VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MINERÍA DE DATOS DISTRIBUIDA BASADA EN SISTEMAS MULTI-AGENTES

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
JONATHAN LUIS CÓRDOBA LUNA

TUTOR PRINCIPAL
DRA. MARÍA DEL PILAR ÁNGELES
FACULTAD DE INGENIERÍA

MÉXICO, D. F. AGOSTO DE 2013

Índice general

Capítulo 1

Introducción

Hoy en día la minería de datos se ha vuelto una de las principales herramientas para poder extraer conocimiento, gran parte de su éxito se debe a que esta tecnología incorpora métodos y técnicas que permiten generar tal conocimiento de forma eficiente; sin embargo, con el gran avance tecnológico que existe actualmente, la capacidad de los dispositivos de cómputo ha aumentado, lo que ha permitido a muchas organizaciones poder generar una inmensa cantidad de datos, pero este incremento en la cantidad de datos ha exigido que el mismo proceso de minería de datos tenga que incorporar nuevas estratégias para poder atacar de forma efectiva la escalada de datos generados.

La forma tradicional de procesar datos y almacenarlos en la minería de datos ha demostrado ser ineficiente para poder accesar tal cantidad de datos y poder administrarlos de manera efectiva. En la actualidad muchos de estos datos se encuentran almacenados en diferentes sitios geográficos, lo que exige en primera instancia un impresionante costo de comunicación lo cual puede llegar a no ser factible en muchos sistemas que tienen limitación de recursos. Otro aspecto a destacar es que la minería de datos es un proceso que requiere un intenso poder de cómputo para poder analizar tal volúmen de datos, por lo que el rendimiento de un sistema dedicado a la minería de datos puede verse afectado considerablemente. Además si se considera que la entrada de datos a estos sistemas cambia constantemente esto puede generar que el sistema se vuelva lento.

Para poder enfrentar estos problemas ha surgido una nueva disciplina que pone atención a todos ellos, la **Minería de Datos Distribuida** (DDM). La cual está enfocada principalmente a la distribución de recursos sobre la red, así como a los procesos de minería de datos. La tarea principal de la DDM entonces es el poder hacer minado de datos sobre fuentes de datos que se encuentran distribuidas en diversos sitios de la red, así como la de ofrecer soluciones algorítmicas que permitan llevar a cabo este minado de forma efectiva, poniendo atención a las restricciones de recursos existentes.

Actualmente la investigación en DDM ha sido dirigida principalmente a la búsqueda de alternativas que permitan mejorar el rendimiento de los procesos de minería de datos. Una de las alternativas es la de incorporar **Sistemas Multi-Agentes** (MAS) **[Wooldridge, 2002]** a los sistemas DDM. Los agentes por sus características distribuidas permiten mejorar los sistemas DDM en aspectos de interoperabilidad, configuración dinámica y rendimiento.

El presente trabajo de tesis se enfoca en la integración de sistemas DDM y MAS buscando aprovechar las propiedades de éste último para mejorar el rendimiento del sistema DDM **[Chaimontree, Atkinson, and Coenen, 2010]**. Para ello se presenta un marco de trabajo basado en la incorporación de agentes al sistema DDM. La misión de estos será poder trabajar de forma coordinada y colaborativa para poder llevar a cabo tareas de minería de datos de tal manera que se pueda obtener un mejor rendimiento en el sistema DDM.

1.1. Antecedentes

1.1.1. Contexto de la minería de datos distribuida

El proceso de minería de datos generalmente adopta un enfoque centralizado en el que aquellos datos que existen en diferentes sitios geográficos son recolectados en un solo sitio central (comúnmente un data warehouse), en él, estos son remodelados para presentar un esquema único y son preparados para que posteriormente algoritmos de minería de datos puedan extraer conocimiento útil que permita a los usuarios finales tomar decisiones acerca del negocio.

Para extraer conocimiento de los datos en el enfoque centralizado se construyen modelos (con base en algoritmos de minería de datos) de los datos recolectados, de tal forma que según la tarea a realizar, estos modelos ayuden a clasificar o predecir otras fuentes de datos que contengan información desconocida. En términos de minería de datos, se dice que los modelos aprenden de datos históricos para predecir o clasificar datos futuros.

Pero esta forma de minar los datos aunque puede generar resultados precisos presenta muchos inconvenientes, entre estos se encuentran los siguientes **[Rao, 2010]**:

- La cantidad de datos generados en cada sitio puede ser impresionante. Este volúmen de datos puede exceder el límite del terabyte o petabyte, por lo que transferir datos a un sitio central puede representar un costo de comunicación bastante alto sobre todo en sitios que tienen un ancho de banda limitado.

- La minería de datos es un proceso que involucra mucho poder de cómputo por lo que el rendimiento en manejo de memoria y en procesamiento del sistema pueden verse afectado considerablemente.
- Los datos que entran al sistema pueden cambiar rápidamente, por ejemplo, en muchas aplicaciones los datos a ser minados pueden ser producidos en una tasa muy alta o presentarse en flujos, lo cual puede generar fallas en el sistema y afectar la escalabilidad del mismo. Complementario a esto la capacidad de almacenamiento puede ser insuficiente para tal cantidad de información.

En el enfoque centralizado los resultados pueden ser muy buenos en el sentido de que los modelos construidos generan predicciones o clasificaciones más precisas dado que se genera un modelo global único de todos los datos recolectados. Pero el hecho de que los datos estén localizados en sitios geográficamente distribuidos ha hecho necesario tener que buscar otra estrategia que permita minar datos en tal entorno distribuido. En la práctica transferir datos a un sitio central ha llegado a no ser viable. Considerando además los inconvenientes mencionados previamente, se requiere adoptar una forma de minar los datos. En respuesta a lo anterior se creo una nueva área de conocimiento que pone atención a las restricciones de recursos existentes en sistemas de minería de datos y que a la vez permite poder hacer el análisis y minado de datos sobre fuentes de datos distribuidas. Esta área de conocimiento se le conoce como Minería de Datos Distribuida.

La Minería de Datos Distribuida (DDM) se ha vuelto importante en casos donde los datos son inherentemente distribuidos y no pueden ser centralizados en alguna máquina, ya sea por razones de seguridad, tolerancia a fallas, etc. Además muchos entornos de minería de datos también consisten de usuarios, diversos componentes de hardware y software de minería de datos que se encuentran distribuidos por lo que la DDM ofrece una excelente alternativa para administrar todos estos recursos.

En general, los factores que han contribuido al surgimiento de la minería de datos distribuida son los siguientes **[Paul, 2011]**:

- La necesidad de minar subconjuntos de datos que se encuentran distribuidos, cuya integración no es trivial y es bastante cara.
- Muchas aplicaciones para minería de datos generan cuellos de botella que afectan su rendimiento y su escalabilidad.
- La DDM provee un marco de trabajo para la escalabilidad, ya que permite dividir un gran conjunto de datos con alta dimensionalidad en subconjuntos más pequeños los cuales requieren recursos computacionales individuales.

En un sistema de DDM los datos en los diversos sitios son particiones de alguna tabla global. Tal particionamiento puede ser horizontal o vertical según se haya

diseñado la base de datos. En el caso del particionamiento horizontal las tablas en cada sitio son subconjuntos de una tabla global; los cuales tienen los mismos atributos, por otra parte las tablas particionadas verticalmente, en cada sitio representa una colección de columnas (los sitios no tienen los mismos atributos). Sin embargo, en cada sitio se asume que las tablas contenen un identificador único que les permite poder reunirse nuevamente.

Un factor importante en la DDM radica en cómo se realizan las tareas de minería de datos. Para ello generalmente se emplea la estratégia de meta-aprendizaje, la cual consiste de los siguientes pasos: para los datos en cada sitio un modelo de datos es generado, este modelo es conocido como modelo base, después todos estos modelo son recolectados en algún sitio y finalmente se produce un modelo global generado a partir de los modelos base. Para poder generar tal modelo se emplea alguna de las siguientes estrategias **[Prodromidis, Chan, and Stolfo, 2000]**:

- **Voteo**. En este caso, cada modelo base consigue un voto, y el que tenga la mayoría de votos representa el modelo a usar para agrupar o clasificar datos.
- **Arbitraje**. En este caso un *árbitro*, junto con una regla de arbitraje, decide el modelo a emplear basado en aquel modelo que presentó mejor rendimiento.
- **Combinación**. En este caso se busca generar un sólo modelo de todos los modelos generados, generalmente por medio de correlacionar los resultados generados por cada modelo base.

Otro factor importante dentro de la DDM es la sincronización entre los diversos sitios que componen el sistema. Ésta generalmente se constituye de muchos sitios conectados en red que intercambian mensajes. Para que los sitios puedan lograr sus objetivos es indispensable que los datos sean descompuestos de manera que el paso de mensajes entre los sitios se minimice, de lo contrario se podría generar un bajo nivel de rendimiento. Si la cantidad de mensajes transferidos de un sitio a otro es considerablemente grande, algunos sitios podrían ver reducida su capacidad de memoria afectando su rendimiento. Además si la capacidad de ancho de banda es limitada se podría generar un cuello de botella entre los sitios que conforman el sistema.

En general para que un sistema DDM pueda ser aplicable es necesario que un escenario distribuido cumpla con las siguientes características **[Silva, Giannella, Bhargava, Kargupta, and Klush, 2005]**:

- Los sistemas consisten de múltiples sitios independientes de datos y el proceso de comunicación sólo puede llevarse a través de mensajes.
- La comunicación entre sitios genera un costo de comunicación que debe ser

considerado.

- Se deben tener en cuenta restricciones de recursos como: ancho de banda, capacidad de almacenamiento, cantidad de memoria, etc.
- Establecer esquemas de seguridad para mantener la privacidad en los diferentes sitios.

1.1.2. Agentes inteligentes en minería de datos distribuida

Una alternativa para mejorar la comunicación y el rendimiento en un sistema DDM es el incorporar Sistemas Multi-Agentes (MAS). Un MAS es esencialmente una colección de entidades de software (agentes) que cooperan para procesar alguna tarea. Un aspecto importante de ésta cooperación es que los agentes se comportan de manera autónoma, esto quiere decir que ellos negocian con otros agentes para completar una tarea determinada sin que algún proceso principal se los indique **[Albashiri, Coenen, and Leng, 2010]**.

En un MAS los agentes incluyen cierta cantidad de *inteligencia*[1], la cual es la habilidad que tienen éstos para elegir varios cursos de acción, planes, formas de comunicación, formas de adaptarse a los cambios del medio, y la capacidad de aprender de la experiencia. Para ejemplificar esto, un agente inteligente tiene un objetivo abstracto, tiene una forma de evaluar si la información que llega a él es interesante (esto se hace a partir de un *conjunto de lógicas* que van desde programas codificados a inferencias basadas en reglas), además cuenta con un elemento *sensor* que puede recibir eventos, un *reconocedor* ó *clasificador* que determina que eventos ocurrirán, y un *mecanismo* para tomar acciones **[Wooldridge, 2002]**.

Integrar MAS a tareas de minería de datos ofrece la posibilidad de que un grupo de agentes flexibles de minería de datos cooperen en el descubrimiento de conocimiento de fuentes distribuidas. Ellos serían responsables de accesar a los datos y extraer información que ellos han recolectado.

