Metaheurísticas aplicadas a Clustering

Andrea Villagra

Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
(5700) San Luis
Argentina

e-mail: avillagra@uaco.unpa.edu.ar

Tesis para optar por el grado de
Master en Ciencias de la Computación

Tutor:

Dr. Guillermo Leguizamón
Universidad Nacional de San Luis

A Luigi, Luciano y Daniel
los tres amores de mi vida.

Agradecimientos

Quiero agradecer a todos aquellos que hicieron posible que esta tesis llegue a su fin.

En primer lugar a mi director, el Dr. Guillermo Leguizamón, por sus valiosas ideas y comentarios, por su disponibilidad para atender a todas mis consultas.

A la Universidad Nacional de la Patagonia Austral, por la ayuda económica durante el desarrollo de esta tesis.

A mis compañeros del LabTEm por incentivarme a que termine esta tesis.

A mis padres, por estar siempre dispuestos cuando los necesite.

Y finalmente, a mi esposo Daniel, por apoyarme, ayudarme, aconsejarme y tenerme tanta paciencia.

Resumen

Las metaheurísticas son estrategias de alto nivel que utilizan otras estrategias para explorar el espacio de búsqueda. La optimización basada en cúmulo de partículas (PSO) es una metaheurística poblacional que ha sido exitosamente aplicada para resolver problemas de optimización.
Clustering es una tarea de minería de datos y puede definirse como el proceso de agrupar un conjunto de objetos abstractos o físicos en clases similares. Un algoritmo de clustering ampliamente aplicado en diversas aplicaciones es el algoritmo de K-means.
Un problema de clustering puede verse como un problema de optimización que localiza los centroides óptimos de los clusters en lugar de encontrar la partición optima.
En este trabajo se combina la metaheurística PSO con el algoritmo de K-means aplicada al problema de clustering, con el objetivo de mejorar los resultados obtenidos por ambos algoritmos de forma independiente.
Los resultados obtenidos con esta propuesta híbrida han sido promisorios cuando se los ha aplicado a conjuntos de datos globulares y separados. Es por esta razón que se extiende esta propuesta y se presenta un algoritmo que incorpora la determinación automática del numero de clusters. Obteniendo resultados promisorios con esta propuesta dinámica, se presentan dos variantes que incorporan cambios en el tipo de PSO aplicado. Estas variantes resultan ser más robustas que su predecesora. En los siguientes capítulos se presentan conceptos teóricos y se muestran los resultados de los algoritmos y el análisis estadístico de los mismos.

Indice General

Indice de figuras

Indice de Tablas

Indice de Algoritmos

Capítulo 1

Introducción

El término metaheurística fue introducido por Glover en 1986 [37] y deriva de dos palabras griegas. Heurística deriva de *heuristikein* (en griego) que significa encontrar, y el prefijo *Meta*, que significa de alto nivel. El concepto es ampliamente usado en la literatura para denominar algoritmos tales como Algoritmos Evolutivos (AEs), Simulated Annealing (SA), Tabu Search (TS), Particle Swarm Optimización (PSO), entre otros.
La principal característica de estos algoritmos [29] que son descritos en la literatura pueden ser resumidas en propiedades fundamentales tales como: (a) la búsqueda no es guiada por la naturaleza específica del problema sino por una estrategia de búsqueda de alto nivel, (b) usualmente, no son algoritmos determinísticos y usan diferentes mecanismos para no quedar atrapados en óptimos locales y (c) tendencias avanzadas usan mecanismos para mejorar el proceso de búsqueda logrando un buen balance entre la diversificación y la intensificación.
El término intensificación se refiere a la explotación de algún área promisoria del espacio de búsqueda, mientras que la diversificación se refiere a la exploración para identificar estas áreas promisorias. En 2003, Blum y Roli [14] describen la importancia de estos dos conceptos en su estudio del arte acerca de metaheurísticas. La estrategia de exploración del espacio de búsqueda en cada metaheurística en particular depende de su filosofía.

La minería de datos es considerada uno de los puntos más importantes de los sistemas de bases de datos y uno de los desarrollos más prometedores interdisciplinariamente en la industria de la informática. La minería de da-

tos representa la posibilidad de buscar información dentro de un conjunto de datos con la finalidad de extraer información nueva y útil que se encuentra oculta en grandes volúmenes de datos [93].
El clustering es una de las principales tareas en el proceso de minería de datos para descubrir grupos e identificar distribuciones y características interesantes en los datos. Es una disciplina científica muy joven que actualmente se encuentra bajo un vigoroso desarrollo. El análisis de clustering en minería de datos ha desempeñado un rol muy importante en una amplia variedad de áreas tales como: reconocimiento de patrones, análisis de datos espaciales, procesamiento de imágenes, análisis médico, economía, bioinformática y biometría principalmente ([6], [24], [68], [87], [120], [121], [138]).
El proceso de agrupar un conjunto de objetos físicos o abstractos dentro de clases con objetos similares se denomina clustering. El clustering consiste en agrupar una colección dada de datos no etiquetados en un conjunto de grupos de tal manera que los objetos que pertenecen a un grupo sean homogéneos entre sí y buscando además, que la heterogeneidad entre los distintos grupos sea lo más elevada posible [93].
En la literatura existe una amplia variedad de algoritmos de clustering y muchos de ellos requieren especificar con anticipación el número de clusters. Determinar este valor requiere de un conocimiento *a priori* del conjunto de datos y lamentablemente este conocimiento no siempre es posible. Encontrar el número de clusters óptimo para un conjunto de datos, es una tarea muy importante y ha sido abordada en distintas investigaciones [16], [78], [119], entre otras.

1.1. Motivación

El clustering es una de las tareas del aprendizaje no supervisado en minería de datos en la que no se requiere una clasificación predefinida de los datos, para particionarlos y obtener conocimiento de los mismos. Las técnicas de clustering proveen una variedad de algoritmos con características deseables para el descubrimiento de conocimiento contenido en los datos. Un algoritmo de clustering particional muy utilizado es el algoritmo de K-means, pues es

muy sencillo de implementar y es uno de los más eficientes en términos del tiempo de ejecución. Sin embargo, este algoritmo es sensible a la selección de la partición inicial y puede converger a óptimos locales. Además, necesita conocer con antelación el número de clusters del conjunto de datos para poder realizar las particiones. Por estas razones, conociendo que la metaheurística denominada Particle Swarm Optimization (PSO) ha resultado ser eficiente para resolver problemas de optimización y sabiendo que un problema de clustering puede ser visto como un problema de optimización, se decidió utilizar esta metaheurística combinada con el algoritmo de K-means. La hibridación de las metaheurísticas produce muy buenos resultados y es por esto que en esta tesis se proponen algoritmos híbridos. En primer lugar se presenta el algoritmo denominado PSO+K-means que intenta hacer uso de las fortalezas de ambos algoritmos (PSO y K-means) para obtener mejores resultados. Luego se presentan tres variantes del primer algoritmo. Estas variantes denominadas PKD-G, PKD-L y PKD-GL son una extensión del primer algoritmo con las siguientes características: incorporan la determinación automática del número de clusters y cada una de ellas implementa un algoritmo de PSO diferente.

1.2. Aportes de la tesis

- Hybrid PSO Algorithms for Dynamic Clustering; Villagra A., Pandolfi D., Leguizamón G.; The 23rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems; Junio 1-4, 2010 Córdoba (España).(En proceso de evaluación)

- Análisis de medidas no-supervisadas de calidad en clusters obtenidos por K-means y Particle Swarm Optimization; Villagra A., Guzmán A., Pandolfi D., Leguizamón G.; Congreso de Inteligencia Computacional Aplicada (CICA), en la ciudad de Buenos Aires.; Julio 2009.

- Selección de Centroides para Algoritmos de Clustering a través de Técnicas Metaheurísticas; Villagra A., Pandolfi D., Leguizamón G.; XIII Congreso Argentino de Ciencias de la Computación - CACIC 2007; Uni-

versidad Nacional del Nordeste, Corrientes - Universidad Tecnológica Nacional, Resistencia. Octubre 2007. Pág. 1773 - 1783.

- Hibridización de K-Means a través de Técnicas Metaheurísticas; Villagra A., Pandolfi D., Leguizamón G.; X Workshop de Investigadores de Ciencias de la Computación - WICC 2008; La Pampa, Argentina; Mayo 2008.; Pág. 62-65;

- Algoritmos Evolutivos y su aplicabilidad en la tarea de Clasificación; Villagra A., Pandolfi D., Lasso M., de San Pedro M., Leguizamón G. VIII Workshop de Investigadores de Ciencias de la Computación WICC'06. Universidad de Morón, Bs.As. Junio 2006. Pág. 203-207

1.3. Organización de la tesis

En el segundo capítulo se realiza en primer lugar una introducción a las técnicas metaheurísticas de optimización, estableciendo una clasificación de las más populares. En segundo lugar se introduce un tipo concreto de éstas técnicas metaheurísticas como son los algoritmos basados en cúmulos de partículas, una descripción más detallada presentando sus principales características y aspectos fundamentales. Finalizando con una breve descripción de metaheurísticas híbridas.
En el tercer capítulo se describe brevemente la técnica de minería de datos junto con el proceso de descubrimiento de conocimiento en bases de datos. Seguidamente se continúa con *Clustering* una de las tareas de minería de datos, describiendo medidas de distancia, técnicas y medidas de evaluación. Se finaliza con el algoritmo de *K-means*, un algoritmo de *clustering*, presentando sus principales características.
En el cuarto capítulo se presenta una revisión de los algoritmos de clustering con metaheurísticas, en particular metaheurísticas basadas en poblaciones y clustering automático con diferentes algoritmos.
En el quinto capítulo se presenta una propuesta híbrida denominado PSO+K-means, que intenta combinar las mejores características de ambos algoritmos, con el fin de obtener mejores resultados en los clusters obtenidos.

En el sexto capítulo se presenta una propuesta dinámica que agrega a la propuesta anterior la posibilidad de determinar automáticamente el número de clusters óptimo para un conjunto de datos, denominada PKD-G. Seguidamente se proponen dos variantes a esta versión dinámica denominadas PKD-L y PKD-GL, con el objetivo de mejorar los resultados de PKD-G.
Finalmente, en el séptimo capítulo se presentan las conclusiones logradas por los algoritmos propuestos y los trabajos futuros que pueden derivar de esta tesis.

Capítulo 2

Metaheurísticas

2.1. Introducción a las técnicas metaheurísticas de Optimización

La optimización, en el sentido de encontrar la mejor solución o al menos una solución lo suficientemente buena para un problema es un campo de vital importancia en la vida real. Constantemente estamos resolviendo pequeños problemas de optimización, como encontrar el camino más corto para ir de un lugar a otro, la organización de la agenda, etc. En general estos problemas son lo suficientemente pequeños y no requieren de una computadora. Pero cuando se hacen más complejos es inevitable su uso.
Debido a la gran importancia de los problemas de optimización se han desarrollado múltiples métodos para resolverlos. Una clasificación muy simple de estos métodos se muestra en la Figura 2.1.

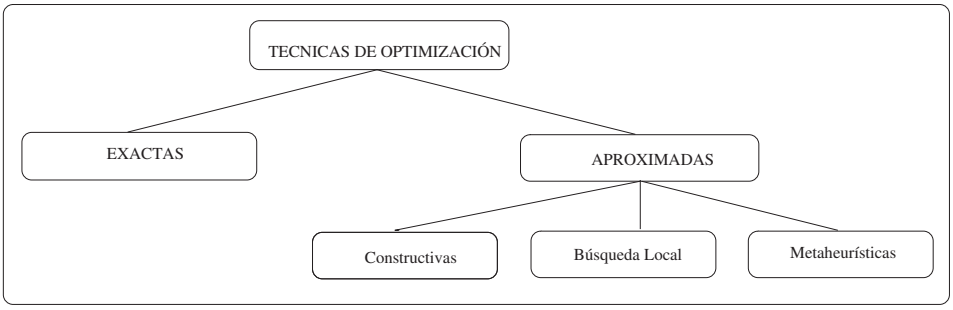


Figura 2.1: Clasificación de las técnicas de optimización

Inicialmente las técnicas se pueden clasificar en exactas (o enumerativas, exhaustivas, etc.) y técnicas aproximadas. Las técnicas exactas garantizan encontrar la solución óptima pero el inconveniente es que para obtener esta

solución óptima el tiempo necesario crece exponencialmente con el tamaño del problema y en algunos casos es imposible encontrarla por el tiempo que demanda. Por esta razón los algoritmos aproximados intentan resolver estos problemas, sacrificando la garantía de encontrar el óptimo a cambio de encontrar una "buena" solución en un tiempo "razonable".
Dentro de los algoritmos no exactos se pueden encontrar tres tipos: los heurísticos constructivos (también llamados voraces), los métodos de búsqueda local (o métodos de seguimiento del gradiente) y las metaheurísticas.
Se puede decir que una metaheurística es una estrategia de alto nivel que usa diferentes estrategias para explorar el espacio de búsqueda.
Hay diferentes formas de clasificar y describir las técnicas metaheurísticas [123]. Dependiendo de las características que se seleccionen se pueden obtener diferentes taxonomías: basadas en la naturaleza o no basadas en la naturaleza, basadas en memoria o sin memoria, con función objetivo estática o dinámica, etc. En este trabajo se ha elegido clasificarlas de acuerdo a si en cada paso manipulan un único punto del espacio de búsqueda o trabajan sobre un conjunto (población) de ellos, es decir, esta clasificación divide a las metaheurísticas en basadas en trayectoria y basadas en población. Esta clasificación es ampliamente utilizada en la comunidad científica y se la muestra de forma gráfica en la Figura 2.2.

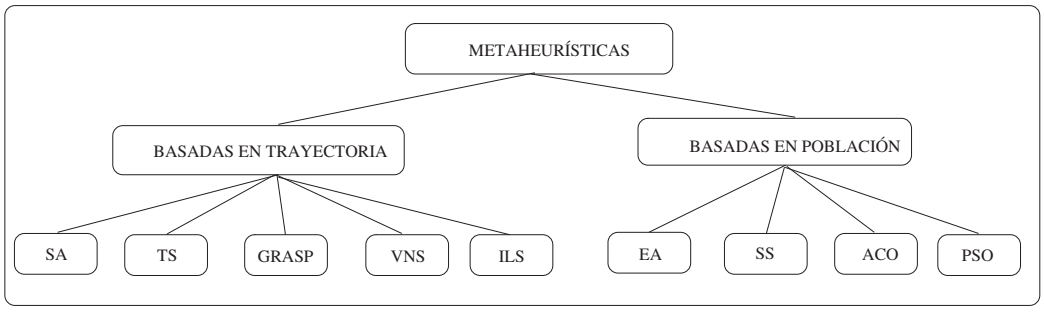


Figura 2.2: Clasificación de las Metaheurísticas

Las siglas de la figura se corresponden de la siguiente manera: *Simulated Annealing* (SA), *Tabu Search* (TS), *Greedy Randomized Adaptive Search Procedure* (GRASP), *Variable Neighborhood Search* (VNS), *Iterated Local Search* (ILS), *Evolutionary Algorithms* (EA), *Scatter Search* (SS), *Ant Colony Optimization* (ACO), *Particle Swarm Optimization* (PSO) y serán descriptas a

continuación.

2.1.1. Metaheurísticas Basadas en Trayectoria

En las metaheurísticas basadas en trayectoria, la principal característica es que parten de un punto y mediante la exploración del vecindario van variando la solución actual, formando una trayectoria. La mayoría de estos algoritmos surgen como extensiones de los métodos de búsqueda local simple a los que se les añade alguna característica para escapar de los mínimos locales. Esto implica la necesidad de una condición de parada más compleja que la de encontrar un mínimo local. Normalmente se termina la búsqueda cuando se alcanza un número máximo predefinido de iteraciones, se encuentra una solución con una calidad aceptable, o se detecta un estancamiento del proceso.

- El Enfriamiento Simulado o *Simulated Annealing* (SA) es uno de los más antiguos entre las metaheurísticas y seguramente es el primer algoritmo con una estrategia explícita para escapar de los óptimos locales. El SA fue inicialmente presentado en [115]. La idea de este algoritmo es simular el proceso de recocido del metal y del cristal. Para evitar quedar atrapado en un óptimo local, el algoritmo permite elegir una solución peor que la solución actual. En cada iteración se elige, a partir de la solución actual s, una solución s' del vecindario $N(s)$. Si s' es mejor que s (es decir que tiene un mejor valor de función de *fitness*), se sustituye s por s' como solución actual. Si la solución s' es peor, entonces es aceptada con una determinada probabilidad que depende de la temperatura actual T y de la variación en la función de *fitness*, $f(s') - f(s)$ (caso de minimización). Esta probabilidad generalmente se calcula siguiendo la distribución de Boltzmann:

$$p(s' \mid T, s) = e^{-\frac{f(s')-f(s)}{T}} \tag{2.1}$$

- La Búsqueda Tabu o *Tabu Search* (TS) es una de las metaheurísticas que se han aplicado con más éxito a la hora de resolver problemas de optimización combinatoria. Los fundamentos de este método fueron introducidos en [37]. La idea básica de la búsqueda tabú es el uso explícito de un historial de la búsqueda (una memoria de corto plazo), tanto para

escapar de los óptimos locales como para implementar su estrategia de exploración y evitar buscar varias veces en la misma región. Esta memoria a corto plazo es implementada en esta técnica como una lista tabú, donde se mantienen las soluciones visitadas más recientemente para excluirlas de los próximos movimientos. En cada iteración se elige la mejor solución entre las permitidas y la solución es añadida a la lista tabú.

- El Procedimiento de Búsqueda Miope Aleatorizado y Adaptativo o *The Greedy Randomized Adaptive Search Procedure* (GRASP)[128] es una metaheurística simple que combina heurísticos constructivos con búsqueda local. GRASP es un procedimiento iterativo compuesto de dos fases: primero una construcción de una solución y después un proceso de mejora. La solución mejorada es el resultado del proceso de búsqueda.

- La Búsqueda en Vecindario Variable o *Variable Neighborhood Search* (VNS) es una metaheurística propuesta en [85], que aplica explícitamente una estrategia para cambiar entre diferentes estructuras de vecindario de entre un conjunto de ellas definidas al inicio del algoritmo. Este algoritmo es muy general y con muchos grados de libertad a la hora de diseñar variaciones e instanciaciones particulares.

- La Búsqueda Local Iterada o *Iterated Local Search* (ILS)[127] es una metaheurística basada en un concepto simple pero muy efectivo. En cada iteración, la solución actual es perturbada y a esta nueva solución se le aplica un método de búsqueda local para mejorarla. Este nuevo óptimo local obtenido por el método de mejora puede ser aceptado como nueva solución actual si pasa un test de aceptación.

2.1.2. Metaheurísticas Basadas en Población

Las metaheurísticas basadas en población se caracterizan por trabajar con un conjunto de soluciones (población) en cada iteración, a diferencia de los métodos anteriormente descritos que únicamente utilizan un punto del espacio de búsqueda por iteración. El resultado final proporcionado por este tipo de algoritmos depende fuertemente de la forma en que manipula la población.

- Algoritmos Evolutivos o *Evolutionary Algorithms* (EA) [122] se inspiran en la capacidad de la naturaleza para evolucionar seres y adaptarlos a los cambios. Esta familia de técnicas sigue un proceso iterativo y estocástico que opera sobre una población de individuos. Cada individuo representa una solución potencial al problema que se está resolviendo. Inicialmente, la población se genera aleatoriamente. Cada individuo en la población tiene asignado, por medio de una función de aptitud (*fitness*), una medida de su bondad con respecto al problema bajo consideración. Este valor es la información cuantitativa que el algoritmo usa para guiar su búsqueda. La modificación de la población se lleva a cabo mediante tres operadores: selección, recombinación y mutación. Se han propuesto diferentes algoritmos basados en este esquema general que fueron desarrollados independientemente: Programación Evolutiva desarrollada por Fogel [74]; Estrategias Evolutivas, propuesta por Rechenberg en [47]; Algoritmos Genéticos introducidos por Holland en [44] y Evolución Diferencial propuesta por Storn y Price en [100] y [101].

- Búsqueda Dispersa o *Scatter Search* (SS) [38] se basa en mantener un conjunto relativamente pequeño de soluciones tentativas (llamado "*conjunto de referencia*") que se caracteriza tanto por contener buenas soluciones como soluciones diversas. Este conjunto se divide en subconjuntos de soluciones a las cuales se les aplica una operación de recombinación y mejora. Para realizar la mejora o refinamiento de soluciones se suelen usar mecanismos de búsqueda local.

- Colonia de Hormigas o *Ant Colony Optimization* [77] (ACO) inspirada en el comportamiento de las hormigas cuando realizan la búsqueda de comida. Este comportamiento es el siguiente: inicialmente, las hormigas exploran el área cercana a su nido de forma aleatoria. Tan pronto como una hormiga encuentra comida, la lleva al nido, dejando en su camino una sustancia química denominada feromona. Esta sustancia ayudará al resto de las hormigas a encontrar el camino de la comida. Esta comunicación indirecta entre las hormigas mediante el rastro de feromona las capacita para encontrar el camino más corto entre el nido y la comi-

da. La metaheurística intenta simular esta funcionalidad para resolver problemas de optimización.

- Algoritmos Basados en Cúmulos de Partículas o *Particle Swarm Optimization* (PSO)[52] están inspirados en el comportamiento social del vuelo de bandadas de aves o el movimiento de los bancos de peces. Se fundamenta en los factores que influyen en la toma de decisión de un agente que forma parte de un conjunto de agentes similares. La decisión de cada agente se realiza conforme a un componente social y un componente individual, mediante los que se determina el movimiento de este agente para alcanzar una nueva posición en el espacio de soluciones. La metaheurística intenta simular este comportamiento para resolver problemas de optimización.

En esta Sección se ha descrito de forma introductoria el campo de las metaheurísticas. Dado que este trabajo se centra en los algoritmos basados en cúmulo de partículas, los mismos se describen detalladamente a continuación.

2.2. Principios de Cúmulos de Partículas

Durante mucho tiempo se ha intentado modelar el comportamiento psicológico, físico y biológico del universo y de los seres de los que se tiene conocimiento. La ciencia de la computación no podría ser la excepción.
La cultura humana y la cognición son aspectos de un proceso individual. Las personas aprenden de otras personas no sólo datos sino métodos para procesarlos. Bandura ha explicado teorías acerca del aprendizaje que ocurre cuando los individuos observan el comportamiento de otro [1] : "Si sólo pudiéramos adquirir el conocimiento a través del efecto de las acciones de uno mismo, el proceso del desarrollo del conocimiento cognitivo y social podría ser en gran medida lento, por no decir extremadamente tedioso .... Afortunadamente, la mayor parte del comportamiento humano es aprendido a través del modelado .... La capacidad de aprender por medio de la observación posibilita ampliar sus conocimientos e ideas sobre las bases de información exhibidas y publicadas por otros."

El aprendizaje no sólo se da de una persona a otra. Un individuo puede adquirir el conocimiento por algunos otros medios como los libros, medios de comunicación, Internet, etc. Sin embargo, el conocimiento y las ideas se transmiten de una persona a otra, por lo tanto la población converge a un proceso óptimo. Kennedy y Eberhart [52] describen un sistema que opera simulando el proceso de aprendizaje en tres niveles:

- Los individuos aprenden localmente de sus vecinos. Las personas son conscientes de la interacción con sus vecinos, recabando conocimientos de ellos y compartiendo los suyos. El aprendizaje social local es fácilmente medible y es un fenómeno bien documentado.

- La difusión del conocimiento a través del aprendizaje social resulta en un proceso emergente a nivel grupal. Este fenómeno sociológico, económico o político es visto regularmente en creencias, actitudes, comportamientos y otros atributos a través de individuos en una población.

- Todas las interacciones son locales, los conocimientos e innovaciones son transmitidos por la cultura desde quien lo origina hasta alguien más distante, impulsando la combinación de varias innovaciones que resultan en métodos de mejora. Este efecto global es, en gran parte, transparente para quienes interactúan en el sistema quienes son los que resultan beneficiados.

2.3. Algoritmos Basados en Cúmulos de Partículas (PSO)

Un "Algoritmo Basado en Cúmulo de Partículas"[1] o *Particle Swarm Optimization* es una técnica metaheurística basada en poblaciones e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces. PSO fue originalmente desarrollado por el psicólogo-sociólogo Jammes Kennedy y por el ingeniero electrónico Russell Eberhart en 1995, basándose en un enfoque conocido como la *"metáfora social"* [52], que describe a este algoritmo y que se puede resumir de la siguiente forma:

[1] En la literatura se puede encontrar como población, nube, enjambre o colmena (swarm) de partículas. En este trabajo se usará el término "cúmulo".

los individuos que conviven en una sociedad tienen una opinión que es parte de un "conjunto de creencias"(el espacio de búsqueda) compartido por todos los posibles individuos. Cada individuo puede modificar su propia opinión basándose en tres factores:

- Su conocimiento sobre el entorno (su valor de *fitness*)
- Su conocimiento histórico o experiencias anteriores (su memoria).
- El conocimiento histórico o experiencias anteriores de los individuos situados en su vecindario.

Siguiendo ciertas reglas de interacción, los individuos en la población adaptan sus esquemas de creencias al de los individuos con más éxito de su entorno. Con el tiempo surge una cultura cuyos individuos tienen un conjunto de creencias estrechamente relacionado.

Como se mencionó anteriormente, el principio natural en el que se basa el PSO es el comportamiento de una bandada de aves o de un cardumen. En la Figura 2.3 se muestran ejemplos de cúmulos en la naturaleza. Supongamos que una de estas bandadas busca comida en un área y que solamente hay una pieza de comida en dicha área. Los pájaros no saben dónde está la comida pero sí conocen su distancia a la misma, por lo que la estrategia más eficaz para hallar la comida es seguir al ave que se encuentre más cerca de ella. PSO simula este escenario para resolver problemas de optimización. Cada solución (partícula) es un "ave"en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere, es decir, cambia su velocidad y posición pero permanece en el cúmulo.

El cúmulo de partículas es un sistema multiagente, es decir, un sistema donde las partículas son los agentes simples que se mueven por el espacio de búsqueda y que guardan (y posiblemente comunican) la mejor solución que han encontrado. Cada partícula tiene un *fitness*, una *posición* y un *vector velocidad* que dirige su "movimiento". El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual. En

Figura 2.3: Ejemplos de cúmulos en la naturaleza

analogía con los paradigmas de computación evolutiva, un *cúmulo* es similar a una población, mientras que una *partícula* es similar a un individuo.

Los algoritmos basados en cúmulo de partículas han sido aplicados exitosamente en diferentes campos de investigación. Ejemplos de algunos trabajos son: optimización de funciones numéricas [142], entrenamiento de redes neuronales [40], aprendizaje de sistemas difusos [73], Clasificación de imágenes, minería de textos [140], Problema del viajante de comercio e ingeniería química [15], entre otros.

2.4. Algoritmo de PSO

Un algoritmo de PSO mantiene un cúmulo de partículas, donde cada partícula representa una solución al problema. Las partículas "vuelan" a través de un espacio de búsqueda multidimensional, donde la posición de cada partícula se ajusta de acuerdo a su propia experiencia y a la experiencia de sus vecinos. Supongamos que $x_i(t)$ representa la posición de la partícula i en el espacio de búsqueda en tiempo t (discreto). La posición de la partícula cambia agregando una velocidad, v_i, a la posición de la partícula, esto es,

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.2}$$

donde $x_i(0) \sim U(x_{min}, x_{max})$.

El vector velocidad dirige el proceso de optimización y refleja el conocimiento de la experiencia de la partícula y la información intercambiada

socialmente con los vecinos de la partícula. La experiencia de la partícula representa la *componente cognitiva* y la información intercambiada socialmente representa la *componente social* de la ecuación de velocidad.

Originalmente se desarrollaron dos algoritmos de PSO, que difieren en el tamaño del vecindario. Estos dos algoritmos llamados PSO Global o *gbest* PSO y PSO Local o *lbest* PSO, se muestran a continuación.

2.4.1. PSO Global

El vecindario para cada partícula en el PSO global o *gbest* PSO es el cúmulo completo. La red social empleada por *gbest* PSO refleja una topología estrella.

Para la topología estrella del vecindario, la actualización del componente social de la velocidad de la partícula refleja la información obtenida de todas las partículas del cúmulo. En este caso la información social es la mejor posición encontrada por el cúmulo, representada por $\hat{y}(t)$.

Para *gbest* PSO, la velocidad de la partícula i se calcula como:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}[\hat{y}_j(t) - x_{ij}(t)] \tag{2.3}$$

donde $v_{ij}(t)$ es la velocidad de la partícula i en la dimension $j = 1, \ldots, n_x$ en el tiempo t, $x_{ij}(t)$ es la posición de la partícula i en la dimensión j en el tiempo t, c_1 y c_2 son constantes de aceleración positiva usadas para escalar la contribución de las componentes sociales y cognitiva respectivamente, y $r_{1j}(t)$, $r_{2j}(t) \sim U(0,1)$ son valores aleatorios entre $[0,1]$, con distribución uniforme. Estos valores introducen el elemento estocástico al algoritmo.

La mejor posición personal, y_i, asociada a la partícula i es la mejor posición que la partícula ha visitado desde el primer momento. Considerando problemas de minimización, para establecer la mejor posición personal, se calcula como:

$$y_i(t+1) = \begin{cases} y_i(t) & si \quad f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & si \quad f(x_i(t+1)) < f(y_i(t)) \end{cases} \tag{2.4}$$

donde f es la función de aptitud (fitness). Al igual que en los algoritmos evolutivos, la función de fitness cuantifica la performance o calidad de una partícula (solución).

La mejor posición global, se define como:

$$\hat{y}(t) \in \{y_0, \ldots, y_{n_s}(t)\} \text{ donde} f(\hat{y}) = \text{mín}\{f(y_0(t)), \ldots, f(y_{n_s}(t))\} \quad (2.5)$$

donde n_s es el número total de partículas en el cúmulo. La ecuación 2.5 representa la mejor posición encontrada por cualquiera de las partículas. La mejor posición global puede seleccionarse entre las partículas del cúmulo actual:

$$\hat{y}(t) = min\{f(y_0(t)), \ldots, f(y_{ns}(t)))\} \quad (2.6)$$

El algoritmo *gbest* PSO se muestra a continuación en el Algoritmo 1.

Algoritmo 1 Algoritmo *gbest* PSO

```
1: Crear e Inicializar S /* S = Cúmulo de tamaño n_s*/
2: repeat
3:    for cada partícula i = 1,...,S.n_s do
4:       /*Establece la mejor posición personal*/
5:       if f(S.x_i) < f(S.y_i) then
6:          S.y_i = S.x_i
7:       end if
8:       /*Establece la mejor posición del vecindario */
9:       if f(S.y_i) < f(S.ŷ_i) then
10:         S.ŷ_i = S.y_i
11:      end if
12:   end for
13:   for cada partícula i = 1,...,S.n_s do
14:      Actualiza la velocidad usando la ecuación (2.3)
15:      Actualiza la posición usando la ecuación (2.2)
16:   end for
17: until Condición de parada sea verdadera
```

2.4.2. PSO Local

En el algoritmo PSO Local, o *Lbest* PSO, la selección del vecindario está basado simplemente en el índice de la partícula (versión básica de PSO), es decir que su estructura no necesariamente refleja una topología particular. Las razones de esto son:

- Es más eficiente.
- Promueve la dispersion de información independientemente de la ubicación de las partículas en el espacio de búsqueda.

La velocidad se calcula como:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \quad (2.7)$$

donde $\hat{y}_{ij}$ es la mejor posición, encontrada por la partícula i en su vecindario en la dimensión j. La mejor posición local de la partícula, $\hat{y}_i$, es decir, la mejor posición encontrada en el vecindario $\mathcal{N}_i$, se define como:

$$\hat{y}_i(t+1) \in \{\mathcal{N}_i | f(\hat{y}_i(t+1)) = min\{f(x)\}, \forall x \in \mathcal{N}_i\} \quad (2.8)$$

donde el vecindario se define como:

$$\mathcal{N}_i = \{y_{i-n_{\mathcal{N}_i}}(t), y_{i-n_{\mathcal{N}_i}+1}(t), \ldots, y_{i-1}(t), y_i(t), \ldots, y_{i+n_{\mathcal{N}_i}}(t)\} \quad (2.9)$$

para vecindarios de tamaño $n_{\mathcal{N}_i}$.

El *gbest* PSO es una caso especial del algoritmo *lbest* PSO con $n_{\mathcal{N}_i} = n_s$. El algoritmo *lbest* PSO se muestra en el Algoritmo 2.

2.4.3. Análisis del PSO

El cálculo de la velocidad dado en las Ecuaciones 2.3 y 2.7 está compuesto por:

- **Velocidad previa**, $v_i(t)$, representa la memoria de la dirección de vuelo previa (o movimiento pasado inmediato) y es la componente *inercial*.

