341,263.29	22,975,952.55	-2.84
'Imgs0016'	3	4	89.45	90.56	-1.24	14,029,268.96	14,176,710.47	-1.05
	4	6	513.23	100.99	80.39	17,420,391.77	17,743,929.57	-1.86
	5	8	751.28	114.83	84.62	20,982,561.90	21,022,141.24	-0.19
	6	10	2,213.94	126.79	93.34	23,456,781.93	23,673,317.19	-0.92

4. Resultados obtenidos

La presente investigación ha demostrado que las modificaciones propuestas en el algoritmo FCM, específicamente el desarrollo del algoritmo FLCM++, genera beneficios significativos en términos de tiempo de ejecución y calidad de agrupamiento en comparación con las versiones mejoradas previamente publicadas, como FCM++ y HOFCM.

Este capítulo se encuentra distribuido de la siguiente forma: la Sección 6.1 muestra las conclusiones de los resultados obtenidos en los conjuntos de datos numéricos. En la Sección 6.2 se describen las conclusiones a las que se llegó con base en los resultados de la segmentación de imágenes. Finalmente, en la Sección 6.3 se plantean las conclusiones generales a las que se llegó.

4.1. Resultados de Conjuntos de Datos Numéricos

En esta sección se presentan las conclusiones obtenidas para el conjunto de datos numéricos que fueron puestos a prueba, destacando los mejores resultados. Así mismo, se mencionan aquellos casos en los que hubo alguna pérdida en la calidad del agrupamiento.

Desempeño de FLCM++ frente a FCM

A continuación, se presentan los aspectos más relevantes basados en los resultados obtenidos:

- **Reducción en el tiempo de ejecución:** FLCM++ logró una reducción significativa en el tiempo promedio de ejecución en comparación con FCM. En las pruebas realizadas, la reducción máxima alcanzada fue del 90.38 %, observada en el conjunto de datos SPAM [33], mientras que la reducción mínima fue del 7.77 % en el conjunto de datos ABALONE [32]. Esto demuestra la alta eficiencia computacional del FLCM++ en diferentes escenarios.
- **Calidad del agrupamiento:** En términos de la sumatoria de las distancias al cuadrado, FLCM++ mostró un comportamiento competitivo. En la mayoría de los casos, la calidad del agrupamiento se mantuvo similar o mejor que con FCM. La mayor ganancia en calidad, de hasta 66.39 %, también se observó en el conjunto de datos SPAM [33]. Sin embargo, en escenarios desafiantes como el conjunto URBAN [34], se registró una pérdida de calidad máxima de 2.27 %.
- **Consistencia en diferentes configuraciones:** Los resultados obtenidos indican que FLCM++ es robusto en diversas configuraciones de centroides y tamaños de datos. Reduce el tiempo de ejecución sin comprometer significativamente la calidad del agrupamiento lo posiciona como una alternativa efectiva frente a FCM.

Desempeño de FLCM++ frente a FCM++

- **Reducción en el tiempo de ejecución:** FLCM++ logra disminuir significativamente el tiempo requerido para procesar los conjuntos de datos evaluados. En el mejor de los casos, se alcanzó una reducción del 76.38 %, destacando la eficiencia del algoritmo incluso frente a una mejora ya publicada por Stetco y colab. [36] como FCM++.

- **Calidad del agrupamiento:** En algunos escenarios se observaron ligeras pérdidas en la calidad del agrupamiento, sin embargo, estas no superan el 2.03%. En otros casos, FLCM++ mostró mejoras, alcanzando un incremento del 3.57% en la sumatoria de las distancias al cuadrado, lo que evidencia un desempeño competitivo en términos de precisión.

Desempeño de FLCM++ frente a HOFCM

- **Comparativa en tiempo de ejecución:** Al compararse con el algoritmo HOFCM propuesto en 2022 por Pérez y colab. en [21], FLCM++ también demostró ser más eficiente, logrando una reducción de hasta un 74.21% en el mejor caso.
- **Calidad del agrupamiento:** En cuanto a la sumatoria de las distancias al cuadrado, FLCM++ mostró un comportamiento competitivo. Aunque en el peor escenario se reportó una pérdida máxima del 3.73%, en otros casos alcanzó mejoras significativas, como un aumento del 62.49% en la calidad del agrupamiento para ciertos conjuntos de datos.