En la actualidad la tecnología de agentes de software ha madurado lo suficiente para poder producir agentes inteligentes que puedan ser usados para controlar un gran número de tareas concurrentes de ingeniería. Los MAS son comunidades de agentes que pueden intercambiar información y datos en forma de mensajes. El comportamiento e inteligencia de los agentes de la comunidad pueden ser obtenidos llevando a cabo minería de datos sobre aplicaciones de datos disponibles y considerando el dominio de conocimientos.

En un MAS cada agente presenta las siguientes características **[Bellifemine, Caire, and Greenwood, 2007]**:

[1] Un agente inteligente ó racional trata de maximixar el valor de una medida de rendimiento, dada la secuencia de percepciones que ha observado hasta el momento.

- **Autonomía**. Los agentes pueden operar sin la intervención de humanos u otros agentes y tienen algún tipo de control sobre sus acciones y estado interno.

- **Habilidad para socializar**. Los agentes pueden interactuar con otros agentes por medio de un *lenguaje de comunicación de agentes.*

- **Reactividad**. Los agentes tienen la capacidad de percibir su entorno y responder oportunamente a los cambios que ocurren en él.

- **Proactividad**. Los agentes no simplemente actúan respondiendo a su entorno, ellos pueden exhibir un comportamiento dirigido a cumplir con sus objetivos tomando iniciativa propia.

Un punto esencial para incorporar agentes en un sistema de DDM es que un MAS presenta diversos mecanismos para que los agentes se puedan comunicar de manera que esta comunicación sea mínima y sin imprecisiones entre ellos. Para ello incorporan un lenguaje de comunicación estandar (ACL), en el cual están establecidos toda una serie de protocolos para intercambiar información entre los agentes.

Los principales problemas en DDM no están enfocados a las técnicas de minería, sino en el desarrollo de una infraestructura y protocolos (algoritmos distribuidos) que soporten operaciones colaborativas de componentes de software distribuido (agentes) responsables de la DDM. Así la DDM es un sistema complejo que se enfoca en la distribución de recursos sobre la red y los procesos de minería de datos en sitios descentralizados. En cada sitio, una estrategia de minería de datos debe ser desplegada específicamente para cierto dominio de datos. Sin embargo, pueden existir otras estrategias que la minería de datos podría probar. En cada sitio de datos deberían integrarse sin problemas métodos externos y llevar a cabo pruebas sobre múltiples estratégias. Los agentes en este caso pueden ser tratados como una unidad computacional que lleve a cabo múltiples tareas basadas en una configuración dinámica. Los agentes interpretarían esa configuración y generarían un plan de ejecución para completar múltiples tareas. Los Agentes en MAS al ser proactivos y autónomos pueden percibir su entorno, dinámicamente razonar acciones basadas en condiciones, e interactuar con otros agentes. Una ventaja de desplegar agentes en sistemas DDM es que al ser los MAS de naturaleza descentralizada, cada agente tiene sólo una limitada visión del sistema. Esta limitación permite de alguna manera mejorar la seguridad de los sistemas DDM ya que los agentes no necesitan observar otros entornos irrelevantes. Los agentes pueden ser programados de manera compacta de modo que pueden ser transmitidos a través de la red en vez de transmitir datos los cuales pueden ser más voluminosos. El poder transmitir agentes de un host a otro permite dinámicamente organizar el sistema. Por ejemplo, el agente *a1*, localizado en el sitio *s1*, posee un algoritmo *alg1*. Una tarea de minería de datos *t1* en un sitio *s2* es instruida para minar los

datos usando *alg1*. En este contexto, transmitir *a1* a *s2* es el camino más probable en vez de transferir todos los datos de *s2* a *s1* donde el *alg1* está disponible **[Rao, 2010]**.

Para complementar el uso de agentes, ellos ofrecen soluciones alternativas ya que ellos pueden viajar através de la red permitiendo a los sistemas mantener privacidad en los datos, aparte gracias a sus caracteristicas de auto-organización, estos eximen al sistema de transferir datos através de la red de este modo añaden seguridad, a los datos.

1.1.3. Clustering distribuido

El clustering ó agrupamiento en minería de datos consiste de particionar registros de bases de datos en subconjuntos ó clusters, los elementos que pertenecen a un cluster comparten un conjunto de propiedades en común que los distinguen de otros clusters.

A diferencia de los métodos de clasificación que tienen etiquetas ya predefinidas, el agrupamiento viene automáticamente con las etiquetas. Por ésta razón el clustering es también llamado inducción no supervisada.

Para el método de clustering los algoritmos más populares son los algoritmos de K-medias y K-vecinos más cercanos (KNN). En el algoritmo de K-Medias la idea es aleatoriamente tomar K puntos de datos como el centro de los clusters. Después, cada registro es asignado a un cluster si es cercano a él en términos de un error cuadrático ó de su distancia Euclideana **[Han and Kamber, 2006]**. Finalmente un nuevo centro es calculado tomando la media de todos los puntos en un cluster, ajustando su estado para la siguiente iteración.

En el algoritmo de K-vecinos más cercanos una distancia es asignada entre todos los puntos en un conjunto de datos. Ésta distancia es definida como la distancia Euclidiana entre dos puntos. De estas distancias, una matriz de distancia es construida entre todos los posibles pares de puntos (x, y). Cada uno de los puntos dentro del conjunto de datos tiene una clase etiqueta en el conjunto, $C = c_1, \ldots, c_2$. Los puntos de datos, en KNN (k es el número de vecinos) son entonces encontrados al analizar la matriz de distancia y una vez encontrados entonces son analizados para determinar cuáles clases de etiquetas son las más comunes entre los conjuntos. La clase de etiqueta más común es entonces asignada a los puntos de datos para ser analizados. Las etiquetas resultantes son entonces usadas para clasificar cada punto de datos en el conjunto de datos **[Han and Kamber, 2006]**.

Para poder llevar éstos algoritmos a un ambiente distribuido generalmente se adopta la estrategia de *meta-aprendizaje* en donde los modelos de clustering lo-

cales son generados en cada sitio y transmitidos (asíncronamente) a un sitio para generar un modelo de clustering global de ellos. Éste método tiene la ventaja que puede requerir sólo una pasada de mensajes, con un ligero grado de sincronización **[Prodromidis, Chan, and Stolfo, 2000]**.

A nivel de les agentes inteligentes pueden ser generados un número determinado de agentes para construir los modelos locales y transmitir los resultados a otros sitios donde los soliciten.

Para poder seleccionar de manera óptima un modelo de clustering y evaluar los resultados de los algoritmos, se tienen dos criterios **[Kovács, Legány, and Babos, 2006]**:

- **Compacidad**: Los miembros de cada cluster deberían ser tan cercanos a otros tanto como sea posible. Una medida común para medir la compacidad es la *varianza*. Si el valor de varianza del conjunto de datos es pequeño indica un alto grado de homogeneidad de los datos del conjunto de datos, en términos de la medida de distancia.
- **Separación**: Los clusters mismos deberían ser ampliamente separados. Hay tres enfoques para medir la distancia entre dos clusters diferentes: la distancia entre los miembros más cercanos de los clusters, la distancia entre los más distantes miembros y la distancia entre los centros de los clusters.

1.2. Planteamiento del problema

1.2.1. Situación

Un sistema de DDM tiene como función principal procesar datos que se encuentran distribudos geográficamente en distintos sitios. Cada sitio procesa un modelo local de sus datos y lo envía a otro sitio para que éste genere un modelo global a partir de los modelos locales. Una vez generado el modelo global este es retransmitido a todos los otros sitios para que cada sitio lo integre a sus modelos locales. Sin embargo, el llevar a cabo esta función puede requerir costos de comunicación significativos y necesitar una cantidad muy importante de recursos computacionales, por lo que el rendimiento del sistema de DDM puede verse afectado.

1.2.2. Formulación

¿Como aprovechar las características de un MAS de modo que el rendimiento de un sistema de DDM puede ser mejorado considerablemente?.

1.3. Justificación

El campo de la DDM trata de enfrentar los desafíos que existen en sitios que regularmente presentan limitación de recursos, aparte de ofrecer soluciones algorítmicas para analizar datos de forma distribuida sin la necesidad de tener que transferir estos datos a algún sitio central. Por otro lado, los MAS tratan de resolver problemas que requieren llevar a cabo cómputo a nivel distribuido. En particular un MAS proporciona una marco de trabajo el cual es inherentemente concurrente, ya que este, esta compuesto de un conjunto de entidades llamadas agentes, los cuales pueden trabajar en paralelo, así como cooperar para lograr sus objetivos. En la actualidad muchos sistemas tienen la necesidad de incorporar cierto grado de inteligencia a sus procesos por lo que los MAS ofrecen esta alternativa dado que incorporan muchos conceptos de inteligencia artificial. En un sistema MAS los agentes tienen un rol bien definido. Por medio de éste rol se definen las responsabilidades que el agente tendrá en términos de sus objetivos. La cooperación en un MAS se logra incorporando tres componentes esenciales: la colaboración, la cual está encargada de asignación de tareas; la coordinación, la cual incluye sincronización y planeación; y la resolución de conflictos la cual es necesaria cuando recursos concurrentes acceden al mismo objetivo.

El aprovechar las características de un MAS permite mejorar un sistema DDM en cuestiones de:

- **Interoperabilidad**. En cada sitio pueden desplegarse agentes que interactuan entre sí, con nuevos agentes que ingresan al sistema, así como con otros sitios de datos. Esta interacción incluye protocolos de comunicación, políticas de integración, y un directorio de servicios.
- **Configuración Dinámica**. Una tarea de minería de datos puede involucrar varios agentes y fuentes de datos. Estos agentes pueden ser configurados equipandolos con un algoritmo de minería de datos, y para tratar diferentes tipos de datos.
- **Rendimiento**. En entornos distribuidos, tareas pueden ser ejecutadas en paralelo, intercambiando datos, y a nivel de concurrencia.

1.4. Hipótesis

Los sistemas multi-agentes mejoran el rendimiento de un sistema de minería de datos distribuida ofreciendo la incorporación de agentes inteligentes que son capaces de procesar, accesar y distribuir datos de manera eficiente, generando mínimo intercambio de mensajes entre sitios distribuidos y maximizando los resultados de minado de datos.

1.5. Objetivos

1.5.1. Objetivo General

Demostrar que el rendimiento de un sistema de Minería de Datos Distribuida se mejora utilizando Sistemas Multi-agentes.

1.5.2. Objetivos Particulares

- Mejorar los procesos de análisis y minado de datos.
- Reducir el intercambio de mensajes sobre los sitios que componen el sistema DDM.
- Mejorar el rendimiento con respecto a memoria y CPU en sitios que contienen recursos limitados.
- Mostrar que el marco de trabajo generado puede ser usado para evaluar diversos escenarios.

