SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Desarrollo de una mejora al algoritmo K-means orientada al paradigma de Big Data

presentada por

Ing. Alber Díaz Lorenzo

como requisito para la obtención del grado de

Maestro en Ciencias Computacionales

Director de tesis

Dr. Joaquín Pérez Ortega

Cuernavaca, Morelos, México. Octubre de 2018.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Morelos a 15 de agosto del 2018
OFICIO No. DCC/220/2018

Asunto: Aceptación de documento de tesis

DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **Ing. Alber Díaz Lorenzo**, con número de control M16CE077, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado "**Desarrollo de una mejora al algoritmo K-means orientada al paradigma de Big Data**" y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Joaquín Pérez Ortega
Doctor en Ciencias Computacionales
4795984

REVISOR 1

Dr. Dante Mújica Vargas
Doctor en Comunicaciones y Electrónica
09131756

REVISOR 2

Dr. José Crispín Zavala Díaz
Doctor en Ciencias Computacionales
3406871

C.p. M.T.I. María Elena Gómez Torres - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

NACS/lmz

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 26 de septiembre de 2018
OFICIO No. SAC/413/2018

Asunto: Autorización de impresión de tesis

ING. ALBER DÍAZ LORENZO
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Desarrollo de una mejora al algoritmo K-means orientada al paradigma de Big Data"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
EXCELENCIA EN EDUCACIÓN TECNOLÓGICA®
"CONOCIMIENTO Y TECNOLOGÍA AL SERVICIO DE MÉXICO"

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

SEP TecNM
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

C.p. M.T.I. María Elena Gómez Torres .- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

Dedicatoria

Dedico este logro a mi familia que son mi vida.

A mi madre Maria Amelia Lorenzo y a mi padre Alberto Diaz Barreto quienes son mi fuerza y ejemplo en la vida. No tengo como agradecerles todo el sacrificio hecho para que mi hermana y yo logremos las metas propuestas. Ellos fueron el mayor apoyo en este emprendimiento y sin ellos hubiera sido imposible venir a un pais extranjero y terminar esta maestria.

A mi hermana querida para la cual espero ser un ejemplo, todas las metas son alcanzables con perseverancia y convicciones firmes en uno mismo.

Agradecimientos

A mis padres, a los cuales les agradezco todo en la vida y me brindaron la ayuda necesaria para emprender este proyecto muy lejos de casa.

Quiero agradecer a la familia Pino por todo su cariño y ayuda en México, me acogieron como un miembro más y me abrieron las puertas de su casa.

Agradezco a todos mis compañeros del CENIDET, los cuales son una gran familia que he forjado lejos de mi tierra querida; especialmente Omar, Ani, Marianita, Lorenzo y Sócrates.

Un gran agradecimiento a Gaby por estar a mi lado, brindarme su amor y vivir momentos maravillosos juntos.

Agradezco al CENIDET por brindarme la oportunidad, siendo extranjero, de estudiar esta maestría, y al CONACyT por el apoyo económico sin el cual hubiera sido imposible estar aquí.

Agradezco al Dr. Joaquín Pérez Ortega por la asesoría y guía durante el proceso de tesis y sobre todo en los momentos finales. También quiero agradecer al Dr. Crispín Zabala Díaz por los conocimientos transmitidos durante sus clases y sus aportes, junto al Dr. Dante Mujica Vargas, como asesores de esta investigación.

Resumen

K-means es uno de los algoritmos de agrupamiento de datos más usados en la actualidad, dada la sencillez y fácil interpretación de sus resultados. El algoritmo está estructurado en cuatro fases: inicialización, clasificación, cálculo de centroides y convergencia. Prácticamente todas las mejoras realizadas al algoritmo están enfocadas a una fase en particular; sin embargo, de acuerdo a la literatura especializada, no se ha propuesto la integración de mejoras de distintas fases. En este sentido, en este trabajo se propone *H-Kmeans*, una mejora híbrida que permite procesar, de manera optimizada, una instancia dada mediante composición de mejoras de diferentes fases.

Para validar *H-Kmeans* se realizaron un conjunto de experimentos con instancias reales reconocidas por la comunidad científica internacional. Se seleccionaron 2 mejoras en las fases de inicialización, clasificación y convergencia, respectivamente. *H-Kmeans* permitió realizar 19 configuraciones para procesar las instancias seleccionadas.

Los resultados obtenidos mostraron que con la combinación de mejoras se obtienen mejores resultados que procesando las instancias con las mejoras individuales o con *K-means*. Es necesario aclarar que al realizar mejoras a *K-means* se suele sacrificar la calidad de los resultados para lograr un menor tiempo de procesamiento; en muchos de los resultados obtenidos con *H-Kmeans* incluso se mejora la calidad de los resultados. Para la instancia *3D road network*, con 434,874 objetos en 4 dimensiones, se obtuvo una reducción de tiempo del 82% con una ganancia de calidad del 47%. Para la instancia *Household power consumption,* con 2,049,280 objetos en 4 dimensiones, se obtuvo una reducción de tiempo del 97.3% con una ganancia de calidad del 2.3%. Aunque aumente el tamaño de la instancia los resultados del procesamiento siguen siendo excelentes, esto valida la utilización de la mejora *H-Kmeans* para el procesamiento de instancias dentro del paradigma de Big Data.

Abstract

K-means is one of the most commonly cluster algorithm used by its simplicity and easy interpretation of its results. The algorithm is structured in four phases: initialization, classification, centroid calculation and convergence. Almost all the improvements made to the algorithm are focused in one phase in specific; however, according to the specialized literature, the integration of improvements of differents phases has not been proposed. In this way, in this work its proposed *H-Kmeans*, an hibryd improvement that processes, in an optimus way, an instance through the combination of differents phases improvements.

To validate *H-Kmeans* many experiments were made with instances recognised by the international scientific community. Two improvements were selected by phases. *H-Kmeans* allowed to make 19 configurations to process the selected instances.

The results shows that with the combinations of improvements we can obtain better results than processing the instances with individual improvements or with *K-means.* When you develop improvements to *K-means* the common thing is to sacrify the quality of the results to obtain a better performance. In many of the obtained results, their quality is even improved with *H-Kmeans* rather than with the individual improvements. For the instance *3D road network,* with 434,874 objects in 4 dimensions, we obtain a time reduction of 82% with a quality increase of 47%. For the instance *Household power consumption,* with 2,049,280 objects in 4 dimensions, we obtain a time reduction of 97.3% with a quality increase of 2.3%. Even if the size of the instance increases the quality of the results are still excellents; this validates the use of *H-Kmeans* to process Big Data.

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

Capítulo 1

Introducción

Con la evolución del hombre apareció la necesidad de almacenar la información de su interés para luego realizar análisis y tomar decisiones con certeza. La llegada de la era digital y posteriormente internet hizo más fácil el problema de almacenar información, por lo que ésta ha crecido exponencialmente. Actualmente, se encuentra en el orden de los 7.9 zettabytes y se espera que para el 2020 llegue a los 35 zettabytes[1]. En los últimos años se han creado más datos que en toda la historia de la humanidad[2]. Por solo citar algunos ejemplos Facebook tiene un promedio de 231 millones de activaciones de usuarios al mes, esto sólo en Estados Unidos y Canadá[3]; la red profesional LinkedIn tiene más de 467 millones de usuarios[4]; en un segundo se suben un promedio de 767 fotos a Instagram, se realizan 2,472 llamadas por Skype, 58,873 búsquedas en Google, se visualizan 68,233 videos en YouTube, y el tráfico de internet es de 42,212 gigabytes[5]. Para describir estos enormes volúmenes de datos se comenzó a usar el término Big Data. A diferencia de los datos tradicionales Big Data incluye mayormente datos no estructurados, lo que requiere un mayor tiempo de análisis[6].

Tanta información almacenada necesita una forma de organizarla para su posterior manipulación. Por tanto, el problema de agrupamiento de datos u objetos ha sido ampliamente estudiado debido a su utilización en una gran variedad de campos como inteligencia artificial, ingeniería eléctrica, reconocimiento de patrones, minería de datos, genética, biología, patología, geografía, geología, negocios y marketing, entre otros. Uno de los algoritmo de agrupamiento de datos más populares y usados es el algoritmo *K-means*[7], debido a su simplicidad, fácil implementación y fácil

interpretación de sus patrones. *K-means* consta de 4 fases, las cuales fueron definidas en [8] tomando como base los artículos [9]–[13]:

1. **Inicialización:** se definen *k* centroides iniciales, de un conjunto de *n* objetos, donde *k* es la cantidad de grupos en los que se desea agrupar el conjunto de objetos. Los centroides iniciales pueden ser generados aleatoriamente o mediante un preprocesamiento de los datos.
2. **Clasificación:** por cada objeto se calcula la distancia a cada centroide *k* y se determina el centroide más cercano. Se asigna el objeto al grupo asociado al centroide más cercano.
3. **Cálculo de centroides:** se calcula el punto medio de cada grupo creado en el paso anterior y se selecciona como nuevo centroide de ese grupo.
4. **Convergencia:** existen varias condiciones de convergencia, como son: detener el algoritmo al llegar a un número determinado de iteraciones, cuando no ocurran cambios de objetos hacia otros grupos con respecto a la iteración anterior, cuando los centroides se mantienen en iteraciones consecutivas, entre otras. Si la condición de convergencia establecida no se cumple se repiten los pasos 2, 3 y 4.

Es importante destacar que el término clasificación puede tener diferentes acepciones dependiendo del área de conocimiento. En este documento se entenderá el concepto de clasificación como la asignación de un objeto a un grupo de acuerdo a un criterio, en este caso el criterio es la distancia euclideana del objeto al centroide más cercano.

1.1 Descripción del problema

En el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), desde el año 2005, se han venido desarrollando varias mejoras al algoritmo *K-means*[8]–[13]. Las mejoras previas no están en su totalidad enfocadas al paradigma de Big Data, por lo tanto emplear el algoritmo *K-means* para análisis de enormes volúmenes de datos sigue siendo un proceso complejo debido a la cantidad de recursos que demanda y el tiempo que demora su análisis.

A partir de la situación antes descrita surgió como problema a resolver: ¿Cómo adecuar el algoritmo *K-means* para el trabajo con Big Data?

1.2 Objetivo general

Desarrollar una mejora híbrida del algoritmo *K-means* orientada a resolver, de manera más eficiente, instancias con las características del paradigma de Big Data.

1.3 Objetivos específicos

- Desarrollar una mejora al algoritmo *K-means* con un enfoque híbrido.
- Seleccionar dos mejoras por cada fase para su integración en la mejora híbrida.
- Desarrollar el mecanismo de integración de mejoras.

1.4 Hipótesis de la investigación

Es posible desarrollar una mejora híbrida, que integre varias mejoras previas de *K-means,* cuya solución al procesar grandes instancias sea más eficiente que la obtenida por mejoras individuales de *K-means*.

Capítulo 2

Estado del arte

Debido a su amplia utilización muchos se han dedicado al estudio y mejora del algoritmo *K-means*. Es importante destacar que se tienen dos vertientes, una que se ha enfocado en adaptar el algoritmo para solucionar problemas de un dominio específico, y otra que se ha enfocado en mejorar el algoritmo; ya sea alguna de sus etapas por separado o en conjunto. Esta sección se centrará en las mejoras hechas al algoritmo, ya que la presente investigación busca realizar una mejora al mismo.

2.1 Fase de Inicialización

Esta posiblemente sea la fase donde más se ha trabajado en busca de mejoras, puesto que una buena selección de los centroides iniciales permite una convergencia más rápida y una mejor calidad en el agrupamiento.

2.1.1 K-means usando MapReduce

En el artículo “Optimized big data K-means clustering using MapReduce”[19], se propuso una mejora que hace uso del framework MapReduce para implementar en paralelo el algoritmo. Esta mejora se basó en reducir el costo de comunicación y el costo en la lectura y escritura de datos entre los procesadores, algo fundamental para el éxito de un algoritmo en paralelo. Para ello se propuso dividir el conjunto de datos original en pequeñas muestras, las cuales se procesan y se obtienen los centroides que serán usados para particionar el conjunto original. El proceso consiste en tres ejecuciones o “*jobs*” de MapReduce y se describen a continuación:

1. En el primer “*job”* se realiza la selección de las muestras: para esto se divide el conjunto original *N* en pequeñas muestras de tamaño *2k (k es el número de*

clústers). La distribución de los datos se realiza dividiendo el conjunto original usando *k*, probabilidad $p_x = \frac{1}{\varepsilon^2 * N}$, donde $\varepsilon \in (0,1)$ y controlando el tamaño de la muestra *N*. De esta forma se obtienen los valores de las muestras, las cuales son lo suficiente pequeñas para ser procesadas de manera rápida.

2. En el segundo "*job*" se aplica el algoritmo *K-means* a cada muestra usando *k* clústers y se obtienen un total de $2k^2$ centroides, los cuales se fusionan para obtener los *k* centroides finales que serán usados para el particionado final.

   Para fusionar los $2k^2$ centroides en *k* centroides se propusieron dos estrategias de fusión:

   - Fusión basada en pesos (*WMC*): durante la ejecución del "*job*" se recoge el número de puntos que pertenecen a cada centroide para que sirva como peso del centroide. En esta estrategia se escogen *k* centroides iniciales de los $2k^2$, usando la estrategia de inicialización del algoritmo *K-means*++[20], con sus respectivos pesos; luego se calcula la media aritmética ponderada de los demás centroides a estos *k* escogidos para ir fusionando los grupos. Este proceso se repite hasta que se tengan *k* grupos finales, y sus centroides serán los centroides iniciales al procesar el conjunto original de datos.

   - Fusión basada en la distribución (*DMC*): en esta estrategia primero se escoge una muestra C_1 de las muestras iniciales, al azar (recordando que cada muestra tiene *2k* centroides), luego se asignan cada centroide de C_1 a otras muestras C_i; una vez hecho esto recorremos todas las muestras restantes asignando todos sus centroides a los grupos más cercanos a estos. Luego calculamos los nuevos centroides de los nuevos grupos surgidos y estos serán los centroides iniciales al procesar el conjunto original de datos.

3. En el tercer "*job*" se aplica el algoritmo *K-means* al conjunto de datos iniciales, con los centroides obtenidos en el segundo "*job*", para obtener el agrupamiento final.

Esta mejora fue probada en una base de datos sintética de una distribución gaussiana con 10,000 valores tridimensionales y dos bases de datos reales, *Bag of Word Data Set (BoW)* con 2,351,710,420 valores en 3 dimensiones, e *Individual household electric power consumption* con 4,296,075,259 valores en 9 dimensiones. Los autores plantearon que los experimentos demostraron la superioridad de la mejora con respecto al *K-means* estándar y al *Kmeans*++ probados en las mismas bases de datos, en implementaciones lineales y paralelas.

2.1.2 Algoritmo basado en promedios (Mean-Based Algorithm)

En el artículo [21] los autores propusieron una mejora para seleccionar los centroides iniciales, la cual consta de los siguientes pasos:

1. Dado un conjunto *D* de *n* datos, si existen valores negativos en alguna dimensión de la muestra seguir con el paso 2, si no ir al paso 3.
2. Buscar el valor mínimo en una dimensión y restárselo a cada valor de la dimensión. Hacer lo mismo con las demás dimensiones.
3. Para cada dato *n* calcular su distancia euclideana al origen (0, 0).
4. Ordenar las distancias obtenidas en el paso anterior, y acorde con las distancias ordenadas ordenar los datos del conjunto *D* original.
5. Dividir el conjunto *D* ordenado en *k* números de particiones iguales, donde *k* es el número de clústers en los que se quiere agrupar lo datos del conjunto *D*.
6. En cada partición calcular el valor medio. Los valores medios serán usados como los centroides iniciales en el algoritmo *K-means*.

Como resultado de la investigación Goyal & Kumar demostraron que es mucho mejor su propuesta que seleccionar los centroides iniciales de manera aleatoria. Además su algoritmo no tiene dificultades con conjuntos de datos uniformes o no uniformes. Otra ventaja es que al determinar con anterioridad los centroides iniciales se disminuyen las iteraciones necesarias para llegar a la convergencia. Sin embargo las bases de datos utilizadas no son de dominio público, por lo que no se pueden analizar las características de estas.

2.1.3 Mejora en inicialización para dos dimensiones

En el trabajo publicado por [22] se propuso un método para escoger los centroides iniciales. Este método tiene como inconveniente que solo es aplicable a conjuntos de datos con dos dimensiones. La investigación está basada en el trabajo de [21] por lo que sus pasos son muy similares:

1. Dado un conjunto de *n* datos y dos dimensiones *X* e *Y*, si existen valores negativos en la muestra seguir con el paso 2, si no ir al paso 4.
2. Buscar el mínimo valor en la dimensión *X*, X_{min}, y el mínimo en la dimensión *Y,* Y_{min}.
3. A cada valor de la dimensión *X* sustraerle X_{min}, hacer lo mismo en la dimensión *Y* y crear un nuevo conjunto de datos con los valores.
4. Encontrar el valor mínimo y máximo en cada dimensión: *Min (X), Max(X), Min(Y), Max(Y)*. Crear 4 punto: *(Min(X);Min(Y)), (Max(X);Min(Y)), (Max(X);Max(Y)), (Min(X);Max(Y))*, los cuales será los extremos del rectángulo.
5. Dividir el rectángulo en *k* partes, donde *k* es la cantidad de clúster. Encontrar el centro de cada clúster y conformar el conjunto de centroides iniciales.

Como resultado de la investigación Vij & Kumar demostraron que su mejora es superior a los trabajos de [23] y [21] (aclarar que el artículo de Goyal & Kumar fue aceptado por la revista en 2012 pero no se publicó hasta 2014, por esto es que Vij & Kumar aquí lo citan) ya que realiza menos iteraciones, esto sin importar la cantidad de datos a agrupar. Sin embargo solo fue probado en bases de datos pequeñas por lo que no se puede asegurar su efectividad en Big Data.

