AGRUPACIÓN EFICIENTE DE PUNTOS CON K-MEANS: UN ANÁLISIS DE RENDIMIENTO ENTRE EL ENFOQUE DE FUERZA BRUTA Y KDTREE

Sergio Daniel Mogollon Caceres[1]
Paul Antony Parizaca Mozo[2]
School of Computer Science
Universidad Nacional de San Agustín de Arequipa
Arequipa, Perú
smogollon@unsa.edu.pe pparizaca@unsa.edu.pe

***Resumen* – Este estudio evalúa la eficiencia de la agrupación de puntos mediante K-means, centrándose en el rendimiento de dos enfoques distintos para la búsqueda de vecinos más cercanos (KNN): el uso de fuerza bruta y la implementación de KD-Tree. Se analiza detalladamente la construcción del árbol, la inserción de puntos y la búsqueda KNN en el contexto del algoritmo K-means. Los resultados experimentales revelan un rendimiento notablemente superior al emplear el KD-Tree en comparación con la estrategia de fuerza bruta, especialmente en conjuntos de datos extensos y espacios de alta dimensionalidad. Este hallazgo destaca la eficacia práctica de la utilización de KD-Tree para mejorar el rendimiento de K-means en la búsqueda de vecinos más cercanos, proporcionando una perspectiva valiosa para aplicaciones prácticas y sugiriendo posibles direcciones futuras para optimizaciones adicionales en este contexto.**

***Index Terms*—K-means, KDtree, KNN, Estructuras de Datos, Recursividad Procesamiento de nubes de puntos**

I. INTRODUCCION

En el contexto de la agrupación eficiente de puntos con K-means, la eficiencia computacional se convierte en un aspecto crucial, especialmente al considerar conjuntos de datos extensos. Este estudio se enmarca en la evaluación de dos enfoques para la búsqueda de vecinos más cercanos (KNN) dentro de K-means: el enfoque de fuerza bruta y la implementación de KD-Tree. Mientras que la introducción destaca la eficiencia del KD-Tree en la búsqueda de puntos cercanos, nuestro enfoque se orienta hacia la comparación de rendimiento específica en el contexto de K-means.

El KD-Tree, implementado desde cero en C++, es esencial para la optimización de las operaciones de búsqueda en espacios multidimensionales, una característica relevante para K-means. A través de funciones personalizadas de inserción, búsqueda y KNN, buscamos analizar cómo esta estructura de datos afecta el rendimiento de K-means en comparación con el enfoque de fuerza bruta.

Este estudio no solo explorará la eficiencia y el rendimiento del KD-Tree en el contexto de K-means, sino que también comparará estos resultados con el enfoque de fuerza bruta en la búsqueda de vecinos más cercanos. La implementación desde cero de funciones clave, como la inserción y la búsqueda, proporciona un control detallado sobre el funcionamiento interno del KD-Tree, permitiendo una evaluación exhaustiva y específica en el contexto de K-means.

A lo largo de este documento, se presentará una análisis detallado de la eficiencia de K-means con KD-Tree en contraste con el enfoque de fuerza bruta. Además, se discutirán posibles aplicaciones prácticas de esta optimización en el procesamiento de grandes conjuntos de datos multidimensionales utilizando K-means. Este trabajo busca contribuir al entendimiento de la agrupación eficiente de puntos con K-means, ofreciendo una perspectiva valiosa para la aplicación práctica en el análisis de datos de gran escala.

El presente artículo esta organizado de la siguiente manera. Comenzaremos con una revisión exhaustiva de trabajos previos en el ámbito de KD-Trees, contextualizando nuestro estudio en el estado del arte [II]. A continuación, exploraremos el proceso de implementacion del K-Means con sus dos variaciones, en fuerza bruta y con integracion con un KD-tree [III]. Se describe brevemente la importancia de KNN para el algoritmo de K-means en [IV]. Posteriormente, se proporcioná una visión detallada de la metodología que implementamos para la implementacion de K-means en [V]. La sección [VI] presenta los resultados derivados de nuestro análisis, incluyendo pruebas de rendimiento y evaluación de eficiencia. Finalmente, presentamos las conclusiones y se discuten posibles direcciones para futuras investigaciones en [VII].

II. ESTADO DEL ARTE

El estudio del rendimiento y eficiencia de las tecnicas de clustering ha sido abordado por diversos investigadores, cada uno aportando valiosas contribuciones a la comprensión y optimización de esta estructura de datos. Algunas de estas contribuciones destacadas han influido significativamente en el panorama actual de la investigación.