Ventajas de FLCM++ en diversos escenarios

- FLCM++ mostró un comportamiento consistente en la mayoría de los conjuntos de datos evaluados, destacándose en escenarios con alta dimensionalidad y diferentes configuraciones de parámetros.
- La reducción en el tiempo de ejecución sin comprometer significativamente la calidad del agrupamiento lo posiciona como una opción eficiente para tareas que requieren un balance entre precisión y rapidez.

4.2. Segmentación de Imágenes

En esta sección se muestran las conclusiones a las que se llegó en la aplicación del algoritmo FLCM++ como estudio de caso, en donde fue sometido a un conjunto imágenes de tomografías computarizadas.

Caso de Prueba para el Algoritmo FLCM++

Las pruebas comparativas entre los algoritmos FCM estándar y FLCM++ fueron enfocadas en la segmentación de imágenes médicas. Los parámetros y configuraciones utilizadas fueron $c = 3, 4, 5, 6$, $\delta = 0.5$, $m = 2$ y $\varepsilon = 0.01$, con 50 ejecuciones por configuración.

Las imágenes utilizadas provienen de tomografías computarizadas de la columna lumbar, como se muestra en la Figura 10. Los resultados, presentados en las Tablas 9 y 10, evalúan dos aspectos principales: tiempo de procesamiento y calidad de la segmentación.

*Conclusiones para Segmentación de Imágenes

- **Eficiencia en Tiempo:** El algoritmo FLCM++ logró una reducción del tiempo de procesamiento superior al 90% en configuraciones con $c = 6$, destacando su utilidad en imágenes de alta resolución.

- **Calidad de Segmentación:** Aunque en la mayoría de los casos la calidad se mantuvo estable, algunos resultados mostraron pequeñas pérdidas de calidad en términos de la sumatoria de distancias al cuadrado.
- **Casos Destacados:**
 - La imagen `Imgs0006` mostró la mayor ganancia en calidad del 6.22 %, junto con una reducción de tiempo del 95.86 %.
 - En imágenes como `Imgs0002`, aunque hubo una mejora del 95.75 % en tiempo, se registró una pérdida de calidad del 7.06 %.

4.3. Conclusiones Generales

La mejora del algoritmo FCM propuesta, denominada FLCM++, se centra en optimizar dos aspectos cruciales: el tiempo de ejecución y la convergencia hacia soluciones globales en lugar de óptimos locales. Las pruebas realizadas con las instancias `Ecoli` [27] y `WDBC` [28] muestran que el algoritmo FCM, a pesar de ser un enfoque robusto para el agrupamiento difuso, presenta limitaciones claras en términos de eficiencia computacional y sensibilidad a la inicialización de los centroides.

1. **Optimización del tiempo de ejecución:** El enfoque de FLCM++ consiste en la identificación y el tratamiento de los objetos inalterados, aquellos cuya asignación de grupo no cambia entre iteraciones. Este enfoque permite reducir significativamente los cálculos innecesarios en iteraciones posteriores, contribuyendo a una mayor eficiencia sin sacrificar la calidad del resultado. Las pruebas realizadas, tanto con la instancia `Ecoli`[27] como con la `WDBC` [28], evidencian que después de un número determinado de iteraciones, la mayoría de los objetos no cambian de grupo, lo que sugiere que el algoritmo podría detenerse antes de las iteraciones finales sin perder precisión, optimizando así el tiempo de procesamiento.
2. **Reducción de la influencia de la inicialización de los centroides:** La implementación del algoritmo `k-Means++` como parte de la propuesta FLCM++ ha mostrado ser una mejora significativa en términos de la inicialización de los centroides. Este enfoque híbrido asegura una distribución con mejoras en la uniformidad de los centroides iniciales, lo que mejora la convergencia y minimiza la posibilidad de que el algoritmo se quede atrapado en óptimos locales. La mejora de la convergencia es evidente en los resultados obtenidos, ya que las iteraciones iniciales muestran un descenso rápido en la función objetivo, seguido de una estabilización gradual, lo que confirma la efectividad del `k-Means++` en la optimización del proceso.
3. **Convergencia del algoritmo FCM:** En las pruebas realizadas, se observó que el algoritmo FCM, tanto en la instancia `Ecoli` [27] como en la `WDBC` [28], converge rápidamente en las primeras iteraciones, con un cambio mínimo en la función objetivo y en los cambios de grupo después de ciertos puntos de iteración. Este comportamiento justifica la optimización del algoritmo FLCM++, que permite detener las iteraciones sin comprometer la calidad del resultado final, lo cual reduce el tiempo de cómputo de manera sustancial.
4. **Recomendaciones para futuras investigaciones:** Se sugiere continuar con la mejora de la estrategia de manejo de objetos inalterados, no solo limitando el cálculo de los grados de pertenencia,

sino también considerando la incorporación de otras técnicas heurísticas para identificar patrones adicionales de estabilidad en el algoritmo. Además, la integración de técnicas híbridas con algoritmos como k-Means++ puede ser explorada aún más en otros métodos de clustering difuso, ampliando su aplicabilidad a diferentes tipos de datos y problemáticas.

