Selección de Centroides para Algoritmos de Clustering a través de Técnicas Metaheurísticas

Andrea Villagra, Daniel Pandolfi
Universidad Nacional de la Patagonia Austral
Ruta 3 Acceso Norte s/n
(9011) Caleta Olivia - Santa Cruz - Argentina
{avillagra,dpandolfi}@uaco.unpa.edu.ar

and

Guillermo Leguizamón
Universidad Nacional de San Luis,
Ejército de los Andes 950, (5700) San Luis, Argentina
legui@unsl.edu.ar

Abstract

The clustering algorithms like c-means are sensitive to the initialization values of the cluster centers and can be trapped by local extrema. In these terms, the use of estimated approaches to obtain the most appropriate cluster centers can be of great utility as a complementary tool during certain phases of the process of data mining; particulary, in some specific task of data mining, e.g., clustering.
In this way, Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) are two population metaheuristic approaches that could be considered as optimization. In this work the use of these two metaheuristic approaches is analyzed to optimize the initialization of the cluster centers values in the functions applied in the c-means algorithms. The respective results are compared using several datasets artificially generated.

keywords: Clustering, c-means, genetic algorithms, particle swarm optimization.

Resumen

Los algoritmos de *clustering* de tipo *c-means* son sensibles a los valores de inicialización de los centroides y pueden quedar atrapados en extremos locales. Planteado en estos términos, el uso de enfoques aproximados para obtener los centroides más adecuados puede ser de gran utilidad como herramienta complementaria durante ciertas fases del proceso de minería de datos, y en particular dentro de las tareas típicas de minería de datos, entre ellas la de *clustering* o agrupamiento. En esta dirección, los *Algoritmos Genéticos* (AGs) y la *Optimización Basada en Cúmulo de Partículas* (PSO)[1] son dos técnicas metaheurísticas poblacionales que podrían utilizarse en este ámbito, más aún cuando los problemas pueden ser planteados como de optimización.

En este trabajo se analiza el uso estas dos técnicas metaheurísticas para optimizar la inicialización de los valores de centroides en las funciones aplicadas en los algoritmos de *clustering* tipo *c-means*. Los respectivos resultados son comparados usando varios conjuntos de datos generados artificialmente.

Palabras claves: *Clustering*, *c-means*, algoritmos genéticos, optimización basada en cúmulo de partículas.

[1]Corresponde a las siglas en inglés para "Particle Swarm Optimization" las que son usadas convencionalmente.

1. INTRODUCCIÓN

La minería de datos constituye el núcleo del análisis inteligente de los datos y ha recibido un gran impulso en los últimos tiempos motivado por distintas causas: a) el desarrollo de algoritmos eficientes y robustos para el procesamiento de grandes volúmenes de datos, b) un poder computacional más barato que permite utilizar métodos computacionalmente intensivos, y c) las ventajas comerciales y científicas que han brindado este tipo de técnicas en las más diversas áreas. Entre las áreas donde han sido utilizadas exitosamente las técnicas de minería de datos podemos mencionar distintas aplicaciones financieras y bancarias, análisis de mercado, seguros y salud privada, educación, procesos industriales, medicina, biología, bioingeniería, telecomunicaciones, Internet, turismo, deportes, etc.

Es importante diferenciar en la minería de datos, el tipo de tareas que se suelen abordar y las técnicas utilizadas en cada caso. Como ejemplos de tareas generales se pueden mencionar el aprendizaje de conceptos, clasificación, categorización, regresión, agrupamiento (o *clustering*), correlaciones y análisis de asociación. Estas tareas pueden ser abordadas mediante distintos métodos o técnicas que suelen adaptarse mejor de acuerdo a la tarea sobre la cual se trabajará. Entre las técnicas más conocidas se puede mencionar el aprendizaje de reglas de clasificación, reglas de asociación, reglas relacionales, reglas difusas, árboles de decisión (y regresión), ecuaciones de regresión, redes neuronales, metaheurísticas, etc.

