Greedy-SG un procedimiento aplicado al problema de clustering

Carlos Peña y David Mauricio*
1cpenapena@uigv.edu.pe, 2dms-research@yahoo.com

Resumen

El problema de clustering consiste en ordenar objetos (personas, cosas, animales, plantas, variables, etc.) en grupos (conglomerados o clústeres) de forma que el grado de asociación o similitud entre los miembros del mismo grupo sea más fuerte y bastante diferente a los que se encuentran en otros grupos. Para resolver el problema tenemos el K-Means, el cual es un algoritmo sencillo y eficiente que procesa los patrones secuencialmente; sin embargo, está sesgado por el orden de presentación de los patrones, tiene alta dependencia de la elección de los centros iniciales y muestra la convergencia a óptimos locales.

El presente trabajo propone una solución denominada algoritmo Greedy-SG, conformado por dos procesos. El primero consta de tres fases: Inicialización, Construcción y Búsqueda Local, y el segundo, de Selección Grupal (SG) es una propuesta de evaluación de los grupos que pasarán a formar parte de la solución final.

Palabras clave:

Clustering, K-Means, H-Means, GRASP

1. Introducción

Los métodos de clustering o agrupamiento son una técnica de Machine Learning que hace énfasis en el aprendizaje no supervisado. En el aprendizaje no supervisado no existen patrones referentes a las salidas de los objetos que vamos introduciendo, sólo sus entradas tienen un conjunto finito de objetos que caracterizan a una población, se trata de encontrar un conjunto de parámetros que la representen óptimamente dada una función objetivo que depende exclusivamente de las relaciones de distancias entre los datos. En los métodos de agrupamiento no se suministran los datos etiquetados, el programa debe descubrir por sí mismo las clases naturales existentes. ([10] y [17]).

El Problema del Clustering

Dado el conjunto de datos X que consta de n puntos de referencias en R^D, los puntos de referencia son denominados como: objetos, instancias, casos, transacciones, etc.

$$X = \{x_1, x_2, \ldots, x_n\}, en\ que\ x_i \in R^D$$

*Universidad Inca Garcilaso de la Vega, FISCT

Los atributos de cada individuo $X_i(i = 1,\ldots,n)$ son criterios o datos asociados a cada objeto, los criterios son cualidades representadas en forma numérica o nominal, y que también son denominados como: característica, variable, dimensión, componente, campo, etc.,

$$x_i = \{x_{i1}, x_{i2}, \ldots, x_{id}\} \in R^D$$

Donde $i = 1 : N$

El problema de agrupamiento consiste en encontrar una partición de X en k subconjuntos disjuntos (clusters) al optimizar un criterio que mida su homogeneidad; así, a mayor homogeneidad se tendrá una mejor descripción de la estructura del cluster. ([3], [19], [20] y [26])

$$\{C_i\}_{i=1,\ldots,k}$$

Donde: $\bigcup C_{i,i=1,\ldots,k} = X$

La manera de abordar el problema del agrupamiento varía de acuerdo a los métodos aplicados. Desde el punto de vista de la asignación de los objetos a los clústeres, según [15], los métodos que obtienen una solución al problema del agrupamiento se dividen en dos tipos: hard clustering y soft clustering.

Hard Clustering

Se asume que los objetos deben ser asignados a uno y sólo uno de los clústeres, como consecuencia, los clústeres encontrados son particiones de X, por lo tanto tendremos que [26]:

$$C_i \cap C_j = \phi, \ \forall\ i, j = 1,\ldots,k \text{ tal que } i \neq j$$

Formulado de esta manera, el agrupamiento es un problema NP-Difícil ([5] y [9]). Los objetos de un clúster son parecidos o similares cuando las distancias entre ellos es mínima, para medir la similitud se formula como función objetivo **f**:

$$f\left(\bigcup_{i=1}^{k} C_i\right) = \sum_{j=1}^{K} \sum_{x_i \in C_j} d(x_i, \overline{x}_j) \qquad (1)$$

Donde $\overline{x}_j$, conocido como elemento representativo del clúster, es la media de los elementos del clúster C_j y corresponde al centro del clúster, se formula como sigue [26]:

$$\overline{x}_j = \frac{1}{|C_j|} \sum x_i \in C_j \qquad (2)$$

Soft Clustering

El soft clustering sigue un procedimiento similar como el hard clustering, con la diferencia, que un objeto soft se agrupa a más de un clúster, según un

esquema de probabilidad del cómputo de calidad del miembro. La naturaleza del objeto recopilado de un clúster es preservada igualmente cargando la información de otro clúster, el algoritmo de agrupamiento suave combina ambas informaciones [12]. El soft clustering asume que cada objeto tiene un valor de membresía con respecto a cada cluster $C_i,\ i=1,...,k$. Al contrario de las técnicas de hard clustering, los clústeres encontrados no son particiones del conjunto de patrones observados, son asociaciones de cada objeto con todos los clústeres, regulando esta asociación con una función de membresía [26].