Arthur en su investigación [1] aborda el desafío de mejorar la eficacia de K-means mediante una técnica de siembra aleatoria. El método propuesto, denominado K-means++, demuestra ser (log k)-competitivo con la agrupación óptima. Los resultados experimentales muestran mejoras tanto en la velocidad como en la precisión de K-means, superando consistentemente a la versión estándar. K-means++ logra una

979-8-3503-2321-4/22/$31.00 © 2022 IEEE

reducción significativa en el valor potencial, en algunos casos por varias órdenes de magnitud, y muestra un tiempo de ejecución más rápido. La siembra D2, aunque ligeramente más lenta que la siembra uniforme, contribuye a una convergencia más rápida de la búsqueda local. La aplicación en conjuntos de datos del mundo real demuestra mejoras sustanciales en la precisión y velocidad, destacando la eficacia de la propuesta de Arthur en comparación con el enfoque convencional de fuerza bruta.

Zhao en su trabajo [2] aborda la importancia de K-means en el aprendizaje no supervisado, particularmente en el procesamiento de lenguaje natural. Se propone un nuevo método de cálculo de similitud basado en distancias ponderadas y euclidianas para mejorar las deficiencias del algoritmo K-means. Los experimentos muestran que este nuevo enfoque supera al algoritmo K-means en eficiencia, corrección y estabilidad. Zhao también destaca la aplicabilidad de K-means en diversos campos, como la gestión de relaciones con clientes y el análisis del rendimiento estudiantil. Además, se presenta una mejora adicional en el algoritmo K-means, denominado MW-K-Means, basado en densidad y distancia euclidiana ponderada, que demuestra una mejora en la eficiencia de la clasificación. La conclusión destaca la relevancia de estos métodos mejorados en el análisis de datos complejos y de gran escala, señalando la necesidad continua de investigar cómo aplicar estos algoritmos en conjuntos de datos masivos y de alta dimensionalidad.

Por su parte, Pham en [3] se centra en uno de los inconvenientes fundamentales del algoritmo K-means: la necesidad de especificar de antemano el número de clústeres, K. El autor revisa en detalle métodos existentes para la selección de K y discute los factores que inciden en esta elección. Propone una novedosa medida destinada a asistir en la selección de K, y concluye con un análisis exhaustivo de los resultados obtenidos al aplicar dicha medida para determinar el número óptimo de clústeres en diversos conjuntos de datos. Las conclusiones resaltan las deficiencias de los métodos convencionales para seleccionar el número de clústeres y la falta de información sobre el rendimiento del algoritmo. El nuevo método propuesto aborda estas limitaciones, considerando información que refleja el desempeño del algoritmo y sugiere múltiples valores de K para casos donde se requieren niveles variables de detalle en los resultados de agrupación. Aunque se reconoce su posible costo computacional en conjuntos de datos extensos, el método ha sido validado con éxito en diversos conjuntos de datos, instando a futuras investigaciones para evaluar su capacidad en escenarios con distribuciones de objetos más complejas. Este trabajo contribuye significativamente a la mejora de la selección de K en el contexto del algoritmo K-means y destaca la necesidad de investigaciones continuas para explorar su aplicabilidad en situaciones más desafiantes.

Estos trabajos previos establecen un sólido marco para la comprensión de k-means en la agrupacion de conjuntos de puntos, sirviendo como base para la investigación actual y futura en este campo [4].

III. Implementacion de K-means

En esta sección, describimos la implementación de K-means en nuestro estudio de agrupación eficiente de puntos. Comenzamos con la construcción de clústeres mediante el enfoque tradicional de fuerza bruta, detallando los pasos de inicialización, asignación de puntos y actualización de centroides. Luego, presentamos la integración del KD-Tree en K-means, explicando la construcción del árbol KD y su función en la optimización de las operaciones de búsqueda de vecinos mas cercanos.

Para ambos casos necesitamos de una funcion que actualice los centroides al final de cada iteracion en base a la media de los puntos de cada cluster. El algoritmo de la implementacion de esta funcion se presenta a continuacion, asimismo la implementacion de esta funcion fue implementada en C++:

Algorithm 1 newCenters: Actualización de Centroides

Require: $clusteres$: Lista de clústeres, cada uno representado como una lista de puntos 2D
Ensure: Actualización de los centroides
  $newCentroides \leftarrow \{\}$
  for $i \leftarrow 0$ **to** (longitud de $clusteres$) $- 1$ **do**
    $newCentroide \leftarrow \{0, 0\}$
    for all $point\ clusteres[i]$ **do**
      $newCentroide \leftarrow newCentroide + point$
    end for
    $size \leftarrow$ longitud de $clusteres[i]$
    if $size \neq 0$ **then**
      $newCentroide \leftarrow newCentroide/size$
    end if
    Agregar $newCentroide$ a $newCentroides$
  end for
  Actualizar los centroides con $newCentroides$ =0

