ISSN 2007-9737

Una heurística eficiente aplicada al algoritmo K-means para el agrupamiento de grandes instancias altamente agrupadas

Joaquín Pérez-Ortega[1], Miguel Hidalgo-Reyes[1], Noé Alejandro Castro-Sánchez[1], Rodolfo Pazos-Rangel[2], Ocotlán Díaz-Parra[3], Víctor Olivares-Peregrino[1], Nelva Almanza-Ortega[1]

[1]Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, México

[2]Instituto Tecnológico de Cd. Madero, Ciudad Madero, México

[3]Universidad Autónoma del Estado de Hidalgo, Pachuca de Soto, México

jpo_cenidet@yahoo.com.mx, mh@cenidet.edu.mx, ncastro@cenidet.edu.mx, r_pazos_r@yahoo.com.mx, ocotlan@diazparra.net, olivares@cenidet.edu.mx, nelvanely@cenidet.edu.mx

Resumen. Con la presencia cada vez mayor de Big Data surge la necesidad de agrupar grandes instancias. Estas instancias presentan un número de objetos de naturaleza multidimensional, los cuales requieren agruparse en cientos o miles de grupos. En este artículo se presenta una mejora al algoritmo K-means, la cual está orientada a la solución eficiente de instancias con un gran número de grupos y de dimensiones. A dicha mejor heurística se le denomina Honeycomb (HC) y está basada en la relación entre el número de dimensiones y el número de centroides que conforman una vecindad, permitiendo reducir el número de cálculos de distancias objeto-centroide para cada objeto. La heurística se validó resolviendo un conjunto de instancias sintéticas obteniendo reducciones del tiempo de ejecución de hasta un 90 % y con disminución de la calidad menor al 1 %, respecto a K-means estándar. Para instancias reales de baja y alta dimensionalidad, HC obtuvo una reducción del tiempo de ejecución entre 84.74 % y 95.44 % y con una reducción de la calidad entre el 1.07 % y 1.62 %, respectivamente. Los resultados experimentales son alentadores porque esta heurística beneficiaría a aquellos dominios que requieren instancias con valores cada vez mayores de objetos, dimensiones y grupos.

Palabras clave. Algoritmo K-means, complejidad computacional, heurística.

An Efficient Heuristic Applied to the K-means Algorithm for the Clustering of Large Highly-Grouped Instances

Abstract. With the increasing presence of Big Data there arises the need to group large instances. These instances present a number of objects with multidimensional features, which require to be grouped in hundreds or thousands of clusters. This article presents a new improvement to the K-means algorithm, which is oriented to the efficient solution of instances with a large number of clusters and dimensions. This heuristic is called Honeycomb (HC) and it is based on the relationship between the number of dimensions and the number of centroids that form a neighborhood. That is, the heuristic allows the reduction of the number of distance calculations for each object. The heuristic was validated by solving a set of synthetic instances obtaining reductions in execution time of up to 90 % and a quality reduction of less than 1 %, with respect to standard K-means. For real instances of low and high dimensionality, HC obtained a reduction of execution time between 84.74 % and 95.44 % with a quality reduction between 1.07 % and 1.62 %, respectively. The experimental results are encouraging because this heuristic would benefit those domains that require instances with a continuous increase of the number of objects, dimensions and clusters.

Keywords. K-means algorithm, computational complexity, heuristic.

1. Introducción

El agrupamiento de datos es una técnica que divide un conjunto de objetos en grupos, donde los objetos que pertenecen a un grupo son muy similares entre sí, y al mismo tiempo, son diferentes respecto a los objetos que pertenecen a otros grupos. Existen diferentes categorías y algoritmos en relación al agrupamiento de datos [15]. Sin embargo, este artículo se enfoca en la categoría de algoritmos particionales y, concretamente, en el algoritmo estándar K-means de Lloyd [20].

K-means se ha utilizado ampliamente desde hace mucho tiempo y una de las razones de su popularidad es su facilidad en la implementación. No obstante, las instancias de datos actuales son cada vez más grandes y se generan de manera más continua, lo que conduce a retomar el reto de la escalabilidad para K-means. Por ejemplo, el agrupamiento de datos en dominios como la astronomía requiere procesar instancias de billones de objetos [7] y en la bioinformática las instancias de tipo microarray se componen de miles de genes [6]. La dimensionalidad ha crecido significativamente en los últimos años, pasando de 1500 en los años 90 a más de 3 millones en el 2009 [1] y hasta 27 millones en el 2010 [31]. Asimismo, las instancias requieren cada vez más agruparse en miles de grupos [25, 2].

Sea $D = i_1, i_2, i_3, \ldots, i_n$ el conjunto de n objetos con d dimensiones que serán agrupados; $C = c_1, c_2, c_3, \ldots, c_k$ la partición del conjunto D en k subconjuntos y M el conjunto de centroides, donde cada elemento de M, $\mu_1, \mu_2, \mu_3, \ldots, \mu_k$, se asocia con cada k subconjunto. El agrupamiento de datos con K-means consiste de las siguientes etapas: una etapa de inicialización donde se establece el número k de subconjuntos y se realiza la selección aleatoria de los centroides iniciales del conjunto M; una etapa de clasificación que realiza el cálculo de distancia $\|i_p - \mu_l\|^2$ para cada objeto y cada uno de los centroides en M, donde, $p = 1, 2, 3, \ldots, n$ y $l = 1, 2, 3, \ldots, k$; y que asigna el objeto al centroide del grupo cuya distancia es la menor, $i_p \in C_l$, si $\|i_p - \mu_l\|^2 < \|i_p - \mu_j\|^2$, donde $j \neq l$; una etapa de cálculo de centroides $M^{(t)}$ para todos los k subconjuntos; y una etapa de convergencia que establece como criterio de paro que los centroides no cambien entre la iteración actual t y la iteración anterior $t-1$, $(M^{(t)} \equiv M^{(t-1)})$ [24].

De las etapas del algoritmo, la etapa de clasificación es la que presenta mayor costo computacional debido al número de cálculos de distancia que se realizan entre cada objeto del conjunto D y los centroides del conjunto M. La complejidad del algoritmo K-means, en cada iteración t, se define como $\mathcal{O}(nkd)$ [14].