Método Propuesto

La propuesta que se presenta es denominada algoritmo Greedy-SG, es una solución que presenta dos procesos, el primer proceso denominado GreedyKM es la adaptación del algoritmo KMeans, consta de tres fases: Inicialización, Construcción y Búsqueda Local, y un segundo proceso Selección Grupal (SG), donde se seleccionan y separan los grupos que conformaran la solución final.

El presente trabajo está organizado en seis capítulos: en el Capítulo 2 aborda los principales algoritmos de agrupamiento. En el Capítulo 3 se describe la solución Greedy-SG propuesta de esta tesis. En el Capítulo 4 se valida el método propuesto, se analiza y compara los resultados de la experimentación entre Greedy-SG con 7 algoritmos heurísticos y metaheurísticos, utilizando 5 diferentes colecciones de datos. Las conclusiones se describen en el Capítulo 6.

2. Métodos de Clustering

H-Means

El algoritmo H-Means ([11], [19], [20], [21] y [24]) es un método de agrupamiento heurístico con número de clases conocido (k). El algoritmo está basado en la minimización de la distancia interna (la suma de las distancias de los patrones asignados a un agrupamiento al centro de dicho agrupamiento). De hecho, este algoritmo minimiza la suma de las distancias al cuadrado de cada patrón al centro de su agrupamiento. La idea básica del algoritmo es obtener los k centros iniciales y formar clusters asociando todos los objetos de X a los centros más cercanos, después se recalculan los centros. Si esos centros no difieren de los centros anteriores, entonces el algoritmo termina; caso contrario, se repite el proceso de asociación con los nuevos centros hasta que no haya variación en los centros, o se cumpla algún otro criterio de parada como poco número de reasignaciones de los objetos [12]. La estructura básica de un algoritmo H-Means es la siguiente:

Algoritmo 2.1. H-Means

Entrada X={x1, … ,xn }, k
1. Inicio
2. Seleccionar centros iniciales $\{\overline{x_i} = Random(X)\}_{i=1,\ldots,k}$
3. Para cada $x_i \in X$ hacer
3.1. Asociar x i con el centro más cercano:
$C_j = C_j \bigcup \{x_i\}$
$Si\ d\left(x_i, \overline{x}_i\right) < d\left(x_i, \overline{x}_p\right) \forall_{j,p=1,\ldots,k\ y\ j \neq p}$
4. Fin de Parar

5. Calcular los centros $\{\overline{x}_i^* = Media(C_j)\}_{j=1,..,k}$
6. Si no hay más reasignaciones: $\overline{x}_i^* = \overline{x}_i, \forall i$ entonces
7. Para
8. Sino
9. Caso contrario, considerar $\overline{x}_i = \overline{x}_i^*$, e ir al paso 3.
10. Fin de Si
11. Fin

K-Means

El algoritmo K-Means ([11], [19], [20], [21] y [24]) es probablemente el algoritmo de agrupamiento más conocido. Su estructura básica es la siguiente:

Algoritmo 2.2. K-Means

Entrada $\mathbf{X=\{x_1, \ldots, x_n\}, k}$
1. Inicio
2. Seleccionar centros iniciales $\{\overline{x_i} = Random(X)\}_{i=1,\ldots,k}$
3. Para cada $x_i \in X$ hacer
3.1. Asociar x i con el centro más cercano:
$C_j = C_j \bigcup \{x_i\}$
$Si\ d\left(x_i, \overline{x}_i\right) < d\left(x_i, \overline{x}_p\right) \forall_{j,p=1,\ldots,k\ y\ j \neq p}$
3.2. Calcular los centros $\{\overline{x}_i^* = Media(C_j)\}_{j=1,..,k}$
4. Fin de Para
6. Si no hay más reasignaciones: $\overline{x}_i^* = \overline{x}_i, \forall i$ entonces
7. Parar
8. Sino
9. Caso contrario, considerar $\overline{x}_i = \overline{x}_i^*$, e ir al paso 3.
10. Fin de Si
11. Fin

El algoritmo es sencillo y eficiente, procesa los patrones secuencialmente (por lo que requiere un almacenamiento mínimo). Sin embargo, está sesgado por el orden de presentación de los patrones (los primeros patrones determinan la configuración inicial de los agrupamientos) y su comportamiento depende enormemente del parámetro k. Dentro de los principales inconvenientes del algoritmo

K-Means está su alta dependencia de la elección de los centros iniciales y su convergencia a óptimos locales. Esta deficiencia es en realidad una debilidad de los algoritmos golosos que son rápidos encontrando soluciones, pero quedan atrapados en óptimos locales (2) ([22] y [23]).

GRASP

GRASP ([7] y [26]) es una técnica de los años 80 propuesta por Feo y Resende, que tiene como objetivo resolver problemas difíciles en el campo de la optimización combinatoria. Esta técnica dirige la mayor parte de su esfuerzo a construir soluciones de alta calidad que son posteriormente procesadas para obtener otras aún mejores.