2.1.4 K-means++

En el artículo [20], titulado “K-means++: The Advantages of Careful Seeding”, los autores propusieron un método de inicialización que posiblemente sea el más estudiado, analizado y usado. *K-means*++ es un método muy específico de seleccionar los centroides iniciales, mucho más rápido que *K-means* y que genera mejores resultados en el agrupamiento.

Suponiendo que queremos seleccionar el conjunto *C* de *k* centroides iniciales, de un conjunto de puntos *X* $(x_1, x_2, x_3, x_4, \ldots, x_n)$, el algoritmo funciona de la siguiente forma:

1. Seleccionar el primer objeto de manera aleatoria del conjunto *X* y adicionarlo al conjunto de centroides *C*.
2. Por cada punto x_i que queda en *X* calcular la distancia $D(x_i)$ al centroide más cercano contenido en *C*.
3. Escoger un número *"y"* aleatorio entre 0 y $\sum_{i=1}^{n} D(x_i)^2$.
4. Encontrar el entero "*i*" tal que $\sum_{j=1}^{i} D(x_j)^2 \geq y > \sum_{j=1}^{i-1} D(x_j)^2$.
5. Adicionar x_i al conjunto de centroides *C*.
6. Repetir los pasos del 2 al 5 hasta que se seleccionen los *k* centroides.

El algoritmo fue probado en 4 instancias y comparados los resultados con la ejecución de *K-means* sobre las mismas instancias. Se utilizaron dos instancias sintéticas y dos reales. La primera instancia real utilizada fue *Cloud*, con 1,024 objetos en 10 dimensiones, y representa la primera nube. La segunda instancia real utilizada fue *Intrusion* con 494,019 objetos en 35 dimensiones. Ambas instancias reales están disponibles en "UC-Irvine Machine Learning Repository", en adelante nos referiremos a este repositorio como UCI.

2.1.5 ElAgha

En la publicación "Efficient and Fast initialization Algorithm for K-means Clustering"[24] propusieron un método de inicialización al que nombran ElAgha initalization. Es un algoritmo que genera centroides iniciales dependiendo de la forma de los datos. La explicación del mismo está realizada contemplando instancias de dos dimensiones para una mejor comprensión pero puede ser aplicado a cualquier número de dimensiones.

ElAgha primeramente busca los límites del conjunto de datos y divide el área de puntos en una matriz de *k* filas * *k* columnas, formando una cuadrícula. El ancho de cada celda de la cuadrícula es $Xw = \frac{Xmax - Xmin}{k}$, donde *Xmax* corresponde al mayor valor de *X* entre todos los puntos y *Xmin* corresponde al menor. El alto de cada celda de la cuadrícula es $Yw = \frac{Ymax - Ymin}{k}$, donde *Ymax* corresponde al mayor valor de *Y* entre todos los puntos y *Ymin* corresponde al menor. Luego *ElAgha* usa la esquina superior izquierda de las celdas en la diagonal como puntos base para generar los centroides iniciales. Los centroides iniciales son generados de manera aleatoria, teniendo en cuenta que estén comprendidos entre *Xw/2* a la izquierda del punto base y *Xw* a la derecha en el eje de las *X,* y *Yw/2* hacia arriba del punto base y *Yw* hacia abajo en el eje de las *Y.*

El algoritmo fue comparado con *K-means*. Las instancias sintéticas utilizadas para las pruebas fueron de 320, 373 y 211 objetos; además, se usó la instancia real Iris, del repositorio UCI, con 150 objetos y 4 dimensiones. Los resultados obtenidos por los autores mostraron que el algoritmo *ElAgha* superó en rendimiento al algoritmo estándar y mejoró la calidad del agrupamiento con un gran margen, especialmente en instancias complejas.

2.2 Fase de Clasificación

2.2.1 Enhanced K-means

Esta mejora ha sido ampliamente estudiada y referenciada en la literatura especializada. En su artículo “An efficient enhanced *k*-means clustering algorithm”, [25] plantearon aprovechar la iteración anterior del algoritmo para reducir el número de cálculos en la clasificación de los objetos. Para esto se guarda la distancia de cada objeto a su centroide, en la próxima iteración se calculan los nuevos centroides y se calcula la distancia, primeramente, de cada objeto al nuevo centroide de su grupo; si la distancia es menor o igual a la distancia anterior el objeto permanece en ese grupo, por lo que no es necesario calcular la distancia a los demás centroides. De esta forma se reducen los cálculos a *k-1* centroides.

El método propuesto describe dos funciones, la primera es llamada *distance()* y no es más que la fase de clasificación de *K-means*, simplemente agregando una estructura de datos que almacena la distancia de cada objeto a su centroide. La segunda función es la misma que la anterior, solo que esta vez se calcula la distancia del objeto a su nuevo centroide y se compara con la distancia almacenada en la estructura de datos usada en la función *distance(),* si esta distancia es menor o igual se descartan los cálculos de distancia de ese objeto al resto de los nuevos centroides ya que el mismo se mantiene en su grupo. Si la distancia es mayor entonces se realizan todos los cálculos de distancia a los nuevos centroides, se almacena el nuevo centroide al cual pertenece ahora el objeto y se actualiza la estructura de datos que contiene la distancia de cada objeto a su centroide.

Esta mejora fue probada en 3 instancias reales: *Letters* con 20,000 objetos en 16 dimensiones, *Abalone* con 4,177 objetos en 7 dimensiones y *WIND* con 6,574 objetos en 15 dimensiones. Estas instancias también forman parte del repositorio UCI. También se probó en una instancia sintética de 2 dimensiones pero no se especifica el tamaño de la misma. Se tomaron como parámetros de comparación el tiempo de ejecución y la calidad del agrupamiento. Como medida de calidad del agrupameinto se tomó el valor de la media del *error al cuadrado*. Se compararon los resultados con las ejecuciones del algoritmo estándar sobre las mismas instancias y se obtuvo una superioridad de un 40% en el tiempo de ejecución, obteniendo una calidad de agrupamiento aproximadamente igual.

2.2.2 Early Classification

En el artículo “Early Classification: A New Heuristic to Improve the Classification Step of K-means”[14] los autores propusieron una heurística que reduce el número de cálculos en la etapa de clasificación de *K-means,* sin pérdidas significativas de la calidad de agrupamiento final, usando información estadística sobre el desplazamiento de centroides en cada iteración. La reducción tienen lugar al identificar objetos que están asignados a grupos específicos en una iteración y es poco probable que cambien de grupo en las siguientes iteraciones. Estos objetos son excluidos de futuros cálculos.

Para selecionar los objetos que se excluyen de los cálculos se plantean dos funciones *índice de equidistancia* y *umbral de equidistancia.*

El *índice de equidistancia* α representa la diferencia entre las distancias de un objeto *i* a los dos centroides más cercanos μ_1 y μ_2, $\alpha=abs((i-\mu_1)^2-(i-\mu_2)^2)$. El menor valor de α es 0, e indica que el objeto *i* está ubicado equidistantemente de los centroides μ_1 y μ_2. El mayor valor de α es $(\mu_1-\mu_2)^2$, e indica que el obtejo *i* está ubicado en la misma posición que el centroide μ_1 o que el centroide μ_2. Cuando el valor de α está más cerca de 0 indica que el objeto tiene grandes probabilidades de cambiar de grupo en la siguiente iteración; mientras que, cuando el valor de α está más cerca de $(\mu_1-\mu_2)^2$ indica que el objeto tiene pocas probabilidades de cambiar de grupo en las siguientes iteraciones.

El *umbral de equidistancia* $ß_j$ ayuda a identificar los objetos con altas probabilidades de cambiar de grupo. Es un valor de referencia definido por la suma de los dos mayores desplazamientos $ß_j=m_1+m_2$ de los centroides μ_1 y μ_2 en la iteración *j* $(j>2)$.

Un objeto *i* tiene pocas probabilidades de cambiar de grupo si $\alpha_i > ß_j$; por tanto si se cumple esta condición y μ_x es el centroide más cercano a *i* entonces el objeto *i* puede ser asignado al grupo C_x en la iteración *j* y excluirlo de futuros cálculos.

Para la comprobación del desempeño de la mejora planteada se utilizaron 3 instancias sintéticas y 3 reales. Las instancias sintéticas fueron de 2,500; 10,000 y 40,000 objetos en 2 dimensiones respectivamente, siguiendo una distribución uniforme. En el caso de las instancias reales se utilizaron las instancias *Iris* con 150 objetos en 3 dimensiones, *Concrete compressive strength* con 1,030 objetos en 8 dimensiones, y *Skin segmentation* con 245,057 objetos en 3 dimensiones. Todas las instancias reales fueron tomadas del repositorio UCI. Los resultados de la experimentación fueron comparados contra los arrojados por *K-means y Enhanced K-means* al procesar las mismas instancias. Como medidas de comparación se utilizaron el tiempo de ejecución y la calidad del agrupamiento. Para medir la calidad del agrupamiento se tomó en cuenta la media del *error al cuadrado*.

En todos los casos el algoritmo obtuvo mejores tiempos de ejecución que *K-means* con pérdidas de calidad poco significativas, siendo la mayor pérdida del 6.9% para la instancia *Skin*. En el caso de la comparación contra *Enhanced K-means* se obtiene menor tiempo de ejecución por parte de *Enhanced K-means* pero la calidad del agrupamiento es mucho mejor con *Early Classification.*

2.2.3 Híbrido entre Early Classification y Enhanced K-means

En la investigación realizada por [15] como tesis de maestría, desarrollada en el CENIDET, se propuso mezclar las mejoras *Early Classification*[14] y *Enhanced K-means*[25] en la fase de clasificación. Si el algoritmo se encuentra en la sexta iteración se propone clasificar usando el método de *K-means* tradicional; si se encuentra en la séptima se propone usar el método *Early Classification*; y en cualquier otra iteración que se encuentre el algoritmo se propone realizar la clasificación utilizando la mejora *Enhanced K-means*. La mejora aquí propuesta ofrece mejores tiempos de ejecución que *Enhanced K-means* y mejora la calidad del agrupamiento de *Early Classification.* Además no se ve afectada por la variación en el número de grupos que se deseen formar. Esta mejora fue probada en las bases de datos reales *WIND* con 6,574 objetos en 15 dimensiones, *Letters* con 20,000 objetos en 16 dimensiones, *Daily and Sports Activities (DASA)* con 1,140,000 objetos en 45 dimensiones, *Household Power Consumption (HPC)* con 2,075,259 objetos en 7 dimensiones. Todas estas instancias forman parte del repositorio UCI. También se probó con tres bases de datos sintéticas con 2,250,000 objetos en 2 dimensiones, 2,248,091 objetos en 3 dimensiones y 2,085,136 objetos en 4 dimensiones respectivamente.

2.2.4 Heurística HoneyComb

En el artículo [16] se propuso la heurística de *HoneyComb* (HC o panal de abejas) la cual consiste en limitar el número de grupos vecinos más cercanos (ω) a los cuales es probable que se traslade un elemento de un grupo. Esto hace que, básicamente, "… *para cada objeto se calculen las distancias únicamente hacia los centroides de los grupos vecinos adyacentes más cercanos ...*". Se plantea una función, la cual define la cantidad de cálculos de distancia que se realizan a los vecinos adyacentes más

cercanos, dependiendo de la dimensionalidad de la instancia. Así queda que $\omega = f(d)$, donde:

$$f(d) = \begin{cases} 5, & (d = 2, 3, 4, 5) \\ 6, & (d = 6, 7) \\ 7, & (d = 8) \\ 6, & (d = 9, 10) \\ 5, & (d = 11, 12, 13, 14) \\ 4, & (d = 15, 16, 17) \\ 3, & (d = 18, 19, 20, 21, 22, 23, 24) \\ 2, & (d = 25, 26, 27, 28, 29, 30) \wedge (d \leq 200) \end{cases}$$

Esta función se traduce en que si la dimensión de la instancia es 2, 3, 4 o 5 los cálculos de distancia se realizarán sólo a los 5 vecinos adyacentes más cercanos; si la dimensión de la instancia es 6 o 7 sólo serán 6 vecinos los adyacentes más cercanos; y así sucesivamente.

La heurística fue comparada con *K-means* y las mejoras propuestas por [26], [27] presentando una reducción en el tiempo de ejecución superior a las tres y con una pérdida de calidad no superior al 1% para instancias sintéticas y no superior al 1.6% para instancias reales. Los experimentos mostraron que su desempeño es mejor para instancias con 3,000 objetos o más, con 10 dimensiones y agrupados en más de 100 grupos. Se experimentó con las instancias reales *MNIST*, que cuenta con 10,000 objetos en 748 dimensiones, y *House*, que cuenta con 34,112 objetos en 3 dimensiones, [28]; se fijó el número de grupos 100 y 200 para *MNIST* y 100, 200 y 400 para *House*. También se empleó una instancia sintética con 25,000 objetos en 2, 10, 30 y 60 dimensiones, los cuales se agruparon en 100 grupos.

2.2.5 N-means

Esta meta-heurística fue propuesta por [16]. La meta heurística consiste en la integración de la mejora *Early Classification*[14] y otra desarollada denominada *Grupos estables*. "*(…) Un grupo estable es aquel que ya no tiene intercambio de objetos con otros grupos en iteraciones posteriores (...)"*; por lo que los objetos que pertenecen a

esos grupos pueden ser descartados de futuros cálculos de distancia ya que no se moverán de grupo.

El cálculo de grupos estables se hace a partir de la segunda iteración; para ello se calculan los nuevos centroides de cada grupo y se calcula su distancia al anterior centroide del grupo. Si la distancia es 0, o sea que el centroide se mantiene entonces ese grupo ya es estable, por lo que todos sus objetos se excluyen de los cálculos en la iteraciones posteriores.

Esta meta-heurística fue comparada con *K-means* para validar su desempeño, para esto se midió el tiempo de ejecución y la calidad del agrupamiento final. Para las pruebas se usaron instancias provenientes del repositorio de la Universidad Eastern Finland. Se utilizó la instancia *Birch1* con 100,000 objetos en 2 dimensiones y el conjunto de instancias *DIM*, el cual consta de 6 instancias de 1,024 objetos cada una y 32, 64, 128, 256, 512 y 1,024 dimensiones respectivamente. Todas las instancias utilizadas fueron sintéticas.

Como resultado de esta investigación los autores plantearon que es factible reducir la complejidad de *K-means* de manera importante haciendo uso de la meta-heurística *N-Means*. Se obtuvieron reducciones de tiempo hasta de un 91% y una disminución de la calidad del 5.5% en el peor de los casos. Con base en el análisis de los resultados los autores observaron un comportamiento cuasi lineal de *N-Means*; esto es importante cuando se resuelven instancias con un gran número de objetos, grupos o dimensiones.

2.3 Fase de Cálculo de Centroides

2.3.1 Mux Kmeans

En el artículo [29] los autores plantearon un método para el cálculo de centroides y la obtención del mejor agrupamiento. En la práctica, los usuarios ejecutan el algoritmo estándar una serie de veces con diferentes centroides iniciales, buscando una distribución normal, y escogen el mejor agrupamiento entre todos. En esta investigación los autores propusieron realizar todas esas ejecuciones de manera

concurrente, comenzando con un grupo de centroides iniciales por cada ejecución concurrente.

En la primera iteración de cada ejecución se obtienen agrupamientos intermedios y se actualizan los grupos de centroides. A estos resultados se les evalúa su calidad de agrupamiento mediante la técnica del e*rror al cuadrado* y se escoge la mitad con mejor calidad de agrupamiento, o sea los que menor valor de e*rror al cuadrado* tengan. El siguiente paso propuesto por los autores es realizar una permutación de centroides.

Para esto se escoge un grupo de centroides como base y se calcula la distancia de cada centroide $c_{i,l}$ del grupo a todos los centroides de los demás grupos. El centroide $c_{i,l}$ se intercambia con el centroide cuya distancia entre ellos sea menor. Esta permutación permite eliminar cálculos futuros, ya que al estar trabajando con diversos grupos de centroides y obtener varios agrupamientos concurrentes es muy probable que exista solapamiento entre muchos de ellos. Luego de terminar la permutación sigue un proceso de incubación, mediante el cual se obtiene la otra mitad de centroides necesaria. Para esto los autores propusieron dos heurísticas: *Random Search within a Definite Scope* (RSDS) y *Average of Dissimilar Group Pairs* (ADGP).

Esta mejora fue probada con las bases de datos *Bio_train* con 145,751 objetos en 74 dimensiones (instacnia de la KDD Cup del 2004), *Netflix movies audience* con 17,770 objetos en 1000 dimensiones, y *Lastfm* con 359,330 objetos en 40 dimensiones. Las pruebas con grandes volúmenes de datos realizadas a esta propuesta demostraron su mejor calidad de agrupamiento con respecto al *K-means* y su menor tiempo de ejecución.

2.4 Fase de Convergencia

2.4.1 Early Stop K-means

Esta mejora fue desarrollada en el CENIDET por [8] y consiste en asociar el valor del *error al cuadrado* a una condición de convergencia para así detener el algoritmo cuando se alcance un mínimo local. La condición de convergencia planteada entra en

acción cuando, en dos iteraciones consecutivas, el valor del *error al cuadrado* de la última iteración excede al de la iteración anterior.