El objetivo principal de esta investigación es mejorar la eficiencia del algoritmo K-means en la etapa de clasificación. Una característica destacable de esta propuesta es que la reducción de cálculos de distancia se relaciona, de manera directa, con los centroides de aquellos grupos que conforman una vecindad con el grupo al cual ya pertenece un objeto. De este modo, se descartan centroides de grupos que no son adyacentes al grupo al cual pertenece un objeto, evitando cálculos de distancia innecesarios y, por consiguiente, reduciendo el número de iteraciones.

La organización de este artículo es la siguiente: en la sección 2 se presenta el estado del arte; en la sección 3 se describe la heurística Honeycomb y se explica cómo ésta modifica etapas específicas del algoritmo K-means; en la sección 4 se describe el proceso experimental y en la sección 5 se presentan los resultados experimentales. La sección 6 presenta un análisis de los resultados y finalmente la sección 7 presenta las conclusiones de esta investigación.

2. Trabajos relacionados

El algoritmo K-means sigue siendo objeto de estudio por parte de la comunidad científica. Desde su aparición, se han presentado muchos artículos relacionados con diferentes aspectos del algoritmo. Si se analizara la importancia del algoritmo K-means en un periodo de tiempo [16], no sería extraño describir la perspectiva, evolución y la continua aparición de nuevos algoritmos de agrupamiento, así como los caminos de investigación a futuro.

De manera general se han identificado dos vertientes importantes. La primera está enfocada

en artículos que analizan la aplicación del algoritmo K-means para resolver un problema de un dominio particular y la segunda está enfocada en artículos que proponen una mejora de una etapa específica del algoritmo. Al considerar esta última vertiente, es importante señalar que el algoritmo K-means se compone de cuatro etapas: inicialización, clasificación, cálculo de centroides y convergencia [17, 30, 27].

En la etapa de inicialización se determina el número de grupos a crear y se eligen los centroides iniciales. Debido a que la elección de los centroides iniciales impacta en la solución del agrupamiento, no existe un método generalizado en la elección de centroides iniciales, pero si la comparación de diferentes métodos de inicialización [23, 3].

Algunas alternativas para calcular los centroides iniciales están basadas en el uso de información de la media y la desviación estándar de los atributos del dataset [19], o utilizando las dos variables que mejor describen el cambio en un dataset con dos ejes del plano cartesiano [9]. Asimismo, el uso de estructuras de datos representa una alternativa de mejora, por ejemplo, con el uso de información de densidad de regiones como en el caso de kd-trees [28]. Otras alternativas han asignado peso a los grupos y optimizan la función objetivo [29], de tal modo que se evita que grupos con alta varianza aparezcan en la solución final.

Para la etapa de clasificación, se han propuesto diferentes enfoques para reducir el número de cálculo de distancias del algoritmo K-means. Por ejemplo, con el uso de una estructura *kd*-tree se almacenan los objetos del dataset y se calcula para cada nodo del árbol un conjunto candidato de centroides. La idea consiste en filtrar los centroides a medida que se recorre el árbol hacia los nodos de niveles inferiores [22, 18].

Derivado del proceso iterativo del algoritmo K-means, se ha identificado también la ventaja de comunicar cuál es el centroide más cercano para un objeto entre la iteración previa y la actual [10], mediante el uso de estructuras de datos. Como resultado de esta comunicación, se ha detectado que determinados objetos permanecen en el mismo grupo en dos iteraciones sucesivas y, por lo tanto, se pueden excluir, así como comprimir los objetos de forma permanente [4].

Al detectar que a medida que aumentan las iteraciones del algoritmo K-means, ciertos objetos permanecen cerca de su centroide asignado y al mismo tiempo alejados de centroides más distantes, la exclusión de centroides resulta interesante gracias al concepto de desigualdad triangular en geometría. Una aplicación de este concepto [26] resultó en la implementación de los algoritmos compare-means y sort-means para evitar comparaciones innecesarias entre objetos y los centroides de los grupos.

En el mismo contexto de la desigualdad triangular, una variante enfocada en un objeto y dos centroides, ha conducido al manejo de límites relacionados con las distancias entre cada objeto y el centroide asignado, y con las distancias entre cada objeto y cada centroide [8, 12, 5]. Estos límites denominados superior e inferior representan un mecanismo para acelerar al algoritmo K-means, a fin de evitar cálculos de distancia objeto-centroide.

La propuesta [8] establece 1 límite superior y *k* límites inferiores para cada objeto. En cada iteración del algoritmo, los límites se actualizan y se recalculan las distancias centroide a centroide. Aunque sus resultados reportan factores de aceleración entre 22 y 107 para $k \geq 20$, su propuesta no resuelve favorablemente instancias de baja dimensionalidad ($d < 20$). No obstante, el algoritmo aumenta su ventaja a medida que se incrementa el valor de k.

La propuesta [12] basada en [8] realiza una mejora mediante el manejo de 1 límite superior y 1 límite inferior para cada objeto. Los resultados son significativos para instancias de baja dimensionalidad, y en algunos casos, de media dimensionalidad al reportar un tiempo de ejecución (por iteración) menor comparado con [20, 22, 8]. Sin embargo, el autor no descarta que un trabajo futuro establezca un límite inferior mayor a 1 y menor que k para cada objeto.

La propuesta [5] presenta la oportunidad de mejora indicada en [12], al establecer un parámetro denominado b, como número de límite inferior entre 1 y k. Este parámetro se relaciona con los b centroides más cercanos para un objeto

y su valor se ajusta, en un intervalo de $k/8$ y $k/4$, en tiempo de ejecución. Los resultados de esta propuesta superan a los reportados por [8, 12] para instancias de media dimensionalidad.

El presente artículo no utiliza directamente la característica de desigualdad triangular, pero sí coincide en el establecimiento de un límite de distancias objeto-centroide para cada objeto. Particularmente, el límite de distancias propuesto (superior + inferior) tiene una relación directa con el número de dimensiones de una instancia. Esto significa que nuestro límite se identifica en la primera iteración y permanece constante hasta que se alcanza el criterio de convergencia del algoritmo. De este modo, el límite establecido no tiene que recalcularse a medida que cambien los centroides como sucede en [8, 12, 5] en cada iteración. No obstante, el límite de distancias presenta un comportamiento similar a [5] debido a que su intervalo se encuentra por arriba de l y por debajo de k.