Fuente: Elaboracion Propia

A. Enfoque por Fuerza Bruta

La implementación del enfoque por fuerza bruta en el algoritmo K-means comienza con la inicialización de centroides. Seleccionamos k puntos aleatorios del conjunto de datos como centroides iniciales. En nuestra implementacion calculamos previamente un boundary (rectangulo que representa los limites de la nube de puntos) para que los centroides iniciales no esten demasiado alejados de los puntos. Posteriormente, se realiza una asignación iterativa de cada punto al clúster cuyo centroide esté más cercano, calculando las distancias euclidianas. Después de la asignación, se actualizan los centroides recalculando sus posiciones como la media de los puntos pertenecientes al respectivo clúster. Este proceso de asignación y actualización se repite hasta que los centroides convergen o hasta que se alcanza un número predefinido de iteraciones. El siguiente algoritmo empleado en la investigacion describe los pasos mencionados:

Algorithm 2 KMeans_BruteForce: Algoritmo K-Means con Fuerza Bruta

Require: all_points: Lista de puntos 2D a agrupar, $cont$: Contador de iteraciones
Ensure: Lista de clústeres resultante
  Crear un KD-Tree $fb\_centroides$
  for all $row\ centroides$ **do**
    $fb\_centroides$.insert(row)
  end for
  Inicializar vector de clústeres $clusters$ con $centroides.size()$ listas vacías
  for $i \leftarrow 0$ **to** longitud de $all\_points - 1$ **do**
    $num \leftarrow fb\_centroides.KNNBruteForce3(all\_points[i], 1)$

    for $j \leftarrow 0$ **to** longitud de $centroides - 1$ **do**
      if $num[0] == centroides[j]$ **then**
        Agregar $all\_points[i]$ a $clusters[j]$
        Romper el bucle
      end if
    end for
  end for
  Llamar a $newCenters(clusters)$ para actualizar los centroides
  if $cont ==$ maxIterations **then**
    return $clusters$
  end if
  Incrementar $cont$
  return $KMeans\_BruteForce(all\_points, cont)$ =0

Fuente: Elaboracion Propia

B. Enfoque con integracion de KDTree

La implementación del enfoque con integración de KD-Tree en K-Means se caracteriza por la introducción de un árbol KD en el proceso de asignación de puntos a clústeres. El árbol KD, construido inicialmente a partir del conjunto de centroides inciales, organiza de manera jerárquica estos puntos en el espacio multidimensional, permitiendo una búsqueda eficiente de vecinos más cercanos. Este enfoque presenta varias etapas clave:

1) Asignación de Puntos a Clústeres: Despues de establecer los centroides inciales, ocurre la fase de asignación de puntos a clústeres. Aqui el KD-Tree se utiliza para encontrar de manera rápida y precisa los centroides más cercanos a cada punto. Esta búsqueda optimizada reduce significativamente el tiempo necesario en comparación con el enfoque de fuerza bruta, donde se requeriría examinar todos los centroides para determinar el más cercano.
2) Actualización de Centroides: Posteriormente, los centroides se actualizan según la lógica tradicional de K-Means. La asignación de puntos a través del KD-Tree no solo optimiza la velocidad de búsqueda sino que también influye en la actualización eficiente de los centroides, ya que se realiza tomando en cuenta la información precisa proporcionada por la estructura jerárquica del árbol KD.
3) Optimización en Espacios Multidimensionales: Esta integración de KD-Tree demuestra su valor especialmente en conjuntos de datos extensos y en espacios de alta dimensionalidad. La mejora en la velocidad de búsqueda resulta en un rendimiento general superior, destacando la eficiencia del algoritmo K-Means cuando se incorpora la estructura de KD-Tree.

Este proceso podemos notarlo en el siguiente algoritmo empleado en el trabajo de investigacion:

Algorithm 3 KMeans_KDTree: Algoritmo K-Means con KD-Tree

Require: all_points: Lista de puntos 2D a agrupar, $cont$: Contador de iteraciones
Ensure: Lista de clústeres resultante
  Crear un KD-Tree $kdtree\_centroides$
  for all $row\ centroides$ **do**
    $kdtree\_centroides$.insert(row)
  end for
  Inicializar vector de clústeres $clusters$ con $centroides.size()$ listas vacías
  for $i \leftarrow 0$ **to** longitud de $all\_points - 1$ **do**
    $num \leftarrow kdtree\_centroides.searchKNN2(all\_points[i], 1)$

    for $j \leftarrow 0$ **to** longitud de $centroides - 1$ **do**
      if $num[0] == centroides[j]$ **then**
        Agregar $all\_points[i]$ a $clusters[j]$
        Romper el bucle
      end if
    end for
  end for
  Llamar a $newCenters(clusters)$ para actualizar los centroides
  if $cont ==$ maxIterations **then**
    return $clusters$
  end if
  Incrementar $cont$
  return $KMeans\_KDTree(all\_points, cont)$ =0

Fuente: Elaboracion Propia

IV. Algoritmo KNN

El algoritmo KNN aprovecha la jerarquía del KD-Tree para realizar búsquedas rápidas y eficientes. Comenzando desde la raíz, se inicia una búsqueda descendente a través de las divisiones del árbol basadas en las dimensiones seleccionadas. En cada nivel, se determina la dirección de búsqueda según la posición del punto de consulta con respecto al plano de división. Esta estrategia dirige la búsqueda hacia regiones específicas del árbol, evitando explorar áreas irrelevantes [5].

