UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 2

MÓDULO DE MINERÍA DE DATOS PARA EL GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Javier Fernández Machín

Tutores: MsC. Darián Horacio Grass Boada
Ing. Husseyn Despaigne Reyes

La Habana, 2015

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Telemática (TLM) de la Facultad 2 y a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estime pertinente con el mismo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

Autor:

_____

Javier Fernández Machín

Tutor:

_____

MsC. Darián Horacio Grass Boada

Tutor:

_____

Ing. Husseyn Despaigne Reyes

DEDICATORIA

A mi papá que aunque no se encuentre físicamente a mi lado, sé que nos pareceríamos mucho y que hubiese estado orgulloso de mí y de lo que soy ahora, te quiero.

A mi mamá que ha sabido ser madre y padre para mí, esta tesis es de los dos, te amo mucho.

A Luis Miguel mi socio del técnico medio, estés donde estés todos te extrañamos.

AGRADECIMIENTOS

Agradecer en primer lugar a mi mamá por el apoyo, cariño y formación durante toda mi vida, si no fuera por ti no estaría hoy aquí. A Aleida, mi segunda mamá, por siempre confiar en mí desde niño y quererme mucho. A mi tía Ibis por su confianza y comprensión. A mi padrastro Mario por lo bueno que ha sido conmigo y con mi mamá. A Omar por ayudarme a dar mis primeros pasos en la Informática y por enseñarme a armar el cubo de Rubik, te quiero mi hermanito.

A mi tutor MsC. Darián Horacio Grass Boada por la enorme labor pedagógica que ejerció sobre mí, por los debates, por las ideas, por el rigor científico y por la formación ingenieril que me ha dado, es sin dudas el mejor tutor del mundo, gracias.

De la UCI a mis amigos del aula, donde todos hemos sabido vencer esta carrera de 5 años satisfactoriamente. A Eiquel, Randy, Jesus Yasser, Lidia, Dariel, Yaikel, son lo máximo en amigos y aunque nuestras vidas se distancien a partir de ahora no vamos a perder el contacto. A mi hermano Lian en estos 5 años de continuo batallar, por ayudarnos mutuamente en los estudios. A Michy, que por siempre será la mejor amiga que tengo, gracias por criticarme y por hacerme mejor persona cada día. A Legna mi amiga, por todos los momentos lindo que viví a su lado, nunca te olvidaré. A la gente del chat, en especial a Mayli la profesora de la CUJAE, quien me ha ayudado con sus opiniones y experiencias para el documento.

Un agradecimiento muy especial para la gente del Instituto Politécnico de Informática "Raúl Cepero Bonilla", donde pasé una de las mejores etapas de mi vida. A mis compañeros del aula: Angel, El Ruso, Frank, Yadira, Glendys, Claudia, Alejandro, El Bob, Herlin, todos los quiero mucho, espero que nos sigamos viendo aunque sea eventualmente. A Mercy, mi vecina, mi amiga, mi profesora una vez, mi hermana, gracias por estar ahí en todo momento. Al profesor Luis Fajardo por incentivar en mí el gusto por las matemáticas. A la familia de Luis Miguel, sus padres: Oscarito y Mey, gracias por la preocupación y por estar siempre pendientes de mí.

A mi abuelo Pedro, mi primo Dariel, mis tíos María Elena y Frank, los quiero mucho, gracias por su apoyo en todo momento. En fin, a todos, MUCHAS GRACIAS.

Javier Fernández Machín

RESUMEN

El sistema Gestor de Recursos de Hardware y Software (GRHS) es una aplicación que recopila información de hardware y software perteneciente a un grupo de estaciones de trabajo. También almacena información sobre incidencias desatadas por parte de los usuarios que van en contra de la política informática de una institución. Actualmente, aunque el registro de incidencias presenta métodos básicos de representar la información almacenada, adolece de funcionalidades que expliquen comportamientos internos en los que puede residir valiosa información de utilidad para directivos y la toma de decisiones. La Minería de Datos es una de las fases del proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD), la cual incluye un compendio de técnicas de Inteligencia Artificial que permiten encontrar información no trivial residente en datos almacenados a través de diferentes tareas, una de ellas es el agrupamiento (*clustering*). En la presente investigación se desarrolla un módulo de extracción de patrones en términos de grupos para el sistema GRHS que implementa un algoritmo de agrupamiento que combina técnicas de metaheurísticas evolutivas con métodos de agrupamiento particional tradicionales. Dicho módulo sirve de sustento a especialistas y directivos para la toma de decisiones sobre los recursos de hardware y software de una organización.

Palabras clave: agrupamiento, metaheurísticas evolutivas, minería de datos, patrones, toma de decisiones.

INTRODUCCIÓN

El control de los recursos es un proceso clave para el éxito de toda organización. Para ello, es necesario la administración de inventarios, cuyo propósito esencial es aumentar la rentabilidad de la organización por medio de una correcta utilización del inventario (1), así como el registro total y documentado de los recursos tangibles e intangibles con que cuenta una organización en un momento determinado. Inicialmente, los inventarios se confeccionaban de manera manual, proceso complejo en el que se cometían irregularidades e imprecisiones. A fin de eliminar tales inconvenientes y con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), se han creado aplicaciones informáticas que gestionan los activos tangibles e intangibles de las organizaciones.

La gestión de inventarios, particularmente de recursos informáticos, asocia toda la información en términos de hardware y software perteneciente a una red de computadoras. Dicha información resulta útil a la hora de detallar características específicas y en sentido general ofrece un mapa de cómo están distribuidos los recursos en la organización.

El Centro de Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI), desarrolla en la actualidad un producto denominado Gestor de Recursos de Hardware y Software (GRHS), el cual permite realizar inventarios de hardware y software en una red de computadoras, así como registrar incidencias producidas en las estaciones de trabajo. Se denomina incidencia a cualquier alteración de software o hardware notificada por el sistema, la cual contiene información referente al propio suceso que le dió origen.

Dicha aplicación cuenta con funcionalidades que le ofrecen al usuario un conjunto de reportes sobre las incidencias registradas, como por ejemplo: total de incidencias, cantidad de incidencias de hardware, de software, entre otros. En dependencia del entorno en donde se despliegue la aplicación, la misma puede generar un gran volumen de información asociada a los ordenadores que monitorea, lo cual dificulta a los usuarios que interactúan con el sistema, el análisis de todos estos datos si no se cuenta con las herramientas necesarias para ello, tal es el caso de la UCI que cuenta con más de 5000 ordenadores en red.

El aumento continuo de la disponibilidad de datos arrojados por el sistema GRHS, particularmente del registro de incidencias, hace imprescindible el empleo de técnicas y herramientas que le den sentido y utilidad a dicha información, para que pueda ser asimilada como experiencia, conocimiento y sabiduría, que sirvan de apoyo a la toma de decisiones por parte de especialistas y directivos. La búsqueda de información relevante siempre es útil a la administración empresarial ya que puede ofrecer las respuestas más apropiadas a las necesidades de información (2). Hoy en día, el sistema

GRHS carece de funcionalidades que permitan descubrir estos tipos de comportamientos, propiedades, asociaciones, tendencias, etc., alojados en los datos, que posibiliten a usuarios capacitados analizar y describir características subyacentes dentro del registro de incidencias con vistas a la toma de decisiones.

Con el desarrollo de la Inteligencia Artificial (IA) aparejado a las técnicas de Minería de Datos (MD), es posible procesar grandes volúmenes de datos con el propósito de descubrir patrones valiosos intrínsecos en ellos. Según (3), la MD se refiere a la aplicación de los métodos de aprendizaje y estadísticos para la obtención de patrones y modelos. Para llevar a cabo las tareas de MD existen dos grandes categorías: predictiva y descriptiva, las cuales pretenden estimar valores futuros o desconocidos e identificar patrones que expliquen comportamientos en los datos respectivamente. Entre las tareas descriptivas de MD se encuentra el agrupamiento *(clustering)*, cuyos algoritmos se encargan de encontrar grupos dentro de un conjunto de objetos (datos), de tal forma que los objetos que pertenezcan al mismo grupo sean "similares" y los que pertenezcan a diferentes grupos sean lo más disímiles posible (4).

Mediante la aplicación de esta técnica a la información almacenada por GRHS, se pueden obtener patrones o sucesos correlacionados que agrupen incidencias que comparten características comunes, lo cual permite focalizar la atención sobre ellas y posteriormente determinar las causas que las provocaron.

Para llevar a cabo la extracción de conocimientos a partir de una fuente de datos, en 1996 fue propuesto el proceso de Descubrimiento de Conocimiento en Bases de Datos (KDD[1]) (5), dentro del cual se inserta la MD como una de sus fases, además de la preparación de los datos y evaluación e interpretación de los resultados.

Debido a lo antes descrito se plantea como **problema a resolver**: Insuficiencias en el análisis de la información recopilada por el sistema GRHS para el apoyo a la toma de decisiones. Como **objeto de estudio** se define la fase de Minería de Datos. El **campo de acción** se enmarca en las técnicas descriptivas de Minería de Datos basadas en agrupamiento orientadas al sistema GRHS. El **objetivo general** es desarrollar un módulo de extracción de patrones en términos de grupos para el sistema GRHS que permita apoyar la toma de decisiones.

Tareas de la investigación:

1) Análisis de las características de la información almacenada por el sistema GRHS.

[1] Knowledge Discovered in Database, KDD por sus siglas en inglés

2) Análisis de los principales algoritmos de agrupamiento.
3) Análisis de las principales bibliotecas de algoritmos para realizar tareas de MD.
4) Análisis de las principales técnicas de representación para MD para una correcta visualización de los resultados.
5) Diseño del módulo de MD.
6) Implementación del módulo de MD.
7) Realización de pruebas al módulo de MD para el sistema GRHS.

Métodos Teóricos:

Histórico-Lógico: permite analizar el desarrollo de los algoritmos existentes para la solución del problema del agrupamiento multiobjetivo en general, así como las bases histórico-conceptuales de los mismos, con el objetivo de detectar deficiencias y proponer soluciones acordes al campo de acción definido.

Analítico-Sintético: utilizado para analizar toda la información recopilada a través de los diferentes medios bibliográficos que pueda servir como base teórica al diseño del sistema así como para sintetizar los resultados en la aplicación.

Métodos Empíricos:

Observación científica: permite analizar el comportamiento de cada clase del marco de trabajo de algoritmos seleccionado, así como observar y caracterizar el proceso de comunicación entre los diferentes paquetes.

La tesis presenta la siguiente estructura capitular:

Capítulo I: Contiene el marco teórico de la investigación, en él se incluyen el estado del arte así como la descripción de las herramientas y metodologías a utilizar para la realización del presente trabajo.

Capítulo II: En este capítulo se especifica y describe la propuesta de solución a las deficiencias en el análisis de la información recopilada por el sistema GRHS. Incluye los artefactos propuestos por la metodología de desarrollo seleccionada.

Capítulo III: Se especifican las pruebas y experimentaciones realizadas al módulo implementado y al algoritmo desarrollado respectivamente para validar los mismos, además se muestran los resultados obtenidos.

CAPÍTULO I
FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se abordan varios aspectos teóricos, -conceptos y definiciones-, relacionados con el contexto sobre el cual se desarrolla este trabajo, con el propósito de obtener los conocimientos necesarios para el desarrollo del módulo. Se explican conceptos fundamentales tanto para el entendimiento del negocio, como para los relacionados con el proceso de Descubrimiento de Conocimientos en Bases de Datos. Se argumentan principios fundamentales del problema del agrupamiento, así como de las diversas maneras de resolverlo. Incluye una descripción de las tecnologías y metodologías utilizadas en la implementación de la solución.

1.2 Sistema Gestor de Recursos de Hardware y Software (GRHS)

El sistema Gestor de Recursos de Hardware y Software es una herramienta informática que facilita el inventario del hardware y el software instalado en una red de computadoras. El mismo surge como un proyecto del Centro de Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI), a raíz de la necesidad de inventariar los recursos informáticos presentes en la universidad. Debido a su arquitectura, es posible la instalación de este sistema en cualquier organización para el control de los recursos de hardware y software, para ello, cuenta con funcionalidades que le permiten recopilar toda esta información procedente de las estaciones de trabajo y en dependencia del escenario donde se despliegue puede llegar a constituir un cúmulo sustancial de datos. A continuación, se explican los dos tipos de datos que el sistema de GRHS es capaz de almacenar: inventarios e incidencias.

1.2.1 Inventario

El inventario representa una relación ordenada de bienes y existencias de una entidad o empresa en un momento determinado (6). Las empresas realizan inventarios para llevar un registro histórico de los activos y conocer el estado y las características de las mercancías, dígase fecha de entrada, número de lote, fecha de caducidad, entre otras. Particularmente en los inventarios de activos informáticos las aplicaciones encargadas de realizar este proceso llevan a cabo un análisis exhaustivo en términos de hardware y software de las estaciones de trabajo de la organización. En el ámbito del sistema GRHS el inventario se realiza mediante una aplicación instalada en cada una de las estaciones de trabajo de la organización, la cual recopila un conjunto de información o propiedades de piezas y/o programas instalados en una computadora.

1.2.2 Incidencias

Una incidencia es un acontecimiento que sobreviene en el curso de un asunto o negocio y tiene con él alguna conexión (7). Para el sistema GRHS una incidencia es un evento que ocurre cuando se detecta algún cambio en el inventario no autorizado por la entidad. Las incidencias pueden ser causadas por cambios de software o hardware, por ejemplo: la instalación/desinstalación de un programa, o la extracción de algún periférico de entrada/salida, respectivamente. No todas las incidencias tienen el mismo nivel de importancia, para ello se define una clasificación totalmente configurable por parte de los directivos del sistema, que puede ser alta, media o baja de acuerdo a la política informática de la empresa.

1.2.3 Dominio de los datos

Los inventarios generados por el sistema relatan minuciosamente características de hardware y de software. Referente a este último se captura: plataforma, nombre de usuario, relación de programas instalados y de cada uno de ellos se obtiene el nombre, la fecha de instalación, la versión, el tamaño, etc. En este aspecto se logra tener gran cantidad de datos en su mayoría de procedencia categórica pues describen rasgos nominales.

En términos de hardware se cuenta con características de: BIOS, CD-ROM, dispositivos de almacenamiento, teclado, ratón, monitor, microprocesador, tarjeta madre, interfaz de red, memoria RAM, etc. Es evidente la aparición de algunas descripciones de naturaleza numérica como por ejemplo la cantidad de memoria RAM, pero igualmente se destacan datos categóricos en su mayoría.

De acuerdo a las incidencias se registra: tipo de incidencia (hardware o software), componente que la generó (teclado, programas instalados, memoria RAM, etc.), estado que puede ser de adición o sustracción, nivel de importancia (alto, medio o bajo), fecha y localización. Nótese que es posible obviar el tipo de incidencia pues el componente infiere si es de hardware o de software.

1.2.4 Análisis de los datos

Como se subrayó anteriormente, el sistema tiene la capacidad de arrojar grandes volúmenes de información de acuerdo al entorno donde se aplique. Actualmente el sistema cuenta con funcionalidades que permiten, a través de reportes, brindar información (en su mayoría cuantitativa) acerca de las incidencias registradas, pero carece de métodos que describan asociaciones o relaciones subyacentes que aporten conocimiento útil para la toma de decisiones, lo cual puede ser corroborado por expertos en el dominio, en caso de que se intuya empíricamente. Debido a que las incidencias son causadas por acción directa de los usuarios, resulta interesante la búsqueda de tendencias que

expliquen el comportamiento de los mismos, con vistas a la toma de decisiones sobre los recursos informáticos. Un ejemplo de patrón inferido pudiera ser: la extracción de dispositivos mouse se encuentra dentro del grupo de incidencias que ocurren en horario nocturno, lo cual posibilitaría la toma de medidas preventivas y posteriormente determinar las causas por las cuales se manifiesta tal comportamiento.

Con respecto al dominio de aplicación UCI, especialistas y directivos con experiencia no poseen ningún conocimiento previo que sirva como punto de partida para la obtención de patrones. La información, y particularmente, el registro de incidencias, tienen una naturaleza completamente no supervisada y aleatoria, debido a que están sujetos directamente a intervenciones por parte de los usuarios. En consecuencia, los tipos de patrones subyacentes en términos de grupos, pueden adquirir diversas estructuras, las cuales se podrían comprender de mejor forma si se enfocan desde distintos puntos de vista como: heterogeneidad entre grupos, homogeneidad dentro de los grupos, etc.

Para proceder a la extracción de conocimiento a partir del análisis de grandes volúmenes de información como son las bases de datos del sistema GRHS, existe el proceso denominado Descubrimiento de Conocimiento en Bases de Datos (8).

1.3 Descubrimiento de conocimiento en bases de datos (KDD)

En un nivel abstracto, KDD es la rama del conocimiento que se preocupa por el desarrollo de métodos y técnicas para dar sentido a los datos (8). Es un proceso complejo el cual tiene como propósito la obtención de regularidades, propiedades, síntomas, patrones, etc. implícitos en grandes volúmenes de datos a partir de construcción de modelos de abstracción. El problema de KDD es partir de un conjunto de datos voluminoso y casi ininteligible y convertirlo en otro más compacto, más abstracto y más útil (9). Incluye no solo la obtención de patrones, sino también la evaluación y posible interpretación de los mismos, para ello consta de una secuencia iterativa e interactiva de pasos (3).

1) *Integración y recopilación:* fase en la que se unifican, preferiblemente en un almacén de datos, todas las fuentes de información sobre la cual se desea aplicar un proceso KDD.
2) *Selección, limpieza y transformación:* fase encargada de discernir entre cuáles datos son útiles y cuáles no, con el objetivo de ser incluidos en la vista minable[2] (10). También se mejora la calidad de los datos, se trabajan los datos con ruido (outliers[3]), se utilizan técnicas de discretización y/o numeración, etc. así como de completamiento de datos faltantes o perdidos.

[2] Estructura final de datos en la que se consolida en una única tabla todas las observaciones y atributos sobre los cuales se aplicarán algoritmos de minería. Las filas conforman los datos a evaluar y las columnas los rasgos que los describen.
[3] Datos anómalos o atípicos, su existencia puede deberse a errores en el proceso de recopilación o valores reales que son diferentes por completo al resto.

3) *Minería de datos:* es la fase más importante del KDD, por tal motivo suele usarse para nombrar todo el proceso, es en donde se aplican algoritmos y técnicas computacionales para la extracción de rasgos representativos de los datos.
4) *Evaluación e interpretación:* a través de esta fase podemos conocer la "calidad" del conocimiento obtenido en términos de precisión, comprensión, relevancia, etc.
5) *Difusión, uso y monitorización:* última fase en la que el conocimiento adquirido se utiliza en el apoyo a la toma de decisiones de la entidad, así como también se mejora en el sistema según los resultados obtenidos.