Algoritmo 2 Algoritmo *lbest* PSO

```
Crear e Inicializar S /* S = Cúmulo de tamaño n_s*/
repeat
   for cada partícula i = 1,...,S.n_s do
      /*Establece la mejor posición personal*/
      if f(S.x_i) < f(S.y_i) then
         S.y_i = S.x_i
      end if
      /*Establece la mejor posición del vecindario */
      if f(S.y_i) < f(S.ŷ_i) then
         S.ŷ_i = S.y_i
      end if
   end for
   for cada partícula i = 1,...,S.n_s do
      Actualiza la velocidad usando la ecuación (2.7)
      Actualiza la posición usando la ecuación (2.2)
   end for
until Condición de parada sea verdadera
```

- **Componente cognitiva**, $c_1 r_1(y_i - x_i)$, cuantifica el desempeño de la partícula i respecto a su desempeño pasado (memoria individual).

- **Componente Social**, $c_2 r_2(\hat{y} - x_i)$, para *gbest*, o $c_2 r_2(\hat{y}_i - x_i)$ para *lbest*, cuantifica el desempeño de la partícula con respecto al cúmulo total (*gbest*) o su respectivo vecindario (*lbest*).

Los coeficientes c_1 y c_2 (y los respectivos valores aleatorios r_1 y r_2) controlan la influencia estocástica de las componentes cognitivas y sociales sobre la velocidad de la partícula; w es el factor de inercia que provee diversidad a la partícula.

Algunos aspectos a tener en cuenta en ambos algoritmos *lbest* y *gbest* PSO son: la inicialización de los parámetros y el criterio de parada.
Generalmente las posiciones de las partículas se inicializan para cubrir uniformemente el espacio de búsqueda. Es importante notar que la eficiencia del PSO está influenciada por la diversidad inicial del cúmulo, es decir, cuanto se cubre del espacio de búsqueda y cuán bien están distribuidas sobre el espacio de búsqueda. Si no se cubren regiones en el espacio de búsqueda con el

cúmulo inicial, el PSO tendrá dificultades para encontrar el óptimo si éste se encuentra en una región que no está cubierta.
Se asume que un óptimo necesita estar ubicado dentro del dominio definido por dos vectores, x_{min} y x_{max}, los que representan los rangos mínimos y máximos en cada dimensión. Por lo tanto, un método de inicialización eficiente para inicialización de una partícula es:

$$x(0) = x_{minj} + r_j(x_{maxj} - x_{minj}), \forall j = 1, \ldots, n_x, \forall i = 1, \ldots, n_s \tag{2.10}$$

donde $r_j \sim U(0, 1)$.

Las velocidades iniciales pueden inicializase en cero, es decir:

$$v_i = 0 \tag{2.11}$$

Se han propuesto diferentes esquemas para inicializar las posiciones de las partículas. Estos esquemas se han utilizado principalmente para asegurar que el espacio de búsqueda se cubra uniformemente, algunos han sido propuestos en [35], [96], [97].

Como criterios de parada para PSO se pueden usar los siguientes:

- Terminar cuando se exceda un número de iteraciones. Este criterio se usa generalmente con criterios de convergencia para forzar la finalización si el algoritmo no converge. Usado por si solo, este criterio es útil para estudios donde el objetivo es evaluar la mejor solución encontrada en un período de tiempo restringido.
- Terminar cuando se encontró una solución aceptable. Se asume que x^* representa el óptimo para la función objetivo f. Entonces este criterio terminará el proceso de búsqueda cuando una partícula x_i cumpla que $f(x_i) \leq |f(x^*) - \varepsilon|$, es decir que se alcance un error aceptable. El valor de ese error debe seleccionarse con cuidado. Esta condición de parada asume un conocimiento previo de cuál es el óptimo, sin embargo, este conocimiento generalmente no está disponible.

- Terminar cuando el radio del cúmulo esté normalizado. Este valor se calcula como [134] :

$$R_{norm} = \frac{R_{max}}{diameter(S)} \tag{2.12}$$

donde $diameter(S)$ es el diámetro del cúmulo inicial y R_{max} es:

$$R_{max} = \parallel x_m - \hat{y} \parallel, m = 1, \ldots, n_s \tag{2.13}$$

con

$$\parallel x_m - \hat{y} \parallel \geq \parallel x_i - \hat{y} \parallel, \forall i = 1, \ldots, n_s \tag{2.14}$$

El algoritmo termina cuando $R_{norm} < \varepsilon$. Si ε es demasiado grande, el proceso de búsqueda finalizará prematuramente antes de encontrar una buena solución. Caso contrario, si ε es demasiado pequeño, la búsqueda llevará demasiadas iteraciones. Una manera alternativa para el radio mínimo es a través de un proceso de "clustering de partículas" de acuerdo al siguiente algoritmo.

El resultado del algoritmo es un cluster $\mathcal{C}$. Si se cumple que $\frac{|\mathcal{C}|}{n_s} > \delta$, se considera que el cúmulo de partículas ha convergido. Un valor de $\delta = 0{,}8$ significa que el 80 % de las partículas están centradas alrededor del *mejor global*.

Algoritmo 3 Algoritmo Clustering de Partículas

```
Cluster inicial C = ŷ
for alrededor de 5 veces do
  Calcular el centroide del cluster C:
  x̂ = (Σ_{i=1, x_i ∈ C}^{|C|} x_i) / |C|
  for ∀x_i ∈ S do
    if ‖ x_x − x̄ ‖ < ε then
      C ⟵ C ⋃ x_i
    end if
  end for
end for
```

2.4.4. Variaciones al PSO Básico

El PSO básico ha sido exitosamente aplicado a un número de problemas. Sin embargo, a veces presenta dificultades para converger a buenas solucio-

nes. Se han propuesto varias modificaciones a este algoritmo con el fin de mejorar la velocidad de convergencia y la calidad de las soluciones encontradas. Estas modificaciones incluyen acotar la velocidad, restricciones en la velocidad, diferentes formas de calcular la mejor posición global y la mejor posición personal y diferentes modelos de velocidad.

- Acotar la velocidad: Un aspecto importante para determinar la eficiencia y la exactitud de un algoritmo de optimización es el seguimiento de la exploración y la explotación. La exploración es la habilidad de un algoritmo de búsqueda para explorar diferentes regiones del espacio de búsqueda con el fin de encontrar soluciones óptimas. La explotación, por otro lado, es la habilidad de profundizar la búsqueda alrededor de un area promisoria, con el fin de refinar una solución candidata. Un buen algoritmo de optimización tratará de encontrar un balance entre estos dos objetivos contrapuestos. En PSO estos objetivos se manejan con la forma de actualizar la velocidad.

  Para controlar la exploración de la partícula la velocidad se ajusta dentro de un límite [103]. Si la velocidad de la partícula excede el máximo de velocidad especificada, la velocidad de la partícula se asigna a esa velocidad máxima. Supongamos que $V_{\text{máx},j}$ representa la velocidad máxima permitida en la dimensión j. Entonces la velocidad de la partícula se ajusta antes de actualizar la posición de la siguiente forma:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{si} \quad v'_{ij}(t+1) < V_{\text{máx},j} \\ V_{\text{máx},j} & \text{si} \quad v'_{ij}(t+1) \geq V_{\text{máx},j} \end{cases} \tag{2.15}$$

  donde v'_{ij} se calcula usando la ecuación 2.3 o 2.7. El valor de $V_{\text{máx},j}$ es muy importante ya que controla la granularidad en la búsqueda ajustando la velocidad. Grandes valores de $V_{\text{máx},j}$ facilitan la exploración global, mientras los valores pequeños mejoran la explotación local. Si $V_{\text{máx},j}$ es muy pequeña, el cúmulo puede explorar no suficientemente el espacio de búsqueda y si $V_{\text{máx},j}$ es demasiado grande, se corre el riesgo de perder buenas regiones.

  Con el fin de encontrar un balance entre la explotación y la exploración, usualmente $V_{\text{máx},j}$ se fija en una fracción del dominio del espacio de

búsqueda. Es decir,

$$V_{\text{máx},j} = \delta(x_{\text{máx},j} - x_{\text{mín},j}) \tag{2.16}$$

donde $x_{\text{máx},j}$ y $x_{\text{mín},j}$ son respectivamente los valores máximos y mínimos del dominio x en la dimensión j, y $\delta \in (0, 1]$. El valor de δ es dependiente del problema [80], [143].

- Peso de inercia: es importante para manejar la convergencia, la exploración y explotación (se denota con w). Valores grandes de w facilitan la exploración, al incrementar la diversidad. Un valor pequeño de w promueve la explotación local. Sin embargo, valores demasiado pequeños eliminan la habilidad del cúmulo en cuanto a la exploración.
  Al igual que la velocidad máxima, el valor óptimo del peso de inercia es dependiente del problema [143]. El valor w puede definirse de forma estática, aunque existen trabajos donde este valor se va cambiando dinámicamente [7], [45], [58], [116], [144], entre otros.

- Coeficiente de restricción: Un enfoque similar al peso de inercia es el coeficiente de restricción propuesto en [75] y [76], donde las velocidades están restringidas por una constante.

- Modelos de velocidad: Se ha investigado un número de variaciones al PSO, que incluyen componentes diferentes en la ecuación de velocidad, y como se determinan las mejores posiciones [53].

Para concluir este capítulo se realizará a continuación una breve descripción de metaheurísticas híbridas la cual es un área promisoria de investigación y además, es el enfoque utilizado en esta tesis como propuesta de mejora.

2.5. Hibridación de metaheurísticas

Un tema de investigación muy prometedor es la hibridación de metaheurísticas. Esta hibridación puede realizarse de diferentes formas [29]. La primera forma consiste en incluir componentes de una metaheurística en otra metaheurística. La segunda forma se refiere a sistemas denominados como

de búsqueda cooperativa que consiste en algoritmos que intercambian información de alguna manera. Y finalmente, la tercera forma corresponde a la integración de metaheurísticas con métodos de inteligencia artificial y con métodos de investigación operativa.
A continuación se describe brevemente cada una de estas hibridaciones:

- *Intercambio de componentes entre metaheurísticas.* Este tipo de hibridación es muy popular y consiste en el uso de métodos basados en trayectoria y en métodos basados en población. El poder de los métodos basados en población se basa en el concepto de recombinar soluciones para obtener nuevas soluciones. Esto permite a este tipo de algoritmos realizar pasos guiados en el espacio de búsqueda y que son usualmente más "grandes" que los pasos realizados por los algoritmos basados en trayectorias. En contraste, la fortaleza de los métodos basados en trayectoria se basa en como explotan las regiones promisorias del espacio de búsqueda. En estos métodos la búsqueda local es un componente conductor; un área promisoria del espacio de búsqueda es explotada de una forma más estructurada que en los métodos basados en población. De esta forma existe menor posibilidad de estar cerca de soluciones buenas pero pasarlas por alto como puede pasar en los métodos basados en población. Por estas razones, las metaheurísticas híbridas son a menudo exitosas ya que combinan las ventajas de los métodos basados en población con la fortaleza de los métodos basados en trayectoria.

- *Búsqueda Cooperativa.* La hibridación por búsqueda cooperativa consiste en una búsqueda realizada por diferentes algoritmos que intercambian información sobre estados, modelos, subproblemas, soluciones o características del espacio de búsqueda. En general este tipo de hibridación consiste en la ejecución paralela de algoritmos de búsqueda con un nivel de comunicación variante.

- *Metaheurísticas integradas con métodos de inteligencia artificial e investigación operativa.* Algunas posibles formas de integración pueden ser por ejemplo, las metaheurísticas integradas con métodos de árboles de búsqueda; donde se podría aplicar un método de árbol de búsqueda para

generar una solución parcial y luego aplicar una metaheurística para finalizar la búsqueda. También se podrían usar técnicas de programación con restricciones para reducir el espacio de búsqueda de un problema. Otra combinación posible podría consistir en incorporar conceptos o estrategias de otra clase de algoritmos dentro de otros algoritmos. Por ejemplo, el concepto de búsqueda tabu en árboles de búsqueda.

En caso de desear profundizar sobre algunas de estas formas de hibridación se puede consultar en [36], [48], [118], [124], entre otros.

2.6. Conclusión

Las metaheurísticas son estrategias de alto nivel que usan diferentes métodos para explorar el espacio de búsqueda. Pueden clasificarse como basadas en trayectorias y basadas en población. Las metaheurísiticas basadas en trayectoria parten de un punto y mediante la exploración del vecindario actualizan la solución actual, generando de esa manera una trayectoria. Las basadas en población, se caracterizan por trabajar con un conjunto de soluciones (población) en cada iteración.

Una metaheurística poblacional es *Particle Swarm Optimization* o PSO que se suele traducir como optimización por cúmulo de partículas. Está basada en modelos psico-sociales respecto a la influencia y al aprendizaje social. Los algoritmos basados en cúmulos de partículas han sido aplicados exitosamente en diferentes campos de investigación y en particular en problemas de optimización.

La hibridación de metaheurísticas es un área promisoria de investigación y se han obtenido exitosas aplicaciones utilizando hibridación.

Capítulo 3

Minería de Datos

En la sociedad actual se ha producido un gran crecimiento de las bases de datos y una necesidad de aumento de las capacidades de almacenamiento que no pueden resolverse por métodos manuales. Por este motivo se hacen necesarias técnicas y herramientas informáticas que ayuden, de forma automática, en el análisis de esas grandes cantidades de datos. La minería de datos (en inglés *data mining*) es una de las técnicas más utilizadas actualmente para analizar la información de las bases de datos. Se fundamenta en varias disciplinas, como la estadística, la visualización de datos, sistemas para toma de decisión, el aprendizaje automático o la computación paralela y distribuida, beneficiándose de los avances en estas tecnologías pero difiriendo de ellas en la finalidad que persigue: extraer patrones, describir tendencias, predecir comportamientos y, sobre todo, ser provechosa en la investigación computarizada que envuelve la sociedad actual con amplias bases de datos de escasa utilidad. La minería de datos no es más que una etapa, aunque la más importante, del descubrimiento de la información en bases de datos (KDD, sus respectivas siglas en inglés para Knowledge Discovery in Databases). Este proceso consta de varias fases, como se detalla más adelante, e incorpora distintas técnicas del aprendizaje automático, las bases de datos, la estadística, la inteligencia artificial y otras áreas de la informática y de la información en general. Son muchos los campos y muy variados en los que la minería de datos puede resultar muy útil y eficaz. En síntesis, se pueden establecer como objetivos prioritarios de la minería de datos los siguientes:

- Identificación de patrones significativos o relevantes.

- Procesamiento automático de grandes cantidades de datos.
- Presentación de los patrones como conocimiento adecuado para satisfacer los objetivos de los usuarios.

A continuación se detalla sintéticamente algunas ventajas del proceso de minería de datos:

- Proporciona poder de decisión a los usuarios y es capaz de medir las acciones y resultados de la mejor manera.
- Contribuye a la toma de decisiones tácticas y estratégicas.
- Supone un ahorro económico a las empresas y abre nuevas posibilidades de negocio.
- Es capaz de generar modelos prescriptivos y descriptivos.

En la siguiente sección se describen los pasos involucrados en el proceso de descubrimiento de conocimiento en bases de datos.

3.1. Pasos necesarios para la extracción de conocimiento

Como se señaló anteriormente, la minería de datos es sólo una fase de un proceso más amplio cuya finalidad es el descubrimiento de conocimiento en bases de datos (KDD). Independientemente de la técnica que se use en el proceso de extracción de datos, los pasos que deben ser seguidos son siempre los mismos:

- **Definición del problema**. En el proceso de minería de datos el primer paso consiste en definir claramente el problema que se intenta abordar.
- **Integración y recopilación de datos**. En un primer momento, hay que localizar las fuentes de información, y los datos obtenidos se llevan a un formato común para que resulten más operativos. Lo más frecuente es que los datos necesarios para llevar a cabo un proceso de KDD

pertenezcan a distintos departamentos, a diferentes organizaciones o incluso nunca hayan sido recopilados por no considerarlos interesantes. Es posible también que haya que buscar datos complementarios de informaciones oficiales. Por tanto, resulta conveniente utilizar algún método de automatización para la exploración de esos datos y encontrar posibles inconsistencias.

- **Filtrado**. Una vez homogeneizados los datos, se filtran y se rechazan los no válidos o los incorrectos, según las necesidades, o bien se corrigen o se reduce el número de variables posibles mediante clustering, redondeo, etc. Este proceso previo es necesario porque se tardaría mucho tiempo en llegar a conclusiones si se trabajara con todos los datos. El subconjunto de datos que se va a minar se denomina vista minable. Aunque se haya procesado, la mayoría de las veces se tiene una gran cantidad de datos. Un buen sistema es utilizar una muestra (sample) a partir de algunos datos.

- **Fase de minería de datos**. Una vez realizado el filtrado, se tiene que producir nuevo conocimiento que pueda ser utilizado por el usuario. Hay que obtener un modelo de conocimiento que se base en los datos recopilados y para ello hay que determinar la tarea de minería más adecuada, descriptiva o predictiva; posteriormente, elegir el tipo de modelo aunque pueden también usarse varias técnicas a la vez para generar distintos modelos teniendo en cuenta que cada técnica obliga a un preproceso diferente de los datos. Y, por último, hay que elegir el algoritmo de minería que solucione la tarea y logre el tipo de modelo que se esté buscando. Las componentes básicas de los métodos de minería son, por tanto:

  - **Lenguaje de representación del modelo**. Es muy importante que se sepan las suposiciones y restricciones en la representación empleada para construir modelos.
  - **Evaluación del modelo**. En cuanto a predictividad se basa en técnicas de validación cruzada (*Cross Validation*) en cuanto a calidad descriptiva del modelo se basan en principios como el de máxima verosimilitud (*Maximum Likelihood*) o en el principio de longitud de

descripción mínima o MDL (*Minimum Description Length*). Actualmente se están utilizando mucho curvas ROC (*Receiver Operating Characteristics*) para evaluar algoritmos.

- **Método de búsqueda**. Se puede dividir en búsqueda de parámetros y búsqueda del modelo y determinan los criterios que se siguen para encontrar los modelos (hipótesis). Algunas de las técnicas más comunes son las siguientes:
  - Árboles de decisión y reglas de clasificación que realizan cortes sobre una variable.
  - Análisis preliminar de datos usando herramientas de consultas. Éste puede ser el caso de realizar una consulta SQL sobre un conjunto de datos con el fin de rescatar algunos aspectos relevantes.
  - Redes neuronales artificiales. Son modelos predictivos, no lineales, que aprenden a través del entrenamiento.
  - Métodos de clasificación y regresiones no-lineales.
  - Métodos basados en ejemplos prototípicos que hacen aproximaciones sobre la base de ejemplos más conocidos.
  - Métodos gráficos de dependencias probabilísticas en donde se usan sobre todo redes bayesianas.
  - Modelos relacionales: programación lógica inductiva o ILP en donde la búsqueda del modelo se basa en lógica y heurística.
  - Reglas de asociación que relacionan un conjunto de pares que relacionan atributo-valor con otros pares atributo-valor.
  - Clustering: agrupan datos cuya distancia multidimensional dentro de la clase es pequeña y entre clases es grande.
- **Fase de interpretación y evaluación del modelo**. Una vez que se ha obtenido el modelo hay que proceder a su validación, comprobando que las conclusiones son válidas y satisfactorias, es decir, verificando si los resultados obtenidos son coherentes. Si se hubieran obtenido varios modelos por utilizar diferentes técnicas habría que buscar el que mejor se ajuste al tema. Así, habría que compa-

rar esos resultados con los obtenidos por métodos estadísticos y de visualización gráfica.

En la Figura 3.1 se pueden apreciar en forma resumida las distintas fases del proceso KDD que fueron explicadas anteriormente y cómo éstas se relacionan entre ellas.

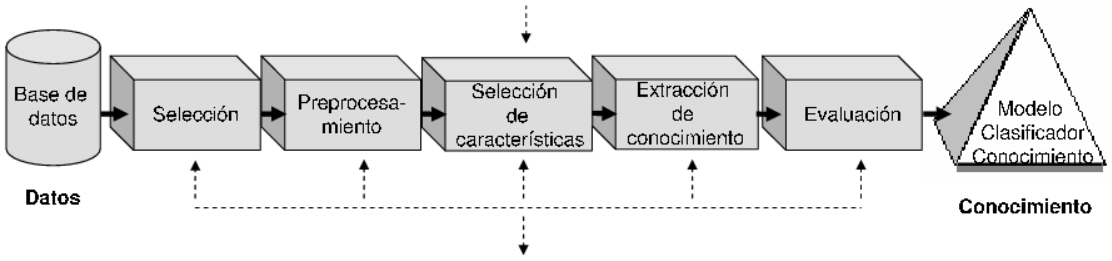


Figura 3.1: Fases del proceso de descubrimiento de conocimiento en base de datos (KDD)

3.2. Clustering

El proceso de agrupar un conjunto de objetos abstractos o físicos en clases similares recibe el nombre de clustering. Un cluster es, una colección de datos que son parecidos entre ellos y diferentes a los datos pertenecientes a otros clusters. Un cluster de datos puede ser tratado colectivamente como un único grupo en numerosas aplicaciones. Las técnicas de clustering son técnicas de clasificación no supervisada de patrones en conjuntos denominados clusters. El problema del clustering ha sido abordado por gran cantidad de disciplinas y es aplicable a una gran cantidad de contextos, lo cual refleja su utilidad como uno de los pasos en el análisis experimental de datos. El análisis de clusters es una importante actividad humana. Desde la infancia se aprende a distinguir entre perros y gatos, o entre plantas y animales, mediante una continua mejora de los esquemas de clasificación subconscientes. Las técnicas de clustering han sido ampliamente utilizadas en múltiples aplicaciones tales como reconocimiento de patrones, análisis de datos, procesado de imágenes o estudios de mercado. Gracias al clustering se pueden identificar regiones tanto pobladas como dispersas y, por consiguiente, descubrir patrones de distribución general y correlaciones interesantes entre los atributos de los datos. En el

área de los negocios, el clustering puede ayudar a descubrir distintos grupos en los hábitos de sus clientes y así, caracterizarlo en grupos basados en patrones de compra. En el ámbito de la biología puede utilizarse, por ejemplo, para derivar taxonomías animales y vegetales o descubrir genes con funcionalidades similares. De igual manera, el clustering puede ayudar a identificar áreas en las que la composición de la tierra se parece y, más concretamente, en teledetección se pueden detectar zonas quemadas, superpobladas o desérticas. En internet se puede utilizar para clasificar documentos y descubrir información relevante de ellos. El análisis de clusters se puede usar para hacerse una idea de la distribución de los datos, para observar las características de cada cluster y para centrarse en un conjunto particular de datos para futuros análisis. Alternativamente, se puede usar como un paso de preprocesado para otros algoritmos, tales como clasificación y caracterización, siempre que se opere sobre los clusters detectados. El clustering de datos es una disciplina científica incipiente sujeta a un vertiginoso desarrollo. Existen multitud de estudios y artículos esparcidos en actas de conferencias y revistas, la mayor parte de ellos en los campos de minería de datos, estadística, aprendizaje automático, biología y marketing; más aún, la mayor parte de ellos enfatizan en diversos intentos de construir técnicas específicas para cada área. Algunos ejemplos de trabajos en éstas áreas se pueden encontrar en [68], [138], [24], [6], [87], [121] y [120], entre otros. Debido a la enorme cantidad de datos contenidos en las bases de datos, el clustering se ha convertido en un tema muy activo en las investigaciones de minería de datos. Como rama de la estadística, el análisis de clusters ha sido objeto de estudio durante muchos años, centrándose principalmente en técnicas basadas en la medida de distancias. En lo referente al aprendizaje automático, el clustering suele venir referido como aprendizaje no supervisado. A diferencia de la clasificación, el clustering no depende de clases previamente definidas ni en ejemplos de entrenamientos etiquetados a priori. Por esta razón, se trata de una forma de aprendizaje por observación en vez de aprendizaje por ejemplos. En el clustering conceptual un grupo de objetos forma una clase sólo si puede ser descrito mediante un concepto, lo que difiere del clustering convencional que mide similitudes basadas en distancias geométricas. El clustering conceptual consiste en dos

componentes:

- Descubre las clases apropiadas.
- Forma descripciones para cada clase, tal y como sucede en la clasificación.

En minería de datos se vienen realizando numerosos estudios para aplicar el análisis de clusters de forma efectiva y eficiente en grandes bases de datos. En la actualidad, las líneas de investigación se centran en la escalabilidad de los métodos de clusters, en la efectividad de métodos para agrupar formas y tipos de datos complejos, técnicas para clustering de alta dimensión y, finalmente, métodos dirigidos a gestionar datos mixtos (tanto numéricos como nominales) en bases de datos grandes. El clustering es, hoy en día, un campo de investigación en el que sus aplicaciones potenciales plantean sus propios requerimientos específicos. Dichos requerimientos se pueden resumir en:

- **Escalabilidad**. Muchos algoritmos de clustering trabajan bien sobre conjuntos de datos pequeños, menos de 200 objetos, sin embargo, una gran base de datos puede contener varios millones de objetos. Aplicar clustering sobre una muestra de una gran base de datos dada puede arrojar resultados parciales. El reto, pues, es el desarrollar algoritmos de clustering que sean altamente escalables en grandes bases de datos.
- **Capacidad para tratar con diferentes tipos de atributos**. Muchos algoritmos se diseñan para clusters de datos numéricos. Sin embargo, multitud de aplicaciones pueden requerir clusters de otro tipo de datos, ya sean binarios, nominales, datos ordinales, o una mezcla de ellos.
- **Descubrir clusters de forma arbitraria**. Muchos algoritmos de clustering determinan clusters basándose en medidas de distancia de Manhattan o Euclídea. Tales algoritmos tienden a encontrar clusters esféricos con tamaños y densidades similares. Sin embargo, un cluster puede tener cualquier tipo de forma. Es por ello que es importante desarrollar algoritmos capaces de detectar clusters de forma arbitraria.
- **Requisitos mínimos para determinar los parámetros de entrada**. Muchos algoritmos requieren que los usuarios introduzcan ciertos

parámetros en el análisis de clusters (como puede ser el número de clusters deseado). El clustering es frecuentemente muy sensible a dichos parámetros, que son difíciles de determinar especialmente en los casos en los que los conjuntos de datos contienen objetos multidimensionales. Este hecho no sólo preocupa a los usuarios sino que también hace que la calidad del clustering sea difícil de controlar.

- **Capacidad para enfrentarse a datos ruidosos**. La mayor parte de las bases de datos reales contienen datos atípicos (*outliers* o datos ausentes), desconocidos o erróneos. Algunos algoritmos de clustering son sensibles a este tipo de datos lo que puede acarrear una baja calidad en los clusters obtenidos.

- **Insensibilidad al orden de los registros de entrada**. Determinados algoritmos son sensibles al orden de los datos de entrada, es decir, el mismo conjunto de datos presentados en diferente orden puede generar clusters extremadamente diferentes. Se hace evidente, pues, la necesidad de desarrollar algoritmos que sean insensibles al orden de la entrada.

- **Alta dimensionalidad**. Una base de datos puede contener varias dimensiones o atributos. Muchos algoritmos de clustering son buenos cuando manejan datos de baja dimensión (dos o tres dimensiones). El ojo humano es adecuado para medir la calidad del clustering hasta tres dimensiones. Es un reto agrupar objetos en un espacio de alta dimensión, especialmente considerando que en dicho espacio los datos pueden estar altamente esparcidos y distorsionados.

- **Clustering basado en restricciones**. Las aplicaciones del mundo real pueden necesitar realizar clustering bajo ciertos tipos de restricciones.

- **Interpretabilidad y usabilidad**. Los usuarios esperan que los resultados proporcionados por el clustering sean interpretables, comprensibles y útiles. Esto es, el clustering puede necesitar ser relacionado con interpretaciones semánticas específicas. Así, es importante estudiar cómo el objetivo buscado por una aplicación puede influir en la selección de los métodos de clustering.

3.2.1. Componentes de una tarea de clustering

Los pasos de una tarea de clustering típica se pueden resumir en cinco pasos [10], de los cuales los tres primeros son los que realizan el agrupamiento de los datos en clusters, mientras que los dos últimos se refieren a la utilización de la salida:

- Representación del patrón (opcionalmente incluyendo características de la extracción y/o selección).
- Definición de una medida de la proximidad de patrones apropiada para el dominio de los datos.
- Clustering propiamente dicho (agrupamiento de los patrones).
- Abstracción de los datos (si es necesario).
- Evaluación de la salida (si es necesario).

La *representación del patrón* se refiere al número de clases, el número de patrones disponible, y el número, tipo, y escala de las características disponibles para el algoritmo de clustering. Es posible que parte de esta información no sea controlada. La *selección de las características* es el proceso de identificar el subconjunto más apropiado de características dentro del conjunto original para utilizarlo en el proceso de agrupamiento.

La *proximidad de patrones* se mide generalmente según una función de distancia definida para pares de patrones. Existe una gran variedad de funciones de distancias que son utilizadas por diferentes autores y que serán descritas en la Sección 3.2.3 (Medidas de Distancia).

El paso de agrupamiento o clustering propiamente dicho, puede ser realizado de diversas formas. El clustering de salida puede ser "*hard*" (duro) o "*fuzzy*" (borroso o difuso). El primero de ellos realiza una partición de los datos en grupos y en el segundo cada patrón tiene un grado variable de la calidad en cada uno de los clusters de la salida. Los algoritmos de clustering *Jerárquicos* generan una serie jerarquizada de particiones basadas en

un criterio de combinación o división de clusters según su semejanza. Los algoritmos de clustering *Particionales* identifican la partición que optimiza (generalmente de manera local) un criterio de agrupamiento. Otras técnicas adicionales incluyen métodos probabilísticos y gráfico-teórico. Toda esta variedad de técnicas de clustering se detalla en la Sección 3.2.2 (técnicas de clustering).

La *abstracción de datos* es el proceso de extraer una representación simple y compacta de un conjunto de datos. Aquí, simplemente se trata de seleccionar una forma de representar los datos adecuada para el procesamiento automatizado de la información (de modo que una máquina pueda realizar la transformación posterior eficientemente), o bien, adecuada para la interpretación de la información por parte de un ser humano (de modo que la representación que se obtiene sea fácil de comprender e intuitivamente interpretable). En el contexto de clustering, una abstracción típica de los datos es la descripción compacta de cada cluster, generalmente en términos de los patrones representativos.

Es difícil evaluar si la salida de un algoritmo de clustering ha sido "buena" o "mala", es decir, si el algoritmo ha obtenido clusters válidos o útiles para el contexto concreto en el que se aplica. Además, aunque está demostrado que ciertos tipos de algoritmos de clustering obtienen mejores resultados que otros, hay que tener en cuenta la cantidad y calidad de recursos de que se dispone, así como las restricciones de tiempo y espacio establecidas. Debido a estas razones, entre otras, es posible que haya que realizar un análisis previo de la información que se desea procesar.
El análisis de validez de clusters consiste en la evaluación de la salida obtenida por el algoritmo de clustering. Este análisis utiliza a menudo un criterio específico. Hay tres tipos de estudios de la validación:

- La evaluación externa de la validez compara la estructura obtenida con una estructura *a priori*.
- La evaluación interna intenta determinar si una estructura es intrínsecamente apropiada para los datos.

- La evaluación relativa compara dos estructuras y mide la calidad relativa de ambas.

3.2.2. Técnicas de Clustering

Existe un gran número de algoritmos de clustering en la actualidad. La elección de una técnica u otra dependerá tanto del tipo de datos disponibles como del propósito de la aplicación. Si se utiliza el análisis de clustering como una herramienta descriptiva o exploratoria, es posible que se prueben distintos algoritmos sobre los mismos datos con el fin de ver cuál es el método que mejor se ajusta al problema. En general, los métodos de clusters se pueden agrupar en las siguientes categorías:

- **Métodos particionales**. Dada una base de datos con n objetos, un método particional construye k grupos de datos, donde cada partición representa a un cluster y $k \leq n$. Esto es, clasifica a los datos en k grupos que satisfacen los siguientes requisitos:

  - Cada grupo debe contener, al menos, un elemento.
  - Cada elemento debe pertenecer únicamente a un grupo.

  Nótese, que el segundo requerimiento se relaja en ciertas técnicas particionales difusas. Dado k, el número de particiones que se deben construir, los métodos particionales realizan una partición inicial. A continuación, utilizan una técnica iterativa de recolocación que intenta mejorar la partición moviendo los objetos de un grupo a otro. El criterio general para decidir si una partición es buena es que los objetos pertenecientes al mismo cluster estén cerca mientras que los objetos pertenecientes a los clusters restantes estén lejos de ellos.

  Conseguir una optimización global de un clustering basado en particiones requeriría una enumeración exhaustiva de todas las posibles particiones. Por el contrario, la mayoría de las aplicaciones adoptan una de las dos heurísticas más populares:

- Algoritmo K-means, donde cada cluster se representa por medio de los objetos en el cluster. Existen algunas variaciones de este algoritmo.
- Algoritmo K-medianas, donde cada cluster se representa por uno de los objetos situados cerca del centro del cluster.

Estas heurísticas funcionan bien para bases de datos pequeñas o medianas que tengan una forma esférica. Para encontrar clusters con formas más complejas y en bases de datos más grandes, se debe recurrir a extensiones de los mismos [12], [13],[106], [119].

- **Métodos jerárquicos.** Estos métodos crean una descomposición jerárquica del conjunto de datos objeto de estudio. Un método jerárquico puede ser clasificado como aglomerativo o divisorio:
  - Aglomerativo: comienza con cada patrón en un cluster distinto y combina sucesivamente clusters próximos hasta que se satisface un criterio preestablecido.
  - Divisorio: comienza con todos los patrones en un único cluster y se realizan particiones de éste, creando así nuevos clusters hasta satisfacer un criterio predeterminado.