Para validar los resultados de la mejora presentada en este artículo se procesaron 6 bases de datos reales procedentes del repositorio UCI; además se procesaron las mismas instancias con las herramientas Weka, Statistical Package for the Social Science (SPSS) y el algoritmo *K-means* estándar para comparar los resultados. Las instancias procesadas fueron: *Vehicle;* con 846 objetos en 18 dimensiones; *Glass,* con 214 objetos en 9 dimensiones; *Diabetes,* con 768 objetos en 8 dimensiones; *Heart;* con 270 objetos en 13 dimensiones; *Wine,* con 178 objetos en 13 dimensiones; *Liver,* con 345 objetos en 6 dimensiones.

Los resultados de la experimentación mostraron que la mejora planteada obtuvo una reducción del número de iteraciones de hasta un 20% en comparación con *K-means;* en el apartado de calidad se obtuvo un aumento en la misma entre el 0.01% y el 7.05%.

2.4.2 Early Stop Heuristic

En el artículo “The Early Stop Heuristic: A New Convergence Criterion for K-means”, [30] propusieron otra condición de convergencia relacionada con el desplazamiento de los centroides en cada iteración. Luego de un análisis del comportamiento de *K-means* los autores observaron que, en general, el mayor desplazamiento de centroides ocurre en la primera iteración y este desplazamiento va disminuyendo a medida que se acerca a la iteración de convergencia. Para sacar provecho de esta característica del algoritmo los autores proponen un umbral como condición de convergencia. El tamaño de este umbral es definido como el 5% del mayor desplazamiento de centroides en la primera iteración. Por ejemplo; si en la primera iteración el centroide que más se desplazó fue el quinto con un desplazamiento de 10 unidades entonces se tiene un umbral $\alpha=0.05*10.$ Si en las siguientes iteraciones el mayor desplazamiento obtenido es menor o igual a α el algoritmo converge.

Para validar el beneficio de usar esta mejora la misma fue comparada con *K-means.* Para esto se utilizaron 8 instancias: 3 sintéticas con 2,500; 10,000 y 40,000 objetos en 2 dimensiones respectivamente. Las instancias *Concrete compressive* y

Skin segmentation provenientes del repositorio UCI, con 1,030 objetos en 8 dimensiones y 245,057 objetos en 3 dimensiones, respectivamente. La instancia *Transaction* con 284,284 objetos en 3 dimensiones, proveniente del repositorio KEEL (http://sci2s.ugr.es/keel/datasets.php). Las instancias *Paris* y *New York* provenientes de Flickr (https://www.flickr.com/map/), con 414,528 y 657,308 objetos en 2 dimensiones, respectivamente.

Los resultados mostraron que la mejora planteada en este artículo reduce el tiempo de procesamiento en un 83.68% con una pérdida de calidad del 1.23% para instancias sintéticas. En el caso de las instancias reales el mejor resultado fue obtenido al procesar la instancia *Transaction* con una reducción del tiempo de procesamiento de un 87.06% y una pérdida de calidad del 2.46% para un agrupamiento de 100 clústers. Para medir la calidad del agrupamiento se toma el valor de la media del *error al cuadrado* como medida, la cual se compara con el mismo valor obtenido al procesar la instancia con el *K-means*.

2.5 Conclusiones del capítulo

En este capítulo se hizo un estudio de algunas de las publicaciones de mayor interés y más actuales relacionadas con las mejoras realizadas a *K-means.* Como se pudo observar las mejoras realizadas están enfocadas en una fase específica del algoritmo, no se ha propuesto combinar mejoras de distintas fases al procesar instancias. Por otra parte, la gran mayoría de las instancias utilizadas para realizar las pruebas son pequeñas, por lo que no se puede asegurar que las mejoras realizadas funcionen bien en Big Data. Las instancias utilizadas para probar las mejoras son disímiles y rara vez coinciden en más de un artículo, por lo que también fue interés en esta investigación utilizar las mismas instancias para analizar el comportamiento de las mejoras independientes así como de sus combinaciones.

Capítulo 3

Características y diseño de la mejora

Como se evidencia en el capítulo anterior las mejoras realizadas a *K-means* son independientes y están enfocadas en una fase; es por eso que en este trabajo se propuso combinar las mejoras de distintas fases y analizar su comportamiento en conjunto.

3.1 Descripción de la mejora

Para dar solución al problema que dio paso a esta investigación se planteó desarrollar una aplicación que permitiera combinar mejoras del algoritmo *K-means.* A esta mejora se denominó *H-Kmeans;* el funcionamiento sería como una caja negra, la cual recibiría como parámetros iniciales:

- La instancia a procesar.
- La cantidad de grupos en los cuáles se desea que el algoritmo agrupe la instancia (k).
- La ruta de destino de los archivos generados por la aplicación.
- La mejora a utilizar por fase.

Luego de ser procesada la instancia la aplicación generará dos archivos:

- “salida.txt”: contiene el tiempo en segundos empleado durante el procesamiento de la instancia, la cantidad de iteraciones realizadas y el valor de la función objetivo (*error al cuadrado*), el cual se toma como medida de la calidad del agrupamiento.

- “agrupamiento.txt”: consta de un valor que indica el grupo al que pertenece el objeto correspondiente con esa posición en la instancia.

3.2 Selección de las mejoras a integrar en H-Kmeans

Durante el estudio del estado del arte relacionado con las mejoras realizadas a *K-means* se detectó que algunas son meta-heurísticas o híbridas. Se decidió tomar mejoras concretas al algoritmo para su integración en *H-Kmeans* y escoger dos mejoras por cada fase.

En la fase de inicialización se seleccionó la mejora *K-means++,* la cual es una de las más reconocidas por la comunidad cientifica al ser citada alrededor de 2,166 veces por otros autores y tener 746 lecturas. Otra mejora seleccionada en esta fase fue *ElAgha,* la cual cuenta con alrededor de 28 citas por otros autores, 98 lecturas y fue publicada en la revista “International Journal of Intelligent Systems and Applications” en el año 2012.

En la fase de clasificación se selecionó la mejora *Enhanced K-means,* la cual también es una mejora muy reconocida por la comunidad internacional al ser citada por alrededor de 145 autores y leída 823 veces; publicada en el año 2006 por la revista “Journal of Zhejiang University-SCIENCE A”, perteneciente a Springer. Otra mejora seleccionada fue *Early Classification*; esta mejora es de autores del CENIDET y fue publicada en el año 2013 por la revista “Journal of Information and Data Management”. Esta mejora fue seleccionada como best paper en esta publicación.

En la fase de convergencia se seleccionó la mejora *Early Stop*, publicada por autores del CENIDET en “International Conference on Computational Science and Its Applications”, en el año 2007, como parte de las “Lecture Notes in Computer Science” de esta conferencia. Este artículo cuenta con 18 citas por otros autores y ha sido leído alrededor de 54 veces. Otra mejora seleccionada en esta fase fue *Early Stop Heuristic*. Esta mejora también es de autores mexicanos y fue publicada en el año 2016 en la revista “AIP Conference Proceedings”. Ha sido sitada 4 veces y cuenta con 47 lecturas hasta el momento.

Según las consultas realizadas en la literatura no se encontraron mejoras en la fase de cálculo de centroides aplicables al propósito de la investigación, por lo tanto esta fase no se tuvo en cuenta y se realiza de la manera establecida por *K-means*. Se decidió incluir a *K-means* en *H-Kmeans* para que así el usuario tenga la posibilidad también de procesar la instancia con el algoritmo original. Todos estos valores de citas y lecturas fueron consultados en la página de ResearchGate (www.researchgate.net).

3.3 Metodología propuesta

En la Figura 3.1 se observa la metodología propuesta en esta investigación. La mejora híbrida *H-Kmeans* presenta una interfaz de usuario, la cual será explicada en el siguiente epígrafe. Primeramente el usuario selecciona la instancia y la cantidad de grupos en que se desea procesar la misma, así como la dirección donde se almacenarán los resultados del procesamiento (1). Seguidamente se seleciona *K-means* o la mejora a combinar por fase (2). *H-Kmeans* conforma una configuración de ejecución atendiendo a los parámetros seleccionados y realiza el procesamiento de la instancia (3). Este proceso es transparente para el usuario. Para terminar la aplicación genera dos archivos con extensión ".txt" en la dirección de salida proporcionada con la información resultante del procesamiento (4).

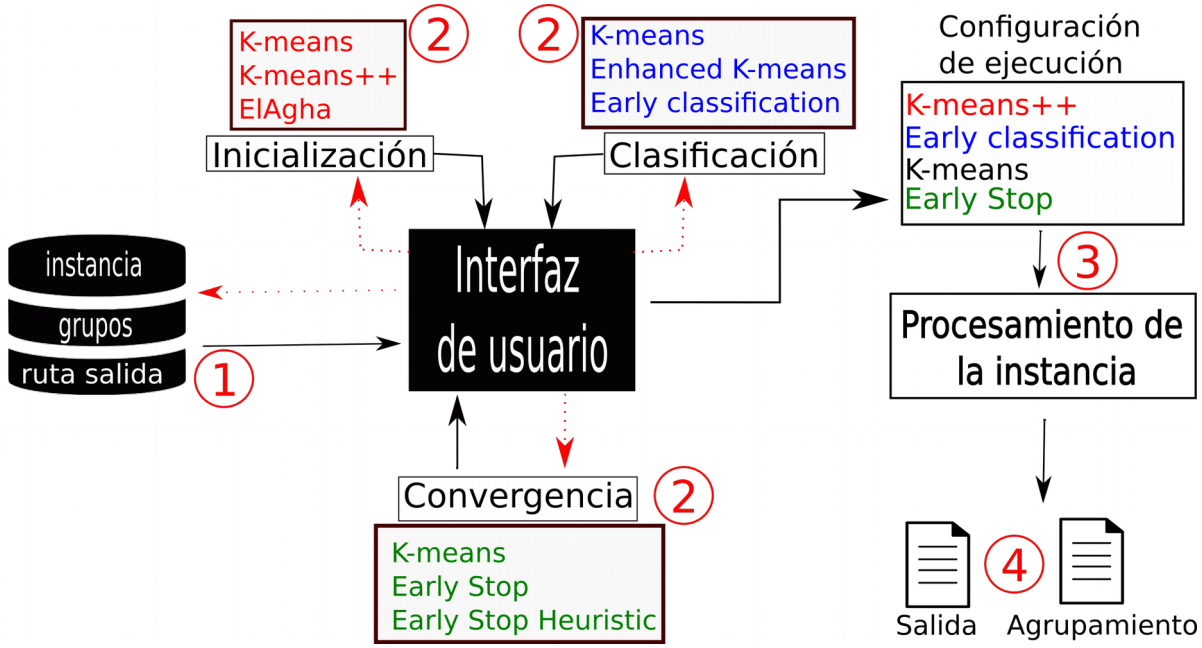

Figura 3.1: Metodología propuesta

3.4 Implementación de la mejora híbrida H-Kmeans

La mejora híbrida *H-Kmeans* fue implementada en lenguaje C para reducir el consumo de recursos del sistema y así obtener una buena relación rendimiento↔tiempo de procesamiento. La aplicación es un ejecutable y funciona sobre sistemas GNU/Linux. Para ver el pseudocódigo de las mejoras seleccionadas ir al Anexo A.

Para una mayor usabilidad se desarrolló una interfaz de usuario programada en GTK3, Figura 3.2. Primeramente el usuario selecciona el archivo .csv que corresponde a la instancia a procesar (1), introduce el número de grupos (2), y selecciona la carpeta donde se guardarán los resultados del procesamiento (3). En la fase de inicialización el usuario puede seleccionar un archivo .csv que contenga los centroides iniciales (4). En cada fase el usuario selecciona la mejora que desee, o bien selecciona *K-means* (5). Al dar click en el botón “Apply” la aplicación comienza el procesamiento de la instancia y una vez terminado se cierra automáticamente. Los resultados del procesamiento se encuentran en la dirección de destino que se fijó en (3). Como resultado se obtienen los archivos “salida” y “agrupamiento”. El archivo “salida” contiene el tiempo de ejecución del procesamiento en segundos, el número de iteraciones y el valor de la sumatoria de la raíz del error al cuadrado, Figura 3.3. El archivo “agrupamiento” contiene un valor por cada línea, ese valor representa el grupo al que pertenece el objeto que se encuentra en esa línea en el archivo de la instancia original, Figura 3.4.

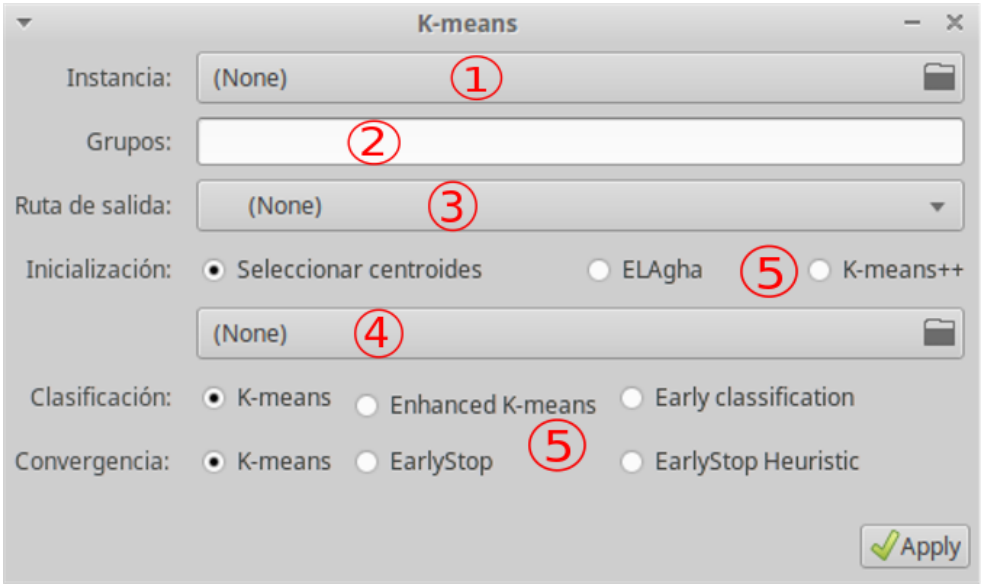

Figura 3.2: Interfaz de la aplicación

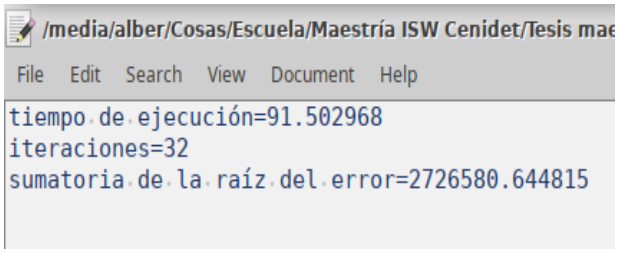

Figura 3.3: contenido del archivo "salida"

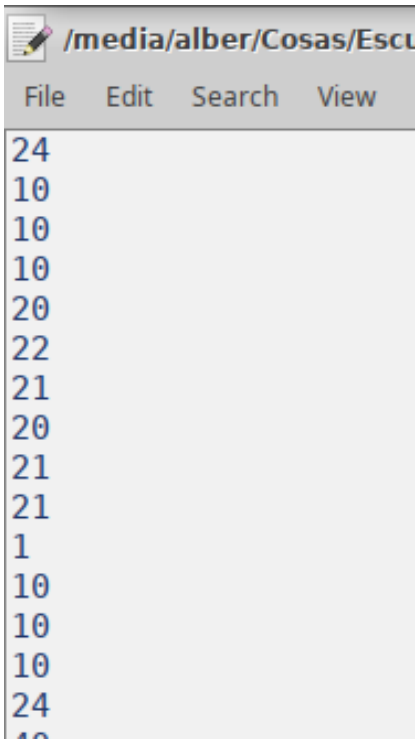

Figura 3.4: contenido del archivo "agrupamiento"

3.5 Análisis de complejidad algorítmica

En esta investigación se recomienda utilizar la combinación de mejoras *K-means++→Early classification→Early Stop* para el procesamiento de las instancias. Las conclusiones que llevan a término esta recomendación son planteadas en el Capítulo 5: Conclusiones y trabajos futuros; pero se cree necesario hacer un paréntesis en este punto para mostrar la complejidad algorítmica de esta combinación.

La complejidad de la combinación es *O(n'kd)* por cada iteración; por lo que podemos decir que la complejidad es *O(n'kdt)*, donde *n'* es la cantidad de objetos a agrupar, *k* los grupos, *d* las dimensiones de los objetos y *t* el número de iteraciones que se realizan. $n' \leqslant n$, donde *n* es la cantidad de objetos totales de la instancia. Que $n' \leqslant n$ está determinado por el funcionamiento de *Early classification*.

Como se analizó en el Capítulo 2 *Early classification* define un umbral y a partir de ahí define qué objetos tienen mayores o menores posibilidades de cambiar de grupos. Los objetos con las menores posibilidades no son tomados en cuenta en los cálculos de distancia hacia el resto de los grupos ya que se considera que ellos permanecen en el grupo asignado. Al estos objetos ser desechados de futuros cálculos se reduce el valor de *n*, por tanto podemos decir que la *n'* involucrada en la complejidad O*(n'kdt)* de la combinación *K-means*++→*Early classification*→*Early Stop* es menor que la *n* involucrada en la complejidad *O(nkdt)* de *K-means*. Además; experimentalmente, en el siguiente capítulo observamos como el tiempo empleado para procesar las diversas instancias de pruebas seleccionadas es mucho menor con esta combinación de mejoras que con *K-means.*

3.6 Conclusiones del capítulo

En este capítulo se definió la metodología para desarrollar la mejora *H-Kmeans*, la cual funciona como una caja negra y realiza la composición de mejoras y el procesamiento de la instancia de manera transparente al usuario, guardando los resultados en archivo específicos para un posterior análisis. También fueron seleccionadas las mejoras de cada fase a integrar en la solución propuesta. Se decidió desarrollar una interfaz gráfica que permitiera una mejor interacción del usuario con la mejora, esto es algo que no se ha visto en la literatura especializada relacionada con mejoras al algoritmo *K-means.*

Capítulo 4

Experimentación y análisis de resultados

El objetivo principal de esta investigación es aplicar la mejora obtenida para procesar instancias de Big Data; sin embargo, no existe un concenso preciso de este término. El artículo *“Big Data: A Survey”* hace un resumen de varias definiciones planteadas sobre el término Big Data. Una primera definición se refiere a instancias que no pueden ser adquiridas, administradas y procesadas por la tecnología, el hardware y el software tradicional dentro de un tiempo razonable. Otra segunda definición, planteada por Apache Hadoop en 2010 y recogida en este artículo, define Big Data como instancias que no pueden ser capturadas, administradas y procesadas por computadoras normales dentro de un ámbito aceptable. Por otra parte, la consultora Mckinsey & Company definió Big Data en 2011 como instancias que no pueden ser adquiridas, almacenadas, y administradas por software de base de datos tradicionales[6].