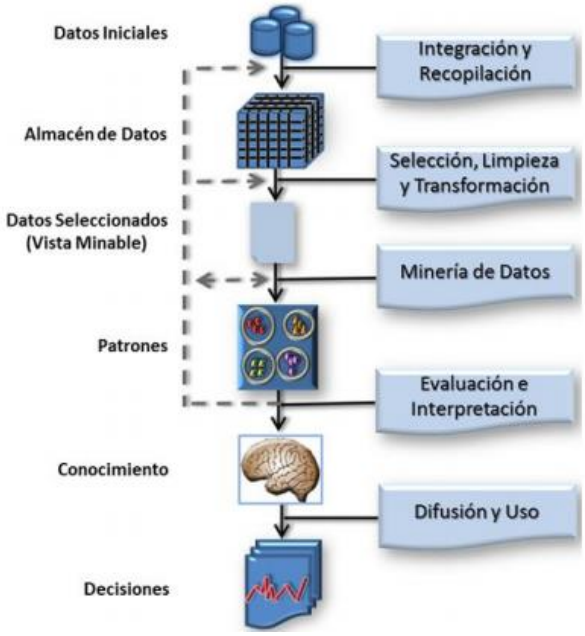


Figura 1 *Fases del KDD (11)*

1.3.1 Minería de Datos

Una simple definición es la siguiente: *"el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos",* (3). La minería de datos conforma una tecnología compleja que tiene su basamento en diversas disciplinas, tales como las Matemáticas, Estadística, Ciencias de la Computación, Física, Biología, etc., y tiene una inmensa utilidad en dominios de aplicación donde surge la necesidad de extraer conocimiento subyacente en grandes volúmenes de datos (12). Debido a su importancia, algunos autores suelen utilizar este término indistintamente para referirse a todo el proceso KDD, aunque es solo una etapa del mismo.

Se define como la construcción de modelos abstractos, los cuales representan el conocimiento inferido, pueden utilizarse para predecir comportamientos futuros de los datos o para explicar algún tipo de propiedad que explique el comportamiento actual de los datos (3), para ello existe en la literatura una gran cantidad de algoritmos.

1.3.2 Modelos de Minería de Datos

Una de las interrogantes fundamentales a responder a la hora de enfrentar un problema de MD es el tipo de tarea que se desea llevar a cabo. En general existen dos grandes modelos en los que se encierran las tareas de minería, estos pueden ser descriptivos o predictivos. Los modelos predictivos (clasificación, regresión) realizan inferencia sobre los datos con el objetivo de realizar predicciones; en cambio, los modelos descriptivos (agrupamiento, reglas de asociación, análisis correlacional) sirven para identificar y explorar patrones subyacentes en los datos que pueden ser fácilmente comprendidos e interpretados por un usuario. En la presente investigación se aborda dentro del modelo descriptivo la tarea del agrupamiento, ya que constituye objetivo de la misma.

1.4 Agrupamiento

El agrupamiento es una de las tareas descriptivas de la MD y se considera uno de los problemas más difíciles en el aprendizaje automático, particularmente por su naturaleza no supervisada. El objetivo es encontrar un conjunto finito de grupos (clústeres) que describan un conjunto de datos de acuerdo a alguna medida subjetiva de (di)similaridad entre ellos. Según Rui Xu y Donald C. Wunsch II, los grupos son definidos en términos de homogeneidad interna y separación externa (13), por tanto los objetos pertenecientes al mismo grupo deben ser “parecidos” mientras que con respecto a los objetos de otros grupos deben ser “diferentes”. La semántica atribuida a qué tan parecidos/diferentes son dos objetos requiere conocimiento del dominio de aplicación e influye directamente en el resultado del agrupamiento. En algunas fuentes esta tarea también se denomina sumarización de los datos puesto que el resultado final puede interpretarse como un resumen de los datos originales (14).

Los métodos tradicionales de agrupamiento puede ser clasificados en tres grandes clases: jerárquico, basado en densidad, y particional (15).

1.4.1 Agrupamiento particional

Formalmente se define como: sea $X = \{\vec{x}_1,\ \vec{x}_2, \dots, \vec{x}_n\}$ un conjunto de n objetos cuyas etiquetas o nombres de las clases a las cuales pertenecen no se conocen, en un espacio d-dimensional en donde cada elemento $\vec{x}_{i,j}$ describe el rasgo j-ésimo $(j = 1, 2, \dots d)$ del i-ésimo objeto $(i = 1,2, \dots n)$ el agrupamiento particional intenta encontrar una configuración de K particiones $C = \{C_1, C_2, \dots C_K\}$ con $K \leq n$, de manera tal que (16):

$$(1)\ \ C_i \neq \emptyset\ \forall i\ \in \{1,2, \dots, K\}$$

$$(2)\ \ \cup_{i=1}^{K} C_i = X$$

De manera general existen dos enfoques para el agrupamiento particional: el duro *(hard)* y el difuso *(fuzzy)*. En el caso del agrupamiento particional duro cada objeto es exclusivamente asignado a un grupo, por tanto, además de las anteriores, debe cumplir con la siguiente condición: $C_i \cap C_j = \emptyset \ \forall i \neq j$ donde $i, j \ \in \{1,2,\dots,K\}$, la cual refleja la propiedad de mutua exclusión entre cualquier par de grupos. Para el caso del agrupamiento particional difuso, dicha condición no es tenida en cuenta ya que los objetos pertenecen a los K grupos con cierto grado de pertenencia. La pertenencia difusa del i-ésimo objeto al j-ésimo grupo puede representarse fácilmente mediante una matriz de coeficientes $U = [\mu_{ik}]_{N\times K}$ en la cual $i = (1,\dots,n)$, $k = (1,\dots,K)$, $\mu_{ik} \in [0,1]$ y $\sum_{k=1}^{K} \mu_{ik} = 1\,, \forall i = 1,\dots,n$. Este tipo de enfoque difuso es conveniente cuando las fronteras entre grupos no están bien definidas y donde los objetos pueden pertenecer a más de un grupo.

El proceso de creación de particiones trae aparejado consigo la optimización (maximización o minimización) de una o más funciones objetivo, las cuales pueden medir homogeneidad, compactación, conectividad, separación, desviación, etc. Bajo un enfoque enumerativo es posible encontrar la partición o el cubrimiento ideal hallando todos los posibles agrupamientos, lo cual supondría un problema NP-completo [4] (17) puesto que el número de soluciones crece exponencialmente a medida que aumenta el número de objetos y la cantidad de grupos (18). La complejidad espacial está representada por la siguiente fórmula (13):

$$P(N,K) = \frac{1}{K!}\sum_{m=1}^{k}(-1)^{K-m} C \binom{m}{K} m^N$$

$$N \ = Cantidad\ de\ Objetos$$
$$K = Cantidad\ de\ Grupos$$

Ecuación 1 *Complejidad espacial*

En términos de recursos computacionales resulta incosteable atacar este problema siguiendo un enfoque enumerativo, por tanto se considera un problema de optimización complejo, lo cual da lugar a la idea de resolver esta problemática mediante métodos o procedimientos de propósito general. Estos métodos conocidos como metaheurísticas se independizan de las características del problema y definen una estrategia o heurística a seguir para explorar el espacio de búsqueda, los cuales si bien no ofrecen seguridad de encontrar el óptimo global (configuración de particiones que mejor se ajuste a los criterios a optimizar), se aproximan a una solución buena en un tiempo prudente.

[4] Clase de problema perteneciente al conjunto de problemas NP y al conjunto de problemas NP-duros

1.4.2 Representación del agrupamiento particional

La representación de una configuración de N objetos en K particiones muestra cómo están distribuidos los objetos. Esta repartición no tiene por qué necesariamente ser mejor que otra, simplemente representa una solución más dentro del espacio de búsqueda del problema. Es importante destacar que el tipo de representación que se utilice debe estar acorde con las características que describen a cada uno de los objetos; en este sentido, existen tres grandes maneras de representación: basada en prototipo, en etiqueta y en grafo (19). Se ha demostrado que la forma en que se representan los agrupamientos influye de manera directa en el resultado final, de hecho, en el trabajo antes citado se realiza una comparación en términos de calidad, exploración del espacio de búsqueda y coste de tiempo al aplicar cada una de estas representaciones a problemas reales.

Una de las representaciones más utilizadas en la literatura es la de prototipo (20) (21), la misma define un vector de elementos representativos por cada grupo. Dicho vector es de tamaño $K \times d$ en donde K es la cantidad de grupos y d la dimensión (cantidad de rasgos) de cada objeto. Dos de las formas de obtener este prototipo es: por centroide en donde un objeto ficticio es construido mediante medidas de tendencia central (moda, media, etc.) o por medoide escogiendo un objeto real dentro del conjunto de objetos, en este caso el tamaño solo sería de K ya que solo es necesario el identificador de los objetos que encabezan cada grupo.

Por otra parte, la representación basada en etiqueta consiste en un vector de tamaño n, donde n es la cantidad de objetos. Este vector está conformado por identificadores de grupos en el que cada posición contiene un valor j donde $1 \leq j \leq k$, de esta manera se representa la pertenencia del objeto i al identificador de grupo que aparece en esta posición.

Por último, en (22) se hace uso de la representación basada en grafos, la misma está sustentada sobre los conceptos y propiedades de la teoría de grafos. La utilización de esta teoría puede ser aplicada al contexto del agrupamiento, tratando de buscar todas las componentes máximas conexas de un subgrafo, de manera tal que cada componente representa un grupo (23). Para representar este grafo se utiliza un vector de tamaño n en el que la posición i representa al objeto i-ésimo y contiene el identificador de objeto al cual él está asociado. Para la extracción de los grupos es necesario recorrer el grafo enlazando los nodos, en trabajos recientes se ha demostrado que este proceso puede hacerse en tiempo lineal $O(n)$.

1.4.3 Algoritmos de agrupamiento particional

K-Means

El surgimiento de este algoritmo se remonta a la década de los sesenta del siglo pasado. La entrada consiste en un único parámetro: k, el cual impone el número de grupos que van a encontrarse. Inicialmente se seleccionan k objetos[5] de manera aleatoria o mediante un conocimiento previo que se tenga sobre el conjunto de datos, luego se asignan los objetos restantes al centroide más cercano de acuerdo a una medida de proximidad o de distancia $d(i,j)$ conformando así los grupos. Posteriormente comienza el proceso iterativo de reajustes de centroides como el promedio de los objetos que pertenecen a cada grupo y de reasignación de objetos, donde se minimiza la siguiente función:

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - m_j|^2$$

Ecuación 2 *Suma del error cuadrático en K-Means*

El algoritmo trata de minimizar E que es la suma del error cuadrático, p y m_j son los objetos y el centroide del j-ésimo grupo respectivamente. Este proceso se repite hasta que los centroides no varíen con respecto a la iteración anterior o se alcance un número prefijado de iteraciones. Es un algoritmo fácilmente de implementar y es la solución a muchos problemas prácticos en donde los grupos son bien compactos e hiperesféricos (23). Aun así, posee las siguientes limitantes:

1) Es necesario conocer el número de grupos (k).
2) Tiende a converger en óptimos locales.
3) Es vulnerable a datos anómalos o atípicos.
4) Depende de la selección de los centroides.
5) Al estar basado en el concepto de distancia solo es aplicable a dominios de aplicación en el que los datos están descritos por rasgos numéricos.

[5] Al inicio del algoritmo se les denomina semillas o *seeds*, por su traducción al inglés

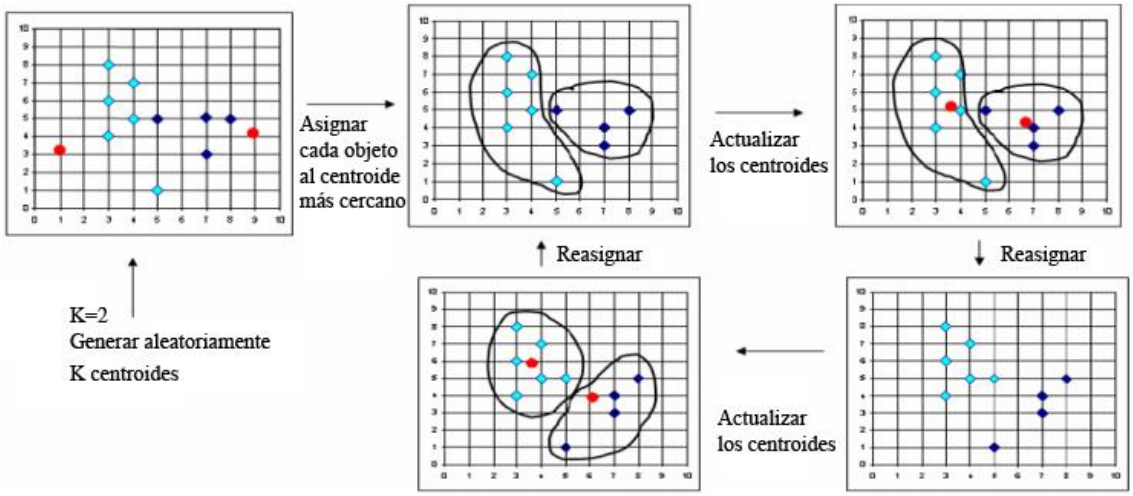


Figura 2 *Funcionamiento del algoritmo K-Means* (24)

K-Medoids

Clásico algoritmo de agrupamiento particional el cual es una variante del algoritmo *K-Means*, con la diferencia de que los objetos que representan a los grupos son medoides, en lugar de centroides. Debido a que utiliza objetos representativos reales dentro de los grupos, es menos sensible que el *K-Means* en cuanto a valores atípicos (25) y eficiente ante dominios de aplicación en el que los datos están descritos por rasgos categóricos. Al igual que *K-Means* el método consiste en reducir al mínimo la suma de la diferencias entre cada objeto y su correspondiente medoide (**Ecuación 2**). Al inicio se seleccionan aleatoriamente k objetos (medoides), luego ocurre el proceso iterativo de asociación de los objetos al medoide más cercano y recolocación de los medoides, esto último puede interpretarse como la búsqueda del objeto que mejor represente al grupo. El procedimiento continúa hasta que no haya cambios en la configuración de los medoides. *K-Medoids* presenta los siguientes inconvenientes (19):

1) Es necesario conocer el número de grupos (k).
2) Tiende a converger en óptimos locales.
3) El resultado y el tiempo total de funcionamiento dependen de la selección de los medoides iniciales.

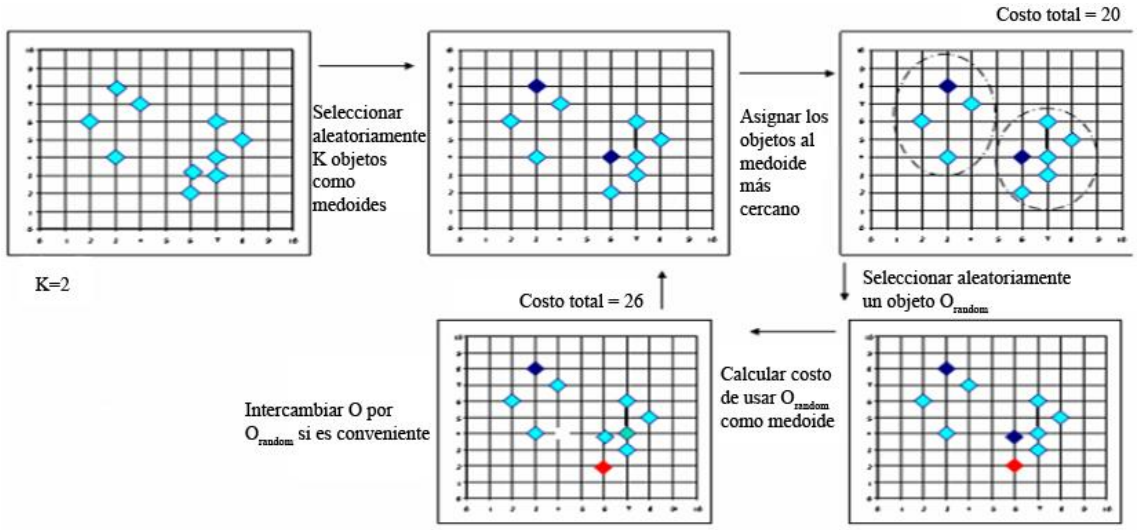


Figura 3 *Funcionamiento del algoritmo K-Medoids* (24)

Fuzzy C-Medoids (FCMdd)

Es la versión difusa del algoritmo *K-Medoids* y una extensión del algoritmo *Fuzzy C-Means* (26). Debido a su naturaleza difusa, contiene una matriz $U = [\mu_{ik}]_{N\times K}$ que denota el grado de membresía de los objetos a cada uno de los grupos. Similar al *K-Medoids*, comienza con la selección aleatoria de k medoides, luego se inicia el proceso iterativo de actualización de matriz de pertenencia y recolocación de medoides. Durante este proceso se va minimizando la siguiente función de compactación (27):

$$J_m(U, Z: X) = \sum_{k=1}^{n}\sum_{i=1}^{K} \mu_{ik}^m D(z_i, x_k)$$

Ecuación 3 *Función que optimiza Fuzzy C-Medoids*

Donde $X = \{x_1, x_2, \dots, x_n\}$ es el conjunto de objetos a agrupar, $Z = \{z_1, z_2, \dots, z_K\}$ el conjunto de medoides, K la cantidad de grupos, m el exponente difuso y $D(z_i, x_k)$ alguna medida de distancia entre z_i y x_k. Para el cálculo de los valores de la matriz de pertenencia se utiliza la siguiente fórmula (28):

$$\mu_{ik} = \frac{1}{\sum_{j=1}^{K}(\frac{D(z_i, x_k)}{D(z_j, x_k)})^{\frac{1}{m-1}}}$$

Ecuación 4 *Cálculo de valores de pertenencia de Fuzzy C-Medoids*

Correspondiente a la actualización de los medoides, un medoide z_i es reemplazado por un objeto x_p donde:

$$p = \underset{1 \le j \le n}{\operatorname{argmin}} \sum_{k=1}^{n} u_{ik}^{m} D(x_j, x_k)$$

Ecuación 5 *Actualización de medoides en Fuzzy C-Medoids*

Al igual que su homólogo en entornos duros, este método tiene las siguientes desventajas:

1) Es necesario conocer el número de grupos (k).
2) Tiende a converger en óptimos locales.

1.4.4 Agrupamiento particional mono objetivo

Como se destacó anteriormente, el proceso de generación de grupos está guiado por una o más funciones objetivos a optimizar. Cuando el número de funciones es 1 se trata de un problema de optimización mono objetivo. Los métodos tradicionales anteriormente explicados (*K-Means*, *K-Medoids* y *Fuzzy C-Medoids*) abordan el problema del agrupamiento siguiendo una estrategia avara en la que interviene un solo criterio. De manera general, estos criterios pueden agruparse en tres grandes clases: compactación, conectividad y separabilidad (15).

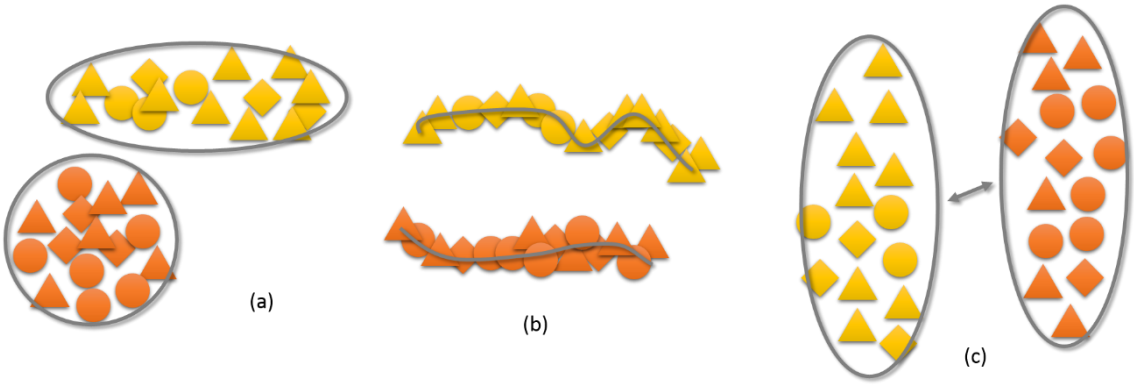


Figura 4 *Clases de criterios de agrupamientos: (a) compactación, (b) conectividad y (c) separabilidad*

Como se puede observar en la **Figura 4**, la clase compactación (a) intenta encontrar grupos que conformen una figura regularmente esférica respecto a un centroide, esto se debe a que la variación dentro de los grupos es mínima, por tanto objetos que pertenezcan al mismo grupo deben tener un comportamiento similar. En tanto, la conectividad (b) plantea la conformación de grupos mediante la conexión de elementos próximos (vecinos). Esta clase tiene la peculiaridad de no determinar un tipo de representación gráfica como la anterior (esferas), sino que es útil a la hora de encontrar grupos no convexos con disímiles formas, sin embargo cuando los objetos están suficientemente separados como para no tener relación entre ellos su utilización no resulta ser práctica. La separabilidad (c) construye los grupos de manera tal que las fronteras entre ellos estén bien delimitadas. Los criterios de compactación y separabilidad suelen ir acompañados de acuerdo al principio mismo del agrupamiento

en el que los objetos dentro de un grupo deben ser homogéneos mientras que a nivel de grupos estos deben ser lo más heterogéneos posible.