  Los métodos jerárquicos presentan un pequeño inconveniente y es que una vez que un paso se realiza (unión o división de datos), éste no puede deshacerse. Esta falta de flexibilidad es tanto la clave de su éxito, ya que arroja un tiempo de computación muy bajo, como su mayor problema puesto que no es capaz de corregir errores. Si se usa primero el algoritmo aglomerativo jerárquico y después la recolocación iterativa se puede sacar más provecho de estas técnicas. Existen, de hecho, ciertos algoritmos como BIRCH [102] y CURE [114] que han sido desarrollados basándose en esta solución integrada.

- **Métodos basados en densidad**. La mayoría de los métodos particionales sólo pueden encontrar clusters de forma esférica. Para tratar este efecto, se han desarrollado técnicas de clustering basados en la noción de densidad. La idea subyacente es continuar aumentando el tamaño del

cluster hasta que la densidad (número de objetos o datos) en su vecindad exceda un determinado umbral, es decir, para cada dato perteneciente a un cluster, la vecindad de un radio dado debe contener al menos un mínimo de número de puntos. El DBSCAN es un método típicamente basado en densidad.

- **Métodos basados en grids**. En éstos, se divide cada dirección del espacio de los datos en cuadrículas (grids) lo que produce celdas hiper-cúbicas. Sobre cada celda se establece la densidad o algún tipo de estadístico que la caracterice. Luego, se utiliza esta información para establecer la semejanza entre celdas y la existencia de clusters. Así, estos métodos se hacen independientes del número de datos aun cuando algunos autores resaltan la alta dependencia de estos métodos a la definición de la dimensión de la cuadrícula [49].
- **Métodos basados en modelos**. Proponen un modelo para diferentes subconjuntos en los datos y luego miden la semejanza de los mismos para poder establecer los grupos existentes en los datos. Ejemplos de estos métodos son las redes SOM y ART2.

3.2.3. Medidas de Distancia

La medida de distancia es un aspecto clave en multitud de técnicas de minería de datos. Puesto que la semejanza entre patrones es fundamental a la hora de definir un cluster, es necesario establecer una forma de medir esta semejanza. La gran variedad de tipos de atributos hace que la medida (o medidas) de semejanza debe ser elegida cuidadosamente. Lo más común es calcular el concepto contrario, es decir, la diferencia o disimilitud entre dos patrones usando la medida de la distancia en un espacio de características. Existen varios métodos para definir la distancia entre objetos. La medida de distancia más popular es la distancia Euclídea que se define como:

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2} \qquad (3.1)$$

donde $i = (x_{i1}, x_{i2}, ..., x_{ip})$ y $j = (x_{j1}, x_{j2}, ..., x_{jp})$ son dos objetos de dimensión p.

Otra métrica ampliamente utilizada es la distancia Manhattan, definida por:

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}| \tag{3.2}$$

Tanto la distancia Euclídea como la distancia Manhattan satisfacen los siguientes requisitos matemáticos para la función de distancia:

- $d(i,j) \geq 0$. Esto es, la distancia es un número no negativo.
- $d(i,i) = 0$. Es decir, la distancia de un objeto a él mismo es cero.
- $d(i,j) = d(j,i)$. La distancia es una función simétrica.
- $d(i,j) \leq d(i,h) + d(h,j)$. Se trata de una desigualdad triangular que afirma que ir directamente desde un punto i hasta un punto j nunca es más largo que pasando por un punto intermedio h.

Finalmente, la distancia Minkowski es una generalización de las distancias Manhattan y Euclídea. Se define por:

$$d(i,j) = (|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)^{1/q} \tag{3.3}$$

donde q es un entero positivo. Representa a la distancia Manhattan cuando $q = 1$ y a la Euclídea cuando $q = 2$. El principal inconveniente que presenta la distancia de Minkowski es que la distancia de los atributos de mayor magnitud tienden a dominar al resto. Para solucionar esta desventaja se puede normalizar los valores de los atributos continuos, de forma que tomen valores dentro de un mismo rango. Por otro lado, la correlación entre los distintos atributos puede influir negativamente en el cálculo de la distancia. Para dar solución a este problema se usa la **distancia de Mahalanobis**:

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)} \tag{3.4}$$

donde x_i y x_j son vectores fila y $\sum$ es la matriz de covarianza de los patrones. La distancia asigna diferentes pesos a cada característica basándose en la varianza y en la correlación lineal de los pares. Si a cada variable se le asigna un peso de acuerdo con su importancia, la nueva **distancia euclídea ponderada** se puede calcular de la siguiente manera:

$$d(i,j) = \sqrt{w_1|x_{i1} - x_{j1}|^2 + w_2|x_{i2} - x_{j2}|^2 + \ldots + w_p|x_{ip} - x_{jp}|^2} \qquad (3.5)$$

Esto es aplicable también a las distancias de Manhattan y Minkowski.

3.2.4. Evaluación de los clusters

La evaluación de los clusters es importante y debe ser parte de cualquier análisis de clusters. Una de las motivaciones es que casi todos los algoritmos de clustering encontrarán clusters en el conjunto de datos, aún cuando este conjunto no contenga clusters naturales en su estructura. Existen diferentes aspectos a tener en cuenta para la validación de clusters:

1. Determinar cómo tienden a agruparse los datos del conjunto, es decir, distinguir si existe en los datos una estructura no aleatoria.
2. Determinar el número correcto de clusters.
3. Evaluar cuán bien los resultados del análisis de un cluster se adecúan a los datos sin tener referencia de información externa.
4. Comparar los resultados de un análisis de cluster con resultados externos conocidos.
5. Comparar dos conjuntos de clusters para determinar cuál es el mejor.

Los items 1,2 y 3 no utilizan información externa, son técnicas no-supervisadas, mientras que el item 4 requiere información externa. El item 5 puede realizarse de ambas formas, supervisada o no-supervisada.
Las medidas de evaluación que se aplican para juzgar varios aspectos de la validez de los clusters se clasifican generalmente en:

- No-supervisada: Mide la bondad de la estructura del clustering sin referirse a información externa.
- Supervisada: mide el grado de coincidencia de la estructura de cluster descubierta por el algoritmo de clustering con alguna estructura externa.

A continuación se describirán detalles específicos concernientes a la validación de clusters y particularmente a la evaluación de clusters no-supervisada.

3.2.5. Evaluación no-supervisada de clusters

Las medidas no-supervisadas de la validez de un cluster a menudo se dividen en dos clases: medida de cohesión de los clusters (compacto, apretado), que determina lo cercanos que están los objetos dentro del cluster, y la medida de separación (aislamiento), que determina lo distinto y bien separado que está un cluster con respecto a los otros. Las medidas no-supervisadas a menudo se llaman índices internos debido a que usan sólo información presente en el conjunto de datos.
La cohesión de un cluster puede definirse como la suma de las proximidades con respecto al centroide del cluster. La cohesión está dada en la ecuación 3.6 y la medida para la separación se muestra en la ecuación 3.7. Notar que en estas ecuaciones la medida de distancia utilizada es la distancia Euclideana.

- Distancia Intra-Cluster:

$$Intra\_C = \frac{\sum_{i=1}^{K}(\sum_{x,t \in C_i} dist(x,t)/m_i)}{K} \tag{3.6}$$

  donde C_i representa el cluster i, K es la cantidad de clusters y m_i es la cantidad de elementos pertenecientes al cluster i.

- Distancia Inter-Cluster:

$$Inter\_C = \frac{\sum_{i=1}^{K-1}\sum_{j=i+1}^{K} dist(c_i, c_j)}{\sum_{i=1}^{K-1} i} \tag{3.7}$$

  donde c_i representa al elemento central (centroide) del cluster i, y c_j representa el centroide del cluster j y K es la cantidad de clusters.

Algunos de los índices disponibles en la literatura son por ejemplo, índice de Silhouette [60], índice de Dunn's (D) [65], índice de Calinski-Harabasz [11], índice de Davis-Bouldin (DB)[27], índice PBM [84], la medida CS [18]. De igual forma para clustering difusos los índices más prominentes son Coeficiente de Partición [63], índice Xie-Beni[9], etc. Todos estos índices podrían

ser usados como una función de optimización en cualquier algoritmo de optimización para encontrar los clusters apropiados en un conjunto de datos. En las siguientes secciones, se describirán algunos de estos índices de validación, aplicables a problemas de clustering.

3.2.6. Coeficiente Silhouette

Una medida interesante vinculada a los objetos en sí es la denominada Silhouette, la cual combina cohesión y separación. A continuación se explican los pasos de cómo calcular el coeficiente de Silhouette para un punto individual:

- Para el *i-ésimo* objeto, calcular su distancia promedio a todos los otros objetos en su propio cluster (este valor se denota a_i).
- Para el *i-ésimo* objeto y todo cluster que no contenga al objeto i, calcular la distancia promedio a todos los objetos de un cluster dado. Encontrar el mínimo de los valores con respecto a todos los clusters (este valor se denota b_i).
- Para el *i-ésimo* objeto, el coeficiente Silhouette se calcula como $s_i = (b_i - a_i/max(a_i, b_i))$

donde $-1 \leq s_i \leq 1$. No es deseable un valor negativo debido a que corresponde a un caso en el que a_i, la distancia promedio de los puntos en el cluster, es mayor que b_i, la menor distancia promedio a los puntos en otro cluster. Se espera que el coeficiente silhouette tenga un valor positivo ($a_i < b_i$), y para a_i que esté lo más cercano a 0 posible, ya que el coeficiente asume su máximo valor de 1 cuando $a_i = 0$.

Para poder entender de forma más clara la salida de esta función, se hace uso de la Figura 3.2 donde se muestran los valores de Silhouette de objetos que pertenecen a diferentes clusters, es decir que los objetos que pertenecen a un mismo cluster, aparecen juntos formando un bloque. El valor de Silhouette para cada objeto muestra cuanto se parece un objeto al resto de los objetos de su propio cluster y a su vez comparado con los objetos de los otros clusters,

tomando valores dentro del rango -1 y 1. Cuando un valor de Silhouette es cercano a 1, se puede inferir que ese objeto ha sido asignado al cluster apropiado. Por el contrario, cuando ese valor es cercano a -1 , se puede inferir que ese objeto ha sido asignado a un cluster incorrecto.

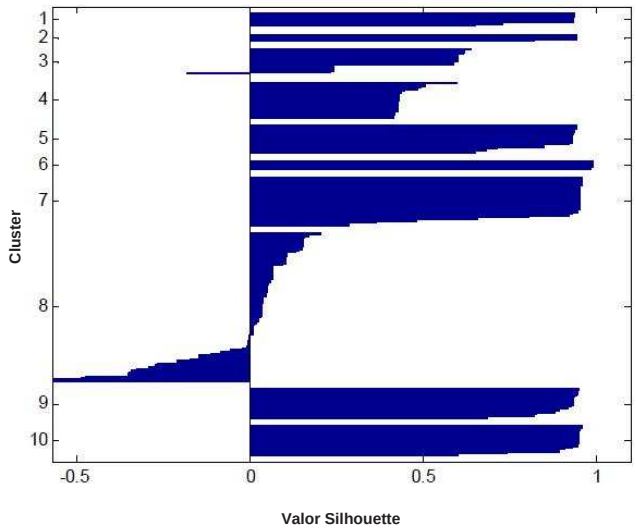


Figura 3.2: Curvas Silhouette obtenidas para 10 clusters

Se puede calcular el coeficiente Silhouette promedio de un cluster simplemente tomando el promedio de los coeficientes silhouette de los puntos que pertenecen al cluster. Una medida global de la bondad de un clustering se puede obtener calculando el coeficiente silhouette promedio de todos los puntos.
Para cualquier partición $U \leftrightarrow C : C_1 \bigcup ... \bigcup C_i \bigcup ... C_K$, el valor de *Silhouette Global* GS_u puede usarse como un índice de validez efectivo para U.

$$GS_u = \frac{1}{K} \sum_{j=1}^{c} S_j \tag{3.8}$$

La Ecuación 3.8 se puede aplicar para estimar el número de clusters para U. En este caso la partición con el máximo S_u se toma como partición óptima.

3.2.7. Indice Dunn's (D)

Este índice identifica al conjunto de clusters que es compacto y bien separado. Para cualquier partición $U \leftrightarrow C : C_1 \bigcup ... \bigcup C_i \bigcup ... C_K$ donde C_i representa el $i - esimo$ cluster de esa partición, el índice de Dunn's, D, se

define como:

$$D(U) = \min_{1\leq i\leq K}\left\{\min_{1\leq j\leq K j\neq i}\left\{\frac{\delta(C_i,C_j)}{\max_{1\leq t\leq K}\{\Delta(C_t)\}}\right\}\right\} \tag{3.9}$$

donde $\delta(C_i, C_j)$ de define como la distancia entre los clusters C_i y C_j (distancia Inter-cluster) y $\Delta(C_t)$ representa la distancia intra-cluster del cluster C_t y K es el número. El principal objetivo de esta medida es maximizar las distancias inter-cluster mientras se minimizan las distancias intra-cluster. Por lo tanto valores grandes de D representan buenos clusters. Es decir que el número de clusters que maximiza D se toma como el número óptimo de clusters, K.

3.2.8. Indice Davis-Bouldin (DB)

De forma similar al índice de Dunn's, el índice de Davis-Bouldin intenta identificar el conjunto de clusters que sea compacto y bien separado. El índice de validación de Davis-Bouldin se define de la siguiente forma:

$$DB(U) = \frac{1}{K}\sum_{i=1}^{K}\max_{i\neq j}\left\{\frac{\Delta(C_i)+\Delta(C_j)}{\delta(C_i,C_j)}\right\} \tag{3.10}$$

donde $j \in \{1,\ldots,K\}$; U, $\delta(C_i, C_j)$,$\Delta(C_i)$,$\Delta(C_j)$ y K se definen en la Ecuación 3.9.
Valores pequeños de DB corresponden a clusters que son compactos y están bien separados. Por lo tanto, la configuración de clusters que minimice DB se toma como el número de clusters óptimo, K es el número de clusters.

3.3. Algoritmo K-means

El algoritmo de K-means fue propuesto por MacQueen en el año 1968 [55] desde su desarrollo ha sido muy popular y aparece en varios libros de métodos multivariados ([3], [54]), análisis de clustering ([23], [62], [95]), aprendizaje estadístico ([89]). Este algoritmo toma el parámetro de entrada k y particiona el conjunto de n datos en los k clusters de tal manera que la similitud intra-cluster es elevada mientras que la inter-cluster es baja. Dicha similitud se mide en relación al valor medio de los objetos en el cluster, lo que puede

ser visto como si fuera su centro de gravedad. El algoritmo procede como sigue. En primer lugar, selecciona aleatoriamente k objetos haciendo que éstos representen el centro del cluster (centroide). Cada uno de los objetos restantes se va asignando al cluster que sea más similar basándose en la distancia del objeto a la media del cluster. Entonces calcula la nueva media de cada cluster y el proceso sigue iterando hasta que se consigue la convergencia, es decir, se minimiza el error cuadrático medio (el Algoritmo 4 describe formalmente este proceso).

Algoritmo 4 Algoritmo K-means

1: Selecciona k centroides iniciales
2: **repeat**
3: Formar k clusters asignando cada punto a su cluster más cercano
4: Recalcular el centroide de cada cluster
5: **until** Centroides sin cambio

A continuación veremos con más detalle algunos pasos de este algoritmo:

- **Asignar los puntos al centroide más cercano**. Para asignar un punto al centroide más cercano, se necesita una medida de proximidad que cuantifique la noción de "*más cercano*" para el conjunto específico de datos que se está considerando.
  Generalmente las medidas de similitud usadas en K-means son relativamente simples ya que el algoritmo calcula repetidamente la similitud de cada punto a cada centroide.

- **Centroides y Funciones Objetivo**. El centroide puede variar dependiendo de la medida de proximidad para los datos y del objetivo del clustering. El objetivo del clustering se expresa generalmente por una función objetivo que depende de la proximidad entre los puntos o la proximidad a los clusters. Si se consideran datos cuya medida de proximidad es la distancia Euclideana, para una función objetivo que mide la calidad del agrupamiento, se puede usar la suma del error cuadrado (*sum of the squared error* - SSE) que se la conoce también como dispersión. En otras palabras, se calcula el error de cada punto de dato, es decir, su distancia Euclideana al centroide más cercano, y luego se calcula la

suma total de los errores cuadrados. Usando la notación en la Tabla 3.1, SSE se puede definir como en la Ecuación 3.11:

Tabla 3.1: Tabla de notación para SSE.

Símbolo	Descripción
x	Un objeto.
C_i	El $i-esimo$ cluster.
c_i	El centroide del cluster C_i.
c	El centroide de todos los puntos.
m_i	El número de objetos en el $i-esimo$ cluster
m	El número de objetos en el conjunto de datos.
K	El número de clusters.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} (dist(c_i, x)^2) \tag{3.11}$$

donde *dist* es la distancia Euclideana estándar entre dos objetos.
El algoritmo de K-means trata de minimizar SSE.

- **Elegir los centroides iniciales**. En investigaciones anteriores se ha encontrado que el algoritmo de K-means no necesariamente provee un óptimo global y dependiendo de los valores de comienzo que se usen el algoritmo finaliza en un óptimo local. Para evitar los óptimos locales, algunos autores [32], [62] sugieren ejecutar el algoritmo de K-means varias veces, con distintos valores de comienzo, aceptando la mejor solución (en términos de SSE-Suma de Error Cuadrado); sin embargo, se ha demostrado que el número de óptimos locales para conjuntos de datos de un tamaño moderado pueden alcanzar los miles [26] indicando que los resultados de estudios que usan un número aleatorio, pequeño de recomienzos pueden ser engañosos [131].

Cuando los centroides son inicializados aleatoriamente, diferentes corridas de K-means generalmente producen distintas sumas de errores cuadrados. Elegir los centroides iniciales apropiados es un paso clave en el algoritmo de K-means. Un enfoque común es inicializar aleatoriamente

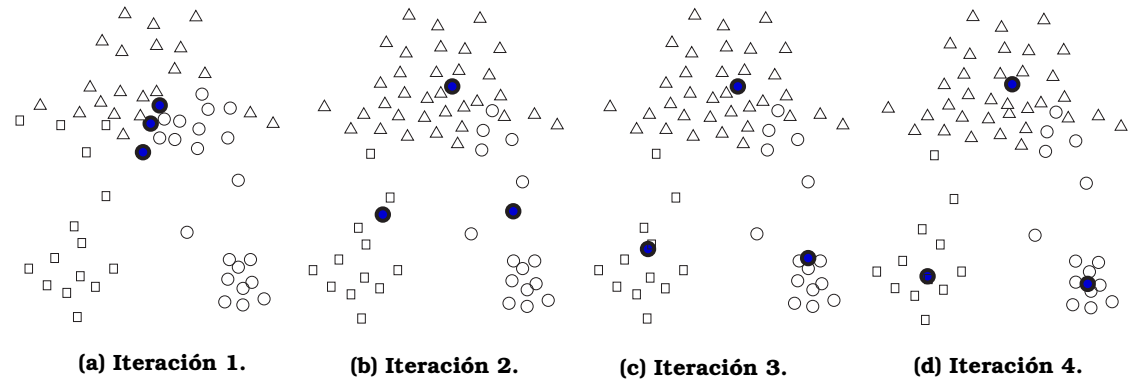


Figura 3.3: Uso de K-means para encontrar tres clusters

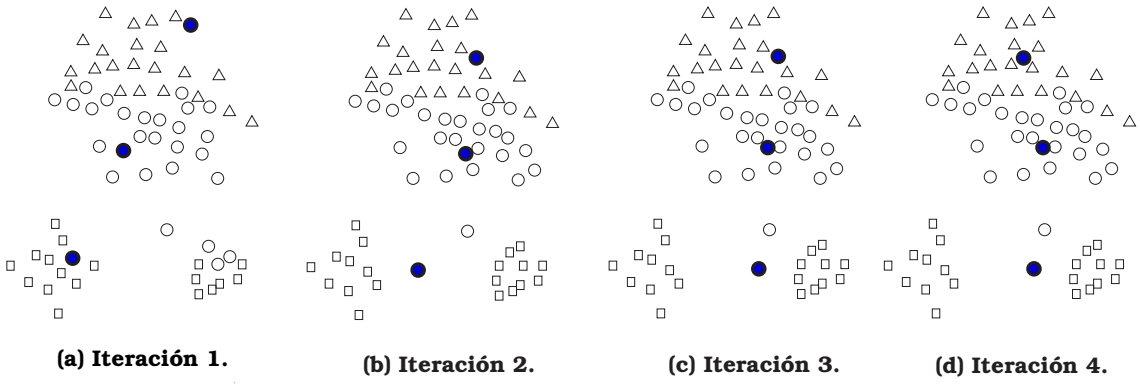


Figura 3.4: K-means con centroides de comienzo poco adecuados

los centroides pero los clusters que resultan generalmente son bastante pobres. A continuación en las Figuras 3.3 y 3.4 se muestran los clusters resultantes de dos elecciones particulares de centroides iniciales. Se puede observar que aunque en la Figura 3.3 inicialmente los centroides se encuentran ubicados en un único cluster se obtienen los clusters correctos. En cambio en la Figura 3.4 aún cuando los centroides iniciales parecen estar mejor distribuidos, al final de la iteraciones no se encuentran los clusters correctos.

Una técnica que se utiliza comúnmente para este problema de elección de los centroides iniciales es realizar múltiples corridas del algoritmo, cada una con un conjunto de centroides iniciales diferentes elegidos aleatoriamente y luego seleccionar el conjunto de clusters que minimice la SSE. Aunque esta estrategia es simple puede no obtener buenos resultados pues está directamente relacionada con el conjunto de datos y con el

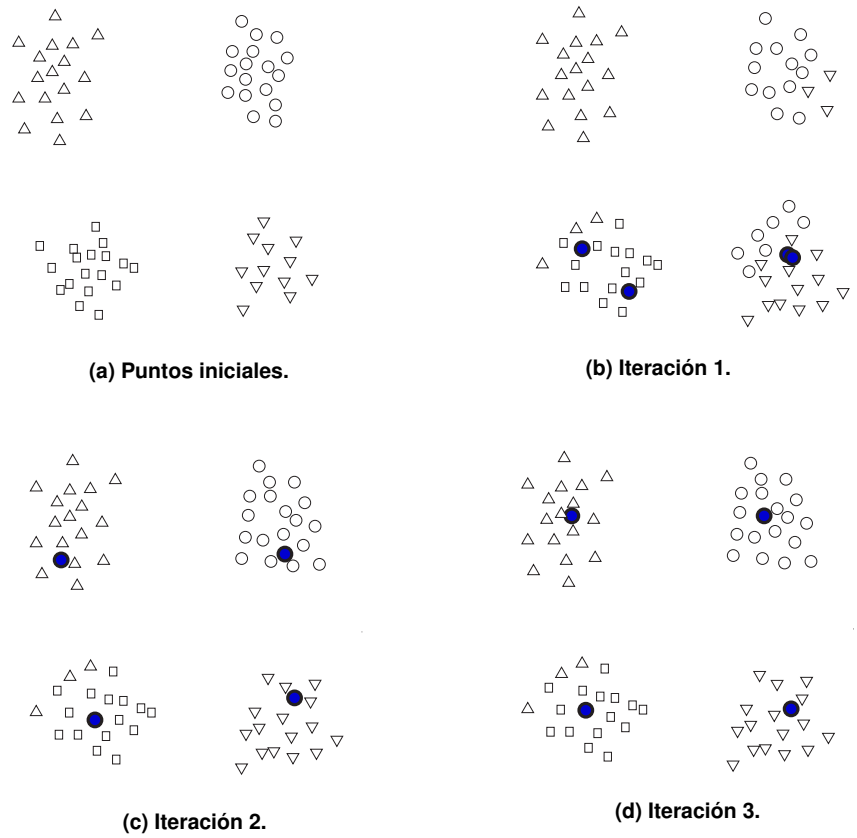


Figura 3.5: Dos pares de clusters con un par de centroides iniciales dentro de cada par de cluster

número de clusters que se definan. En [93] se demuestra esto utilizando el conjunto de datos de la Figura 3.5(a) donde se muestran los datos de dos pares de clusters donde los clusters en cada par (arriba - abajo) están más cercanos uno del otro que a los clusters en el otro par. Las Figuras 3.5(b-d) muestran que si se comienza con dos centroides iniciales por par de cluster, aún cuando ambos centroides se encuentren en un sólo cluster, los centroides se redistribuirán encontrando los centroides correctos. Sin embargo, la Figura 3.6 muestra que si un par de clusters tiene sólo un único centroide inicial y el otro par tiene tres entonces, se combinarán dos de los clusters correctos y uno de los clusters correcto se dividirá. Es importante resaltar que se obtienen los clusters correctos aún cuando dos centroides iniciales estén en cualquier par de clusters, pues se redistribuirán en cada uno de los clusters. Desafortunadamente, cuando se trata de un número de cluster grande, se aumenta la posibilidad de que al menos un par de cluster tenga un sólo centroide inicial.

En este caso, debido a que los pares de cluster están más separados que los clusters dentro de un par, el algoritmo de K-means no distribuirá los centroides entre los pares de clusters y por lo tanto encontrará un único mínimo local.

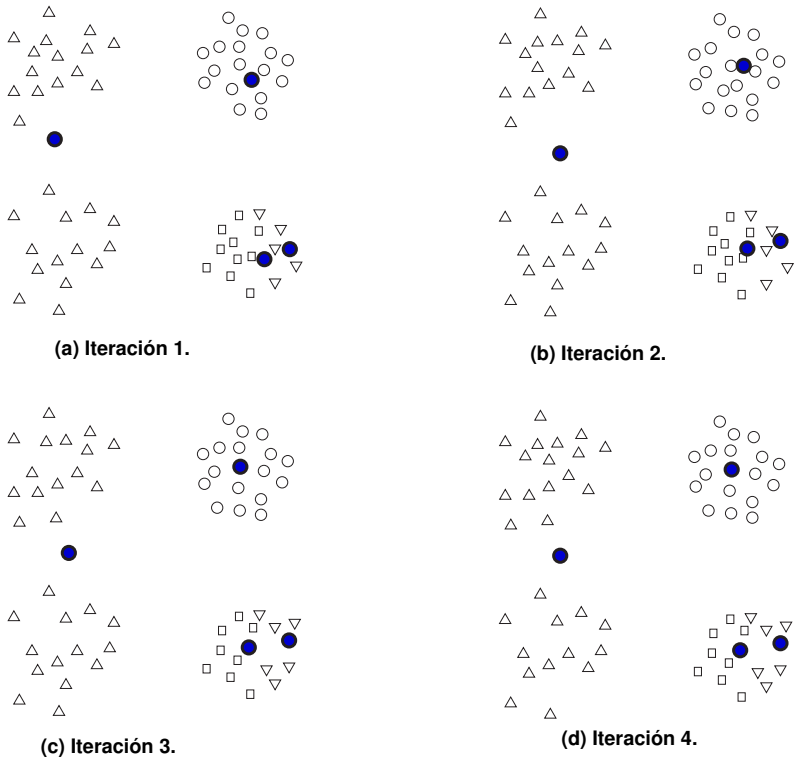


Figura 3.6: Dos pares de clusters con mas o menos de dos centroides iniciales dentro de cada par de cluster

Podemos observar que usar centroides iniciales seleccionados aleatoriamente puede obtener centroides no adecuados, aún cuando se realicen repetidas corridas del algoritmo, es por ello que existen otras técnicas de inicialización. Un enfoque efectivo es tomar una muestra de puntos, clusters y luego usar una técnica de clustering jerárquico. Se extraen K clusters del clustering jerárquico y los centroides de esos clusters son usados como centroides iniciales. Esta estrategia a menudo funciona bien si:

- La muestra es relativamente pequeña pues el clustering jerárquico es costoso.
- K es relativamente pequeño comparado con el tamaño de la muestra.

Otra estrategia consiste en seleccionar el primer punto de forma aleatoria o tomar el centroide de todos los puntos. Luego, para cada sucesivo centroide inicial seleccionar el punto que está más lejos de cualquier otro centroide inicial ya seleccionado. De esta manera, se obtiene un conjunto inicial de centroides que garantizan no sólo ser elegidos aleatoriamente sino también estar bien separados. Desafortunadamente este enfoque puede seleccionar *outliers* en lugar de puntos en regiones densas (clusters). Además, es un enfoque costoso y por eso suele aplicarse a una muestra de puntos. Por esta razón se emplean a menudo otras técnicas para la inicialización de los centroides. Para una mayor profundización dirigirse a [93].

- **Tiempo y espacio**. El requerimiento de espacio de K-means es pequeño ya que solamente se almacenan los puntos y los centroides. Específicamente el espacio requerido es $O((m + k)n)$, donde m es el número de puntos y n es el número de atributos. K-means también requiere poco tiempo, básicamente lineal $O(I * k * m * n)$, donde I es el número de iteraciones requeridas para la convergencia.

3.3.1. Consideraciones importantes sobre K-means

- Manejo de Clusters vacíos: uno de los problemas del algoritmo de K-means es que puede encontrar clusters vacíos si no se le asigna ningún punto en la etapa de asignación. Si esto sucede se necesita una estrategia para elegir un centroide de reemplazo, ya que de otro modo, el error cuadrado será muy grande. Esto elimina el punto que más contribuye al error cuadrado total. Otro enfoque es elegir el centroide de reemplazo del cluster que tenga el SSE más grande. Si hay varios clusters vacíos, este proceso puede repetirse varias veces.

- Outliers: Cuando se utiliza el criterio de error cuadrado los outliers pueden influir excesivamente en los clusters que se encuentran. En particular, cuando están presentes los outliers los centroides de los clusters resultantes pueden no ser tan representativos como deberían y por ello la SSE será alta. Por esta razón, es útil descubrir y eliminar con antelación

a los outliers. Cuando el clustering se utiliza para compresión de datos, todo dato debe ser agrupado y por ejemplo algunos outliers pueden ser aparentes y tratarse de datos realmente interesantes para el problema. Existen numerosas técnicas para identificar outliers. Si se usan enfoques que remueven los outliers antes del clustering evitamos puntos de clustering que podrían no agrupar bien. Los outliers también pueden identificarse en una etapa de post procesamiento. Por ejemplo, se puede mantener la SSE de cada punto y eliminar aquellos puntos que usualmente tienen contribuciones altas, especialmente luego de múltiples corridas. También se pueden eliminar clusters pequeños ya que ellos frecuentemente representan grupos de outliers.

- Reducir SSE con postprocesamiento: Una forma de reducir SSE es encontrar más clusters, es decir, usar un K grande. Sin embargo, en muchas situaciones se pretende mejorar la SSE, sin incrementar el número de clusters. Esto a menudo es posible porque K-means generalmente converge a un óptimo local mínimo. Se han usado varias técnicas para "arreglar" los clusters resultantes a fin de encontrar un cluster que minimice SSE. La estrategia es enfocarse en clusters individuales ya que SSE total es simplemente la suma de los SSE producidas por cada cluster. Se puede cambiar la SSE total realizando varias operaciones sobre los clusters, como partirlos o mezclarlos. Un enfoque comúnmente usado es intercalar las fases de partición y de mezcla de los clusters. Durante la fase de partición los clusters se dividen mientras que durante la fase de mezcla los clusters se combinan. De esta forma, se intenta generar soluciones con el número de clusters deseado.

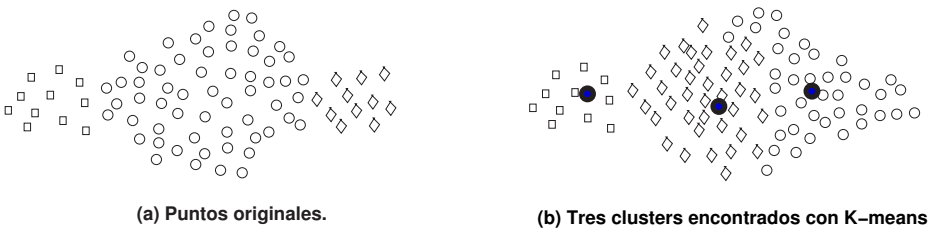


Figura 3.7: K-means con clusters de diferentes tamaños

- Clusters de diferentes tipos: K-means tiene limitaciones con respecto a

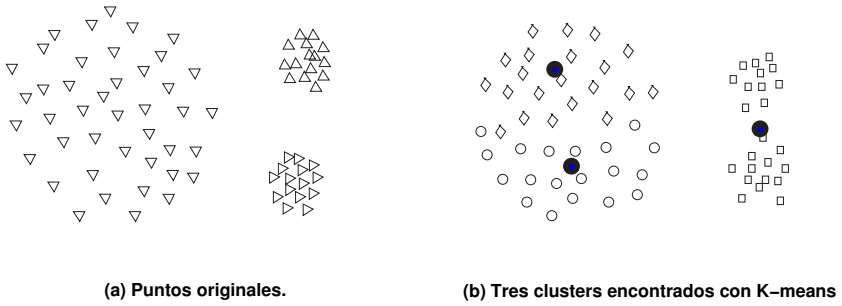


Figura 3.8: K-means con clusters de diferentes densidades

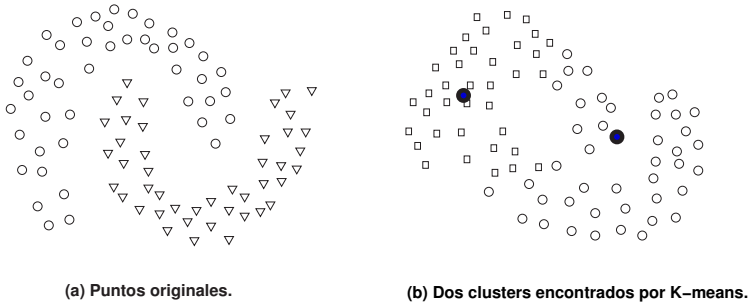


Figura 3.9: K-means con clusters no-globulares

encontrar clusters de diferentes tipos. En particular presenta dificultades cuando los clusters tienen formas no-esféricas, tamaños y densidades ampliamente diferentes. Esto se ilustra en las Figuras 3.7, 3.8 y 3.9. En la Figura 3.7 K-means no encuentra los clusters "naturales" porque uno de los clusters es mucho más grande que los otros dos y por lo tanto se parte el cluster más grande y mientras que uno de los clusters más pequeño se combina con una parte del cluster más grande. En la Figura 3.8 K-means no encuentra los tres clusters "naturales" debido a que dos clusters pequeños son más densos que el más grande. Finalmente, la Figura 3.9 muestra que K-means encuentra dos clusters que mezclan partes de los dos clusters naturales debido a que la forma de los clusters es no-globular.