El algoritmo GRASP es un procedimiento de búsqueda voraz, aleatoria y adaptativa, es una meta heurística para encontrar soluciones aproximadas de problemas de optimización combinatoria, mediante un proceso iterativo. En cada iteración se realizan dos fases de operaciones: construcción y búsqueda local. En la fase de construcción se genera un conjunto solución S de una instancia E de un problema combinatorio, y en la fase de búsqueda local se determina una posible mejor solución a S; finalmente, se elige la solución mejor entre la solución de la iteración anterior y la actual. La mejor solución será indicada por una función objetivo f. Cada iteración es realizada un número máximo de veces (MAX_ITER) ([7] y [26]).

En [6] se describe un pseudo-código genérico GRASP para la fase de construcción del agrupamiento. En este proceso cuando los objetos k se han adoptado, la búsqueda local (K-Means, en este caso) se aplica tomando la agrupación obtenida en la inicialización. Luego, compara la solución óptima local obtenida con la mejor solución encontrada, y se toma la mejor. Este proceso continúa hasta que todas las iteraciones se han realizado. El uso de K-Means ofrece un método eficiente y de bajo coste computacional para obtener soluciones relativamente buenas, pero converge a un mínimo. La fase de construcción del GRASP corrige este problema al ampliar la búsqueda de la exploración espacial.

Algoritmo 2.3. GRASP

1. Select the most centrally located instance as the firstseed.
2. FOR every non selected instance e_i DO
2.1. FOR every non selected instance e_j DO
Calculate $C_{ji} = \max\left(D_j - d_{ji}, 0\right)$

$$\text{where } d_{ji} = \| e_i - e_j \| \quad D_j = \min_s d_{sj}$$

being s one of the selected seeds

2.2. Calculate the gain of selecting e_i by $\sum_j C_{ji}$
3. MakeRCL(RCL) by selecting the l instantes e_i which maximizes $\sum_j C_{ji}$
4. SelectElementAtRandom(RCL)
5. If there are k selected seeds THEN stop ELSE go to step 2
6. For having a clustering assign each nonselected
instance to the cluster represented by the nearest seed.

3. Algoritmo Greedy-SG

El presente modelo, denominado Greedy-SG, es una metaheurística de tipo hard clustering compuesto por el algoritmo GreedyKM y un método propuesto denominado Selección Grupal (SG), los cuales la convierten en una excelente solución al problema del clustering.

Parámetros y funciones

Los parámetros son los siguientes:

- SGMax = Número máximo de iteraciones para seleccionar grupos
- IterMax = Número máximo de iteraciones del algoritmo para generar soluciones
- k = Número de grupos que se desea obtener.
- X = Conjunto de datos u objetos
- xi = Dato u objeto.
- La función f es igual a la Función objetivo descrita en (1).

Greedy-SG

El algoritmo Greedy-SG ha sido desarrollado para la minimización de la función objetivo (1). Esta solución tiene dos procesos, el primer proceso es un GreedyKM que se ejecuta en repetidas veces (IterMax), y un segundo proceso SG donde los grupos se seleccionan y separan en varias oportunidades (SGMax).

El primer proceso GreedyKM consta de tres fases: Inicialización, Construcción y Búsqueda Local, cuyo objetivo es encontrar las mejores soluciones para la minimización de la función objetivo (1). El segundo proceso. SeleccionGrupal (SG), procede a la selección comparativa de grupos de las dos mejores soluciones del primer proceso, y a separar los grupos iguales del conjunto de objetos X.

Presentamos la estructura general del algoritmo Greedy-SG, después se describirá en detalle cada una de los procesos y fases mencionadas. Se consideran como datos de entrada el conjunto de objetos X, un número k de clústeres a generar, el máximo número de iteraciones (IterMax) y máximo número de selecciones (SGMax). Debemos esperar como resultado el conjunto C^5.

Algoritmo 3.1. Greedy-SG
Entrada : $X, k, IterMax, SGMax$ 1. Inicio 2. $C^i := \{\}_{1 \le i \le 5}$ 3. *Para* $a = 1$ *hasta* $SGMax$ 3.1. $GreedyKM(X, k, IterMax, C^1, C^4)$ 3.2. $C^5 := SeleccionGrupal(C^1, C^4)$ 4. Fin de Para 5. Re*tornar* C^5 6. Fin

Primer proceso: GreedyKM

El primer proceso es un algoritmo GreedyKM con tres fases. En la primera fase (Inicialización) se obtiene los k centros iniciales y se definen los grupos C_j en torno de sus centros iniciales. La segunda fase (Construcción), se redistribuye los objetos del clúster menos poblado y el más disperso considerando la distancia de los centros. La tercera fase (BusquedaLocal), asigna los objetos al grupo más cercano. La fase segunda y tercera se ejecuta varias veces hasta que no sea posible mejorar la solución, es decir que la función objetivo de C^2 sea menor que la función objetivo de C^3, cumpliéndose así la minimización de la función objetivo (1). Por último, se determinan las dos mejores soluciones del primer proceso; para tal fin se evalúa si función objetivo de C^2 es menor (mejor solución) que la menor de las funciones objetivos de las iteraciones anteriores f^*, de ser afirmativo C^4 asume los valores de C^1 (segunda mejor solución), C^1 los valores de C^2 (primera mejor solución) y f^* se iguala con $f(C^2)$.