Existen muchos criterios de agrupamientos, los cuales de manera general están enmarcados dentro de estas tres clases, además pueden estar orientados a entornos difusos o duros. Algunos de ellos son (16) (14):

$$Dev(C) = \sum_{C_k \in C} \sum_{i \in C_k} \delta(i, z_k)$$

Ecuación 6 *Función de Compactación*

$$Conn(C) = \sum_{i=1}^{N} \sum_{j=1}^{L} x_{i,nn_{ij}} \qquad donde \qquad x_{r,s} = \begin{cases} \frac{1}{j} & si \quad \nexists C_k : r \in C_k \wedge s \in C_k \\ 0 & en\ otro\ caso \end{cases}$$

Ecuación 7 *Función de conectividad*

$$Sep = \sum_{i=1}^{K} \sum_{j=1, j \neq i}^{K} \delta(z_i, z_j)$$

Ecuación 8 *Función de separabilidad*

Con el objetivo de expresar compactación en los datos, la **Ecuación 6** se define como la suma de todas las distancias de los objetos al centroide del grupo al cual pertenece. Donde C es el conjunto de grupos, z_k representa el k-ésimo centroide y $\delta(.,.)$ expresa una medida de proximidad que suele ser la euclideana. La **Ecuación 7**, evalúa el nivel de vecindad de los objetos que se encuentran dentro de cada grupo, donde nn_{ij} es el j-ésimo objeto más cercano al objeto i, N es la cantidad de objetos y L es un parámetro definido por el usuario que especifica la cantidad de vecinos que intervienen en la evaluación de la conectividad. En ambas funciones el objetivo es minimizarlas y están enfocadas a entornos duros. La **Ecuación 8** expresa la separabilidad que existe entre los objetos representativos (medoides o centroides) cuyo objetivo es maximizar esta distancia.

$$J_m = \sum_{i=1}^{K} \sum_{j=1}^{n} \mu_{ij}^m \left\| x_i - z_j \right\|^2$$

Ecuación 9 *Función de compactación* J_m

$$XB = \frac{\sum_{i=1}^{K} \sum_{j=1}^{n} \mu_{ij}^2 \left\| x_i - z_j \right\|^2}{n \times min_{i \neq j} \left\| x_i - x_j \right\|^2}$$

Ecuación 10 *Función de separabilidad Xie-Beni*

En la **Ecuación 9** y **Ecuación 10**, K es el número de grupos, n la cantidad de objetos, x_i son los objetos representativos por grupos (centroides o medoides), z_j el j-ésimo objeto, en tanto μ_{ij} denota el grado de pertenencia del objeto z_j al grupo C_i, m $(m \geq 1)$ es un parámetro a definir por parte del usuario. Como funciones objetivos son una variante de compactación y separabilidad respectivamente, orientadas a entornos difusos, que deben ser minimizadas para obtener buenos resultados.

La optimización de un único criterio obliga a que los patrones encontrados se correspondan con el tipo de propiedades que la función a optimizar define. Por tanto, las estructuras obtenidas responderán a este criterio, el cual en un entorno no supervisado se desconoce, al no contar con información previa acerca de la distribución de los objetos en grupos. Por tanto, un solo criterio de optimización no logra captar diferentes aspectos en el agrupamiento: tamaño, formas convexas o no, uniformidad, compactación inter-grupos, etc. Debido a esto surge el agrupamiento con enfoque multiobjetivo, el cual optimiza más de una función simultáneamente con el propósito de reflejar de mejor forma características subyacentes en los datos que de antemano son desconocidas. Recientemente es notable el desarrollo de métodos de agrupamiento de este tipo (15) (29) (27).

1.4.5 Agrupamiento particional multiobjetivo

El agrupamiento particional en el cual se optimiza más de una función constituye un problema de optimización multiobjetivo (MOP[6]). En general, un MOP radica en la optimización de $r \geq 2$ funciones objetivos, las cuales pueden ser conceptos contrapuestos en el que mejorar uno implicaría empeorar otro, por lo que la solución a estos problemas es en sí un conjunto de soluciones que no pueden ser consideradas diferentes entre los objetivos que optimizan (30) (31). Bajo este concepto, uno de los enfoques consiste en modelar el problema como una única ecuación donde intervienen los criterios a optimizar, asociados a un factor de peso, quedando el problema formulado de la siguiente forma (32):

$$Optimizar \sum_{i=1}^{r} w_i f_i$$

Ecuación 11 *Combinación lineal de pesos*

Donde w_i es el peso asociado a la función f_i. Otro enfoque es el de multiobjetivo puro, el cual consiste en la optimización simultánea de las funciones objetivos buscando un consenso entre ellas: de esta manera la noción de óptimo cambia. Este basamento está apoyado en el concepto de dominancia de Pareto definido en (33) como:

[6] Multiobjective Optimization Problem, por sus siglas en inglés

Una solución $s_1 \in X$ domina a otra solución $s_2 \in X$ si y solo si $\forall i \in \{1,2,\dots,r\}$, $f_i(s_1) \leq f_i(s_2)$ y $\exists j \in \{1,2,\dots,r\} : f_j(s_1) < f_j(s_2)$. En otras palabras: una solución domina a otra si y sólo si, es al menos tan buena como la otra en todos sus objetivos y es mejor en al menos uno de ellos.

Una solución s^* es no dominada si y sólo si no existe otra solución $s \in X$ tal que s domine a s^*. El grupo de soluciones no dominadas constituye el conjunto de Pareto y su proyección en las funciones a optimizar se le denomina frente de Pareto (**Figura 5**). Para el problema del agrupamiento multiobjetivo, y en general para cualquier problema de optimización multiobjetivo, es en el conjunto de Pareto donde yacen los agrupamientos resultantes que necesitan de un apoyo supervisado para seleccionar la respuesta más adecuada respecto al dominio de aplicación.

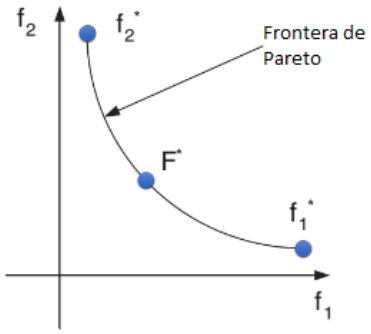


Figura 5 *Concepto de frontera de Pareto*

En relación a la propia naturaleza de los datos (**Sección 1.2.4**) y teniendo en cuenta las desventajas de optimizar un único criterio, en la presente investigación resulta conveniente utilizar un enfoque multiobjetivo ya que no se posee conocimiento a priori acerca de los patrones que pudieran encontrarse, por tanto la exploración del espacio de búsqueda a través de un grupo de funciones objetivos que se optimicen al unísono, en principio, puede describir de mejor forma las soluciones encontradas.

1.5 Metaheurísticas

Tal como se explicó anteriormente (**Sección 1.4.1**), se considera el agrupamiento un problema de optimización que puede ser tratable bajo algoritmos metaheurísticos. Los algoritmos de este tipo deben su nombre a Fred Glover quien los define en 1986 (34) como: *"una metaheurística es referida a la estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquellas que son normalmente generadas en la búsqueda de un óptimo local. Las heurísticas guiadas por una estrategia meta pueden ser un procedimiento de alto nivel o pueden contener una descripción de movimientos permitidos para trasformar una solución en otra, conjuntamente con una regla asociada de evaluación".* El prefijo "meta" indica que estos algoritmos solo definen la estrategia

general para guiar el proceso de búsqueda pero no especifican los detalles, por lo cual son flexibles a la hora de ser adaptados a una heurística local en una aplicación específica (35).

Las metaheurísticas son métodos generales que no están atados a un tipo de problema en específico. Tal como se plantea en (36), estos métodos son utilizados para encontrar soluciones buenas sin ninguna garantía de óptimo mediante una exploración inteligente del espacio de búsqueda. Existen varias taxonomías de las metaheurísticas, la más importante es la referente a la cantidad de soluciones que se exploran en cada iteración; éstas pueden basarse en: trayectoria simple, donde a partir de una solución y mediante la exploración de la vecindad se actualiza la solución actual (Escalador de Colina, Recocido Simulado, Búsqueda Tabú, etc.), o en poblaciones en la que iterativamente se explora un grupo de soluciones (Algoritmos Evolutivos e Inteligencia Colectiva, etc.), que evitan o que reducen la probabilidad de caer en óptimos locales. Estas dos familias tienen características complementarias, de modo que las de trayectoria simple están orientadas a intensificar la exploración en regiones locales, mientras que las basadas en poblaciones ofrecen una mayor diversificación de todo el espacio de búsqueda (37). Muchos han sido los trabajos realizados en la literatura para resolver el problema del agrupamiento multiobjetivo utilizando metaheurísticas (38), (39), (29).

1.5.1 Metaheurísticas poblacionales

Las metaheurísticas poblacionales son aquellas que exploran el espacio de búsqueda a través de un conjunto de soluciones. Inicialmente se comienza con una población de soluciones, luego iterativamente se generan nuevas soluciones y se reemplazan o actualizan las anteriores hasta que se alcance algún criterio de parada. Diversas metaheurísticas de este tipo difieren en la forma en que se llevan a cabo las fases de generación y actualización (37). La mayoría de los algoritmos de este tipo están inspirados en la naturaleza y pueden ser divididos en dos grandes clases: Estrategias Evolutivas (Algoritmos Genéticos, Programación Genética, etc.) y de Inteligencia Colectiva (SI[7]) (Optimización Basada en Enjambre de Partículas, Optimización Basada en Colonia de Hormigas, etc.).

1.5.1.1 Algoritmos Genéticos

Los Algoritmos Genéticos (GAs[8]), según el documento que los popularizó (40), son definidos de la siguiente forma: *"son algoritmos de búsqueda basados en los mecanismos de selección natural y genética".* Están basados en una de las teorías más importantes de la ciencia, la teoría de la evolución y supervivencia de los más aptos (36). Cada generación contiene una población de individuos o cromosomas (posible solución dentro del espacio de búsqueda) codificados a los cuales se les aplica

[7] Swarm Intelligence, por sus siglas en inglés
[8] Genetic Algorithms, por sus siglas en inglés.

una serie de transformaciones (operadores) para simular el proceso de evolución, donde iterativamente son aplicados los siguientes pasos (41):

1) *Selección:* consiste en seleccionar los individuos para la reproducción. El propósito de los mecanismos de selección es asegurar que los individuos más aptos (soluciones más óptimas) sean los que generen nuevos genotipos en el proceso de cruzamiento y generación de nuevas poblaciones. La selección es un método que define aleatoriamente individuos de la población de acuerdo a su valor de adaptabilidad para usarlos en las tareas de recombinación.
2) *Reproducción:* creación de nuevos individuos. Para la generación de nuevos cromosomas, se pueden utilizar operadores de cruzamiento y mutación.
3) *Evaluación:* es el valor del genotipo de un individuo para una función objetivo en particular. Indica no solamente que tan buena es la solución, sino también que tan cerca se encuentra del óptimo global.
4) *Reemplazo:* actualización de la población por los nuevos individuos generados.

1.5.1.2 Estrategia de Algoritmos Genéticos NSGA-II

NSGA-II[9] es una estrategia evolutiva de GAs para problemas de optimización multiobjetivo. Fue propuesto por Kalyanmoy Deb en (42) y descrito en (43). Entre sus características se encuentra la obtención aproximada de la frontera de Pareto basado en el ordenamiento por no dominancia de las soluciones, y la generación iterativa de poblaciones mediante un proceso elitista. El objetivo de este tipo de ordenamiento es realizar una clasificación por frentes, donde los individuos que corresponden al primer frente son los no dominados (mejores soluciones), los que pertenecen al segundo frente son los no dominados en ausencia del frente anterior, y así sucesivamente (44). La siguiente figura ilustra gráficamente la disposición de los objetos en los frentes:

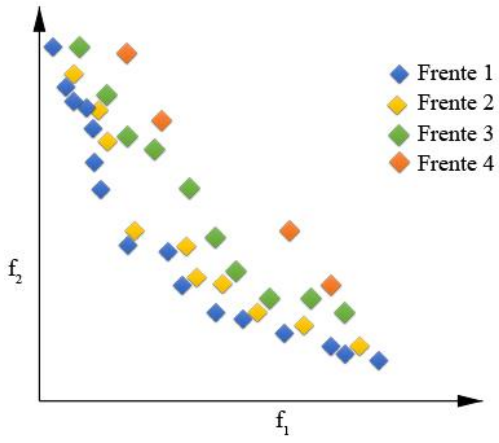


Figura 6 *Conformación de frentes de Pareto en NSGA-II*

[9] Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization, por sus siglas en inglés.

El ordenamiento rápido por no dominancia (Fast Non-Dominated Sorting), requiere de $O(m \times n)$ comparaciones para cada individuo, siendo m el número de funciones objetivos a optimizar y n el tamaño en número de individuos de la población (30).

El algoritmo comienza con la generación de N individuos de la población P_0, luego dicha población es ordenada utilizando el principio de no dominancia. Después se genera una población Q_0 de tamaño N como resultado de haber aplicado operadores de selección, recombinación y mutación a P_0. A partir de la primera generación $(t \geq 1)$ el proceso es diferente. P_t y Q_t son mezcladas, obteniendo una población $R_t = P_t \cup Q_t$ de tamaño $2 \times N$ la cual también es ordenada bajo no dominancia para crear el conjunto de fronteras $\Gamma = (\Gamma_1, \Gamma_2, \dots, \Gamma_k)$. Luego, para construir la población P_{t+1} que trasciende a la siguiente generación, se van añadiendo las soluciones a partir de Γ_1 hasta que P_{t+1} contenga una cantidad de individuos mayor a N, a cada frontera Γ_i añadida se le calcula su distancia de empuje *(crowding distance)*, la cual se define como la suma de las distancias entre las soluciones adyacentes correspondiente a cada dimensión del espacio objetivo (45). Finalmente, se ordena P_{t+1} de acuerdo al criterio de distancia de empuje con el objetivo de romper empates entre soluciones potencialmente competitivas y se seleccionan los N primeros individuos. De esta forma se construye la población P_{t+1} de tamaño N que pasa a la siguiente generación para que le sean aplicados los operadores genéticos y obtener la población resultante Q_{t+1} (42) (46). En la siguiente figura se muestra un diagrama de flujo del algoritmo (47):

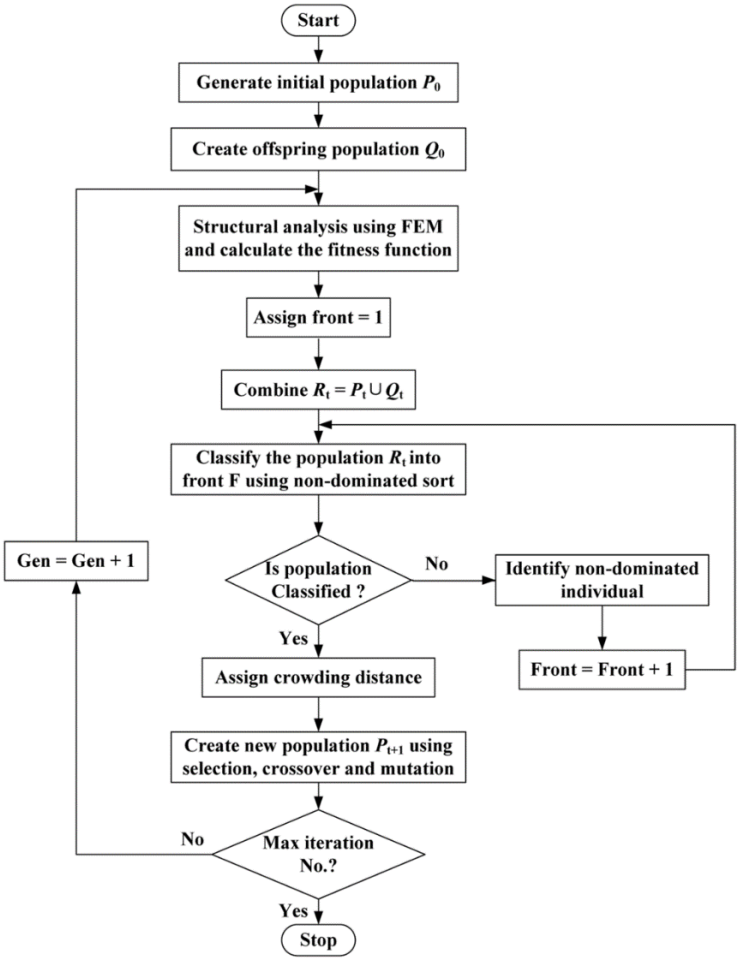


Figura 7 *Diagrama de flujo del algoritmo NSGA-II*

1.5.1.3 Aplicaciones de GAs al problema de agrupamiento multiobjetivo

El teorema "No Free Lunch", plantea que: *por cada par de algoritmos de búsqueda que se considere, hay tantos problemas en los que el primer algoritmo funciona mejor que el segundo, como problemas en los que el segundo algoritmo supera al primero. Es decir, no es posible establecer qué algoritmo funciona mejor en todas las clases de problemas posibles* (48), de ahí que la creación de nuevos métodos y algoritmos que encuentren una solución óptima en todos los escenarios para el problema de optimización multiobjetivo, específicamente el agrupamiento, se ha convertido en una línea de investigación en la que muchas han sido las soluciones propuestas. Según los autores Bong y Rajeswari (49), plantean que en un estudio reciente el desarrollo y uso de técnicas metaheurísticas con enfoque multiobjetivo basadas en la naturaleza, como lo son los Algoritmos Evolutivos Multiobjetivos (MOEAs[10]) para el problema del agrupamiento ha crecido exponencialmente desde el año 2006 hasta el 2010. En la siguiente tabla se ilustran los trabajos revisados en la literatura aplicando algoritmos de este tipo al problema de agrupamiento multiobjetivo:

[10] Multi-Objective Evolutionary Algorithms, por sus siglas en inglés

Referencia	Estrategia multiobjetivo	Tipo de dato	Representación	Funciones objetivo
(50)	MOGA[11]	Continuo	Centroide	Homogeneidad[12], Separabilidad
(51)	MOGA	Continuo	Centroide	Homogeneidad, Separabilidad
(15)	PESA-II[13]	Continuo	Grafo	Compactación (**Ecuación 6**), Conectividad (**Ecuación 7**)
(52)	NSGA-II	Continuo	Centroide	J_m (**Ecuación 9**), XB (**Ecuación 10**)
(29)	NSGA-II	Continuo	Centroide	J_m (**Ecuación 9**), XB (**Ecuación 10**)
(53)	NSGA-II	Continuo	Centroide	Homogeneidad, Separabilidad
(54)	NSGA-II	Continuo	Centroide	Entropía[14], Separabilidad
(55)	NSGA-II	Continuo	Centroide	J_m (**Ecuación 9**), Separabilidad
(56)	MOGA	Categórico	Medoide	Homogeneidad, Separabilidad
(57)	MOGA	Categórico	Medoide	Homogeneidad, Separabilidad
(58)	NSGA-II	Categórico	Medoide	Compactación (**Ecuación 6**), Índice Silhouette
(59)	NSGA-II	Categórico	Centroide	J_m (**Ecuación 9**), Separabilidad
(27)	NSGA-II	Categórico	Medoide	J_m (**Ecuación 9**), Separabilidad (**Ecuación 8)**
(60)	MOGA	Mezclado	Centroide	Homogeneidad, Separabilidad

Tabla 1 *Algunas aplicaciones de GAs al agrupamiento multiobjetivo*

Analizando la **Tabla 1** y a modo de resumen, se observan las siguientes regularidades en la revisión bibliográfica realizada:

1) Amplia utilización de la estrategia evolutiva NSGA-II para la resolución al problema del agrupamiento multiobjetivo.
2) La mayoría de los trabajos están enfocados a escenarios donde las variables son continuas.
3) Dentro de las clases de funciones objetivos a optimizar se destacan compactación y separabilidad.