3.3.2. Fortalezas y Debilidades

El algoritmo de K-means es simple y se puede usar en una amplia variedad de conjuntos de datos. Sin embargo, no sirve para todos los conjuntos de datos. No puede manejar clusters no-globulares o clusters con diferentes ta-

maños y densidades, aunque puede encontrar típicamente subclusters puros si se especifica un número importante de clusters. K-means también presenta inconvenientes con el clustering cuando los datos contienen *outliers*, datos atípicos y/o erróneos. En estas situaciones puede ser de ayuda la detección y remoción de *outliers*. Finalmente, K-means es restrictivo a datos donde existe una noción de centro (centroide). Una técnica de clustering relacionada a K-means llamada K-medoid, no tiene esta restricción, pero es mucho más costosa.

Se han realizado varias mejoras al algoritmo de K-means estándar relacionadas con varios aspectos asociados a los pasos de: (1) Inicialización, (2) Clasificación, (3) Cálculo de centroides y (4) Condición de convergencia. Con respecto a la mejora de estos aspectos se puede mencionar trabajos en [67] y [79]. En cuanto al segundo paso, se han definido varias medidas de afinidad para los objetos de los grupos como en [41] y [83]; en cuanto a los dos últimos pasos se puede mencionar [28] y [59].

Por otra parte, otros autores han hecho contribuciones importantes en relación a la implementación computacional del algoritmo, presentando diferentes variantes como en [4], [25] y [125], entre otras.

3.4. Conclusión

La minería de datos es una etapa importante del proceso de KDD. Es una técnica que permite buscar información dentro de un conjunto de datos con la finalidad de extraer información nueva y útil que se encuentra oculta en grandes volúmenes de datos.

El clustering es una tarea dentro de la minería de datos y consiste en agrupar un conjunto de objetos abstractos o físicos en clases similares denominadas clusters. Existe un gran número de algoritmos de clustering y se pueden agrupar en métodos particionales, métodos jerárquicos, métodos basados en densidad, métodos basados en grids y basados en modelos.

Un algoritmo de clustering particional es el conocido algoritmo de K-means. Este algoritmo es fácil de implementar y muy eficiente en términos de tiempo de ejecución. Sin embargo, es sensible a la selección de la partición inicial y

puede converger a óptimos locales. Es importante resaltar que existen en la literatura varios trabajos que presentan variantes y mejoras a este algoritmo, como en [4], [25], [28], [41], [59], [67], [83], [79] y [125], entre otros.

Capítulo 4

Clustering con Metaheurísticas

El problema de clustering de datos ha sido tratado desde diversos campos de conocimientos como estadística (análisis multivariable), teoría de grafos [21], redes neuronales [56], [88], [126], computación evolutiva [31], [117], clustering usando inteligencia colectiva [66], entre otros. A pesar de ser antiguo, el problema de clustering ha llamado la atención de investigadores de diversos campos de conocimiento, y se hace casi imposible revisar la enorme cantidad y multifacética literatura sobre clustering. Las técnicas de clustering basado en metaheurísticas han mostrado mejorar varios métodos clásicos de partición de un conjunto de datos. Por esta razón, este capítulo se limita a clustering basado en poblaciones y en particular basado en PSO que es la metaheurística con la cual se intenta proponer una mejora.

4.1. Método de Clustering basado en Algoritmos Evolutivos

A continuación se describen los más prominentes algoritmos de clustering basados en algoritmos evolutivos como Algoritmos Genéticos (AG), Estrategias Evolutivas (EE) y Programación Evolutiva (PE).

4.1.1. Algoritmos de Clustering Particional basados en Algoritmos Genéticos

La idea principal de los algoritmos de clustering basados en AGs es crear una población de soluciones candidatas a un problema de optimización, el

cual se refina iterativamente por la alteración y la selección de buenas soluciones para la siguiente iteración. Las soluciones candidatas se seleccionan de acuerdo a una función de aptitud (fitness) la que evalúa su calidad con respecto al problema de optimización. En el caso de AGs la alteración consiste en la mutación para explorar soluciones en el vecindario local existente y en la recombinación (crossover) para recombinar información entre distintas soluciones candidatas. Una ventaja importante de estos algoritmos es su habilidad de hacer frente a óptimos locales, manteniendo, recombinando y comparando simultáneamente varias soluciones candidatas. Contrariamente a esta situación, las heurísticas de búsqueda local, como el algoritmo de enfriamiento simulado (simulated annealing) [115], refina solamente una única solución candidata y son débiles para manejar óptimos locales. La búsqueda local determinística, que se usa en algoritmos como K-means, siempre converge al óptimo local más cercano desde la posición de comienzo de la búsqueda. El primer enfoque que usó una codificación directa asociando objeto-cluster en clustering con AG fue desarrollado por Raghavan y Birchand [136]. La idea de este enfoque es usar una codificación directa de n objetos a los k clusters, es decir que cada posible solución contiene n genes con valores enteros en el intervalo $[1, k]$. Por ejemplo para $n = 6$ y $k = 4$ la codificación "114332" asigna el primer y segundo objeto al cluster 1, el tercer objeto al cluster 4, el cuarto y quinto objeto al cluster 3 y sexto objeto al cluster 2 y por lo tanto se han identificado los siguientes clusters ({1,2},{3},{4,5},{6}). Con esta representación del problema el AG trata de encontrar la partición óptima de acuerdo a la función de *fitness* que mide cuan buena es la partición. Se ha demostrado en el análisis de bases de datos reales y artificiales que este algoritmo mejora a K-means (ver [17]). Sin embargo, esta representación tiene la desventaja de la redundancia y es por ello que en Falkenauer et al. [31] tratan este problema de una manera elegante. Además, de usar la codificación de n genes que representan cada asociación objeto-cluster, incorporan un grupo de etiquetas como genes adicionales a la codificación y aplican operadores evolutivos en forma *ad hoc* sobre ellos.

Otro ejemplo de clustering particional con AGs es el propuesto en los trabajos de Bandyopadhyay et al. [107],[108], [109] donde determinan hiperplanos

como límites de decisión. Codifican la ubicación y orientación de un conjunto de hiperplanos con una representación de genes de longitud flexible. Además, de minimizar el número de objetos incorrectamente clasificados, este enfoque trata de minimizar el número de hiperplanos requeridos.
Otro enfoque que utiliza AG en clustering particional codifica una variable representativa (generalmente un centroide) y un conjunto de parámetros para describir la extensión y la forma de la varianza para cada cluster. Srikanth et al. [99], proponen un enfoque que codifica el centro, la extensión y orientación de una elipsoide para cada cluster. Por otra parte, muchos autores proponen centroides, baricentros o medios como puntos representantes a fin de asignar cada objeto a un cluster específico (por ejemplo [110], [117], [130], [145]). La idea es determinar un punto representante de cada cluster y asignar cada objeto a su cluster más cercano, esa cercanía se refiere a una medida de distancia como por ejemplo la distancia Euclídea. El *fitness* de una posible solución se calcula según como se adecúa la partición identificada de acuerdo a un criterio estadístico, como por ejemplo el criterio de radio de la varianza. Muchos estudios han mostrado que este enfoque es más robusto en la convergencia a la partición óptima que los algoritmos de partición clásicos ([110], [117], [130], [145]).
Finalmente, algunos autores desarrollan algoritmos de clustering híbridos, que combinan técnicas clásicas de clustering con AGs. Por ejemplo, Krishna and Murty [71], utilizando un AG con una codificación directa de asociaciones objeto-cluster como en [136], pero aplican K-means para determinar la cantidad de soluciones candidatas. Para esto cada solución candidata del AG se usa como punto de comienzo para cada corrida de K-means. La calidad de la solución encontrada por K-means luego se usa como función de *fitness* de la solución candidata del AG.

- **Algoritmos de Clustering Basados en EE y PE**
  Se utiliza una EE en el trabajo presentado por Lee y Antonsson [22] donde se agrupa automáticamente un conjunto de datos sin tener conocimiento *a priori* sobre la cantidad de clusters. La EE implementa individuos de longitud variable que buscan simultáneamente los centroides de los clusters y el número de clusters. En este enfoque cada

individuo representa un conjunto de centroides. La longitud de cada individuo se elige aleatoriamente de un rango especificado por el usuario para el número de clusters. Se aplica mutación y crossover de dos puntos. Se usa una versión modificada de la cuantización del error medio MSE (*mean-square-error*) como función de *fitness*.

El enfoque propuesto por Sarkar et al. [82] clasifica dinámicamente un conjunto de datos usando PE donde se optimizan simultáneamente dos funciones de fitness: una para el número óptimo de clusters y la otra para la identificación apropiada de los centroides de los clusters. Este algoritmo determina el número de clusters óptimo y el número de centroides óptimos de tal forma de escapar a soluciones locales. Otra ventaja es que el agrupamiento es independiente de la selección inicial de los centroides. Sin embargo, el método propuesto es aplicable para clustering donde los clusters no están superpuestos y tienen una forma natural esférica.

4.2. Algoritmos de Clustering usando Inteligencia Colectiva

En esta sección se presenta el estado del arte en algoritmos basados en inteligencia colectiva, específicamente algoritmos ACO y PSO.

4.2.1. Algoritmos de Clustering basados en colonia de hormigas

Las colonias de hormigas han provisto un medio para formular poderosas heurísticas inspiradas en la naturaleza para resolver problemas de clustering. Entre otros desplazamientos sociales los investigadores han simulado la forma en que las hormigas trabajan cooperativamente en tareas de agrupamiento de los cuerpos de otras hormigas muertas con el fin de mantener el hormiguero limpio [30]. En la naturaleza muchas especies organizan su cementerio. Si los cuerpos, o más precisamente, la suficiente cantidad de partes de cuerpos están distribuidas aleatoriamente en el espacio al comienzo del experimento, las trabajadoras forman clusters de cementerios en pocas horas, siguiendo el comportamiento similar a la agregación. Si el espacio experimental no es lo suficientemente grande, o si contiene heterogeneidades espaciales, los clusters

se formarán a lo largo de los bordes del espacio experimental, o más generalmente, siguiendo esas heterogeneidades. El mecanismo básico subyacente a este fenómeno de agregación es la atracción entre los cuerpos muertos y las hormigas trabajadoras: un cluster de items pequeño crece al atraer trabajadoras a depositar más items. Es este feedback positivo y auto-catalítico que lleva a la formación de clusters más y más grandes. En este caso, es por consiguiente la distribución de los clusters en el ambiente que juega el rol de variable "*stigmergic*" o variable de ambiente. Puede observarse, que con tiempo las hormigas tienden a agrupar todos los cuerpos muertos en una región específica del ambiente, formando así, pilas de cuerpos. Este comportamiento de ordenamiento de larvas y limpieza de cuerpos, de las hormigas fue modelado por Deneubourg et al. para realizar ciertas tareas en robótica [66]. Esto inspiró al algoritmo de clustering basado en hormigas [51]. Lumer y Faielo modifican este algoritmo a fin de adaptarlo mejor al clustering de datos [33]. Esto dió origen al algoritmo estándar de clustering con hormigas (Ant Clustering Algorithm o ACA) que posteriormente ha sido utilizado por el análisis numérico de datos [34], grafo particional [90], [91], [92] y minería de texto [50], [69], [133]. Varios autores [50], [132], proponen numerosas modificaciones para mejorar la convergencia y la obtención del número óptimo de clusters. Un enfoque híbrido utiliza el algoritmo de clustering basado en hormigas con el algoritmo de K-means [86] y lo comparan con el tradicional K-means utilizando varios conjuntos de datos.

Al igual que el ACO estándar, el clustering basado en hormigas es un proceso que emplea *feedback* positivo. Las hormigas son modeladas como simples agentes que se mueven aleatoriamente en su ambiente. Este ambiente en general es un plano bidimensional con una grilla cuadrada. Inicialmente cada objeto que representa un patrón multidimensional se distribuye aleatoriamente sobre el espacio bidimensional. Los items de datos que se dispersan en el ambiente, pueden ser tomados, transportados o soltados de una forma probabilística por los agentes. Las operaciones de tomar o soltar están influenciadas por la similitud y la densidad de los items de datos dentro del vecindario local de la hormiga. Generalmente el tamaño del vecindario es de 3×3. La probabilidad de tomar un item de dato es mayor cuando el objeto

está aislado o rodeado por items no similares. Los agentes intentan soltarlos en la vecindad de items similares y de esta forma se obtiene el clustering de los items en la grilla.
Se hibrida este sistema de hormigas con el algoritmo FCM clásico [64](o *Fuzzy c-Means*) para determinar automáticamente el número de clusters en un conjunto de datos [94].
Se han incorporado además, otras modificaciones basadas en hormigas que mejoran la calidad de los clusters, la velocidad de convergencia y en particular la separación espacial entre los clusters en la grilla. Se encuentra una descripción detallada de todas estas variantes y los resultados obtenidos en [137].

4.2.2. Algoritmos de Clustering basados PSO

En los últimos años, PSO ha probado ser efectivo y rápido para resolver ciertos problemas de optimización [52]. También ha sido exitosamente utilizado en muchas áreas de aplicación y en diversas investigaciones [2], [52], [104].
Es posible ver al problema de clustering como un problema de optimización que ubica los centroides óptimos de los clusters en lugar de encontrar una partición óptima. Viendo al problema de clustering de esta forma, se puede aplicar el algoritmo de PSO a un problema de clustering. De forma similar a cualquier otro algoritmo de clustering particional, el objetivo del algoritmo de clustering con PSO es descubrir los centroides apropiados a cada cluster, con el fin de minimizar la distancia Intra-Cluster así como también maximizar la distancia entre los clusters. El algoritmo de PSO realiza una búsqueda globalizada de soluciones mientras que la mayoría de otros procesos de clustering particional realizan una búsqueda localizada. En la búsqueda localizada, la solución obtenida generalmente se ubica en la vecindad de la solución obtenida en las etapas previas. Por ejemplo, el algoritmo de clustering de K-means usa las semillas generadas aleatoriamente como centroides iniciales de los clusters y refina la posición de los centroides en cada iteración. Este proceso de refinamiento del algoritmo de K-means indica que este algoritmo únicamente explora una vecindad muy estrecha que rodea a los centroides iniciales gene-

rados aleatoriamente. El comportamiento del algoritmo de clustering basado en PSO se puede describir en dos fases: una fase de búsqueda global y una fase de refinamiento local. En las iteraciones iniciales, basado en la velocidad inicial de la partícula (ecuación 2.7), se actualiza la velocidad, los dos valores generados aleatoriamente (r_1, r_2) en cada generación y el factor de inercia w que provee la diversidad al cúmulo de partículas para evitar estancamiento en óptimos locales. Múltiples partículas realizan una búsqueda paralela, utilizando soluciones diferentes al mismo momento, lo cual permite explorar más ampliamente el espacio de búsqueda. Luego de varias iteraciones la velocidad de la partícula se reduce gradualmente y la partícula explora en un área más reducida cuando se acerca a la solución óptima. La fase de búsqueda global cambia gradualmente a la fase de refinamiento local. Seleccionando diferentes parámetros se puede controlar el tiempo de cambio de la fase de búsqueda global a la fase de refinamiento local. Mientras más tarde la partícula en cambiar de la fase de búsqueda global a la fase de refinamiento local, mayor es la posibilidad de que pueda encontrar una solución óptima.
En el contexto de clustering, una partícula representa los k centroides de los *clusters*. Es decir que cada partícula x_i se construye de la siguiente manera:

$$x_i = (c_1, ..., c_j, ..., c_K) \tag{4.1}$$

donde c_j representa el $j-$ésimo centroide de la $i-$ésima partícula en el *cluster* C_j.
El fitness de cada partícula se calcula de la siguiente forma:

$$f(x) = \frac{\sum_{j=1}^{K} [\sum_{z \in C_j} d(z, c_j)]}{n} \tag{4.2}$$

donde d es la distancia Euclideana y n es la cantidad de elementos del conjunto de datos, z representa un elemento del conjunto de datos y K es la cantidad de *clusters*.

Los resultados de Omran et al. [80], [81] muestran que el algoritmo de clustering basado en PSO mejora a *K-means*, FCM y a otros algoritmos de

clustering. Van der Merwe y Engelbrecht hibridan este enfoque con un algoritmo de *K-means* [135]. Una sola partícula del cúmulo es inicializada con el resultado del algoritmo de K-means. El resto del cúmulo es inicializado aleatoriamente. Un nuevo enfoque basado en el *sinergismo* de PSO y *Self Organizing Maps* propuesto por Xiao et al. [141], obtiene resultados promisorios aplicando un algoritmo denominado SOM-PSO.
Ciu et al. [139] presentan un algoritmo híbrido basado en PSO para la clasificación de documentos. Aplicando PSO, K-means y un algoritmo híbrido de clustering basado en PSO sobre cuatro diferentes conjuntos de documentos de texto. Los resultados muestran que el algoritmo de PSO híbrido puede generar clusters más compactos que los obtenidos por el algoritmo de K-means.

4.3. Clustering Automático

Muchas de las técnicas de clustering existentes, basadas en AEs, reciben como parámetro el número de clusters k en lugar de determinarlo en tiempo de ejecución. Más aún, en muchas situaciones, el número apropiado de grupos en un conjunto de datos no usados previamente puede ser desconocido o imposible de determinar aún aproximadamente. Por ejemplo, cuando se intenta clasificar una base de datos grande de caracteres escritos a mano en un lenguaje no conocido, no es posible determinar de antemano el número correcto de cartas distintas. De igual forma, en una cooperación robótica multi-agente, un robot debe identificar sus compañeros, los obstáculos, objetivo, etc. de las imágenes de su alrededor tomadas por la cámara digital que funciona como su ojo. Antes de segmentar y ubicar los diferentes objetos, las imágenes tomadas por el robot necesitan ser rápidamente agrupadas como para que los objetos similares puedan marcarse idénticamente en la imagen.
Encontrar el número óptimo de clusters en una base de datos es una tarea desafiante. Este problema ha sido abordado por varios investigadores [16], [78], [119]. Una propuesta de mejora del algoritmo de K-means ISODATA [39] agrega la posibilidad de mezclar clases y partir clases extendidas. Una alternativa a esta propuesta es Syneract [70] que combina K-means con enfoques jerárquicos. Otra mejora al algoritmo de K-means es propuesta por

Rosenberger y Chehdi [16] que determina automáticamente el número de clusters en imágenes usando resultados intermedios. Otra mejora es la propuesta por Pelleg y Moore [25] donde extienden el algoritmo de K-means con una estimación eficiente del número de clusters, denominando a esta variante X-means. Este algoritmo busca eficientemente en el espacio de búsqueda la ubicación de los clusters y el número de clusters para optimizar la medida Criterio de Información Bayesiano (Bayesian Information Criterion or BIC). Otra propuesta de mejora de K-means es la presentada en [42] con el algoritmo de G-means. Este algoritmo comienza con valores pequeños para K, y en cada iteración divide los datos que no corresponden a una distribución Gausiana. De acuerdo a los autores [42] G-means mejora a X-means, sin embargo, funciona para datos que tienen clusters con forma esférica o elíptica. Un programa llamado *snob* [19], [98] utiliza varios métodos para asignar objetos a clases de una forma inteligente. Luego de cada asignación se calcula la medida de Información Wallace (*Wallace Information Measure*) [20], [57] y en base a este cálculo la asignación se acepta o se rechaza. Otro algoritmo basado en K-means es el propuesto por Bischof et al. que usa la descripción de longitud mínima (Minimum Description Length o MDL). Otros autores proponen un algoritmo de clustering no supervisado combinando FCM y la estimación de la velocidad máxima [46]. Lorette et al. [5] propone un algoritmo basado en clustering difuso para determinar dinámicamente el número de clusters generados. Otro enfoque es un algoritmo de clustering basado en un proceso de aglomeración competitiva [43] sin embargo, en este enfoque la inicialización afecta significativamente el resultado del algoritmo. Para identificar automáticamente el número de clusters puede utilizarse también el método de Kohonen (*Self Organizing Maps* o SOM) [61], [126]. SOM combina el aprendizaje competitivo (en el cual los nodos de la red de Kohonen compiten para ser ganadores cuando se presenta un patrón de entrada) con una estructura topológica de nodos [8]. Como SOM es dependiente del orden en el cual se presentan los puntos de datos en [72] se propone una solución a este problema, donde la elección de los puntos de datos puede aleatorizarse durante cada período.

El clustering automático ha sido abordado también en Computación Evo-

lutiva. A continuación se describen algunos desarrollos con algoritmos de clustering evolutivo. Los dos primeros están basados en AG y el tercero usa PSO.

4.3.1. Algoritmo de Clustering Automático basado en AG

El algoritmo propuesto en [111] se basa en un AG llamado GCUK (*Genetic Clustering with Unknown number of cluster*) con un número desconocido de k clusters. En este esquema, los cromosomas se representan con números reales (que son las coordenadas de los centroides en el espacio) y el símbolo ♯ que representa valores no existentes. El número de k centroides se determina con la ejecución del algoritmo. Se fija únicamente un límite superior para k (K_{min}, K_{max}). El *fitness* del cromosoma se calcula a través del índice de Davis-Bouldin (ecuación 3.10). Se aplica crossover de un punto sobre los cromosomas con una probabilidad de crossover fija u_c. Durante el crossover cada cluster se toma como un gen indivisible. Cada alelo válido en un cromosoma, es decir un alelo que no tiene el símbolo ♯, se muta con una probabilidad de mutación constante, u_m. Se genera con distribución uniforme un número δ en el rango $[0, 1]$. Para la mutación si el valor en una posición de un gen es v entonces

$$v \times (1 \pm 2\delta) \text{ si } v \neq 0$$

$$\pm 2\delta \text{ si } v = 0 \tag{4.3}$$

El símbolo + o − aparece con igual probabilidad. Se aplica la selección proporcional descrita en [44]. El proceso de cálculo de *fitness*, selección, mutación y crossover se repite un número máximo de iteraciones. El algoritmo implementa elitismo en cada generación, manteniendo el mejor cromosoma de cada generación fuera de la población. Cuando finaliza el algoritmo, ésta contendrá los clusters finales.
Se utilizó el índice de Davis-Bouldin para medir la validez de los clusters. Se obtuvieron resultados muy alentadores, al probar la efectividad de la propuesta utilizando conjuntos de datos reales y artificiales.

4.3.2. El algoritmo FVGA

El algoritmo FVGA [129] es una extensión del algoritmo GCUK descrito anteriormente. En este algoritmo se define una población de tamaño P, que inicialmente se genera en forma aleatoria. Las cadenas contienen diferentes números de centroides. Se consideran datos $d-$dimensionales y una cadena representa k clusters, entonces en una cadena habrá $k \times d$. En la población inicial los centroides se seleccionan dentro de un rango de datos. Se considera una cadena de longitud mínima $K_{min} = 2$ que representa una solución con 2 clusters. La elección de K_{max} depende del conjunto de datos usado ($K_{max} \leq \sqrt{n}$) donde n es el número de patrones presente en el conjunto de datos.

Supongamos que un cromosoma codifica k centroides y digamos que los centroides son $\overrightarrow{m_1}, \overrightarrow{m_2}, ..., \overrightarrow{m_k}$, los valores miembros μ_{ij} $(i = 1, 2, ..., k; j = 1, 2, ..., n)$ se calculan de la siguiente manera:

$$\mu_{ij} = \frac{1}{\sum_{p=1}^{k} \left(\frac{d(\overrightarrow{m_i}, \overrightarrow{Z_j})}{d(\overrightarrow{m_p}, \overrightarrow{Z_j})} \right)^{\frac{2}{q-1}}} \tag{4.4}$$

donde q es un coeficiente de peso, un término difuso. La función de *fitness* para el cromosoma se define como $\frac{1}{XB}$, donde XB es el índice de validez propuesto por Xie y Beni [9] y se aplica crossover de un punto y mutación.

Los resultados obtenidos con este enfoque, que intenta manejar el problema de FCM, con esta estrategia que no requiere un conocimiento *a priori* del número de clusters, provee soluciones cercanas al óptimo y que superan ampliamente a FCM.

4.3.3. Clustering dinámico con Evolución Diferencial (ED)

En esta propuesta se ha cambiado la representación clásica de un algoritmo de ED para mejorar sus propiedades de convergencia. Además, se usa un nuevo esquema de representación para las variables de búsqueda con el fin de determinar el número óptimo de clusters. Este enfoque se ha denominado ACDE (*Automatic Clustering Differential Evolution*)[112].

En este algoritmo la representación de los datos se realiza de la siguiente

manera:
Para n datos, cada uno $d-$dimensional y para un número máximo de clusters (especificado por el usuario) K_{max}, un cromosoma es un vector de números reales de dimensión $K_{max} + K_{max} \times d$. Las primeras K_{max} posiciones son números reales entre $[0, 1]$, cada posición indica si el cluster correspondiente está activo o no. El resto de las posiciones corresponden a los K_{max} centroides cada uno $d-$dimensional. Luego de experimentar con varios índices de validez, el seleccionado fue CS presentado en [18] y que para este algoritmo se lo utilizó como función de fitness debido a su capacidad para manejar clusters de diferentes densidades y/o tamaños. La población inicial se inicializa aleatoriamente con valores entre $[0, 1]$. Cada vector de la población contiene k centroides elegidos aleatoriamente y k valores de activación (elegidos aleatoriamente).
Luego, hasta que no se alcanza el criterio de parada, para cada dato se calcula su distancia con respecto a los clusters activos y se lo asigna al cluster al que pertenece. En caso que corresponda, se actualizan los centroides del cromosoma y se cambian los miembros de la población de acuerdo a la función de *fitness*. Este proceso se repite. Al llegar al criterio de parada se obtendrán los centroides óptimos. Según los autores este algoritmo presenta una alternativa atractiva para el clustering dinámico [113].

4.3.4. Clustering dinámico con PSO

Este algoritmo (Dynamic Clustering with Particle Swarm Optimization o DCPSO) fue propuesto por Omran et at. en el 2005 [81]. El algoritmo funciona de la siguiente manera: un pool de centroides M que se eligen aleatoriamente del conjunto de puntos Z. El cúmulo de partículas S, se inicializa aleatoriamente. Se aplica PSO binario [114], para encontrar el "mejor" conjunto de centroides M_τ, de M. Se aplica K-means a M_τ con el fin de refinar el centroide elegido. Luego M es el conjunto de M_τ más M_r que es aleatoriamente elegido desde Z (para agregar diversidad a M). Luego se repite el algoritmo usando el nuevo M. Cuando se alcanza el criterio de terminación, M_τ será el número de clusters "óptimo" para el conjunto Z. De igual forma que GCUK y FVGA, se puede usar como función de fitness cualquier medida

de índices de validación de clustering. Omran et al. [81] usan como función de fitness el índice de validación propuesto en [105] de la siguiente forma:

$$V = (c \times \mathcal{N}(2,1) + 1) \times \frac{Intra}{Inter} \tag{4.5}$$

donde c es un parámetro especificado por el usuario y $\mathcal{N}(2,1)$ es una distribución Gausiana con media 2 y desviación estándar 1. $Intra$ corresponde a la medida de distancia Intra-Cluster e $Inter$ corresponde a la medida de distancia Inter-Cluster. Este algoritmo ha sido aplicado a imágenes naturales y ha sido comparado con otras técnicas de clustering no supervisado obteniendo mejoras significativas.

4.4. Conclusión

En la literatura se encuentran diferentes propuestas que resuelven el problema de clustering utilizando metaheurísticas y obtienen muy buenos resultados.

La idea principal de los algoritmos de clustering basados en AGs es crear una población de soluciones candidatas a un problema, las cuales se refinan iterativamente, seleccionando las buenas soluciones para la siguiente iteración. Una ventaja de estos algoritmos es su habilidad de hacer frente a óptimos locales. Además, existen propuestas de clustering usando EE y PE con promisorios resultados.

Otro tipo de propuestas utilizan enfoques basados en colonia de hormigas, formulando poderosas heurísticas inspiradas en su comportamiento. Los enfoques basados en PSO han demostrado ser efectivos para resolver problemas de optimización y tomando al problema de clustering como un problema de optimización han demostrado obtener buenos resultados.

Finalmente se muestran distintos enfoques para clustering automático basados en AGs, ED y PSO también con buenos resultados.

Los algoritmos de Inteligencia Colectiva han mostrado mejorar métodos clásicos de agrupamiento de grandes y complejas bases de datos. Por ello, se presenta una propuesta híbrida que utiliza Particle Swarm Optimization combinado con un clásico algoritmo de clustering particional (K-means) para el procesamiento estático y dinámico de tareas de clustering.

Capítulo 5

PSO Híbrido para la tarea de Clustering

El *clustering* de datos ayuda a discernir la estructura y simplifica la complejidad de cantidades masivas de datos. Es una técnica común y se utiliza en diversos campos como, aprendizaje de máquina, minería de datos, reconocimiento de patrones, análisis de imágenes y bioinformática, donde la distribución de la información puede ser de cualquier tamaño y forma. La eficiencia de los algoritmos de *clustering* es extremadamente necesaria cuando se trabaja con enormes bases de datos y tipos de datos de grandes dimensiones. Por esta razón en este capítulo se propone un algoritmo híbrido que combina la capacidad explorativa de PSO y las ventajas del conocido algoritmo K-means con el objetivo de obtener mejores resultados que los obtenidos de forma separada por cada uno de ellos.

5.1. Aplicación de PSO para *Clustering*

Es posible ver al problema de *clustering* como un problema de optimización que localiza los centroides óptimos de los *clusters* en lugar de encontrar la partición óptima. A diferencia de la búsqueda localizada del algoritmo de *K-means*, el algoritmo de *clustering* con PSO realiza una búsqueda global del espacio de búsqueda. Con el fin de combinar ambas características se utiliza un algoritmo de *clustering* híbrido denominado PSO+K-means. En el algoritmo de PSO+K-means se combina la habilidad de la búsqueda globalizada del algoritmo de PSO con la rápida convergencia del algoritmo de K-means. Con

la búsqueda global se garantiza que cada partícula busque en forma amplia cubriendo todo el espacio del problema y con la búsqueda local se trata de que todas las partículas converjan al óptimo cuando una partícula se acerca a la vecindad de la solución óptima.
En este contexto, una partícula representa los k centroides de los *clusters*. Es decir que cada partícula x_i se construye de la siguiente manera:

$$x_i = (c_1, ..., c_j, ..., c_K) \tag{5.1}$$

donde c_j representa el *j-ésimo* centroide de la *i-ésima* partícula en el *cluster* C_j.

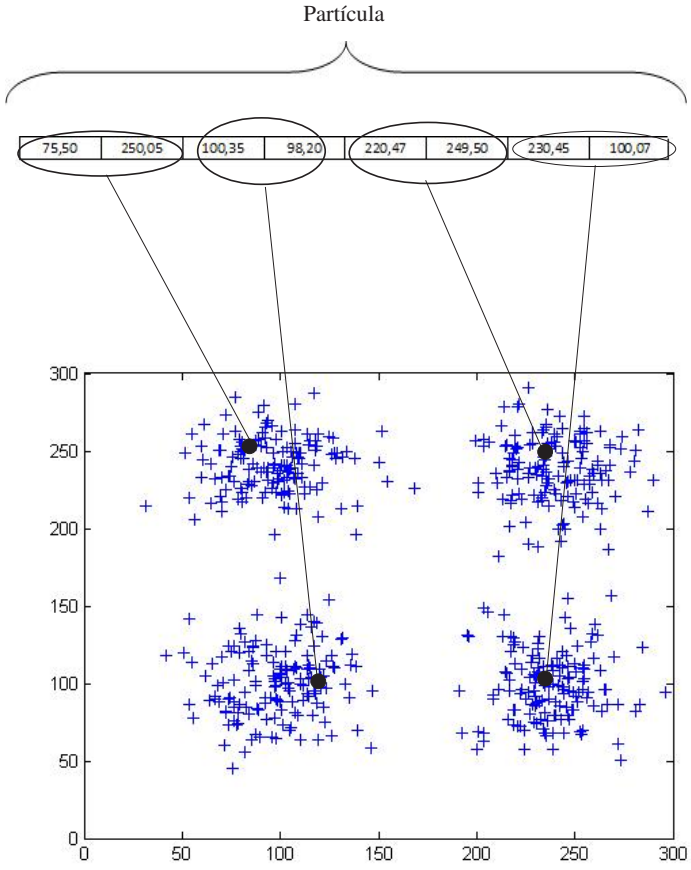


Figura 5.1: Representación de una partícula con cuatro centroides

En la Figura 5.1 se muestra un ejemplo de esta representación. Supongamos que cada elemento del conjunto es de dimensión dos y que existen cuatro clusters. En la gráfica se representa una partícula que está formada

por cuatro posibles centroides que corresponden a cuatro puntos del espacio de búsqueda (los cuatro círculos negros de la Figura 5.1). Basándose en esta representación, en la siguiente sección se describe el algoritmo propuesto denominado PSO+K-means.