1.6. Metodología

El presente proyecto de tesis propone construir un marco de trabajo que incorpore MAS a un sistema DDM. Dicho marco estará compuesto por los siguientes agentes:

- **Agente Usuario**. Éste agente es responsable de la interacción entre los usuarios finales y el agente coordinador con el fin de cumplir las tareas asignadas.
- **Agente Coordinador**. Éste agente es responsable de la correcta transmisión de mensajes entre los agentes dentro de la red. Toma los requerimientos del usuario y los envía a los correspondientes agentes.
- **Agente Coordinador de Algoritmos**. Éste agente es responsable de la interacción entre los agentes de clustering. Este recibe la información procesada de los agentes de clustering y ejecuta el algoritmo globalmente con el fin de garantizar una mejor calidad de clustering.
- **Agente de Clustering**. Es el encargado de llevar a cabo una algoritmo de clustering. Una vez que los agentes de clustering han hecho su tarea, estos envían la información local procesada a el agente coordinador de algoritmo.

- **Agente de Datos**. Éste está encargado de una fuente de datos. Este interactua y permite el acceso a los datos. Hay un agente de datos por fuente de datos.

- **Agente de Validación**. El agente de validación es responsable de evaluar la calidad de los resultados del clustering. Hay una agente de validación por una técnica de medida de una configuración de cluster dada. Para evaluar la calidad, estos agente toman en consideración la cohesión y separación de un cluster, para el caso de k-medias. En el caso del clustering jerárquico, toma como medida la distancia cofenética para medir la proximidad del algoritmo de clustering aglomerativo. Esta distancia ayuda a determinar la precisión del clustering jeráquico. Para calcularla se tiene que obtener una matriz de similaridad y la matriz cofenética. La distancia cofenética puede ser vista como una correlación entre la matriz de distancia y la matriz cofenética. Si el valor calculado es cercano al 100 %, la calidad del clustering es buena.

- **Agente de Rendimiento**. Éste agente toma medidas de recursos del sistema operativo con el fin de obtener el rendimiento total de los algoritmos procesados en términos de la transmisión de datos, acceso a datos y procesos de dados como sigue:

  - ⋆ Memoria Usada: Es la cantidad de memoria física consumida por el algoritmo cuando éste ha sido ejecutado. El valor del resultado es expresado en megabytes (MB).

  - ⋆ Tiempo de procesamiento: Es la cantidad de tiempo que le tomó al algoritmo ser procesado. El valor resultante es expresado en nanosegundos.

  - ⋆ Cantidad de datos transmitidos: Es el tamaño total de todos los datos procesados y transferidos. La cantidad es expresada en MB.

  - ⋆ Ancho de Banda PC-LAN: Es la cantidad de información que puede ser enviada sobre una conexión de red en un periodo de tiempo dado. El ancho de banda es usualmente medido en bits por segundo (bps), kilobits (kbps) o megabits por segundo (mps).

  - ⋆ Tiempo de respuesta: Es el intervalo de tiempo desde que el usuario hizo la solicitud hasta que se le presentó el resultado.

  - ⋆ Tiempo de transmisión: Tiempo en que los datos son transferidos de un sitio a otro.

  - ⋆ Tiempo de respuesta total: La respuesta total es la suma del tiempo de procesamiento, el tiempo de transmisión y el tiempo de respuesta.

  - ⋆ Lecturas Físicas: Número total de bloques leídos de disco.

⋆ Lecturas Lógicas: Número total de bloques de datos leídos de la memoria principal (RAM/cache).

Todas estas medidas son almacenadas en una tabla como una *bitacorá* o *log* al cual el agente de datos puede accesar e informar al agente de rendimiento. De este modo, cuando un usuario envie una solicitud, ésta será evaluada de acuerdo a la información histórica almacenada en el *log*, y una estratégia de ejecución será desarrollada. Si la cantidad de datos procesados es pequeña, el agente de rendimiento establecerá un estatus "bajo", de esta forma la creación de un sólo agente de clustering será llevada a cabo para efectuar el análisis de clustering. Si la cantidad considerada es alta, el agente de rendimiento establecerá el estatus "mediano", con el fin de que se creen dos agentes para procesar los datos y obtener el análisis de clustering. Si la cantidad de datos es muy grande, el agente de rendimiento establecerá un estatus "alto', con el fin de crear tres agentes para el análisis de clustering. Este estatus es enviado al agente coordinador, el cual es el responsable de construir los agentes solicitados. Con el fin de mejorar los resultados del clustering y el rendimiento del sistema de DDM se ha implementado negociación entre los agentes por medio de un protocolo de comunicación. Par instancia, considerando la cantidad de datos que se tienen que analizar, hay una negociación de cual método de clustering es mejor preguntando a cada agente de clustering si pueden llevar a cabo el análisis de cluster de acuerdo a los recursos del sistio donde los agentes residen.

1.7. Implementación

En este trabajo de tesis se propone la construcción de una plataforma web que incorpore un MAS. Para la implementación de los agentes descritos previamente se utilizará la plataforma de desarrollo JADE **[Bellifemine, Caire, and Greenwood, 2007]**, la cuál incorpora un Lenguaje de Comunicación de Agentes (ACL) propio; pero éste puede ser modificado de acuerdo a los requerimientos del sistema, además incluye un Sistema Administrador de Agentes (AMS) el cual es responsable del estatus de los agentes y el Facilitador de Directorios (DF) el cuál consiste de un servicio de "páginas amarillas" para mantener un registro de las capacidades de los agentes.

En general, la ventaja que ofrece JADE es que al ser programada en Java permite su integración. Para el caso de JADE se integra la bibilioteca `jade.gateway` para poder programar agentes en una interface web.

En cuanto al diseño de la aplicación web, ésta proporciona una arquitectura la cual provee los siguientes elementos:

- **Interface Web**. Ésta interface ofrece el medio por el cual los usuarios pueden ejecutar sus solicitudes para las tareas de mineria de datos.
- **Repositorios de datos**. Éstos consisten de carpetas ó sistemas de bases de datos que puedan almacenar datos en un sistema de DDM. En la parte de base de datos se incorporará un SMBD: PostgreSQL.
- **Repositorio de Clustering**. En este bloque se encuentran todos los algoritmos que son implementados para tareas de clustering. Y los algoritmos para validar los resultados de los algoritmos de clustering.
- **Motor del sistema**. Éste administra los agentes diseñados, llevando a cabo las acciones de preprocesamiento de los datos y accediendo al DBMS. Para el sistema MAS también se incorpora el lenguaje ACL el cual permite la comunicación entre sitios.

1.8. Plan de Experimentación

El plan de experimentación consiste en analizar los siguientes cuatro escenarios:

1. Centralizado sin agentes.
2. Centralizado con agentes.
3. Distribuido sin agentes.
4. Distribuido con agentes.

Considerando como variables la ubicación de los datos y la utilización de los agentes.

En cada escenario se ejecutarán tareas de minería de datos y se realizarán las siguientes mediciones:

- Costo de tiempo de acceso a los datos: Tiempo requerido para que el sistema accese a los datos en disco.
- Costo de transmision de los datos: Tiempo que tarda para que la informacion sea transmitida a otro sitio.
- Costo de procesamiento (CPU): Periodo de tiempo utilizado en el procesamiento de una tarea de mineria de datos.

Posteriormente se recabarán los resultados a fin de analizar los resultados y se establecerán conclusiones acerca de la hipótesis a través de los objetivos.

1.9. Organización de la tesis

Éste documento está organizado de la siguiente manera:

En el capítulo II brevemente se explicará qué es la minería de datos, su papel dentro del proceso de descubrimiento de conocimiento en bases de datos (KDD), se mencionan las tareas mas representativas de la minería de datos, así como la arquitectura de minería de datos centralizada, los medios de almacenamiento en minería de datos y cómo los procesos computacionales que están involucrados en ésta han motivado que los sistemas de mineria de datos sean implementados en forma distribuida.

En el capítulo III se describirá en detalle la implementación de los algoritmos K-Medias y Clustering Jerárquico, así como los criterios más importantes para evaluar el rendimiento de los algoritmos, y obtener su precisión.

En el capítulo IV se mostrará qué es un sistema basado en agentes, que características tienen los sistemas multi-agentes, como está estructurada una arquitectura basada en agentes, y como implementar un sistema basado en agentes.

En el capítulo V se presenta la implementación del marco de trabajo para realizar de tareas de mineria de datos en forma distribuida utilizando multi-agentes. Se presentará como estará estructurado el sistema, se establecerá una fase de análisis en la cual se detalla cada componente del sistema através de la metodología GAIA, las tareas de análisis de clustering que serán implementadas y una descripción de como está construído cada agente que participa en el sistema, que tarea realizan y cómo los agentes interactúan y se comunican.

En el capítulo VI se presenta el plan de experimentación constituido por cuatro escenarios: [1] centralizado con agentes, [2] centralizado sin agentes, [3] distribuido sin agentes y [4] distribuido con agentes, donde las variables independientes serán: el tiempo en acceso a los datos, el tiempo de transmición de los datos y el costo de procesamiento.

En el capítulo VII se presenta el análisis de resultados del plan de experimentación para saber si se llegó al objetivo general.

En el capítulo VIII se establecen las conclusiones a partir del análisis de resultados con respecto a los objetivos específicos y se describen los trabajos futuros para DDM y MAS.

Capítulo 2

Minería de Datos Centralizada y Distribuida

Muchas aplicaciones corporativas producen una gran cantidad de datos de los cuales se requiere obtener información que le pueda resultar útil a las organizaciones al momento de tomar decisiones. Esto ha generado el desarrollo de una variedad de técnicas computacionales que puedan convertir una gran cantidad de datos en conocimiento que resulte invaluable para muchas organizaciones. Con el propósito de poder mejorar la precisión de esta información el proceso *minería de datos* ha integrado muchas de estas técnicas y ha emergido como uno de los procesos más importantes para el descubrimiento de conocimiento debido a que permite procesar una enorme cantidad de datos generados por diversos sistemas de cómputo, y además incorpora muchos mecanismos útiles para el análisis de datos. El proceso de minería de datos ha mejorado con el tiempo, volviéndose una de las tecnologías más importantes para la exploración y análisis de datos. El uso de la minería de datos permite tratar datos masivos de forma eficiente y extraer conclusiones e información relevante para las organizaciones de negocio. Puede tratar además, diversos tipos de datos los cuales se encuentran almacenados en distintos repositorios, que van desde archivos planos, hojas de cálculo, hasta aquellos más estructurados como una base de datos o un datawarehouse.

En el presente capítulo se define el concepto de minería de datos y el papel que desempeña dentro del Proceso de Descubrimiento de Conocimiento en Bases de Datos (*Knowledge Discovery in Databases* ó *KDD*). Se describirán los modelos y las técnicas más importantes que son aplicados en minería de datos. Finalmente se analizará como está estructurado un sistema de minería de datos centralizado y como los procesos computacionales que están involucrados en este tipo de sistemas promueven el uso de sistemas de minería de datos distribuida.

2.1. ¿Qué es la minería de datos?

La *minería de datos* es un proceso que engloba sus fundamentos teóricos en otras áreas de conocimiento como son la estadística, la inteligencia artificial y las máquinas de aprendizaje. Así que la minería de datos puede ser definida en primera instancia como un conjunto de metodologías y técnicas derivadas de la estadística, la inteligencia artificial, y las máquinas de aprendizaje que están enfocadas en descubrir patrones desconocidos de los datos **[Wang and Fu, 2005]**. La información y conocimiento generados por la minería de datos han atraído tanta atención que otros campos de estudio diferentes a los de las ciencias de la computación han empezado a usar la minería de datos; entre estos están: la administración de negocios, finanzas, el análisis de mercado, los procesos industriales, medicina, química, biología, entre otras.