La prueba de Wilcoxon indicó que las diferencias en los tiempos de ejecución entre FCM y FLCM++ son estadísticamente significativas ($p = 0.0000038 < 0.05$), lo que sugiere que FLCM++ es significativamente más rápido que FCM en los conjuntos de datos analizados.

En cuanto a la calidad de las particiones (sumatoria de distancias al cuadrado), el valor $p = 0.083518$ no permitió rechazar la hipótesis nula, lo que indica que no se encontraron diferencias estadísticamente significativas en la calidad de las particiones generadas por FCM y FLCM++.

Al comparar HOFM++ y FLCM++, para el tiempo de ejecución, el valor $p = 0.0000037896$ mostró diferencias estadísticamente significativas, favoreciendo a FLCM++ en términos de eficiencia temporal. Sin embargo, en la métrica de calidad de las particiones ($p = 0.69867$), no se encontraron diferencias significativas, lo que sugiere que ambos algoritmos tienen un rendimiento comparable en este aspecto.

Referencias

- [1] L. Rokach and O. Maimon, *Clustering Methods*. Boston, MA: Springer US, 2005, pp. 321–352. [Online]. Available: https://doi.org/10.1007/0-387-25465-X_15
- [2] E. Diday and J. C. Simon, *Clustering Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 47–94. [Online]. Available: https://doi.org/10.1007/978-3-642-96303-2_3
- [3] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy cluster analysis: methods for classification, data analysis and image recognition*. John Wiley & Sons, 1999.
- [4] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.
- [5] H. Mittal, A. C. Pandey, M. Saraswat, S. Kumar, R. Pal, and G. Modwel, “A comprehensive survey of image segmentation: clustering methods, performance parameters, and benchmark datasets,” *Multimedia Tools and Applications*, pp. 1–26, 2021.
- [6] H. Verma, D. Verma, and P. K. Tiwari, “A population based hybrid fcm-pso algorithm for clustering analysis and segmentation of brain image,” *Expert systems with applications*, vol. 167, p. 114121, 2021.
- [7] S. Ramasubbareddy, T. A. S. Srinivas, K. Govinda, and S. Manivannan, “Comparative study of clustering techniques in market segmentation,” *Innovations in Computer Science and Engineering: Proceedings of 7th ICICSE*, pp. 117–125, 2020.
- [8] H. Zou, “Clustering algorithm and its application in data mining,” *Wireless Personal Communications*, vol. 110, no. 1, pp. 21–30, 2020.
- [9] A. Mexicano, R. Rodriguez, S. Cervantes, P. Montes, M. Jimenez, N. Almanza, and A. Abrego, “The early stop heuristic: a new convergence criterion for k-means,” in *AIP conference proceedings*, vol. 1738, no. 1. AIP Publishing, 2016.