5.2. PSO+K-means

En el presente algoritmo se intenta combinar las exploración (por medio de PSO) con la intensificación (lograda con K-means) de la siguiente forma: del cúmulo inicial se establece la mejor posición local y la mejor posición global, luego se actualiza la velocidad y la posición de la partícula, se selecciona la mejor partícula global y esa partícula es la entrada al K-means, al finalizar K-means el centroide resultante reemplaza a la partícula global y este proceso se repite. En la siguiente gráfica se muestra de forma genérica como sería esta hibridación.

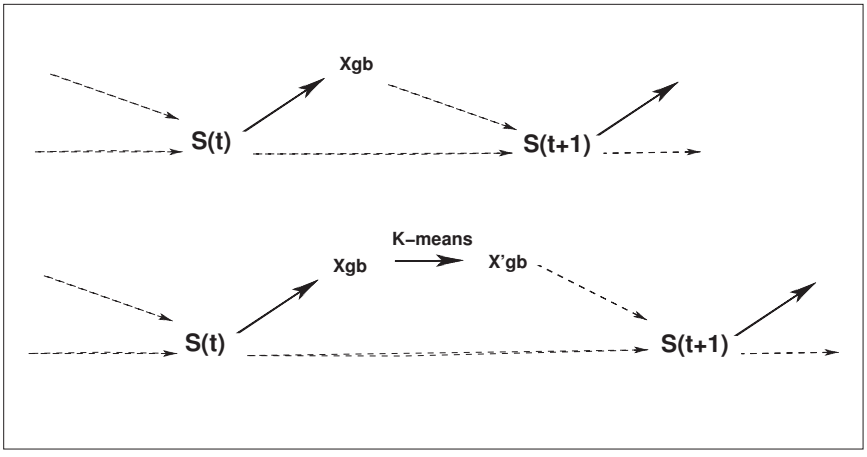


Figura 5.2: Representación genérica de la propuesta PSO+K-means

En la parte superior de la gráfica se representa el comportamiento general de un algoritmo de PSO donde del cúmulo actual $S(t)$ se obtiene la mejor partícula (x_{gb}). Para obtener el próximo cúmulo ($S(t+1)$) las partículas se mueven teniendo en cuenta la posición de la mejor partícula. Mientras que en

la parte inferior de la gráfica se representa el comportamiento de la propuesta presentada que aplica un proceso híbrido con búsqueda local representada por K-means. En este caso, luego de seleccionada la mejor partícula (x_{gb}) y antes de obtener el próximo cúmulo, se aplica el proceso de búsqueda local implementado por el algoritmo de K-means, obteniendo (x'_{gb}) y produciendo de esta manera una intensificación en la búsqueda. Es importante aclarar, que dadas las características del K-means $f(x'_{gb}) \leq f(x_{gb})$, en el caso de que $f(x'_{gb}) = f(x_{gb})$, luego $x'_{gb} = x_{gb}$. Finalmente el cúmulo se mueve ($S(t+1)$) teniendo en cuenta la posición de esta nueva mejor partícula.

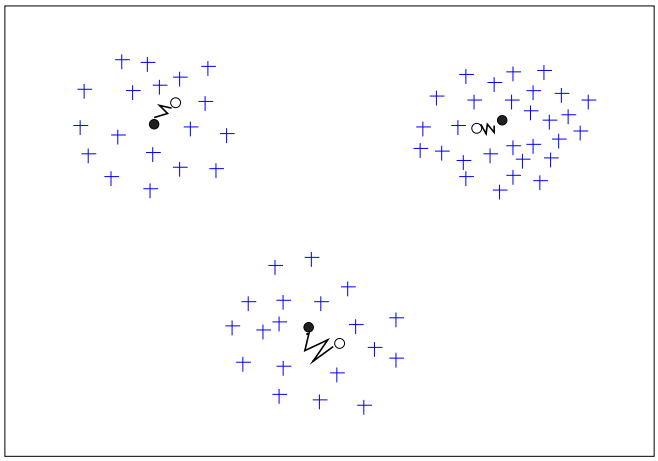


Figura 5.3: Centroides iniciales y centroides finales luego de aplicar la intensificación con K-means

Para ver este comportamiento en un ejemplo particular se muestra en la siguiente gráfica (Figura 5.3) una nube de puntos, donde los círculos sin relleno corresponden a los centroides encontrados por el PSO (es decir la mejor partícula global) y luego de realizar la intensificación con el proceso de K-means, usando la mejor partícula global como entrada, se obtienen nuevos centroides representados por los círculos negros.

A continuación se describe en detalle el procedimiento de PSO+K-means en el Algoritmo 5.

En la siguiente se sección se describe en conjunto de datos utilizado para probar este algoritmo así como también las medidas de calidad utilizadas y finalmente los experimentos y resultados obtenidos.

Algoritmo 5 Algoritmo PSO+K-means

1: Inicializa cada partícula con k centroides seleccionados aleatoriamente del conjunto de datos
2: **for** $t = 1$ to t_{max} {t_{max} es el máximo número de iteraciones} **do**
3: **for** cada partícula x_i **do**
4: **for** cada vector z del conjunto de datos **do**
5: Calcula la distancia Euclideana $d(z, c_j)$ a todos los centroides C_j
6: Asignar z a su *cluster* C_j más cercano tal que la distancia a ese *cluster* sea la mínima.
7: Calcula el fitness usando la ecuación 4.2
8: **end for**
9: Actualiza la mejor posición global y la mejor posición local
10: Actualiza la velocidad y posición de la partícula usando la ecuación 2.7 y 2.2, respectivamente
11: **end for**
12: La mejor partícula encontrada es el vector de centroides iniciales en el proceso de *K-means*
13: **repeat**
14: asigna cada elemento a su *cluster* más cercano
15: recalcula cada centroide $c_j = \frac{1}{|C_j|} \sum_{\forall z \in C_j}$; donde $\mid C_j \mid$ corresponde al número de elementos en el *cluster* C_j
16: **until** Condición de parada sea verdadera
17: Reemplazar la mejor partícula global con los centroides obtenidos por *K-means*
18: **end for**

Para el caso del algoritmo PSO simple para *clustering*, el algoritmo es el mismo, excepto que éste no realiza los pasos 12 al 17 correspondientes a la hibridación con el K-means.

5.3. Experimentos y Resultados

A fin de comprobar la calidad de los resultados obtenidos por el algoritmo propuesto, se generó un conjunto de datos artificiales y se implementaron los algoritmos de PSO y K-means para poder comparar el algoritmo propuesto. Todos los algoritmos se codificaron en lenguaje JAVA usando la plataforma de desarrollo Eclipse 3.3.0, y se ejecutaron en una computadora tipo PC, procesador Intel Core2 Duo, $2Gz$ y $4\ GB$ de RAM. Para el análisis estadístico y la generación de las gráficas se utilizó MATLAB Versión 7.1.

A continuación se describe el conjunto de datos utilizado, la configuración de los algoritmos y los resultados obtenidos.

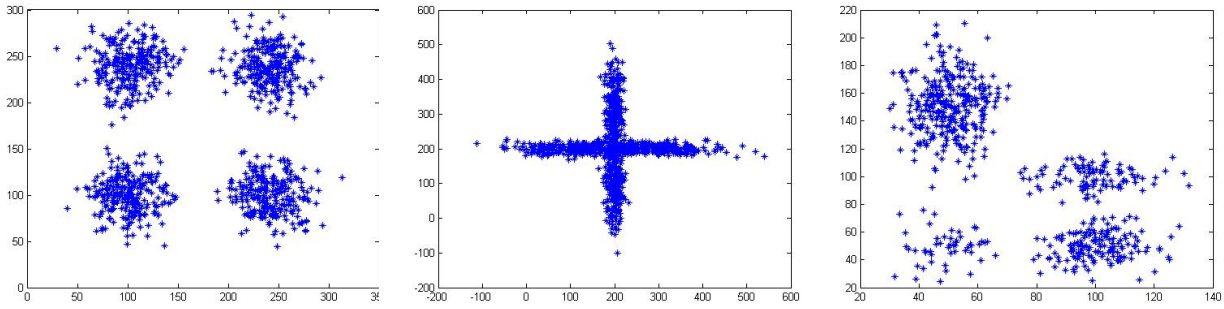
5.3.1. Conjunto de datos

Los algoritmos se probaron con tres tipos de conjuntos de datos generados artificialmente y con tres conjuntos de datos provistos por *Donald Bren School of Information and Computer Sciences*[1]. Los datos artificiales [2] tienen las siguientes características:

- Se utilizó una distribución Normal y se generaron números aleatorios a partir del método de Box-Müller. Se generaron instancias con elementos de 2, 3, 10, 50 y 100 dimensiones, con el fin de poder trasladar estos datos a problemas reales con dimensiones grandes que representen distintos atributos de un elemento. Además, de trabajar con tamaños de conjuntos de datos de 600, 800 y 1000 elementos. Cada conjunto de datos estaba formado por cuatro *clusters* donde los elementos se distribuyeron de forma tal de obtener tres tipos diferentes de *clusters*:
  - *Clusters* de igual cantidad de elementos, globulares y separados (Figura 5.4(a)).
  - *Clusters* de igual cantidad de elementos, con forma elíptica y entremezclados (Figura 5.4(b)).
  - *Clusters* de diferente cantidad de elementos, globulares y separados (Figura 5.4 (c)).

La Figura 5.4 muestra tres conjuntos de datos artificiales de dimensión 2, con 1000 elementos cada uno (instancias denominadas I26Ba2-1000, I41Bb2-1000, I56Bc2-1000; el nombre de cada instancia se forma de la siguiente manera por ejemplo, I24Ba50-800 representa lo siguiente: *I24* identificador de la instancia *Ba* se refiere a la distribución de los elementos en el conjunto de datos con la forma de la Figura 5.4(a), *50* es la dimensión de cada elemento dentro del conjunto y *800* es la cantidad de puntos en el conjunto de datos). Con las características anteriormente descritas se generaron 45 instancias.

[1] *(http://mlean.ics.uci.edu/MLSummary.html).*
[2] *(http://www.uaco.unpa.edu.ar/uaco/www/labtem/).*



(a) (b) (c)

Figura 5.4: Conjunto de tres tipos de datos artificiales

Por cuestiones de espacio para la generación de las gráficas que se muestran en la siguiente sección se agruparon la instancias de la siguiente manera:

- Grupo-A: está formado por 15 instancias con los datos distribuidos con la forma de la Figura 5.4 (a) donde cinco instancias tienen respectivamente, dimensión 2, 3, 10, 50 y 100 con 600 elementos del conjunto de datos cada una, otras cinco instancias tienen respectivamente, dimensión 2, 3, 10, 50 y 100 con 800 elementos del conjunto de datos cada una y las cinco últimas instancias tienen respectivamente, dimensión 2, 3, 10, 50 y 100 con 1000 elementos del conjunto de datos cada una.

- Grupo-B: incluye 15 instancias que están agrupadas de manera similar al Grupo-A pero en este grupo, los datos están distribuidos como se muestra en la Figura 5.4(b).

- Grupo-C: incluye 15 instancias que están agrupadas de manera similar al Grupo-A pero en este grupo, los datos están distribuidos como se muestra en la Figura 5.4(c).

El conjunto de datos reales está formado por las siguientes bases de datos:

- "*Iris plants*": es una base de datos con 4 atributos numéricos, 3 clases y 150 elementos.

- "*Glass identification*": esta base de datos contiene 6 tipos de vidrio definidos en términos de su contenido de óxido (es decir, Na, Fe, K, etc), con 9 atributos y 214 elementos.

- "*Teaching assistant evaluation*" (TAE): esta base de datos contiene 151 elementos, 6 atributos, 3 clases. Los datos corresponden al desempeño de la enseñanza de tres semestres regulares y dos semestres de verano de 151 asistentes de enseñanza en la Universidad de Wisconsin - Madison.

5.3.2. Medidas de calidad utilizadas y Configuración de Algoritmos

A los efectos de establecer la calidad de los *clusters* encontrados por los distintos algoritmos estudiados, hemos considerado una serie de medidas no-supervisadas. Las medidas no-supervisadas de validez de un *cluster* pueden dividirse en dos clases. Medida de cohesión de los *clusters* (compacto, apretado) que determina cuán cercanos están los objetos dentro del *cluster* y la separación (aislamiento) que determina lo distinto y bien separado que está un *cluster* con respecto a los otros. Estas medidas a menudo son llamadas índices internos debido a que usan sólo información presente en el conjunto de datos. La calidad de los *clusters* obtenidos por cada algoritmo se analizó según los siguientes índices internos. La Distancia Intra-Cluster y la Distancia Inter-Cluster fueron descritas en las Ecuaciones 3.6 y 3.7, pero se describen nuevamente a continuación.

- Cuantización del error:

$$Error\_C = \frac{\sum_{i=1}^{K}(\sum_{z \in C_i} dist(c_i, z) / \mid C_i \mid)}{K} \tag{5.2}$$

  donde K es el número de cluster, $|C_i|$ es la cantidad de elementos dentro del cluster i y c_i representa el centroide del cluster i.

- Distancia Intra-Cluster:

$$Intra\_C = \frac{\sum_{i=1}^{K}(\sum_{x,t \in C_i} dist(x,t)/m_i)}{K}$$

  donde C_i representa el cluster i, K es la cantidad de clusters y m_i es la cantidad de elementos pertenecientes al cluster i.

- Distancia Inter-Cluster.

$$Inter\_C = \frac{\sum_{i=1}^{K-1} \sum_{j=i+1}^{K} dist(c_i, c_j)}{\sum_{i=1}^{K-1} i}$$

donde c_i representa al elemento central (centroide) del cluster i, y c_j representa el centroide del cluster j y K es la cantidad de clusters.

Para todos los algoritmos se realizaron 30 corridas independientes. En PSO y PSO+K-means se utilizó un cúmulo de 10 partículas para espacios del problema con dimensiones pequeñas ($n_x < 10$) y para dimensiones mayores ($n_x \geq 10$), el cúmulo utilizado fue de 15 partículas. Los valores de las contantes c_1 y c_2 se fijaron en 1.49 y $w = 0{,}72$. Estos valores se fijaron teniendo en cuenta trabajos anteriores con óptimos resultados y además, han sido probados experimentalmente.

5.3.3. Resultados

Esta subsección ha sido organizada de la siguiente manera, en primer lugar se presentan los resultados obtenidos por los algoritmos PSO, K-means y PSO+K-means, para todas las instancias analizadas y en segundo lugar se eligen algunas instancias en forma aleatoria para realizar un análisis más detallado de los resultados.

A continuación se muestran los resultados de las 45 instancias analizadas. Por cuestiones de espacio se presentan las gráficas organizadas por grupos correspondientes a cada grupo del conjunto de datos y se muestra el valor $\frac{Intra-Cluster}{Inter-Cluster}$ que representa la combinación de dos de las medidas analizadas la distancia Intra-Cluster y distancia Inter-Cluster. Donde un valor pequeño de $\frac{Intra-Cluster}{Inter-Cluster}$ representa homogeneidad dentro de los *clusters* y heterogeneidad entre los *clusters*. Teniendo en cuenta, que un buen *clustering* debería ser aquel que contenga cada uno de los *clusters* individuales homogéneos y

entre todos los *clusters* tan heterogéneos como sea posible.

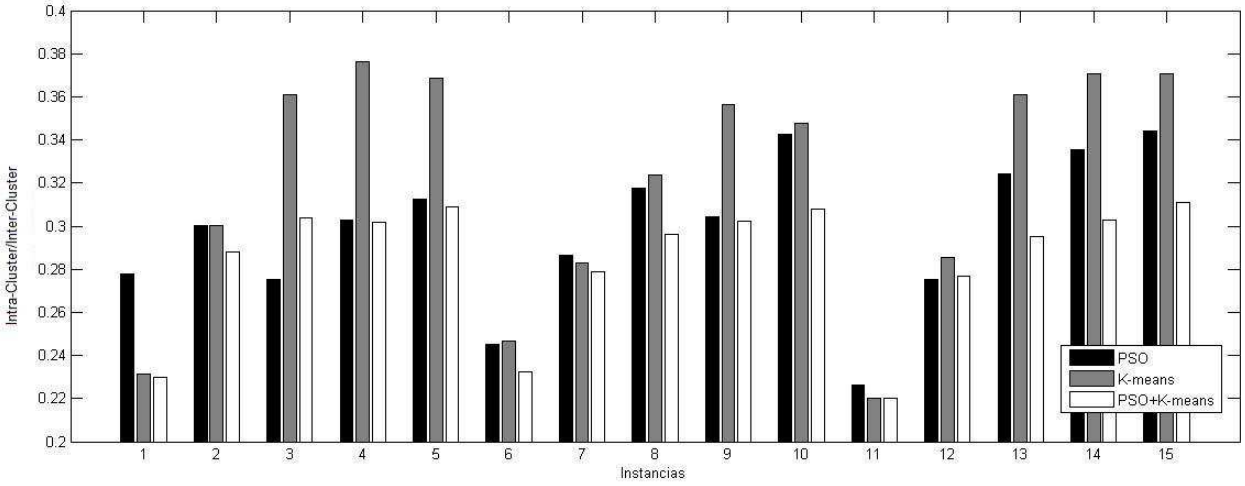


Figura 5.5: Valores $\frac{Intra-Cluster}{Inter-Cluster}$ para el Grupo-A

En la Figura 5.5 se muestran los respectivos valores de $\frac{Intra-Cluster}{Inter-Cluster}$ de los tres algoritmos para el grupo de 15 instancias denominado Grupo-A. Para poder visualizar mejor los resultados, el eje Y comienza en el valor 0,2. En esta figura se puede observar que con PSO+K-means se obtienen mejores resultados para esta variable analizada en 13 de las 15 instancias. Únicamente para las instancias 3 y 12 en el eje X (que corresponden a las instancias denominadas $I18Ba10-600$ y $I27Ba3-1000$) el algoritmo PSO obtiene mejores resultados que los otros dos algoritmos.

La Figura 5.6 muestra los valores de $\frac{Intra-Cluster}{Inter-Cluster}$ obtenidos por los tres algoritmos para las instancias del Grupo-B. En este caso, también para poder visualizar mejor los resultados, el eje Y comienza en el valor 0,3. Se puede observar que con PSO+K-means se obtienen mejores resultados para la variable analizada en 9 de las 15 instancias. Para las instancias 1, 2, 6, 7, 11 y 12 (que corresponden a las instancias denominadas $I31Bb2-600$, $I32Bb3-600$, $I36Bb2-800$, $I37Bb3-800$, $I41Bb21000$ y $I42Bb3-1000$, respectivamente) el algoritmo K-means obtuvo mejores resultados que los otros dos.

Por su parte, en la Figura 5.7 se muestran los valores $\frac{Intra-Cluster}{Inter-Cluster}$ de los

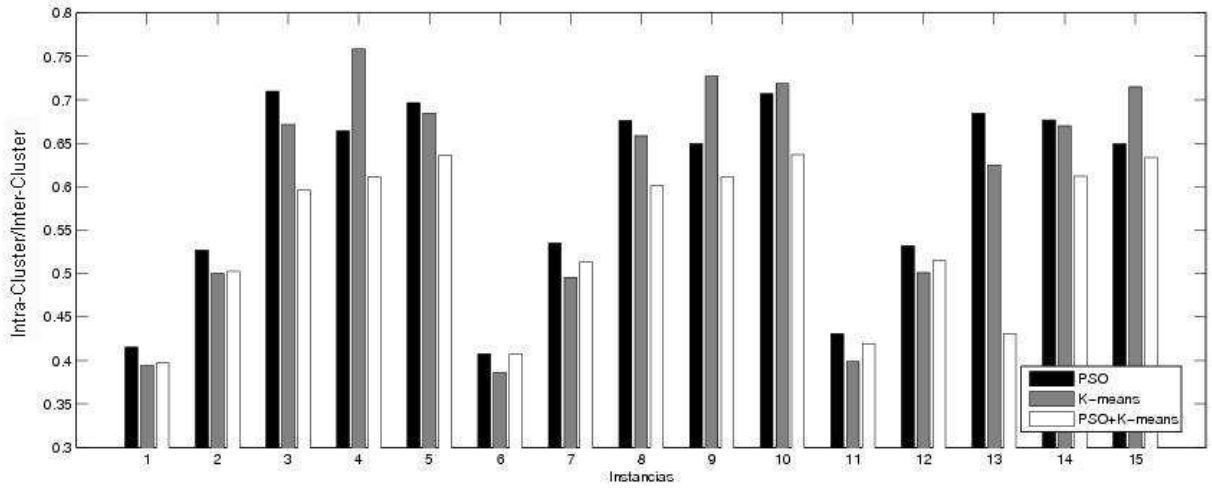


Figura 5.6: Valores $\frac{Intra-Cluster}{Inter-Cluster}$ para el Grupo-B

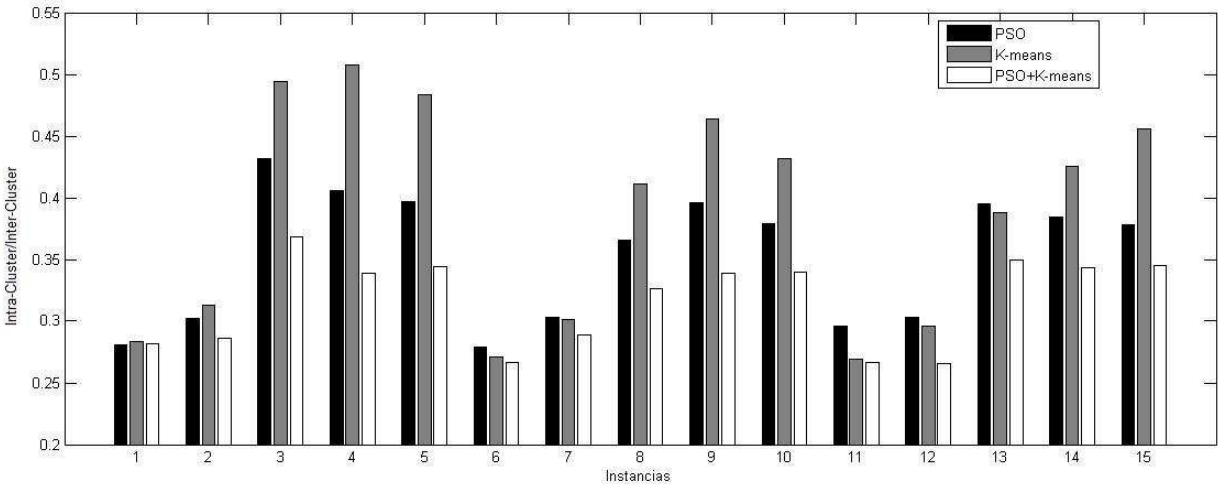


Figura 5.7: Valores $\frac{Intra-Cluster}{Inter-Cluster}$ para el Grupo-C

tres algoritmos para las instancias del Grupo-C. De igual forma que en las dos figuras anteriores, el eje Y comienza en el valor 0,2. Se puede observar que con esta distribución de los datos con el algoritmo PSO+K-means se obtienen mejores resultados para las variable analizada en 14 de las 15 instancias. Únicamente, para la instancia 1 (que corresponde a $I46Bc2-600$) y para la variable analizada, el algoritmo PSO obtuvo una leve mejoría con respecto a los otros dos algoritmos.

Por último, para el conjunto de datos reales Figura 5.8, se ha elegido una escala logarítmica en el eje Y para mostrar los resultados. Para cada instan-

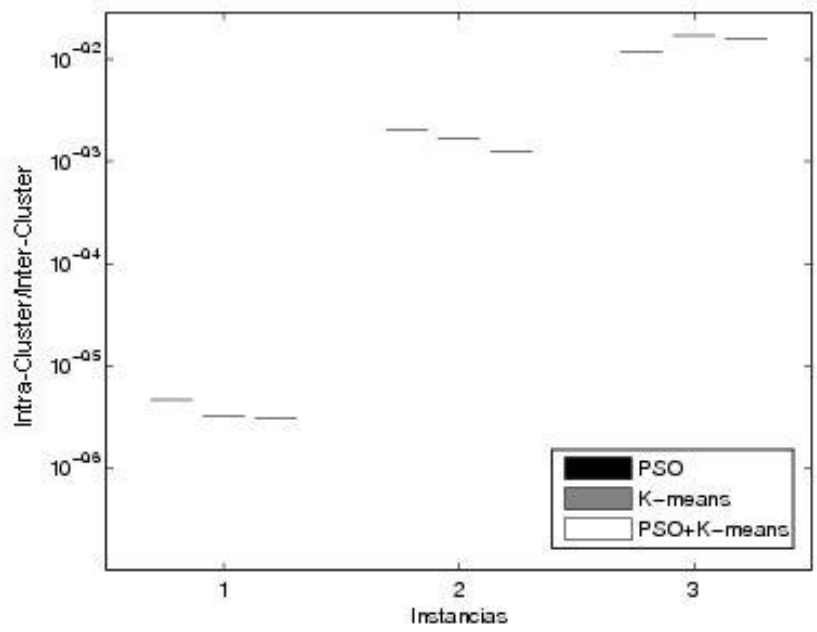


Figura 5.8: Valores $\frac{Intra-Cluster}{Inter-Cluster}$ para el conjunto de datos reales, donde 1=Iris, 2=Glass y 3=TAE

cia representada, el segmento de la izquierda corresponde a PSO, el segmento central corresponde a K-means y el segmento de la derecha corresponde a PSO+K-means. Se puede observar que se obtienen mejores resultados con PSO+K-means en dos de las 3 instancias. Para la instancia 3 (correspondiente la base de datos TAE), PSO obtiene un menor valor comparado con los otros dos algoritmos.

En la Tablas 5.1, 5.2, 5.3 y 5.4 se muestran los valores medios de 30 corridas independientes obtenidos por los algoritmos PSO, *K-means* y PSO+K-means, para las medidas Cuantización del Error, Distancia Intra-Cluster, Distancia Inter-Cluster, el valor $\frac{Intra-Cluster}{Inter-Cluster}$ y se seleccionaron algunos resultados que fueron elegidos aleatoriamente. De todo el conjunto de instancias se eligieron tres instancias de cada grupo y las tres instancias que pertenecen al conjunto de datos reales. Es importante comentar que se realizó el análisis estadístico para las 45 instancias generadas artificialmente y para las cuatro medidas, Cuantización del error, Distancia Intra-Cluster, Distancia Inter-Cluster y $\frac{Intra-Cluster}{Inter-Cluster}$ analizadas. En la mayoría de los resultados se aplicó el test de Kruskall-Wallis, pues los datos no tenían una distribución normal. Para

la medida Cuantización del Error el 95 % de las instancias tuvieron diferencias estadísticamente significativas. En cuanto a las Distancia Intra-Cluster e Inter-Cluster, en ambos casos el 80 % de las instancias tuvieron diferencias estadísticamente significativas. Finalmente para el valor $\frac{Intra-Cluster}{Inter-Cluster}$ el 75 % de las instancias tuvieron diferencias estadísticamente significativas.
Todas las tablas que se mostrarán a continuación contienen la siguiente información por medida analizada. La primer columna corresponde a la instancia elegida, la segunda columna es el resultado de la Cuantización del Error obtenido por el algoritmo PSO, la tercer columna es el resultado de la Cuantización del Error obtenido por el algoritmo K-means, la cuarta columna corresponde a los resultados obtenidos por PSO+K-means, la quinta columna denominada Significativa muestra si entre los algoritmos existen diferencias estadísticamente significativas para la medida analizada y finalmente la última columna denomina Entre-Quién, muestra entre cuál o cuáles algoritmos existen diferencias que son estadísticamente significativas. Cabe aclarar que para esta última columna el valor 1 corresponde al algoritmo PSO, el valor 2 corresponde a K-means y el valor 3 corresponde a PSO+K-means. El o los algoritmos que se encuentran a la izquierda de la flecha ($\longrightarrow$) tienen diferencias estadísticamente significativas con respecto a él o los algoritmos que se encuentran a la derecha de la flecha. Seguidamente se muestran algunas instancias de éstos resultados obtenidos.

En la Tabla 5.1 se muestran los resultados obtenidos para la medida Cuantización del Error por los algoritmos PSO, K-means y PSO+K-means. En negrita se colocan los menores valores. En esta tabla se puede observar que para todas las instancias elegidas los valores obtenidos por PSO+K-means son menores a los obtenidos por los otros dos algoritmos. Sin embargo, no existen diferencias estadísticamente significativas entre K-means y PSO+K-means. Por ejemplo en la primer instancia los algoritmos 2 y 3 es decir, K-means y PSO+K-means tienen diferencias estadísticamente significativos con respecto al algoritmo 1, es decir, con respecto a PSO para la medida Cuantización del Error. Esto significa que PSO+Kmeans y K-means tienen una mejor performance con respecto a PSO para esa medida de calidad y

Tabla 5.1: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la Cuantización del Error

Cuantización del Error					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I22Ba3-800	36,96	31,94	**31,67**	Si	$2,3 \longrightarrow 1$
I23Ba10-800	81,04	67,40	**62,33**	Si	$2,3 \longrightarrow 1$
I24Ba50-800	153,91	161,31	**151,19**	Si	$1,3 \longrightarrow 2$
I38Bb10-800	166,84	156,46	**151,27**	Si	$2,3 \longrightarrow 1$
I39Bb50-800	407,95	369,07	**361,26**	Si	$2,3 \longrightarrow 1$
I43Bb10-1000	176,35	151,10	**146,40**	Si	$2,3 \longrightarrow 1$
I58Bc10-1000	50,05	38,98	**36,64**	Si	$2,3 \longrightarrow 1$
I59Bc50-1000	118,49	95,78	**90,11**	Si	$2,3 \longrightarrow 1$
I60Bc100-1000	168,68	138,98	**136,52**	Si	$2,3 \longrightarrow 1$
Iris-Plants	0,72	0,65	**0,64**	Si	$3 \longrightarrow 1,2$
Glass	1,46	1,58	**1,44**	Si	$1,3 \longrightarrow 2$
TAE	11,21	10,15	**9,83**	Si	$3 \longrightarrow 1,2$

en esta instancia. En una única instancia PSO y PSO+K-means tienen una mejor performance que K-means ($I24Ba50 - 800$), ya que ambos presentan diferencias estadísticamente significativas con respecto a K-means.

Tabla 5.2: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la Distancia Intra-Cluster

Distancia Intra-Cluster					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I22Ba3-800	46,43	45,25	**44,90**	Si	$2,3 \longrightarrow 1$
I23Ba10-800	89,65	93,83	**88,32**	Si	$1,3 \longrightarrow 2$
I24Ba50-800	**199,38**	222,10	**199,38**	Si	$1,3 \longrightarrow 2$
I38Bb10-800	**213,90**	220,65	214,48	Si	$1,2 \longrightarrow 3$
I39Bb50-800	**481,96**	518,64	491,96	Si	$1,3 \longrightarrow 2$
I43Bb10-1000	208,69	213,17	**192,03**	Si	$2,3 \longrightarrow 1$
I58Bc10-1000	55,16	54,67	**51,43**	No	—
I59Bc50-1000	119,64	**113,88**	115,02	Si	$1,3 \longrightarrow 2$
I60Bc100-1000	168,68	194,35	**161,93**	Si	$1,3 \longrightarrow 2$
Iris-Plants	0,93	**0,92**	**0,92**	No	—
Glass	1,97	2,29	**1,96**	Si	$1,3 \longrightarrow 2$
TAE	14,22	13,96	**13,71**	Si	$1,2 \longrightarrow 3$

En la Tabla 5.2 se muestran los resultados que corresponden a los pro-

medios obtenidos para la medida Intra-Cluster por los algoritmos PSO, K-means y PSO+K-means. En este caso se puede observar que en 9 de las 12 instancias PSO+K-means obtiene menores valores para la medida analizada. En 2 instancias las diferencias no son significativas estadísticamente. Y en 6 instancias para PSO y PSO+K-means se puede afirmar que tienen un mejor comportamiento que K-means, pues las diferencias son estadísticamente significativas.