A continuación presentamos la estructura del algoritmo GreedyKM. Se consideran como datos de entrada el conjunto de objetos X, un número k de clústeres a generar y el máximo número de iteraciones IterMax. Debemos esperar como resultado el conjunto C^1 y C^4

Algoritmo 3.2. GreedyKM

Entrada : $X, k, IterMax, C^1, C^4$

1. Inicio
2. $f^* := \infty, k' := k$
3. *Para* $b = 1$ *hasta IterMax hacer*
   - 3.1. $C^2 := Inicializacion(X, k'), C^3 := C^2$
   - 3.2. *Mientras* $f(C^2) \geq f(C^3)$
     - 3.2.1. $C^2 := C^3$
     - 3.2.2. $C^2 := Construccion(X, k', C^2)$
     - 3.2.3. $C^3 := BusquedaLocal(X, k', C^2)$
   - 3.3. Fin Mientras
   - 3.4. *Si* $f(C^2) < f^*$ *entonces*
     - 3.4.1. $C^4 := C^1$
     - 3.4.2. $C^1 := C^2$
     - 3.4.3. $f^* := f(C^2)$
   - 3.5. Fin de Si
4. Fin de Para
5. Re*tornar* C^1, C^4
6. Fin

Fase Inicialización

En esta fase sobre la base del K-Means se construye una solución de la siguiente manera: primero se seleccionan aleatoriamente k objetos que serán los centros iníciales de $C_{i=1,..,k}$, segundo se asignan cada objeto $x \in X$

al C_j que presenta menos distancia a su centroide. En cada asignación del objeto x al clúster C_j se calcula el centro de C_j.

Se consideran como datos de entrada el conjunto de objetos X y un número k de clústeres a generar. Debemos esperar como resultado el conjunto C^2.

Algoritmo 3.3. Inicialización
Entrada: **X, k`**
1. Inicio
2. Seleccionar k` diferentes centro iniciales
   $\{\overline{x}_i = Random(X)\}_{i=1,..,k}$
3. $C_i^2 := \{\overline{x}_i\}$
4. Para cada x ∈ X hacer
   4.1. Asignar x a C_j^2 cuando
   4.2. *Calcular centro* $\{\overline{x}_i = Media(C_j^2)\}$
5. Fin Para
6. Re*tornar* C^2
7. Fin

Fase Construcción

Al igual que en la fase de Reagrupación del algoritmo GraspKM, la fase de Construcción requiere la identificación de los grupos C_l^2 de menor número de elementos y C_h^2 de mayor dispersión, de la siguiente forma [65]:

C_l^2, para el grupo con menor número de elementos se hace el siguiente cálculo:

$$l := ArgMin\{|\, C_i^2 \,|\}_{i=1,\ldots,K}$$

C_h^2, para el grupo con mayor dispersión se calcula de la siguiente forma:

$$Ej = \frac{\sum_{x \in Cj} d(x, \overline{x_j})}{|\, C_j^2 \,|}$$

Antes de identificar el clúster más disperso, se describirán dos casos extremos: el primero de un clúster compacto y el segundo de un clúster disperso. El primer caso es un grupo con error promedio bajo y que tiene una buena cantidad de objetos; esta situación nos da la idea de que el grupo está bastante compacto. Por el contrario, si tenemos un grupo con error promedio alto y con gran cantidad de elementos, entonces diremos que el grupo está disperso. Es decir, cuanto mayor sea el valor de E_j, mayor será la dispersión del grupo, por el contrario, cuanto menor sea la dispersión, mayor será su compactación. De manera aleatoria se selecciona $\overline{x_r} := Random(C_h^2)$

Con el propósito de mejorar los resultados del agrupamiento se propone considerando las distancias euclidianas reasignar los elementos del grupo más disperso con el grupo menos denso. Para ello se calcula m que es la distancia euclidiana entre $\overline{x_r}$ y $\overline{x_h}$, el valor de m nos va a permitir separar

los objetos x_i del clúster C_h^2 y asignarlos al clúster $C_{l,}^2$ para ello primero es necesario calcular la distancia euclidiana entre x_i y x_h, cuando la distancia calculada es mayor que m se le asigna a C_l^2 y se le separa de C_h^2.

Se consideran como datos de entrada el conjunto de objetos X, un número k de clústeres a generar y el conjunto C^2. El resultado es la reasignación de los objetos del clúster C_h^2 al clúster C_l^2.