1.6 Metodologías

Metodología es el conjunto integrado de técnicas y métodos por los cuales se rige una investigación científica**.** Un método es el procedimiento por el cual se lleva a cabo la consecución de determinados objetivos (61).

[11] Multi-Objective Genetic Algorithms, por sus siglas en inglés
[12] Función objetivo que pertenece a la clase de compactación
[13] Pareto Envelope based Selection Algorithm II, por sus siglas en inglés
[14] Función objetivo que pertenece a la clase de compactación

1.6.1 Metodologías para procesos KDD

Tal y como se explicó anteriormente, la realización de un proceso KDD es una tarea no trivial en el que interviene la MD como fase fundamental. Esta fase resulta compleja, por lo cual la utilización de una metodología que guíe la ejecución del proyecto y facilite la planificación, dirección y seguimiento del mismo constituye una buena práctica. Algunas de estas metodologías son: CRISP-DM[15] (62), SEMMA[16] (63), CRITIKAL[17] (64), etc.

Se decidió la utilización de la metodología CRISP-DM ya que las tareas que propone dentro de la fase de minería (modelado) están acordes con la presente investigación, estas son (65):

1) *Selección de la técnica de modelado:* Donde se selecciona de la técnica de MD apropiada al tipo de problema a resolver.
2) *Generación del plan de prueba:* Se crea un procedimiento destinado a probar la calidad y validez del modelo.
3) *Construcción del modelo:* Se escoge la mejor configuración de parámetros del modelo.
4) *Evaluación del modelo:* Se interpretan los modelos de acuerdo al conocimiento preexistente del dominio y los criterios de éxito preestablecidos.

Además, al estar la fase de MD insertada dentro del proceso KDD, no es objetivo de la presente investigación realizar un estado del arte en profundidad que permita discernir entre cuales metodologías utilizar para afrontar esta única fase. A esto se le suma que los coautores de esta investigación poseen experiencia con esta metodología, pues la han utilizado anteriormente en trabajos de pregrado e investigaciones.

1.6.1.1 CRISP-DM

La metodología CRISP-DM define un modelo de procesos jerárquicos para llevar a cabo proyectos KDD. Actualmente es la guía de referencia de mayor uso en el desarrollo de este tipo de proyectos debido a que concibe correctamente el acoplamiento del proceso KDD a un negocio determinado, ha sido creada para la construcción de proyectos de explotación de datos, por su diseño genérico es conocida como metodología imparcial sin importar las herramientas que se utilicen para el desarrollo (66). Consta de seis fases que guían el ciclo de vida durante la implantación de un proceso KDD, las cuales son detalladas brevemente a continuación (67):

[15] CRoss Industry Satandard Process for Data Mining, por sus siglas en inglés
[16] Sample, Explore, Modify, Model, Assess, por sus siglas en inglés
[17] Client-Server Rule Induction Technology for Industrial Knowledge Acquisition from Large Databases, por sus siglas en inglés

1) *Comprensión del Negocio:* Se determinan los objetivos y requerimientos del negocio, se evalúa la situación para definir el problema como un proyecto de MD y se elabora el plan del proyecto.
2) *Comprensión de los Datos:* Se recopilan los datos iniciales, se describen, se verifica la calidad de los mismos y se exploran para detectar los primeros puntos de vista o subconjuntos de carácter interesante que permitan crear hipótesis sobre la información oculta.
3) *Preparación de los Datos:* Se seleccionan, construyen, integran y estructuran los datos para ser minados. Estas tareas incluyen tablas, registros, selección de atributos, así como la transformación y limpieza de los datos para que sean utilizados en las herramientas de modelado.
4) *Modelado:* Se seleccionan y aplican las técnicas de modelado, se genera el diseño del experimento y se construyen y evalúan los modelos.
5) *Evaluación:* Se evalúa a fondo el modelo revisando el proceso para construirlo, se asegura que el modelo cumple apropiadamente con los objetivos del negocio, se determinan temas relevantes en el negocio que no hayan sido suficientemente considerados y se valoran los resultados.
6) *Despliegue:* Se traza la estrategia de empleo de los resultados, se planifica el mantenimiento del proceso, se organiza y presenta el conocimiento obtenido como resultado de manera que el cliente pueda usarlo, y se documentan las experiencias.

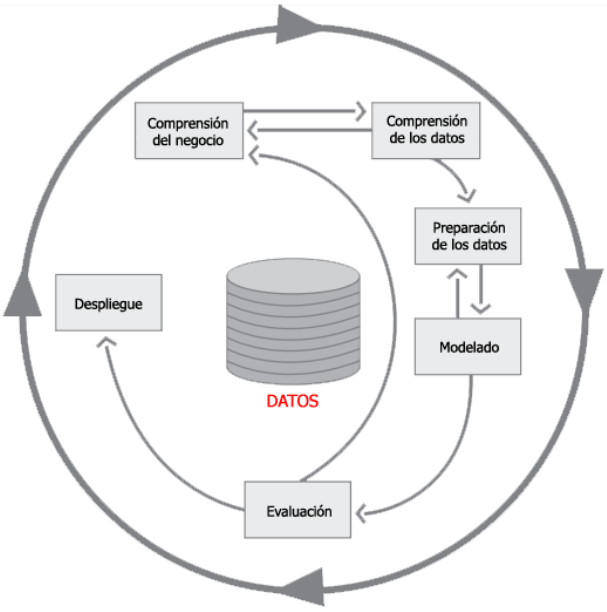


Figura 8 *Fases del modelo de referencia CRISP-DM* (62)

Como se puede apreciar en la **Figura 8**, CRISP-DM plantea un orden lógico en sus fases aunque la secuencia de ellas no es rigurosa ya que en numerosas ocasiones es necesario volver atrás para reanalizar los resultados obtenidos. La salida de cada fase determina la próxima fase o tarea en

particular de ella que será ejecutada. La flechas muestran la frecuente relación que existe entre cada una de las fases, mientras que el círculo que engloba todo el proceso representa la naturaleza cíclica de la MD, debido a que no se termina con el despliegue de la solución, al contrario, el conocimiento obtenido puede provocar un replanteamiento del negocio, donde los procesos de minería subsecuentes se beneficiarán de las experiencias anteriores.

1.6.2 Metodologías de desarrollo de software

Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de sistemas de información. Constituye un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Las metodologías aplicadas en el desarrollo de software optimizan el proceso y el producto software, ofrecen métodos que guían en la planificación y en el desarrollo del software además definen qué hacer, cómo y cuándo durante todo el desarrollo y mantenimiento de un proyecto (68).

Según la filosofía de desarrollo pueden ser clasificados en dos grupos: tradicionales (RUP[18]), que se centran en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir y las herramientas y notaciones que se usarán, y ágiles (XP[19]), las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

1.6.2.1 RUP

Según (69), RUP es un proceso de ingeniería de software bien definido y estructurado en el que se plantea claramente quién es responsable de qué, cómo y cuándo se hacen las cosas. RUP, como metodología de desarrollo de software, integra todos los aspectos a tener en cuenta durante el ciclo de vida del software, con el objetivo de ser abarcable tanto a pequeños como a grandes proyectos (70).

RUP es una metodología guiada por casos de usos lo cual permitió el desarrollo del módulo a través de una serie de flujos de trabajo y posteriormente la realización de cada uno de ellos. Se afirma además que está centrada en la arquitectura donde se involucran los elementos más significativos del sistema, permite tener una radiografía del diseño del módulo, resaltando los elementos arquitectónicos fundamentales que constituyeran las bases del sistema a desarrollar. La presente investigación, en términos prácticos, se adjunta como un módulo de MD para la extracción de patrones sobre un sistema ya existente (GRHS). Actualmente toda la documentación y artefactos de diseño de la plataforma GRHS

[18] Rational Unified Process, por sus siglas en inglés
[19] eXtreme Programming, por sus siglas en inglés

están sustentados bajo la metodología RUP, por tal motivo resulta conveniente su elección con el objetivo de continuar con esta guía durante el proceso de desarrollo.

1.7 Marcos de trabajo para problemas de optimización multiobjetivo

El agrupamiento se trata como un problema de optimización que para resolverlo es conveniente la utilización de marcos de trabajo[20] que implementen algoritmos metaheurísticos. Constituye una herramienta valiosa que ayuda a aplicar nuevas ideas, facilitado por la reutilización de código y entendimiento de algoritmos ya existentes. Tal como se plantea en (71), características deseables de este tipo de herramientas comprenden:

1) El estado del arte de los métodos existentes.
2) Problemas de optimización multiobjetivos convencionales de referencia.
3) La inclusión de indicadores de calidad que evalúen el desempeño de los algoritmos en un problema determinado.
4) Asistencia a los usuarios en la realización de estudios de investigación.

Es por ello que resulta de interés en este trabajo seleccionar un *framework* para desarrollar métodos de agrupamiento particional multiobjetivo basados en algoritmos metaheurísticos como GAs y PSO para la extracción de patrones en términos de grupos sobre la plataforma GRHS. A continuación se describen dos de ellos (ECJ[21] y jMetal[22]).

1.7.1 ECJ

Es un marco de trabajo de computación evolutiva escrito en Java (72). Está diseñado para el uso intensivo en tarea de experimentación, contando con mecanismos de chequeo y control del trabajo. Soporta variedad de técnicas de computación evolutiva como: algoritmos genéticos, programación genética, estrategias evolutivas, coevolución, optimización basada en enjambre de partículas y evolución diferencial. Cuenta con una gran variedad de operadores de cruzamiento y mutación además de definiciones e interfaces para los principales componentes de un algoritmo genético, lo cual facilita el trabajo a lo hora de aplicar estrategias evolutivas en un contexto práctico. ECJ es una herramienta libre y posee más de diez años de creada, lo cual la convierte en un marco de trabajo maduro y estable.

[20] En la literatura es común el término frameworks
[21] Evolutionary Computing in Java, por sus siglas en inglés
[22] Metaheuristic Algorithms in Java

1.7.2 jMetal 4.5

Es un marco de trabajo escrito en Java y orientado a objetos para la optimización multiobjetivo con técnicas metaheurísticas. Brinda un rico conjunto de clases que se pueden utilizar como los componentes básicos de las técnicas multiobjetivo, de esta manera, aprovechando la reutilización de código, los algoritmos comparten los mismos componentes, tales como operadores genéticos y estimadores de densidad, lo que facilita no solo el desarrollo de nuevas técnicas multiobjetivo, sino también la realización de diferentes experimentos pues cuenta con una colección de problemas clásicos implementados. Incluye técnicas y algoritmos clásicos dentro del estado del arte tales como: NSGA-II, SPEA2, PAES, PESA-II, OMOPSO, MOEA/D entre otros, (73). Gracias a su diseño, es posible modelar un problema de optimización definiendo cada uno de los componentes básicos que lo integran: algoritmo, problema y codificación de una solución. Fue desarrollado por el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga. Es código Open Source y puede obtenerse en: http://www.sourceforge.net/projects/jmetal

Se seleccionó este marco de trabajo ya que se adapta fácilmente a las necesidades de esta investigación, pues implementa estrategias evolutivas de GAs y posee una amplia gama de operadores ya definidos que posibilitaron su reutilización y agilizaron el proceso de desarrollo. Además resulta un marco de trabajo flexible, por lo que se pueden adaptar métodos existentes a un dominio de aplicación específico.

1.8 Lenguajes de programación

1.8.1 Java

Java es un lenguaje de programación de propósito general que soporta las características esenciales del paradigma de programación a objetos: encapsulamiento, herencia y polimorfismo. Hace uso de la definición de entidades formadas por métodos y variables que reciben el nombre de clases. Debido a que el marco de trabajo jMetal está escrito en Java, se adopta este lenguaje para la implementación de un método de agrupamiento particional multiobjetivo adaptado a las características del sistema GRHS.

1.8.2 Python

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Su elegante sintaxis, tipado dinámico y naturaleza interpretada, hacen de éste un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (74). Fue

necesaria su utilización en la creación de la vista del módulo ya que el sistema GRHS esta implementado sobre este lenguaje.

1.8.3 JavaScript

JavaScript es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase. Es un lenguaje multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa (75). Ofrece muchas posibilidades y es utilizado para crear programas que son insertados en páginas web. Con JavaScript se pueden crear diferentes efectos gráficos e interactuar con usuarios (76). En la presente investigación se utiliza este lenguaje para la representación gráfica de los resultados obtenidos.

1.8.4 HTML[23]

HTML es el lenguaje de marcas de texto utilizado normalmente en la WWW[24] para el desarrollo de páginas de internet. Es un lenguaje de composición de documentos que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque sí le indica cómo desplegar el contenido del mismo, incluyendo texto, imágenes y otros medios soportados. Está compuesto por una serie de etiquetas que el navegador interpreta y da forma en la pantalla (77).

1.9 Formato de intercambio de datos entre el cliente y el servidor (JSON[25])

Es un formato ligero de intercambio de datos que es completamente independiente del lenguaje, utiliza convenciones ampliamente conocidas por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. Está constituido por dos estructuras: colección de pares de nombre/valor y lista ordenada de valores (78). Fue útil su utilización ya que permitió el intercambio efectivo de información referente a los patrones encontrados entre el cliente y el servidor sobre la arquitectura de la solución propuesta.

1.10 Entorno de desarrollo integrado

1.10.1 NetBeans 8.0.1

NetBeans es un Entorno de Desarrollo Integrado (IDE[26]), donde jMetal se integra fácilmente. Ofrece además una serie de ventajas que lo hacen realmente codiciado, como: la agilidad y flexibilidad en la

[23] HyperText Markup Language, por sus siglas en inglés
[24] World Wide Web, por sus siglas en inglés
[25] JavaScript Object Notation, por sus siglas en inglés
[26] Integrated Development Environment, por sus siglas en inglés

implementación, la creación de aplicaciones multiplataforma, facilidades y garantías para la migración, facilidad de uso, cumplimiento de regulaciones y flexibilidad entre plataformas.

1.10.2 JetBrains PyCharm 4.0.4

PyCharm es un IDE multiplataforma desarrollado por la compañía Jetbrains utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica y soporte para el desarrollo web con Django, entre otras funcionalidades (79).

Algunas características de este IDE son (80):

- Autocompletado, resaltador de sintaxis, herramienta de análisis y refactorización.
- Integración con frameworks web como: Django, Flask, Pyramid, Web2Py.
- Gestión de los intérpretes de Python con una nueva interfaz de usuario
- Soporta entornos virtuales e intérpretes de Python 2.x, 3.x, PyPy, Iron Python y Jython.

1.11 Marcos de trabajo para el desarrollo

1.11.1 Django 1.4.5

Django es un marco de trabajo web de código abierto escrito en Python que permite construir aplicaciones web más rápido y con menos código (81). Ofrece un conjunto de herramientas para agilizar la creación de páginas web siguiendo los principios DRY[27], para evitar duplicidad en las líneas de código e invertir el menor esfuerzo posible. También implementa el diseño Modelo-Vista-Controlador (MVC), por lo que las diferentes partes del sitio están claramente separadas (82). Su utilización fue imprescindible para el desarrollo del módulo ya que Django es utilizado como marco de trabajo para el desarrollo del sistema GRHS.

1.11.2 Highcharts 4.1.4

Highcharts es una biblioteca escrita en JavaScript que permite construir gráficos interactivos para proyectos web. Posee una gran cantidad de interfaces que permiten representar gráficamente cualquier información. Tiene algunas características de gran alcance y está basada en tecnologías de navegación nativas, no necesita *plug-ins* y es compatible con todos los navegadores modernos incluyendo sus versiones móviles. Es una biblioteca de código libre que tiene soporte para gráficos de tipo línea, área, columnas, barras, circulares, dispersión, patrones angulares, burbuja, diagrama de caja, cascada, gráficos polares, entre otros (83). Se considera que fue muy útil su empleo en la

[27] Don’t Repeat Yourself, por sus siglas en inglés, su significado en español es: No Te Repitas

representación de los patrones resultantes puesto que ofrece un conjunto de gráficas que facilitan la observación e interpretación de los resultados después de aplicar un proceso de agrupamiento.

1.12 Herramienta CASE

Dentro de las herramientas claves en el desarrollo de aplicaciones informáticas se encuentran las de Ingeniería de Software Asistida por Ordenador (CASE[28]), las cuales son utilizadas para apoyar las actividades del proceso de software, tales como: ingeniería de requerimientos, diseño, desarrollo de programas y pruebas (84). Durante el ciclo de desarrollo pueden ayudar en el proceso de diseño del proyecto, en el cálculo de costes, pueden implementar una parte del código, compilación automática y documentación. En el presente trabajo se utilizó Visual Paradigm para UML.

1.12.1 Visual Paradigm 8.0

Es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Emplea UML[29] (Lenguaje de Modelado Unificado) como lenguaje de modelado. Se diseñó para distintos usuarios interesados en la construcción de sistemas de software de forma confiable, utilizando un enfoque orientado a objetos. Unas de sus principales características son (85):

1) Licencia gratuita y comercial.
2) Es fácil de instalar, actualizar y compatible entre ediciones.
3) Capacidades de ingeniería directa e inversa.
4) Generación de código - modelo a código, diagrama a código.
5) Diagramas de Procesos de Negocio.
6) Uso de lenguaje estándar común.

Se decidió utilizar Visual Paradigm 8.0 porque es una herramienta CASE profesional que garantiza y agiliza la creación de los diferentes tipos de diagramas propuesto por la metodología RUP. Esta herramienta presenta licencia gratuita y comercial, además la universidad cuenta con la licencia para su uso.

[28] Computer Aided Software Engineering, por sus siglas en inglés
[29] Unified Modeling Language, por sus siglas en inglés

1.13 Conclusiones parciales

Al término de este capítulo se arribaron a las siguientes conclusiones:

1) Actualmente las funcionalidades del sistema GRHS son insuficientes ante el proceso de obtención de patrones.
2) El equipo de trabajo que interactúa con el sistema GRHS en la UCI no cuenta con información que sirva de como punto de partida al proceso de obtención de patrones en términos de grupos, por tanto, un enfoque multiobjetivo en el que se optimice más de una función, permite descubrir distintos tipos de patrones que aporten información en la toma de decisiones.
3) El agrupamiento particional con enfoque multiobjetivo constituye un problema de optimización complejo, para el cual, se han desarrollado métodos que buscan aproximaciones al óptimo global mediante el uso de técnicas metaheurísticas.
4) Algoritmos de agrupamiento como *K-Means* y *K-Medoids* regularmente convergen a óptimos locales; para ello, se han desarrollado metaheurísticas poblacionales como los GAs que minimizan esta posibilidad.
5) La librería utilizada: jMetal, ofrece la extensibilidad suficiente como para desarrollar nuevos métodos de agrupamiento y adaptarlos a escenarios reales como la plataforma del GRHS.