Un caso especial es el de las metaheurísticas, las que pueden jugar un papel bien definido en las distintas etapas del proceso del análisis inteligente de los datos en general y la minería de datos en particular [1] y [6]. El uso de metaheurísticas en estos casos puede realizarse en forma aislada o en combinación con otros algoritmos incluyendo el aprendizaje de redes neuronales y árboles de decisión. Por lo tanto, las metaheurísticas pueden ser usadas para mejorar la robustez y precisión de las técnicas más tradicionales usadas en extracción de características, selección de características, clasificación y agrupamiento (*clustering*). Éste último, un caso particular del aprendizaje no-supervisado, ha mostrado ser muy útil en el análisis exploratorio de datos, segmentación de imágenes y con algunas clases de conocimiento agregado, puede ser usado también para clasificación.

En este trabajo se presentan dos enfoques, uno guiado genéticamente y otro a través de cúmulo de partículas para optimizar un problema de *clustering*. Estos enfoques pueden ser directamente aplicados a cualquier modelo de *clustering* que pueda representarse como una función dependiente de un conjunto de centroides (centros de *cluster* o puntos prototipos). En un futuro pueden generalizarse para modelos que requieran parámetros en lugar de centroides. Ambos enfoques utilizan una represententación binaria de los centroides y la función (J_1) se usa como posible función objetivo dentro del enfoque *hard c-means* (denotado como HCM) tal como es usado en [9] y [12]. Una característica negativa de estos algoritmos de *clustering*, es que pueden quedar atrapados por extremos locales en el proceso de optimizar el criterio de *clustering*. Siendo además, muy sensibles a la inicialización de los centroides. En este trabajo se ha optado por dos metaheurísticas como herramientas alternativas para el problema planteado anteriormente. La primera, los Algoritmos Genéticos (AGs) introducidos por Holland en [5], inspirados en la capacidad de la naturaleza para evolucionar seres para adaptarlos a los cambios de su entorno. Los AG se han utilizado mucho en optimización y particularmente en problemas de *clustering* con resultados muy satisfactorios [11]. Por otro lado, se plantea el uso de una metheurística desarrollada más recientemente; los algoritmos Basados en Cúmulos de Partículas o *Particle Swarm Optimization* (PSO) [8], técnicas metaheurísticas inspiradas en el comportamiento social del vuelo de las bandadas, movimiento de los cardúmenes, entre otros sistemas sociales altamente cohesionados y que permite, simulando este modelo de comportamiento, obtener métodos eficientes para resolver problemas de optimización. En consecuencia, los enfoques presentados y analizados aquí son usados como una alternativa para la inicialización de los centroides con el fin de

aminorar la sensibilidad a la inicialización de los algoritmos de *clustering* de tipo *c-mean*.

El resto del trabajo está organizado como sigue. La siguiente sección ofrece una descripción general del denominado *hard c-means* clustering o HCM y la descripción del AG y PSO para su resolución. La sección 3 muestra los resultados y el análisis de los experimentos realizados; finalmente, son presentadas las conclusiones y posibles direcciones para estudios futuros.

2. CLUSTERING CON HCM

En esta sección se describe el enfoque HCM sobre el cual se aplican las metaheurísticas para encontrar los mejores centroides iniciales. Consideremos un conjunto de n vectores $X = \{x_1, x_2, \ldots, x_n\}$ a ser agrupados en c grupos de datos parecidos. Cada x_i es un vector de características del objeto representado por x_i. Las características pueden ser longitud, ancho, color, etc.

La forma en que se determina la pertenencia de un objeto a un *cluster* u otro se puede determinar, por ejemplo, de manera estricta (llamado *hard*) o por el contrario, de manera más relajada (llamado *fuzzy*) en la cual un objeto puede pertenecer a diferentes *clusters*, pero con diferentes niveles de pertenencia. En este trabajo, estamos interesados en el primer caso, o *hard clustering*. De aquí en más asumiremos este tipo de agrupamiento estricto o *hard* cuando hagamos referencia a un *cluster*.

Los distintos *clusters* de los objetos pueden ser representados por una matriz de miembros llamada partición *hard*. El conjunto de datos $c \times n$ matrices de partición *hard* no degeneradas se denota por M_{cn} y se define como:

$$M_{cn} = \left\{ U \in R^{c \times n} \middle| \sum_{i=1}^{c} U_{ik} = 1, 0 < \sum_{k=1}^{n} U_{ik} < n, \text{ y } U_{ik} \in \{0,1\}; 1 \leq i \leq c; 1 \leq k \leq n. \right\} \quad (1)$$

Mientras que el criterio para determinar la calidad de un *cluster* para particiones *c-means* puede ser medido por la función HCM:

$$J_1(U, V) = \sum_{i=1}^{c} \sum_{k=1}^{n} (U_{ik}) D_{ik}^2(v_i, v_k) \quad (2)$$

donde $U \in M_{cn}$ es una matriz de partición *hard*; $V = [v_1, ..., v_c]$ es una matriz de parámetros centroides $v_i \in R^s$ y $D_{ik}(v_i, v_k)$ es una medida de distancia de v_k al i-ésimo centroide. Por ejemplo, en [9] se usa como métrica de distancia la Euclideana.