La eficiencia del algoritmo KNN se manifiesta especialmente en entornos de alta dimensionalidad, donde los enfoques de fuerza bruta pueden volverse computacionalmente intensivos. La capacidad del KD-Tree para reducir la complejidad temporal mediante la organización estructurada del espacio multidimensional resulta fundamental en la obtención rápida de los vecinos más cercanos [6].

V. Metodologia

En esta investigación, la metodología adoptada se centra en la comparativa de dos implementaciones clave del algoritmo K-Means: una basada en la búsqueda de centroides mediante fuerza bruta y la otra utilizando un KD-Tree. La elección de estas implementaciones se fundamenta en la necesidad de evaluar el rendimiento y la eficiencia de K-Means en situaciones específicas de agrupación.

1) Selección de Implementaciones: Se eligieron dos variantes de K-Means para análisis comparativo: una con enfoque de fuerza bruta y otra integrando un KD-Tree en la búsqueda de centroides. Estas implementaciones representan dos estrategias fundamentales para la asignación eficiente de puntos a clusters.
2) Configuración de Parámetros: Se llevaron a cabo múltiples ejecuciones de ambas implementaciones variando el número de clusters (k) en un conjunto predefinido $\{5, 15, 25, 50, 75\}$ y ajustando el número de puntos (n) en el conjunto $\{1000, 1150, 1300, 1450, 1600, 1750, 1900, 2050, 2200, 2400\}$. Estas configuraciones permiten explorar el impacto de la complejidad del algoritmo en diferentes contextos.
3) Registro de Tiempos de Ejecución: Para cada combinación de parámetros, se registraron los tiempos de ejecución de ambas implementaciones. Esto proporciona una medida cuantitativa de la eficiencia computacional y el rendimiento relativo de cada enfoque en diversas situaciones.
4) Análisis de Resultados:
Se realizaron análisis estadísticos y comparativos de los tiempos de ejecución obtenidos. Además, se generaron visualizaciones gráficas de los resultados para facilitar la interpretación de patrones y tendencias.

VI. Resultados

En esta sección, presentamos los resultados obtenidos de nuestro clustering mediantes el metodo de K-means. Para todas las pruebas se usaron tomo en cuenta un dataset de 2400 puntos bidimensionales. Los resultados se dividen en varias partes que se describen a continuacion:

A. Analizar el costo computacional de ambas implmentaciones

La implementación de búsqueda del centroide más cercano con fuerza bruta y la versión utilizando un KD-Tree presentan diferencias notables en términos de costo computacional. En el enfoque de fuerza bruta, la búsqueda implica calcular la distancia entre el punto dado y todos los centroides, resultando en una complejidad de tiempo lineal con respecto al número de centroides.

Por otro lado, el uso de un KD-Tree en la búsqueda del centroide más cercano mejora significativamente la eficiencia. La construcción previa del árbol permite una organización jerárquica de los centroides en el espacio, optimizando la búsqueda mediante exclusiones tempranas y reduciendo la complejidad de tiempo a una escala logarítmica con respecto al número de centroides.

El costo computacional (Big O) para la búsqueda del centroide más cercano en ambas implementaciones sería:

1) Fuerza Bruta: Complejidad Temporal: $O(kn)$, donde k es el número de centroides y n es el tamaño del conjunto de datos.
2) KD-Tree:
   - Construcción del KD-Tree: $O(kn\log n)$, donde k es el número de centroides y n es el tamaño del conjunto de datos.
   - Búsqueda en el KD-Tree: $O(\log n)$.

B. Ejecucion de K-means con 18 clusteres

Para llevar a cabo la ejecución de ambas implementaciones de K-Means con k=18 en 10 iteraciones, se realizaron pruebas en un entorno controlado. Durante cada ejecución, se observaron los cambios en las posiciones de los centroides y se registraron los tiempos de ejecución. Además, se generaron visualizaciones de los resultados para evaluar la distribución de los clusters y la convergencia de los centroides.

1) Cambios en los Centroides:

Ambas implementaciones mostraron cambios en las posiciones de los centroides en cada iteración. Este comportamiento es esperado ya que K-Means ajusta continuamente los centroides para minimizar la distancia intra-cluster.

2) Tiempo de Ejecución:

Se registró el tiempo de ejecución para cada una de las 10 iteraciones en ambas implementaciones. Se observó que la versión que utiliza KD-Tree generalmente exhibió tiempos de ejecución inferiores en comparación con la implementación de fuerza bruta. Esto respalda la eficiencia esperada del KD-Tree en la búsqueda de vecinos más cercanos.