Tabla 5.3: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la Distancia Inter-Cluster

Distancia Inter-Cluster					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I22Ba3-800	**161,74**	159,85	160,95	Si	$1,3 \longrightarrow 2$
I23Ba10-800	282,26	289,76	**298,37**	Si	$2,3 \longrightarrow 1$
I24Ba50-800	654,94	623,29	**659,56**	Si	$2 \longrightarrow 3$
I38Bb10-800	316,29	334,98	**356,83**	Si	$2,3 \longrightarrow 1$
I39Bb50-800	741,84	713,08	**788,53**	Si	$1,2 \longrightarrow 3$
I43Bb10-1000	304,80	341,13	**445,42**	Si	$1,3 \longrightarrow 2$
I58Bc10-1000	139,46	140,82	**147,21**	Si	$1 \longrightarrow 3$
I59Bc50-1000	311,24	314,64	**334,93**	No	—
I60Bc100-1000	446,02	426,41	**469,57**	No	—
Iris-Plants	3,16	3,27	**3,28**	Si	$1,3 \longrightarrow 2$
Glass	3,65	**4,34**	3,84	Si	$1,3 \longrightarrow 2$
TAE	**22,19**	20,98	20,73	No	—

En la Tabla 5.3 se muestran los resultados que corresponden a los promedios obtenidos para la medida Inter-Cluster por los algoritmos PSO, K-means y PSO+K-means. En este caso se puede observar que en 9 de las 12 instancias PSO+K-means obtiene mayores valores para la medida analizada. En 3 instancias las diferencias no son significativas estadísticamente. Y solo en 5 instancias de las instancias donde PSO+K-means obtiene mejores valores las diferencias son estadísticamente significativas con respecto a K-means, permitiendo afirmar que este algoritmo es más robusto que K-means, para esta medida y para estas instancias.

En la Tabla 5.4 se muestran los resultados que corresponden a los promedios obtenidos para el valor $\frac{Intra-Cluster}{Inter-Cluster}$ por los algoritmos PSO, K-means y PSO+K-means. En este caso se puede observar que en 11 de las 12 ins-

Tabla 5.4: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para el valor $\frac{Intra-Cluster}{Inter-Cluster}$

Valor $\frac{Intra-Cluster}{Inter-Cluster}$					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I22Ba3-800	0,2864	0,2831	**0,2790**	Si	$1,3 \longrightarrow 2$
I23Ba10-800	0,3176	0,3239	**0,2960**	Si	$2,3 \longrightarrow 1$
I24Ba50-800	0,3045	0,3920	**0,3023**	Si	$3 \longrightarrow 2$
I38Bb10-800	0,6763	0,6587	**0,6011**	Si	$3 \longrightarrow 1$
I39Bb50-800	0,6497	0,7273	**0,6112**	Si	$3 \longrightarrow 1,2$
I43Bb10-1000	0,6847	0,6249	**0,4311**	Si	$3 \longrightarrow 1,2$
I58Bc10-1000	0,3955	0,3883	**0,3493**	Si	$3 \longrightarrow 1$
I59Bc50-1000	0,3844	0,4255	**0,3434**	No	—
I60Bc100-1000	0,3782	0,4558	**0,3449**	No	—
Iris-Plants	0,2931	0,2825	**0,2816**	Si	$1,3 \longrightarrow 2$
Glass	0,5390	0,5282	**0,5117**	Si	$1,3 \longrightarrow 2$
TAE	**0,6409**	0,6657	0,6610	No	—

tancias PSO+K-means obtiene menores valores para el valor analizado. En 3 instancias las diferencias no son significativas estadísticamente. Y sólo en 3 instancias PSO y PSO+K-means se puede afirmar éstos algoritmos son más robustos que K-means, pues las diferencias son estadísticamente significativas.

Para observar con mayor detalle los resultados obtenidos se tomó una instancia por cada medida analizada y a continuación se muestra el gráfico de la comparación de significancia estadística y la distribución de los valores según la mediana.

En la Figura 5.9 se muestra la comparación de los tres algoritmos para la medida Cuantización del Error en la instancia $I23Ba10 - 800$, que determina si las medias son significativas estadísticamente. En este caso podemos observar que no existen diferencias estadísticamente significativas entre PSO+K-means y K-means. Posicionados en el segmento correspondiente a PSO+K-means (coloreado gris oscuro) como hay un solapamiento entre el otro segmento correspondiente a K-means (coloreado gris claro) significa que no hay diferencias significativas entre ellos pero si ambos con respecto a PSO.

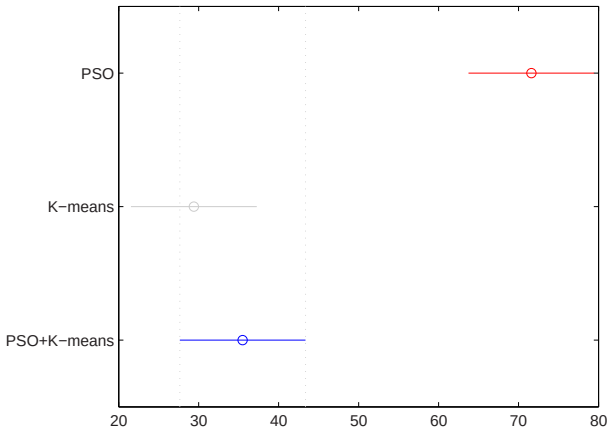


Figura 5.9: Comparación de diferencias estadísticamente significativas entre los algoritmos PSO, K-means y PSO+K-means para la medida Cuantización del Error en la instancia $I23Ba10-800$

Como no existen diferencias estadísticamente significativas entre PSO+K-means y K-means se muestra el diagrama de cajas (Box-Plot) para ver como se distribuyen los datos de cada algoritmo.

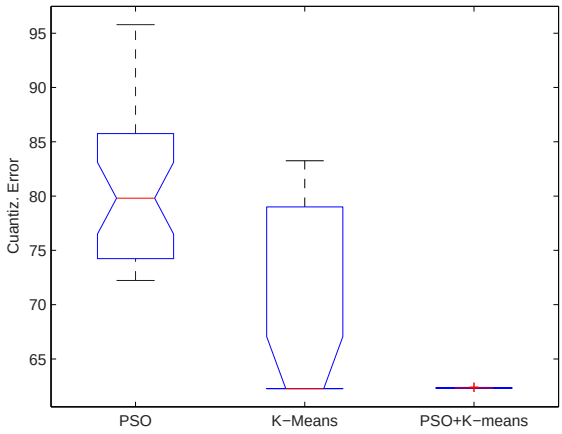


Figura 5.10: Diagrama de cajas de los algoritmos PSO, K-means y PSO+K-means para la Distancia Cuantización del Error en la instancia $I23Ba10-800$

En la Figura 5.10 se muestra el diagrama de cajas (Box-Plot) de los algoritmos PSO, K-means y PSO+K-means para la medida Cuantización del Error en la instancia $I23Ba10-800$, donde podemos observar la distribución de los valores, el valor de la mediana en el centro de la caja. En esta figura vemos que tanto K-means y PSO+K-means presentan medianas similares. Sin

embargo, K-means presenta valores distribuidos en el cuartil superior de la gráfica, es decir valores que superan a la mediana. Por el contrario PSO+K-means, presenta una figura aplanada, lo cual significa que todos los valores son iguales o muy cercanos a la mediana. Se puede afirmar que PSO+K-means para esta medida y en esta instancia es más robusto.

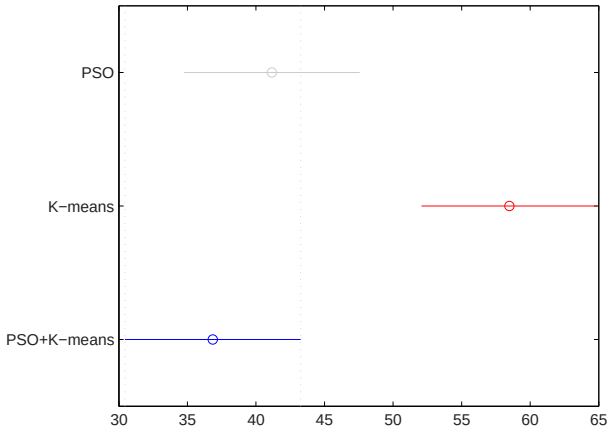


Figura 5.11: Comparación de diferencias estadísticamente significativas entre los algoritmos PSO, K-means y PSO+K-means para la medida Distancia Intra-Cluster en la instancia $I59Bc50 - 1000$

En la Figura 5.11 se muestra la comparación de las medias obtenidas por los tres algoritmos para la medida Distancia Intra-Cluster en la instancia $I59Bc50 - 1000$, determinando si las medias son significativas estadísticamente. En este caso podemos observar que no existen diferencias estadísticamente significativas entre PSO+K-means y PSO, pero si existen diferencias estadísticamente significativas con respecto a K-means. Se puede afirmar que PSO+K-means es mejor que K-means en esta instancia y para esta medida. Veamos que pasa con la distribución de los resultados en la Figura 5.12.

En la Figura 5.12 se muestra el diagrama de cajas (Box-Plot) de los algoritmos PSO, K-means y PSO+K-means para la medida Distancia Intra Cluster en la instancia $I59Bc50 - 1000$, donde podemos observar la distribución de los valores, el valor de la mediana en el centro de la caja y los valores atípicos representados con el signo +. En esta figura vemos que tanto PSO

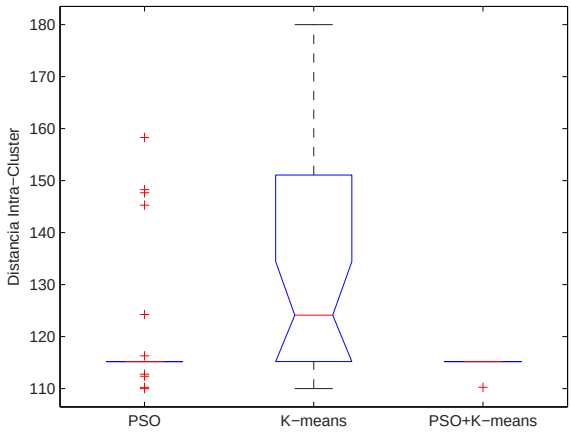


Figura 5.12: Diagrama de cajas de los algoritmos PSO, K-means y PSO+K-means para la Distancia Intra-Cluster en la instancia $I59Bc50 - 1000$

y PSO+K-means presentan medianas similares, no obstante PSO presenta valores atípicos. Por el contrario K-means presenta valores distribuidos tanto en el cuartil superior como en el inferior, es decir valores que superan a la mediana y valores por debajo de la mediana. PSO y PSO+K-means presentan una figura aplanada es decir que todos los valores son iguales a la mediana. Se puede afirmar que PSO+K-means para esta medida y en esta instancia es más robusto que K-means.

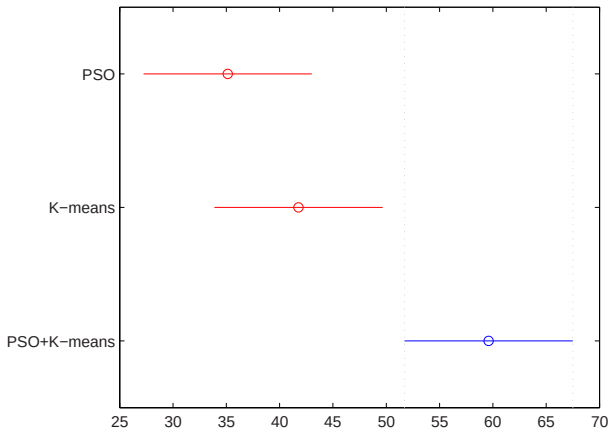


Figura 5.13: Comparación de diferencias estadísticamente significativas entre los algoritmos PSO, K-means y PSO+K-means para la medida Distancia Inter-Cluster en la instancia $I39Bb50 - 800$

En la Figura 5.13 se muestra la comparación de las medias obtenidas por los tres algoritmos para la medida Distancia Inter-Cluster en la instancia $I39Bb50-800$, determinando si las medias son significativas estadísticamente. En este caso podemos observar que existen diferencias estadísticamente significativas entre PSO+K-means con respecto a PSO y a K-means, ya que no hay solapamiento entre ninguno de los segmentos correspondientes a cada algoritmo. Para esta instancia y en esta medida PSO+K-means tiene mejor desempeño que los otros dos algoritmos. Veamos que sucede con la distribución de los resultados en la Figura 5.14.

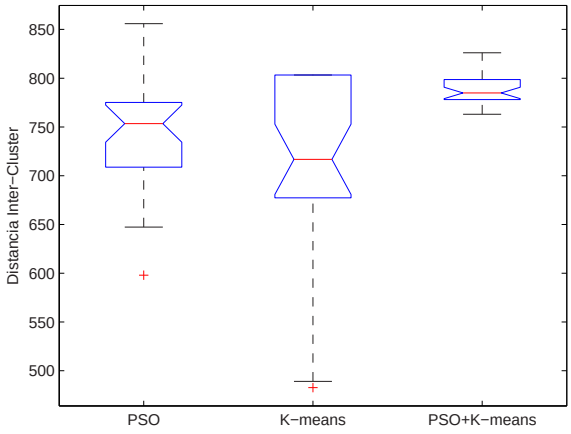


Figura 5.14: Diagrama de cajas de los algoritmos PSO, K-means y PSO+K-means para la Distancia Inter Cluster en la instancia $I39Bb50-800$

Nuevamente en la Figura 5.14 se muestra el diagrama de cajas (Box-Plot) de los tres algoritmos para la medida Distancia Inter-Cluster en la instancia $I39Bb50-800$, donde podemos observar que PSO+K-means presenta una mediana mejor que los otros dos algoritmos, además, una distribución más compacta de los valores restantes. Se puede afirmar que PSO+K-means para esta medida y en esta instancia es más robusto que los otros dos algoritmos.

En la Figura 5.15 se muestra la comparación de las medias obtenidas por los tres algoritmos para el valor $\frac{Intra-Cluster}{Inter-Cluster}$ en la instancia $I24Ba50-800$, determinando si las medias son significativas estadísticamente. En este caso podemos observar que existen diferencias estadísticamente significativas entre

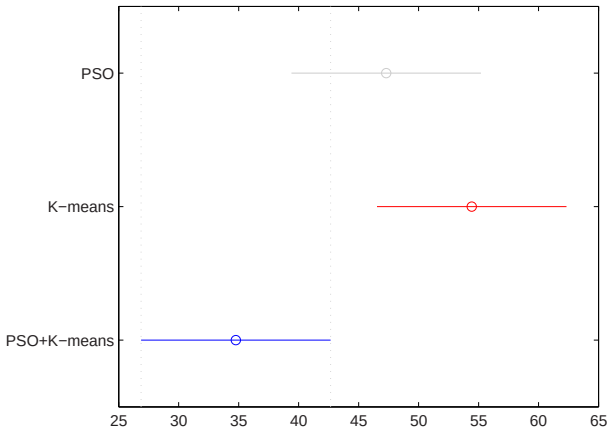


Figura 5.15: Comparación de diferencias estadísticamente significativas entre los algoritmos PSO, K-means y PSO+K-means para el valor $\frac{Intra-Cluster}{Inter-Cluster}$ en la instancia $I24Ba50-800$

PSO+K-means y K-means. En cuanto a la distribución de los resultados para cada algoritmo en la Figura 5.16 se muestra el diagrama de cajas (Box-Plot), donde se puede observar que tanto PSO y PSO+K-means presentan una mediana menor y los restantes valores están más compactos alrededor de la mediana. Por el contrario, K-means presenta una mediana superior a los otros dos algoritmos, aunque existen valores por debajo del cuartil inferior. Se puede decir que para este valor PSO y PSO+K-means obtienen mejores resultados que K-means.

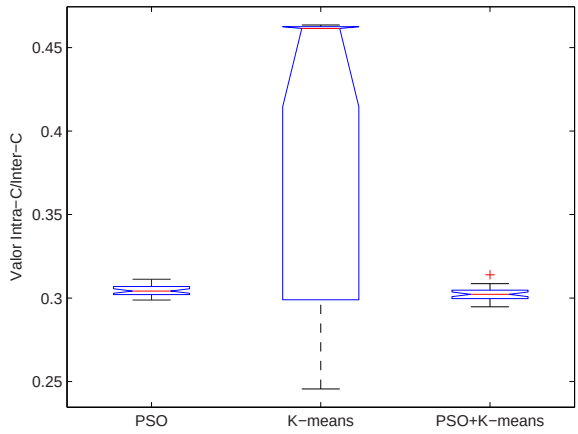


Figura 5.16: Diagrama de cajas de los algoritmos PSO, K-means y PSO+K-means para el valor $\frac{Intra-Cluster}{Inter-Cluster}$ en la instancia $I24Ba50-800$

5.4. Discusión

En este capítulo se presenta un algoritmo híbrido denominado PSO+K-means. PSO es una técnica de optimización que realiza una búsqueda globalizada de las soluciones y puede aplicarse a problemas de *clustering*. K-means es un algoritmo simple, directo y muy utilizado en *clustering* pero puede quedar atrapado en óptimos locales. Se analizaron medidas de calidad de los *clusters* obtenidos por tres algoritmos PSO, K-means y PSO+K-means. Se mostró que PSO+K-means logró encontrar en la mayoría de los casos analizados mejores resultados comparados con los obtenidos por K-means y PSO de manera individual con respecto a la Cuantización del Error, las distancias Inter-Cluster e Intra-Cluster y al valor $\frac{Intra-Cluster}{Inter-Cluster}$. Es decir que este algoritmo híbrido en general logró encontrar *clusters* más cohesionados y separados. Además, en los casos en que las diferencias entre los algoritmos no fueron estadísticamente significativas, PSO+K-means tuvo un comportamiento robusto.

Es importante comentar que si bien el algoritmo propuesto es más robusto que K-means en muchas de las instancias analizadas, este último converge a una solución mucho más rápido que PSO+K-means. Además, como no se ha tenido en cuenta otra de las desventajas de K-means, que es la necesidad de conocer *a priori* el número de clusters existente en el conjunto de datos, a continuación se presentará una extensión a esta propuesta, incorporando la determinación automática del número de clusters.

Capítulo 6

Algoritmo Híbrido para el procesamiento dinámico de tareas de Clustering

En el capítulo anterior se propuso un algoritmo híbrido denominado PSO+K-means que combina dos algoritmos (PSO y K-means) con el objetivo de mejorar la calidad de los clusters encontrados aprovechando las fortalezas de cada uno de ellos. Si bien este algoritmo propuesto es más robusto, obteniendo en general clusters más compactos y separados que los obtenidos por los algoritmos PSO y K-means en forma separada, sigue existiendo la necesidad de conocer *a priori* el número de clusters para que este algoritmo realice el agrupamiento de los elementos. Por esta razón, y teniendo en cuenta los enfoques presentados en [81] y [112] se propone una versión dinámica del algoritmo híbrido denominado PKD-G que se detalla en la siguiente sección.

6.1. PKD-G

El algoritmo propuesto es una extensión del algoritmo presentado en el capítulo anterior, el nombre PKD-G está dado por los algoritmos que utiliza ya que, la letra P corresponde al algoritmo PSO, la letra "K" corresponde a K-means, la letra "D" representa la incorporación de dinamismo en la determinación del número de clusters y por último la letra "G" corresponde al tipo de algoritmo de PSO que se está implementando, que en este caso es un PSO Global.

La idea general de esta propuesta es mantener un cúmulo de partículas, donde

cada partícula representa una solución al problema con un número determinado de clusters. Es decir, que en el mismo cúmulo conviven partículas de diferente tamaño, que representan posibles soluciones al problema. Por ejemplo, la primer partícula contiene 3 clusters, la segunda 5 clusters, la tercera 4 clusters y así hasta finalizar el cúmulo. El cúmulo va moviéndose y las partículas van siguiendo a la que mantiene un mejor valor de aptitud. De igual forma que en la propuesta anterior, se combina el PSO con K-means para lograr una intensificación en el proceso de búsqueda. A continuación se describen las características de esta propuesta para la incorporación del dinamismo:

- Todas las partículas tienen el mismo número máximo de clusters (K_{max}). Cada partícula se representa con un vector de tamaño $K_{max} + K_{max} \times d$ donde el primer término, K_{max} corresponde a valores $0's$ y $1's$ cada uno controla si el centroide correspondiente está activo (es decir, si se lo usa realmente para clasificar datos) o no. El segundo término, $K_{max} \times d$ corresponde a los posibles centroides de los clusters, cada uno $d-$dimensional. La Figura 6.1 representa a una partícula, donde cada elemento del conjunto de datos es de dimensión 2. Supongamos que en este caso $K_{max} = 7$, donde las primeras 7 posiciones representan el estado del centroide (activo o no activo). El resto de las posiciones corresponde a la ubicación de cada centroide. En esta figura la partícula tiene cinco centroides activos.

- Eliminación del número de centroides activos, dentro de una partícula (proceso denominado $Ordena\_y\_Desactiva$). Inicialmente las partículas dentro del cúmulo se generan aleatoriamente y la cantidad de centroides activos dentro de una partícula también es aleatoria, con la restricción de que cada partícula tenga inicialmente activos más del 85 % de los centroides totales. Esta restricción permite que la cantidad de centroides activos al iniciar el proceso sea grande y a medida que se vaya realizando iteraciones del proceso la cantidad de centroides activos se ajuste automáticamente. Para ordenar y desactivar los centroides activos en una partícula se realiza lo siguiente, en cada partícula:

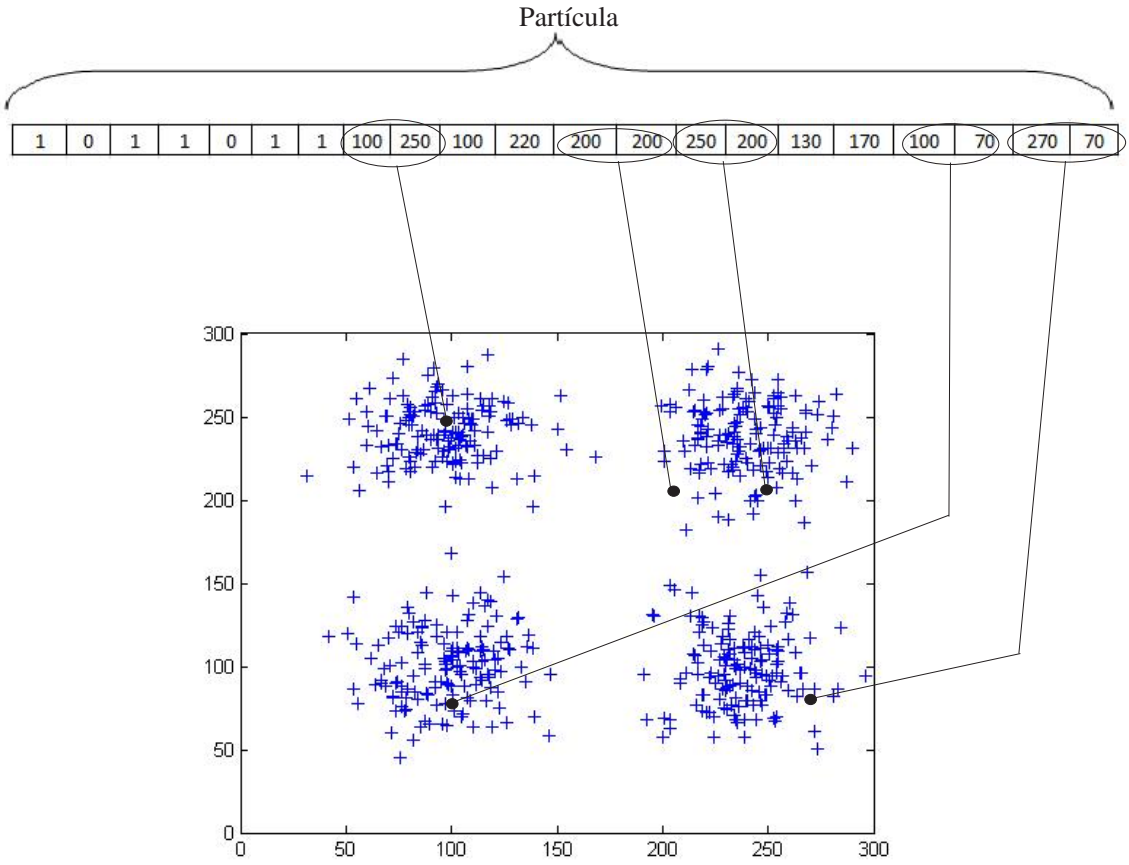


Figura 6.1: Representación de una partícula que contiene cinco centroides activos

- Se ordenan los centroides según la cantidad de elementos que agrupen (de mayor a menor) y se mantienen activos los que acumulen hasta un porcentaje de elementos (fijado en 85 % o 75 %, dependiendo del conjunto de datos).

- El resto de los centroides que están en la cola (es decir, que agrupan pocos elementos) se pasan a no activos.

  Por ejemplo: Supongamos que $K_{max} = 6$ y el conjunto de datos es de 100 elementos.
  En la Figura 6.2 se muestra el conjunto **A** que contiene 6 pares de elementos, donde el primer elemento de cada par corresponde al centroide, y el segundo elemento de ese par representa la cantidad de elementos del conjunto de datos que agrupa ese centroide. Es decir que, el centroide 1 agrupa 45 elementos, el centroide 2 agrupa 2 elementos, el centroide 3 agrupa 1 elemento y así sucesivamente. Luego

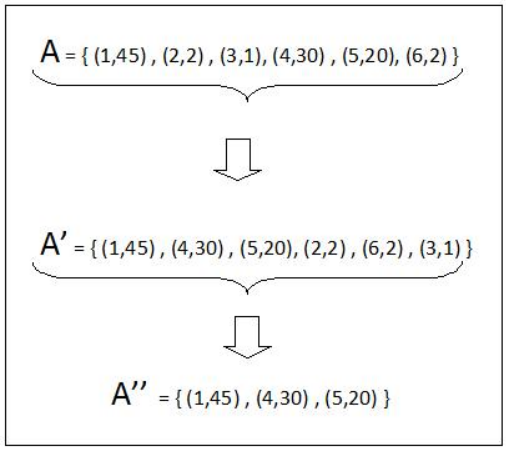


Figura 6.2: Mecanismo para desactivar centroides activos

se ordenan los centroides por cantidad de elementos que agrupa y se obtiene el conjunto $\mathbf{A\prime}$. Seguidamente, de acuerdo al porcentaje que se esté usando para mantener activos (en este ejemplo 85 %), se mantienen activos los 3 primeros centroides (1, 4 y 5) que corresponden al conjunto $\mathbf{A\prime\prime}$. Finalmente, se deben distribuir los elementos que pertenecen a los centroides restantes (2, 6, y 3), pasando éstos centroides a estar inactivos.
Más adelante se explicará los detalles del algoritmo y sus parámetros.

- Partición de super centroides (método denominado Divide_Super_Centroide). Para cada partícula se analiza la existencia de un "super" centroide, se lo denomina de esta manera cuando el centroide agrupa más de un determinado porcentaje de los elementos del conjunto de datos. Cuando esto sucede, este super centroide es modificado de la siguiente manera: se envía a K-means el conjunto de elementos que agrupa y se ejecuta K-means para que busque los dos mejores clusters, es decir $K = 2$ en este caso. Al salir de K-means se reemplaza el super centroide por los dos centroides encontrados por K-means. De esta manera se intenta evitar la aparición de centroides que agrupen la mayoría de los elementos del conjunto.

En el Algoritmo 6 se describe en detalle esta propuesta dinámica denominada PKD-G. El algoritmo es similar al PSO+K-means, excepto en la

inicialización, representación de la partícula y los pasos 8,9 y 20-22, correspondientes a la incorporación del dinamismo.
La función objetivo es la utilizada en PSO+K-means, pero en este caso se tiene en cuenta los clusters activos.
Dentro de las funciones Ordena_y_Desactiva y Divide_Super_Centroide se incluyen ciertos parámetros que serán explicados en detalle en la Sección 6.2.2.

Algoritmo 6 Algoritmo PKD-G

1: Inicializa cada partícula con k centroides seleccionados aleatoriamente dentro del conjunto de datos
2: **for** $t = 1$ to t_{max} {t_{max} es el máximo número de iteraciones} **do**
3: **for** cada partícula x_i **do**
4: **for** cada vector z del conjunto de datos **do**
5: Calcular la distancia Euclideana $d(z, c_j)$ a todos los centroides C_j
6: Asigna z a su *cluster* C_j más cercano tal que la distancia a ese *cluster* sea la mínima.
7: Calcula el fitness usando la ecuación 4.2
8: Ordena_y_Desactiva (partícula)
9: Divide_Super_Centroide (partícula)
10: **end for**
11: Actualiza la mejor posición global y la mejor posición local
12: Actualiza la velocidad y posición de la partícula usando la ecuación 2.7 y 2.2, respectivamente
13: **end for**
14: La mejor partícula encontrada es el vector de centroides iniciales en el proceso de *K-means*
15: **repeat**
16: asigna cada elemento a su *cluster* más cercano
17: recalcula cada centroide $c_j = \frac{1}{|C_j|} \sum_{\forall z \in C_j}$; donde $\mid C_j \mid$ corresponde al número de elementos en el *cluster* C_j
18: **until** Condición de parada sea verdadera
19: Reemplazar la mejor partícula global con los centroides obtenidos por *K-means*
20: **repeat**
21: Divide_Super_Centroide (partícula global)
22: **until** Condición de parada sea verdadera {No existan super centroides}
23: **end for**

6.2. Experimentos y Resultados

Antes de analizar los resultados de la propuesta dinámica se ejecutó el algoritmo PSO+K-means con diferentes números de clusters. Este experimento se realizó a los efectos de determinar el número de clusters adecuados según diferentes índices para medir su calidad. De esta manera se puede estudiar y comparar el desempeño del algoritmo que determina en forma automática el número de clusters. En este caso, se realizan ejecuciones del algoritmo con diferentes números de clusters y luego se analizan los resultados obtenidos aplicando diferentes índices de calidad de clusters. A continuación se presentan los resultados obtenidos por el algoritmo PSO+K-means cuando determina en forma no automática el número de clusters.

6.2.1. Determinación no automática de la cantidad de clusters

Tomando el algoritmo híbrido PSO+K-means con el que se obtuvieron mejores clusters, se lo ejecutó con diferentes números de clusters. Luego los resultados obtenidos se analizaron *a posteriori* con los índices de Dunn's, Davis-Bouldin y el coeficiente de Silhouette que fueron definidos en las Ecuaciones 3.9, 3.10 y 3.8, respectivamente y presentados nuevamente a continuación. Coeficiente de Silhouette Global (GS):

$$GS_u = \frac{1}{K}\sum_{j=1}^{c} S_j$$

donde valores grandes de S_u representan clusters óptimos.
Indice de Dunn's (D):

$$D(U) = \min_{1\leq i\leq K}\left\{\min_{1\leq j\leq K j\neq i}\left\{\frac{\delta(C_i,C_j)}{\max_{1\leq t\leq K}\{\Delta(C_t)\}}\right\}\right\}$$

donde valores grandes de D representan clusters óptimos.
Indice de Davis-Bouldin (DB):

$$DB(U) = \frac{1}{K}\sum_{i=1}^{K}\max_{i\neq j}\left\{\frac{\Delta(C_i)+\Delta(C_j)}{\delta(C_i,C_j)}\right\}$$

en este caso valores pequeños de DB corresponden a clusters óptimos.

Para cada una de las instancias se buscaron $2, 3, 4, 5, 6$ y 7 clusters excepto la instancia Glass que se ejecuto con $3, 4, 5, 6, 7$ y 8 clusters pues esta ultima es la que contiene 6 clusters en su estructura. Se aplicaron a cada uno de los resultados los tres índices (Sihouette, Dunn's y Davis-Bouldin) con el fin de determinar el número de clusters óptimo.
La Tabla 6.1 muestra los resultados promedios de 30 corridas obtenidos para cada una de las instancias con cada uno de los índices. La primer columna corresponde al nombre de la instancia, la segunda columna represente el índice utilizado y las restantes columnas ($K = 2$ a $K = 7$) los resultados obtenidos en esa cantidad de clusters, es decir $K = 2$ significa que se realizó la ejecución del algoritmo con la cantidad de clusters definidas en 2, luego en 3 y así sucesivamente. En la Tabla 6.1 se puede observar que para la ins-

Tabla 6.1: Resultados algoritmo PSO+K-means

Instancia	Indice	K=2	K=3	K=4	K=5	K=6	K=7	K=8
I19Ba50-600	Silhouette	0,484	0,574	**0,657**	0,405	0,265	0,182	–
I19Ba50-600	Dunn's	2,042	1,781	**3,825**	0,375	0,380	0,374	–
I19Ba50-600	Davis-Bouldin	0,975	0,663	**0,530**	2,335	3,259	3,808	–
I43Bb10-1000	Silhouette	0,315	0,371	**0,432**	0,317	0,251	0,215	–
I43Bb10-1000	Dunn's	1,227	1,376	**1,704**	0,646	0,623	0,636	–
I43Bb10-1000	Davis-Bouldin	1,443	1,087	**0,955**	1,471	1,658	1,868	–
I60Bc100-1000	Silhouette	0,518	0,539	0,612	**3,049**	0,275	0,191	–
I60Bc100-1000	Dunn's	2,100	2,045	**2,277**	0,224	0,212	0,195	
I60Bc100-1000	Davis-Bouldin	0,806	0,718	0,722	**0,393**	4,047	4,667	–
Iris-Plants	Silhouette	**0,708**	0,551	0,463	0,407	0,392	0,377	–
Iris-Plants	Dunn's	**3,877**	2,237	1,579	1,327	1,190	1,182	–
Iris-Plants	Davis-Bouldin	**0,397**	0,712	0,841	0,922	0,928	0,955	–
Glass	Silhouette	–	**0,420**	0,364	0,388	0,353	0,332	0,420
Glass	Dunn's	–	**1,542**	0,887	0,703	0,628	0,509	1,542
Glass	Davis-Bouldin	–	**0,949**	1,024	0,951	0,967	1,016	0,949
TAE	Silhouette	**0,369**	0,333	0,341	0,356	0,367	0,366	–
TAE	Dunn's	**1,685**	1,463	1,474	1,520	1,381	1,277	–
TAE	Davis-Bouldin	1,107	1,175	1,104	1,002	0,944	**0,939**	–

tancia $I19Ba50 - 600$ donde la cantidad de clusters es de 4, los tres índices determinan que 4 es el número de clusters óptimo. Lo mismo ocurre para la instancia $I43Bb10 - 1000$ que contiene 4, los tres índices obtienen 4 clusters como la cantidad óptima. Para la instancia $I60Bc100 - 1000$, que contiene

4 clusters, sólo uno de los índices obtiene como número de clusters óptimo, 4. Para la instancia $Iris - Plants$, que contiene 3 clusters, ninguno de los índices obtiene 3 como número de clusters óptimo. Para la instancia *Glass* que contiene 6 clusters, ninguno de los índices obtiene ese valor como número de clusters óptimo, los tres índices obtienen 3 clusters como número óptimo. Y finalmente para la instancia TAE, que contiene 3 clusters, ninguno de los índices obtiene 3 como número de clusters óptimo.
Se ha buscado la cantidad de clusters óptimo, aplicando tres índices de calidad, a continuación se presentarán los resultados obtenidos por la propuesta dinámica, la cual determina en forma automática la cantidad de clusters óptimo.