2.1. Algoritmos Genéticos

Los algoritmos genéticos fueron desarrollados por John H. Holland a principios de los 1960s [3], [4], motivado por resolver problemas de aprendizaje de máquina. Desde entonces, los AGs han evolucionado rápidamente (junto con otros algoritmos de la familia de algoritmos evolutivos) hasta llegar a conformar una de las metaheurísticas más ampliantente estudiadas y aplicadas durante los últimos años ([15], [2], [13], [14]).

En términos generales, el AG enfatiza la importancia de la recombinación (operador principal) sobre el de la mutación (operador secundario), y usa selección probabilística.
Los pasos del algoritmo genético básico descripto en forma general son los siguientes:

- Generar (aleatoriamente) una población inicial.
- Calcular la aptitud de cada individuo.
- Seleccionar (probabilísticamente) en base a la aptitud.
- Aplicar operadores genéticos (recombinación y mutación) para generar la siguiente población.
- Ciclar hasta que cierta condición se satisfaga.

Para el problema de *clustering* según planteado al inicio de esta sección, se puede visualizar a la población del AG de la siguiente manera: en cualquier generación el elemento i de una población es V_i, una matriz de $c \times n$ centroides en la notación HCM. La población inicial de tamaño P se contruye por la asignación aleatoria de números para cada una de las s características de los c centroides. Los valores iniciales están restringidos a estar en el rango (determinado por el conjunto de datos) del cual son asignados, pero por otra parte son aleatorios. Debido a que únicamente los Vs serán usados por el AG es necesario reformular la función objetivo (2) para la optimización. Para cada vector de datos HCM se asigna el *cluster* más cercano a través de la distancia métrica (en este caso se usa la distancia Euclideana, aunque otras alternativas son posibles).

Dada la manera en que las asignaciones a los *cluster* se hacen, se da que

$$R_1(V) = \sum_{k=1}^{n} \text{mín}\{D_{1k}, D_{2k}, \ldots, D_{ck}\} \tag{3}$$

es una reformulación equivalente a J_1 que elimina U [11].

En el Algoritmo 1 se muestra el algoritmo genético simple utilizado en este problema de *clustering*. El algoritmo crea una población inicial $P(0)$ con μ soluciones, en este caso con c centroides y luego evalua esas soluciones. La evaluación se realiza utilizando la ecuación 3. Luego la población ingresa en un ciclo donde evoluciona, lo cual significa que se le aplica operadores de recombinación y mutación y se crean λ hijos. Finalmente, cada iteracion finaliza seleccionando μ individuos para construir la nueva población. En este caso, el criterio de parada para el ciclo es alcanzar el número máximo de evaluaciones ($maxEvaluaciones$).

Algorithm 1 Algortimo Genético Simple

t=0; {generación actual}
inicializa $(P(t))$;
evalua $(P(t))$;
while (**not** $maxEvaluaciones$) **do**
  $P'(t) =$ evoluciona $(P(t)$; {recombinación y mutación}
  evalua$(P'(t))$;
  $P(t+1) =$ selecciona la nueva población de $P'(t) \bigcup P(t)$
  $t = t + 1$;
end while

2.2. Optimización Basada en Cúmulo de Partículas

El PSO es un algoritmo que utiliza una población de soluciones potenciales que evolucionan a la solución óptima (o muy cercana de ésta) de un determinado problema. La principal diferencia del

PSO y la computación evolutiva es un mecanismo que no parece tener un análogo en estas últimas, en el cual las partículas o individuos sobrevuelan a través del hiperespacio de búsqueda del problema [8]. El movimiento de una partícula está influenciado por su velocidad y las posiciones donde se encontraron buenas soluciones.