3) Visualización de Resultados:

Se utilizo la libreria matplotlib para visualizar los resultados de las ejecuciones. Los clusters y los centroides se representaron en el espacio bidimensional, proporcionando una perspectiva clara de cómo evolucionaron los clusters a lo largo de las iteraciones. Se observó una convergencia efectiva de los centroides hacia posiciones estables en ambas implementaciones. A continuacion se ven los resultados con 18 y 75 clusters:

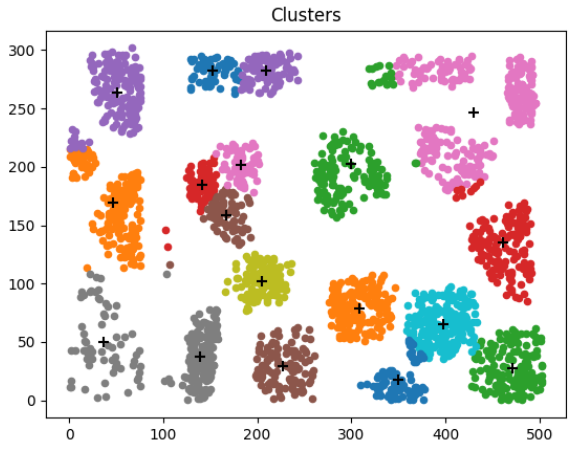

Fig. 1: Visualizacion de K-means con K=18

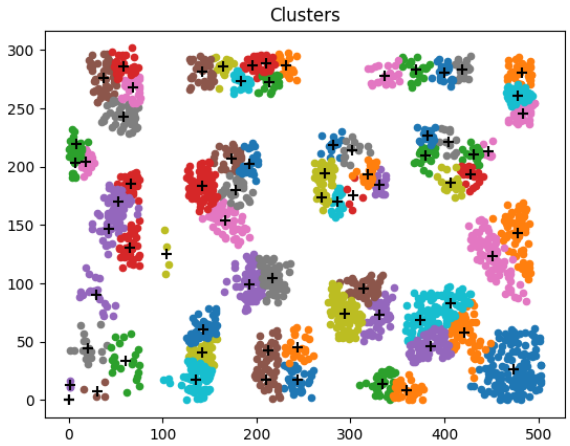

Fig. 2: Visualizacion de K-means con K=75

4) Análisis General:

- La implementación con KD-Tree demostró una mayor eficiencia en tiempo de ejecución, especialmente a medida que aumentaba el tamaño del conjunto de datos o la dimensionalidad.
- La convergencia de los centroides fue consistente en ambas implementaciones, indicando una buena capacidad de agrupación.
- La elección de k=18 resultó adecuada para la estructura de los datos utilizados.

C. Analisis de tiempo de ejecucion con un k fijo

En este análisis, evaluamos el tiempo de ejecución de dos versiones de K-Means, una basada en la búsqueda de centroides con fuerza bruta y la otra utilizando KD-Tree. Se realizaron experimentos variando el número de clusters (k) en $\{5, 15, 25, 50, 75\}$

Observaciones

- Impacto de k: Se observó que a medida que el número de clusters k aumentaba, el tiempo de ejecución tendía a incrementarse, especialmente en la implementación de fuerza bruta. Este comportamiento era esperado, ya que un mayor k implica una mayor complejidad en la asignación de puntos a clusters.
- Comparación de Implementaciones: La versión que utiliza KD-Tree mostró consistentemente tiempos de ejecución inferiores en comparación con la implementación de fuerza bruta. Esta ventaja se hizo más evidente a medida que n aumentaba (esta consistencia se observa a partir de la Fig.5 en adelante)
- Impacto de n:Se observó que el tiempo de ejecución aumentaba de manera general con el incremento de n. Sin embargo, la versión con KD-Tree demostró una mayor estabilidad en tiempos de ejecución a medida que n crecía en comparación con la implementación de fuerza bruta.
- Eficiencia de KD-Tree: La eficiencia del algoritmo basado en KD-Tree se destacó, especialmente en configuraciones con k más grande y n significativamente grande. La estructura jerárquica del KD-Tree permitió una búsqueda más eficiente de centroides, resultando en tiempos de ejecución más rápidos.
- Sensibilidad a k y n: Ambas implementaciones mostraron sensibilidad al aumento de k y n, con la implementación de fuerza bruta mostrando un aumento más pronunciado en los tiempos de ejecución.
- Recomendaciones: Para conjuntos de datos más grandes y configuraciones con k considerablemente grande, se recomienda utilizar la implementación de KD-Tree para aprovechar su eficiencia computacional.