CAPÍTULO II
PROPUESTA DE SOLUCIÓN

2.1 Introducción

Este capítulo contiene la descripción de la propuesta de solución. Para un mejor entendimiento del algoritmo de agrupamiento diseñado, se brinda detalladamente una explicación de su funcionamiento y se describen las etapas por las que transita el proceso de búsqueda de las soluciones. Además, se relacionan los requisitos funcionales, no funcionales, modelo de casos de uso del sistema y otros artefactos propios de la metodología RUP.

2.2 Propuesta de solución

Debido a las insuficiencias detectadas sobre el sistema GRHS en el análisis de la información recopilada para el apoyo a la toma de decisiones, se decidió la creación de un módulo de MD que: **(1)** consuma información de un origen de vista minable que almacene datos pertinentes al registro de incidencias del sistema GRHS, **(2)** obtenga patrones en términos de grupos que expliquen comportamientos en esos datos y **(3)** permita la visualización de los resultados de manera eficiente para que puedan ser interpretados fácilmente y dicho conocimiento sirva de sustento al criterio de directivos en la toma de decisiones.

2.3 Selección del algoritmo de agrupamiento

Como parte de la fase de modelado de la metodología CRISP-DM, fue necesaria la selección de un algoritmo de agrupamiento multiobjetivo adaptable a las características del dominio de aplicación del sistema GRHS. Para ello, la presente investigación tomó como referencia el trabajo realizado en el 2013 titulado "*Hybrid Evolutionary Multiobjective Fuzzy C-Medoids Clustering of Categorical Data*" (27).

El algoritmo de agrupamiento formulado por los autores Anirban Mukhopadhyay, Ujjwal Maulik y Sanghamitra Bandyopadhyay fue seleccionado como método de agrupamiento para la solución propuesta. Dicho trabajo propone un algoritmo de agrupamiento multiobjetivo automático orientado a entornos difusos para datos categóricos que utiliza los principios del algoritmo *Fuzzy C-Medoids* y de la metaheurística evolutiva de algoritmo genético NSGA-II. También emplea una representación basada en prototipo, específicamente en medoide (**Sección 1.4.2**), donde los cromosomas contienen los medoides de los grupos que representan.

El algoritmo halla automáticamente el número de grupos (K), esto resulta conveniente debido a que no es necesario que el usuario posea conocimiento a priori sobre la cantidad real de grupos (en pocos

casos se conoce) subyacentes en los datos. Como método de agrupamiento multiobjetivo optimiza dos funciones: compactación (**Ecuación 9**) y separabilidad (**Ecuación 8**), las cuales pertenecen a clases de funciones muy utilizadas en la literatura (**Tabla 1**), que se minimizan y maximizan respectivamente. Está orientado a datos categóricos principalmente, lo cual hace conveniente su elección de acuerdo al dominio de los datos (**Sección 1.2.3**). La selección del algoritmo está avalada además porque al no existir conocimiento previo sobre las estructuras internas que pudieran encontrarse en los datos (**Sección 1.2.4**), las fronteras entre los grupos no están bien definidas, por tanto un enfoque difuso expresa mejor el cubrimiento de cada objeto a cada grupo, pues denota para cada uno un grado de pertenencia (**Sección 1.4.1**).

2.4 Aplicación del algoritmo de agrupamiento

Configuración del grado de especificidad por rasgos

Como proceso inmediato previo a la búsqueda de patrones es necesaria la definición por parte del usuario del grado de especificidad de ciertos rasgos. Cada rasgo representa una característica de las incidencias, o sea, una columna en la vista minable que aporta información sobre ellas, estos pueden ser: tipo de incidencia (hardware o software), componente (lector de CD-ROM, teclado, microprocesador, dispositivo de almacenamiento, monitor, tarjeta madre, mouse, RAM, partición, antivirus, BIOS, computadora, usuario, controlador, sistema operativo, programa, dispositivo de inicio, expansión, interfaz de red), estado (adición o sustracción), nivel de importancia (alto, medio o bajo), localización, día, día de la semana, mes, año y sesión (madrugada, mañana, almuerzo, tarde, comida o noche).

Para los rasgos componente, nivel de importancia, día y día de la semana es necesario definir el grado de especificidad con que se compararán las incidencias atendiendo a esos rasgos. Este grado consta de dos niveles: absoluto y genérico. Según la semántica atribuida a estos niveles, cuanto más específica o genérica sea la comparación de las incidencias, más granulares o generales quedarán los grupos resultantes respectivamente, facilitando así la comprensión e interpretación de los mismos, acorde a los intereses del tipo de usuario que desea indagar acerca del comportamiento de las incidencias.

En el rasgo componente, el grado de especificidad genérico emplea el término de jerarquía de concepto, el cual consiste en asociar un grupo de elementos bajo una propiedad común. Con este propósito, son agrupados los componentes de acuerdo a dos categorías: hardware y software.

$$D_{componente}(x_i, x_j) = \begin{cases} 0 & x_i = x_j \\ 0.5 & (x_i \in HW \ \wedge x_j \in HW) \vee (x_i \in SW \ \wedge x_j \in SW) \\ 1 & en\ otro\ caso \end{cases}$$

$$HW = \begin{Bmatrix} cdrom, keyboard, microprocessor, storagedevice, \\ monitor, motherboard, mouse, ram \end{Bmatrix}$$

$$SW = \begin{Bmatrix} partition, antivirus, bios, computer, user, controller, \\ operating\ system, program, bootdevice, expansion, \\ function, standard, networkinterface \end{Bmatrix}$$

Ecuación 12 *Función de comparación del rasgo componente en el grado genérico*

De forma análoga sucede con los rasgos nivel de importancia, día y día de la semana:

$$D_{nivel}(x_i, x_j) = \begin{cases} 0 & x_i = x_j \\ 0.5 & (x_i = bajo \ \wedge \ x_j = medio) \vee (x_i = medio \ \wedge \ x_j = bajo) \vee \\ & (x_i = medio \ \wedge \ x_j = alto) \vee (x_i = alto \ \wedge \ x_j = medio) \\ 1 & en\ otro\ caso \end{cases}$$

Ecuación 13 *Función de comparación del rasgo nivel de importancia en el grado genérico*

$$D_{día}(x_i, x_j) = \begin{cases} 0 & x_i = x_j \\ 0.1 & x_i \in S_k \wedge x_j \in S_k, k = (1,2,3,4) \\ 0.25 & (x_i \in S_k \wedge x_j \in S_{k+1}) \vee (x_i \in S_p \wedge x_j \in S_{p-1}), k = (1,2,3), p = (2,3,4) \\ 0.50 & (x_i \in S_1 \wedge x_j \in S_3) \vee (x_i \in S_3 \wedge x_j \in S_1) \vee \\ & (x_i \in S_2 \wedge x_j \in S_4) \vee (x_i \in S_4 \wedge x_j \in S_2) \\ 1 & en\ otro\ caso \end{cases}$$

$$S_1 = \{1,2,\dots,8\}, S_2 = \{9,10,\dots,16\}, S_3 = \{17,18,\dots,24\}, S_4 = \{25,26,\dots,31\}$$

Ecuación 14 *Función de comparación del rasgo día en el grado genérico*

$$D_{día\_semana}(x_i, x_j) = \begin{cases} 0 & x_i = x_j \\ 0.5 & (x_i \in SM \ \wedge x_j \in SM) \vee (x_i \in FS \ \wedge x_j \in FS) \\ 1 & en\ otro\ caso \end{cases}$$

$$SM = \{lunes, martes, miércoles, jueves, viernes\}, FS = \{sábado, domingo\}$$

Ecuación 15 *Función de comparación del rasgo día de la semana en el grado genérico*

Para todos los rasgos el grado de especificidad absoluto está determinado por la función de comparación por coincidencia simple:

$$D(x_i, x_j) = \begin{cases} 0 & x_i = x_j \\ 1 & en\ otro\ caso \end{cases}$$

Ecuación 16 *Función de comparación de los rasgos en el grado de especificidad absoluto*

Función de disimilaridad entre incidencias

Analogía o similaridad es el parecido que tienen entre sí dos objetos en dependencia de los rasgos que los describen (86). Una función de disimilaridad entre incidencias define qué tan “diferente” es una

incidencia con respecto a otra. Menor valor de disimilaridad expresa mayor valor de "parecido" entre dos incidencias. Constituye un elemento imprescindible para los algoritmos de agrupamiento pues se utiliza en la conformación de los grupos. En la presente investigación, dicha función está definida como la sumatoria de las funciones de comparación por rasgos de la siguiente forma:

$$D(x_i, x_j) = \sum_{k \in X} D_k(x_i, x_j)$$

Ecuación 17 *Función de disimilaridad entre incidencias*

Siendo X el conjunto de rasgos definidos por el usuario en la vista minable.

Representación cromosómica

Cada cromosoma representa una posible solución del problema. En el presente trabajo, una solución es un agrupamiento de n incidencias en K grupos, siendo $K \geq 2$. Para ello, se adopta una representación basada en medoide la cual consiste en un vector $A = [m_1, m_2, \dots, m_K]$ de tamaño $K \times d$, donde cada elemento m_i representa el medoide del grupo i-ésimo descrito por d rasgos y siendo $i = (1, \dots, K)$. Con el objetivo de crear soluciones válidas, se precisa como restricción la desigualdad absoluta entre cada uno de los medoides m_i. Al estar enfocado a entornos difusos, cada solución tiene asociada una matriz de pertenencia $U = [\mu_{ik}]_{n \times K}$ de cada uno de los objetos (incidencias) a los medoides.

Inicialización de la población

La población constituye el conjunto de cromosomas sobre los cuales se aplicarán los operadores evolutivos. Para inicializar la población se construyen T agrupamientos con cantidad de grupos K_i variables en el rango $[2, \dots, K^*]$, siendo T el tamaño de la población y K^* una aproximación estimada del número de grupos, cada uno de ellos internamente escoge K_i medoides aleatorios y sin repetición.

Evaluación de las soluciones

La evaluación de las soluciones permite determinar cuándo un agrupamiento es "mejor" que otro en términos de compactación y separabilidad. Como parte de este proceso, el método realiza una iteración de los pasos definidos por el algoritmo *Fuzzy C-Medoids* (**Sección 1.4.3**).

Pseudocódigo Algoritmo de evaluación de soluciones

```
procedure evaluar(solución)
    solución.ActualizarMatriz()
    solución.ActualizarMedoides()
    solución.ActualizarMatriz()
    solución.setJM(Compactación(solución)) (Ecuación 3)
    solución.setSep(Separabilidad(solución)) (Ecuación 8)
endif
```

Pseudocódigo 1 *Algoritmo de evaluación de soluciones*

En la evaluación de una solución se ejecutan los métodos de actualización de medoides y matriz de pertenencia correspondientes al algoritmo *Fuzzy C-Medoids*, obteniéndose un agrupamiento que minimiza compactación entre grupos (**Ecuación 3**). Además, se califica una solución en términos de separabilidad (**Ecuación 8**), razón por la cual lo convierte en un proceso de optimización multiobjetivo que intenta satisfacer dos criterios. El par de valores correspondientes al resultado de las funciones objetivos es tenido en cuenta posteriormente a la hora de comparar las soluciones.

Operadores

Los operadores de selección, cruzamiento y mutación empleados son los mismos que se proponen en el trabajo original. Para seleccionar los cromosomas candidatos a generar nuevas soluciones se utiliza el operador que trae por defecto el algoritmo NSGA-II, la selección por torneo binario. El operador de cruzamiento se aplica con una probabilidad P_c, en donde el punto de cruzamiento solo puede estar entre dos medoides y es definido de la siguiente forma: sean P_1 y P_2 los cromosomas que se quieren cruzar, M_1 y M_2 la cantidad de medoides de cada uno de esos cromosomas respectivamente, λ_1 el punto de cruzamiento en P_1 definido como $\lambda_1 = rand()\%M_1$, λ_2 el punto de cruzamiento en P_2 el cual varía entre $\left[\underline{\lambda_2}; \overline{\lambda_2}\right]$ donde $\underline{\lambda_2}$ y $\overline{\lambda_2}$ indican el límite inferior y superior respectivamente del rango de λ_2, $\underline{\lambda_2} = min\left[2; max[0; 2 - (M_1 - \lambda_1)]\right]$ y $\overline{\lambda_2} = \left[M_2 - max[0; (2 - \lambda_1)]\right]$, si $\overline{\lambda_2} \geq \underline{\lambda_2}$ entonces λ_2 es generado aleatoriamente entre $\underline{\lambda_2}$ y $\overline{\lambda_2}$, en caso contrario $\lambda_2 = 0$. Una vez definidos λ_1 y λ_2, son cruzados P_1 y P_2 para generar dos soluciones $H_1 = \left(\cup_{i=0}^{\lambda_1} P_{1i}\right) \cup \left(\cup_{i=\lambda_2}^{M_2} P_{2i}\right)$ y $H_2 = \left(\cup_{i=0}^{\lambda_2} P_{2i}\right) \cup \left(\cup_{i=\lambda_1}^{M_1} P_{1i}\right)$ donde P_{1i} y P_{2i} indican el i-ésimo medoide de los cromosomas P_1 y P_2 respectivamente. Para el caso del operador de mutación sobre un cromosoma P_i se selecciona aleatoriamente un medoide del mismo y se intercambia por alguno de los objetos del conjunto de datos de manera tal que no existan medoides repetidos, este proceso ocurre con una probabilidad P_m.

Proceso evolutivo

El algoritmo comienza con la inicialización y evaluación de la población. Posteriormente se inicia el proceso evolutivo, generando en cada iteración una población de individuos producto de aplicar los

operadores de selección, cruzamiento y mutación. Dicha población se mezcla con la población principal y se ordena bajo el principio de no dominancia de Pareto (**Sección 1.4.5**) implementado en la estrategia NSGA-II.

El proceso evolutivo de las soluciones puede verse como una gran cantidad de agrupamientos que constantemente son evaluados para que evolucionen independientemente, con el objetivo de optimizar las funciones objetivos y que a su vez generen nuevos agrupamientos mediante la aplicación de operadores de mutación y cruzamiento. La evaluación de un agrupamiento trae consigo la evolución del mismo pues necesita de los pasos de actualización de medoides y de matriz de pertenencia, lo cual, la mayoría de las veces, genera cambios en los medoides y con ello la correspondencia de cada objeto a cada grupo. Por tanto, la optimización de los criterios de compactación y separabilidad está influenciada por el propio proceso evolutivo al aplicar los operadores y por la recolocación de los medoides del algoritmo *Fuzzy C-Medoids*.

La población de individuos que pasa a la siguiente generación es resultado de escoger las mejores soluciones de las fronteras de Pareto halladas mediante el ordenamiento por no dominancia.

Método híbrido

El algoritmo combina técnicas para generar sus soluciones. Es un híbrido entre la estrategia evolutiva NSGA-II y el algoritmo de agrupamiento *Fuzzy C-Medoids*, en el que las bondades que ofrecen cada uno son aprovechadas con el objetivo de mitigar las limitantes del otro. La mayor desventaja del *Fuzzy C-Medoids* es que tiende a converger en óptimos locales, lo cual se compensa con la idea de utilizar una estrategia evolutiva, ya que al mantener constantemente un grupo de soluciones existe poca probabilidad de caer en óptimos locales. Otra desventaja es que también necesita como entrada el número de grupos; no obstante, al mezclar este algoritmo con la estrategia evolutiva NSGA-II, dicha entrada se vuelve innecesaria pues al codificar inicialmente los cromosomas con variable número de grupos y aplicar el operador de cruzamiento durante el proceso evolutivo, solo trascienden las soluciones cuyo K es el óptimo.

Pseudocódigo Algoritmo diseñado HEMFCM

```
procedure HEMFCM_Algorithm(tamañoPoblación, evaluacionesMáximas)
t ← tamañoPoblación
maxEvals ← evaluacionesMáximas
evals ← 0
población ← InicializarPoblación()
for i ← 0 to t do
   evaluar(población[i])                                    (Ver Pseudocódigo 1)
endfor
while evals < maxEvals do
   offspringPopulation ← ∅
```

```
    //GENERACIÓN DE NUEVAS SOLUCIONES
    for i ← 0 to t/2 do
       if evals < maxEvals then
          p1 ← OperadorSelección(población)
          p2 ← OperadorSelección(población)
          hijos[] ← OperadorCruzamiento(p1,p2)
          OperadorMutación(hijos[0])
          OperadorMutación(hijos[1])
          evaluar(hijos[0])                                    (Ver Pseudocódigo 1)
          evaluar(hijos[1])                                    (Ver Pseudocódigo 1)
          offspringPopulation ← offspringPopulation ∪ hijos
          evals ← evals + 2
       endif
    endfor
    //ESTRATEGIA NSGA-II
    U ← población ∪ offspringPopulation
    R ← OrdenarPorNoDominancia(U)
    restantes ← t
    índice ← 0
    población.borrar()
    frontera ← R.ObtenerFrontera(índice)
    while restantes > 0 and restantes >= |frontera| do
       AsignarDistanciaDeEmpuje(frontera)
       población ← población ∪ frontera
       restantes = restantes - |frontera|
       índice ← índice + 1;
       if restantes > 0 then
          frontera ← R.ObtenerFrontera(índice)
       endif
    endwhile
    if restantes > 0 then
       AsignarDistanciaDeEmpuje(frontera)
       OrdenarPorDistanciaDeEmpuje(frontera)
       población ← población ∪ frontera
       restantes ← 0
    endif
endwhile
R ← OrdenarPorNoDominancia(población)
población ← R.ObtenerFrontera(0)
solución ← población.MejorIndiceDeSilueta() ∪ población
return solución
```

Pseudocódigo 2 *Algoritmo de agrupamiento híbrido diseñado bajo la estrategia NSGA-II*

En el **Pseudocódigo 2** se puede observar el carácter híbrido del método propuesto, al mezclar la estrategia evolutiva para problemas de optimización multiobjetivo que define NSGA-II con un método para agrupamiento particional como lo es *Fuzzy C-Medoids*.

Solución final

Como solución final se devuelve la que posea mejor índice de silueta[30] (27) dentro de las soluciones no dominadas (en el **Pseudocódigo 2** frontera 0). Este índice es un método que mide compactación y

[30] En la literatura aparece con el nombre de Silhouette Index

separabilidad en un agrupamiento, el cual varía entre $[-1; 1]$. Está definido de la siguiente forma: sea a el promedio de disimilitud de un objeto x_i al resto de los objetos del grupo al cual él pertenece y b el mínimo promedio de disimilitud a los objetos de otros grupos, el índice de la silueta para x_i equivale a $s(x_i) = \frac{b-a}{max\{a,b\}}$. El índice total $S(K)$ es el promedio de los índices de todos los objetos, por tanto $S(K) = \frac{1}{n}\sum_{i=1}^{n} s(x_i)$. Mientras $S(K)$ se acerque más a 1 implica mejores resultados en el agrupamiento (14). Aunque esta sea la solución que se recomiende como resultado final, en el presente trabajo se decidió devolver además, todas las soluciones no dominadas del conjunto de Pareto, con el fin de ofrecer una gama de agrupamientos entre los cuales ninguno es mejor que otro en términos de compactación y separabilidad para expresar distintos patrones en los datos, aprovechando así las posibilidades que ofrece un proceso de optimización multiobjetivo.