Algoritmo 3.4. Construcción

Entrada : X, K', C^2

1. Inicio
2. $l := ArgMin\{|C_j^2|\}_{j=1,\ldots,K}$
3. $h := ArgMax\left\{E_j\right\}_{j=1,\ldots,K, j\neq l}$
4. $\overline{x_r} := Random(C_h^2)$
5. $m := d(\overline{x_r}, \overline{x_h})$
6. *Para cada* $x_i \in C_h^2$
   - 6.1. *Si* $d(x_i, \overline{x_h}) > m$
     - 6.1.1. $C_l^2 := C_l^2 \cup \{x_i\}$
     - 6.1.2. $C_h^2 = C_h^2 - \{x_i\}$
   - 6.2. Fin de Si
7. Fin de Para
8. Fin

Fase Búsqueda Local

La fase búsqueda local es una adaptación del K-Means que reasigna cada uno de los objeto $x \in X$ al centro $\overline{x}_j$ de C_j con menor distancia, en cada asignación del objeto al clúster C_j se vuelve a calcular sólo el centro C_j. Una vez finalizada las asignaciones se calcula los centros de los k clúster, haciendo uso de la expresión (2) $C := \{C_j\}_{j=1,\ldots,k}$. Se consideran como datos de entrada el conjunto de objetos X, un número k de clústeres a generar y el conjunto C^2. Debemos esperar como resultado un nuevo conjunto C^2

Algoritmo 3.5. Búsqueda Local

Entrada : X, K', C^2

1. Inicio
2. *Para cada* $x \in X$ *hacer*
   - 2.1. Re*asignar* x *a* C_l^2 *cuando* $j := ArgMin\{d(x, \overline{x_i})\}_{i=1,\ldots,k}$
   - 2.2. *Calcular centro* $\{\overline{x}_l = Media(C_l^2)\}$
3. Fin Para
4. Re*tornar* C^2
5. Fin

Segundo proceso: Selección Grupal (SG)

Este proceso consiste en la construcción de un agrupamiento de k grupos a partir de un k-1 grupo, es decir los grupos son seleccionados y separados para conformar una nueva solución. Es por ello que al finalizar las iteraciones señaladas en el primer proceso, se procede a seleccionar los grupos de las dos mejores soluciones obtenidas, esto es C^1 y C^4, los objetos x de los clúster seleccionados se separan de X y se restan de k. La selección se realiza tomando en cuenta lo siguiente:

Se calcula la dispersión de C^1,

$$E_j^1 = \frac{\sum_{x \in Cj} d(x, \overline{x_j})}{|C_j^1|} \quad donde\ j = 1,\ldots,|C^1|$$

Se calcula el límite máximo de la dispersión (ds), para lo cual primero se determina el promedio de la dispersión y luego se le suma la desviación estándar de la dispersión de C^1 multiplicada por 1.5.

Se verifica que la dispersión de cada clúster es menor a ds, en caso contrario se da por finalizado el segundo proceso.

Finalmente, primero se ordenan los clúster C^1 y C^4 por la suma de la distancia euclidiana de sus elementos, segundo se comparan, según orden, la suma de la distancia euclidiana de sus objetos y el número de objetos entre los clúster C^1 y C^4.

Los seleccionados son separados de X y los clúster se restan de k.

Algoritmo 3.6. SeleccionGrupal
Entrada : C^1, C^4
1. Inicio
2. Calcular el límite de dispersión (ld) de C^1
   2.1 $E_j^1 = \frac{\sum_{x \in Cj} d(x, \overline{x_j})}{|C_j^1|}$ *donde* $j = 1,\ldots,|C^1|$
   2.2. $d := \frac{\sum E_j^1}{|C^1|} + \left[DesStd\left(E_j^1\right)\right] * 1.5)$
3. $sw := 1$
4. *Para cada* $j = 1,\ldots,|C^1|$
   4.1. *Si* $E_j^1 < ld$
      4.1.1. $sw := 0$
   4.2. Fin de Si
5. Fin de Para
6. *Si* $sw := 1$

7 . *Para cada* $j = 1,\ldots,|C^1|$
7.1. *Si* $C_j^1 = C_i^2$ *para* $i = 1,\ldots,|C^1|$
7.1.1.$\Rightarrow C^5 := C^5 \cup C_j^1$
7.1.2.$X := X - C_j^1$
7.2. Fin de Si
8. Fin de Para
9. Fin de Si
10. Retornar C^5
11. Fin

4. Experimentación del algoritmo SN

Colección de Datos

Las colecciones de datos que se usan para la evaluación del algoritmo son: Iris, Glass y Prima [25], Crude Oil [13] y Vowel [21]. Las colecciones de datos son:

- Iris. Consiste en mediciones de 4 atributos de una muestra de 150 plantas de lirio. Los atributos observados son: el ancho del pétalo, el largo del pétalo, el ancho del sépalo y el largo del sépalo, los cuales están expresados en milímetros. La muestra es hecha sobre tres especies de lirio: setosa, versicolor y vergínica, con 50 muestras por cada especie. El valor de k para este conjunto de datos es de 3 y 4.
- Glass. Consiste en 214 instancias de diferentes tipos de vidrio. El número de atributos es 9, correspondientes al índice de refracción y otras mediciones de los elementos que conforman el vidrio: sodio, magnesio, aluminio, silicio, potasio, calcio, bario y fierro. El valor de k para este conjunto de datos es de 2, 7 y 10.
- Crude Oil. Consiste en 56 muestras de petróleo crudo tomado de tres zonas diferentes de rocas areniscas. Los datos presentan 6 atributos correspondientes al tipo de roca y mediciones de los componentes: vanadio, fierro, berilio, hidrocarbonos saturados e hidrocarbonos aromáticos. El valor de k para este conjunto de datos es 3.
- Vowel. Consiste en 871 muestras de seis clases de vocales del dialecto hindú Telugu. Los atributos corresponden a la medición de tres frecuencias y están expresados en valores enteros. El valor de k para este conjunto de datos es de 6.
- Pima. Consiste en 768 vectores en dos dimensiones formando 8 grupos de manera natural. El valor de k para este conjunto de datos es de 4 y 5.

Comparación de resultados

Las comparaciones se efectúan con respecto al mejor valor encontrado para la minimización de la función objetivo (1). Como sabemos cada método presenta sus características propias, el hecho de establecer el número de ejecuciones o el máximo de iteraciones similares es con la intención de es-

tablecer condiciones parecidas para la ejecución de los métodos; en todo caso, la comparación se realizará principalmente en los valores obtenidos para la función objetivo.

La primera comparación se realizada entre Greedy-SG y el algoritmo GraspKM propuesto en [26]. Los resultados obtenidos por cada algoritmo en las colecciones de datos IRIS, GLASS, CRUDE OIL y VOWEL, para 50 ejecuciones con un máximo de 100 iteraciones y para el caso de GraspKM con un valor de $\alpha = 1$. Los resultados obtenidos se muestran en el cuadro 4.1.

Cuadro 4.1 Comparación entre GraspKM y Greedy-SG con colección de datos IRIS, GLASS, CRUDE OIL y VOWEL

Base de datos	GraspKM		
	Mejor optimo	Peor optimo	Promedio
IRIS - K = 3	97.22213	97.257195	97.226074
GLASS-K = 7	200.58238	201.85007	201.02412
CRUDE OIL - K = 3	278.96515	278.96515	278.96515
VOWEL - K = 6	149350.52	149400.92	149379.84
	GRASP-SG		
	Mejor optimo	Peor optimo	Promedio
IRIS - K = 3	96.9594123	97.1382392	97.0426238
GLASS - K = 7	200.35517	200.95812	200.65370
CRUDE OIL - K = 3	257.134609	278.775064	261.799907
VOWEL - K = 6	149265.88	149355.388	149338.775

Como se puede observar en el cuadro 4.1, el Greedy-SG obtiene mejores resultados en todas las comparaciones.

La segunda comparación se realizada entre Greedy-SG y otras 7 heurísticas de la literatura de agrupamiento: el algoritmo Random usado en ([4] y [6]), el algoritmo Forgy ([1] y [8]), el algoritmo MacQueen propuesto por [16], el algoritmo Kaufman propuesto por [14], el algoritmo GRASP propuesto por [6], K-Means usado en [26] y GraspKHM propuesto en [26].

En [6] se muestran los resultados obtenidos por cada algoritmo en las colecciones de datos IRIS, GLASS y PIMA, para 10 ejecuciones con un máximo de 16 iteraciones. Los resultados obtenidos se muestran en el cuadro 4.2, 4.3 y 4.4.

Cuadro 4.2 Comparación entre Random, Forgy, Macqueen, Kaufman, GRASP-KM y Greedy-SG con colección de datos IRIS

Algoritmos	K = 3	
	Promedio	Mejor optimo
Random	97.339	97.326
Forgy	97.326	97.326
MacQueen	97.326	97.326
Kaufman	97.326	97.326
GRASP	97.326	97.326
GRASP-SG	97.0671269	96.9615154
	K = 4	
	Promedio	Mejor optimo
Random	83.769	83.729
Forgy	83.749	83.729
MacQueen	83.750	83.729
Kaufman	83.786	83.786
GRASP	83.729	83.729
GRASP-SG	83.4409745	83.3060956

Cuadro 4.3 Comparación entre Random, Forgy, Macqueen, Kaufman, K-Means, GRASP, GraspKM y Greedy-SG con colección de datos GLASS

Algoritmos	K = 2	
	Promedio	Mejor optimo
Random	323.323	323.323
Forgy	323.323	323.323
MacQueen	323.323	323.323
Kaufman	323.323	323.323
K-Means		
GRASP	323.323	323.323
GraspKM		
GRASP-SG	316.581471	316.186762
	K = 7	
	Promedio	Mejor optimo
Random	206.632	205.307
Forgy	206.493	205.889
MacQueen	206.635	205.889
Kaufman	211.658	211.158
K-Means	212.591	203.865
GRASP	205.239	204.992
GraspKM	202.233	201.126
GRASP-SG	201.925218	200.701765
	K = 10	
	Promedio	Mejor optimo
Random	179.042	175.945
Forgy	176.71	176.044
MacQueen	177.144	176.044
Kaufman	188.691	188.691
K-Means		
GRASP	176.058	175.945
GraspKM		
GRASP-SG	173.77421	172.785219