Los artículos [18], [31] de la literatura especializada, y que hacen referencia a procesar instancias de Big Data, utilizan instancias de 25,000, 10,000, 34,112 y 1,000,000 de objetos.

Lo hasta aquí planteado nos demuestra que no hay un tamaño definido para Big Data, ya que lo que hoy es considerado como tal, al no poder ser procesada por las computadoras actuales, en un tiempo puede ya no ser considerado Big Data debido al desarrollo de tecnologías superiores.

Para validar la idea propuesta en esta investigación se seleccionaron 4 instancias de prueba con características diferentes, las cuales fueron seleccionadas de la literatura consultada y de fuentes de prestigio. En la Tabla 4.1 se resumen las instancias utilizadas; en la primera columna se encuentra el nombre de la instancia, en

la segunda el tipo de instancia (real o sintética), en la tercera la cantidad de objetos (*n*) y en la cuarta el total de dimensiones (*d*). Todas las instancias fueron agrupadas en 50 y 100 grupos respectivamente, a excepción de la instancia *3D road network* la cual también se agrupó en 150 grupos. Se decidió este tipo de agrupamiento para seguir el patrón de grupos utilizado en investigaciones anteriores en el CENIDET, ya que en los artículos consultados no se evidencia un concenso en cuanto al número de grupos a utilizar para las pruebas; los autores utilizan 3, 4, 6, 10, 25, 50, 200, 400 grupos, entre otros. Se realizaron 30 ejecuciones por cada agrupamiento para cada instancia. En total se experimentó con 19 configuraciones de algoritmos distintos teniendo en cuenta *K-means*, las dos mejoras seleccionadas por fase, y la combinación de las mismas. En todas las ejecuciones se utilizaron centroides iniciales diferentes, generados de manera aleatoria. En el caso de los experimentos con *K-means, Early Classification, Enhanced K-means, Early Stop* y *Early Stop Heuristic* los centroides iniciales fueron los mismos en cada ejecución, de cada instancia, respectivamente y fueron asignados manualmente; no se hizo esto para las mejoras *K-means*++ y *ElAgha* porque son de inicialización y generan sus propios centroides iniciales. Los experimentos fueron realizados en una computadora con procesador Intel Core i7-2630QM a 2.0 GHz, 6 GB DDR3 de RAM y sistema operativo Xubuntu 16.04.03 LTS.

Tabla 4.1: Resumen de instancias de prueba

Nombre	Tipo	Objetos (n)	Dimensiones (d)
WIND	Real	126,692	6
3D road network	Real	434,874	4
Household power consumption	Real	2,049,280	4
Letter recognition	Real	20,000	16

La comparación entre los algoritmos se llevó a cabo en términos del tiempo de procesamiento empleado y de la calidad del agrupamiento. Para medir la calidad del agrupamiento se empleó la función del e*rror al cuadrado* (1), donde k representa los grupos y $d(x_j,m_l)$ la distancia del objeto x_j al centroide m_l del grupo al que pertenece. A menor valor del *error al cuadrado* mejor es el agrupamiento de datos.

$$e = \sum_{l=1}^{k} \sum_{x_j \in c_l} d(x_j, m_l) \quad (1)$$

Para comparar la calidad del agrupamiento se llevó a porcentaje el valor del *error al cuadrado* obtenido por cada mejora (e_1), o combinación de mejoras, con respecto al valor obtenido por *K-means* al procesar la misma instancia (e_2). Si el porcentaje es positivo significa que hubo una ganancia de calidad con respecto a *K-means*; por el contrario si el porcentaje es negativo significa que hubo una pérdida de calidad (2).

$$c = \frac{e_1 * 100}{e_2} \quad (2)$$

En el caso del tiempo empleado por las mejoras, o combinaciones de estas, con respecto al empleado por *K-means* al procesar la misma instancia este se definió como el porcentaje de tiempo reducido (3), donde t_1 representa el tiempo empleado por la mejora en cuestión para procesar la instancia y t_2 representa el tiempo empleado por K-means para procesar la instancia. Ejemplo: c = 80%, hubo una reducción de tiempo del 80%.

$$c = 100 - \frac{t_1 * 100}{t_2} \quad (3)$$

4.1 Descripción de las instancias de prueba

WIND

Es una instancia real con 126,692 objetos en 6 dimensiones que recoje datos del viento de los años 2004, 2005 y 2006 en intervalos de 10 minutos para ayudar a los expertos en energía a realizar estudios de viento y estimar la cantidad de energía que podrían generar turbinas eólicas en Estados Unidos. Se puede acceder a la instancia original a través de la siguiente dirección: https://data.nrel.gov/submissions/54. Las dimensiones de la instancia, luego de ser procesadas para este estudio, son longitud, latitud, fracción de viento usable, capacidad, velocidad del viento, y factor de capacidad. Para tener una idea aproximada de la distribución de los datos en la Figura 4.1 se muestra la representación gráfica de las dimensiones longitud y latitud.

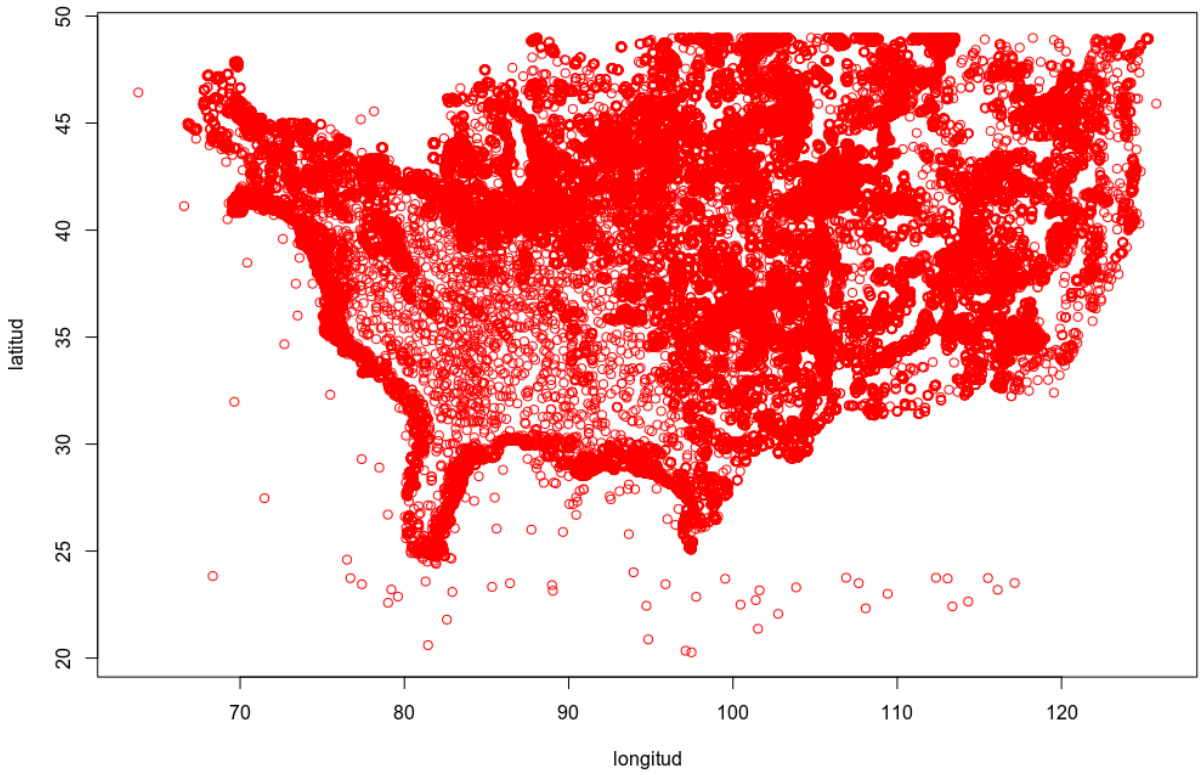

Figura 4.1: Representación de las dimensiones longitud y latitud, instancia WIND

3D road network

Es una instancia real con 434,874 objetos en 4 dimensiones que recoge los datos de una red de carreteras en North Jutland, Dinamarca. Esta instancia es usada con frecuencia para probar algoritmos de estimación de combustibles y de contaminación por CO_2. Las dimensiones de la instancia son: Id de OpenStreetMap para cada borde del grafo, longitud, latitud, y altitud. Se puede acceder a la instancia a través de la siguiente dirección: https://archive.ics.uci.edu/ml/datasets/3D+Road+Network+(North+Jutland,+Denmark). Para tener una idea aproximada de la distribución de los datos en la Figura 4.2 se muestra la representación gráfica de las dimensiones longitud y latitud.

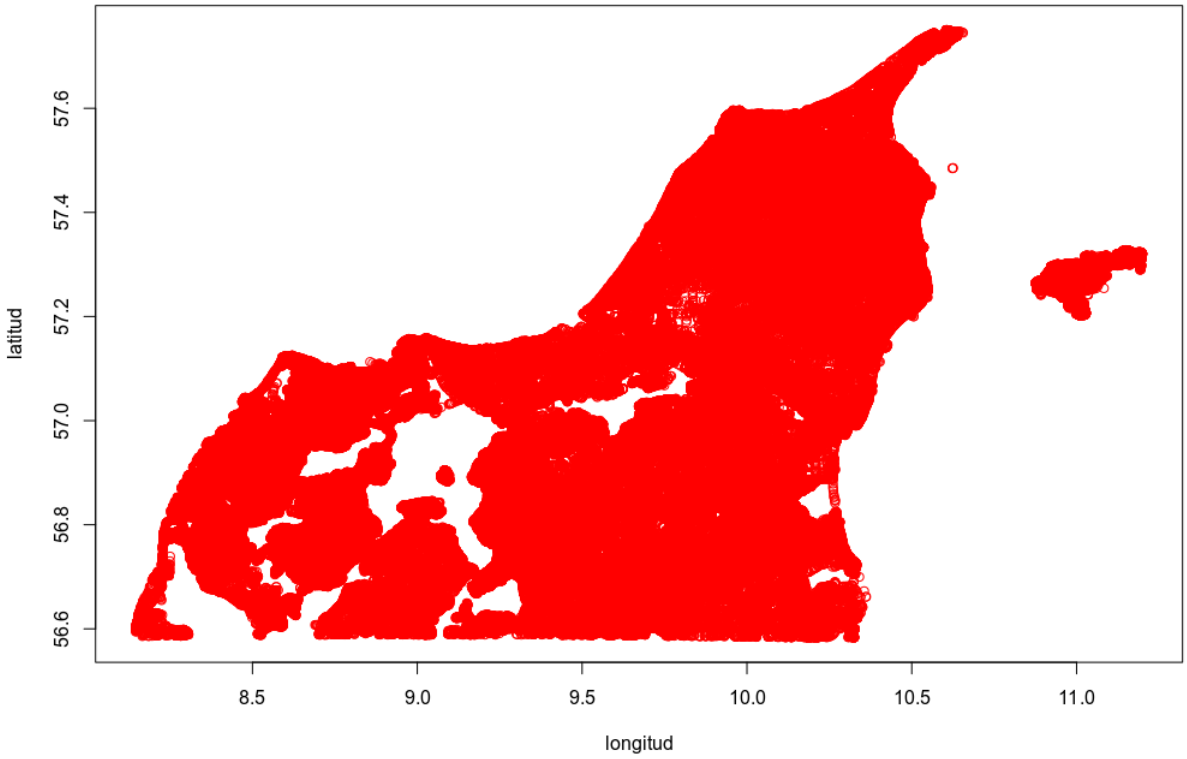

Figura 4.2: Representación de las dimensiones longitud y latitud, instancia 3D road network

Household power consumption

Es una instancia real con 2,049,280 objetos en 4 dimensiones que recoge la medición del consumo eléctrico en hogares desde diciembre de 2006 hasta noviembre de 2010 (47 meses). La instancia original consta de 9 dimensiones, pero para esta investigación se eliminaron 5 dimensiones que no son de utilidad ya que son fechas, horarios, y mediciones que solapan a las mediciones principales y arrojan resultados erroneos o confusos. La instancia original contiene datos perdidos o incompletos los cuales fueron eliminados. Se puede acceder a la instancia original en esta dirección: https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption. Las dimensiones de la instancia, luego de ser procesada para este estudio, son: global active power: representa el consumo promedio de electricidad por minuto en un hogar (kW), global reactive power: representa la energía promedio reactiva global del

hogar (kW), voltaje: promedio de voltaje consumido por minuto en un hogar (V), global intensity: promedio de la intensidad de la corriente por minuto en un hogar (A). Para tener una idea aproximada de la distribución de los datos en la Figura 4.3 se muestra la representación gráfica de las dimensiones voltaje y global intensity.

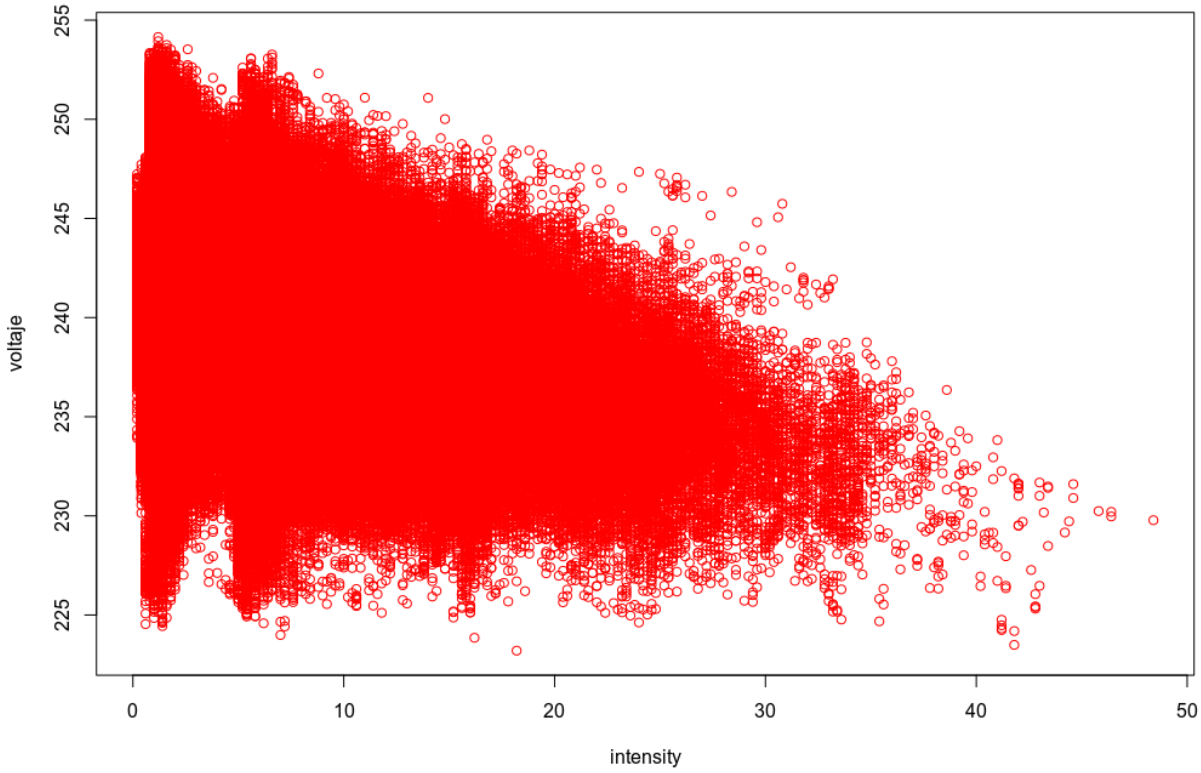

Figura 4.3: Representación de las dimensiones voltaje y global intensity, instancia Household power consumption

Letter recognition

Es una instancia real. El objetivo es identificar cada pixel rectangular blanco y negro mostrado como una de las 26 letras mayúsculas del alfabeto inglés. Las imágenes de los caracteres se basaron en 20 tipos de letras diferentes y cada letra dentro de estos 20 tipos se distorsionaron aleatoriamente para producir un archivo de 20,000 estímulos únicos. Cada estímulo se convirtió en 16 atributos numéricos primitivos (momentos estadísticos y recuentos de bordes) que luego se escalaron para ajustarse a un rango

de valores enteros de 0 a 15. Se puede acceder a la instancia a través de la siguiente dirección: https://archive.ics.uci.edu/ml/datasets/letter+recognition. Para tener una idea aproximada de la distribución de los datos en la Figura 4.4 se muestra la representación gráfica de las dimensiones atributo4 y atributo7.