6.2.2. Determinación automática de la cantidad de clusters usando PKD-G

En esta subsección se mostrarán los resultados obtenidos por la propuesta dinámica. Se analizó la performance de este algoritmo con seis instancias, tres instancias pertenecientes a cada uno de los grupos generados artificialmente ($I19Ba2600, I43Bb101000, I60Bc1001000$) y las tres instancias de las bases de datos reales (Iris-Plants, Glass y TAE). Para cada una de las instancias se realizaron 150 corridas independientes. La asignación de las variables pertenecientes a PSO se mantuvo igual que las utilizadas en el estudio experimental del Capítulo 5. A continuación se muestran los parámetros incorporados en este nuevo algoritmo dinámico y los valores utilizados en cada instancia:

Tabla 6.2: Valores de las instancias y valores de parámetros

Instancia	Dim	Tamaño	Clusters	MaxClu	PT	PA
I19Ba50-600	50	600	4	10	0,85	0,50
I43Bb10-1000	10	1000	4	10	0,85	0,50
I60Bc100-1000	100	1000	4	10	0,75	0,50
Iris-Plants	4	150	3	9	0,85	0,50
Glass	9	214	6	15	0,85	0,50
TAE	5	151	3	10	0,70	0,70

La primer columna corresponde al nombre de la instancia, la segunda

columna (Dim) representa la dimensión de los elementos del conjunto de datos, la tercer columna ($Tamaño$) corresponde a la cantidad de elementos dentro del conjunto de datos, la cuarta columna ($Clusters$) corresponde a la cantidad de clusters existentes en el conjunto de datos, las tres ultimas columnas corresponden a la asignación de valores de los parámetros para la propuesta dinámica. La columna denominada $MaxClu$ representa el número máximo de clusters definidos en cada instancia. La ante-última columna (PT) corresponde al valor utilizado en la siguiente ecuación:

$$agrupamiento\_final = \text{Tamaño} * PT \tag{6.1}$$

Este valor calculado $agrupamiento\_final$ es utilizado para determinar la cantidad de elementos agrupados por los centroides y luego desactivar aquellos centroides que agrupen pocos elementos con el método Ordena_y_Desativa.

En cuanto a la última columna (PA) es el porcentaje utilizado para determinar si un centroide es un "super centroide", es decir, es un centroide que agrupa un porcentaje muy alto de la cantidad de elementos del conjunto. Este valor denominado superCentroide se calcula de la siguiente manera:

$$superCentroide = agrupamiento\_final * PA \tag{6.2}$$

El valor obtenido para $superCentroide$ es utilizado en el método Divide_Super_Centroide, para buscar dos centroides que reemplacen a ese centroide que agrupa muchos elementos del conjunto de datos.

Los valores para PT y PA se fijaron teniendo en cuenta los mejores resultados obtenidos en 30 experimentos realizados.
Los resultados obtenidos por la propuesta dinámica PKD-G se muestran a continuación. Para todas las instancias se realizaron 150 corridas independientes. Las características del hardware y del software utilizado son las mismas que las descriptas para los experimentos del Capítulo 5.

La Tabla 6.3 muestra los resultados obtenidos para la instancia $I19Ba50-600$ donde se puede observar que en 137 corridas se encuentran 4 clusters, es representa el 91 % de las corridas. Recordar que este conjunto de datos pertenece al grupo donde los clusters son 4 y fueron generados con igual cantidad de elementos, globulares y separados (Grupo-A).

Tabla 6.3: Resultados I19Ba50-600

Nro. Clusters	Frecuencia
4	**137**
5	7
6	6

Tabla 6.4: Resultados I60Bc100-1000

Nro. Clusters	Frecuencia
3	93
4	**57**

Tabla 6.5: Resultados I43Ba10-1000

Nro. Clusters	Frecuencia
3	9
4	**115**
5	21
6	5

Tabla 6.6: Resultados TAE

Nro. Clusters	Frecuencia
3	**35**
4	85
5	24
6	6

Tabla 6.7: Resultados Iris-Plants

Nro. Clusters	Frecuencia
3	**48**
4	55
5	33
6	12
9	2

Tabla 6.8: Resultados Glass

Nro. Clusters	Frecuencia
5	10
6	**91**
7	36
8	8
9	4
10	1

La Tabla 6.4 muestra los resultados obtenidos para la base de datos I60Bc100-1000. Se puede observar que en 57 corridas se encuentran 4 clusters, esto representa un 38 % de las corridas. Recordar que este conjunto de datos pertenece al grupo c donde los clusters son 4 y fueron generados con diferente cantidad de elementos, con forma globular y separados.

La Tabla 6.5 muestra los resultados obtenidos para la instancia $I43Bb10-1000$ donde se puede observar que en 115 corridas se encuentran 4 clusters, esto representa el 77 % de las corridas. Recordar que este conjunto de datos pertenece al grupo donde los clusters son 4 y fueron generados con igual cantidad de elementos, con forma elíptica y entremezclados (Grupo-B).

La Tabla 6.6 muestra los resultados obtenidos para la instancia TAE donde se puede observar que en solo 35 corridas se encuentran 3 clusters, esto representa solo 23 % de las corridas.

La Tabla 6.7 muestra los resultados obtenidos para la instancia $Iris -$

Plants donde se puede observar que en 48 corridas se encuentran 3 clusters, esto representa el 32 % de las corridas.

Finalmente, la Tabla 6.8 muestra los resultados obtenidos para la instancia *Glass* donde se puede observar que en 91 corridas se encuentran 6 clusters, esto representa el 61 % de las corridas.

Teniendo en cuenta los resultados obtenidos por PKD-G se tomó una instancia por grupo donde no se obtuvieron los resultados esperados, y se aplicaron los tres índices.

La Tabla 6.9 se muestra el promedio de 30 corridas independientes de los resultados de mayor frecuencia obtenidos en las instancias $I60Bc100-1000$ y TAE.

Tabla 6.9: Resultados Indices Silhouette, Dunn's y Davis-Bouldin

Instancia	Indice	K=3	K=4
I60Bc100-1000	Silhouette	0,5337	**0,6350**
I60Bc100-1000	Dunn's	1,128	**1,875**
I60Bc100-1000	Davis-Bouldin	1,739	**0,656**
TAE	Silhouette	0,300	**0,309**
TAE	Dunn's	1,183	**1,288**
TAE	Davis-Bouldin	1,385	**1,194**

Se puede observar que para la instancia $I60Bc100-1000$, que se aplicaron los tres índices para la cantidad de clusters (3 y 4 los de mayor frecuencia en PKD-G) se obtuvo 4 como el número óptimo de clusters.
Y para la instancia TAE, se puede observar por los resultados obtenidos que los tres índices coinciden en que 4 el número de clusters óptimo, y en realidad esta instancia contiene 3 clusters.

Teniendo en cuenta los resultados obtenidos con este algoritmo propuesto, y entendiendo que se lo puede utilizar como una alternativa en la búsqueda de clusters existentes en un conjunto de datos a continuación se presentan dos variantes de este algoritmo con el fin de mejorar los resultados.

6.3. Variantes de PKD-G

En la sección anterior hemos mostrado una versión de PKD-G, donde el algoritmo de PSO implementado es un PSO global, explicado en la Sección 2.4.1. En este caso el vecindario utilizado para cada partícula es todo el cúmulo.
Con el fin de obtener mejoras en los resultados obtenidos se presentan dos variantes de la propuesta. La primer variante denominada PKD-L utiliza un PSO Local, explicado en la Sección 2.4.2. En este caso el vecindario utilizado es de tamaño 3. Donde los vecinos para la partícula i, serían las partículas en las posiciones $i-1$ e $i+1$.
La segunda variante denominada PKD-GL es una combinación de las dos anteriores, donde existe la misma probabilidad para aplicar PSO Local o PSO Global.

Se ejecutaron ambas variantes con las mismas instancias y la misma asignación de las variables, que se usaron para la versión PKD-G.

La Tabla 6.10 muestra el identificador de la instancia, el número de clusters y luego para cada algoritmo muestra la frecuencia con la que se encuentra esa cantidad de clusters (Hits), el porcentaje que representa el valor de Hits en 150 corridas (%) y la cantidad de evaluaciones realizadas (Evals).

Tabla 6.10: Resultados obtenidos con PKD-G, PKD-L y PKD-GL

		PKD-G			PKD-L			PKD-GL		
Instancia	Clusters	Hits	%	Evals	Hits	%	Evals	Hits	%	Evals
I19Ba50-600	4	137	91	123479	**141**	**94**	121825	**141**	**94**	121776
I43Bb10-1000	4	115	77	146280	**121**	**81**	146930	118	78	148004
I60Bc100-1000	4	**57**	**38**	143429	50	33	138951	**57**	**38**	143417
Iris-Plants	3	**48**	32	130856	32	21	102598	43	29	108340
Glass	6	91	61	172318	**96**	**64**	170139	82	55	136207
TAE	3	35	23	255902	**53**	**35**	254930	42	28	255430

En la Tabla 6.10, se puede observar que si bien los algoritmos tienen un comportamiento bastante similar, el algoritmo PKD-L obtiene las mejores frecuencias en 4 de las 6 instancias ($I19Ba50-600$, $I43Bb10-1000$, $Glass$

y TAE). Y en cuanto a las evaluaciones realizadas, la propuesta PKD-G es la que realiza en promedio el mayor número de evaluaciones, comparada a las evaluaciones realizadas por los otros dos algoritmos.

A continuación se mostrará la distribución de los resultados de tres instancias, a través de un gráfico de cajas. Se tomaron 30 resultados de las corridas realizadas para cada algoritmo.

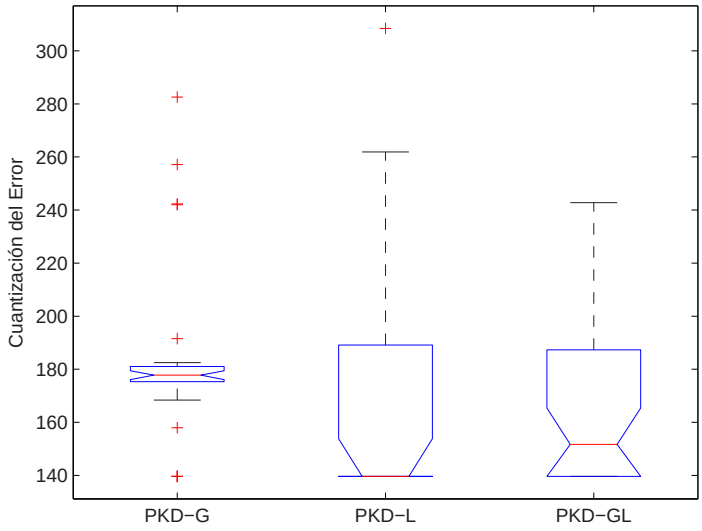


Figura 6.3: Resultados de los algoritmos PKD-G, PKD-L y PKD-GL para la medida Cuantización del Error en la instancia I19Ba50-600

En Figura 6.3 se puede observar que PKD-L obtiene una mejor mediana que los otros dos algoritmos. PKD-G presenta una figura más aplanada pero existen muchos valores atípicos.

Los resultados en la Figura 6.4 no son estadísticamente signaficativos para esta medida pero se puede observar que si bien los tres algoritmos tienen un comportamiento similar, en el cuartil superior hay una pequeña diferencia para PKD-L donde los valores son un poco mas pequeños que los otros dos algoritmos.

Finalmente, en la Figura 6.5 la menor mediana es obtenida por el algoritmo PKD-GL. Aunque presenta un valor atípico, se encuentran varios valores en el cuartil inferior, que son menores a la mediana.

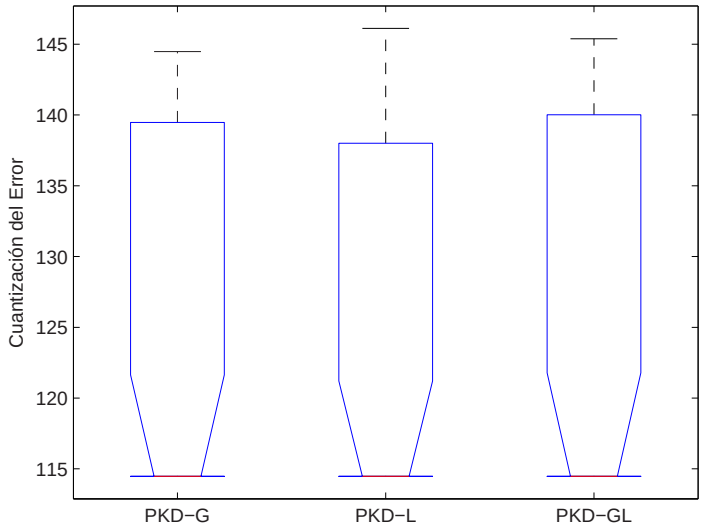


Figura 6.4: Resultados de los algoritmos PKD-G, PKD-L y PKD-GL para la medida Cuantización del Error en la instancia I60Bc100-1000

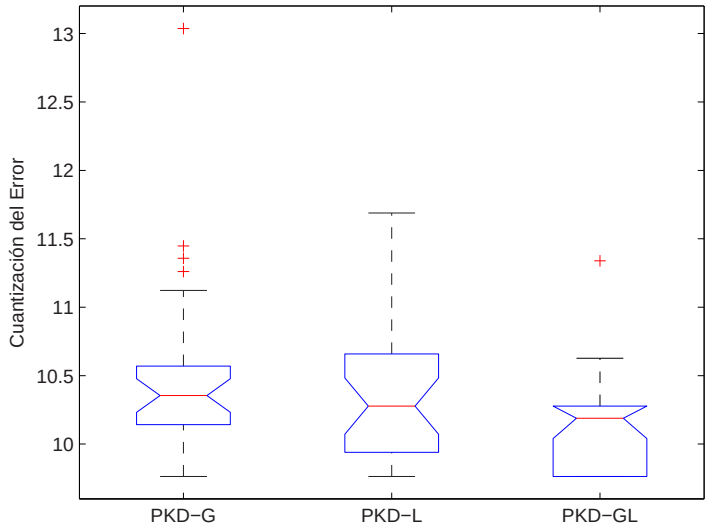


Figura 6.5: Resultados de los algoritmos PKD-G, PKD-L y PKD-GL para la medida Cuantización del Error en la instancia TAE

6.4. Discusión

En este capítulo se ha presentado una propuesta dinámica para la determinación automática del número de clusters denominada PKD-G, que es una extensión del algoritmo PSO+K-means. Antes de realizar un estudio de los resultados obtenidos por esta propuesta, se determina el número de clusters

óptimo a través de los índices de validez de clusters Dunn's, Davis-Bouldin y Silhouette. Seguidamente se analizan los resultados obtenidos por PKD-G y se puede decir que con ciertos conjuntos de datos, globulares y bien separados los resultados son promisorios. Es decir, que se puede utilizar PKD-G para la determinación del número de clusters óptimo. En los casos que los resultados obtenidos por PKD-G no estén bien definidos (cuando no hay marcadas diferencias en la frecuencia de aparición de algún cluster), este algoritmo puede ayudar a reducir la cantidad posible de clusters a verificar con los índices de validez de clusters. En síntesis, esta propuesta representa una alternativa para la determinación de clusters y predicción del número de clusters óptimo. Finalmente, se presentan dos variantes a esta propuesta dinámica denominadas PKD-L y PKD-GL, con el objetivo de mejorar los resultados obtenidos por PKD-G al variar el tipo de algoritmo de PSO utilizado en cada uno de ellos. De los resultados obtenidos con estas variantes se puede decir que las dos variantes propuestas presentan en general resultados más robustos que PKD-G.

Capítulo 7

Conclusiones y Trabajos Futuros

Las metaheurísticas son estrategias de alto nivel que usan diferentes métodos para explorar el espacio de búsqueda. Pueden clasificarse como basadas en trayectorias y basadas en población. Las metaheurísiticas basadas en trayectoria parten de un punto y mediante la exploración del vecindario actualizan la solución actual, generando de esa manera una trayectoria. Las basadas en población, se caracterizan por trabajar con un conjunto de soluciones (población) en cada iteración.

PSO o *Particle Swarm Optimization*, que se suele traducir como optimización por cúmulo de partículas es una metaheurística poblacional. Está basada en modelos psico-sociales respecto a la influencia y al aprendizaje social. Los algoritmos basados en cúmulos de partículas han sido aplicados exitosamente en diferentes campos de investigación y en particular en problemas de optimización.

La minería de datos es una etapa importante del proceso de KDD. Es una técnica que permite buscar información dentro de un conjunto de datos con la finalidad de extraer información nueva y útil que se encuentra oculta en grandes volúmenes de datos.

El clustering es una tarea dentro de la minería de datos y consiste en agrupar un conjunto de objetos abstractos o físicos en clases similares denominadas clusters. Existe un gran número de algoritmos de clustering y se pueden agrupar en métodos particionales, métodos jerárquicos, métodos basados en densidad, métodos basados en grids y basados en modelos.

Un algoritmo de clustering particional es el conocido algoritmo de K-means. Este algoritmo es fácil de implementar y muy eficiente en términos de tiempo

de ejecución. Sin embargo, es sensible a la selección de la partición inicial y puede converger a óptimos locales.

El problema de clustering puede ser visto como un problema de optimización donde se intenta encontrar los centroides óptimos de un cluster en lugar de encontrar la partición óptima. Esta forma de ver el problema permite aplicar un algoritmo de PSO a un problema de clustering.

La hibridación de metaheurísticas es un área promisoria de investigación y se han obtenido exitosas aplicaciones al utilizarla.

En este trabajo se ha presentado un algoritmo híbrido denominado PSO+K-means que combina la habilidad de la búsqueda globalizada de PSO con la búsqueda local de K-means. Se mostró que PSO+K-means logró encontrar en la mayoría de los casos analizados mejores resultados comparados con los obtenidos por K-means y PSO de manera individual con respecto a la cuantización del error, las distancias Inter-Cluster e Intra-Cluster y el valor $\frac{Intra-Cluster}{Inter-Cluster}$. Es decir, que este algoritmo híbrido en general logró encontrar *clusters* más cohesionados y separados. De las instancias analizadas se encontraron diferencias estadísticamente significativas entre PSO+K-means y K-means. Además, al ver la distribución de las medianas se observó que PSO+K-means es más robusto que K-means.

Luego se presentó una variante dinámica denominada PKD-G, que es una extensión del algoritmo PSO+K-means. Del análisis de los resultados obtenidos por PKD-G se puede decir que con ciertos conjuntos de datos, globulares y separados los resultados son promisorios. Es decir, que se puede utilizar PKD-G para la determinación del número de clusters óptimo. En los casos que los resultados obtenidos por PKD-G no estén bien definidos (cuando no hay marcadas diferencias en la frecuencia de aparición de algún cluster), este algoritmo puede ayudar a reducir la cantidad posible de clusters a verificar con los índices de validez de clusters. Finalmente, se presentaron dos variantes a esta propuesta dinámica denominadas PKD-L y PKD-GL, presentando mejoras sobre el enfoque que utiliza PSO Global (PKD-G).

En síntesis, la propuesta dinámica representa una alternativa para la determinación de clusters y predicción de número de clusters óptimo. No obstante, estas propuestas pueden extenderse como se detalla a continuación:

- Realizar un análisis detallado de los algoritmos propuestos teniendo en cuenta el tiempo de computación requerido.
- Incorporar a las propuestas otras métricas de distancia con el fin de analizar si esto provoca una mejora en los resultados obtenidos.
- Disminuir el número de parámetros requeridos en ambas propuestas y definir el mayor número de parámetros de forma automática.
- Probar con diferentes funciones objetivo.

Todas estas acciones podrán ser analizadas en trabajos futuros.

Apéndice A

Anexo

En este anexo se muestra como complemento del capítulo 5 los valores promedio resumen de las 30 corridas independientes para todas las instancias para los algoritmos PSO, K-means y PSO+K-means.

Tabla A.1: Resumen resultados obtenidos con PSO y K-means para las medidas Cuantización del Error, Distancia Intra-Cluster, Distancia Inter-Cluster y valor $\frac{Intra-Cluster}{Inter-Cluster}$ con las Instancias 16 a 30.

	PSO				K-means			
Instancia	C.Error	Intra-C	Inter-C	$\frac{Intra-C}{Inter-C}$	C.Error	Intra-C	Inter-C	$\frac{Intra-C}{Inter-C}$
I16Ba2-600	34,03	43,18	155,50	0,2777	26,16	37,02	159,88	0,2315
I17Ba3-600	41,90	49,04	163,29	0,3003	32,85	46,40	154,46	0,3004
I18Ba10-600	79,28	87,48	317,54	0,2755	71,88	98,68	273,26	0,3611
I19Ba50-600	153,29	198,08	653,83	0,3029	161,65	223,59	593,78	0,3765
I20Ba100-600	233,88	283,11	906,64	0,3123	225,89	313,73	851,53	0,3684
I21Ba2-800	29,98	37,69	153,56	0,2454	27,40	38,47	156,04	0,2465
I22Ba3-800	36,96	46,31	161,74	0,2864	31,94	45,25	159,85	0,2831
I23Ba10-800	81,04	89,65	282,26	0,3176	67,40	93,85	289,76	0,3239
I24Ba50-800	153,91	199,38	654,94	0,3044	161,31	222,10	623,29	0,3563
I25Ba100-800	269,85	284,48	829,82	0,3428	228,81	313,92	902,90	0,3477
I26Ba2-1000	27,18	35,29	155,97	0,2262	24,77	35,05	159,04	0,2204
I27Ba3-1000	35,64	44,06	159,95	0,2755	31,67	44,72	156,74	0,2853
I28Ba10-1000	82,48	90,69	279,81	0,3241	70,73	97,47	270,16	0,3608
I29Ba50-1000	196,71	201,84	601,90	0,3353	163,70	225,32	608,19	0,3705
I30Ba100-1000	278,97	287,76	836,46	0,3440	232,07	319,45	861,37	0,3709

Tabla A.2: Resumen resultados obtenidos con PSO y K-means para las medidas Cuantización del Error, Distancia Intra-Cluster, Distancia Inter-Cluster y valor $\frac{Intra-Cluster}{Inter-Cluster}$ con las Instancias 31 a 60 y las Instancias Iris-plants, Glass y TAE

	PSO				K-means			
Instancia	C.Error	Intra-C	Inter-C	$\frac{Intra-C}{Inter-C}$	C.Error	Intra-C	Inter-C	$\frac{Intra-C}{Inter-C}$
I31Bb2-600	50,75	68,16	164,23	0,4150	50,45	69,20	175,63	0,3940
I32Bb3-600	64,40	87,90	166,85	0,5268	62,27	87,88	175,68	0,5002
I33Bb10-600	180,29	218,88	308,31	0,7099	159,55	225,07	335,02	0,6718
I34Bb50-600	433,38	483,53	727,95	0,6642	366,44	514,78	678,66	0,7585
I35Bb100-600	631,40	704,81	1011,78	0,6966	512,94	724,19	1057,62	0,6847
I36Bb2-800	49,11	66,95	164,28	0,4075	48,72	67,20	174,05	0,3861
I37Bb3-800	64,13	88,27	164,99	0,5350	62,31	87,49	176,50	0,4957
I38Bb10-800	166,84	213,90	316,29	0,6763	156,46	220,65	334,98	0,6587
I39Bb50-800	407,95	481,96	741,84	0,6497	369,07	518,64	713,08	0,7273
I40Bb100-800	636,40	701,12	991,78	0,7069	524,91	740,42	1030,16	0,7187
I41Bb2-1000	50,31	69,54	161,40	0,4309	50,32	69,27	173,84	0,3985
I42Bb3-1000	65,14	88,40	166,17	0,5320	63,46	88,98	177,54	0,5012
I43Bb10-1000	176,35	208,69	304,80	0,6847	151,10	213,17	341,13	0,6249
I44Bb50-1000	436,34	479,96	709,32	0,6767	349,17	492,13	734,44	0,6701
I45Bb100-1000	432,48	482,24	742,47	0,6495	520,89	733,03	1025,10	0,7151
I46Bc2-600	14,61	20,85	74,26	0,2808	15,15	21,20	74,89	0,2831
I47Bc3-600	24,51	28,52	94,28	0,3025	21,00	29,16	93,17	0,3130
I48Bc10-600	53,63	59,20	137,12	0,4317	45,18	62,75	126,91	0,4944
I49Bc50-600	122,38	123,60	304,74	0,4056	107,61	149,82	295,32	0,5073
I50Bc100-600	175,88	172,10	433,36	0,3971	150,16	209,02	432,15	0,4837
I51Bc2-800	17,08	20,99	75,21	0,2792	14,60	20,41	75,37	0,2708
I52Bc3-800	24,80	28,64	94,40	0,3034	20,57	28,48	94,45	0,3016
I53Bc10-800	48,61	52,87	144,61	0,3656	41,97	58,25	141,74	0,4110
I54Bc50-800	119,19	119,69	302,06	0,3963	103,99	144,08	310,88	0,4634
I55Bc100-800	167,84	166,62	439,30	0,3793	140,72	194,97	451,85	0,4315
I56Bc2-1000	17,33	21,03	70,98	0,2962	14,24	20,09	74,68	0,2690
I57Bc3-1000	23,86	28,05	92,46	0,3034	19,84	27,79	93,88	0,2960
I58Bc10-1000	50,05	55,16	139,46	0,3955	38,98	54,67	140,82	0,3883
I59Bc50-1000	118,49	119,64	311,24	0,3844	95,78	133,88	314,64	0,4255
I60Bc100-1000	169,12	168,68	446,02	0,3782	138,56	194,35	426,41	0,4558
Iris-plants	0,72	0,93	3,16	0,2931	0,6519	0,9243	3,2715	0,2825
Glass	1,46	1,97	3,65	0,5390	1,5829	2,2953	4,3421	0,5286
TAE	11,21	14,22	22,19	0,6409	10,1479	13,9629	20,9760	0,6657

Tabla A.3: Resumen resultados obtenidos con PSO+K-means para las medidas Cuantización del Error, Distancia Intra-Cluster, Distancia Inter-Cluster y valor $\frac{Intra-Cluster}{Inter-Cluster}$ con las instancias 16 a 30

	PSO+K-means			
Instancia	C.Error	Intra-C	Inter-C	$\frac{Intra-C}{Inter-C}$
I16Ba2-600	25,61	36,48	158,83	0,2297
I17Ba3-600	31,92	45,37	157,50	0,2880
I18Ba10-600	63,00	88,94	292,87	0,3037
I19Ba50-600	150,25	198,08	656,81	0,3016
I20Ba100-600	232,74	283,11	917,02	0,3087
I21Ba2-800	26,06	37,00	159,35	0,2322
I22Ba3-800	31,67	44,90	160,95	0,2790
I23Ba10-800	62,33	88,32	298,37	0,2960
I24Ba50-800	151,19	199,38	659,56	0,3023
I25Ba100-800	229,14	282,65	918,31	0,3078
I26Ba2-1000	24,76	35,05	159,30	0,2200
I27Ba3-1000	31,02	44,00	158,88	0,2770
I28Ba10-1000	61,78	87,38	296,16	0,2951
I29Ba50-1000	152,76	200,18	660,91	0,3029
I30Ba100-1000	235,16	283,46	911,27	0,3111

Tabla A.4: Resumen resultados obtenidos con PSO+K-means para las medidas Cuantización del Error, Distancia Intra-Cluster, Distancia Inter-Cluster y valor $\frac{Intra-Cluster}{Inter-Cluster}$ con las Instancias 31 a 60 y las Instancias Iris-plants, Glass y TAE

	PSO+K-means			
Instancia	C.Error	Intra-C	Inter-C	$\frac{Intra-C}{Inter-C}$
I31Bb2-600	48,62	67,87	170,75	0,3975
I32Bb3-600	60,60	86,34	171,77	0,5027
I33Bb10-600	150,18	213,11	357,47	0,5962
I34Bb50-600	357,49	476,13	778,86	0,6113
I35Bb100-600	544,98	689,99	1084,68	0,6361
I36Bb2-800	49,11	66,95	164,28	0,4075
I37Bb3-800	61,16	86,25	168,06	0,5132
I38Bb10-800	151,27	214,48	356,83	0,6011
I39Bb50-800	361,26	481,96	788,53	0,6112
I40Bb100-800	546,94	690,73	1084,37	0,6370
I41Bb2-1000	49,44	68,74	163,96	0,4192
I42Bb3-1000	62,30	87,94	170,74	0,5151
I43Bb10-1000	146,40	192,03	445,42	0,4311
I44Bb50-1000	358,86	479,95	783,88	0,6123
I45Bb100-1000	544,19	687,63	1085,23	0,6336
I46Bc2-600	14,49	20,74	73,62	0,2818
I47Bc3-600	19,07	27,09	94,78	0,2859
I48Bc10-600	38,01	53,66	145,59	0,3686
I49Bc50-600	91,06	115,11	339,81	0,3387
I50Bc100-600	145,96	161,76	470,36	0,3439
I51Bc2-800	14,25	20,20	75,69	0,2669
I52Bc3-800	19,56	27,53	95,23	0,2891
I53Bc10-800	34,68	49,14	150,78	0,3259
I54Bc50-800	90,30	115,25	339,88	0,3391
I55Bc100-800	136,10	162,22	477,52	0,3397
I56Bc2-1000	13,98	19,93	74,80	0,2664
I57Bc3-1000	18,08	25,73	96,73	0,2660
I58Bc10-1000	36,64	51,43	147,21	0,3493
I59Bc50-1000	90,11	115,02	334,93	0,3434
I60Bc100-1000	136,52	161,93	469,57	0,3449
Iris-plants	0,6450	0,9227	3,2764	0,2816
Glass	1,44	1,9628	3,8361	0,5117
TAE	9,83	13,7073	20,7376	0,6610

Tabla A.5: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la medida Intra-Cluster en las Instancias 16 a 30

Intra-Cluster					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I16Ba2-600	36,53	37,02	36,48	Si	3$\longrightarrow$ 1,2
I17Ba3-600	49,04	46,40	45,37	Si	3 $\longrightarrow$ 1,2
I18Ba10-600	87,48	98,68	88,94	Si	2,3$\longrightarrow$ 1
I19Ba50-600	198,08	223,59	198,08	No	–
I20Ba100-600	283,11	313,73	232,74	No	–
I21Ba2-800	37,69	38,47	37,00	Si	3$\longrightarrow$ 1
I22Ba3-800	46,31	45,25	44,90	Si	2,3$\longrightarrow$ 1
I23Ba10-800	89,65	93,85	88,32	Si	1,3$\longrightarrow$ 2
I24Ba50-800	89,65	93,83	88,32	Si	1,3$\longrightarrow$ 2
I25Ba100-800	199,38	222,10	199,83	Si	1,3$\longrightarrow$ 2
I26Ba2-1000	35,29	35,05	35,05	Si	2,3$\longrightarrow$ 1
I27Ba3-1000	44,06	44,72	44,00	Si	1,3$\longrightarrow$ 2
I28Ba10-1000	90,69	97,47	87,38	No	–
I29Ba50-1000	201,84	225,32	200,18	Si	1,3$\longrightarrow$ 2
I30Ba100-1000	287,76	319,45	283,46	Si	3$\longrightarrow$ 1,2

Tabla A.6: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la medida Intra-Cluster en las Instancias 31-60 e Instancias Iris-plants, Glass y TAE