Cuadro 4.4 Comparación entre Random, Forgy, Macqueen, Kaufman, GRASP-KMeans, Gras pKM y Greedy-SG con colección de datos PIMA

Algoritmos	K = 2	
	Promedio	Mejor optimo
Random	52072.211	52072.203
Forgy	52072.211	52072.203
MacQueen	52072.211	52072.203
Kaufman	52072.233	52072.219
GRASP	52072.233	52072.219
GRASP-SG	47730.3057	47668.765
	K = 6	
	Promedio	Mejor optimo
Random	29415.066	29403.762
Forgy	29426.369	29403.762
MacQueen	29428.447	29403.762
Kaufman	30729.2	30729.195
GRASP	29403.763	29403.762
GRASP-SG	29349.2711	29307.5148
	K = 12	
	Promedio	Mejor optimo
Random	24287.956	24120.801
Forgy	24136.125	24117.748
MacQueen	24137.881	24117.748
Kaufman	24340.811	24340.811
GRASP	24127.77	24117.748
GRASP-SG	22744.4155	22685.0147

Como se puede observar en los cuadros 4.2, 4.3 y 4.4, el Greedy-SG obtiene mejores resultados en todas las comparaciones.

La tercera y última comparación realizada es entre Greedy-SG y otras tres metaheurísticas: el Algoritmo Genético KGA-Clustering propuesto por [2], el algoritmo K-Means usado en [26] y el algoritmo GraspKM propuesto por [26]. Las colecciones de datos de Iris, Vowel y Crude Oil, fueron tomadas por [2] donde presenta los resultados obtenidos en 50 ejecuciones con máximo de 1 000 generaciones, y por [26] para presentar los resultados de 50 ejecuciones con un valor de $a = 1$ y MAX_ITER de 1,000. En el caso del GRASP-SG la comparación se hace en base a los resultados obtenidos en el cuadro 4.1. Los resultados son resumidos en el cuadro 4.5.

Algoritmos	IRIS - K = 3		
	Mejor optimo	Peor optimo	Promedio
K-Means	97.32594	128.40422	104.26579
KGA-Clustering	97.100777	97.100777	97.100777
GraspKM	97.22213	97.22213	97.22213
GRASP-SG	96.9594123	97.1382392	97.0426238
Algoritmos	CRUDE OIL - K = 3		
	Mejor optimo	Peor optimo	Promedio
K-Means	279.270997	359.761992	284.24806
KGA-Clustering	278.96515	278.96515	278.96515
GraspKM	278.96515	278.96515	278.96515
GRASP-SG	257.134609	278.775064	261.799907
Algoritmos	VOWEL - K = 6		
	Mejor optimo	Peor optimo	Promedio
K-Means	149399.49	168764.32	154150.77
KGA-Clustering	149356.01	149378.03	149368.45
GraspKM	149342.64	149374.36	149360.25
GRASP-SG	149265.88	149355.388	149338.775

Cuadro 4.5 Comparación entre K-Means, KGA-Clustering, GraspKM y Greedy-SG con colección de datos IRIS, CRUDE OIL y VOWEL

Como se puede observar en el cuadro 4.5, el Greedy-SG obtiene mejores resultados con todas las comparaciones.

5. Conclusiones

En todos los casos, el algoritmo Greedy-SG arroja los mejores resultados, por lo tanto se puede concluir que el algoritmo Greedy-SG logra un agrupamiento eficiente.

Como se demuestra en las pruebas realizadas, el segundo proceso del algoritmo Greedy-SG es una solución que selecciona y separa los grupos del clustering que conducen a la convergencia a óptimos locales, lo cual le permite al algoritmo seguir en la búsqueda de una mejor optimización y por consecuencia de lograr mejores resultados.

El segundo proceso del Algoritmo Greedy-SG, puede ser mejorado trabajando en los siguientes puntos:

Determinar el límite de la dispersión de los clúster, variando los valores de la desviación estándar hasta lograr un óptimo.

Utilizar otros métodos estadísticos o matemáticos para evaluar la densidad del clúster y así mejorar los resultados.

En este proceso se ha divido el agrupamiento en dos subgrupos, a uno se le separa y al otro se vuelve a buscar su optimización. Sería conveniente analizar la posibilidad de optimizar los resultados del primer subgrupo.

Dividir el agrupamiento en tres o más subgrupos, y establecer los criterios de división que permitan lograr mejores resultados.

En este proceso se evalúa los resultados a nivel grupal, pero sería necesario considerar una evaluación a nivel de objetos.