En la diversa literatura que existe sobre minería de datos podemos encontrar también otras definiciones como las siguientes:

> **[Han and Kamber, 2006]** Minería de datos se refiere a extraer ó "minar" conocimiento de grandes volúmenes de datos.
>
> **[Witten and Frank, 2005]** La minería de datos es definida como el proceso de descubrir patrones en los datos. El proceso debe ser automático ó (más usualmente) semi-automático. Los patrones descubiertos deben ser significativos de manera que permitan alguna ventaja, usualmente una ventaja económica.
>
> **[Hand, Mannila, and Smyth, 2001]** Minería de datos es el análisis (frecuentemente largo) de conjuntos de datos observacionales para encontrar interrelaciones no sospechadas y para sumarizar los datos en nuevas maneras que sean entendibles y útiles al propietario de los datos.
>
> **[Sumathi and Sivanandam, 2006]** Minería de datos es el proceso de descubrir nuevas correlaciones significativas, patrones, y tendencias, extraídos (minados) de grandes cantidades de datos almacenados en un almacén de datos, usando técnicas de estadística, máquinas de aprendizaje, inteligencia artificial (AI), y visualización de datos.

Todas estas definiciones concuerdan en que la minería de datos es un proceso que principalmente se refiere a extraer información de grandes volúmenes de datos, ésta información extraída puede representar patrones, correlaciones y tendencias que resultan útiles para los usuarios finales. Otro detalle que se observa es que éstos datos se encuentran almacenados en un almacén de datos (data warehouse). Un *data warehouse* es un repositorio de información recolectada de múltiples sitios, almacenada bajo un esquema unificado, y que usualmente reside en un sólo sitio central **[Han and Kamber, 2006]**.

En un sistema de minería de datos centralizada los datos que se encuentran en diferentes sitios (que pueden estar almacenados en bases de datos, archivos planos, hojas de cálculo, etc.) son recolectados en un datawarehouse para que alguna técnica de minería de datos pueda ser aplicada a ellos para obtener información. Es cierto que las técnicas de minería de datos podrían ser aplicadas directamente sobre los datos contenidos en una base de datos o archivo plano. Sin embargo, la ventaja que ofrece el data warehouse es que los datos pueden ser unificados bajo un mismo esquema y recolectados en un sólo repositorio para que más adelante se les aplique alguna técnica de minería de datos. Hacer esta integración de datos permitirá que la información obtenida sea más precisa. Como último punto a destacar, la minería de datos como se mencionó anteriormente es un proceso que integra técnicas de diversas disciplinas de las ciencias de la computación, principalmente de la estadística, máquinas de aprendizaje y la inteligencia artificial, entre otras, como se muestra en la **figura 2.1**.

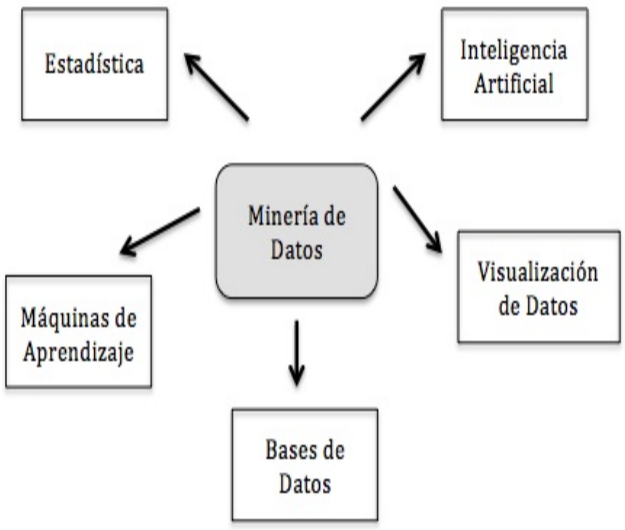


Figura 2.1: **Diversas disciplinas asociadas con la minería de datos**. Adaptado de **Han and Kamber, 2006**.

Los factores que han impulsado al uso de la minería de datos son **[Larose, 2005]**:

- El explosivo crecimiento de la recolección de datos.
- El almacenamiento de los datos en data warehouses, y el acceso seguro a los datos.
- La disponiblidad de incrementar el acceso a datos de internet.
- La presión competitiva para incrementar el mercado compartido en un mundo globalizado.

- El tremendo crecimiento en el poder de cómputo y capacidad de almacenamiento.

2.2. Proceso de descubrimiento de conocimiento en Bases de Datos

Al paso del tiempo el término *minería de datos* se ha manejado indistintamente con otros términos, tales como, *extracción de conocimiento, descubrimiento de información, recolección de la información, arqueología de datos, y procesamiento de patrones de datos*. Esto ha generado que los nombres *minería de datos* y *descubrimiento de conocimiento en bases de datos* (*KDD* del ingles *Knowledge discovery in databases*) también sean considerados similares. Por ello es necesario hacer enfásis en la diferencia entre estos dos términos. Cuando el término KDD[1] fue propuesto éste era empleado para describir todos los procesos que estaban involucrados en la extracción de conocimiento de los datos. Cada uno de estos procesos intenta completar una tarea en particular en el descubrimiento de conocimiento. Esta serie de procesos incluye como los datos son almacenados y accesados, como usar algoritmos eficientes y escalables para analizar conjuntos de datos masivos, como interpretar y visualizar los resultados, y como modelar y soportar la interacción entre la computadora y el humano. **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**. Así que una definición del KDD es: "la extracción no trivial de conocimiento implícito previamente desconocido y potencialmente útil de los datos". **[Adriaans and Zantinge, 1996]**. Para poder marcar cuál es la diferencia entre estos dos términos simplemente se puede decir que KDD es una serie de procesos que están enfocados a descubrir conocimiento en bases de datos, mientras que la minería de datos viene a ser la aplicación de las diferentes técnicas de la inteligencia artificial, las máquinas de aprendizaje y la estadística sobre los datos. Esto también nos permite entonces comprender que la minería de datos constituye un proceso dentro del KDD, y que es considerado como uno de los procesos más importantes del KDD debido a que la minería de datos es el proceso donde se extrae el conocimiento, los patrones y tendencias.

El descubrimiento de conocimiento consiste entonces de una serie de etapas (en cada etapa se lleva a cabo un proceso) que se llevan a cabo de forma iterativa como se muestra en la **Figura 2.2**, y que se describen a continuación **[Han and Kamber, 2006]**:

- **Limpieza de datos**: Se remueve ruido y datos inconsistentes.

[1]"El término *Knowledge Discovery in Database* fue propuesto en el año de 1995, en la primera Conferencia en Descubrimiento de Conocimiento y Minería de Datos, llevada a cabo en Agosto de 1995 en Montreal, Quebec, Canada"

- **Integración de datos**: Múltiples fuentes de datos pueden ser combinadas.
- **Selección de datos**: Los datos que son relevantes para tareas de análisis son recuperados de la base de datos.
- **Transformación de los datos**: Los datos son transformados o consolidados en forma apropiada para minarlos, por medio de sumarización u operaciones de agregación, por instancia.
- **Minería de Datos**: Es el proceso esencial donde métodos inteligentes son aplicados para extraer los patrones de datos
- **Evaluación de Patrones**: Para identificar que patrones representan verdaderamente conocimiento considerando **medidas de interes** [2].
- **Presentación del Conocimiento**: Visualización y técnicas de representación de conocimiento son usadas para representar el conocimiento minado a los usuarios.

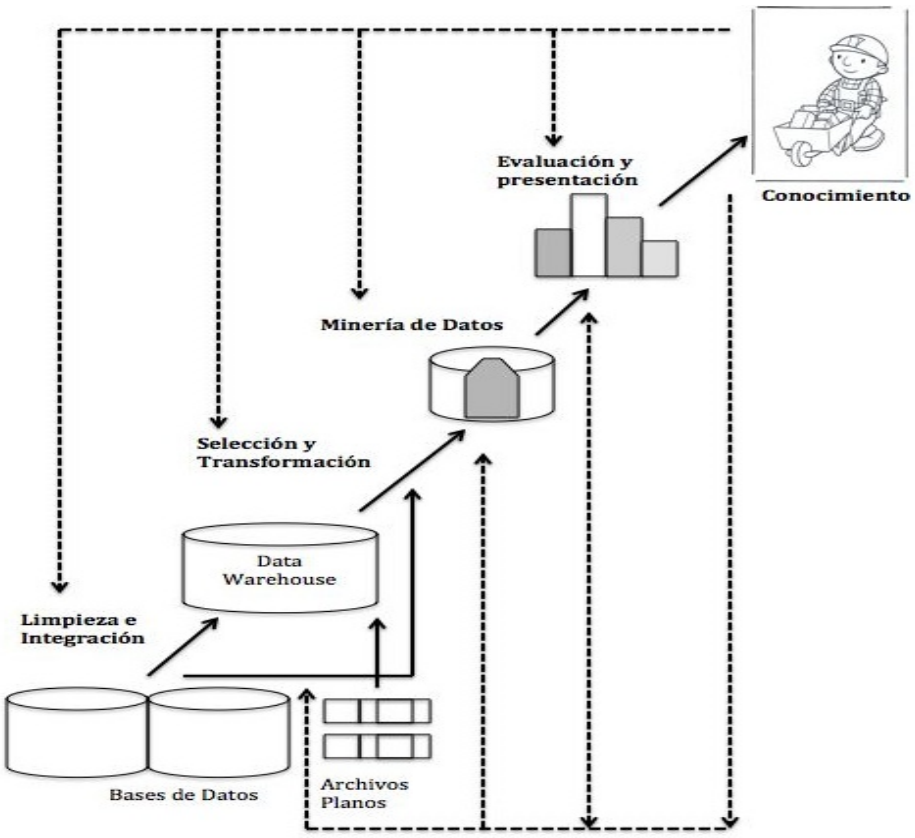


Figura 2.2: **Proceso de descubrimiento de conocimiento en bases de datos(KDD)**. Adaptado de **Han and Kamber, 2006**.

[2]"Estas medidas generalmente son proporcionadas por los usuarios"

2.3. Estructura de un sistema de Minería de Datos

Como la minería de datos trata frecuentemente con grandes cantidades de datos, es necesario que estos sean administrados de manera especial, utilizando sistemas que permitan procesar tal cantidad de datos. El uso de estos sistemas está basado en cuatro razones **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**:

- *El conjunto de datos correspondientes puede ser que no quepa en memoria en una computadora usada para minería de datos*, y por lo tanto el sistema debe poder administar la recuperación de estos datos.
- *Los métodos de minería de datos puede ser que necesiten ser aplicados a diferentes subconjuntos de datos* y por lo tanto se requiere que el sistema recupere los trozos de datos requeridos.
- *Los datos necesitan ser dinámicamente añadidos, actualizados y algunas veces por diferentes personas en diferentes localizaciones*. Un sistema de administración de datos requiere manejar control de concurrencia, control de integridad y recuperación a fallas.
- *Los archivos planos pueden incluir porciones importantes de información redundante*, lo cual puede ser evitado si un sólo conjunto de estos datos son almacenados en tablas múltiples (un rasgo característico de los sistemas de administración).