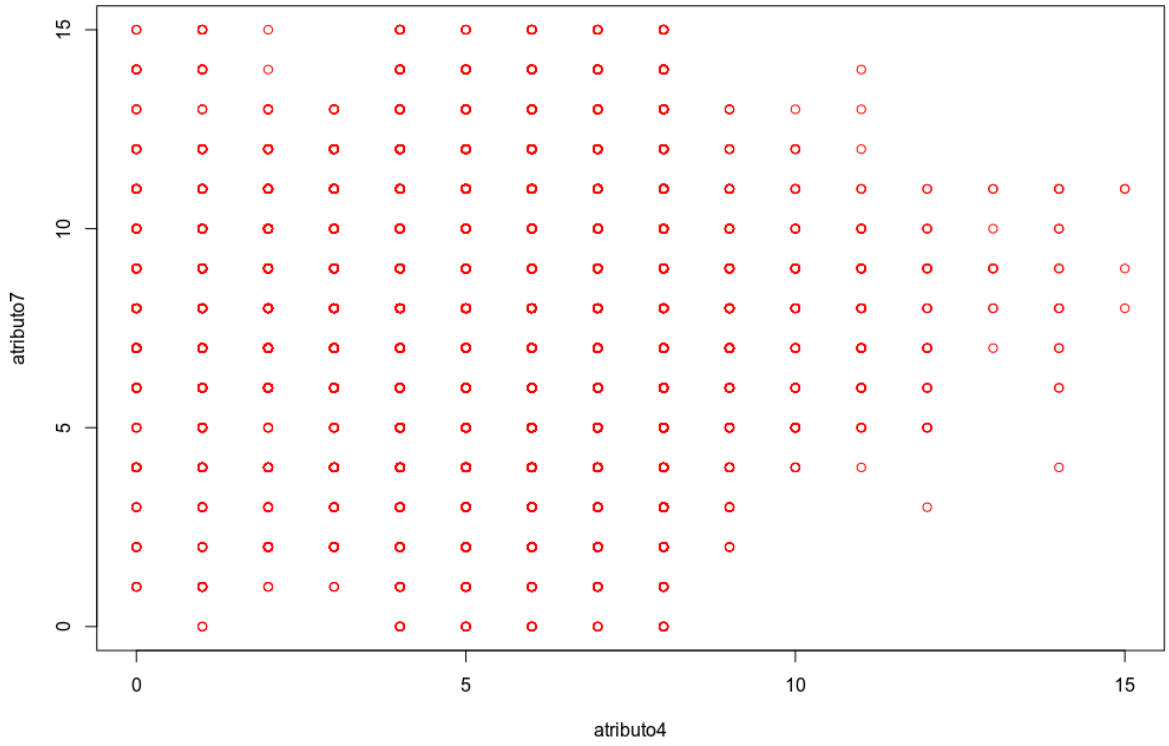

Figura 4.4: Representación de las dimensiones atributo 4 y atributo 7, instancia Letter recognition

4.2 Resultados de la experimentación

El uso de la aplicación a través de la interfaz visual mostrada en la Figura 3.2 solo permite realizar un agrupamiento a la vez; por este motivo, y para optimización del tiempo de experimentación, se utilizó el mismo código pero sin interfaz visual para las pruebas, esto permitió crear scripts de ejecución para las pruebas masivas realizadas. Se confeccionaron tablas comparativas para cada agrupamiento por cada instancia procesada. La distribución de las tablas es como se describe a continuación:

- Columna 1: Nombre de la mejora o las combinaciones.
- Columna 2: Tiempo empleado en procesar la instancia.
- Columna 3: Porcentaje de reducción del tiempo con respecto a *K-means*.
- Columna 4: Valor del error que representa la calidad del agrupamiento.
- Columna 5: Porcentaje de calidad del agrupamiento con respecto a *K-means*. Los valores negativos indican pérdida de calidad mientras que los positivos indican ganancia de calidad en comparación con *K-means*.
- Columna 6: Iteraciones realizadas al procesar la instancia.

En color amarillo destacan las combinaciones donde se obtiene un agrupamiento superior en calidad a *K-means*. En color azul destacan las combinaciones donde se pierde calidad en el agrupamiento.

En las Tablas 4.2 y 4.3 se muestran los resultados del procesamiento de la instancia *WIND* para 50 y 100 grupos respectivamente.

Tabla 4.2: Resultados instancia "WIND", 50 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	129		302324.30		170
K-means++	91	29.5%	279995.19	7.4%	97
ElAgha	70	45.7%	308358.09	-1.9%	83
Enhanced K-means	63	51.2%	304893.09	-0.8%	170
Early classification	15	88.4%	396205.11	-31.1%	41
Early Stop	85	34.1%	307576.05	-2.7%	112
Early Stop Heuristic	5	96.1%	490976.85	-62.4%	6
K-means++ → Early classification	12	90.7%	289252.48	4.3%	29
K-means++ → Enhanced K-means	37	71.3%	281156.71	7.0%	84
ElAgha → Early Classification	10	92.3%	318674.69	-5.4%	29
ElAgha → Enhanced	30	76.7%	309206.66	-2.3%	82

K-means					
K-means++ → Early Classification → Early Stop	11	91.5%	287789.00	4.8%	23
K-means++ → Enhanced → Early Stop	29	77.5%	285042.21	5.7%	58
ElAgha → Early Classification → Early Stop	9	93.0%	317313.25	-5.0%	25
ElAgha → Enhanced → Early Stop	22	82.9%	310175.32	-2.6%	55
K-means++ → Early Classificacion → Early Stop Heuristic	5	96.1%	298829.05	1.2%	7
K-means++ → Enhanced → Early Stop Heuristic	3	97.8%	308655.04	-2.1%	5
ElAgha → Early Classification → Early Stop Heuristic	3	97.8%	348733.80	-15.4%	4
ElAgha → Enhanced → Early Stop Heuristic	2	98.4%	353521.76	-16.9%	4

Tabla 4.3: Resultados instancia "WIND", 100 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	436		239854.75		286
K-means++	148	66.1%	214479.79	10.6%	93
ElAgha	162	62.8%	247540.26	-3.2%	95
Enhanced K-means	146	66.5%	241589.68	-0.7%	222
Early classification	43	90.1%	344352.03	-43.6%	44
Early Stop	267	38.8%	242242.66	-1.0%	179
Early Stop Heuristic	7	98.4%	430973.65	-79.7%	6
K-means++ → Early classification	22	95.0%	220846.74	7.9%	33
K-means++ → Enhanced K-means	49	88.8%	215305.68	10.2%	75
ElAgha → Early Classification	26	94.0%	254104.38	-5.9%	35
ElAgha → Enhanced	63	85.5%	248776.39	-3.7%	97

K-means					
K-means++ → Early Classification → Early Stop	21	95.2%	219909.90	8.3%	25
K-means++ → Enhanced → Early Stop	50	88.5%	215515.24	10.2%	75
ElAgha → Early Classification → Early Stop	24	94.5%	254480.80	-6.1%	30
ElAgha → Enhanced → Early Stop	48	89.0%	248176.86	-3.5%	67
K-means++ → Early Classificacion → Early Stop Heuristic	11	97.5%	226443.85	5.6%	7
K-means++ → Enhanced → Early Stop Heuristic	8	98.2%	233050.97	2.8%	7
ElAgha → Early Classification → Early Stop Heuristic	5	98.9%	298651.20	-24.5%	4
ElAgha → Enhanced → Early Stop Heuristic	4	99.1%	299925.84	-25.1%	4

En las Tablas 4.4, 4.5 y 4.6 se muestran los resultados del procesamiento de la instancia *3D road network* para 50, 100 y 150 grupos respectivamente. Son destacables los resultados obtenidos con esta instancia, donde todas las combinaciones de mejoras reportan ganancias en cuanto a la calidad de hasta un 47% y reducción del tiempo de hasta un 93%.

Tabla 4.4: Resultados instancia "3D road network", 50 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	77		390034114893		45
K-means++	36	53.3%	235152509356	39.7%	20
ElAgha	32	58.4%	275665445045	29.3%	18
Enhanced K-means	24	68.8%	447775730160	-14.8%	33
Early classification	17	77.9%	525937597059	-34.8%	19

Early Stop	22	71.4%	514571714075	-31.9%	20
Early Stop Heuristic	7	90.9%	610079789783	-56.4%	5
K-means++ → Early classification	12	84.4%	234342833780	39.9%	13
K-means++ → Enhanced K-means	13	83.1%	240718773085	38.3%	15
ElAgha → Early Classification	10	87.0%	282682086088	27.5%	13
ElAgha → Enhanced K-means	11	85.7%	282242548509	27.6%	15
K-means++ → Early Classification → Early Stop	12	84.4%	234597395672	39.9%	13
K-means++ → Enhanced → Early Stop	13	83.1%	238042322109	39.0%	14
ElAgha → Early Classification → Early Stop	10	87.0%	282789288150	27.5%	12
ElAgha → Enhanced → Early Stop	14	81.8%	281940541065	27.7%	14
K-means++ → Early Classificacion → Early Stop Heuristic	9	88.3%	238421573342	38.9%	5
K-means++ → Enhanced → Early Stop Heuristic	7	90.9%	242210312329	37.9%	5
ElAgha → Early Classification → Early Stop Heuristic	7	90.9%	291382994396	25.3%	5
ElAgha → Enhanced → Early Stop Heuristic	5	93.5%	291567628920	25.3%	5

Tabla 4.5: Resultados instancia "3D road network", 100 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	155		185913938975		45
K-means++	66	57.4%	111584880533	40.0%	18
ElAgha	66	57.4%	142032813209	23.6%	18
Enhanced K-means	37	76.1%	203934662190	-9.7%	33
Early classification	42	72.9%	229782219273	-23.6%	21

Early Stop	123	20.7%	186323372406	-0.2%	36
Early Stop Heuristic	15	90.3%	282131244828	-51.8%	5
K-means++ → Early classification	25	83.9%	112478723341	39.5%	13
K-means++ → Enhanced K-means	23	85.2%	113500482813	39.0%	14
ElAgha → Early Classification	20	87.1%	143151904160	23.0%	13
ElAgha → Enhanced K-means	21	86.5%	144207679861	22.4%	16
K-means++ → Early Classification → Early Stop	26	83.2%	112501434518	39.5%	13
K-means++ → Enhanced → Early Stop	23	85.2%	113062966580	39.2%	14
ElAgha → Early Classification → Early Stop	19	87.7%	143689370241	22.7%	13
ElAgha → Enhanced → Early Stop	24	84.5%	143781766427	22.7%	16
K-means++ → Early Classificacion → Early Stop Heuristic	18	88.4%	114129776599	38.6%	5
K-means++ → Enhanced → Early Stop Heuristic	14	91.0%	114661972902	38.3%	5
ElAgha → Early Classification → Early Stop Heuristic	14	91.0%	147018190466	20.9%	5
ElAgha → Enhanced → Early Stop Heuristic	11	92.9%	146630193980	21.1%	6

Tabla 4.6: Resultados instancia "3D road network", 150 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	219		134884749001		37
K-means++	93	57.5%	71456415284	47.0%	17
ElAgha	77	64.8%	91296786527	32.3%	15
Enhanced K-means	52	76.3%	155072896146	-15.0%	28
Early classification	67	69.4%	169676734064	-25.8%	21

Early Stop	195	11.0%	136874450709	-1.5%	34
Early Stop Heuristic	20	91.0%	218315574473	-61.9%	5
K-means++ → Early classification	40	81.7%	71494574299	47.0%	13
K-means++ → Enhanced K-means	31	85.9%	71827445445	46.8%	14
ElAgha → Early Classification	31	85.9%	92486207285	31.4%	13
ElAgha → Enhanced K-means	27	87.7%	93476138256	30.7%	13
K-means++ → Early Class → Early Stop	43	80.4%	71657149973	46.9%	12
K-means++ → Enhanced → Early Stop	31	85.9%	72388141273	46.3%	13
ElAgha → Early Classification → Early Stop	33	84.9%	91486689994	32.2%	13
ElAgha → Enhanced → Early Stop	30	86.3%	92978864116	31.1%	12
K-means++ → Early Classificacion → Early Stop Heuristic	29	86.8%	72819995387	46.0%	5
K-means++ → Enhanced → Early Stop Heuristic	22	90.0%	73052893372	45.8%	5
ElAgha → Early Classification → Early Stop Heuristic	24	89.0%	93555678637	30.6%	6
ElAgha → Enhanced → Early Stop Heuristic	16	92.7%	94278169953	30.1%	6

En las Tablas 4.7 y 4.8 se muestran los resultados del procesamiento de la instancia *Household power consumption* para 50 y 100 grupos respectivamente. Aquí se evidencia la gran pérdida de calidad de las combinaciones, donde está presente la mejora *ElAgha,* al procesar instancias grandes. En este caso es una instancia con 2,049,280 objetos.

Tabla 4.7: Resultados instancia "Household power consumption", 50 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	3582		1538070.01		420
K-means++	3284	8.3%	1476969.78	4.0%	316
ElAgha	4816	-34.5%	1683490.14	-9.5%	561
Enhanced K-means	2184	39.0%	1535624.68	0.2%	465
Early classification	113	96.9%	1968532.96	-28.0%	34
Early Stop	1716	52.1%	1543015.71	-0.3%	158
Early Stop Heuristic	55	98.5%	2127186.19	-38.3%	7
K-means++ → Early Classification	149	95.8%	1500365.57	2.5%	33
K-means++ → Enhanced K-means	1836	48.7%	1479393.25	3.8%	311
ElAgha → Early Classification	139	96.1%	2531997.46	-64.6%	37
ElAgha → Enhanced K-means	3028	15.5%	1672938.20	-8.8%	626
K-means++ → Early Classification → Early Stop	98	97.3%	1502070.93	2.3%	23
K-means++ → Enhanced → Early Stop	617	82.8%	1484905.81	3.5%	125
ElAgha → Early Classification → Early Stop	116	96.8%	2576043.51	-67.5%	26
ElAgha → Enhanced → Early Stop	1664	53.6%	1699914.77	-10.5%	288
K-means++ → Early Classificacion → Early Stop Heuristic	50	98.6%	1545394.72	-0.5%	5
K-means++ → Enhanced → Early Stop Heuristic	41	98.9%	1543334.07	-0.3%	6
ElAgha → Early Classification → Early Stop Heuristic	20	99.4%	3668846.63	-138.5%	3
ElAgha → Enhanced → Early Stop Heuristic	17	99.5%	3726636.72	-142.3%	3

Tabla 4.8: Resultados instancia "Household power consumption", 100 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	9484		1106446.69		562
K-means++	7340	22.6%	1077036.72	2.7%	388
ElAgha	15206	-60.3%	1252898.45	-13.2%	837
Enhanced K-means	4434	53.3%	1106896.96	-0.04%	497
Early classification	261	97.3%	1376909.90	-24.4%	38
Early Stop	4286	54.8%	1108505.46	-0.2%	236
Early Stop Heuristic	113	98.8%	1504485.40	-36.0%	8
K-means++ → Early classification	257	97.3%	1087714.17	1.7%	37
K-means++ → Enhanced K-means	2794	70.5%	1079986.35	2.4%	312
ElAgha → Early Classification	280	97.1%	2493934.72	-125.4%	41
ElAgha → Enhanced K-means	7023	26.0%	1255193.40	-13.4%	775
K-means++ → Early Classification → Early Stop	244	97.4%	1091534.64	1.4%	29
K-means++ → Enhanced → Early Stop	1328	86.0%	1080861.90	2.3%	133
ElAgha → Early Classification → Early Stop	244	97.4%	2467814.38	-123.0%	32
ElAgha → Enhanced → Early Stop	5463	42.4%	1260732.25	-13.9%	532
K-means++ → Early Classificacion → Early Stop Heuristic	102	98.9%	1119137.41	-1.2%	5
K-means++ → Enhanced → Early Stop Heuristic	76	99.2%	1128385.22	-2.0%	5
ElAgha → Early Classification → Early Stop Heuristic	37	99.6%	3689130.52	-233.4%	3
ElAgha → Enhanced → Early Stop Heuristic	37	99.6%	3715120.67	-235.8%	3

En las Tablas 4.9 y 4.10 se muestran los resultados del procesamiento de la instancia *Letter recognition* para 50 y 100 grupos respectivamente.

Tabla 4.9: Resultados instancia "Letter recognition", 50 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	24		95138.13		96
K-means++	23	4.2%	94724.79	0.4%	89
ElAgha	28	-16.7%	110027.27	-15.7%	103
Enhanced K-means	9	62.5%	95298.37	-0.2%	74
Early classification	4	83.3%	96132.09	-1.1%	31
Early Stop	19	20.8%	95154.47	-0.02%	76
Early Stop Heuristic	22	8.3%	95407.97	-0.3%	88
K-means++ → Early classification	4	83.3%	95476.79	-0.4%	30
K-means++ → Enhanced K-means	9	62.5%	94811.42	0.3%	78
ElAgha → Early Classification	4	83.3%	112427.07	-18.2%	31
ElAgha → Enhanced K-means	16	33.3%	109964.69	-15.6%	124
K-means++ → Early Classification → Early Stop	4	83.3%	95715.45	-0.6%	29
K-means++ → Enhanced → Early Stop	8	66.6%	94801.22	0.4%	66
ElAgha → Early Classification → Early Stop	4	83.3%	112443.43	-18.2%	28
ElAgha → Enhanced → Early Stop	13	45.8%	109948.88	-15.6%	89
K-means++ → Early Classificacion → Early Stop Heuristic	2	91.7%	97427.26	-2.4%	6
K-means++ → Enhanced → Early Stop Heuristic	1	95.8%	98175.75	-3.2%	6
ElAgha → Early Classification → Early	1	95.8%	119090.15	-25.2%	5

Stop Heuristic					
ElAgha→Enhanced→ Early Stop Heuristic	1	95.8%	121949.20	-28.2%	5

Tabla 4.10: Resultados instancia "Letter recognition", 100 grupos.