2.5 Requisitos funcionales

Un requisito de software es la necesidad que tiene un usuario de que el software sea capaz de resolver un problema o de alcanzar un objetivo (87). A continuación se muestran los requisitos funcionales con los que debe cumplir el módulo.

R1 Configurar grados de especificidad: Selecciona el grado de especificidad por cada rasgo.

R2 Buscar patrones: Aplica el algoritmo de agrupamiento sobre la vista minable previamente configurada por el usuario.

R2.1 Consumir vista minable: Consume la vista minable.

R2.2 Aplicar algoritmo: Aplica el algoritmo de agrupamiento a la vista minable.

R3 Visualizar patrones: Representa gráficamente los resultados obtenidos.

2.6 Requisitos no funcionales

Los requisitos no funcionales definen cualidades globales del sistema y describen las características y restricciones que gobiernan su desarrollo (88). A continuación se describen los requisitos no funcionales que se deben tener en cuenta durante la implantación de la solución propuesta.

2.6.1 Requisitos de Hardware

Los requerimientos de hardware de la estación de trabajo donde se va a desplegar la aplicación son:

- 2 GB de RAM
- Procesador multinúcleo, preferiblemente Intel® Core™ i3 o superior

Los requerimientos de memoria RAM varían en dependencia de la cantidad de entradas existentes en el registro de incidencias y la cantidad de rasgos presentes en la vista minable. En este sentido, se observa que con más de 5000 registros de incidencias y teniendo en cuenta todos los rasgos es necesario aumentar la RAM a más de 2 GB.

2.6.2 Requisitos de Software

En relación con el software, la estación de trabajo deberá contar con la instalación de:

- JDK[31]
- Servidor de Aplicaciones Web Tomcat

2.6.3 Requisitos de Eficiencia

Los algoritmos evolutivos, debido a su naturaleza de manejar muchas soluciones al mismo tiempo, frecuentemente tardan en devolver una respuesta óptima. Con el objetivo de minimizar el tiempo de ejecución del algoritmo y aprovechar más los recursos del procesador, fue necesaria la evaluación de las soluciones de forma paralela haciendo uso de cada procesador físico y lógico instalado, esto fue posible gracias a una de las clases integradas en el marco de trabajo jMetal (*ParallelEvaluator*).

2.7 Modelo de Casos de Uso del Sistema

Un caso de uso (CU) es una secuencia de acciones que el sistema lleva a cabo para ofrecer algún resultado de valor para un actor. Un modelo CU está compuesto por todos los actores y CU del sistema, así como sus relaciones (89).

2.7.1 Actores del Sistema

Los sistemas interactúan con usuarios, cada tipo de usuario se representan mediante un actor, quien utiliza el sistema para interactuar con los CU (89). En la propuesta de solución intervienen dos actores: especialista o directivo de los recursos de hardware y/o software de una entidad interesado en obtener patrones de comportamiento sobre el registro de incidencias del sistema GRHS y el sistema que implementa el algoritmo de agrupamiento.

ACTOR	DESCRIPCIÓN
Interesado	Persona que necesita conocer patrones de comportamiento sobre el registro de incidencias que le ayuden en la toma de decisiones.
Constructor de Grupos	Representa la aplicación que contiene el algoritmo de agrupamiento.

Tabla 2 *Actores del sistema*

[31] Java Development Kit, por sus siglas en inglés

2.7.2 Diagrama de CU del Sistema

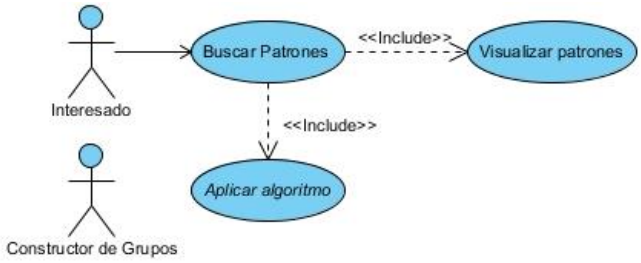


Figura 9 *Diagrama de CU del sistema*

2.7.3 Descripción del Caso de Uso Buscar Patrones

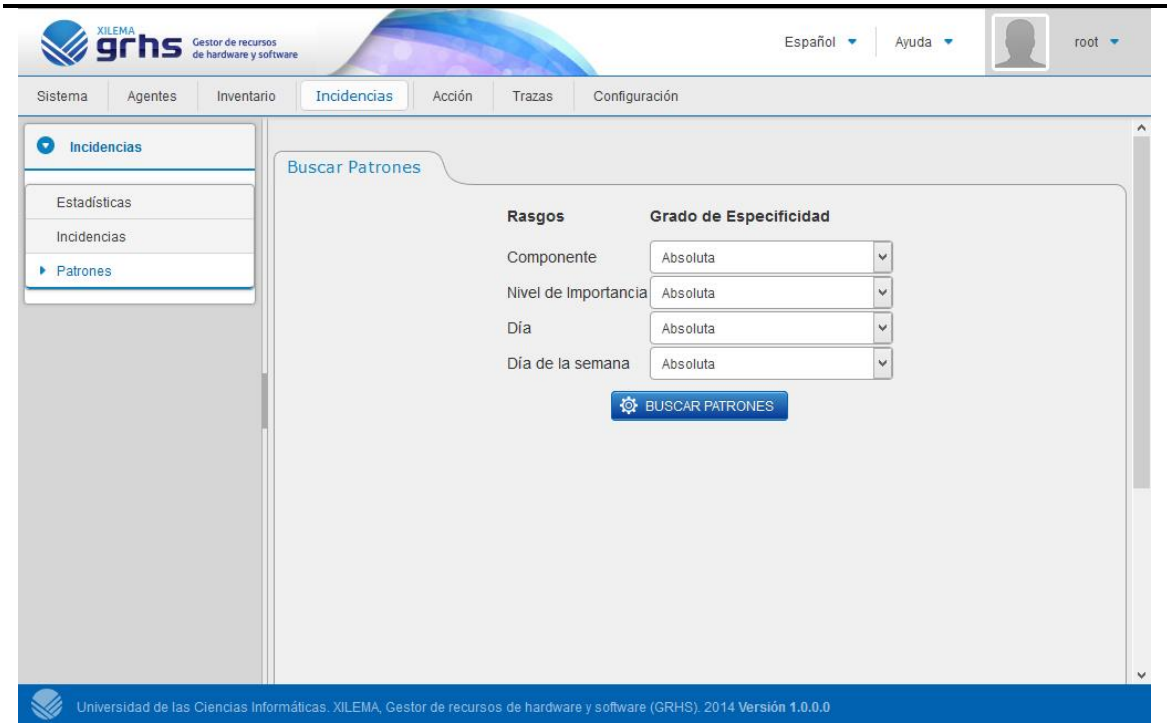
CASO DE USO	
CU-1	Buscar Patrones
Actor	Interesado
Resumen	Se declara el grado de especificidad por rasgos, se aplica el algoritmo de agrupamiento y los resultados son representados gráficamente.
Precondiciones	El interesado debe estar autenticado en el sistema
Referencia	R1, CU-2 (inclusión), CU-3 (inclusión)
Prioridad	Crítico

FLUJO NORMAL DE EVENTOS

SECCIÓN "Buscar Patrones"

Acción del actor	*Respuesta del Sistema*
El interesado define por cada rasgo su nivel de especificidad y selecciona la opción de Buscar Patrones.	1. Ordena al Constructor de Grupos a que ejecute el algoritmo de agrupamiento empleando los niveles de especificidad definidos sobre la vista minable (Véase CU-2). 2. Muestra los patrones encontrados (Véase CU-3).

PROTOTIPO DE INTERFAZ



FLUJOS ALTERNOS

Acción del actor	*Respuesta del Sistema*
	1. Error en la conexión al servidor: el sistema muestra un mensaje de error.
Poscondiciones	Representación gráfica de los patrones encontrados

Tabla 3 *Descripción detallada del CU Buscar Patrones*

2.7.4 Descripción del Caso de Uso Aplicar Algoritmo

CASO DE USO

CU-2	Aplicar Algoritmo
Actor	Constructor de Grupos
Resumen	Aplica el algoritmo de agrupamiento a la vista minable.
Precondiciones	Haber definido previamente la vista minable
Referencia	R2, R2.1, R2.2
Prioridad	Crítico

FLUJO NORMAL DE EVENTOS

SECCIÓN "Aplicar Algoritmo"

Acción del actor	*Respuesta del Sistema*
Recibe la petición de buscar patrones proveniente del sistema principal	1. Accede al origen de la vista minable. 2. Aplica el algoritmo de agrupamiento siguiendo las métricas de especificidad. 3. Devuelve los patrones encontrados al sistema

PROTOTIPO DE INTERFAZ	
No tiene	
Poscondiciones	Conjunto de agrupamientos

Tabla 4 *Descripción detallada del CU Aplicar Algoritmo*

2.7.5 Descripción del Caso de Uso Visualizar Patrones

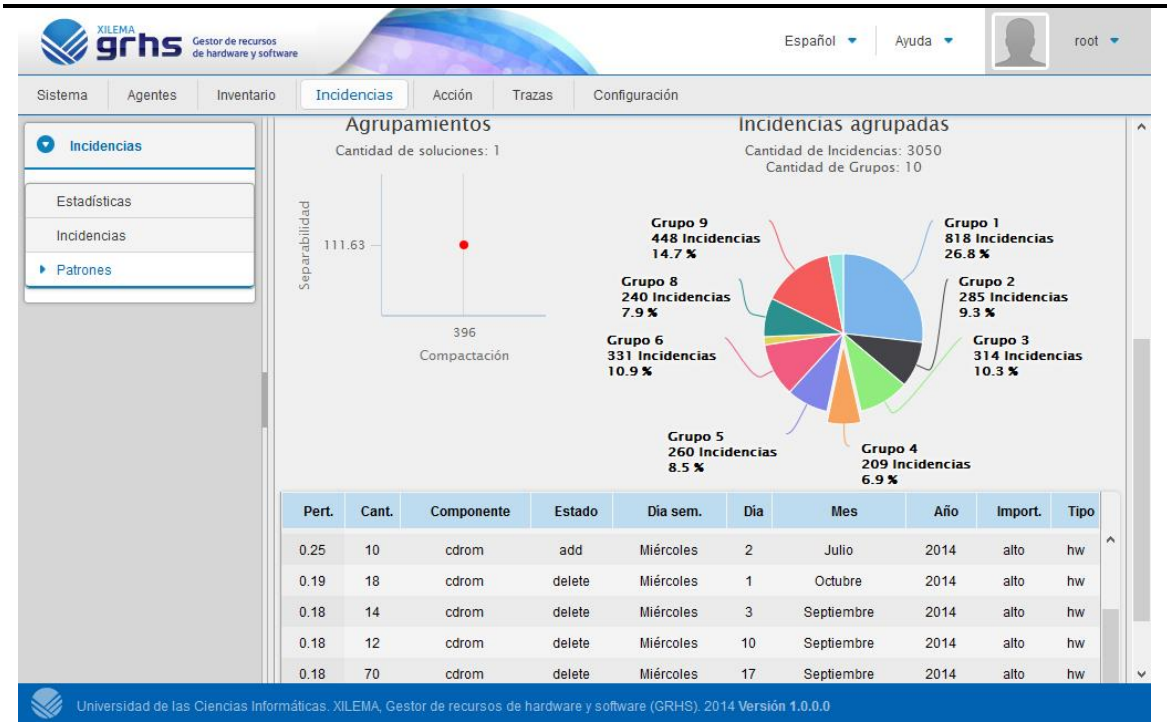
CASO DE USO	
CU-3	Visualizar Patrones
Actor	Interesado
Resumen	Representa gráficamente los patrones encontrados
Precondiciones	Conjunto de agrupamientos
Referencia	R3
Prioridad	Crítico

FLUJO NORMAL DE EVENTOS

SECCIÓN "Visualizar Patrones"

Acción del actor	*Respuesta del Sistema*
	1. Obtiene el listado de los patrones encontrados.
	2. Muestra gráficamente los distintos patrones encontrados.

PROTOTIPO DE INTERFAZ



Poscondiciones	Gráfico de grupos de incidencias

Tabla 5 *Descripción detallada del CU Visualizar Grupos*

2.8 Arquitectura de la solución

Figura 10 *Arquitectura cliente-servidor de la solución propuesta*

En la **Figura 10** se muestra un esbozo de la arquitectura cliente-servidor correspondiente a la propuesta de solución, la cual es una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente aunque se encuentren en diferentes plataformas. En el modelo cliente-servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y éste envía uno o varios mensajes con la respuesta. Según las bases de este concepto en la solución, un cliente sería aquella persona que desea buscar patrones de incidencias sobre el sistema GRHS, para ello, primeramente configura la vista minable y realiza la petición de buscar patrones al servidor, el cual es el encargado de aplicar el algoritmo de agrupamiento y devolver el resultado para que sea interpretado por la máquina del cliente y representado gráficamente. El intercambio de información entre las dos máquinas se efectúa mediante un servicio web publicado por el servidor, sobre la base del formato JSON.

2.9 Diagrama de Clases

El algoritmo fue desarrollado sobre el marco de trabajo jMetal (**Sección 1.7.2**). Para hacer uso del mismo, es necesario implementar las clases abstractas que comparten cualquier problema de optimización, ellas son: *Problem*, *Algorithm* y *Solution*.

En la clase *Problem* es donde interviene la definición del problema a resolver y donde se evalúa una solución en relación a los criterios a optimizar. En ella se modela el agrupamiento como un problema de optimización multiobjetivo donde se optimizan las funciones de compactación y separabilidad. Esta clase recibió el nombre de *FCMC_Problem*[32]. La clase *Algorithm* implementa el método que se utiliza para resolver el problema, esta se denominó *HEMFCM_Algorithm*[33]. Dado que el marco de trabajo jMetal trae implementado la estrategia NSGA-II, solo fue necesario redefinir la forma en que se inicializa la población, ya que NSGA-II y el método propuesto difieren en ese sentido. La clase *Solution*

[32] Su nombre se debe a las siglas de Fuzzy C-Medoids Clustering
[33] Su nombre se debe a las siglas de Hybrid Evolutionary Multiobjective Fuzzy C-Medoids Clustering

representa cómo está estructurada una solución para el problema a resolver (**Sección 2.4**). Para el problema del agrupamiento esta clase fue nombrada *ClusteringSolution*. El marco de trabajo empleado dispone de una serie de problemas, algoritmos, tipos de soluciones y operadores ya programados que agilizan el proceso de desarrollo, por ejemplo: el operador de selección *BinaryTournament* que utiliza la investigación sobre la cual se basa el presente trabajo. No obstante, fue pertinente desarrollar los operadores de cruzamiento (*ClusteringCrossover*) y mutación (*ClusteringMutation*), pues su funcionamiento y restricciones que poseen están muy vinculados al problema del agrupamiento.

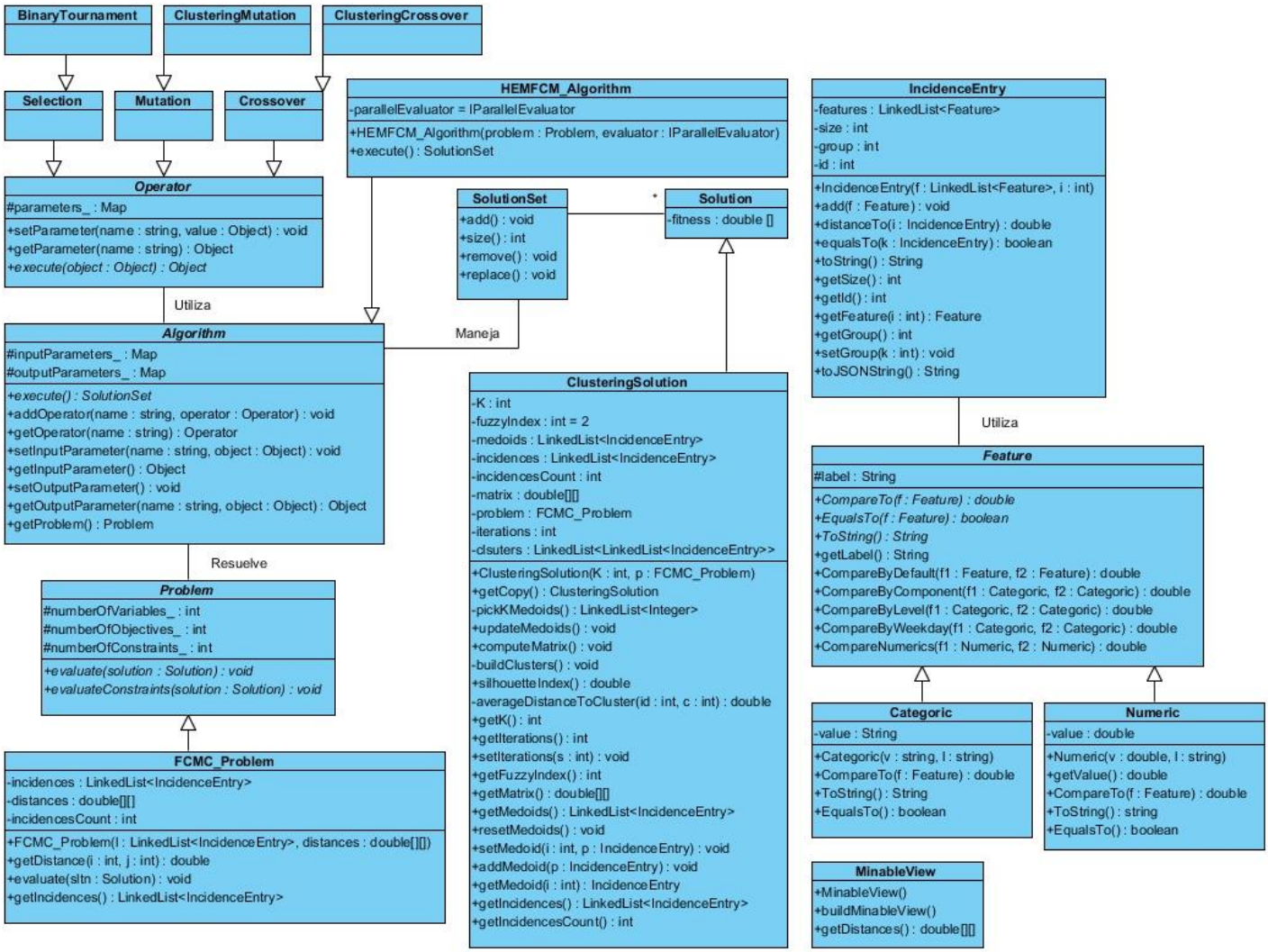


Figura 11 *Diagrama de clases de la solución*

Clases implementadas:

- *HEMFCM_Algorithm:* Clase principal donde se ejecuta el algoritmo de agrupamiento.
- *ClusteringMutation*: Operador de mutación.
- *ClusteringCrossover*: Operador de cruzamiento.
- *FCMC_Problem*: Clase encargada de evaluar una solución.

- *ClusteringSolution*: Contiene características de un agrupamiento: número de grupos, lista de objetos tipo incidencias, lista de medoides, matriz de pertenencia, entre otros.
- *IncidenceEntry*: Representa un objeto incidencia descrito por un listado de rasgos.
- *Feature*: Clase abstracta que representa un rasgo.
- *Categoric*: Clase que hereda de *Feature* especializada en rasgos categóricos.
- *Numeric*: Clase que hereda de *Feature* especializada en rasgos numéricos.
- *MinableView*: Define la interfaz que va a acceder al origen de vista minable.