Basado en las razones anteriormente mencionadas razones una arquitectura para un sistema de minería de datos, como se representa en la **Figura 2.3** debe contar con los siguientes componentes **[Han and Kamber, 2006]**:

- **Repositorio de datos**. Este repositorio puede ser una base de datos, un data warehouse, hojas de cálculo u otros tipos de repositorios.
- **Servidor de base de datos o datawarehouse**. Este es responsable de recuperar datos relevantes, basados en los requerimientos de minería de datos de los usuarios.
- **Base de Conocimiento**. Esta es usada para salvar conocimiento que es necesario en minería de datos. Éste conocimiento puede ser usado para guiar los procesos de búsqueda de la minería de datos ó ayudar a evaluar los resultados generados en el proceso de minería de datos.
- **Ingeniería de minería de datos**. Es la parte básica de un sistema de minería de datos, el cual está compuesto de grupos de módulos funcionales, que son usados para analizar los datos, clasificarlos, agruparlos, entre otras funciones.

- **Módulo de evaluación de patrones**. Este componente emplea medidas de interés e interactúa con el módulo de minería de datos que se enfoca en descubrir patrones de interés.
- **Interface de usuario**. Ésta se comunica con los usuarios y el sistema de minería de datos, permite al usuario interactuar con el sistema, recibiendo sus solicitudes y proveyendole información.

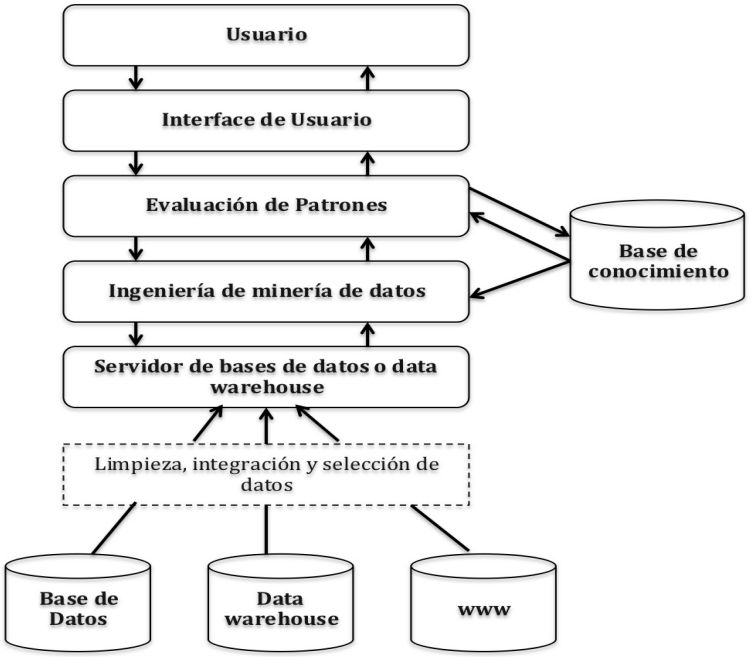


Figura 2.3: **Estructura de un sistema de minería de datos**. Adaptado de **Han and Kamber, 2006**.

2.4. Almacenamiento de los Datos

Para poder descubrir conocimiento importante en minería de datos es necesario que los datos disponibles mantegan una buena calidad y que la cantidad de estos sea lo suficientemente grande. Debido a la diversidad de formatos que los datos pueden tener es necesario que estos mantengan una adecuada organización y que existan las adecuadas técnicas para poder almacenarlos.

En un nivel elemental los datos pueden representar la unidad más simple de información, el **valor**, a partir de un valor o conjunto de valores un **atributo** ó **característica** puede ser determinado. Los **objetos** representados por entidades pueden ser descritos por una ó más características, los cuales pueden ser

combinados para formar **conjuntos de datos**, que pueden ser almacenados en **archivos planos** y en otros formatos usando **bases de datos** y **data warehouse [Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**. La relación entre todos estos conceptos serán descritos en la **sección 2.4.1** y mostrados en la **figura 2.4**.

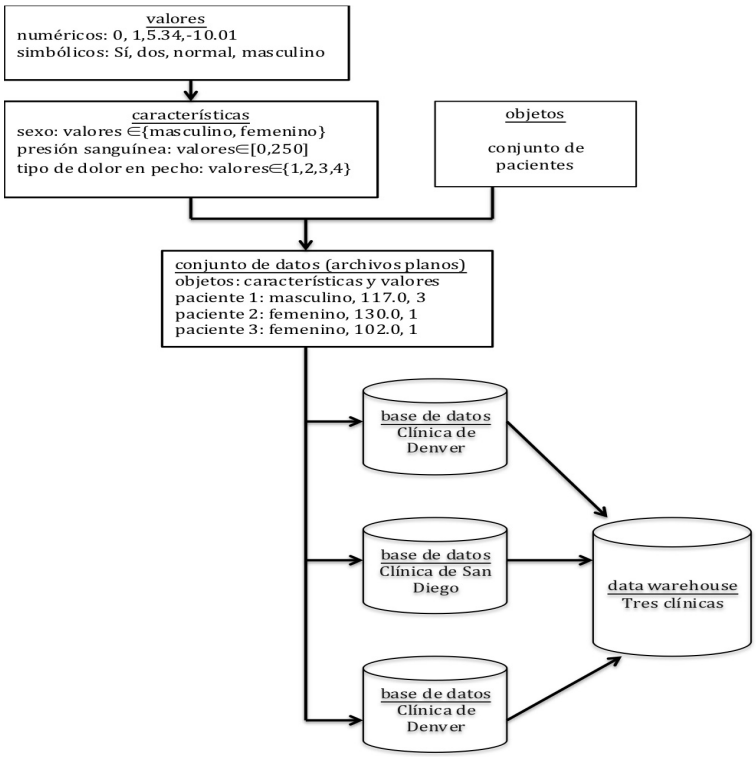


Figura 2.4: **Relación entre valores, características, objetos, conjuntos de datos, bases de datos y data warehouses**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

2.4.1. Valores, Características, y Objetos

Los datos pueden representar dos tipos de valores: **numéricos** y **simbólicos**. Los numéricos están representados por números reales (-1.09,123.5), enteros (1, 46, 138), números primos (1, 3, 5), etc. Por otro lado también pueden ser simbólicos representando datos cualitativos, tales como colores (blanco, rojo) o tamaños (pequeño, mediano, grande), entre otros.

A partir de un conjunto de valores, un **atributo** ó **característica** puede ser expresado, donde éstos conjuntos de valores pueden ser **discretos** (categóricos) o **continuos**. Los atributos discretos están representados por un número total de valores relativamente pequeños (finitos). Un caso especial de atributo discreto es el **binario** el cual puede tener sólamente dos valores distintos. Por otro lado

los atributos continuos pueden representar un número total de valores muy grande (infinitos) y pueden cubrir un intervalo específico (rango). En el proceso de minería de datos, una etapa de **discretización** es llevada a cabo, en donde los atributos continuos son transformados en atributos discretos **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**.

Objetos (también conocidos como registros, muestras, casos, individuos, puntos de datos) representan entidades descritas por una o más características ó atributos. El caso en donde un objeto es descrito por muchas características es conocido como *dato multivariado*, y en el caso donde es descrito por una sóla característica es conocido como *dato univariado*.

En la figura 2.5 se observa el ejemplo de pacientes en una clínica de enfermedades cardíacas. Un paciente es un objeto que puede ser descrito por un número de atributos o características, tales como nombre, sexo, edad, resultados diagnósticos tales como presión sanguínea, nivel de colesterol, y evaluaciones cualitativas como pueden ser dolor en el pecho y la severidad de éste.

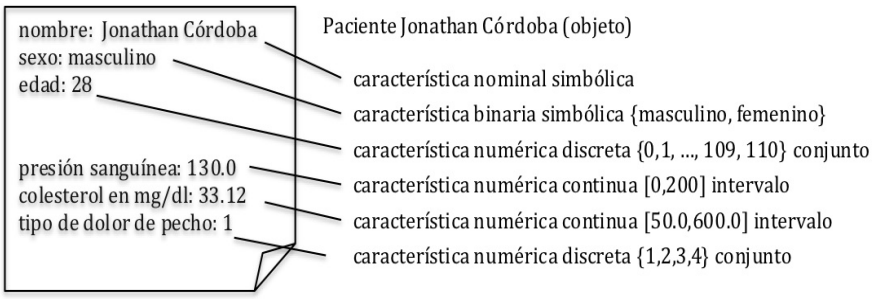


Figura 2.5: **Objeto Paciente**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

2.4.2. Conjunto de datos

Muchas herramientas para análisis de datos estadísticos y de minería de datos asumen que los datos están organizados en **archivos planos**, en los cuales cada fila representa un objeto y las columnas representan las características o atributos, resultando en un archivo plano que forma un arreglo en dos dimensiones. Los archivos planos permiten almacenar datos en un formato sencillo de texto, y ellos son frecuentemente generados de datos almacenados en otros formatos, como pueden ser una hoja de cálculo ó una tabla entidad-relación.

En la **tabla 2.1** se ve el diseño de un conjunto de datos para el caso de los pacientes de la clínica de enfermedades cardíacas, en donde cada paciente (objeto) es descrito a partir de las siguientes de características **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**:

- *nombre* (característica simbólica)
- *edad* (característica numérica discreta de el conjunto {0,1,. . . ,109,110})
- *sexo* (característica binaria simbólica de el conjunto {masculino, femenino})
- *presión sanguínea* (característica numérica continua de el intervalo [0,200])
- *fecha de examen de presión sanguínea* (característica tipo fecha)
- *colesterol en mg/dl* (característica numérica continua de el intervalor [50.0, 600.0])
- *fecha de examen de colesterol* (característica tipo fecha)
- *tipo de dolor en el pecho* (característica numérica discreta de el conjunto {1, ,2, 3, 4})
- *tipo de defecto* (característica simbólica de el conjunto {normal, fijo, reversible})
- *diagnóstico* (característica simbólica binaria de el conjunto {presente, ausente})

Nombre	Edad	Sexo	Presión Sanguínea	Fecha de examen de Presión Sanguínea	Colesterol en mg/dl	Fecha de examen de Colesterol	Tipo de dolor en el pecho	Tipo de defecto	Diagnóstico
Jonathan Córdoba	28	masculino	130.0	05/05/2011	NULL	NULL	NULL	NULL	NULL
Jonathan Córdoba	28	masculino	130.0	05/05/2011	331.2	05/21/2011	1	normal	ausente
Magda González	26	femenino	115.0	01/03/2011	NULL	NULL	4	fijo	presente
Magda González	26	femenino	115.0	01/03/2011	407.5	06/22/2011	NULL	NULL	NULL
Anna López	56	femenino	120.0	12/30/2010	45.0	12/30/2010	2	normal	ausente
...	...	...	...	...	...	...	...	...	...