Para el problema de *clustering* estudiado en el presente trabajo se utilizó un modelo de PSO binario donde cada individuo de la población sólo tiene en mente la decisión binaria que tiene que tomar: si/no, falso/verdadero, etc. Cada individuo está rodeado por otros que también tienen que tomar su propia decisión. La información que tiene disponible para poder tomar la mejor decisión es su propia experiencia y la experiencia de sus vecinos. Para ello, Kennedy y Eberhart [7] proponen un modelo matemático donde la probabilidad de que un individuo decida si/no, falso/verdadero o alguna otra decisión binaria es una función f que depende de factores personales y sociales:

$$P[x_{id}(t) = 1] = f[x_{id}(t-1), v_{id}(t-1), p_{id}, p_{gd}] \tag{4}$$

donde:

- $P[x_{id}(t) = 1]$ es la probabilidad de que el individuo i seleccione 1 para el bit en la posición d de la cadena binaria.
- $x_{id}(t)$ es el estado actual del bit d de la cadena binaria
- $v_{id}(t-1)$ es la medida de la predisposición individual o la probabilidad actual de seleccionar 1.
- p_{id} es el mejor estado encontrado para el bit d de la cadena binaria de acuerdo a la experiencia personal del individuo.
- p_{gd} es el mejor estado encontrado para el bit d de la cadena binaria encontrado por el mejor individuo del vecindario.

Si $v_{id}(t)$ es grande, el individuo tiene una mayor predisposición a seleccionar 1, mientras que valores pequeños favorecen al 0. Para ubicar el valor de este parámetro dentro del rango $[0{,}0; 1{,}0]$, se utiliza la siguiente función sigmoidal:

$$s(v_{id}) = \frac{1}{1 + \exp(-v_{id})} \tag{5}$$

Para ajustar la disposición de cada individuo hacia el éxito personal y de la comunidad, se construye una fórmula para cada v_{id} en el momento actual, que será una función de la diferencia entre el estado actual del individuo y el mejor estado encontrado por él y sus vecinos. En cualquier situación no se puede determinar cuál será el factor de mayor influencia para tomar la decisión: el factor individual o el social, por lo cual cada uno de estos factores es multiplicado por un número aleatorio para que se vaya alternando el efecto de cada uno de ellos. La fórmula de decisión binaria es:

$$\begin{gathered} v_{id} = v_{id}(t-1) + \phi_1 r_1[p_{id} - x_{id}(t-1)] + \phi_2 r_2[p_{gd} - x_{id}(t-1)] \\ \text{Si } w_{id} < s[v_{id}(t)] \text{ entonces } x_{id}(t) = 1; \text{ en otro caso, } x_{id}(t) = 0 \end{gathered} \tag{6}$$

donde ϕ_1 y ϕ_2 representan constantes positivas de aceleración que escalan respectivamente la contribución cognitiva y social; $r_1, r_2 \sim U(0,1)$ y w_{id} es un vector de números aleatorios entre 0,0 y 1,0 con distribución uniforme. Cada una de estas fórmulas es aplicada repetidamente en cada una de las dimensiones de cada individuo, verificando cada vez si el valor actual de x_{id} resulta en una mejor evaluación que p_{id}, en cuyo caso se actualiza el valor. Algunas veces la decisión que tome el individuo estará basada en su experiencia personal y otras en su percepción de lo que los otros individuos creen. El version binaria de PSO binario se muestra en el Algoritmo 2.

Algorithm 2 Versión Binaria de PSO

repeat
  for cada individuo i en la población **do**
    if $aptitud(\overrightarrow{x_i}) > aptitud(\overrightarrow{p_i})$
      for cada dimensión d del individuo **do** $p_{id} = x_{id}$
    $g = i$
    for cada vecino j del individuo i **do**
      if $aptitud(\overrightarrow{p_j}) > aptitud(\overrightarrow{p_g})$
        $g = j$
    for cada dimensión d del individuo **do**
      $v_{id} = v_{id}(t-1) + \phi_1 r_1[p_{id} - x_{id}(t-1)] + \phi_2 r_2[p_{gd} - x_{id}(t-1)]$
      $v_{id} \in (-V_{max}, V_{max})$
      if $w_{id} < s[v_{id}(t)]$
        $x_{id}(t) = 1$
      else $x_{id}(t) = 0$
  end for
until Alcance condición de parada