Intra-Cluster					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I31Bb2-600	68,16	69,20	67,87	No	–
I32Bb3-600	87,90	87,88	86,34	No	–
I33Bb10-600	218,88	225,07	213,11	Si	3⟶ 1,2
I34Bb50-600	483,53	514,78	476,13	Si	1,3 ⟶ 2
I35Bb100-600	704,81	724,19	689,99	Si	2⟶ 3
I36Bb2-800	66,95	67,20	66,95	No	–
I37Bb3-800	88,27	87,49	86,25	Si	3⟶ 1,2
I38Bb10-800	213,90	220,65	214,48	Si	3⟶ 1,2
I39Bb50-800	481,96	518,64	481,96	Si	1,3 ⟶ 2
I40Bb100-800	701,12	740,42	690,73	Si	1,3⟶ 2
I41Bb2-1000	69,54	69,27	68,74	No	–
I42Bb3-1000	88,40	88,98	87,94	No	–
I43Bb10-1000	208,69	213,17	192,03	Si	3⟶ 1,2
I44Bb50-1000	479,96	492,13	479,95	Si	2,3⟶ 1
I45Bb100-1000	482,24	733,03	687,63	Si	2,3 ⟶ 1
I46Bc2-600	20,85	21,20	20,74	No	–
I47Bc3-600	28,52	29,16	27,09	Si	3⟶ 1,2
I48Bc10-600	59,20	62,75	53,66	Si	3⟶ 1,2
I49Bc50-600	123,60	149,82	115,11	Si	1,3⟶ 2
I50Bc100-600	172,10	209,02	161,76	Si	1,3 ⟶ 2
I51Bc2-800	20,99	20,41	20,20	Si	2,3⟶ 1
I52Bc3-800	28,64	28,48	27,53	Si	3⟶ 1
I53Bc10-800	52,87	58,25	49,14	Si	1,3⟶ 2
I54Bc50-800	119,69	144,08	115,25	Si	1,3 ⟶ 2
I55Bc100-800	166,62	194,97	162,22	Si	1,3⟶ 2
I56Bc2-1000	21,03	20,09	19,93	Si	2,3⟶ 1
I57Bc3-1000	28,05	27,79	25,73	Si	3⟶ 1,2
I58Bc10-1000	55,16	54,67	51,43	No	–
I59Bc50-1000	119,64	133,88	115,02	Si	1,3 ⟶ 2
I60Bc100-1000	168,68	194,35	161,93	Si	1,3⟶ 2
Iris-plants	0,93	0,92	0,92	No	–
Glass	1,97	2,30	1,96	Si	1,3⟶ 2
TAE	14,22	13,96	13,71	Si	3⟶ 1,2

Tabla A.7: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la medida Inter-Cluster en las Instancias 16 a 30

Inter-Cluster					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I16Ba2-600	155,50	159,88	158,83	Si	1,3 $\longrightarrow$ 2
I17Ba3-600	163,29	154,46	157,50	Si	1,3 $\longrightarrow$2
I18Ba10-600	317,54	273,26	292,87	Si	1,3 $\longrightarrow$ 2
I19Ba50-600	653,83	593,78	656,81	No	–
I20Ba100-600	906,64	851,53	917,02	No	–
I21Ba2-800	153,56	156,04	159,35	Si	1,3 $\longrightarrow$ 2
I22Ba3-800	161,74	159,85	160,95	Si	1,3 $\longrightarrow$ 2
I23Ba10-800	282,26	289,76	298,37	Si	2,3 $\longrightarrow$ 1
I24Ba50-800	654,94	623,29	659,56	Si	2$\longrightarrow$3
I25Ba100-800	829,82	902,90	918,31	Si	2,3 $\longrightarrow$ 1
I26Ba2-1000	155,97	159,04	159,30	Si	3 $\longrightarrow$ 1,2
I27Ba3-1000	159,95	156,74	158,88	Si	1,3 $\longrightarrow$ 2
I28Ba10-1000	279,81	270,16	296,16	Si	3$\longrightarrow$ 1,2
I29Ba50-1000	601,90	608,19	660,91	Si	3$\longrightarrow$ 1,2
I30Ba100-1000	836,46	861,37	911,27	Si	2,3 $\longrightarrow$ 1

Tabla A.8: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para la medida Inter-Cluster en las Instancias 31-60 e Instancias Iris-plants, Glass y TAE

Inter-Cluster					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I31Bb2-600	164,23	175,63	170,75	Si	1,3 $\longrightarrow$2
I32Bb3-600	166,85	175,68	171,77	Si	1,3 $\longrightarrow$ 2
I33Bb10-600	308,31	335,02	357,47	Si	3 $\longrightarrow$1,2
I34Bb50-600	727,95	678,66	778,86	Si	3 $\longrightarrow$ 2
I35Bb100-600	1011,78	1057,62	1084,68	Si	1 $\longrightarrow$ 2
I36Bb2-800	164,28	174,05	164,28	Si	1,3 $\longrightarrow$ 2
I37Bb3-800	164,99	176,50	168,06	Si	1,3 $\longrightarrow$ 2
I38Bb10-800	316,29	334,98	356,83	Si	2,3 $\longrightarrow$1
I39Bb50-800	741,84	713,08	788,53	Si	3 $\longrightarrow$1,2
I40Bb100-800	991,78	1030,16	1084,37	No	–
I41Bb2-1000	161,40	173,84	163,96	Si	1,3 $\longrightarrow$ 2
I42Bb3-1000	166,17	177,54	170,74	Si	1,3 $\longrightarrow$2
I43Bb10-1000	304,80	341,13	445,42	Si	3 $\longrightarrow$ 1,2
I44Bb50-1000	709,32	734,44	783,88	Si	2,3 $\longrightarrow$1
I45Bb100-1000	742,47	1025,10	1085,23	Si	2,3 $\longrightarrow$ 1
I46Bc2-600	74,26	74,89	73,62	Si	1,3 $\longrightarrow$ 2
I47Bc3-600	94,28	93,17	94,78	No	–
I48Bc10-600	137,12	126,91	145,59	Si	3 $\longrightarrow$ 1,2
I49Bc50-600	304,74	295,32	339,81	Si	3$\longrightarrow$1,2
I50Bc100-600	433,36	432,15	470,36	Si	1,2 $\longrightarrow$3
I51Bc2-800	75,21	75,37	75,69	Si	3 $\longrightarrow$1,2
I52Bc3-800	94,40	94,45	95,23	No	–
I53Bc10-800	144,61	141,74	150,78	Si	3$\longrightarrow$2
I54Bc50-800	302,06	310,88	339,88	Si	3 $\longrightarrow$ 1,2
I55Bc100-800	439,30	451,85	477,52	Si	3 $\longrightarrow$1
I56Bc2-1000	70,98	74,68	74,80	Si	3 $\longrightarrow$1,2
I57Bc3-1000	92,46	93,88	96,73	Si	2,3 $\longrightarrow$1
I58Bc10-1000	139,46	140,82	147,21	Si	3 $\longrightarrow$ 1
I59Bc50-1000	311,24	314,64	334,93	No	–
I60Bc100-1000	446,02	426,41	469,57	No	–
Iris-plants	3,16	3,2715	3,2764	Si	1,3 $\longrightarrow$ 2
Glass	3,65	4,3421	3,8361	Si	1,3 $\longrightarrow$ 2
TAE	22,19	20,9760	20,7376	No	–

Tabla A.9: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para el Valor $\frac{Intra-Cluster}{Inter-Cluster}$ en las Instancias 16 a 30

$\frac{Intra-Cluster}{Inter-Cluster}$					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I16Ba2-600	0,2777	0,2315	0,2297	Si	1,3$\longrightarrow$ 2
I17Ba3-600	0,3003	0,3004	0,2880	Si	3$\longrightarrow$1,2
I18Ba10-600	0,2755	0,3611	0,3037	Si	1,3$\longrightarrow$ 2
I19Ba50-600	0,3029	0,3765	0,3016	No	–
I20Ba100-600	0,3123	0,3684	0,3087	No	–
I21Ba2-800	0,2454	0,2465	0,2322	Si	1,3$\longrightarrow$ 2
I22Ba3-800	0,2864	0,2831	0,2790	Si	1,3$\longrightarrow$ 2
I23Ba10-800	0,3176	0,3239	0,2960	Si	2,3$\longrightarrow$ 1
I24Ba50-800	0,3044	0,3563	0,3023	Si	2$\longrightarrow$ 3
I25Ba100-800	0,3428	0,3477	0,3078	Si	2,3$\longrightarrow$ 1
I26Ba2-1000	0,2262	0,2204	0,2200	Si	3$\longrightarrow$ 1,2
I27Ba3-1000	0,2755	0,2853	0,2770	Si	1,3$\longrightarrow$ 2
I28Ba10-1000	0,3241	0,3608	0,2951	Si	3$\longrightarrow$ 1,2
I29Ba50-1000	0,3353	0,3705	0,3029	Si	3$\longrightarrow$ 1,2
I30Ba100-1000	0,3440	0,3709	0,3111	Si	3$\longrightarrow$ 1,2

Tabla A.10: Resultados promedios obtenidos con PSO, K-means y PSO+K-means para el Valor $\frac{Intra-Cluster}{Inter-Cluster}$ en las Instancias 31-60 e Instancias Iris-plants, Glass y TAE

$\frac{Intra-Cluster}{Inter-Cluster}$					
Instancia	PSO	K-means	PSO+K-means	Significativas	Entre-Quién
I31Bb2-600	0,4150	0,3940	0,3975	Si	1⟶ 2
I32Bb3-600	0,5268	0,5002	0,5027	Si	1⟶ 2
I33Bb10-600	0,7099	0,6718	0,5962	Si	3⟶ 1,2
I34Bb50-600	0,6642	0,7585	0,6113	Si	3⟶ 1,2
I35Bb100-600	0,6966	0,6847	0,6361	No	–
I36Bb2-800	0,4075	0,3861	0,4075	Si	1,3⟶ 2
I37Bb3-800	0,5350	0,4957	0,5132	Si	1⟶ 2
I38Bb10-800	0,6763	0,6587	0,6011	Si	3⟶ 1
I39Bb50-800	0,6497	0,7273	0,6112	Si	3⟶ 1,2
I40Bb100-800	0,7069	0,7187	0,6370	No	–
I41Bb2-1000	0,4309	0,3985	0,4192	Si	1,3⟶ 2
I42Bb3-1000	0,5320	0,5012	0,5151	Si	1,3⟶ 2
I43Bb10-1000	0,6847	0,6249	0,4311	Si	3⟶ 1,2
I44Bb50-1000	0,6767	0,6701	0,6123	Si	2,3⟶ 1
I45Bb100-1000	0,6495	0,7151	0,6336	No	–
I46Bc2-600	0,2808	0,2831	0,2818	No	–
I47Bc3-600	0,3025	0,3130	0,2859	No	–
I48Bc10-600	0,4317	0,4944	0,3686	Si	3⟶ 1,2
I49Bc50-600	0,4056	0,5073	0,3387	Si	3⟶1,2
I50Bc100-600	0,3971	0,4837	0,3439	Si	3⟶1,2
I51Bc2-800	0,2792	0,2708	0,2669	No	–
I52Bc3-800	0,3034	0,3016	0,2891	No	–
I53Bc10-800	0,3656	0,4110	0,3259	Si	3⟶ 1,2
I54Bc50-800	0,3963	0,4634	0,3391	Si	3⟶ 1,2
I55Bc100-800	0,3793	0,4315	0,3397	Si	3⟶1,2
I56Bc2-1000	0,2962	0,2690	0,2664	Si	3⟶ 1
I57Bc3-1000	0,3034	0,2960	0,2660	Si	3⟶ 1
I58Bc10-1000	0,3955	0,3883	0,3493	Si	3⟶ 1
I59Bc50-1000	0,3844	0,4255	0,3434	No	–
I60Bc100-1000	0,3782	0,4558	0,3449	No	–
Iris-plants	0,2931	0,2825	0,2816	Si	1,3⟶ 2
Glass	0,5390	0,5286	0,5117	Si	1,3⟶ 2
TAE	0,6409	0,6657	0,6610	No	–

Bibliografía

[1] Bandura A. *Social Fundations of Thought and Action: A Social Cognitive Theory*. Prentice Hall, 1986.

[2] Carlisle A. and Dozier G. An off-the-shelf pso. In *Workshop on Particle Swarm Optimization*, pages 1–6, 2001.

[3] Johnson R. A. and Wichern D. W. *Applied multivariate statistical analysis*. Prentice Hall, 5th edition, 2002.

[4] Likas A., Vlassis N., and Verbeek J.J. The global k-means clustering algorithm. *Pattern Recognition*, (451-461), 2003.

[5] Lorette A., Descombes X., and Zerubia J. Fully unsupervised fuzzy clustering with entropy criterion. In *International Conference on Pattern Recognition*, volume 3, pages 3998–4001, 2000.

[6] Lumini A. and Nanni L. Rapid and brief communication: A clustering method for automatic biometric template selection. *Pattern Recogn.*, 39(3):495–497, 2006.

[7] Ratnaweera A., Halgamuge S., and Watson H. Particle swarm optimization with self-adaptive acceletarion coefficients. In *First International Conference on Fuzzy Systems and Knowledge Discovery*, number 264-268, 2003.

[8] Ultsch A. Emergence in self-organizing feature maps. In *Workshop on Self-Organizing Maps*, 2007.

[9] Xie A. and Beni G. Validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Machine Learning*, 3:841–846, 1991.

[10] Jain A.K. and Dubes R.C. *Algorithms for Clustering Data*. Englewood Cliffs, N.J.:Prentice Hall, 1988.

[11] Calinski R. B. and Harabasz J. Adendrite method for cluster analysis. *Commun Statistics*, pages 1–27, 1974.

[12] Zhang B. Generalized k-harmonic means - boosting in unsupervised learning. Technical report, Hewlett-Packard Labs, 2000.

[13] Zhang B., Hsu M., and Dayal U. K-harmonic means - a data clustering algorithm. Technical report, Hewlett-Packard Labs, 1999.

[14] Blum C. and Roli A. Metaheuristic in combinatorial optimization: Overview and conceptual algorithms comparation. *ACM Computing Survey*, 35(3):268–308, 2003.

[15] Ourique C., Biscaia E., and Pinto J. The use of particle swarm optimization for dynamical analysis in chemical processes. In *Computers and Chemical Engineering*, volume 26, pages 1783–1793, 2002.

[16] Rosenberger C. and Chehdi K. Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation. In *15th International Conference on Pattern Recognition*, volume 1, pages 1656–1659, 2000.

[17] Murthy C.A. and Chowdhuryb N. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17:825–832, 1996.

[18] Chou C.H., Su M.C., and Lai E. A new cluster validity measure and its application to image compression. *Pattern Analysis and Applications*, 7(2):205–220, 2004.

[19] Wallace C.S. and Dowe D.L. Intrinsic classification by mml - the snob program. In *7th Australian Joint Conference on Artificial Intelligence*, pages 37–44, 1994.

[20] Wallace C.S. and Boulton D.M. An information measure for classification. *The Computer Journal*, 11:185–194, 1968.

[21] Zahn C.T. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, C-20:68–86, 1971.

[22] Lee C.Y. and Antonsson E.K. Dynamic partitional clustering using evolution strategies. In *Third Asia Pacific Conference on Simulated Evolution and Learning*, pages 25–27.

[23] Gordon A. D. *Classification*. FL: Chapman and Hall/CRC, 2nd edition, 1999.

[24] Guo D., Peuquet D., and Gahegan M. Opening the black box: interactive hierarchical clustering for multivariate spatial patterns. In *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 131–136, New York, NY, USA, 2002. ACM.

[25] Pelleg D. and Moore A. X-means: Extending k-means with efficient estimation of the number of clusters. In *17th International Conference on Machine Learning*, pages 727–734, 2000.

[26] Steinley D. Local optima in k-means clustering: What you don't know may hurt you. *Psychological Methods*, (8):294–304, 2003.

[27] Davies D.L. and Bouldin D.W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224–227, 1979.

[28] Pham D.T., Dimov S.S., and Nguyen C.D. An incremental k-means algorithm. *Institution of Mechanical Engineers, Part C: J. mechanical Engineering Science*, 218(7):783–795, 2004.

[29] Alba E. *Parallel Metaheuiristics: A new class of algorithm*. Wiley Interscience, 2005.

[30] Bonabeau E., Dorigo M., and Theraulaz G. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press Inc., 1999.

[31] Falkenauer E. *Genetic Algorithms and Grouping Problems*. John Wiley and Son, Chichester, 1998.

[32] Falkenauer E. and Marchand A. Using k-means? consider arrayminer. In *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, pages 25–28, 2001.

[33] Lumer E. and Faieta B. Diversity and adaptation in populations of clustering ants. In *Third International Conferenceon Simulation of Adaptive Behaviour: from Animals to Animates 3*, pages 499–508. MIT press, Cambridge, 1994.

[34] Lumer E. and Faieta B. Exploratory database analysis via self-organization. Unpublished manuscript, 1995.

[35] Parsopoulos K. E. and Vrahatis M. N. Particle swarm optimizer in noisy and continuosly changing environments. In *Proceedings of IASTED International Conference on Artificial Intelligence and Soft Computing*, pages 289–294, 2001.

[36] Della Croce F. and T´kindt V. A recovering beam search algorithm for the one machine dynamic total completation time scheduling problem. *Operational Research Society*, 53(11):1275–1280, 2002.

[37] Glover F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

[38] Glover F. and Kochenberger G. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Norwell, MA, 2002.

[39] Ball G. and Hall D. A clustering technique for summarizing multivariate data. *Behavioral Science*, 12:153–155, 1967.

[40] Gudise G. and Venayagamoorthy G. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *IEEE Swarm Intelligence Symposium*, pages 110–117, 2003.

[41] Hamerley G. and Elkan C. Alternatives to the k-means algorithm that find better clustering. In *International Conference on Information and Knowledge Management ACM*, 2002.

[42] Hamerly G. and Elkan C. Learning the k in k-means. In *Seven-teenth Annual Conference on Neural Information Processing Systems*, pages 281–288, 2003.

[43] Frigui H. and Krishnapuram R. A robust competitive clustering algorithm with applications in computaer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, 1999.

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

[45] Yoshida H., Fukuyama Y., Takayama S., and Nakanishi Y. A particle swarm optimization for reactive power voltage control in electric power systems considering voltage security assessment. *IEEE International Conference on Systems, Man and Cybernetics*, 6:497, October 1999.

[46] Gath I. and Geva A. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–781, 1989.

[47] Rechenberg I. *Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution.* Blackwell Scientific Publishing, Oxford, UK, 1973.

[48] Denzinger J. and Offerman T. On cooperation between evolutionary algorithms and other search paradigms. In *Congress on Evolutionary Computation*, pages 2317–2324, 1999.

[49] Han J. and Kamber M. *Data mining: concepts and techniques.* Morgan Kaufmann, 2001.

[50] Handl J. and Meyer B. Improved ant-based clustering and sorting in a document retrieval interface. In *Parallel Problem Solving from Nature*, volume 2439, pages 913–923. Springer Berlin Heidelberg, 2002.

[51] Handl J., Knowles J., and Dorigo M. Ant-based clustering: A comparative study of its relative importance with respect to k-means, average link and ld-som. Technical report, Universite Libre de Bruxelles, Belgium, 2003.

[52] Kennedy J. and Eberhart R. C. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[53] 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.

[54] Lattin J., Carroll J. D., and Green P. E. *Analyzing multivariate data*. Pacific Grove, CA: Thomson: Brooks/Cole., 2003.

[55] MacQueen J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symp. Math. Statist, Prob*, pages 281–297, 1968.

[56] Mao J. and Jain A.K. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Trans. Neural Network*, 6:296–317, 1995.

[57] Oliver J. and Hand D. Introduction to minimum encoding inference. Technical Report 94/205, Department of Computer Science, 1994.

[58] Peng J., Chen Y., and Eberhart R.C. Battery pack state of charge estimator design using compuational intelligence approaches. In *Annual Battery Conference on Applications and Advances*, pages 173–177, 2000.

[59] Pérez J., Henriques M.F., Pazos R., Cruz L., Reyes G., Salinas J., and Mexicano A. Mejora al algoritmo de agrupamiento k-means mediante un nuevo criterio de convergencia y su aplicación a bases de datos poblacionales de cáncer. In *II Taller Latino Iberoamericano de Investigación de Operaciones*, 2007.

[60] Rousseeuw P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comp App. Math*, 20:53–65, 1987.

[61] Vesanto J. and Alhoniemi E. Clustering of the self organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.

[62] Hartigan J.A. *Clustering Algorithms*. New York: Wiley, 1975.

[63] Bezdek J.C. Numerical taxonomy with fuzzy sets. *Journal of Math. Biol.*, (157-171), 1974.

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

[65] Dunn J.C. Well separated clusters and optimal fuzzy partitions. *J. Cybern*, 4:95–104, 1974.

[66] Deneubourg J.L., Goss S., Franks N., Sendova-Franks A., Detrain C., and Chetien L. The dynamics of collective sorting: Robot-like ants and ant-like robots. In Meyer J.A. and Wilson S.W., editors, *Proceedings of the First International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 1*, number 356-363. MIT Press, Cambridge, 1991.

[67] Peña J.M., Lozano J.A., and Larrañaga P. An empirical comparison of four initialization methods for the k-means algorithm, 1999.

[68] Blekas K., Nikou C., Galatsanos N., and Tsekos N. Curve clustering with spatial constraints for analysis of spatiotemporal data. In *ICTAI '07: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pages 529–535, Washington, DC, USA, 2007. IEEE Computer Society.

[69] Hoe K., Lai W., and Tai T. Homogenous ants for web document similarity modeling and categorization. In Dorigo M., Dicaro G.A., and M. Sampels, editors, *Ant Algorithms. LNCS*, volume 2463, pages 256–261. Springer Berlin Heidelberg, 2002.

[70] Huang K. A synergistic automatic clustering technique (syneract) for multispectral image analysis. *Photogrammetric Engineering and Remote Sensing*, 1(1):33–40, 2002.

[71] Krishna K. and Murty M.N. Genetic k-means algorithm. *IEEE Transaction on Systems, Man and Cybernetics*, 29, 1999.

[72] Mehrotra K., Mohan C., and Ranka S. *Elements of Artificial Neural Networks*. MIT Press, 1997.

[73] Parsopoulos K., Papageorgiou E., Groumpos P., and Vrahatis M. A first study of fuzzy cognitive maps learning using particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pages 1440–1447, 2003.

[74] Fogel L., Owens J., and Walsh M. *Artificial Intelligence Through Simulated Evolution*. 1966.

[75] Clerc M. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. *IEEE Congress on Evolutionary Computation*, 3:1951–1957, July 1999.

[76] Clerc M. and Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[77] Dorigo M. *Optimization, Learning and Natural Algorithms*. PhD thesis, Dipartamento di Elettornica, Politecnico di Milano, 1992.

[78] Halkidi M., Batistakis Y., and Vazirgiannis M. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[79] Meila M. and Heckerman D. An experimental comparison of several clustering and initialization methods. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 386–39, San Francisco, CA, 1998. Morgan Kaufmann.

[80] Omran M., Salman A., and Engelbrecht A. Image classification using particle swarm optimization. In *4th Asia-Pacific Conference on Simulated Evolution and Learning*, pages 370–374, 2002.

[81] Omran M., Engelbrecht A.P., and Salman A. Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligenge*, 19(3):297–322, 2005.

[82] Sarkar M., Yegnanarayana B., and Khemani D. A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters*, 18:975–986, 1997.

[83] Su M.C. and Chou C.H. A modified version of the k-means algorithm with distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):674–680, 2001.

[84] Pakhira M.K., Bandyopadhyay S., and Maulik U. Validity index for crisp and fuzzy clusters. *Pattern Recognition Letters*, 37:487–501, 2004.

[85] Mladenovic N. and Hansen P. Variable neighborhood search. *Computers Oper. Res*, 24:1097–1100, 1997.

[86] Monmarche N., Slimane M., and Venturini G. Ant class: discovery of clusters in numeric data by a hybridization of ant colony with k means algorithm. Technical report, Internal Report No.213, E3i, Laboratoire dÍnformatique, Universite de Tours, 1999.

[87] Loris Nanni. Cluster-based pattern discrimination: A novel technique for feature selection. *Pattern Recogn. Lett.*, 27(6):682–687, 2006.

[88] Pal N.R., Bezdek J.C., and Tsao E.C. Generalizad clustering networks and kohonen's self-organizing scheme. *IEEE Trans. Neural Network*, 4:594–557, 1993.

[89] Duda R. O., Hart P. E., and Stork D. G. *Pattern recognition.* New York: Wiley, 2001.

[90] Kuntz P. and Snyers D. Emergent colonization and graph partitioning. In *Third international conference on Simulation of adaptive behavior : from animals to animats 3: from animals to animats 3*, pages 494 – 500. MIT Press Cambridge, MA, USA, 1994.

[91] Kuntz P. and Snyers D. New results on an ant-based heuristic for highlighting the organization of large graphs. In *Congress on Evolutionary Computation*, pages 1451–1458. IEEE Press, Piscataway, 1999.

[92] Kuntz P., Snyers D., and Layzell P. A stochastic heuristic for visualizing graph clusters in a bi-dimensional space prior to partitioning. *Journal of Heuristics*, 5(3):327–351, 1998.

[93] Tan P., Steinbach M., and Kumar V. *Introduction to Data Mining.* Addison Wesley, 2006.

[94] Kanade P.M. and Hall L.O. Fuzzy ants as a clustering concept. In *22nd International Conference of North American Fuzzy Information Processing Society*, number 227-232, 2003.

[95] Anderberg M. R. *Cluster analysis for applications.* New York: Academic Press, 1973.

[96] Brits R., Engelbrecht A.P., and van den Bergh F. A niching particle swarm optimizer. In *Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning*, pages 692–696, 2002.

[97] Brits R., Engelbrecht A.P., and van den Bergh F. Solving systems of unconstrained equations using particle swarm optimization. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 3, pages 102–107, 2002.

[98] Kass R. and Wasserman L. A reference bayesian test for nested hypotheses and its relatioship to the schwarz criterion. *Journal of the American Statistical Association*, 90(431):928–934, 1995.

[99] Srikanth R., George R., Warsia N., Prabhu D., Petry F. E., and Buckles B. P. A variable-length genetic algorithm for clustering and classification. *Pattern Recognition Letters*, 16:789–800, 1995.

[100] Storn R. and Price K. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, TR-95-012, ICSI, 1995.

[101] Storn R. and Price K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Global Optimization*, 11(4):341–359, 1997.

[102] Dubes R.C. *Handbook of Pattern Recognition and Computer Vision - Cluster Analysis and Related Issues.* World Scientific Publishing Co, 1993.

[103] Eberhart R.C., Simpson P.K., and Dobbins R.W. *Computational Intelligence PC Tools*. Academic Press Proffesional, first edition, 1996.

[104] Eberhart R.C. and Shi Y. Comparising inertia weights and constriction factors in particle swarm optimization. In *Congress on Evolutionary Computing*, volume 1, pages 84–88, 2000.

[105] Turi R.H. *Clustering-based colour image segmentation*. PhD thesis, Monash University Australia, 2001.

[106] Ng R.T. and Han J. Efficient and effective clustering methods for spatial data mining. In *Proc. 20th Int'l Conf. Very Large Databases*, pages 144–155, Sept. 1994.

[107] Bandyopadhyay S., Murthy C.A., and Pal S.K. Pattern classification with genetic algorithms. *Pattern Recognition Letters*, 16:801–808, 1995.

[108] Bandyopadhyay S., Murthy C.A., and Pal S.K. Pattern classification using genetic algorithms: determination of h. *Pattern Recognition Letters*, 19:1171–1181, 1998.

[109] Bandyopadhyay S., Murthy C.A., and Pal S.K. Theoretic performance of genetic pattern classifier. *Journal of Franklin Institute*, 336:387–422, 1999.

[110] Bandyopadhyay S. and Maulik U. An evolutionary technique based on k-means algorithm for optimal clustering in rn. *Information Sciences*, 146:221–237, 2002.

[111] Bandyopadhyay S. and Maulik U. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35:1197–1208, 2002.

[112] Das S., Abraham A., and Konar A. Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on Systems Man and Cybernetics - Part A*, 38(1):218–237, 2008.

[113] Das S., Abraham A., and Konar A. *Metaheuristic Clustering*. Springer, 2009.

[114] Guha S., Rastogi R., and Shim K. Cure: an efficient data clustering method for very large databases. In *ACM-SIGMOD Proceedings of the International Conference Management of Data*, pages 73–84, 1998.

[115] Kirkpatrick S., Gelatt C.D., and Vecchi M.P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[116] Naka S., Genji T., and Fukuyama Y. Practical distribution state estimation using hybrid particle swarm optimization. *IEEE Power Engineering Society Winter Meeting*, 2:815–820, January 2001.

[117] Paterlini S. and Minerva T. Evolutionary approaches for bluster analysis. In Bonari A., Masulli F., and Pasi G., editors, *Soft Computing Applications*, pages 167–178. Springer, Berlin, 2003.

[118] Prestwich S. Combining the scalability of local search with the pruning techniques of systematic search. *Annals of Operations Research*, 115:51–72, 2002.

[119] Theodoridis S. and Koutroumbas K. *Pattern recognition*. Elsevier Academic Press, 2003.

[120] Wu S. and Yan H. Microarray image processing based on clustering and morphological analysis. In *APBC '03: Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003*, pages 111–118, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[121] Alexios Savvides, Vasilis J. Promponas, and Konstantinos Fokianos. Clustering of biological time series by cepstral coefficients based distances. *Pattern Recogn.*, 41(7):2398–2412, 2008.

[122] Bäck T., Fogel D., and Michalewicz Z. *HandBook of Evolutionary Computation*. IOP Publishing and Oxford University Press, New York and Bristol (UK), 1997.

[123] Crainic T. and Toulouse M. *Hanbook of Metaheuristics*, chapter Parallel Strategies, pages 475–513. Kluwer Academic Publishers, 2003.

[124] Hogg T. and Williams P. Solving the really hard problems with coopative search. In *AAAI93*, pages 213–235, 1993.

[125] Kanungo T., Nathan S., and Wu A.Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, July 2002.

[126] Kohonen T. Self-organizing maps. *Springer Series in Information Sciences*, 30, 1995.

[127] Stützle T. Local search algorithms for combinatorial problemas analysis, algorithms and new applications. Technical report, DISKI Dissertation zur Künatliken Intelligenz, Sankt Augustin Germany, 1999.

[128] Feo T.A. and Resende M.G.C. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[129] Maulik U. and Badyopadhyay S. Fuzzy patitioning using real coded variable length genetic algorithm for pixel clasification. *IEEE Transactions on Geosciences and Remote Sensing*, 41(5):1075–1081, 2003.

[130] Maulik U. and Bandyopadhyay S. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33:1455–1465, 2000.

[131] Makarenkov V. and Legendre P. Optimal variable weighting for ultrametric and additive trees and k-means partitioning: Methods and software. *Journal of Classification*, (8):245–271, 2001.

[132] Ramos V., Muge F., and Pina P. Self-organized data and image retrieval as a consequence of inter-dynamic synergistic artificial ant colonies. In Ajith Abraham Javier Ruiz-del Solar and Mario Köppen, editors, *Soft Computing Systems - Design, Management and Applications*, pages 500–509, 2002.

[133] Ramos V. and Merelo J.J. Self-organizad stigmergic document mpas: Environments as mechanism for context learning. In *First Sapnish Conference on Evolutionary and Bio-Inspired Algorithms*, pages 284–293. Centro Univ Mérida, Mérida, España, 2002.

[134] van den Bergh F. *An Analysis of Paricle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.

[135] van der Merwe D.W. and Engelbrecht A.P. Data clustering using particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 215–220, 2003.

[136] Raghavan V.V. and Birchand K. A clustering strategy based on a formalism of the reproductive process in natural systems. In *Conference on Research and Development in Information Retrieval*, pages 10–22, 1979.

[137] Tsang W. and Kwong S. Ant colony clustering and feature extraction for anomaly intrusion detection. In Abraham A., Grosan C., and Ramos V., editors, *Swarm Intelligence in Data Mining*, pages 101–121, 2006.

[138] Chang W.Y. and Yang H.T. Partial discharge pattern recognition of cast-resin current transformers using fuzzy c-means clustering approach. *WSEAS Trans. Comp. Res.*, 3(3):172–181, 2008.

[139] Cui X and Potok T.E. Document clustering analysis based on hybrid pso+ k-means algorithm. *Journal of Computer Sciences - Special Issue*, pages 27–33, 2005.

[140] Cui X., Potok T.E., and Palathingal P. Document clustering using particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, 2005.

[141] Xiao X., Dow E.R., Eberhart R.C., Miled Z.B., and Oppelt R.J. Gene clustering using self-organizing maps and particle swarm optimization. In *Parallel and Distributed Processing Symposium*, 2003.

[142] Xie X.F., Zhang W.J., and Yang Z.L. Solving numerical optimization problems by simulating particulates in potential field with cooperative agents. In *International Conference on Artificial Intelligence*, 2002.

[143] Shi Y. and Eberhart R.C. Parameter selection in particle swarm optimization. In *Annual Conference on Evolutionary Programming*, pages 591–600, 1998.

[144] Shi Y. and Eberhart R.C. Fuzzy adaptive particle swarm optimization. *IEEE Congress of Evolutionary Computation*, 2:1682–1687, May 2001.

[145] Chiou Y.C. and Lan L.W. Theory and methodology. genetic clustering algorithms. *European Journal of Operational Research*, 135:413–427, 2001.