Tabla 2.1: **Archivo plano con un conjunto de datos para la clínica de pacientes cardíacos**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

2.4.3. Bases de Datos

Muchas bases de datos son generadas a partir de archivos planos. Si los datos están almacenados en una base de datos es más fácil para los programas poder administrar y accesar a los datos. Una base de datos comúnmente contiene un tipo de **esquema relacional** el cual consiste de un conjunto de tablas, las tablas consisten de **atributos** (llamados columnas o campos) y **tuplas** (también llamados filas). En los archivos planos las tuplas representan objetos y los atributos representan características.

Las bases de datos relacionales incluyen el modelo **entidad-relación** (ER), en el cual se definen un conjunto de entidades (tablas y tuplas, etc.) y sus relaciones. Para aprovechar mejor las capacidades de una base de datos, el poder dividir un conjunto de datos en tablas se vuelve una ventaja importante ya que esto permite poder remover información redundante en los datos **[Silberschatz, Korth, and Sudarshan, 2002]**.

Otra característica importante de las bases de datos es el contar con un lenguaje especializado, el **SQL** (Strutured Query Language) el cual provee una manera rápida de acceder a los datos, ya que permite a los usuarios poder realizar consultas y obtener un conjunto relevante de datos que proporcionan información **[Donahoo and Speegle, 2005]**. El acceso a los datos puede ser de manera concurrente (varios usuarios pueden acceder a los datos al mismo tiempo) y distribuida (los datos se encuentran almacenados en diferentes localizaciones).

Un punto clave de las bases de datos es que proporcionan seguridad y consistencia al momento de almacenar datos. Lo que garantiza que no podrá haber accesos no autorizados a los datos o que los datos que accesen al sistema presenten inconsistencias.

Para el ejemplo de la clínica de enfermedades cardíacas, en el modelo entidad-relación el archivo plano estaría dividido en cuatro tablas relacionales. La tabla *paciente* almacena la información básica sobre los pacientes, junto con su información diagnóstica. Las tablas *examen_presion_sanguinea* y *examen_colesterol* almacenan los correspondientes valores para estos examenes. Finalmente la tabla *examenes_efectuados*, representa las relaciones entre las otras tablas. Para relacionar las tablas, un conjunto de atributos clave es usado. Por ejemplo, el atributo *paciente_ID* es usado para vincular la tabla *paciente* con las tablas *examen_presion_sanguinea*, *examen_colesterol* y *examenes_efectuados*. Por otro lado los atributos *examen_presion_sanguinea_ID* y *examen_colesterol_ID*, se usan para relacionar las tablas *examen_presion_sanguinea* y *examen_colesterol* con la tabla *examenes_efectuados*. La tabla paciente es representada en la **figura 2.6 [Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**.

paciente

paciente_ID	nombre	edad	sexo	tipo_dolor_pecho	tipo_defecto	diagnostico
P1	Jonathan Cordoba	28	masculino	1	normal	ausente
P2	Magda González	26	femenino	4	fijo	Presente
P3	Anna López	56	femenino	2	normal	Ausente
...	...	...	...	...	...	...

examen_presion_sanguinea

examen_presion_sanguinea_ID	paciente_ID	presion_sanguinea	fecha_examen_presion_sanguinea
BPT1	P1	130.0	05/05/2011
BPT2	P2	115.0	01/03/2011
BPT3	P3	120.0	12/30/2010
...	...	...	...

examen_colesterol

examen_colesterol_ID	paciente_ID	colesterol_mg/dl	fecha_examen_colesterol
SCT1	P1	331.2	05/21/2011
SCT2	P2	407.5	06/22/2011
SCT3	P3	45.0	12/30/2010
...	...	...	...

examenes_efectuado

paciente_ID	examen_presion_sanguinea	examen_colesterol
P1	BPT1	SCT1
P2	BPT2	SCT2
P3	BPT3	SCT3
...	...	...

Figura 2.6: **Base de Datos relacional para la clínica de pacientes cardiacos**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

2.4.4. Data Warehouses

Un **data warehouse** es un repositorio de datos en el cual los datos son recolectados de diferentes localizaciones y almacenados usando un esquema unificado. Un data warehouse es usado comúnmente para aplicar un conjunto de pasos de procesos para los datos que provienen de múltiples bases de datos. Además de proveer un sistema para la administración de datos para las tomas de decisiones. Una arquitectura para un datawarehouse consiste principalmente de los siguientes componentes **[Linoff and Berry, 2011]**:

- **Sistemas fuentes**. De donde vienen los datos.
- **Herramientas para extracción, transformación, y carga (ETL)**. Los cuales resuelven el problema de recolectar datos de sistemas dispares proveyendo la habilidad para mapear y mover datos de sistemas fuentes a otros entornos.
- **Repositorio Central**. Es el almacén principal para el data warehouse. Este almacén es usualmente una base de datos relacional la cual es accesada através de algunas variantes de SQL.

- **Sandbox analítico**. Provee un entorno para hacer análisis más complejo por medio de consulta SQL ó herramientas de minería de datos.
- **Repositorio de metadatos**. Describe que es lo que se puede hacer y en donde. En el nivel más bajo los metadatos son los esquemas de las base de datos y proveen información de la disponibilidad de los datos, proporcionando herramientas para buscarlos dentro del contenido del datawarehouse.
- **Data marts**. Proporciona acceso especializado para los usuarios finales y las aplicaciones.
- **Feedback operacional**. Integra soporte para la toma de decisiones en la aplicación.
- **Usuario Final**. Es para quien está dirigido la construcción del data warehouse.

Un data warehouse usualmente usa una estructura de base de datos multimensional, donde cada dimensión corresponde a un atributo o conjunto de atributos seleccionados por el usuario que serán incluídos en el esquema. Cada celda en la base de datos corresponde a alguna medida de agregación, tales como promedios, cuentas, minimos, etc. La implementación real del data warehouse puede ser una base de datos relacional o un **cubo de datos** multidimensional. La arquitectura para un datawarehouse se muestra en la **figura 2.7**

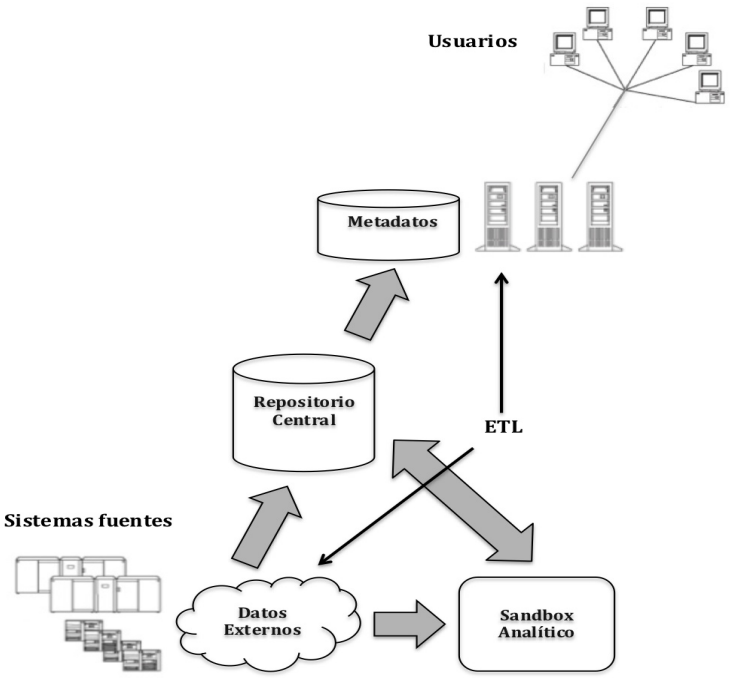


Figura 2.7: **Arquitectura para un data warehouse**. Adaptado de **Linoff and Berry, 2011**.

Para el caso de la clínica de enfermedades cardíacas, considere un escenario en el que esta clínica pertenece a cierto grupo de clínicas ubicadas en diferentes ciudades, las cuales pertenecen a la misma compañia. Cada clínica tiene su propia base de datos y se requiere analizar los datos de todas las clínica para obtener información. Una forma para obtener información sería analizar los datos de cada clínica individualmente, pero esto podría ser difícil de llevar a cabo para todas las clínicas. En este caso usar un data warehouse sería una mejor estratégia, debido a que este ofrece un sólo repositorio en cual todos los datos de las diferentes localizaciones (bases de datos relacionales) podrían ser recolectados, almacenados y unidos en un sólo esquema. Si cierto usuario accesa a la información del data warehouse este podría seleccionar ciertos atributos como temas de interes, como son los pacientes, los tipos de examenes clínicos y los diagnósticos. Y además podría obtener información desde una perspectiva histórica debido a que en el data warehouse se puede recopilar y organizar la información de años anteriores **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**. La **figura 2.8** muestra la arquitectura de un datawarehouse para la clínica cardiaca.

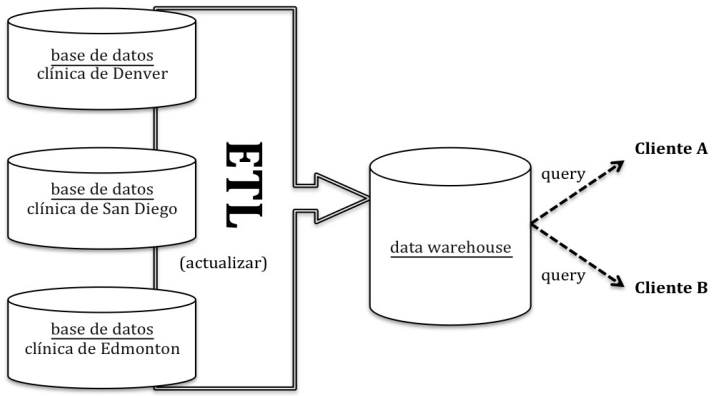


Figura 2.8: **Arquitectura de un data warehouse para la clínica de enfermedades cardíacas**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

2.4.5. Avances en Almacenamiento de datos

Usualmente las bases de datos relacionales son usadas por organizaciones para almacenar grandes cantidades de datos tales como tiendas comerciales, bancos, etc. Pero en años recientes han surgido nuevas bases de datos más especializadas las cuales tratan de satisfacer las necesidades de usuarios más especializados quienes no sólo manejan datos numéricos o nominales. Los manejadores de bases de datos actuales pueden manipular datos **transaccionales**, datos **espaciales**, **hipertextos**, tales como HTML y XML; datos **multimedia** (imágenes, videos

y audio); datos **temporales**, tales como **series de tiempo** orientadas a la bolsa de valores y registros históricos; y el **internet**. Los desafíos para los manejadores de bases de datos están en hacer frente a objetos con longitud variables, datos estructurados y no estructurados, formatos de datos multimedia, y a objetos con grandes cantidades de información **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007] [Han and Kamber, 2006]**.

2.5. Modelos y tareas de minería de datos

Para que la minería de datos pueda extraer conocimiento útil de los datos mediante análisis, ésta recurre a modelos que permiten encontrar relaciones, patrones o reglas inferidas. Los modelos que emplea la minería de datos son: el *modelo descriptivo* y el *modelo predictivo*.