Para el algoritmo PSO existe una gran variedad de parámetros que pueden ser ajustados para modificar la manera en que éste lleva a cabo la búsqueda. Los dos parámetros más importantes son V_{max} y ϕ los cuales son establecidos al inicio del algoritmo y se utilizan a lo largo de toda la búsqueda. La manipulación de estos parámetros puede causar cambios abruptos en el comportamiento del sistema. El parámetro V_{max} es utilizado para evitar que la trayectoria de la partícula se salga de control y que se expanda en ciclos cada vez más amplios en el espacio del problema hasta que eventualmente tienda al infinito. El parámetro de control ϕ también es llamado la constante de aceleración y determina el tipo de trayectoria que tomarán las partículas. La aptitud de una partícula se calcula de la misma forma que se calculó para el algoritmo AG, utilizando la ecuación 3.

3. EXPERIMENTOS Y RESULTADOS

Para probar ambos enfoques metaheurísticos estudiados en este trabajo (es decir, AGs y PSO) se generó un conjunto de datos artificiales que tiene múltiples extremos locales. Dicho conjunto de objetos tiene dos características vinculadas entre sí por la salida y de la ecuación no lineal $y = (1 + x_1^{-2} + x_2^{-1,5})^2$, donde $1 \leq x_1, x_2 \leq 5$. Esta ecuación ha sido tomada de [10], donde se encontraron seis clases para proveer un agrupamiento útil en el desarrollo de reglas difusas para modelar la salida de la ecuación sobre un conjunto de 50 valores distintos. Para el análisis realizado en el presente trabajo se generaron 10 instancias de 50 valores cada una. Para cada enfoque metaheurístico se mostrarán dos

experimentos diferentes cambiando ciertos parámetros de cada una de las metaheurísticas estudiadas aquí. Cada experimento consistió en la ejecución de 30 corridas independientes para cada una de las 10 instancias consideradas.

3.1. Asignación de valores de parámetros

Para analizar la eficiencia de los algoritmos se ha realizado un estudio previo de parámetros y se decidió mostrar los resultados obtenidos utilizando un tamaño de población (cúmulo de partículas) grande en el primer caso y pequeño en el segundo caso, con la finalidad de observar la calidad de los resultados en cada caso.

En ambos enfoques se utilizó una representación binaria del individuo o partícula, según corresponda. Donde un individuo representa una solución que contiene los seis posibles centroides. Por lo tanto, cada individuo se representó por medio de un arreglo de 96 elementos binarios. De los cuales se usaron 8 bits para representar un valor de x_1 o de x_2 con una precisión de dos lugares decimales. Como en un individuo se representaron 6 centroides, esto hizo un total de 96 $(16 * 6)$ elementos por individuo. En la Figura 1 se muestra la representación de un individuo o partícula según corresponda.

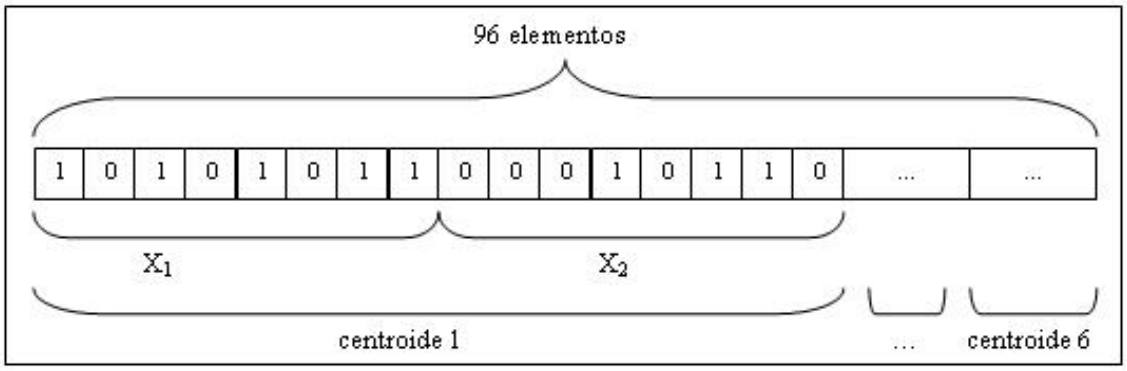

Figura 1: Representación de una solución

Para el AG se fijó una población de 250 individuos, y el número máximo de generaciones se fijó en 5000 para el primer experimento; mientras que para el segundo, se fijó una población de 25 individuos y el número máximo de generaciones de 1000. En ambos casos se aplicó un operador de recombinación de dos puntos o *Two Point Crossover* (TPX) con una probabilidad de 0,65, y el operador de mutación que se utilizó fue el de intercambio de un bit o *Bit-Flip mutation* con una probabilidad de 0,05.