3. Heurística Panal de abeja

3.1. Origen

La heurística panal de abeja (Honeycomb, HC) está inspirada en la observación de la forma de los grupos creados por el algoritmo K-means. Esta observación se relaciona con la solución del agrupamiento para una instancia sintética, desde la primera iteración y hasta alcanzar el criterio de convergencia. En esta instancia de dos dimensiones y con una distribución uniforme de 40,000 objetos, se observó que los 100 grupos obtenidos presentaban formas de polígonos irregulares de hasta 7 y 8 lados en la primera iteración (véase Figura 1). Sin embargo, la observación completa de la creación de los grupos hasta la última iteración ejecutada por el algoritmo K-means, permitió identificar otras características de interés.

La primera característica está relacionada con el cambio de pertenencia de grupo para un objeto, hacia algunos de los grupos que componen una vecindad adyacente (véase Figura 2). Este cambio de pertenencia sucede con aquellos objetos que

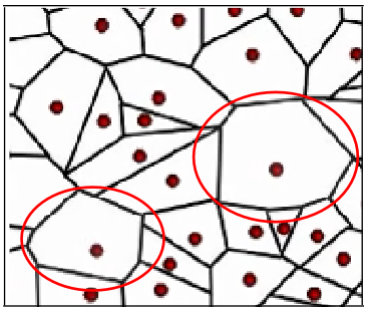

Fig. 1. La forma inicial de algunos grupos es de polígonos irregulares de hasta 7 y 8 lados (círculos de color rojo)

se encuentran en los límites del grupo al cual pertenecen.

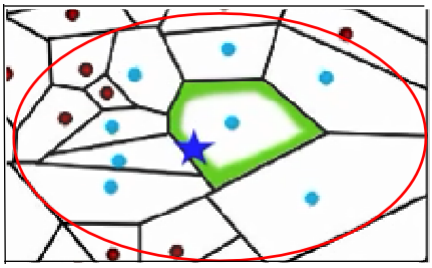

Fig. 2. El área delimitada por el círculo rojo muestra los grupos vecinos adyacentes para el grupo con límites de color verde

En la Figura 3 se muestra el caso del objeto i_5 (estrella de color azul) que pertenece al grupo g_1. El grupo g_1 está representado por el centroide μ_1 denotado con un círculo pequeño de color azul y borde de color naranja. Asimismo, los otros centroides de color azul representan a los grupos que conforman la vecindad adyacente para el grupo g_1.

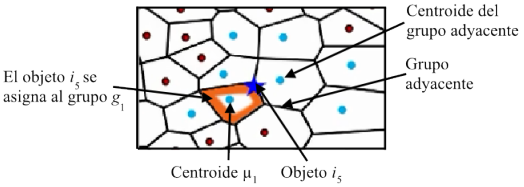

Fig. 3. Ilustración de la pertenencia del objeto i_5 al grupo g_1 y los grupos vecinos adyacentes a este

ISSN 2007-9737

La segunda característica está relacionada con la migración de un objeto hacia grupos vecinos adyacentes. Con base en la Figura 4, se observa que un objeto sólo cambia de grupo hacia aquellos centroides que representan a los grupos vecinos adyacentes y no traspasa a grupos ajenos a la vecindad. En este caso, el objeto i (representado por el punto de color negro) pertenece al grupo representado por el centroide μ_1.

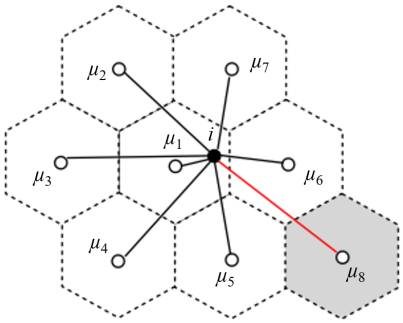

Fig. 4. El objeto i presenta más posibilidades de migrar hacia aquellos grupos vecinos que son adyacentes al grupo al que pertenece

Al considerar el criterio de grupos vecinos adyacentes, el objeto i tiene mayor posibilidad de migrar hacia los grupos representados por los centroides $\mu_2, \mu_3, \mu_4, \mu_5, \mu_6$ y μ_7. Por lo tanto, es poco probable que el objeto i traspase a grupos (μ_8) que no pertenecen a la vecindad de grupos adyacentes, debido a que están más alejados.

La tercera característica está relacionada con los centroides de los grupos vecinos adyacentes que son más cercanos a un objeto. En la Figura 5 se observa que un objeto (estrella de color azul) se encuentra asignado al grupo g_1 en la iteración t (izquierda); sin embargo, en la iteración $(t+1)$, el objeto ha migrado al grupo vecino adyacente más cercano g_2 (centro) y finalmente, en la última iteración, el objeto migra nuevamente al grupo vecino adyacente más cercano g_1 (derecha).

Esto significa que la migración de un objeto, hacia el grupo vecino adyacente más cercano, está determinada por la distancia entre dicho objeto y el grupo vecino adyacente más cercano.

Por ejemplo, en la Figura 6 se observa que los grupos vecinos adyacentes más cercanos al objeto i están representados por los centroides μ_5, μ_6 y

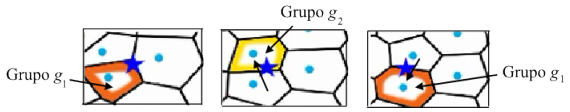

Fig. 5. Migración de un objeto hacia grupos que son adyacentes entre la iteración t, $t+1$ y la última iteración

μ_7. Por lo tanto, el cálculo de distancias se podría realizar únicamente hacia esos grupos y no hacia todos los grupos vecinos adyacentes.