2.5.1. Modelo Descriptivo

El modelo descriptivo trata de proporcionar información entre las relaciones de los datos y sus características. Generalmente ésta clase de modelo trata de responder a preguntas como por ejemplo **[Linoff and Berry, 2011]**:

- Los clientes que compran X también compran Y.
- Los niños que no tienen X son muy distintos del resto.
- X y Y son los factores más influyentes en contraer la enfermedad Z.
- etc, ...

Para proporcionar tal información el modelo descriptivo lleva a cabo tareas de *agrupamiento* (*clustering*) y *reglas de asociación* que principalmente tratan de descubrir asociaciones y correlaciones de los patrones que frecuentemente ocurren dentro de los datos. Los modelos descriptivos siguen un tipo de *aprendizaje no supervisado*, que consiste en adquirir conocimiento de los datos que están disponibles, sin requerir influencia externa que indique un comportamiento deseado del sistema **[Sumathi and Sivanandam, 2006]**.

2.5.2. Modelo Predictivo

El modelo predictivo trata de estimar valores futuros de variables de interés. Para hacer tal predicción el proceso hace un estudio de comportamientos de datos pasados. Algunas de las preguntas que podríamos responder con este tipo de modelo son **[Linoff and Berry, 2011]**:

- ¿Qué tal se venderá un producto X el próximo año?
- ¿Qué producto comprará, X persona?
- ¿Donde se producirá el siguiente atentado terrorista?
- Con base en las características de una persona, ¿qué riesgo tiene de contraer una enfermedad X?
- etc, ...

El estudio de comportamiento pasado de los datos puede hacerse mediante clasificación, análisis de regresión, redes neuronales, clasificación de k-vecinos cercanos o cualquier otro método que permita encontrar un modelo (o función) que describa y distinga datos en clases o conceptos, con el propósito de poder usar tal modelo para predecir la clase de objetos cuyas etiquetas son desconocidas. A diferencia de los modelos descriptivos, los modelos predictivos siguen un *aprendizaje supervisado*, que consiste en aprender mediante el control de un supervisor o maestro (el modelo generado) que determina la respuesta que se desea generar del sistema **[Sumathi and Sivanandam, 2006]**.

2.5.3. Tareas de minería de datos

Como se describió brevemente, cada modelo de minería datos consiste de una serie de tareas que tratan de resolver un tipo de problema específico en el proceso de mineria de datos **[Siraj and Abdoulha, 2011]**. En la **figura 2.9** se muestra una representación general de los modelos de minería de datos y las distintas tareas que son llevadas en cada modelo.

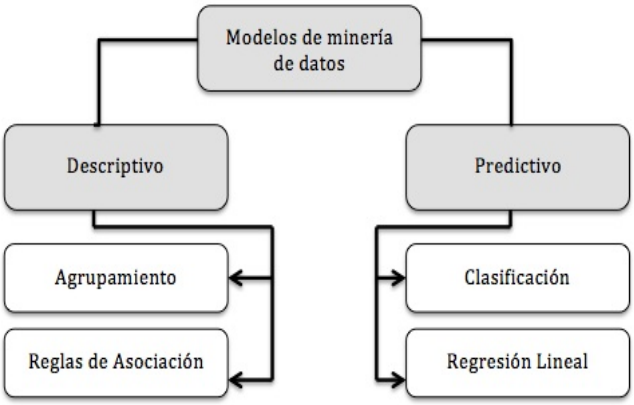


Figura 2.9: **Representación general de los modelos y las tareas de minería de datos**. Basado en **Sumathi and Sivanandam, 2006**.

En esta sección se menciona de manera general las tareas que son llevadas acabo en los modelos de minería de datos. En el **capítulo 3**, se describirán algunas de ellas de manera más detallada enfocando ese capítulo a la agrupación de datos, que van a ser un elemento importante del sistema que se construirá en el **capítulo 5**.

2.5.3.1. Reglas de Asociación

Las reglas de asociación están asociadas con el concepto de *conjuntos de elementos frecuentes* el cual trata de encontrar elementos en un conjunto de datos que frecuentemente aparecen juntos en un transacción. Formalmente una regla de asociación es una implicación de la forma $X \Longrightarrow I_j$, donde X es un subconjuto de items del conjunto I, e I_j es un elemento de I que no está presente en X. Una regla de la forma $X \Longrightarrow I_j$ se satisface en el conjunto de transacciones T con un factor de confianza $0 \leq c \leq 1$ si al menos un c % de las transacciones que satisface X también satisfacen I_j **[Agrawal, Imielinski, and Swami, 1993]**. Alguna restricción que existe en las reglas de asociación tiene que ver con el número de transacciones que dan soporte a un regla o el porcentaje de transacciones que deben cumplir esa regla.

A continuación se presenta un ejemplo que ilustra este concepto:

La **tabla 2.2** contiene una serie de transacciones de la rama de tiendas *All Electronics* **[Han and Kamber, 2006]**:

TID	Lista de Items
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Tabla 2.2: **Datos transaccionales para la rama de tiendas *AllElectronics***. Basado en **Han and Kamber, 2006**.

Ahora suponga que el conjunto de datos que contiene los items es I : {I1, I2, I3, I4, I5} y que los items frecuentes son {I1, I2, I5}. ¿Cuántas reglas de asociación podrían ser generadas de este conjunto? El conjunto no vacío de subconjuntos: {I1,

I2},{I1, I5},{I2, I5}, {I1},{I2},{I5}. Las reglas de asociación generadas se muestran junto con sus factores de confianza **[Han and Kamber, 2006]**:

$$
\begin{array}{lll}
1 & I1 \wedge I2 \Longrightarrow I5 & \text{confianza} = 2/4 = 50\,\% \\
2 & I1 \wedge I5 \Longrightarrow I2 & \text{confianza} = 2/2 = 100\,\% \\
3 & I1 \wedge I5 \Longrightarrow I1 & \text{confianza} = 2/2 = 100\,\% \\
4 & I1 \Longrightarrow I2 \wedge I5 & \text{confianza} = 2/6 = 33\,\% \\
5 & I2 \Longrightarrow I2 \wedge I5 & \text{confianza} = 2/7 = 29\,\% \\
6 & I5 \Longrightarrow I1 \wedge I5 & \text{confianza} = 2/2 = 100\,\%
\end{array}
$$

Si tuviéramos un soporte de 70 % entonces las reglas 2,3 y 6 serían salidas ya que son las más generadas fuertemente.

2.5.3.2. Agrupación

Las técnicas de agrupación consideran a las tuplas de datos como objetos. Éstas particionan los objetos en grupos o *clusters*, donde los objetos dentro de un cluster son "similares" y "disimilares" a objetos en otros clusters. La similaridad se define como que tan "cercanos" son los objetos en un espacio. Ésta cercanía se mide con base en una función de distancia **[Maimon and Rokach, 2005]**. La "calidad" de un cluster puede ser representada por su *diámetro*, la distancia máxima entre dos objetos en el cluster **[Halkidi, Batistakis, and Vazirgiannis, 2001]**.

La agrupación representa un *aprendizaje no supervisado*, en el cual los elementos en el sistema tienen que descubrir sus propias clases. Para llevar acabo la agrupación de elementos, se emplean algunos de los siguientes métodos **[Tang, Steinbach, and Kumar, 2006]**:

- **K-medias**. La idea de este método es tomar aleatoriamente puntos de datos como centros de cluster. Después, cada registro o punto es asignado al cluster más cercano en términos de un error cuadrático o distancia Euclidiana. Un nuevo centro es calculado tomando la media de todos los puntos en un cluster, ajustando el escenario para la siguiente iteración **[Lloyd, 2003]** **[MacQueen, 1967]**.

- **Algoritmos jerarquícos**. Éstos métodos empiezan con un conjunto de puntos distintos, cada uno formando su propio cluster. Entonces recursivamente, dos clusters que están cercanos son mezclados en uno, hasta que todos los puntos pertenezcan a un sólo cluster **[Day and Edelsbrunner, 1984]** **[Kaufman and Rousseeuw, 1990]**.

Para mostrar como se efectua la agrupación de datos se verá el siguiente ejemplo, en el cual se usa el método de **K-Medias**. Previamente se describirá este algoritmo, para despues mostrar el ejemplo **[Teknomo, 2007]**:

K-Medias. Agrupar datos en K-Medias es simple. Inicialmente se determina el número de grupos o clusters K y se asume el centro de esos grupos. Para determinar los centros hay dos alternativas prácticas: la primera es tomar de forma aleatoria K objetos como los centros iniciales y la segunda es tomar los primeros K objetos en secuencia.

En el algoritmo de **K-Medias** se deben de ejecutar tres pasos importantes **[Teknomo, 2007]**:

1. Determinar el centro ó los centros iniciales de acuerdo al número de clusters esperado.
2. Determinar la distancia de cada objeto con relación a los centroides.
3. Agrupar los objetos con base en la mínima distancia.

Ejemplo:

Suponga que tiene cuatro tipos de medicina con dos atributos (peso e índice PH) como se muestra en la siguiente tabla:

MEDICINA	PESO	INDICE PH
A	1	1
B	2	1
C	4	3
D	5	4

Tabla 2.3: Tipos de Medicina. Basada en **Teknomo, 2007**.

Cada medicina puede ser representada como un punto en un espacio coordenado:

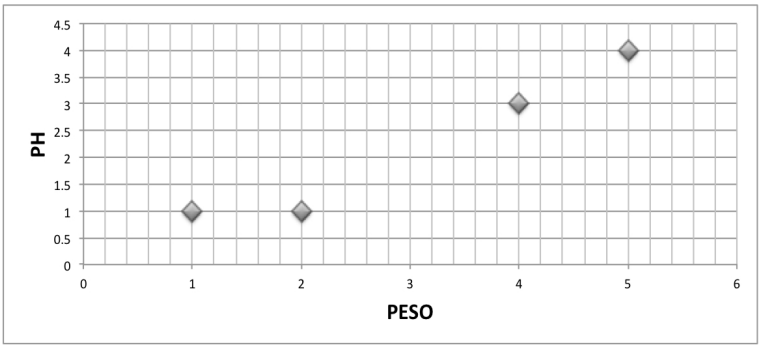


Figura 2.10: **Tipos de Medicina representados como espacio coordenado**. Adaptado de **Teknomo, 2007**.

Valor inicial de los centros: Sean $C1$ y $C2$ los centros de los K-grupos Si se

eligen las medicinas A y B como los primeros centros que quedan en los puntos $(1, 1)$ y $(2, 1)$.

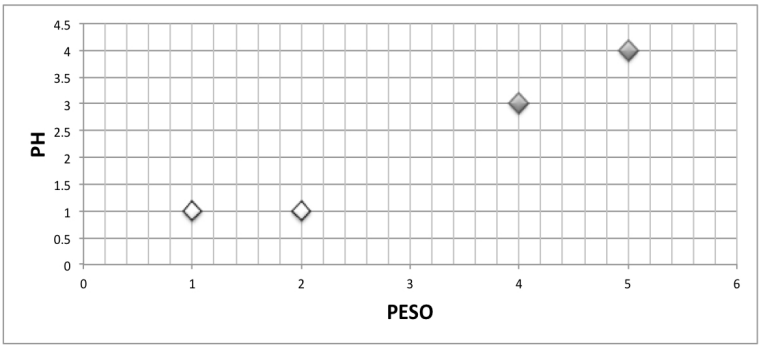


Figura 2.11: **Medicinas A y B tomadas como centros como espacio coordenado**. Adaptado de **Teknomo, 2007**.