Mejora	*Tiempo (seg.)*	*% reducción tiempo*	*Calidad agrupamiento*	*% calidad agrupamiento*	*Iteraciones*
K-means	36		82309.35		76
K-means++	37	-2.8%	82282.45	0.03%	75
ElAgha	59	-63.9%	99579.07	-21.0%	89
Enhanced K-means	11	69.4%	82495.85	-0.2%	61
Early classification	7	80.6%	82904.11	-0.7%	34
Early Stop	31	13.9%	82322.16	-0.02%	64
Early Stop Heuristic	33	8.3%	82457.41	-0.2%	70
K-means++→Early classification	8	77.8%	82809.66	-0.6%	33
K-means++→Enhanced K-means	12	66.7%	82479.74	-0.2%	60
ElAgha→Early Classification	8	77.8%	101849.20	-23.7%	35
ElAgha→Enhanced K-means	24	33.3%	99758.54	-21.2%	112
K-means++→Early Classification→Early Stop	8	77.8%	82885.75	-0.7%	32
K-means++→Enhanced →Early Stop	12	66.7%	82429.51	-0.2%	60
ElAgha→Early Classification→Early Stop	9	75.0%	101725.56	-23.6%	34
ElAgha→Enhanced→ Early Stop	22	38.9%	99776.04	-21.2%	97
K-means++→Early Classificacion→Early Stop Heuristic	4	88.9%	84040.75	-2.1%	8
K-means++→Enhanced →Early Stop Heuristic	3	91.7%	84809.82	-3.0%	7
ElAgha→Early Classification→Early	2	94.4%	108511.80	-31.8%	5

Stop Heuristic					
ElAgha→Enhanced→ Early Stop Heuristic	2	94.4%	109765.80	-33.4%	5

En el anexo B se pueden consultar los resultados de la experimentación de manera gráfica.

4.3 Conclusiones del capítulo

En este capítulo se llevaron a cabo las pruebas para la validación de la investigación. Se definieron instancias que cumplieran con las características de Big Data deseadas y que además fueran disímiles entre sí. Se seleccionaron 4 instancias para la experimentación a partir de fuentes de prestigio internacional. Los resultados de los experimentos llevados a cabo fueron recogidos en tablas y en gráficas para un análisis posterior. En este capítulo se muestran las tablas con los datos resultantes. Dentro de las tablas la información más relevante es el porcentaje de reducción de tiempo y de calidad en el agrupamiento, son estos datos los que nos permiten validar la efectividad y factibilidad de combinar mejoras de distintas fases para procesar instancias, sobre todo instancias grandes. En el anexo B pueden ser consultadas las gráficas con los resultados para una mejor visualización de los mismos.

Capítulo 5

Conclusiones y trabajos futuros

En este capítulo se recogen las conclusiones de la investigación y las recomendaciones para continuar investigando en futuros trabajos.

5.1 Conclusiones

Las pruebas realizadas muestran que es factible la integración de mejoras del algoritmo *K-means*. Los resultados muestran que se puede reducir el tiempo de cálculo considerablemente y en muchos de los casos obtener una calidad de agrupamiento muy buena.

Para el caso de la instancia *Letter recognition*, con 20,000 objetos en 16 dimesiones y 50 grupos, son destacables las combinaciones: *K-means*++→*Enhanced K-means*→*Early Stop* con una reducción del tiempo del 66.6%, reportando una ligera ganancia en cuanto a la calidad del agrupamiento del 0.4%; *K-means*++→*Early Classificacion*→*Early Stop* con una reducción del tiempo del 83.3% y una ligera pérdida de calidad del 0.6%; *K-means*++→*Early Classificacion*→*Early Stop Heuristic* con una reducción del tiempo del 91.7% y una pérdida de calidad en el agrupamiento del 2.4%; *K-means*++→*Enhanced K-means*→*Early Stop Heuristic* con una pédida de calidad en el agrupamiento del 3.2% pero reportando una reducción del tiempo de procesamiento del 95.8%. En el caso de 100 grupos estas mismas combinaciones reportan los mejores resultados en cuanto a reducción de tiempo con pérdidas de calidad mínimas entre el 0.2% y el 3%.

Para el caso de la instancia *WIND*, con 126,692 objetos en 6 dimensiones y 50 grupos, son destacables las combinaciones: *K-means*++→*Early Classification*→*Early*

Stop con una reducción del tiempo del 91.5%, reportando una ganancia significativa, en cuanto a la calidad, del 4.8%; *K-means++→Enhanced K-means→Early Stop* con una reducción del tiempo del 77.5% y una ganancia también significativa del 5.7%; *K-means++→Early Classificacion→Early Stop Heuristic* con una reducción del tiempo del 96.1% y ganancia del 1.2%; *K-means++→Enhanced K-means→Early Stop Heuristic* con una pérdida de calidad del 2.1% pero reduciendo el tiempo de procesamiento en un 97.8%. Para el caso de 100 grupos se mantienen estas mejoras como las de mejores resultados, reportando una reducción del tiempo entre el 88.5% y el 98.2% con ganancias de calidad muy significativas, las cuales van desde el 2.8% hasta el 10.2%.

Para el caso de la instancia *Household power consumption*, con 2,049,280 objetos en 4 dimensiones y 50 grupos, son destacables las combinaciones: *K-means++→Early Classification→Early Stop* con una reducción del tiempo del 97.3% y una ganancia en calidad del 2.3%; *K-means++→Enhanced K-means→Early Stop* con 82.8% de reducción del tiempo de procesamiento y una ganancia de calidad del 3.5%; *K-means++→Enhanced K-means→Early Stop Heuristic* y *K-means++→Early Classificacion→Early Stop Heuristic* con reducciones de tiempo del 98.9% y el 98.6% respectivamente, y pérdidas de calidad insignificantes del 0.3% y 0.5%, respectivamente. Para el caso de 100 grupos el comportamiento es muy similar.

H-Kmeans mantiene los mismos excelentes resultados a medida que aumenta el tamaño de la instancia o los grupos, por lo que podemos afirmar que es factible utilizar *H-Kmeans* para el procesamiento de instancias dentro del paradigma de Big Data. En este sentido se recomienda utilizar la combinación *K-means++→Early Classificacion→Early Stop* ya que reporta excelentes resultados en cuanto a la relación calidad del agrupamiento↔ tiempo empleado para procesar las instancias. Esta combinación presenta una complejidad computacional *O(n'kdt);* donde *n* representa la cantidad de objetos a agrupae, *k* la cantidad de grupos en los que se desea agrupar la instancia, *d* la cantidad de dimensiones o columnas que tiene la instancia, y *t* la cantidad de iteraciones necesarias para agrupar toda la instancia en los grupos deseados. Por otra parte, si se desea obtener resultados en el menor tiempo posible aunque se pueda sacrificar un poco la calidad se recomienda utilizar la combinación *K-means++→Enhanced K-means→Early Stop Heuristic.*

Por otra parte, los resultados obtenidos con la mejora de inicialización *ElAgha* no son alagüeños. En todas las instancias analizadas reporta pérdidas de calidad que llegan a ser muy significativas en muchos casos (Tablas 4.7 y 4.8). Si bien es cierto que las mejoras reducen el tiempo de ejecución sacrificando calidad en algunos casos esta mejora llega a necesitar mucho más tiempo que el algortimo *K-means* (Tabla 4.7, 4.8, 4.9 y 4.10). Debido a esto no recomendamos usar esta mejora para procesar instancias, sobre todo si son instancias grandes y se desea obtener muchos grupos.

5.2 Trabajos futuros

En esta investigación se decidió que el usuario seleccionara las mejoras a combinar, sin embargo se recomienda para futuros trabajo que *H-Kmeans* sea capaz de determinar la mejor combinación de mejoras. Esta tarea es compleja, ya que requiere de mucho tiempo de experimentación pues es necesario tipificar muy bien las instancias y el comportamiento de las mejoras sobre las mismas.

La aplicación desarrollada solo permite hacer un agrupamiento por instancia a la vez; es sabido que para cuestiones de pruebas se suelen hacer varios procesamientos con distintos centroides y grupos, por lo que se recomienda lograr que la aplicación pueda hacer múltiples procesamientos, ya sea mediante una lista de espera o utilizando cada núcleo disponible para procesamiento independiente. También se recomienda la programación en paralelo de H-Kmeans para aprovechar mejor el poder de cómputo de los procesadores actuales, esto a su vez influiría notablemente en la reducción, aún más, del tiempo empleado para el procesamiento de las instancias.

Referencias

[1] Computer Sciences Corporation, “Big Data Infographic”. CSC, 2012.

[2] B. Marr, “Big Data: 20 Mind-Boggling Facts Everyone Must Read”, *Forbes*, 30-sep-2015.

[3] Statista, “Facebook: North America MAU 2016 | Statistic”, *Statista*, 2016. [En línea]. Disponible en: https://www.statista.com/statistics/247614/number-of-monthly-active-facebook-users-worldwide/. [Consultado: 20-feb-2017].

[4] Statista, “Number of LinkedIn users”, *Statista*, 2016. [En línea]. Disponible en: https://www.statista.com/statistics/274050/quarterly-numbers-of-linkedin-members/. [Consultado: 20-feb-2017].

[5] “Internet Live Stats in 1 second”, *Internet Live Stats*, 20-feb-2017. [En línea]. Disponible en: http://www.internetlivestats.com/one-second/#instagram-band. [Consultado: 20-feb-2017].

[6] M. Chen, S. Mao, y Y. Liu, “Big Data: A survey”, *Mob. Netw. Appl.*, vol. 19, núm. 2, pp. 171–209, de enero de 2014.

[7] S. Lloyd, “Least Squares Quantization in PCM”, *IEEE Trans. Inf. Theory*, vol. 28, núm. 2, pp. 129–137, mar. 1982.

[8] J. Pérez, R. Pazos, L. Cruz, G. Reyes, R. Basave, y H. Fraire, “Improving the Efficiency and Efficacy of the K-means Clustering Algorithm Through a New Convergence Condition”, en *Lecture Notes in Computer Science*, Heidelberg, Berlin, 2007, vol. 4707, pp. 674–682.

[9] L. Bottou y Y. Bengio, “Convergence Properties of the K-Means Algorithms”, en *Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]*, 1994, pp. 585–592.

[10] G. Hamerly y C. Elkan, “Alternatives to the k-means algorithm that find better clusterings”, en *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002*, 2002, pp. 600–607.

[11] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, y A. Y. Wu, “An efficient k-means clustering algorithm: analysis and implementation”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, núm. 7, pp. 881–892, jul. 2002.

[12] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, y A. Y. Wu, “A Local Search Approximation Algorithm for K-means Clustering”, en *Proceedings of the Eighteenth Annual Symposium on Computational Geometry*, New York, NY, USA, 2002, pp. 10–18.

[13] D. Pelleg y A. W. Moore, “X-means: Extending K-means with Efficient Estimation of the Number of Clusters”, en *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, USA, 2000, pp. 727–734.

[14] P. Joaquín, P. Carlos Eduardo, B. Leandro, M. Adriana, y H. MIguel Ángel, “Early Classification: A New Heuristic to Improve the Classification Step of K-Means”, *J. Inf. Data Manag.*, vol. 4, núm. 2, pp. 94–103, jun. 2013.

[15] A. Hernández Gómez, “Optimización del algoritmo K-means mediante la integración de heurísticas en las fases de clasificación y convergencia”, Tesis de

maestría en Ciencias de la Computación, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México, 2017.
[16] J. Pérez *et al.*, “Mejora del algoritmo K-means mediante una meta-heurística orientada a la reducción de su complejidad computacional”, presentado en Encuentro Nacional de Ciencias de la Computación, Oaxaca, México, 2014.
[17] A. Moreno Hernández, “Mejora del Algoritmo K-Means Incrementando su Eficiencia en la Fase de Clasificación”, Tesis de maestría en Ciencias de la Computación, Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México, 2013.
[18] J. Pérez *et al.*, “An Efficient Heuristic Applied to the K-means Algorithm for the Clustering of Large Highly-Grouped Instances”, *Comput. Sist.*, vol. 22, núm. 2, pp. 607–619, 2018.
[19] X. Cui, P. Zhu, X. Yang, K. Li, y C. Ji, “Optimized big data K-means clustering using MapReduce”, *J. Supercomput.*, vol. 70, núm. 3, pp. 1249–1259, 2014.
[20] A. David y V. Sergei, “k-means++: The Advantages of careful Seeding”, presentado en Proceeding of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, Louisiana, USA, 2007, pp. 1027–1035.
[21] M. Goyal y S. Kumar, “Improving the Initial Centroids of k-means Clustering Algorithm to Generalize its Applicability”, *J. Inst. Eng. India Ser. B*, vol. 95, núm. 4, pp. 345–350, 2014.
[22] R. Vij y S. Kumar, “Improved k- means clustering algorithm for two dimensional data”, presentado en Second International Conference on Computational Science, Engineering and Information Technology, Coimbatore, India, 2012, pp. 665–670.
[23] M. Yedla, S. Rao Pathakota, y T. M. Srinivasa, “Enhancing K-means Clustering Algorithm with improved initial center”, *Int. J. Comput. Sci. Inf. Technol. IJCSIT*, vol. 1, núm. 2, pp. 121–125, 2010.
[24] E. A. Mohammed y A. Wesam M., “Efficient and Fast Initialization Algorithm for K-means Clustering”, *Mod. Educ. Comput. Sci. Press*, feb. 2012.
[25] A. M. Fahim, A.-B. Salem, F. . Torkey, y M. A. Ramadan, “An efficient enhanced k-means clustering algorithm”, *J. Zhejiang Univ. Sci. A*, pp. 1626–1633, may 2006.
[26] J. Drake y G. Hamerly, “Accelerated k-means with adaptative distance bounds”, presentado en 5th NIPS Workshop on Optimization for Machine Learning, 2012.
[27] G. Hamerly, “Making k-means even faster”, presentado en 2010 SIAM International Conference on Data Mining, Renaissance Columbus Downtown Hotel, Columbus, Ohio, 2010.
[28] P. Fränti, “Clustering datasets”, 2015. [En línea]. Disponible en: http://cs.uef.fi/sipu/datasets/. [Consultado: 30-abr-2017].
[29] L. Chen, Z. Yanfeng, J. Minghai, y Y. Ge, “Mux-Kmeans: multiplex kmeans for clustering large-scale data set”, presentado en The 23rd International ACM Symposium on High-Performance Parallel and Distributed Computing, Vancouver, Canada, 2014, pp. 25–32.
[30] A. Mexicano *et al.*, “The Early Stop Heuristic: A New Convergence Criterion for K-means”, presentado en AIP Conference Proceedings, 2016.
[31] S. Ahammad Fahad y M. Mahbub Alam, “A modified K-means Algorithm for Big Data Clustering”, *Int. J. Comput. Sci. Eng. Technol. IJCSET*, vol. 6, núm. 4, pp. 129–132, Abril 2016.

Anexos

A. Pseudocódigo de las mejoras seleccionadas

A continuación se plasman los pseudocódigos de las mejoras empleadas en el desarrollo de la aplicación resultante de esta investigación. Estos pseudocódigos son desarrollados a partir de la interpretación plasmada en los artículos de las mejoras seleccionadas.

Pseudocódigo *K-means++*

Variables:
n: cantidad de objetos de la instancia.
d: cantidad de dimensiones o columnas que tiene la instancia.
c: total de centroides.
k: total de grupos en que se desea agrupar la instancia.
$\vec{D}$ *:* arreglo que guarda la distancia de cada objeto al centroide más cercano.
$\overrightarrow{Aux}$ *:* matriz que contiene todos los objetos de la instancia.
$\overrightarrow{CR}$ *:* contiene los centroides en cada iteración.

```
1  Procedure K-means++
2    Repeat
3      If c→0 then
4        pos←random[0,n]
5      Else
6        For j:=1 … n do
7          For l:=1 … d do
8            suma ← suma+(Aux_jl − CR_(c−1)l)^2
9          End_For
10         If D_j → 0  Or  suma<D_j  then
11           D_j ← suma
12         End_If
13       End_For
14       For i:=1 … n do
15         sumDist ← sumDist+D_i
```

16	**End_For**
17	*a* ← *random*[*0,sumDist*]
18	**Do** $\sum_{i=1}^{n} \vec{D}_i$ **Until** $\sum_{i=1}^{n} \vec{D}_i \geqslant a$
19	*pos* ← *i*
20	**For** *i:=1 … d* **do**
21	$\overrightarrow{CR_{ci}} \leftarrow \overrightarrow{Aux_{pos\,i}}$
22	**End_For**
23	**For** *i:=pos … n-1* **do**
24	**For** *j:=1 … d* **do**
25	$\overrightarrow{Aux_{ij}} \leftarrow \overrightarrow{Aux_{(i+1)\,j}}$
26	$\vec{D}_i \leftarrow \overrightarrow{D_{i+1}}$
27	**End_For**
28	**End_For**
29	*c*++
30	*n*--
31	**End_If_Else**
32	**Until** *c=k*
33	**End_Procedure**
34	**Return** $\overrightarrow{CR_1}$ …... $\overrightarrow{CR_k}$

Pseudocódigo ElAgha

Variables:
n: cantidad de objetos de la instancia.
d: cantidad de dimensiones o columnas que tiene la instancia.
k: total de grupos en que se desea agrupar la instancia.
$\overrightarrow{minV}$ *:* guarda el mínimo valor de cada columna de la instancia.
$\overrightarrow{maxV}$ *:* guarda el máximo valor de cada columna de la instancia.
$\vec{W}$ *:* peso de cada dimensión
$\overrightarrow{CR}$ *:* contiene los centroides en cada iteración.