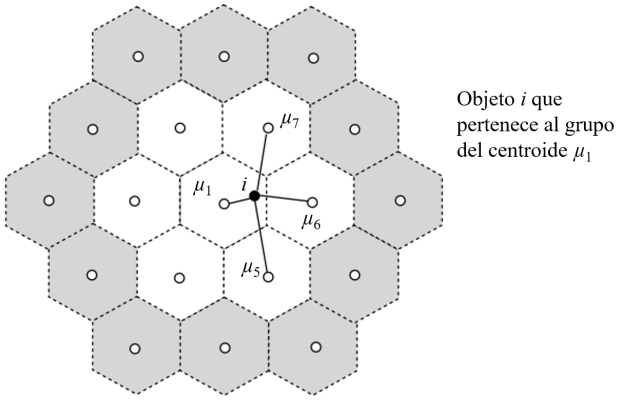

Fig. 6. Grupos vecinos adyacentes más cercanos al objeto i

Por último, la cuarta característica está relacionada con la forma final de los grupos creados por el algoritmo K-means. Esta apariencia se asemeja a la de un panal de abejas debido al predominio de polígonos de seis lados (véase Figura 7).

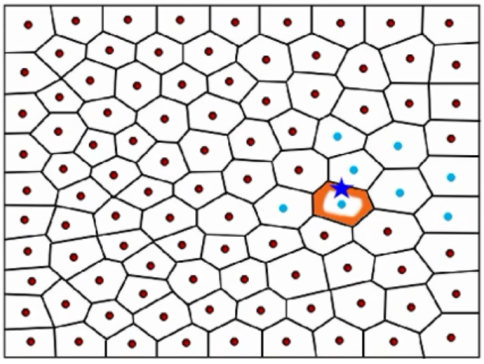

Fig. 7. La forma final de los grupos creados es de polígonos regulares de 6 lados

Para el caso de instancias de dos dimensiones, es conveniente calcular únicamente las distancias entre un objeto y sus 7 centroides más cercanos (seis centroides de grupos vecinos adyacentes más el centroide del grupo al cual ya pertenece dicho objeto). Sin embargo, con el propósito de conocer el comportamiento para instancias con más de dos dimensiones, se repitió la experimentación para una instancia sintética de tres dimensiones y con distribución uniforme.

En este nuevo caso la forma de los grupos creados mostró un cuerpo geométrico de doce caras. Esto nos condujo a determinar que el cálculo de distancias únicamente se realizaría entre un objeto y sus 13 centroides más cercanos (doce centroides de grupos vecinos adyacentes más el centroide del grupo al cual ya pertenece dicho objeto).

Por todo lo anterior, la idea subyacente es que para cada objeto se calculen las distancias únicamente hacia los centroides de los grupos vecinos adyacentes más cercanos. De este modo se obtiene un criterio aplicable a la etapa de clasificación del algoritmo K-means, el cual tiene una relación con la reducción de la complejidad del mismo.

3.2. Descripción de los símbolos utilizados en la heurística HC

A continuación, se describen los símbolos utilizados en la heurística HC.

El Algoritmo 1 muestra el proceso de agrupamiento de la heurística HC. En la etapa de inicialización, la heurística selecciona de manera aleatoria los k centroides (líneas 2-3) y se establece el valor de γ en 10. En la etapa de clasificación y durante las dos primeras iteraciones, la heurística realiza el cálculo de distancias del mismo modo que el algoritmo K-means (líneas 6-26). De manera adicional, en la iteración 2 se identifican los v centroides más cercanos para un objeto (líneas 20-24). A continuación, de la iteración 3 en adelante solo se realizan los cálculos de distancia hacia los v centroides (líneas 28-39). Después de asignar el objeto al grupo más cercano, se recalculan los centroides (línea 40) y el algoritmo finaliza cuando

Algoritmo 1: Proceso de agrupamiento de la heurística HC

input : D,n,k,γ
output: C,$M^{(t)}$

1 *Etapa de inicialización*;
2 Seleccionar aleatoriamente k objetos del conjunto D y asignarlos al conjunto M, $M^{(0)} = \mu_1^{(0)}, \mu_2^{(0)}, \mu_3^{(0)}, \ldots, \mu_k^{(0)}$;
3 Establecer $t \leftarrow 1$, $l \leftarrow 0$, $\gamma \leftarrow 10$, $v \leftarrow \gamma$;
4 *Etapa de clasificación*;
5 **mientras** *not* $(M^{(t)} = M^{(t-1)})$ **hacer**
6 **si** *($t < 3$)* **entonces**
7 **per** $p \leftarrow 1$ **a** n **fai**
8 mindist $\leftarrow \infty$;
9 **per** $j \leftarrow 1$ **a** k **fai**
10 dist $\leftarrow \|i_p - \mu_j\|^2$;
11 **si** *(dist $<$ mindist)* **entonces**
12 mindist $\leftarrow$ dist ;
13 $l \leftarrow j$;
14 **fin**
15 **fine**
16 Asignar el objeto i_p al grupo C_l ;
17 **fine**
18 **fin**
19 **si** $(t = 2)$ **entonces**
20 **per** $p \leftarrow 1$ **a** n **fai**
21 **per** $j \leftarrow 1$ **a** k **fai**
22 dist $\leftarrow \|i_p - \mu_j\|^2$;
23 **fine**
24 Identificar los v centroides más cercanos al objeto i_p;
25 **fine**
26 **fin**
27 **si** $(t \geq 3)$ **entonces**
28 **per** $p \leftarrow 1$ **a** n **fai**
29 mindist $\leftarrow \infty$;
30 **per** $j \leftarrow 1$ **a** v **fai**
31 dist $\leftarrow \|i_p - \mu_j\|^2$;
32 **si** *(dist $<$ mindist)* **entonces**
33 mindist $\leftarrow$ dist ;
34 $l \leftarrow j$
35 **fin**
36 **fine**
37 Asignar el objeto i_p al grupo C_l;
38 **fine**
39 **fin**
40 Recalcular los centroides $M^{(t)}$;
41 $t \leftarrow t + 1$;
42 **fin**
43 Salida $C, M^{(t)}$;

Tabla 1. Símbolos utilizados en la heurística HC

Símbolo	Descripción
γ	Número de grupos vecinos adyacentes que limitan con el grupo al cual pertenece un objeto i
D	Conjunto de n objetos
n	Número de objetos
k	Número de grupos que se van a crear
d	Número de dimensiones
v	Número de centroides de grupos vecinos adyacentes
t	Número de iteraciones
M	Conjunto de centroides asociados con los k grupos
C	Partición del conjunto D
ω	Umbral de número de grupos vecinos adyacentes más cercanos para un objeto i

se alcanza el criterio de convergencia (línea 5 y 43).