6. Agradecimientos

A la Facultad de Ingeniería de Sistemas, Computación y Telecomunicaciones de la Universidad Inca Garcilaso de la Vega por el apoyo brindado para la publicación de esta investigación.

7. Referencias

1. Anderberg, M.: Cluster analysis for applications. Academic Press, 1973, New York, NY.

2. Bandyopadhyay, S. y Maulik, U. An evolutionary technique based on K-Means algorithm for optimal clustering in Rn. Information Sciences. Vol. 146 (1-4), 2002, pp. 221–237.

3. Berkhin, P. Survey of clustering data mining techniques, Technical Report, Accrue Software, 2002

4. Bradley, P. y Fayyad, U.: Refining initial points for K-Means Clustering. Proccedings of the Fifteenth International Conference on Machine Learning 1998, 91-99. Morgan Kaufmann Publisher, Inc. San Francisco, CA.

5. Brucker, P. On the Complexity of Clustering Problems. Lecture Notes in Economics and Mathematical Systems. Vol. 157, 1978, pp. 45–54.

6. Cano, J.; Cordón, O.; Herrera, F. y Sánchez, L.: Greedy Randomized Adaptive Search Procedure Applied to the Clustering Problem as an Initialization Process Using K-Means as a Local Search Procedure. International Journal of

Intelligent and Fuzzy Systems, 2002, 12, pp. 235-242.

7. Feo, T. y Resende, M. Greedy Randomized Adaptative Search Procedure. Journal of Global Optimization Vol. 6, 1995, pp. 109-133.

8. Forgy, E.: Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. Biometrics 1965, 21, 728

9. Garey, M. y Johnson, D. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

10. Garre, M., Cuadrado, J., Sicilia, M., Rodríguez, D. y Rejas, R.. Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software. Revista Española de Innovación, Calidad e Ingeniería del Software, Vol.3, No. 1, 2007

11. Hansen, P. and N. Mladenovic, N. J-Means: a new local search heuristic for minimum sum-of-squares clustering. Pattern Recognition 34 2 (2001), pp. 405–413

12. Iosif, E. y Potamianos, A. A Soft-Clustering Algorithm for Automatic Induction of Semantic Classes. INTERSPEECH 2007 International Conference, Antwerp, Belgium, 2007.

13. Johnson, R. y Wichern, D.: Applied Multivariate Statistical Analysis. Prentice Hall, Englewood Cliffs, NJ. 1982.

14. Kaufman, L. y Rousseeuw, P.J. Finding Groups in Data. An Introduction to Cluster Analysis. Wiley (1990).

15. Kearns, M. Mansour, Y. y Ng, A. An Information-Theoretic Analysis of Hard and Soft Assignment Methods for Clustering. In Proceedings of 13th Annual Conf. on Uncertainty in Artificial Intelligence (UA197) 1997.

16. McQueen, J. Some methods for classification and analysis of multivariate observations. In Preccedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1967, 281–297.

17. Orriols-Puig, A., Casillas, J. y Martínez-López, F. Modelado causal en Marketing mediante aprendizaje no supervisado de reglas de asociación difusas. Proceedings of the XIV Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF'08), Mieres-Langreo, Spain, pp. 529-536, 2008

18. Pacheco J. Análisis de nuevos métodos de clasificación. Un ejemplo ilustrativo de su uso en la agrupación de los municipios de Castilla y León. Dpto. Economía Aplicada. Universidad de Burgos Facultad Ciencias Económicas y Empresariales Plaza Infanta Elena s/n. BURGOS 09001

19. Pacheco, J. A Scatter Search Approach for the Minimum Sum-of-Squares Clustering Problem Computers & Operations Research. (2005) 32, pp. 1325-1335.

20. Pacheco, J. y Valencia, O. Design of Hybrids for the Minimum Sum-of-Squares Clustering Problem. Computational Statistics and Data Analysis. (2003) 43,2:235-248.

21. Pal, S. y Dutta, D. Fuzzy sets and decision making approaches in vowel and speaker recognition. IEEE Transactions on Systems, Man, and Cybernetics, 1977, 7, pp. 625–629.

22. Peña, J.M. Lozano, J.A. y Larrañaga P. An Empirical Comparison of Four Initialization Methods for the K-Means Algorithm. Pattern Recognition Letters 20 (1999) 1027-1040.

23. Selim, S.Z. y Ismail, M.A., K-Means-Type Algorithm: Generalized Convergence Theorem and Characterization of Local Optimality. IEEE Trans. Pattern Anal. Machine Intelligence 6 (1984) 81-87.

24. Stoffel, K. y Belkoniene, A. Parallel K/H-means clustering for Large Data. Sets, Proceedings of the European Conference on Parallel Processing EroPar'99, 1999

25. UCI Machine Learning Repository http://archive.ics.uci.edu/ml/

26. Vicente, E. GraspKM en la recuperación eficiente de la estructura de software. Msc. Tesis, Universidad Nacional Mayor de San Marcos, Perú, 2007.