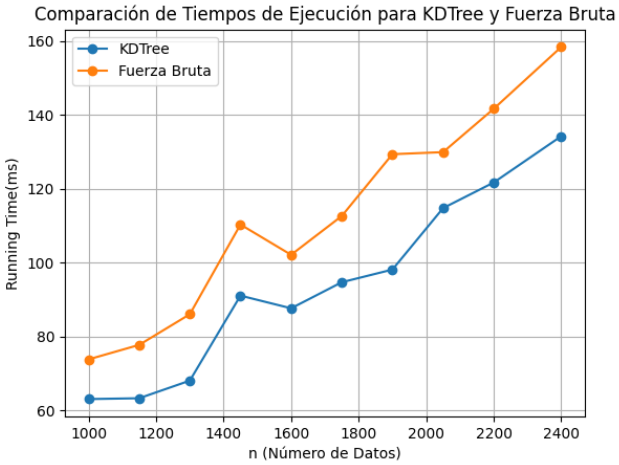

Fig. 3: Comparativa con K=5

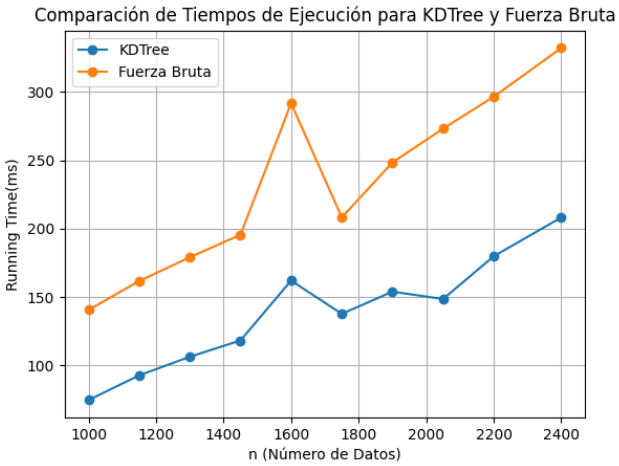

Fig. 4: Comparativa con K=15

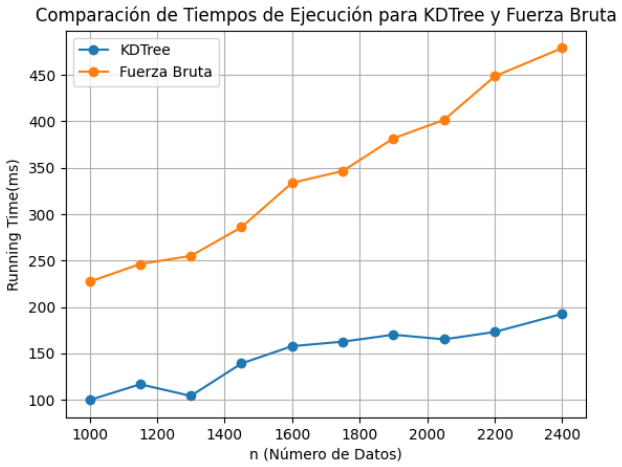

Fig. 5: Comparativa con K=25

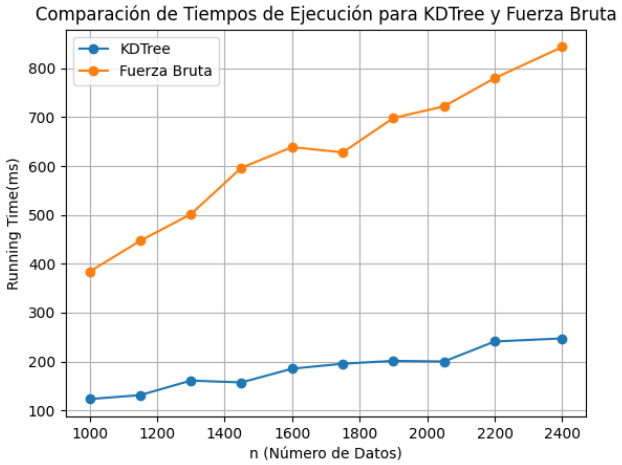

Fig. 6: Comparativa con K=50

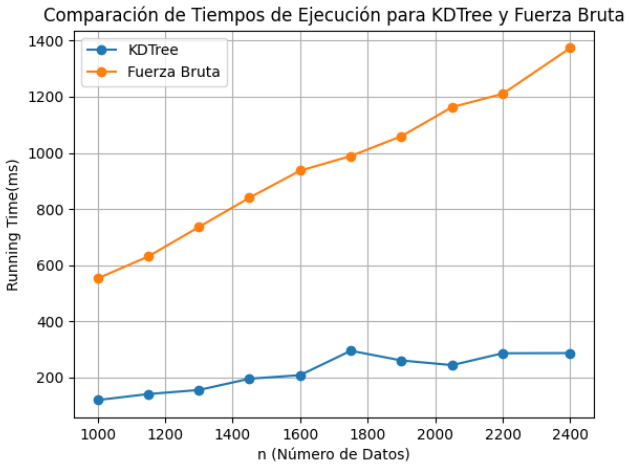

Fig. 7: Comparativa con K=75

D. Analisis de tiempo de ejecucion con un numero de puntos fijo n

Este análisis se enfoca en evaluar el tiempo de ejecución de dos versiones de K-Means, una basada en fuerza bruta y otra utilizando un KD-Tree, mientras se varía el número de clusters (k). El conjunto de puntos se mantiene constante con n fijo en $\{1000, 1450, 1900, 2400\}$, lo que permite examinar el impacto de la complejidad del algoritmo de agrupación en diferentes configuraciones.