Para el PSO en el primer caso se fijó el cúmulo de partículas en 600 partículas y el máximo de iteraciones en 500 y para el segundo caso se fijó el cúmulo de partículas en 10 partículas y el máximo de iteraciones en 1000. En ambos casos las constantes ϕ_1, ϕ_2 se fijaron en 2 respectivamente. El tamaño del vecindario se estableció igual a toda la población, usándose una versión global del PSO. En la Tabla 1 se muestran los parámetros utilizados por los algoritmos en cada caso planteado anteriormente, es decir, AG1, AG2, PSO1 y PSO2, respectivamente.

En la Tabla 2 se presentan los resultados obtenidos con el AG1 y PSO1 utilizando paramétros definidos para cada uno de ellos. La primer columna representa el número de la instancia, las columnas 2 y 3 corresponden a los valores mínimos (Min) y máximos (Max) obtenidos por el AG en las 30 corridas efectuadas, la columna 4 corresponde al promedio (MedEval) de evaluaciones requeridas por

Tabla 1: Parámetros de los algoritmos AG y PSO, donde NC indica “No Corresponde”.

Parámetros	AG1	AG2	Parámetros	PSO1	PSO2
Tamaño Población	250	25	Tamaño Cúmulo	600	10
Número Máx. de Generaciones	5000	1000	Número Máx. de Iteraciones	500	1000
Recombinación	TPX	TPX	ϕ_1	2	2
Mutación	Bit-Flip	Bit-Flip	ϕ_2	2	2
Prob. Recombinación	0,65	0,65	NC	NC	NC
Prob. Mutación	0,05	0,05	NC	NC	NC

dicho algortimo para encontrar el valor mínimo. Similarmente, las columnas 5 y 6 corresponden a los valores mínimos y máximos obtenidos por el PSO en las 30 corridas efectuadas y la ultima columna (columna 7) corresponde al número de evaluaciones promedio necesarias por PSO para encontrar el valor mínimo. Se puede observar que para ambos enfoques los valores mínimos y máximos de la función objetivo difieren muy poco, en todos los casos los valores obtenidos son bastante similares (las posibles diferencias entre ellos no son estadísticamente significativos acorde al análisis de Kruskawallis a un nivel de confianza del 95 %). Sin embargo, existe una importante diferencia en el promedio de evaluaciones (Med.Eval) necesarias en cada enfoque para alcanzar el resultado. Para AG1 el número de evaluaciones promedio varia entre 532225 (intancia 9) y 781883 (instancia 2) mientras que para el PSO1 el número de evaluaciones promedio varia entre 52700 (instancia 8) y 104220 (instancia 10).

Tabla 2: Resultados obtenidos con AG1 y PSO1

	AG1			PSO1		
Instancia	Min	Max	MedEval	Min	Max	MedEval
1	0,0904	0,0947	685283	0,0902	0,0916	77080
2	0,0892	0,0922	781883	0,0890	0,0905	68100
3	0,1043	0,1126	675142	0,1040	0,1123	69120
4	0,1463	0,1588	681333	0,1460	0,1546	81520
5	0,1165	0,1234	610658	0,1165	0,1166	55640
6	0,1195	0,1237	588558	0,1195	0,1202	52620
7	0,1069	0,1262	555492	0,1068	0,1076	67660
8	0,1353	0,1418	533083	0,1352	0,1363	52700
9	0,1098	0,1355	532225	0,1095	0,1104	65440
10	0,0941	0,0962	587425	0,0939	0,0946	104220

En la Tabla 3 se muestra el mismo tipo de información que en la Tabla 2 con resultados obtenidos por los algoritmos AG2 y PSO2. Se puede observar nuevamente que para ambos enfoques los valores mínimos y máximos de la función objetivo difieren muy poco, en todos los casos los valores obtenidos son bastante similares (las posibles diferencias entre ellos no son estadísticamente significativos acorde al análisis de Kruskawallis a un nivel de confianza del 95 %). Aunque igual al caso de AG1 y PSO1, existe una importante diferencia en el promedio de evaluaciones necesarias en cada enfoque para alcanzar los mejores valores. Así, para AG2 el número de evaluaciones promedio varía

entre 6667 (intancia 7) y 12899 (instancia 8), mientras que para el PSO2 el número de evaluaciones promedio varía entre 1682 (instancia 10) y 2560 (instancia 7).