No obstante, la dificultad inherente en la visualización de datos con más de tres dimensiones, impide la identificación del número de centroides de grupos vecinos adyacentes para instancias con mayor dimensionalidad. Con base en los trece centroides identificados en la instancia sintética de tres dimensiones, se define ω como el número de grupos vecinos adyacentes más cercanos, el cual establece que los centroides identificados $(\omega < \gamma)$ no rebasan la pérdida de calidad por arriba del 1 %.

4. Proceso experimental

El proceso experimental tiene como objetivo explorar los grupos vecinos adyacentes (γ) para identificar el número de grupos vecinos adyacentes más cercanos (ω) para un objeto. Este proceso tiene interés en el aumento gradual del número de dimensiones de una instancia (d), así como de la relación con el número de grupos a crear (k). Adicionalmente, este proceso busca comparar la eficiencia de los resultados obtenidos por el algoritmo K-means y la heurística HC. La eficiencia se expresa en términos de tiempo de ejecución (medido en milisegundos) y de la calidad de la solución (SSE – Sumatoria del error al cuadrado).

Para cada uno de los casos de prueba, los centroides iniciales se generaron de forma aleatoria y se realizaron 30 ejecuciones tanto para K-means y HC. Para cada ejecución se utilizaron centroides iniciales diferentes. Finalmente, se menciona que la metodología experimental base es la propuesta en [21].

Las ecuaciones 1 y 2 muestran la manera en que se calculan los porcentajes de la diferencia en el tiempo de ejecución (T) y en la calidad de la solución (E):

$$T = \frac{(t_s - t_i)x100}{t_s}, \tag{1}$$

$$E = \frac{(e_s - e_i)x100}{e_s}. \tag{2}$$

4.1. Caso de prueba A

El proceso experimental comenzó con el caso de prueba A que utiliza una instancia de 25,000 objetos distribuidos de forma aleatoria en 2 dimensiones. Los valores de grupos (k) y grupos vecinos adyacentes (γ) fue de 100 y 10, respectivamente. Los valores de los parámetros para el caso de prueba A se muestran en la Tabla 2.

Como resultado preliminar del caso de prueba A, la gráfica de la Figura 8 muestra que el umbral de centroides de grupos vecinos adyacentes más cercanos (ω) presenta dos patrones. El primer patrón muestra un comportamiento variable entre 2 y 30 dimensiones. El umbral alcanza un valor máximo de 9 centroides para 8 dimensiones y, a partir de ahí, disminuye gradualmente. El segundo patrón muestra un comportamiento estable de 2 centroides entre 30 y 200 dimensiones.

El proceso experimental se complementa con los casos de prueba B, C, D y E que presentan una variación en el número de objetos, de dimensiones y de grupos (véase Tabla 3).

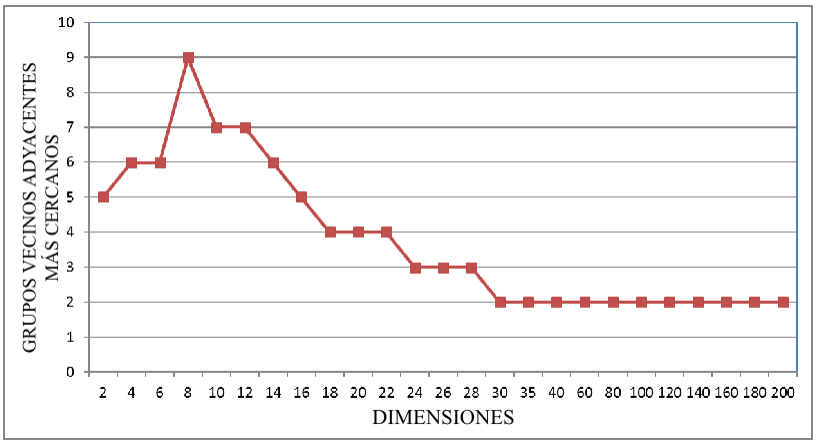

Fig. 8. Resultados para el caso de estudio A

Tabla 2. Configuración del caso de prueba A

Elemento	Valor
Objetos	25,000
Grupos	25,50,75,100
Centroides de grupos vecinos adyacentes	2,3,4,5,6,7,8,9,10,11
Dimensiones	2,3,4,5,6,7,8,9,10,11, 12,13,14,15,16,17,18, 19,20,21,22,23,24,25, 26,27,28,29,30,35,40, 45,50,55,60,65,70,75, 80,85,90,95,100,105, 110,120,130,140,150, 160,170,180,190,200
Rango de valores en cada dimensión	Entre 0.1 y 1.0

Tabla 3. Configuración de casos de prueba adicionales

Caso de prueba	Objetos	Grupos	Dimensiones
B	12,500	25,50,75,100	2,4,6,...,30,60
C	6,500	25,50,75,100	2,4,6,...,30,60
D	3,000	25,50,75,100	2,4,6,...,30,60
E	1,500	25,50,75,100	2,4,6,...,30,60

5. Resultados experimentales

La Tabla 4 reporta, para cada caso de prueba, los porcentajes obtenidos de reducción de tiempo de ejecución, reducción de la calidad al agrupar cada instancia en 100 grupos. Como se puede observar, la ganancia de tiempo aumenta a medida que aumenta el número de dimensiones y el número de objetos. Es destacable que en todos los casos la reducción de la calidad no supera el 1 % y se consigue una reducción promedio del tiempo de ejecución arriba del 80 %.