1	**Procedure ElAgha**
2	*Find min. and max. value of any dimension d and store in* $\overrightarrow{minV}$ *and* $\overrightarrow{maxV}$
3	**For** *i:=1 … d* **do**

4	$\vec{W}_i \leftarrow \frac{\vec{maxV}_i - \vec{minV}_i}{k}$
5	**End_For**
6	**For** *i:=1 … k* **do**
7	**For** *j:=1 … d* **do**
8	$\vec{CR}_{ij} \leftarrow \vec{minV}_j + i * \vec{W}_j + random[0, \vec{W}_j] - random[0, \frac{\vec{W}_j}{2}]$
9	**End_For**
10	**End_For**
11	**End_Procedure**
12	**Return** $\vec{CR}_1$ ….. $\vec{CR}_k$

Pseudocódigo Enhanced K-means

Variables:
n: cantidad de objetos de la instancia.
d: cantidad de dimensiones o columnas que tiene la instancia.
k: total de grupos en que se desea agrupar la instancia.
l: contador de iteraciones.
$\vec{M}$ *:* matriz que contiene todos los objetos de la instancia.
$\vec{CR}$ *:* contiene los centroides en cada iteración.
$\vec{P}$ *:* distancia de cada objeto al centroide más cercano.
$\vec{C}$ *:* grupo al que pertenece cada objeto.

1	**Procedure Enhanced K-means**
2	$\vec{CR} \leftarrow initial\ centroids$
3	**If** *l→2* **then**
4	**For** *j=1 … n* **do**
5	**For** *i=1 … k* **do**
6	**For** *l=1 … d* **do**
7	$suma \leftarrow suma + (\vec{M}_{jl} - \vec{CR}_{il})^2$
8	**End_For**
9	**If** *i→1* **Or** $\vec{P}_j > suma$ **then**
10	$\vec{P}_j \leftarrow suma$
11	$var \leftarrow i$
12	**End_If**

13	**End_For**
14	$\vec{C}_j \leftarrow var$
15	**End_For**
16	**Else**
17	**For** *j:=1 … n* **do**
18	**For** *l:=1 … d* **do**
19	$suma \leftarrow suma + (\vec{M}_{jl} - \vec{CR}_{jl})^2$
20	**End_For**
21	**If** $suma \leqslant \vec{P}_j$ **then**
22	$\vec{P}_j \leftarrow suma$
23	**Else**
24	**For** *i:=1 … k* **do**
25	**For** *l:=1 … d* **do**
26	$suma \leftarrow suma + (\vec{M}_{il} - \vec{CR}_{il})^2$
27	**End_For**
28	**If** $k \rightarrow 0$ **Or** $\vec{P}_j > suma$ **then**
29	$\vec{P}_j \leftarrow suma$
30	$var \leftarrow k$
31	**End_If**
32	**End_For**
33	$\vec{C}_j \leftarrow var$
34	**End_If_Else**
35	**End_For**
36	**End_If_Else**
37	**End_Procedure**
38	**Return** $\vec{CR}_1$ ….. $\vec{CR}_k$

Pseudocódigo Early Classificacion

Variables:
n: objetos de la instancia.
d: dimensiones o columnas que tiene la instancia.
k: grupos en que se desea agrupar la instancia.

I: iteraciones.
$\vec{M}$ *:* matriz que contiene todos los objetos de la instancia.
$\vec{CR}$: contiene los centroides en cada iteración.
$\vec{NCR}$: contiene los centroides de la iteración anterior.
$\vec{C}$ *:* grupo al que pertenece cada objeto.
$\vec{D}$ *:* Si vale 0 el objeto tiene bajas probabilidades de cambiar de grupo se excluye de futuros cálculos. Si vale 1 ocurre lo contrario.
$\vec{Dist}$ **:** distancia de cada objeto a cada centroide.
$\vec{P}$ *:* distancia del objeto a su centroide más cercano.
$\vec{Desp}$ *:* desplazamiento del centroide con respecto al centroide anterior.
U: valor que define si un objeto tiene o no probabilidades de cambiar de grupo.
$\vec{C2}$ *:* segundo centroide más cercano al objeto.

```
1  Procedure Early Classification
2    CR ← initial centroids
3  Repeat
4    For j:=1 … n do
5      If D_j → 1  Or  I ≤ 2  then
6        For i:=1 … k do
7          For l:=1 … d do
8            suma ← suma + (M_jl − CR_il)^2
9          End_For
10         Dist_ji ← suma
11         If i → 1 Or P_j > Dist_ji  then
12           P_j ← Dist_ji
13           var ← i
14         End_If
15           C_j ← var
16       End_For
17     End_If
18   End_For
19 … recalculate centroids
20   For i:=1 … k do
21     For j:=1 … d do
22       If  I ≥ 2  then
23         op ← op + (CR_ij − NCR_ij)^2
```

```
24        End_If
25        If  $\overrightarrow{CR_{ij}} \neq \overrightarrow{NCR_{ij}}$  then
26          flag ← 1
27        End_If
28      End_For
29      If  $I \geq 2$  then
30        $\overrightarrow{Desp_i} \leftarrow \sqrt{op}$
31        op ← 0
32      End_If
33    End_For
34    If flag → 1 And  $I \geq 2$  then
35      umbralEquidistancia
36      indiceEquidistancia
37    End_If
38  Until flag → 0
39  End_Procedure
```

```
1   Procedure umbralEquidistancia
2     For j:=1 … k do
3       If j → 1 Or  $\vec{D}_j > ma$  then
4         ma ← $\vec{D}_j$
5         sub ← j
6       End_if
7     End_for
8     For j:=1 … k do
9       If  $j \neq \vec{C}_i$  then
10        If flag → 0 then
11          me ← $\vec{D}_j$
12          flag ← 1
13        Else If  $\vec{D}_j > me$  then
14          me ← $\vec{D}_j$
15        End_if_else
16      End_if
```

```
17    End_for
18  U←me+ma
19  End_Procedure umbralEquidistancia
 1  Procedure indiceEquidistancia
 2    For i:=1 … n do
 3      If  D_i → 1  Or l→2 then
 4        menor←0
 5        b←0
 6        For j:=1 … k do
 7          If  j≠C_i  then
 8            If b→0 then
 9              menor ← Dist_ij
10              b←1
11              C2_i ← j
12            Else If  Dist_ij ≤ menor  then
13              menor ← Dist_ij
14              C2_i ← j
15            End_if_else
16          End_if
17        End_for
18        suma←0
19        suma2←0
20        For l:=1 ... d do
21          suma ← suma+(M_il − CR_C_i l)^2
22          suma2 ← suma+(M_il − CR_C2_i l)^2
23        End_for
24        If  |√suma − √suma2| > U  then
25          D_i ← 0
26        Else
27          D_i ← 1
28        End_if_else
```

29	**End_if**
30	**End_for**
31	***End_Procedure indiceEquidistancia***

Pseudocódigo Early Stop

Variables:
n: cantidad de objetos de la instancia.
d: cantidad de dimensiones o columnas que tiene la instancia.
k: total de grupos en que se desea agrupar la instancia.
$\vec{M}$ *:* matriz que contiene todos los objetos de la instancia.
$\vec{CR}$: contiene los centroides en cada iteración.
$\vec{NCR}$: contiene los centroides de la iteración anterior.
$\vec{C}$ *:* grupo al que pertenece cada objeto.
eA: guarda el valor del error en la iteración anterior.
cE: lleva el conteo de las veces que el error de la iteración actual es mayor al error de la iteración anterior.

1	**Procedure Early Stop**
2	$\vec{CR} \leftarrow$ *initial centroids*
3	**Repeat**
4	*classification steps*
5	*recalculate centroids*
6	$e \leftarrow 0$
7	**For** $j:=1 \ldots n$ **do**
8	**For** $l:=1 \ldots d$ **do**
9	$suma \leftarrow suma + (\vec{M}_{jl} - \vec{CR}_{C_j l})^2$
10	**End_For**
11	$e \leftarrow e + \sqrt{suma}$
12	**End_For**
13	**If** $e > eA$ **then**
14	$cE{+}{+}$
15	$eA \leftarrow e$
16	**If** $cE \rightarrow 2$ **then**
17	$flag \leftarrow 0$
18	**End_If**
19	**Else**

```
20      cE←0
21      For i:=1 … k do
22        For j:=1 … d do
23          If  CR_ij ≠ NCR_ij  then
24            flag←1
25          End_If
26        End_For
27      End_For
28      eA←e
29    End_If_Else
30  Until flag→0
31  End_Procedure
```

Pseudocódigo Early Stop Heuristic

Variables:
n: cantidad de objetos de la instancia.
d: cantidad de dimensiones o columnas que tiene la instancia.
k: total de grupos en que se desea agrupar la instancia.
l: contador de iteraciones.
$\vec{M}$: matriz que contiene todos los objetos de la instancia.
$\vec{CR}$: contiene los centroides en cada iteración.
$\vec{C}$ *:* grupo al que pertenece cada objeto.
U: variable auxiliar par definir el umbral de la condición de convergencia.

```
 1  Procedure Early Stop Heuristic
 2    CR ← initial centroids
 3  Repeat
 4    classification steps
 5    recalculate centroids
 6    e←0
 7    For j:=1 … n do
 8      For l:=1 … d do
 9          suma ← suma + (M_jl − CR_C_j l)^2
10      End_For
11      e ← e + √suma
```

```
12    End_For
13    If l→0 then
14      For i:=1 .. k do
15        For j:=1 … d do
16          suma ← suma+(CR_ij − NCR_ij)^2
17        End_For
18        If suma > U then
19          U←suma
20        End_If
21      End_For
22      flag←1
23    Else
24      d←0
25      For i:=1 … k do
26        For j:=1 … d do
27          suma ← suma+(CR_ij − NCR_ij)^2
28        End_For
29        If suma > d then
30          d←suma
31        End_If
32      End_For
33      If  d ≤ 0.05 * U  then
34        flag←0
35      End_If
36    End_If_Else
37  Until flag→0
38  End_Procedure
```

B. Resultados de la experimentación de forma gráfica

En este anexo se recogen los resultados de los experimentos de forma gráfica para una mejor comprensión de la factibilidad de combinar mejoras. Por cada instancia procesada se realizaron dos gráficas, una mostrando el tiempo empleado por las

mejoras y sus combinaciones, y otra mostrando los valores del *error al cuadrado*. Los datos numéricos de los resultados aquí mostrados pueden ser consultados a mayor detalle en las tablas del Capítulo 4. Las conclusiones de los análisis aquí mostrados, y mostrados en las tablas del Capítulo 4, pueden ser consultadas en el Capítulo 5, donde se recogen las conclusiones y trabajos futuros que se desprenden de esta investigación.

En la Figura B.1 observamos los resultados del tiempo empleado al procesar la instancia WIND. Aquí podemos ver como el tiempo disminuye considerablemente cuando se combinan las mejoras.

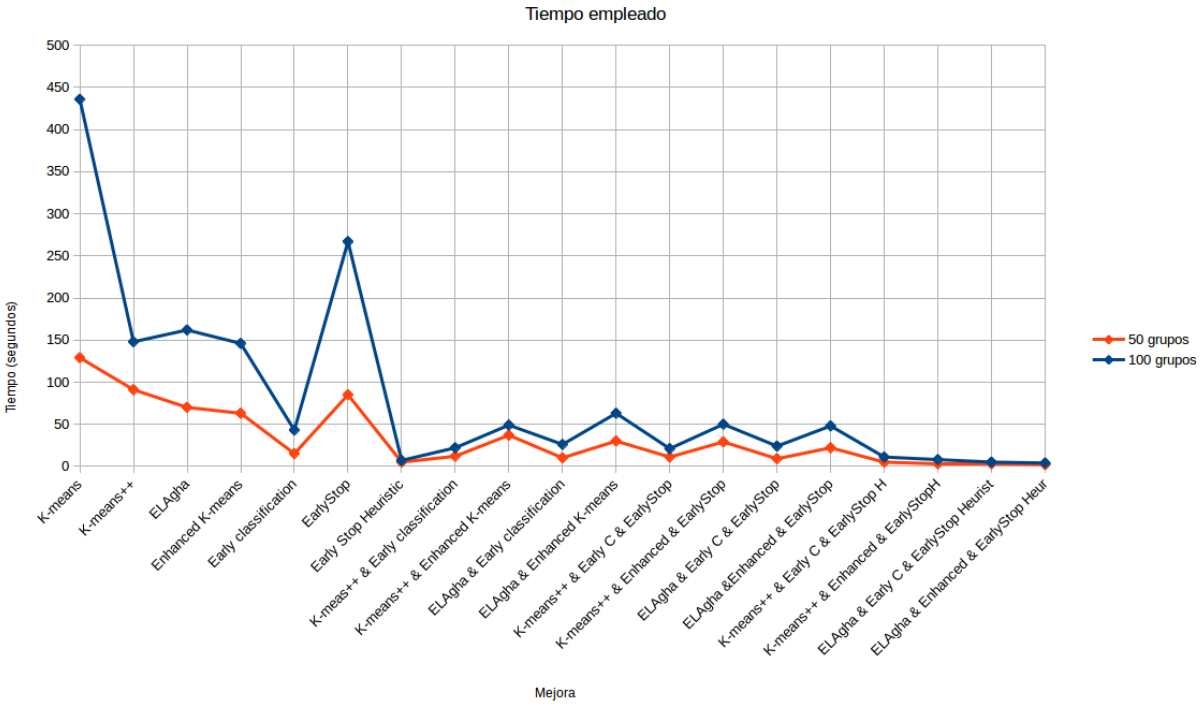

Figura B.1: Tiempo empleado por K-means, las mejoras y sus combinaciones para procesar la instancia WIND.

En la Figura B.2 observamos el valor de la función objetivo, que representa la calidad del agrupamiento, al procesar la instancia WIND. Podemos observar como la calidad se mantiene relativamente estable con respecto a *K-means* y en algunos caso llega a ser menor el valor, lo que indica una mayor calidad en el agrupamiento.

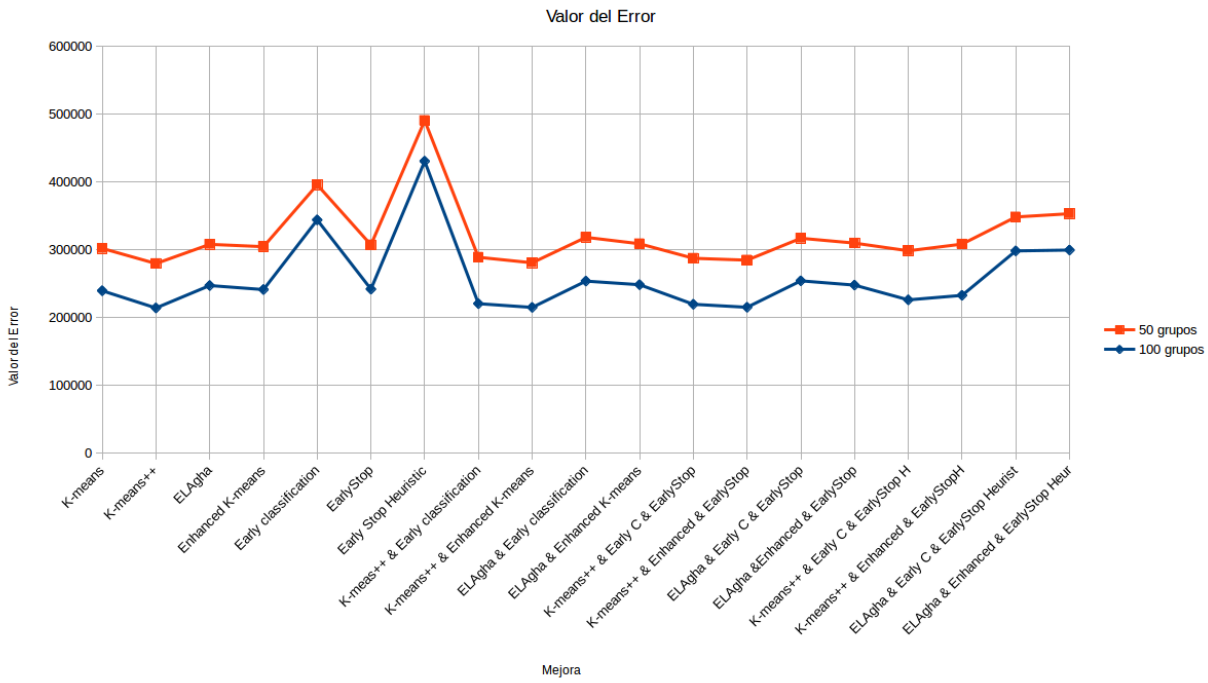

Figura B.2: Función objetivo al procesar la instancia WIND por K-means, las mejoras y sus combinaciones.

En la Figura B.3 observamos los resultados del tiempo empleado al procesar la instancia 3D road entwork. Nuevamente podemos observar como el tiempo disminuye considerablemente cuando se combinan las mejoras.