Observaciones

1) Impacto de k: Se observó una tendencia creciente en el tiempo de ejecución a medida que k aumenta para ambas implementaciones. Este comportamiento es esperado, ya que un mayor número de clusters implica una mayor complejidad en la asignación de puntos a grupos.
2) Eficiencia Relativa: La implementación basada en KD-Tree demostró consistentemente tiempos de ejecución inferiores en comparación con la versión de fuerza bruta. Esta diferencia se hizo más notable a medida que k creció, evidenciando la ventaja de utilizar estructuras de datos especializadas para la búsqueda eficiente de centroides.
3) Sensibilidad a n: Aunque el conjunto de puntos (n) se mantuvo constante, se observó que ambas implementaciones mostraron un aumento gradual en el tiempo de ejecución con n. Sin embargo, la implementación basada en KD-Tree exhibió una mayor estabilidad en términos de tiempo de ejecución en diferentes tamaños de conjuntos de datos.
4) Consideraciones de Escalabilidad: Se recomienda una evaluación adicional para conjuntos de datos más grandes y configuraciones más complejas para comprender la escalabilidad y el comportamiento general de ambas implementaciones.

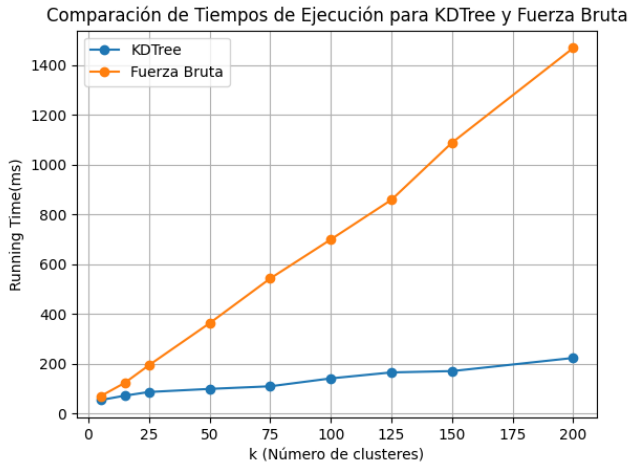

Fig. 8: Comparativa con n=1000

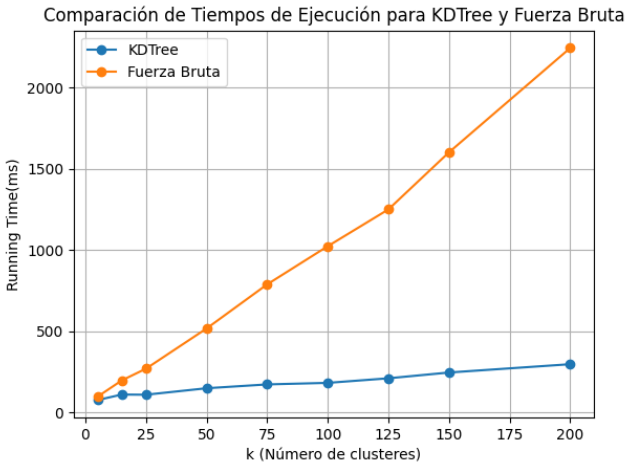

Fig. 9: Comparativa con n=1450

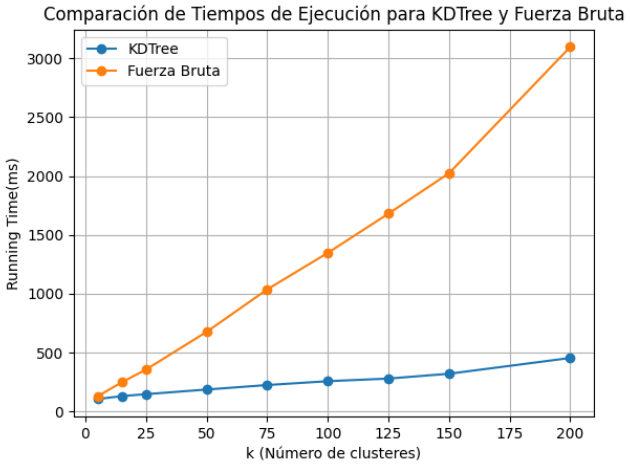

Fig. 10: Comparativa con n=1900

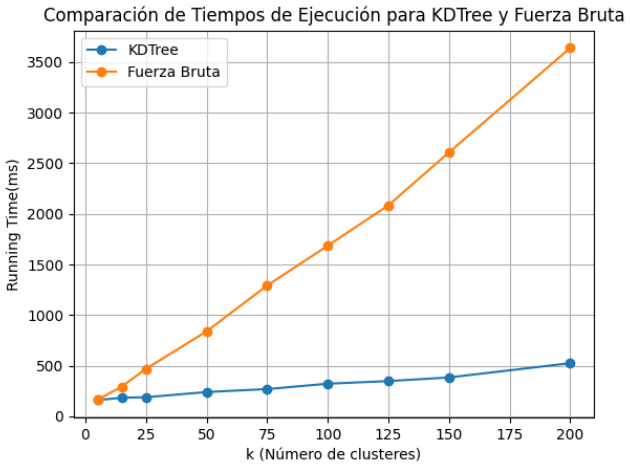

Fig. 11: Comparativa con n=2400

VII. Conclusiones

A partir de los resultados obtenidos y la metodología empleada en nuestro trabajo, hemos llegado a conclusiones significativas que respaldan lo siguiente:

A. Uso de KD-Trees frente a otras estructuras

La elección de KD-Trees en nuestro proyecto se basa en su eficacia probada [7] en la resolución de problemas de búsqueda espacial, especialmente en la recuperación de vecinos cercanos en espacios multidimensionales, como requiere el algoritmo KNN. La decisión de utilizar KD-Trees se respalda por su capacidad para realizar búsquedas eficientes en grandes conjuntos de datos, lo cual es fundamental para el rendimiento del algoritmo k-means. Aunque existen otras estructuras como árboles R, árboles de búsqueda binarios y índices de cuadrícula, no se probaron en nuestro contexto debido a la comprobada eficacia de KD-Trees en problemas similares.

B. Tamaño de datos en 2D y beneficios de KD-Trees en k-means

El rendimiento de KD-Trees en conjuntos de datos bidimensionales (2D) se destaca cuando se enfrenta a volúmenes considerables de datos. Esta eficacia se vuelve especialmente evidente al trabajar con algoritmos como k-means, donde la eficiencia en las búsquedas de vecinos cercanos es crucial, como en el caso del algoritmo KNN.