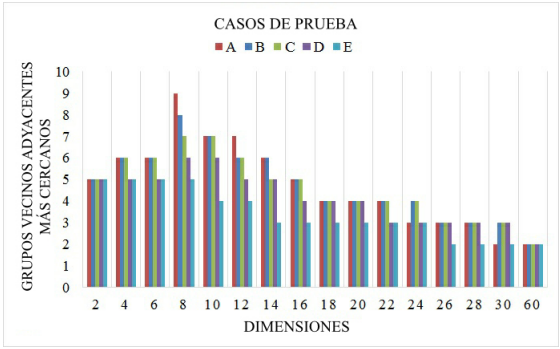

Fig. 9. Comportamiento de ω para los casos de prueba A, B, C, D y E para $k = 100$

La gráfica de la Figura 9 muestra la relación entre el número de dimensiones (d) y el número de grupos vecinos adyacentes más cercanos (ω) para los casos de prueba A, B, C, D y E con 100 grupos.

La Tabla 5 reporta el comportamiento de ω para todos los casos de prueba para 100 grupos y los

Tabla 4. Diferencias en el tiempo de ejecución y calidad de la solución entre K-means y HC para $k = 100$

Objetos	Indices (%)	Dim 2	Dim 6	Dim 8	Dim 10	Dim 14	Dim 18	Dim 22	Dim 26	Dim 30	Dim 60
1500	T	**54.37**	**78.87**	**78.55**	**84.22**	**86.93**	**86.09**	**86.72**	**86.59**	**87.21**	**85.58**
	E	-0.92	-0.80	-0.74	-0.92	-0.96	-0.78	-0.68	-0.93	-0.85	-0.48
3000	T	**60.69**	**84.10**	**83.86**	**83.87**	**87.65**	**88.76**	**90.24**	**90.17**	**90.06**	**90.55**
	E	-0.79	-0.94	-0.84	-0.77	-0.78	-0.84	-0.96	-0.83	-0.74	-0.54
6500	T	**61.02**	**83.41**	**86.27**	**88.00**	**91.45**	**93.44**	**92.75**	**94.28**	**94.16**	**93.80**
	E	-0.84	-0.79	-0.99	-0.88	-0.95	-0.99	-0.83	-0.93	-0.83	-0.67
12500	T	**70.5**	**86.76**	**88.11**	**90.07**	**92.85**	**95.71**	**95.28**	**96.28**	**96.38**	**96.17**
	E	-0.83	-0.86	-0.92	-0.89	-0.89	-0.98	-0.85	-0.93	-0.85	-0.66
25000	T	**69.57**	**88.87**	**87.81**	**92.13**	**93.99**	**96.30**	**96.75**	**97.41**	**97.90**	**97.50**
	E	-0.86	-0.92	-0.86	-0.98	-0.89	-0.98	-0.84	-0.87	-0.98	-0.61

Tabla 5. Comparación de ω en todos los casos de prueba para $k = 100$

Caso de prueba	Dim 2	Dim 6	Dim 8	Dim 10	Dim 14	Dim 18	Dim 22	Dim 26	Dim 30	Dim 60
A	5	6	9	7	6	4	4	3	2	2
B	5	6	8	7	6	4	4	3	3	2
C	5	6	7	7	5	4	4	3	3	2
D	5	5	6	6	5	4	3	3	3	2
E	5	5	5	4	3	3	3	2	2	2
Promedio	**5**	**5.6**	**7**	**6.2**	**5**	**3.8**	**3.6**	**2.8**	**2.6**	**2**
Redondeo	**5**	**6**	**7**	**6**	**5**	**4**	**4**	**3**	**3**	**2**

valores que se muestran abarcan desde 2 y hasta 60 dimensiones.

Por lo tanto, con base en la Tabla 5, la función que modela la relación entre el número de dimensiones y el número de centroides de grupos vecinos adyacentes más cercanos es la siguiente:

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

En la penúltima fila se muestra el promedio obtenido como resultado particular de cada caso de prueba y el número de dimensión. Asimismo, la última fila muestra un redondeo superior cuando el número decimal es mayor o igual a 6 y un redondeo inferior cuando el número decimal es menor o igual a 5.

Como se aprecia en la función, se siguen presentando los dos patrones identificados en la figura 8, sin embargo, el segundo patrón se presenta a partir de las 25 y hasta las 200 dimensiones.

Las Tablas 6 y 7 reportan la comparación de resultados para cuatro instancias del caso de prueba A (2, 10, 30 y 60 dimensiones) y con 100 grupos. La Tabla 6 muestra la diferencia en el tiempo de ejecución entre K-means estándar, Hamerly [12] y Drake [5]. Cabe señalar que para las Tablas 6 y 8 se utilizaron las implementaciones del toolkit Fast K-means [13].

Como se observa en la última columna de la Tabla 6, la propuesta de Drake supera en todos los casos a la propuesta de Hamerly y alcanza valores del 83 % y 76 % respecto a K-means estándar. Debido a que la calidad de la solución es la misma

para K-means, Hamerly y Drake, no se reporta la afectación de la misma.

Sin embargo en la Tabla 7, HC obtiene una reducción destacable del 90 % para las instancias con 10, 30 y 60 dimensiones respecto a nuestra implementación de K-means estándar. Se observa que este incremento en la reducción del tiempo es mayor a medida que aumenta la dimensionalidad y la disminución de la calidad no es mayor al 1 %.

Las Tablas 8 y 9 reportan la comparación de resultados para dos instancias reales denominadas MNIST y Housec8 [11]. La instancia MNIST se compone de 10,000 objetos y 768 dimensiones, mientras que la instancia housec8 se compone de 34,112 objetos y 3 dimensiones. El número de grupos para la instancia MNIST fue de 100 y 200, y para la instancia housec8 fue de 100, 200 y 400 grupos.

Como se observa en la última columna de la Tabla 8, Drake vuelve a superar a Hamerly en todos los casos con un valor promedio de reducción del tiempo de ejecución del 84.48 %. En constraste, la Tabla 9 muestra los porcentajes de reducción del tiempo de ejecución y afectación de la calidad para las dos mismas instancias reales con HC. En el caso de MNIST, HC obtiene una reducción del tiempo de hasta el 95 %, superior a la reportada por Drake. Para la instancia Housec8, se observan resultados muy similares entre HC y Drake, sin embargo, HC supera entre 1 y 2 puntos porcentuales a Drake cuando la instancia Housec8 se agrupa en 200 y 400 grupos.