Tabla 3: Resultados obtenidos con AG2 y PSO2

	AG2			PSO2		
Instancia	Min	Max	MedEval	Min	Max	MedEval
1	0,0941	0,1136	10979	0,0904	0,0959	1793
2	0,0933	0,1568	9540	0,0890	0,0928	2040
3	0,1135	0,1585	12155	0,1042	0,1438	1704
4	0,1588	0,1893	8325	0,1468	0,1577	1798
5	0,1199	0,1433	9666	0,1165	0,1219	2198
6	0,1224	0,1646	10473	0,1195	0,1428	1770
7	0,1158	0,1715	6667	0,1069	0,1268	2560
8	0,1402	0,1906	12899	0,1352	0,1391	2202
9	0,1194	0,1860	12284	0,1098	0,1427	1781
10	0,0969	0,1130	10830	0,0940	0,1005	1682

En la Figura 2 se muestra las curvas de nivel de la función utilizada en este problema $y = (1 + x_1^{-2} + x_2^{-1,5})^2$, donde $1 \leq x_1, x_2 \leq 5$), los 50 puntos (símbolo " + ") correspondientes a la instancia 2 y los seis centoides (símbolo " ∘ ") encontrados por el algoritmo AG2 para esa instancia.

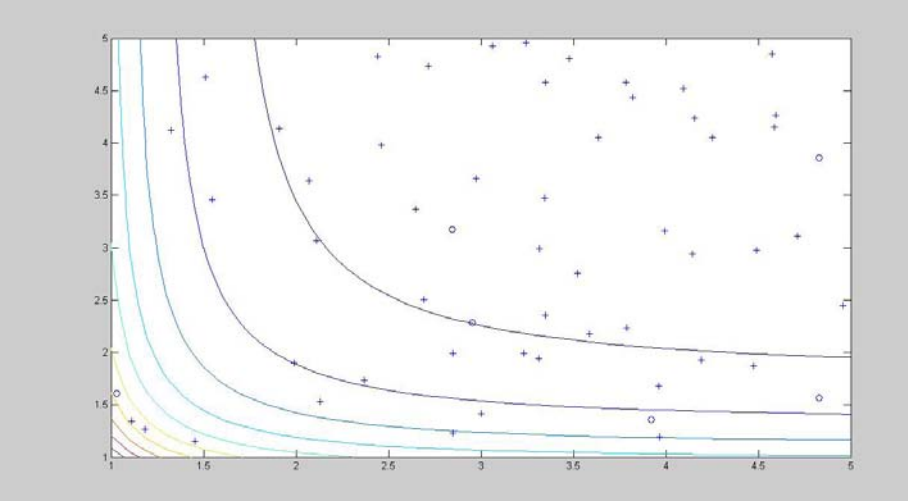

Figura 2: Distribución de centroides (AG2)

En la Figura 3 se muestra las curvas de nivel de la función utilizada en este problema $y = (1 + x_1^{-2} + x_2^{-1,5})^2$, donde $1 \leq x_1, x_2 \leq 5$), los 50 puntos (símbolo " + ") correspondientes a la instancia 2 y los seis centoides (símbolo " ∘ ") encontrados por el algoritmo PSO2 para esa instancia.

4. CONCLUSIONES

En este trabajo se ha presentado el uso de dos metaheurísticas AG y PSO para aminorar el problema de elección de los valores de inicialización para los algoritmos de *clustering c-means*. Se ha