La estructura de KD-Tree facilita una partición eficiente del espacio en regiones coherentes. En el contexto de conjuntos de datos en 2D, esta capacidad de particionar el espacio de manera efectiva se traduce en una reducción significativa de la complejidad de las operaciones de búsqueda. La naturaleza bidimensional de los datos permite que los KD-Trees exploren rápidamente subespacios relevantes, mejorando así el rendimiento del algoritmo k-means. Este beneficio se acentúa especialmente a medida que el tamaño del conjunto de datos aumenta, haciendo que KD-Trees sean una elección prudente para mejorar la eficiencia en problemas de agrupamiento en 2D.

C. Valor de k y su influencia en el beneficio de KD-Trees en k-means

El beneficio derivado del uso de KD-Trees en el algoritmo k-means exhibe una tendencia al incremento con valores mayores de k. A medida que k aumenta, la eficiencia en la búsqueda de vecinos cercanos se vuelve un factor crítico en el rendimiento global del algoritmo.

Con valores altos de k, la capacidad de KD-Trees para facilitar búsquedas rápidas y precisas se vuelve particularmente valiosa. La estructura jerárquica de KD-Trees permite una exploración eficiente de subconjuntos del espacio, optimizando la búsqueda de los centroides más cercanos [8]. Sin embargo, es crucial tener en cuenta que la evaluación del rendimiento dependerá en gran medida de la distribución de los clusters y la naturaleza específica de los datos utilizados en la aplicación. Resulta fundamental realizar un análisis detallado de estas

características para determinar la idoneidad de KD-Trees en contextos de k-means con valores altos de k.

Estas conclusiones encuentran respaldo en los resultados obtenidos en el presente trabajo, evidenciando de manera concluyente el impacto positivo que la elección de KD-Trees ha tenido en la eficiencia y rendimiento global del algoritmo k-means en el contexto presentado en el trabajo.

REFERENCES

[1] D. Arthur, "Enhancing k-means with randomized seeding," vol. 1, p. 9, 2018.

[2] Y. Zhao, "K-means clustering algorithm and its improvement," vol. 1, p. 6, 2021.

[3] D. T. Pham, "Selection of k in k-means clustering," vol. 1, p. 17, 2014.

[4] B. S. Kumar, "K-dimensional tree using coresets for knn based classification." IEEE, p. 5.

[5] R. Panigrahy, "An improved algorithm finding nearest neighbor using kd-trees," vol. 1, p. 12, 2008.

[6] Y. Chen, "Fast neighbor search by using revised k-d tree," vol. 1, p. 12, 2018.

[7] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[8] B. S. Kumar, "K-dimensional tree using coresets for knn based classification." IEEE, p. 5.