6. Análisis de resultados

El principal hallazgo radica en que el agrupamiento de instancias con un número alto de dimensiones, no requiere aumentar el número de centroides de grupos vecinos adyacentes más cercanos que intervienen en el cálculo de distancias. A partir de 30 dimensiones, sólo se necesitan los dos centroides de los grupos vecinos adyacentes más cercanos para un objeto.

Algo interesante del proceso experimental es que el valor del umbral (ω) cambia conforme el número de dimensiones de la instancia. Sin embargo, los resultados generales sugieren que el comportamiento estable de dos centroides a partir de 30 dimensiones podría extenderse por arriba de las 200 dimensiones. Concretamente, para instancias con más de 30 dimensiones, cada objeto sólo debe realizar el cálculo de distancia hacia los dos centroides de grupos vecinos adyacentes más cercanos. Esto significa, calcular la distancia hacia el centroide del grupo al cual ya pertenece el objeto y la distancia hacia el siguiente centroide vecino adyacente más cercano.

La heurística HC ofrece una reducción del tiempo de ejecución promedio de hasta el 90 %, y se ha mostrado experimentalmente que su desempeño es mejor cuando las instancias cuentan con 3000 objetos o más, con 10 dimensiones o más y agrupados a partir de 100 grupos. Los resultados experimentales mostraron también casos en los que HC supera la disminución de la calidad por arriba del 1 %. Esto sucedió en aquellas instancias que tienen menos de 12500 objetos, con menos de 8 dimensiones y agrupados en menos de 50 grupos.

No obstante, para los casos mencionados en el párrafo anterior, HC sigue ofreciendo una reducción del tiempo de ejecución de hasta el 50 % con una disminución de la calidad entre 3 % y 7 %. Particularmente, la heurística HC puede aplicarse en el área de Minería de Datos, donde se presentan escenarios que requieren tiempos de respuesta rápidos a expensas de que la calidad del modelo de agrupamiento no sea alta.

La función obtenida está basada en los resultados experimentales de todos los casos de estudio, en los cuales se realizó un incremento gradual del número de objetos y de dimensiones. No se descarta que el desarrollo de más pruebas experimentales permitiría ajustar la función de centroides de grupos vecinos adyacentes.

Es notable señalar que HC presenta una reducción del tiempo de ejecución mayor que las propuestas de Hamerly y Drake, y sin superar la pérdida de calidad por arriba del 1 % para las instancias del caso de prueba A (véase Tabla 7). En el caso de la instancia real MNIST, HC obtiene reducciones de hasta el 90 % y la afectación de la calidad se encuentra en el 1.3 % y 1.6 %. Para el caso de la instancia Housec8 la afectación de la calidad estuvo entre el 1.05 % y 1.25 %.

Tabla 6. Comparación entre K-means, Hamerly y Drake para las instancias del caso A

Dimensiones	K-means	Hamerly		Drake	
	Tiempo (s)	Tiempo (s)	$T(\%)$	Tiempo (s)	$T(\%)$
2	2.5746	0.4122	83.98	0.3808	85.20
10	16.7685	4.8586	71.02	2.0689	87.66
30	35.2713	13.9625	**60.41**	5.9436	**83.14**
60	49.8928	24.4993	**50.89**	11.7129	**76.52**

Tabla 7. Comparación entre K-means y HC para las instancias del caso A

Dimensiones	K-means		Honeycomb (HC)		Honeycomb (HC)	
	Tiempo (s)	SSE	Tiempo (s)	$T(\%)$	SSE	$E(\%)$
2	4.75	861.21	1.44	69.57	868.63	-0.8618
10	60.45	13,118.92	4.75	**92.13**	13,247.57	-0.9806
30	139.22	31,066.38	2.91	**97.90**	31,371.66	-0.9826
60	229.84	46,953.54	5.72	**97.50**	47,242.25	-0.6148

Tabla 8. Comparación entre K-means, Hamerly y Drake para las instancias reales

Instancia	Grupos	K-means	Hamerly		Drake	
		Tiempo (s)	Tiempo (s)	$T(\%)$	Tiempo (s)	$T(\%)$
MNIST	100	114.09	53.99	52.67	18.86	**83.46**
	200	184.70	107.12	42.00	34.65	**81.23**
Housec8	100	14.03	3.11	77.78	1.84	**86.82**
	200	23.45	6.67	71.54	3.23	**86.22**
	400	32.10	10.53	67.18	4.90	**84.71**

Tabla 9. Comparación entre K-means y HC para las instancias reales

Instancia	Grupos	K-means		Honeycomb (HC)		Honeycomb (HC)	
		Tiempo (s)	SSE	Tiempo (s)	$T(\%)$	SSE	$E(\%)$
MNIST	100	742.13	12,989,501.19	33.76	**95.44**	13,200,662.5	-1.6256
	200	1118.72	12,278,127.24	58.59	**94.76**	12,446,547.67	-1.3717
Housec8	100	32.75	219,787.81	4.99	**84.74**	222,149.57	-1.0745
	200	61.05	176,250.33	7.42	**87.83**	178,456.40	-1.2516
	400	92.41	140,105.98	12.76	**86.19**	141,583.00	-1.0542

7. Conclusiones

Con el agrupamiento de instancias de muy alta dimensionalidad, se muestra que es factible mejorar el algoritmo K-means para la solución de instancias con un número alto de grupos. Concretamente, la heurística propuesta se enfoca en reducir la complejidad del algoritmo K-means.

Esta heurística, denominada HC, modela la relación entre el número de dimensiones de una instancia y el número de centroides de grupos más cercanos que conforman una vecindad con el grupo al cual pertenece un objeto. Esto significa que, para cada objeto, únicamente se realizan los cálculos de distancia hacia los centroides de grupos adyacentes más cercanos y se evita el cálculo de distancias objeto-centroide. Para validar esta propuesta, se realizó un proceso experimental cuyo objetivo fue incrementar de manera gradual el número de objetos, de dimensiones y de grupos, y donde la calidad de la solución final no disminuyera en más del 1 %.