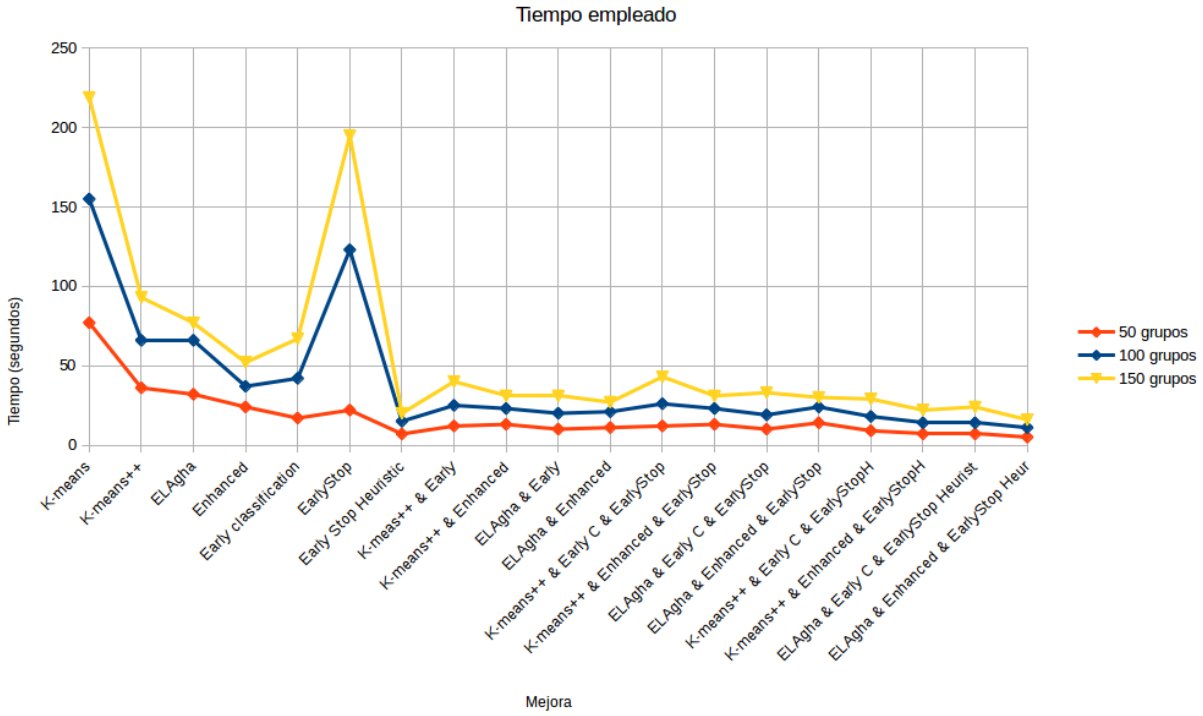

Figura B.3: Tiempo empleado por K-means, las mejoras y sus combinaciones para procesar la instancia 3D road network.

En el caso de la Figura B.4 observamos los resultados de calidad en el agrupamiento para la isntancia 3D road network y podemos observar como la calidad se mantiene relativamente estable al combinar las mejoras.

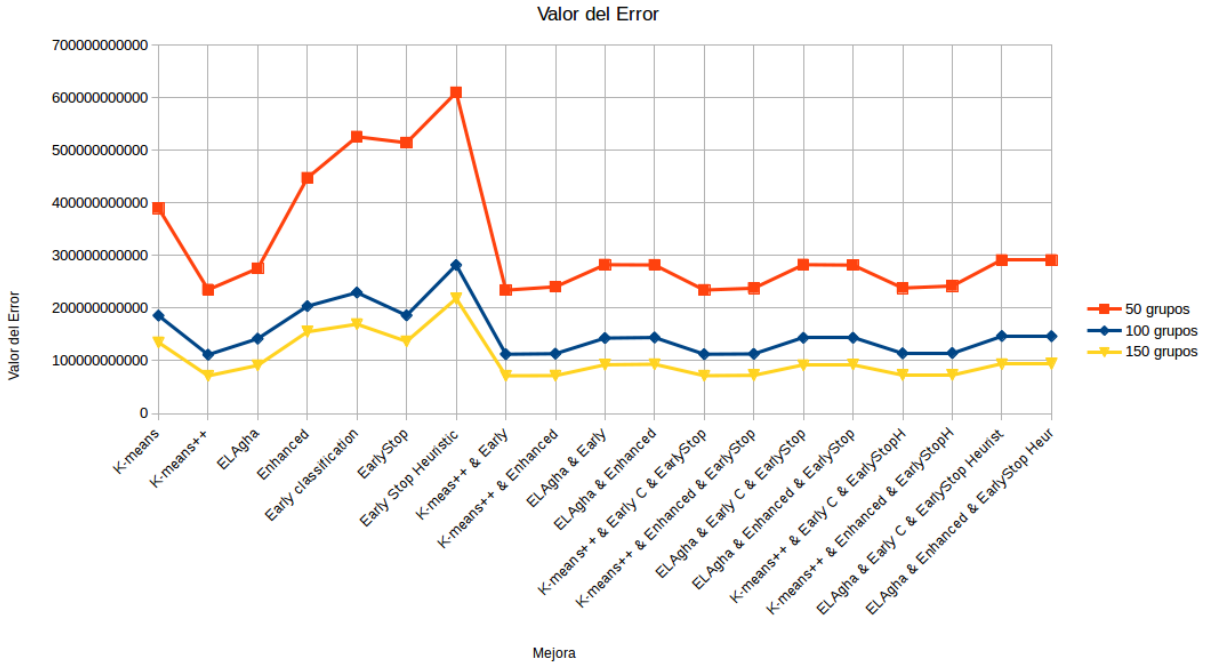

Figura B.4: Función objetivo al procesar la instancia 3D road network por K-means, las mejoras y sus combinaciones.

En la Figura B.5 tenemos los resultados del tiempo empleado al procesar la instancia Household power consumption. A pesar que se observan variaciones en cuanto a la uniformidad de las gráficas con respecto a las mostradas anteriormente podemos observar que el tiempo sigue siendo menor al combinar mejoras que el obtenido al procesar la instancia con el *K-means* original.

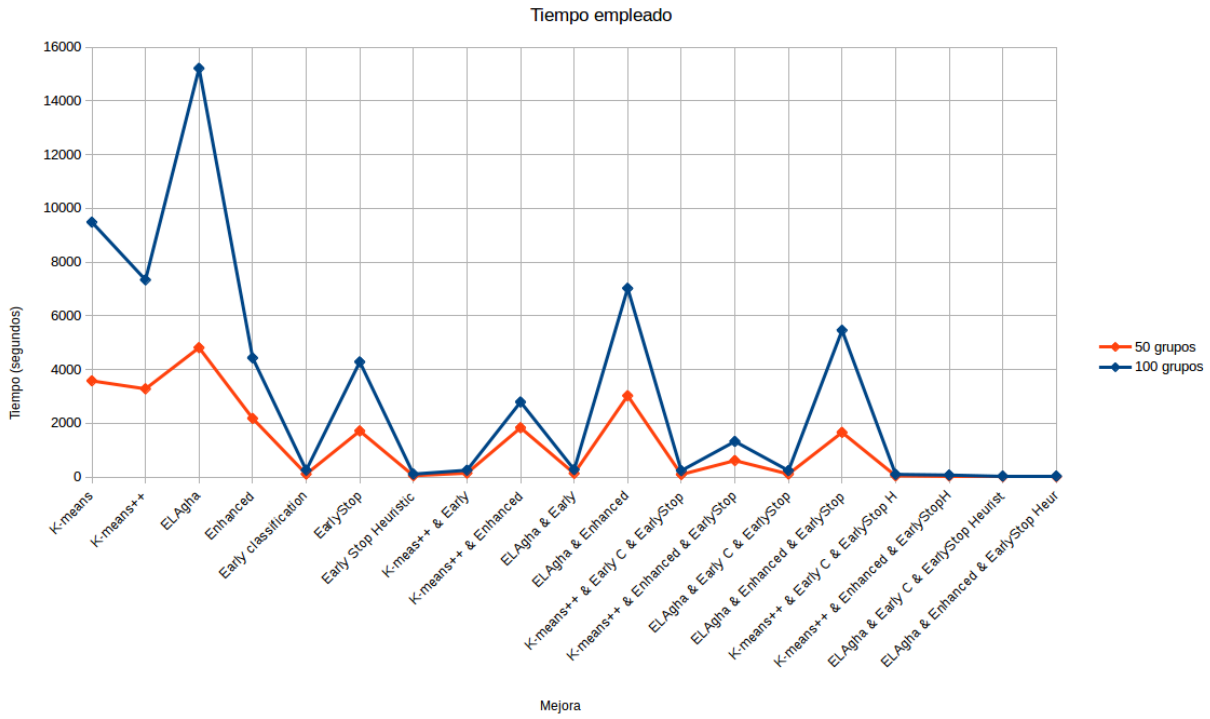

Figura B.5: Tiempo empleado por K-means, las mejoras y sus combinaciones para procesar la instancia Household power consumption.

En la Figura B.6 tenemos los resultados de la calidad del agrupamiento para la instancia Householf power consumption. En este caso nótese la mala calidad del agrupamiento, representada por valores del *error al cuadrado* grandes, de las combinaciones donde está presente la mejora *ElAgha*; sin embargo, en las otras combinaciones donde no está presente *ElAgha* se observa que los resultados se mantienen muy cercanos a los obtenidos en el procesamiento original con *K-means*.

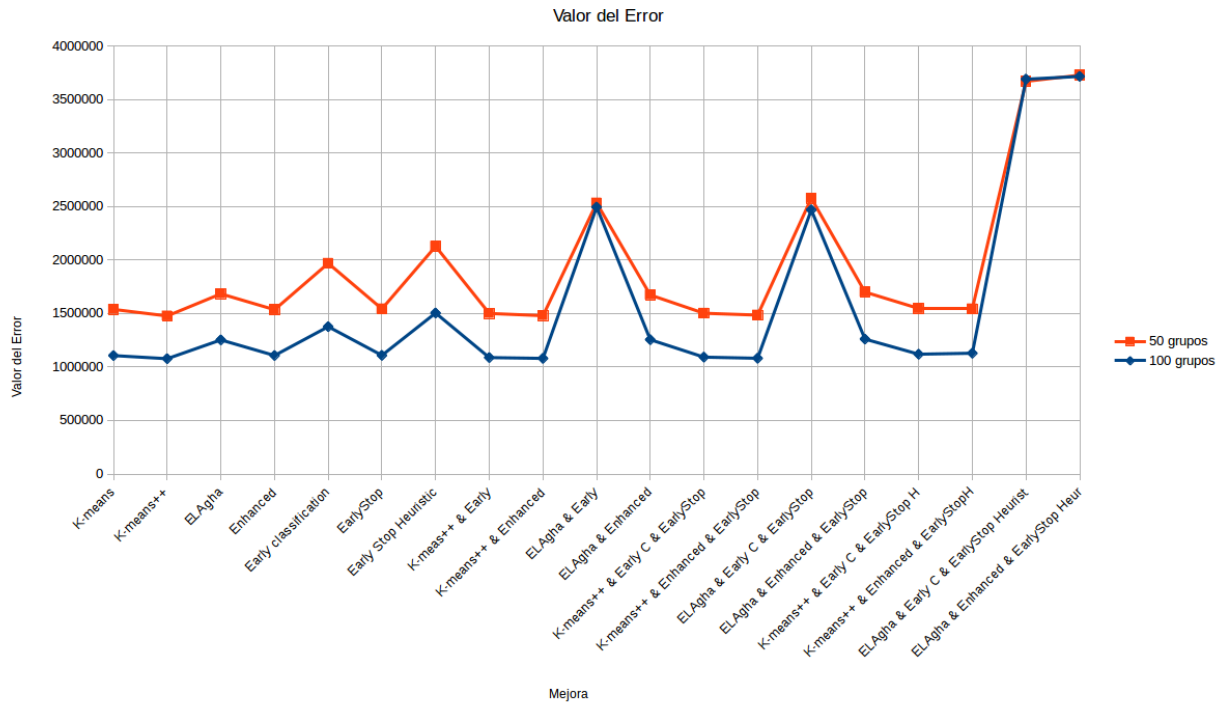

Figura B.6: Función objetivo al procesar la instancia Household power consumption por K-means, las mejoras y sus combinaciones.

En la Figura B.7 se muestran los resultados del tiempo empleado para procesar la instanci y nuevamente observamos como disminuye el mismo a medida que se combinan las mejoras.

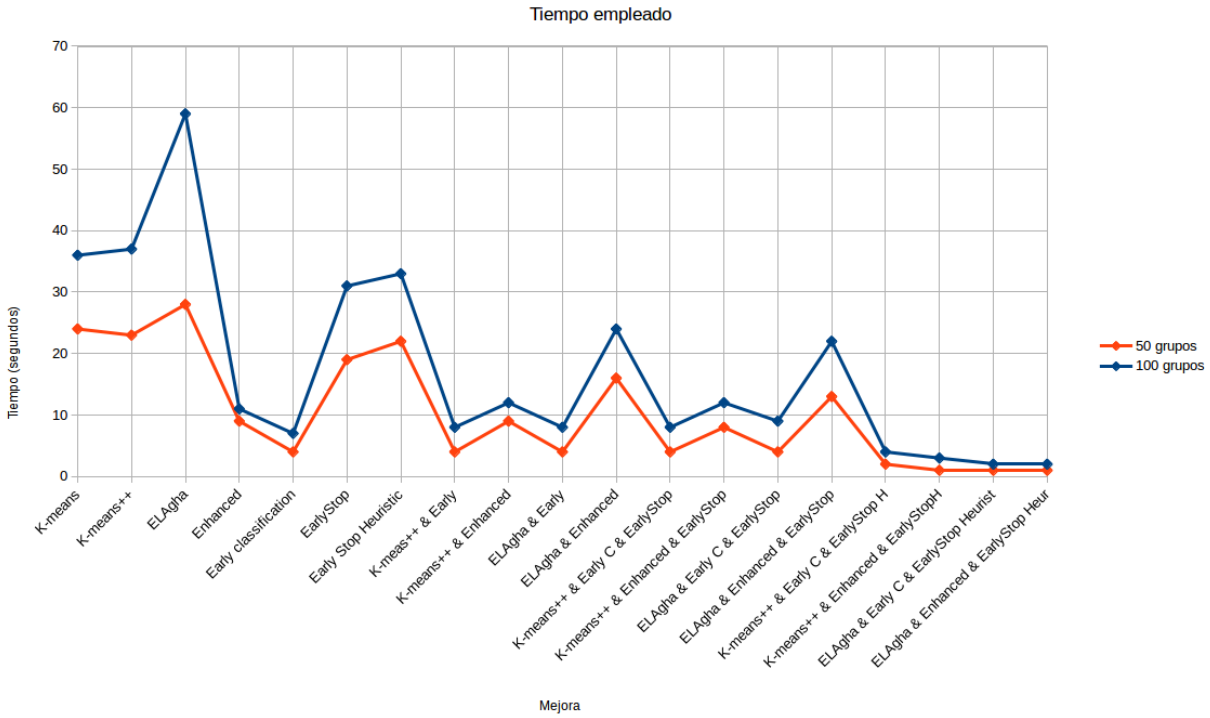

Figura B.7: Tiempo empleado por K-means, las mejoras y sus combinaciones para procesar la instancia Letter recognition.

Nuevamente en la figura B.8 se puede observar la mala calidad del agrupamiento en las combinaciones donde está presente la mejora *ElAgha*; sin embargo en el resto de las combinaciones observamos como la calidad se mantiene estable, sin mucha variación, con respecto a la obtenida por *K-means.*

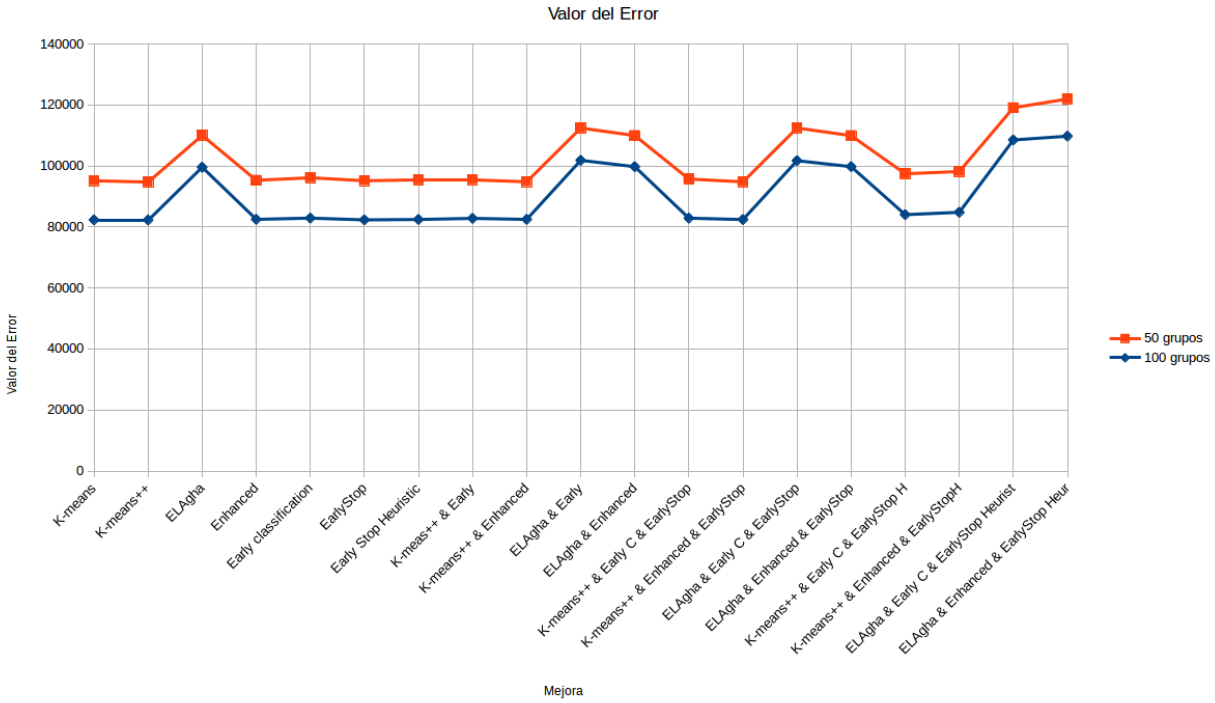

Figura B.8: Función objetivo al procesar la instancia Letter recognition por K-means, las mejoras y sus combinaciones.