2.10 Diagrama de Despliegue

El diagrama de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista del despliegue de los artefactos (90). Cada artefacto representa un elemento de hardware necesario para el despliegue de la aplicación.

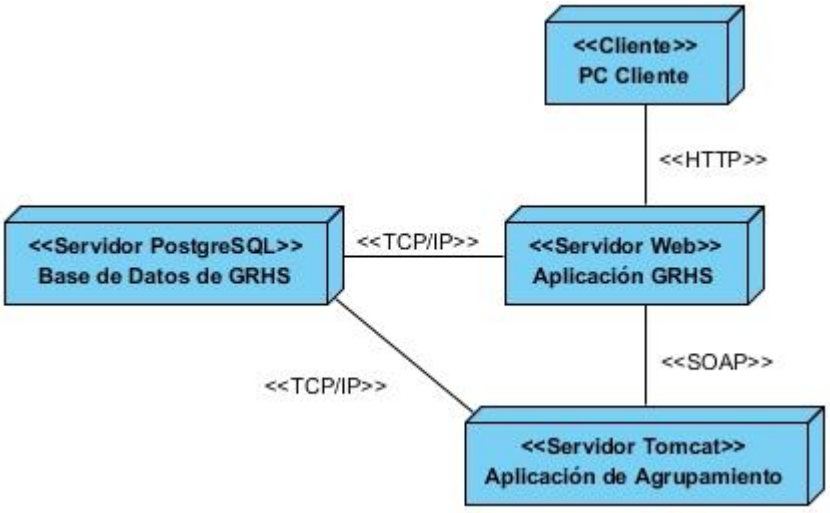


Figura 12 *Diagrama de despliegue*

2.11 Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación.

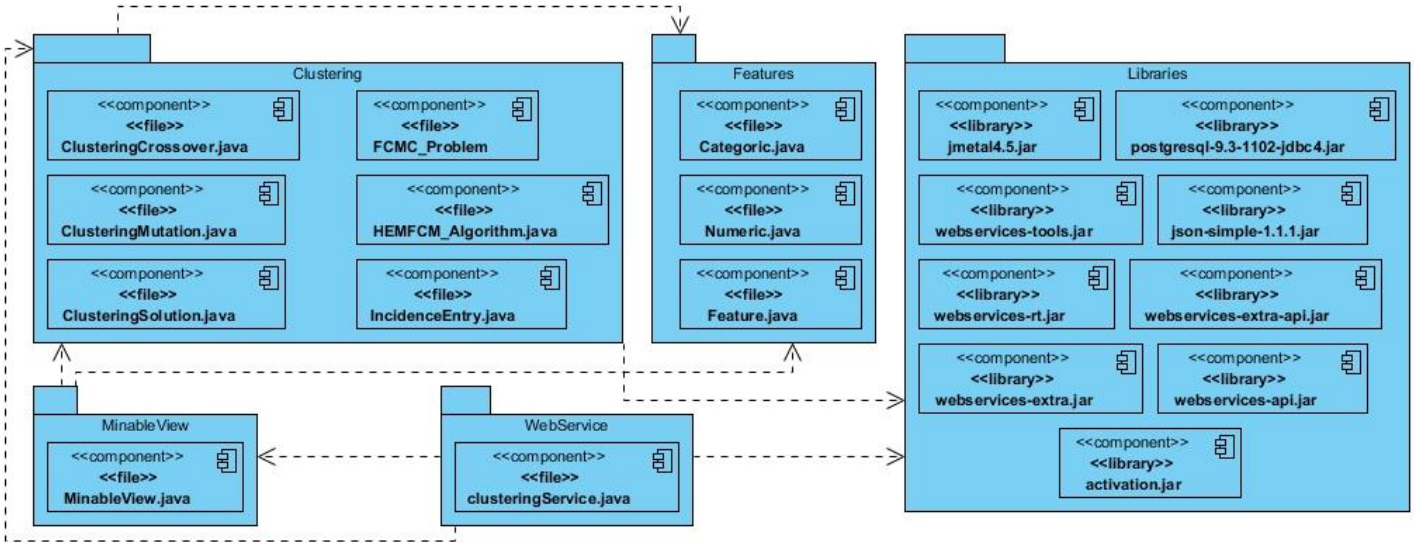


Figura 13 *Diagrama de componentes*

2.12 Conclusiones parciales

Luego del diseño y desarrollo de la propuesta de solución se arriban a las siguientes conclusiones:

1) El algoritmo de agrupamiento seleccionado se encuentra acorde con las características de la información recopilada por el sistema GRHS.
2) La combinación de la estrategia evolutiva NSGA-II y el algoritmo *Fuzzy C-Medoids* permite reducir la probabilidad de que este último caiga en óptimos locales.
3) Los grados de especificad están sujetos a los intereses de búsqueda de patrones del usuario.
4) Mediante el diseño de los artefactos propuestos por la metodología RUP, se logró una fácil concepción arquitectónica del módulo para su posterior implementación.
5) La Arquitectura Orientada a Objetos implementada en el marco de trabajo jMetal, permitió extender las clases necesarias para el desarrollo del algoritmo, así como la reutilización de código existente, agilizando de esta forma el proceso de implementación.

CAPÍTULO III

EXPERIMENTACIÓN Y PRUEBAS

3.1 Introducción

El desarrollo del software debe ir acompañado de alguna actividad que garantice la calidad. La prueba es un elemento crítico para la garantía de calidad del software. En el presente capítulo se hace alusión a las pruebas realizadas a la aplicación y se muestra cómo son presentados los patrones obtenidos con el fin de que sea lo más práctico posible la comprensión e interpretación de los mismos. Además, se realiza un análisis experimental con los parámetros de entrada a partir de un ejemplo de vista minable.

3.2 Descripción de la vista minable

Para la validación, experimentación y realización de pruebas al módulo de MD propuesto, fue necesaria la creación de una vista minable hipotética que sirviera de entrada al algoritmo. En este sentido, se conformó una vista minable a partir de las incidencias desatadas durante el período del 30 de mayo del 2014 al 1 de octubre del 2014 por el sistema GRHS en los docentes 1 y 3 de la UCI. Esta vista, constituye una tabla en una base de datos Postgres a la cual se accede mediante la aplicación. Es importante destacar que en dependencia de la fuente de datos que contiene la vista minable, es necesaria la modificación de la interfaz que sirve para su consumo. Algunas de estas fuentes pueden ser: archivos de texto, hojas de cálculo en Excel, bases de datos, entre otros.

En la vista minable utilizada se registran los datos de 1209 incidencias diferentes, de las cuales 280 (23,16%) son de software y 929 (76,84%) de hardware. La cantidad de incidencias diferentes no refleja el número real de ellas ya que obviamente puede haber objetos idénticos repetidos, lo cual carece de sentido dentro del contexto de una vista minable. Debido a esto, a cada registro se le asocia su multiplicidad (columna *count* en la **Figura 14**), para un total de 21590 incidencias. Este atributo no es tenido en cuenta dentro del algoritmo, pero sí en la visualización de los patrones obtenidos. En la siguiente figura se muestra una parte de la vista minable empleada:

component character varying(200)	state character varying(6)	weekday character varying(15)	day integer	month character varying(15)	year integer	level character varying(5)	incidencetype character varying(20)	count bigint
antivirus	add	Domingo	6	Julio	2014	medio	sw	5
antivirus	add	Domingo	13	Julio	2014	medio	sw	6
antivirus	add	Domingo	14	Septiembre	2014	medio	sw	4
antivirus	add	Domingo	21	Septiembre	2014	medio	sw	5
antivirus	add	Domingo	31	Agosto	2014	medio	sw	2
antivirus	add	Jueves	3	Julio	2014	medio	sw	3
antivirus	add	Jueves	4	Septiembre	2014	medio	sw	13
antivirus	add	Jueves	10	Julio	2014	medio	sw	41
antivirus	add	Jueves	11	Septiembre	2014	medio	sw	9
antivirus	add	Jueves	17	Julio	2014	medio	sw	5
antivirus	add	Jueves	18	Septiembre	2014	medio	sw	7
antivirus	add	Jueves	25	Septiembre	2014	medio	sw	3
antivirus	add	Lunes	1	Septiembre	2014	medio	sw	52
antivirus	add	Lunes	7	Julio	2014	medio	sw	11
antivirus	add	Lunes	8	Septiembre	2014	medio	sw	33
antivirus	add	Lunes	14	Julio	2014	medio	sw	11
antivirus	add	Lunes	15	Septiembre	2014	medio	sw	2
antivirus	add	Lunes	22	Septiembre	2014	medio	sw	15
antivirus	add	Lunes	29	Septiembre	2014	medio	sw	3
antivirus	add	Lunes	30	Junio	2014	medio	sw	1
antivirus	add	Martes	1	Julio	2014	medio	sw	1
antivirus	add	Martes	2	Septiembre	2014	medio	sw	15

Figura 14 *Vista minable utilizada en las pruebas*

3.3 Parámetros de entrada

Durante el tiempo de ejecución de un algoritmo genético, intervienen una serie de parámetros útiles para su funcionamiento, entre los que se encuentran: tamaño de la población (T), cantidad de generaciones (G), y probabilidad de cruzamiento (P_c) y mutación (P_m). En la estrategia NSGA-II no existe el parámetro G, el mismo puede inferirse a través de la cantidad máxima de evaluaciones (E_{max}), ya que para este algoritmo G equivale a $\frac{E_{max}}{T}$, por tanto es necesario definir E_{max}. Además, como se expuso en el capítulo anterior, el método propuesto es capaz de optimizar el número de grupos (K) durante el proceso evolutivo, para ello es necesario especificar la aproximación superior de K (K^*). Aunque K^* sea el límite superior del rango de valores admisibles para la creación de los agrupamientos iniciales con variable K, puede darse el caso de que una vez concluido el proceso evolutivo existan uno o más agrupamientos cuyo K sea mayor a K^*, esto se debe a que el operador de cruzamiento mezcla soluciones con variable número de grupos.

Con el objetivo de ajustar los parámetros antes mencionados, a continuación se realiza un experimento con diferentes valores a fin de encontrar la parametrización que arroje mejores resultados. Algunos de los valores como P_c y P_m son fijados convenientemente de acuerdo a los resultados obtenidos por los autores de la investigación original. Los valores seleccionados son los siguientes:

Parámetro	Valores	
T	10	20
E_{max}	200	500
P_c	0.8	
P_m	0,01	
K^*	5	15

Tabla 6 *Valores de los parámetros utilizados para el experimento*

3.4 Resultados

La **Tabla 7** muestra el promedio de los resultados obtenidos por el algoritmo utilizando la parametrización antes mencionada después de 5 corridas del mismo por cada combinación de parámetros, alcanzando un total de 40 ejecuciones. De la vista minable se escogieron las 500 primeras incidencias, las cuales fueron comparadas de acuerdo al grado de especificidad absoluto en cada uno de los rasgos. Estos resultados se obtuvieron sobre un ordenador con un procesador de cuatro núcleos AMD A8-3520M APU 1.60 Ghz con 6 GB de RAM.

T	E_{max}	K^*	J_m	SEP	K	$S(K)$	Soluciones	Tiempo (segundos)
10	200	5	150,57	175,70	8	0,6031	2	159
		15	61,89	1140,93	20	0,5865	2	357
	500	5	127,98	273,96	10	0,6431	2	383
		15	61,27	1167,53	20	0,5809	2	884
20	200	5	86,22	460,53	14	0,5659	1	159
		15	39,02	2633,8	30	0,5663	1	405
	500	5	90,91	491,16	14	0,6018	2	511
		15	32,77	4088,43	36	0,6184	3	1350

Tabla 7 *Resumen de los resultados para 500 incidencias*

Las columnas J_m, SEP, K y $S(K)$ muestran los valores en promedio de compactación, separabilidad, número de grupos e índice de silueta respectivamente por cada ejecución del algoritmo. Se puede apreciar que con $K^* = 15$ aumenta el número de grupos de las soluciones obtenidas al concluir el proceso evolutivo, sin embargo no ofrece mejoría relevante en cuanto a $S(K)$, por otra parte, para $E_{max} = 500$ existen mejoras considerables de los valores de las funciones objetivos que se optimizan, esto se debe a que a mayor número de evaluaciones, mayor duración del proceso evolutivo (generaciones) y mayor explotación del espacio de búsqueda. Con respecto al parámetro T, si bien existen casos en el que para $T = 20$ se alcanzaron mayor número de soluciones que para $T = 10$, se pudo observar que de manera general mantienen el mismo comportamiento en cuanto al número de soluciones que arrojan. Una vez realizado el experimento se adoptan las siguientes parametrizaciones:

Parámetro	Valores
T	10
E_{max}	500
P_c	0.8
P_m	0,01
K^*	5

Tabla 8 *Parámetros ajustados una vez terminado el experimento*

3.5 Visualización de los patrones

Las técnicas de visualización de datos se utilizan fundamentalmente con dos objetivos: aprovechar la gran capacidad humana de extraer patrones a partir de imágenes y ayudar al usuario a comprender más rápidamente patrones descubiertos automáticamente por un sistema KDD (91). Como parte de la representación de los patrones se brinda al usuario un conjunto de agrupamientos en términos de compactación y separabilidad. Ninguno de dichos agrupamientos es mejor que otro, solamente describen distintos patrones en sus grupos. La solución con mejor índice de silueta se resalta en rojo tal como se muestra en la **Figura 15**.

Figura 15 *Ejemplo de gráfica que muestra los agrupamientos encontrados*

Para la representación de un agrupamiento se utiliza un gráfico de pastel (**Figura 16**), en donde se puede observar cómodamente la cantidad de incidencias por grupos, así como cuál de ellos ocupa mayor espacio y con ello las incidencias que mayor probabilidad de ocurrencia tienen. Cada región coloreada del gráfico significa un grupo, de cada uno se muestra la cantidad de incidencias que lo integra, el porciento que representa del total y la incidencia representativa del mismo (medoide), tal como se muestra en la **Figura 17**. Para un mayor nivel de detalle, al seleccionar un grupo determinado, son relacionadas todas las incidencias que pertenecen a ese grupo, las cuales aparecen ordenadas por el grado de pertenencia (**Figura 18**). Este grado de pertenencia se encuentra en estrecha relación con el nivel de especificidad por rasgos que define el usuario previamente, expresa cuán similar es un objeto determinado respecto al medoide del grupo y sirve para comprender el nivel de solapamiento entre los grupos y sus fronteras. Las incidencias medoides siempre tienen grado de pertenencia igual a 1.

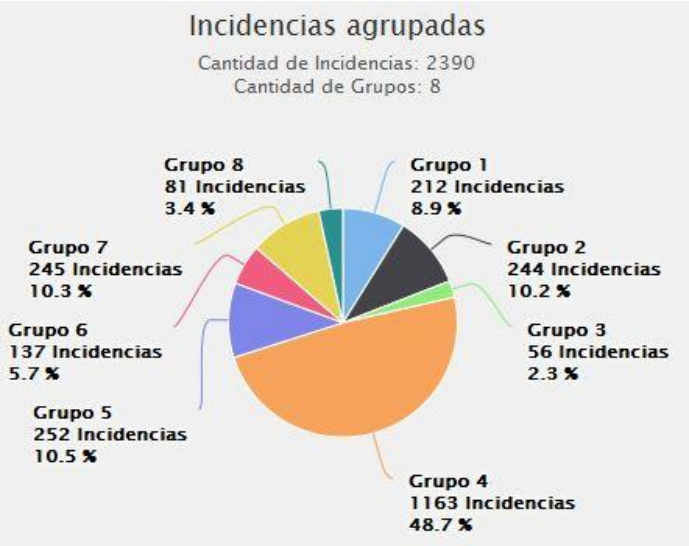


Figura 16 *Ejemplo de gráfico de pastel que muestra los grupos de incidencias*

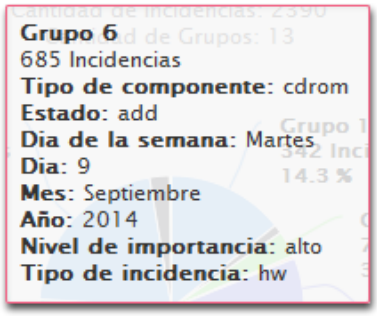


Figura 17 *Ejemplo de medoide de un grupo*

Pert.	Cant.	Componente	Estado	Día sem.	Día	Mes	Año	Import.	Tipo
1	2	cdrom	delete	Martes	15	Julio	2014	alto	hw
0.3	1	cdrom	delete	Martes	1	Julio	2014	alto	hw
0.3	15	cdrom	delete	Martes	8	Julio	2014	alto	hw
0.26	9	cdrom	add	Martes	15	Julio	2014	alto	hw
0.19	19	cdrom	delete	Martes	9	Septiembre	2014	alto	hw
0.19	77	cdrom	delete	Martes	16	Septiembre	2014	alto	hw
0.19	1	cdrom	delete	Martes	17	Junio	2014	alto	hw

Figura 18 *Ejemplo de tabla donde se muestran las incidencias asociadas a un grupo ordenadas por grado de pertenencia*

3.6 Pruebas de Integración

Las pruebas de integración es la fase en pruebas de software cuyos módulos individuales se combinan en grupo. Pertenecen a las pruebas de caja blanca y su propósito es verificar el correcto ensamblaje entre los distintos componentes con el fin de comprobar que interactúan correctamente a través de sus interfaces, tanto internas como externas, cubren la funcionalidad establecida y se ajustan a los requisitos funcionales especificados (92).

CASO DE PRUEBA DE INTEGRACIÓN:

Verificar la integración del módulo de MD en la funcionalidad “Patrones” del sistema GRHS

Número de caso de prueba: 1

Módulo a integrar: Minería de Datos

Condiciones de ejecución: Interfaz o módulo capaz de gestionar la vista minable.

Descripción de la prueba: Comprobar que el módulo de MD obtenga patrones.

Entrada/Pasos de ejecución: El usuario accede al módulo "Patrones", define el grado de especificidad por rasgos y selecciona la opción "Buscar Patrones".

Resultado esperado: Obtención y visualización de los patrones.

Evaluación de la prueba: Prueba satisfactoria

Tabla 9 *Prueba de integración al módulo de minería*

3.7 Pruebas de Aceptación

Son pruebas de caja negra, especificadas por el cliente y se centran en las características y funcionalidades generales del sistema. Su propósito es garantizar el cumplimiento de los requisitos pautados además de asegurar su correcto funcionamiento.

CASO DE PRUEBA DE ACEPTACIÓN

Código de caso de prueba: CP1 **Requisito:** Configurar grados de especificidad

Responsable de la prueba: Javier Fernández Machín

Descripción de la prueba: Prueba para la funcionalidad de configurar grados de especificidad.

Condiciones de ejecución: Usuario autenticado en el sistema.

Entrada/Pasos de ejecución: El usuario selecciona el grado de especificidad deseado por rasgos.

Resultado esperado: Grados de especificidad definidos

Evaluación de la prueba: Prueba satisfactoria

Tabla 10 *Caso de prueba de aceptación Configurar Grados de Especificidad*

CASO DE PRUEBA DE ACEPTACIÓN

Código de caso de prueba: CP2 **Requisito:** Buscar y visualizar patrones

Responsable de la prueba: Javier Fernández Machín

Descripción de la prueba: Prueba para la funcionalidad de buscar y visualizar patrones.

Condiciones de ejecución: Usuario autenticado en el sistema.