Con base en los resultados experimentales se observó que la heurística HC obtiene una

reducción del tiempo de ejecución, de hasta el 90 % y donde el impacto es directamente mayor en aquellas instancias altamente agrupadas (k), y de manera indirecta beneficia también la resolución de instancias con valores altos de objetos (n) y de dimensiones (d).

Finalmente, el número de centroides de grupos vecinos más cercanos sugeridos por la función presenta un buen desempeño para instancias reales al presentarse una reducción de calidad del 1.6 %. Se considera que la heurística HC puede resultar útil en aquellos dominios que requieren agrupar instancias muy grandes, por ejemplo, Big data, Internet de las cosas y redes de sensores.

Agradecimientos

Se agradece al Tecnológico Nacional de México (TecNM) su apoyo para la realización de esta investigación. Los estudiantes de doctorado Miguel Hidalgo-Reyes (becario 187217) y Nelva Almanza-Ortega (becaria 268539) agradecen al Consejo Nacional de Ciencia y Tecnología (CONACYT) el apoyo económico.

Referencias

1. **Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2016).** *Feature Selection for High-Dimensional Data*. Springer.
2. **Broder, A., Garcia-Pueyo, L., Josifovski, V., Vassilvitskii, S., & Venkatesan, S. (2014).** Scalable k-means by ranked retrieval. *Proceedings of the 7th ACM International Conference on Web search and data mining*, ACM, pp. 233–242.
3. **Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013).** A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, Vol. 40, No. 1, pp. 200–210.
4. **Chiang, M.-C., Tsai, C.-W., & Yang, C.-S. (2011).** A time-efficient pattern reduction algorithm for k-means clustering. *Information Sciences*, Vol. 181, No. 4, pp. 716–731.
5. **Drake, J. & Hamerly, G. (2012).** Accelerated k-means with adaptive distance bounds. *5th NIPS Workshop on Optimization for Machine Learning*.
6. **Dua, S. & Chowriappa, P. (2013).** *Data Mining for Bioinformatics*. CRC Press.
7. **Edwards, K. J. & Gaber, M. M. (2014).** *Astronomy and Big Data*. Studies in Big Data. Springer.
8. **Elkan, C. (2003).** Using the triangle inequality to accelerate k-means. *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 147–153.
9. **Erisoglu, M., Calis, N., & Sakallioglu, S. (2011).** A new algorithm for initial cluster centers in k-means algorithm. *Pattern Recognition Letters*, Vol. 32, No. 14, pp. 1701–1705.
10. **Fahim, A., Salem, A., Torkey, F., & Ramadan, M. (2006).** An efficient enhanced k-means clustering algorithm. *Journal of Zhejiang University SCIENCE A*, Vol. 7, No. 10, pp. 1626–1633.
11. **Fränti, P. (2015).** Clustering datasets. `http://cs.joensuu.fi/sipu/datasets/`.
12. **Hamerly, G. (2010).** Making k-means even faster. *SIAM International Conference on Data Mining*, SIAM, pp. 130–140.
13. **Hamerly, G. (2014).** Fast k-means toolkit. `https://github.com/BaylorCS/baylorml`.
14. **Hamerly, G. & Drake, J. (2015).** Accelerating Lloyd's algorithm for k-means clustering. In **Celebi, M. E.**, editor, *Partitional Clustering Algorithms*. Springer, Shreveport, LA, USA, pp. 41–78.
15. **Han, J., Kamber, M., & Pei, J. (2012).** *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, third edition.
16. **Jain, A. K. (2010).** Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, Vol. 31, No. 8, pp. 651–666.
17. **Jain, A. K. & Dubes, R. C. (1988).** *Algorithms for Clustering Data*. Prentice-Hall, Inc.
18. **Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002).** An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 7, pp. 881–892.
19. **Khan, S. S. & Ahmad, A. (2004).** Cluster center initialization algorithm for K-means clustering. *Pattern Recognition Letters*, Vol. 25, No. 11, pp. 1293–1302.
20. **Lloyd, S. (1982).** Least squares quantization in PCM. *IEEE Transactions on Information Theory*, Vol. 28, No. 2, pp. 129–137.

21. **McGeoch, C. C. (2012).** *A Guide to Experimental Algorithmics*. Cambridge University Press.
22. **Pelleg, D. & Moore, A. (1999).** Accelerating exact k-means algorithms with geometric reasoning. *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 277–281.
23. **Peña, J. M., Lozano, J. A., & Larrañaga, P. (1999).** An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, Vol. 20, No. 10, pp. 1027–1040.
24. **Pérez, J., Pires, C. E., Balby, L., Mexicano, A., & Hidalgo, M. (2013).** Early classification: A new heuristic to improve the classification step of k-means. *Journal of Information and Data Management*, Vol. 4, No. 1, pp. 1–10.
25. **Pham, D., Dimov, S., & Nguyen, C. (2005).** Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 219, No. 1, pp. 103–119.
26. **Phillips, S. J. (2002).** Acceleration of k-means and related clustering algorithms. In **Mount, D. M. & Stein, C.**, editors, *Algorithm Engineering and Experiments*. Springer Berlin Heidelberg, pp. 166–177.
27. **Reddy, C. K. & Vinzamuri, B. (2013).** A survey of partitional and hierarchical clustering algorithms. In **Aggarwal, C. C. & Reddy, C. K.**, editors, *Data Clustering Algorithms and Applications*. CRC Press, pp. 87–110.
28. **Redmond, S. J. & Heneghan, C. (2007).** A method for initialising the K-means clustering algorithm using kd-trees. *Pattern Recognition Letters*, Vol. 28, No. 8, pp. 965–973.
29. **Tzortzis, G. & Likas, A. (2014).** The MinMax k-means clustering algorithm. *Pattern Recognition*, Vol. 47, No. 7, pp. 2505–2516.
30. **Xu, R. & Wunsch II, D. C. (2008).** *Clustering*. Wiley-IEEE Press.
31. **Zhai, Y., Ong, Y.-S., & Tsang, I. W. (2014).** The emerging ”big dimensionality”. *IEEE Computational Intelligence Magazine*, Vol. 9, No. 3, pp. 14–26.

Article received on 12/11/2016; accepted on 01/02/2017.
Corresponding author is Joaquín Pérez-Ortega.