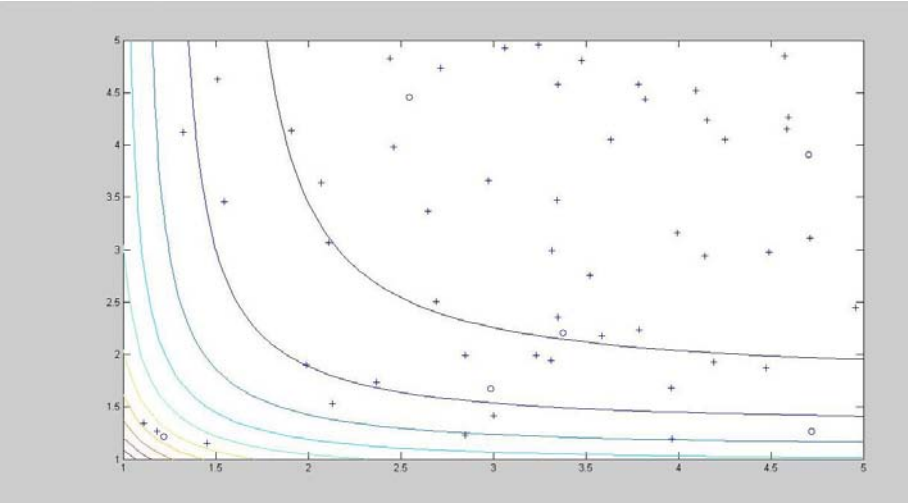

Figura 3: Distribución de centroides (PSO2)

realizado una serie de experimentos preliminares utilizando una ecuación no lineal de una sóla característica que contiene múltiples extremos locales. Los resultados demuestran que ambas técnicas proveen centroides bastante similares, pero PSO requiere un menor número de evaluaciones para obtener los resultados. Para un tamaño de población grande PSO requiere un 86 % menos en el número de evaluaciones promedio respecto al AG. Por otro lado, cuando el tamaño de la población es menor, el número de evaluaciones promedio del PSO se reduce en un 80 % respecto al AG. Consecuentemente, se puede concluir que independientemente del tamaño de la población PSO obtiene resultados muy similares con una importante reducción del número de evaluaciones requeridas.

Como trabajo futuro se pretende analizar en detalle el porque de la disminución abrupta en el número de evaluaciones del algoritmo PSO respecto al AG, lo que se produjo sin comprometer la calidad de los resultados. Además, se incluirá con un estudio en profundidad considerando instancias con una mayor cantidad de características y de mayor cantidad de datos y objetos a agrupar.

5. AGRADECIMIENTOS

EL primer y segundo autor agradecen a la Universidad Nacional de la Patagonia Austral por su apoyo al grupo de investigación y además, la cooperación de los integrantes del proyecto que continuamente proveen de nuevas ideas y críticas constructivas. El tercer autor agradece el constante apoyo brindado por la Universidad Nacional de San Luis y la ANPYCIT que financian sus actuales investigaciones

REFERENCIAS

[1] Hussein A., Ruhul A. S., and Charles S. N. *DATA MINING: A heuristic approach*. Idea Group Publishing, 2002.

[2] Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman, Inc, 2002.

[3] Holland J. H. Concerning efficient adaptive systems. In M.C. Yovits, G.T. Jacobi, and G.D. Goldstein, editors, *Self-Organizing Systems-1962*, pages 215–230, Washington D. C., 1962. Spartan Books.

[4] Holland J. H. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, pages 9:297–314, 1962.

[5] Holland J. H. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, Massachusetts, first edition edition, 1975.

[6] Witten I. H. and Frank E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

[7] Kennedy J. and Eberhart R. A discrete binary version of the particle swarm algorithm. In *Proceedings of the 1997 IEEE Conference on Systems, Man, and Cybernetics*, pages 4104–4109, Piscataway, New Jersey, 1997. IEEE Service Center, 1997.

[8] Kennedy J. and Eberhart R. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, California, 2001.

[9] Bezdek J.C. *Pattern Recognition with Fuzzy Objective Functions*. 1981.

[10] Sugeno M. and Yasukawa T. A fuzzy logic based approach to qualitative modeling. *IEEE Trans. Fuzzy Systems*, 1:7–31, Feb 1993.

[11] Hall L. O., Özyurt I. B., and Bezdek J. C. Clustering with a genetically optimized approach. In *IEEE Transactions on Evolutionary Computation*, volume 2, pages 103–112, July 1999.

[12] Duda R.O. and Hart P.E. *Pattern Classification and Scence Analysis*. Wiley, new york edition, 1973.

[13] Bäck T. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[14] Bäck T., Fogel D.B., Whitley D., and Angeline P.J. Mutation operators. In *Evolutionary Computation 1. Basic Algorithms and Operators*. IOP Publishing Lt., 2000.

[15] Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.