Grados de especificidad configurados

Entrada/Pasos de ejecución: El usuario selecciona la opción "Buscar Patrones"

Resultado esperado: Visualización de los patrones obtenidos

Evaluación de la prueba: Prueba satisfactoria

Tabla 11 *Caso de prueba de aceptación Buscar y Visualizar Patrones*

3.8 Conclusiones parciales

Una vez finalizado el presente capítulo se pueden afirmar las siguientes conclusiones:

1) La configuración de los parámetros T, E_{max} y K^* está en función de las características de la vista minable.
2) Los artefactos mediante los cuales se representan los patrones obtenidos apoyan a la compresión de los resultados y por tanto a la toma de decisiones sobre los recursos informáticos de una organización.

CONCLUSIONES GENERALES

A partir del estudio realizado para dar cumplimiento al objetivo general y los objetivos específicos, y sobre la base de los resultados obtenidos, se arriban a las siguientes conclusiones generales:

1) El estudio de las características de la información representa un factor clave para el éxito de un proyecto de MD.
2) La mezcla del método tradicional de agrupamiento *Fuzzy C-Medoids* con la metaheurística de GAs es una alternativa viable para abordar el problema del agrupamiento particional multiobjetivo desde un enfoque de pertenencia difusa.
3) La correcta visualización y representación de los patrones influye directamente en la interpretación de los mismos y por ende en las decisiones futuras.
4) El módulo de MD desarrollado para el sistema GRHS obtiene patrones en términos de grupos que permiten mejorar la toma de decisiones por parte de especialistas y directivos en cuanto a los recursos de hardware y software disponibles en una organización.

RECOMENDACIONES

En aras de enriquecer la presente investigación mediante futuros trabajos, se recomienda:

1) Una mayor experimentación sobre el método propuesto, adoptando diferentes parametrizaciones y juegos de datos.
2) La utilización de otros índices de validación que contrasten y estén acorde con las funciones objetivo que se optimizan para observar el rendimiento y confiabilidad del método de agrupamiento desarrollado.
3) Mejorar la validación del módulo con los potenciales y eventuales usuarios de la aplicación.
4) La implementación de un módulo de MD que gestione la configuración de la vista minable.
5) Añadir al sistema GRHS la funcionalidad de notificar al usuario una vez haya finalizado el proceso de búsqueda de patrones.

REFERENCIAS BIBLIOGRÁFICAS

1. *Administración de inventarios.* Caldentey, Eugenio y Pizarro, Claudio. s.l. : Diapositivas exposición tomadas de Internet, Diapositivas exposición tomadas de Internet.

2. Febles Rodríguez, Juan Pedro y González Pérez, Abel. Scielo. [En línea] 13 de Noviembre de 2001. [Citado el: 15 de Abril de 2015.] http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352002000200003.

3. Hernandez Orallo, Jose, Ferri Ramírez, Cesar y Ramírez Quintana, M. Jose. *Introducción a la Minería de Datos.* Madrid : Editorial Pearson Educación SA, 2004.

4. Sumathi, S. y Sivanandam, S. N. *Introduction to Data Mining and its Applications.* s.l. : Springer, 2006.

5. *The KDD Process for Extracting Useful Knowledge from Volumes of Data.* Fayyad, Usama, Piatetsky-Shapiro, Gregory y Smyth, Padhraic. 11, s.l. : Communications of the ACM, 1996, Communications of the ACM, Vol. 39, págs. 27-34.

6. La gran enciclopedia de economía. *La gran enciclopedia de economía.* [En línea] [Citado el: 12 de Abril de 2015.] http://www.economia48.com/spa/d/inventario/inventario.htm.

7. Real Academia Española. *Real Academia Española.* [En línea] 27 de Enero de 2015. http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=incidencia.

8. *From Data Mining to Knowledge Discovery in Databases.* Fayyad, Usama , Piatetsky-Shapiro, Gregory y Smyth, Padhraic. 3, 1996, AI Magazine, Vol. 17, págs. 37-54.

9. Febles Rodríguez, Juan Pedro. *KDD y MD.* 2005.

10. Moine, Juan Miguel. *Metodologías para el descubrimiento de conocimiento en bases de datos: un estudio comparativo.* s.l. : Facultad de Informática de la Universidad Nacional de La Plata, 2013. pág. 111.

11. Bressán, G. E. *Almacenes de datos y minería de datos.* s.l. : Facultad de Ciencias Exactas y Naturales y Agrimensura, 2003.

12. Gorunescu, Florin. *Data Mining: Concepts, models and techniques.* s.l. : Springer, 2011. Vol. 12.

13. Xu, Rui y C. Wunsch, Donald. *Clustering.* s.l. : IEEE Press, 2009.

14. Das, Swagatam, Abraham, Ajith y Konar, Amit. *Metaheuristic Clustering.* s.l. : Springer, 2009.

15. *An evolutionary approach to multiobjective clustering.* Handl, Julia y Knowles, Joshua. 1, 2007, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, Vol. 11.

16. *A Survey of Evolutionary Algorithms for Clustering.* Hruschka, Eduardo R., y otros. 2, 2009, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, Vol. 39, págs. 133-155.

17. Docsetools. [En línea] [Citado el: 28 de Abril de 2015.] http://docsetools.com/articulos-utiles/article_114882.html.

18. Oyarzún Rodríguez, Jonnatan Washington. *Generación Automática de Algoritmos de Agrupamiento.* s.l. : Universidad de Santiago de Chile, 2013.

19. *Large-Scale Experimental Evaluation of Cluster Representations for Multiobjective Evolutionary Clustering.* García-Piquer, Alvaro, y otros. 2013, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

20. *A Framework for Multi-objective Clustering and its Application to Co-location Mining.* Jiamthapthaksin, Rachsuda, F. Eick, Christoph y Vilalta, Ricardo. 2009, In Advanced Data Mining and Applications, págs. 188-199.

21. *Automatic kernel clustering with a Multi-Elitist Particle Swarm Optimization Algorithm.* Das, Swagatam, Abraham, Ajith y Konar, Amit. 5, s.l. : Elsevier, 2008, Pattern Recognition Letters, Vol. 29, págs. 688-699.

22. *A Multi-Objective Genetic Graph-based Clustering Algorithm with Memory Optimization.* D. Menéndez, Héctor, F. Barrero, David y Camacho, David. Cancun, Mexico : s.n., Junio, 2013, Congress on Evolutionary Computation (CEC).

23. *Survey of Clustering Algorithms.* Xu, Riu y Wunsch II, Donald. 3, Mayo de 2005, Neural Networks, IEEE Transactions, Vol. 16, págs. 645-678.

24. *K-means v/s K-medoids: A Comparative Study.* Singh, Shalini S. y Chauhan, N. C. 2011.

25. *A K-means-like Algorithm for K-medoids Clustering and Its Performance.* Park, Hae-Sang, Lee, Jong-Seok y Jun, Chi-Hyuck. s.l. : Proceedings of ICCIE, 2006, págs. 102-117.

26. *FCM: The fuzzy c-means clustering algorithm.* Bezdek, James C., Ehrlich, Robert y Full, William. 2, s.l. : Computers & Geosciences, 1984, Vol. 10, págs. 191-203.

27. *Hybrid evolutionary multiobjective fuzzy c-medoids clustering of categorical data.* Mukhopadhyay, Anirban, Maulik, Ujjwal y Bandyopadhyay, Sanghamitra. 12, s.l. : 2013 IEEE Workshop on, 2013, Hybrid Intelligent Models and Applications (HIMA), Vol. 7, pág. 6.

28. *A Fuzzy Relative of the k-Medoids Algorithm with Application to Web Document and Snippet Clustering.* Krishnapuram, Raghu, Joshi, Anupam y Yi, Liyu. [ed.] Fuzzy Systems Conference Proceedings. 1999, págs. 1281-1286.

29. *Unsupervised Pixel Classification in Satellite Imagery Using Multiobjective Fuzzy Clustering Combined With SVM Classifier.* Mukhopadhyay, Anirban y Maulik, Ujjwal. 4, Abril de 2009, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, Vol. 47, págs. 1132-1138.

30. *Metaheurísticas multiobjetivo adaptativas.* Machin Navas, Mirialys y Nebro Urbaneja, Antonio J. 1, 2013, Computación y Sistemas, Vol. 17, págs. 53-62.

31. *Búsqueda de entorno variable multiobjetivo para resolver el problema de particionamiento de datos espaciales con características poblacionales.* Bernábe Loranca, María Beatríz y Guillén Galván, Carlos. 3, 2012, Computación y Sistemas, Vol. 16, págs. 335-347.

32. *Multiobjective clustering with metaheuristic optimization technology.* Caballero, Rafael, y otros. Valencia, España : s.n., 2006.

33. *Strength Pareto particle swarm optimization and hybrid EA-PSO for multi-objective optimization.* Elhossini, Ahmed, Areibi, Shawki y Dony, Robert. 1, 2010, Evolutionary Computation, Vol. 18, págs. 127-156.

34. *Future Paths for Integer Programming and Links To Artificial Intelligence.* Glover, Fred. 5, 1986, Computers & operations research, Vol. 13, págs. 533-549.

35. Infante Abreu, Ana Lilian. *Algoritmos de trayectoria multiobjetivo aplicados al problema de asignación de recursos.* La Habana, Cuba : Instituto Superior Politécnico José Antonio Echeverría (CUJAE), 2012.

36. Rosete, Alejandro. *Meta Heurística.* 2010.

37. Talbi, El-Ghazali. *Metaheuristics: from design to implementation.* s.l. : John Wiley & Sons, 2009. Vol. 74.

38. *Genetic algorithm based clustering technique.* Maulik, Ujjwal y Bandyopadhyay, Sanghamitra. s.l. : Elsevier, 2000, Pattern Recognition, Vol. 33, págs. 1455-1465.

39. *A New Multiobjective Simulated Annealing Based Clustering Technique Using Stability and Symmetry.* Saha, Sriparna y Bandyopadhyay, Sanghamitra. s.l. : IEEE, 2008.

40. Golberg, David E. *Genetic algorithms in search, optimization, and machine learning.* s.l. : Addion wesley, 1989. Vol. 242.

41. Sivanandam, S.N. y Deepa, S.N. *Introduction to genetic algorithms.* s.l. : Springer Science & Business Media, 2007.

42. *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II.* Deb, Kalyanmoy, y otros. 2000, Lecture notes in computer science, Vol. 1917, págs. 849-858.

43. *Building general hyper-heuristics for multi-objective cutting stock problems.* Gómez, Juan Carlos y Terashima-Marín, Hugo. 3, 2012, Computación y Sistemas, Vol. 16, págs. 321-334.

44. *Aplicación de NSGA-II y SPEA-II para la optimización multiobjetivo de redes multicast.* Alvarado, Carolina , y otros. 17, 2005, Ingeniería y desarrollo: revista de la División de Ingeniería de la Universidad del Norte, págs. 28-53.

45. *A new multi-objective technique for differential fuzzy clustering.* Saha, Indrajit, Maulik, Ujjwal y Plewczynski, Dariusz. 2, 2011, Applied Soft Computing, Vol. 11, págs. 2765-2776.

46. *Multiobjective optimization using nondominated sorting in genetic algorithms.* Srinivas, Nidamarthi y Deb, Kalyanmoy. 3, 1994, Evolutionary computation, Vol. 2, págs. 221-248.

47. *Multi-Objective Structural Optimization Design of Horizontal-Axis Wind Turbine Blades Using the Non-Dominated Sorting Genetic Algorithm II and Finite Element Method.* Zhu, Jie, y otros. 2, 24 de Febrero de 2014, Energies, Vol. 7, págs. 988-1002.

48. Cagnina, Leticia Cecilia. *Optimización Mono y Multiobjetivo a través de una Heurıstica de Inteligencia Colectiva.* 2010.

49. *A survey on nature inspired metaheuristic algorithms for partitional clustering.* Jagannath Nanda, Satyasai y Panda, Ganapati. s.l. : Elsevier, 2014, Swarm and Evolutionary Computation, Vol. 16, págs. 1-18.

50. *Clustering by Multi Objective Genetic Algorithm.* Dutta, Dipankar, Dutta, Paramartha y Sil, Jaya. 2012, Recent Advances in Information Technology.

51. *Simultaneous Continuous Feature Selection and K Clustering by Multi Objective Genetic Algorithm.* Dutta, Dipankar, Dutta, Paramartha y Sil, Jaya. 2013, IEEE International Advance Computing Conference (IACC), págs. 937-942.

52. *Multiobjective genetic clustering for pixel classification in remote sensing imagery.* Bandyopadhyay, Sanghamitra, Maulik, Ujjwal y Mukhopadhyay, Anirban. 5, Mayo de 2007, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, Vol. 45, págs. 1506-1511.

53. *Multi-Objective Data Clustering using Variable-Length Real Jumping Genes Genetic Algorithm and Local Search Method.* Nawaz Ripon, Kazi Shah, Tsang, Chi-Ho y Kwong, Sam. s.l. : IEEE, Julio de 2006, International Joint Conference on Neural Networks, págs. 3609-3616.

54. *Evolutionary Multi-Objective Clustering for Overlapping Clusters Detection.* Nawaz Ripon, Kazi Shah y Siddique, M. N. H. s.l. : IEEE, 2009, págs. 976-982.

55. *A multiobjective approach to MR brain image segmentation.* Mukhopadhyay, Anirban y Maulik, Ujjwal. 1, 2011, Applied Soft Computing, Vol. 11, págs. 872-880.

56. *Clustering Data Set with Categorical Feature Using Multi Objective Genetic Algorithm.* Dutta, Dipankar, Dutta, Paramartha y Sil, Jaya. s.l. : IEEE, 2012, International Conference on Data Science & Engineering (ICDSE), págs. 103-108.

57. *Simultaneous Feature Selection and Clustering for Categorical Features Using Multi Objective Genetic Algorithm.* Dutta, Dipankar, Dutta, Paramartha y Sil, Jaya. 2012, International Conference onHybrid Intelligent Systems (HIS), págs. 191-196.

58. *Multiobjective Approach to Categorical Data Clustering.* Mukhopadhyay, Anirban y Maulik, Ujjwal. 2007, IEEE Congresson Evolutionary Computation (CEC 2007), Vols. 1296-1303.

59. *Multiobjective Genetic Algorithm-Based Fuzzy Clustering of Categorical Attributes.* Mukhopadhyay, Anirban, Maulik, Ujjwal y Bandyopadhyay, Sanghamitra. 5, Octubre de 2009, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, Vol. 13, págs. 991-1005.

60. *Data Clustering with Mixed Features by Multi Objective Genetic Algorithm.* Dutta, Dipankar, Dutta, Paramartha y Sil, Jaya. 2012, International Conference onHybrid Intelligent Systems (HIS), págs. 336-341.

61. Definición ABC. [En línea] 12 de Febrero de 2015. http://www.definicionabc.com/ciencia/metodologia.php.

62. Chapman, Pete, y otros. *CRISP-DM 1.0 Step-by-step data mining guide.* 2000.

63. *Estudio comparativo de metodologías para Minería de Datos.* Moine , Juan Miguel, Haedo, Ana Silvia y Gordillo, Silvia. 2011, XIII Workshop de Investigadores en Ciencias de la Computación.

64. *CRITIKAL: client-server rule induction technology for industrial knowledge acquisition from large databases.* Al-Attar, A. Abril, 1997, Research Issues in Data Engineering, págs. 70-72.

65. Gallardo Arencibia, José Alberto. *Metodología para el Desarrollo de Proyectos en Minería de Datos CRISP-DM.* 2010.

66. *Caracterización del Proceso de Obtención de Conocimiento y Algunas Metodologías para Crear Proyectos de Minería de Datos.* Giraldo Mejia, Juan Camilo y Jiménez Builes, Jovani Alberto . 2, 2013, Revista Latinoamericana de Ingeniería, Vol. 1, págs. 42-44.

67. *CRISP-DM: Towards a standard process model for data mining.* Wirth, Rüdiger y Hipp, Jochen. Abril, 2000, Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, págs. 29-39.

68. INTECO, Laboratorio Nacional de Calidad del Software de. *Ingeniería del software: Metodologías y ciclos de vida.* Marzo, 2009.

69. Kroll, Per y Kruchten, Philippe. *The rational unified process made easy: a practitioner's guide to the RUP.* s.l. : Addison-Wesley Professional, 2003.

70. Martínez, Alejandro y Martínez, Raúl. *Guía a Rational Unified Process.* s.l. : Escuela Politécnica Superior de Albacete-Universidad de Castilla la Mancha, 2002.

71. *jMetal: A Java framework for multi-objective optimization.* Durillo, Juan J. y Nebro, Antonio J. 10, 2011, Advances in Engineering Software, Vol. 42, págs. 760-771.

72. *The ECJ Owner's Manual.* Luke, Sean. 2010, Department of Computer Science, George Mason University.

73. Nebro, Antonio J. y Durillo, Juan J. *jMetal 4.5 User Manual.* 2014.

74. van Rossum, Guido. *El tutorial de Python.* [ed.] Fred L. Drake. s.l. : Python Software Foundation, 2014. pág. 105.

75. Mozilla Developer Network. [En línea] [Citado el: 12 de Abril de 2015.] https://developer.mozilla.org/es/docs/Web/JavaScript.

76. Pérez Valdés, Damián. Maestros del Web. [En línea] 3 de Julio de 2007. [Citado el: 12 de Abril de 2015.] http://www.maestrosdelweb.com/que-es-javascript/.

77. González, Enrique. Aprender a Programar. [En línea] [Citado el: 12 de Abril de 2015.] http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=435:ique-es-y-para-que-sirve-html-el-lenguaje-mas-importante-para-crear-paginas-webs-html-tags-cu00704b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192.

78. json.org. [En línea] [Citado el: 15 de Abril de 2015.] http://json.org/json-es.html.

79. [En línea] 10 de Junio de 2014. [Citado el: 16 de Marzo de 2015.] http://jorgeespinozapresenta.blogspot.com/2014/06/pycharm-el-ide-de-python-lanza-su.html.

80. Cristalab. [En línea] 16 de Septiembre de 2014. [Citado el: 16 de Marzo de 2015.] http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c114084l/.

81. Django en Español. [En línea] [Citado el: 16 de Marzo de 2015.] http://django.es/.

82. ComputerHoy.com. [En línea] [Citado el: 16 de Marzo de 2015.] http://computerhoy.com/noticias/internet/descubre-que-es-django-framework-web-moda-8641.

83. ZOEDEV. [En línea] [Citado el: 12 de Abril de 2015.] http://www.zoedev.com/javascript/la-mejor-biblioteca-de-graficos-con-html5-y-javascript-highcharts-js/.

84. Sommerville, Ian. *Software Engineering.* Octava. 2007.

85. Jherson, Dickey. [En línea] 11 de Febrero de 2015. http://herramientascasecomparaciones.blogspot.com/.

86. Ruiz Shulcloper, Jose, Guzmán Arenas, Adolfo y Martínez Trinidad, Jose Franciso. *Enfoque Lógico Combinatorio al Reconocimiento de Patrones I. Selección de variables y clasificación supervisada.* s.l. : Instituto Politécnico Nacional, 1999. pág. 147.

87. Oberg, Roger, Probasco, Leslee y Ericsson, Maria. *Applying Requirements Management with Use Cases.* 2000.

88. Pressman, Roger S. *Ingeniería del Software: Un enfoque práctico.* Quinta. 1997. pág. 601.

89. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* [ed.] Andrés Otero. Madrid : Addison Wesley, 2000. pág. 464.

90. [En línea] 23 de Abril de 2013. [Citado el: 23 de Marzo de 2015.] http://umldiagramadespliegue.blogspot.com/.

91. Ankerst, Mihael. *Visual Data Mining.* s.l. : Dissertation, 2001. ISBN 3-89825-201-9.

92. Slideshare. [En línea] [Citado el: 27 de Mayo de 2015.] http://es.slideshare.net/pablis001/estrategias-de-aplicaciones-para-las-pruebas-de-integracin.