- [10] P. J. Kaur *et al.*, “Cluster quality based performance evaluation of hierarchical clustering method,” in *2015 1st International conference on next generation computing technologies (NGCT)*. IEEE, 2015, pp. 649–653.
- [11] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, “Efficient agglomerative hierarchical clustering,” *Expert Systems with Applications*, vol. 42, no. 5, pp. 2785–2797, 2015.
- [12] Y. Yang, C. Qian, H. Li, Y. Gao, J. Wu, C.-J. Liu, and S. Zhao, “An efficient dbSCAN optimized by arithmetic optimization algorithm with opposition-based learning,” *The Journal of Supercomputing*, vol. 78, no. 18, pp. 19 566–19 604, 2022.
- [13] B. Gondaliya, “Review paper on clustering techniques,” *International Journal of Engineering Technology, Management and Applied Sciences*, vol. 2, no. 7, pp. 234–237, 2014.
- [14] A. Vardhan, P. Sarmah, and A. Das, “A comprehensive analysis of the most common hard clustering algorithms,” in *Inventive Computation Technologies 4*. Springer, 2020, pp. 48–58.
- [15] B. Bede and B. Bede, “Fuzzy clustering,” *Mathematics of Fuzzy Sets and Fuzzy Logic*, pp. 213–219, 2013.
- [16] V. K. Malhotra, H. Kaur, and M. A. Alam, “An analysis of fuzzy clustering methods,” *International Journal of Computer Applications*, vol. 94, no. 19, pp. 9–12, 2014.
- [17] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. New York, NY, USA: Springer Science & Business Media, 2013.
- [18] L. Pang, K. Xiao, A. Liang, and H. Guan, “A improved clustering analysis method based on fuzzy c-means algorithm by adding PSO algorithm,” in *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012, Salamanca, Spain, March 28-30th, 2012. Proceedings, Part I 7*. Springer, 2012, pp. 231–242.
- [19] S. Ghosh and S. K. Dubey, “Comparative analysis of k-means and fuzzy c-means algorithms,” *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 4, 2013.
- [20] D. J. Bora and D. A. K. Gupta, “A comparative study between fuzzy clustering algorithm and hard clustering algorithm,” *arXiv preprint arXiv:1404.6059*, 2014.
- [21] J. Pérez-Ortega, S. S. Roblero-Aguilar, N. N. Almanza-Ortega, J. Frausto Solís, C. Zavala-Díaz, Y. Hernández, and V. Landero-Nájera, “Hybrid fuzzy c-means clustering algorithm oriented to big data realms,” *Axioms*, vol. 11, no. 8, p. 377, 2022.
- [22] N. Radwan, “Big data ethics,” *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 19, no. 1, 2021.
- [23] J. Wall and T. Krummel, “The digital surgeon: How big data, automation, and artificial intelligence will change surgical practice,” *Journal of Pediatric Surgery*, vol. 55, pp. 47–50, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0022346819306542>
- [24] V. W. Ajin and L. D. Kumar, “Big data and clustering algorithms,” in *2016 International Conference on Research Advances in Integrated Navigation Systems (RAINS)*, 2016, pp. 1–5.
- [25] S. M. Sait and H. Youssef, “Iterative computer algorithms: and their applications in engineering,” 1999.

- [26] S. Sait and H. Youssef, *VLSI physical design automation: theory and practice*. World Scientific, 1999, vol. 6.
- [27] K. Nakai, "Ecoli," UCI Machine Learning Repository, 1996, DOI: <https://doi.org/10.24432/C5388M>.
- [28] M. O. S. N. Wolberg, William and W. Street, "Breast Cancer Wisconsin (Diagnostic)," UCI Machine Learning Repository, 1993, DOI: <https://doi.org/10.24432/C5DW2B>.
- [29] C. Merz, P. Murphy, and D. Aha. (1992) Uci repository of machine learning databases. University of California, Department of Information and Computer Science. Irvine, CA. [Online]. Available: <https://archive.ics.uci.edu/datasets>
- [30] D. Arthur, S. Vassilvitskii *et al.*, "k-means++: The advantages of careful seeding," in *Soda*, vol. 7, 2007, pp. 1027–1035.
- [31] J. Pérez-Ortega, C. D. Rey-Figueroa, S. S. Roblero-Aguilar, N. N. Almanza-Ortega, C. Zavala-Díaz, S. García-Paredes, and V. Landero-Nájera, "Pofcm: A parallel fuzzy clustering algorithm for large datasets," *Mathematics*, vol. 11, no. 8, 2023. [Online]. Available: <https://www.mdpi.com/2227-7390/11/8/1920>
- [32] S. T. T. S. C. A. Nash, Warwick and W. Ford, "Abalone," UCI Machine Learning Repository, 1994, DOI: <https://doi.org/10.24432/C55C7W>.
- [33] R. E. F. G. Hopkins, Mark and J. Suermondt, "Spambase," UCI Machine Learning Repository, 1999, DOI: <https://doi.org/10.24432/C53G6X>.
- [34] "UrbanGB, urban road accidents coordinates labelled by the urban center," UCI Machine Learning Repository, 2019, DOI: <https://doi.org/10.24432/C5CD0F>.
- [35] H. F. Aguinaga, J. A. Rivera, L. J. Tamayo, M. Tobón, and R. C. Osorno Ch, "Tomografía axial computarizada y resonancia magnética para la elaboración de un atlas de anatomía segmentaria a partir de crio secciones axiales del perro," *Revista Colombiana de Ciencias Pecuarias*, vol. 19, no. 4, pp. 451–459, 2006.
- [36] A. Stetco, X.-J. Zeng, and J. Keane, "Fuzzy c-means++: Fuzzy c-means with effective seeding initialization," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7541–7548, 2015.