Distancia de cada objeto a los centroide: Se calcula la distancia de cada objeto utilizando la distancia euclidiana:

$$D(i, j) = \left(\sum_{k=1}^{n} (X_{i,k} - X_{j,k}) \right)^{1/2} \tag{2.1}$$

en donde:

i,j: representan dos puntos n-dimensionales.

Así por ejemplo la distancia entre la medicina C $(4, 3)$ y el primer centro $(1, 1)$ es $\sqrt{(4-1)^2 + (3-1)^2} = 3{,}61$, y su distancia con el segundo centro $(2, 1)$ es $\sqrt{(4-2)^2 + (3-1)^2} = 2{,}83$. Resultado en la primera iteración la siguiente matriz de distancias:

$$D^0 = \begin{bmatrix} 0{,}00 & 1{,}00 & 3{,}61 & 5{,}00 \\ 1{,}00 & 0{,}00 & 2{,}83 & 4{,}24 \end{bmatrix}$$

Agrupación: Se asigna entonces cada objeto a cada grupo teniendo en cuenta el mínimo error de partición de cada elemento en cada grupo, entonces la medicina A se asigna al grupo 1 (con centor en $(1, 1)$) y las medicinas B, C y D al grupo 2 (con centro en $(2, 1)$).

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Iteración 1, determinar centros: Conociendo los miembros de los grupos, se vuelve a calcular el centro de cada grupo y la matriz de distancias. Como el grupo 1 tiene sólo un miembro el centro de ese grupo permanece en (1,1). El grupo 2 tiene ahora 3 miembros, por lo que el nuevo centro es:

$c_2 = \left(\frac{2+4+5}{2}, \frac{1+3+4}{3}\right) = \left(\frac{11}{3}, \frac{8}{3}\right)$

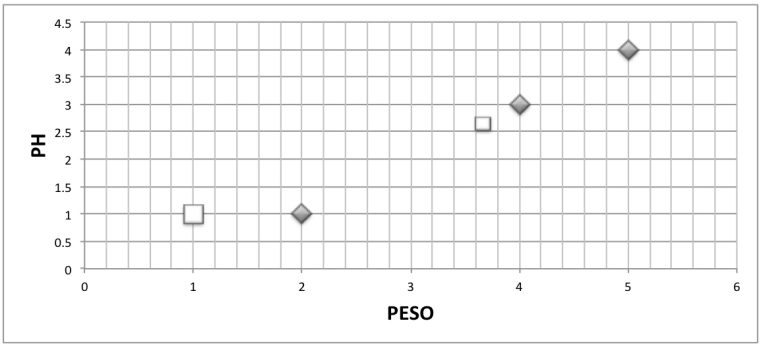


Figura 2.12: **Nuevo centros para los dos grupos**. Adaptado de **Teknomo, 2007**.

Iteración 1, Distancia de los objetos a los centros: Nuevamente se calcula las distancias para todos los objetos con base en los nuevos centroides, similar al paso 2. Obteniendo la siguiente matriz de distancia

$$D^0 = \begin{bmatrix} 0{,}00 & 1{,}00 & 3{,}61 & 5{,}00 \\ 3{,}14 & 2{,}36 & 0{,}47 & 1{,}89 \end{bmatrix}$$

Iteración 1, Agrupación de Objetos: Similar al paso 3, se asigna nuevamente cada objeto a cada grupo teniendo en cuenta el mínimo error de partición de cada elemento en cada grupo, entonces ahora las medicinas A y B quedan asignadas al grupo 1 (con centro $(1, 1)$) y las medicinas C y D al grupo 2 (con centro en $(11/3, 8/3)$).

$$G^0 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Iteración 2, determinación de Centros: Repitiendo el paso 4, se recalculan los centros:

$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = \left(1\frac{1}{2}, 1\right)$; $c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = \left(4\frac{1}{2}, 3\frac{1}{2}\right)$

Iteración 2, Distancia de los objetos a los Centros: Repitiendo el paso 2, se calcula la matriz de distancias:

$$D^0 = \begin{bmatrix} 0{,}50 & 0{,}50 & 3{,}20 & 4{,}61 \\ 4{,}30 & 3{,}54 & 0{,}71 & 0{,}71 \end{bmatrix}$$

Iteración 2, Agrupación de Objetos: Nuevamente se asigna los objetos a cada grupo basados en el criterio de mínima distancia:

$$G^0 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

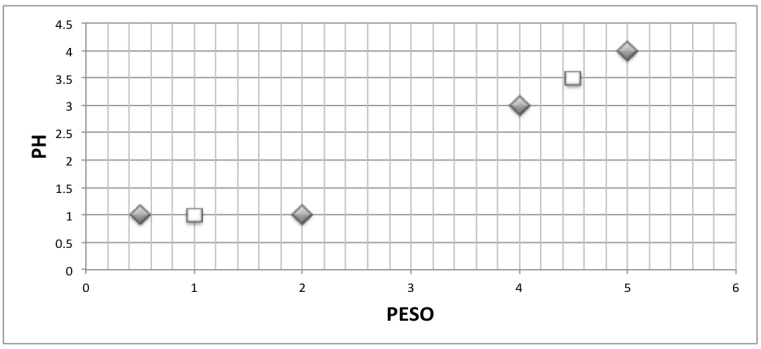


Figura 2.13: **Calculando nuevos centros en la segunda iteración**. Adaptado de **Teknomo, 2007**.

Los casos de entrada tendrán bajo esta nueva iteración la misma clasificación, concluyéndose que el cálculo de los centros se ha estabilizado, y no es necesario continuar iterando.

2.5.3.3. Clasificación

La clasificación tiene como meta asignar un nuevo elemento de dato a una de varias clases categóricas predefinidas. Ya que el campo a predecir es pre-etiquetado, la clasificación también es conocida como **aprendizaje supervizado**. Hay varios métodos de clasificación como son: redes neuronales, algoritmos genéticos y árboles de decisión. Este último es particularrnente usado para minería de datos, ya que puede ser construído relativamente rápido, y es muy simple y fácil de entender **[Han and Kamber, 2006]**.

Un árbol de decisión es construído usando un enfoque de particionamiento recursivo. Cada nodo interno en el árbol representa una decisión de un atributo, la cuál divide a la base de datos en dos o más hijos. Inicialmente la raiz contiene la base entera, con muestras de clases mezcladas. El punto de corte elegido es el mejor que separa o discrimina las clases. Cada nuevo nodo es recursivamente dividido de la misma manera hasta que el nodo sólo contenga una o más clases mayoritarias **[Maimon and Rokach, 2005]**.

Para ilustrar como es usado un árbol de decisión, se usará el siguiente ejemplo, en el cual un árbol de decisión es inducido usando el algoritmo ID3, y a partir de este las reglas de decisión que permiten clasificar nuevos elementos de datos son establecidas. Previo al ejemplo, se describirá como trabaja el algoritmo ID3 **[Quinlan, 1987]**.

Algoritmo ID3. En el algoritmo ID3, cada nodo corresponde a un atributo de corte y cada ramificación es un posible valor de ese atributo. En cada nodo, el atributo de corte seleccionado es el que posee más información entre los atributos

que aun no son considerados en la ruta de la raíz. El algoritmo usa el criterio de ganancia de información para determinar la calidad de un corte. El atributo con la ganancia de información más grande es tomado como el atributo de corte, y el conjunto de datos es dividido por todos los distintos valores de el atributo de corte. Para calcular la ganancia de información, previamente se tiene que calcular la entropía la cual provee información teórica para medir la calidad de un corte. Esta entropía mide la cantidad de información en un atributo **[Jain, 2001]** **[Quinlan, 1987]**. Para calcularla se emplea la siguiente ecuación **[WinterSchool, 2011]**:

$$Entropia(S) = \sum_{I=1}^{C}(-p(I)log_2 p(I)) \tag{2.2}$$

en donde:

S: Representa el conjunto de datos con C resultados.
P(I): Representa la proporción de S que pertenece a una clase I en donde I varía de 1 a C para el problema de clasificación con C clases.

La ganancia de información de un conjunto de muestras S para un atributo A, es definido como **[WinterSchool, 2011]**:

$$Ganancia(S, A) = Entropia(S) - \sum \left(\left(\frac{|S_V|}{|S|} \right) * Entropia(S_V) \right) \tag{2.3}$$

en donde:

la suma $\sum$ es para cada valor de V de todos los posibles valores de el atributo A,
S_V es el subconjunto de S para el cual los atributos de A tienen valor V,
$|S_V|$ es el número de elementos en S_V, y
$|S|$ es el número de elementos de S.

Ejemplo [WinterSchool, 2011]:

El problema del clima. El problema del clima trata de un pequeño conjunto de datos que es enteramente ficticio, el cual supuestamente se refiere a las condiciones que son adecuadas para jugar algún juego no especificado. Los atributos del conjunto de datos son {Pronóstico, Temperatura, Humedad, Viento} y el atributo de decisión es **Jugar** dependiendo el tiempo o **No Jugar**. En su simple forma, los cuatro atributos tienen valores categóricos. Los valores de los atributos Pronóstico, Temperatura, Humedad y Viento son {soleado, nublado, lluvioso}, {caluroso, templado , frío}, {alta, normal} y {débil, fuerte} respectivamente. Este conjunto de datos es mostrado en la **tabla 2.4**.

ID	Pronóstico	Temperatura	Humedad	Viento	**Jugar**
X1	soleado	caluroso	alta	débil	no
X2	soleado	caluroso	alta	fuerte	no
X3	nublado	caluroso	alta	débil	si
X4	lluvioso	templado	alta	débil	si
X5	lluvioso	frío	normal	débil	si
X6	lluvioso	frío	normal	fuerte	no
X7	nublado	frío	normal	fuerte	si
X8	soleado	templado	alta	débil	no
X9	soleado	frío	normal	débil	si
X10	lluvioso	templado	normal	débil	si
X11	soleado	templado	normal	fuerte	si
X12	nublado	templado	alta	fuerte	si
X13	nublado	caluroso	normal	débil	si
X14	lluvioso	templado	alta	fuerte	no

Tabla 2.4: **Conjunto de Datos del Clima**. Basado en **WinterSchool, 2011** y **Witten and Frank, 2005**

Los valores de todos los cuatro atributos producen 3*3*2*2 = 36 posibles combinaciones de las cuales 14 son presentadas como salidas.

Para poder inducir un **árbol de decisión** para la **tabla 2.4** se va usar el algoritmo ID3 **[Quinlan, 1987]**. Debe ser notado que S es una colección de 14 muestras con 9 sí's y 5 no's. Por lo tanto la entropía es la siguiente **[WinterSchool, 2011]**:

Entropia(S) = -(9/14)*log_2(9/14)-(5/14)*log_2(5/14) = 0.940

Hay 4 atributos condicionales en la tabla. Con el fin de encontrar que atributo puede servir como nodo raíz en el árbol de decisión a ser inducido, la ganancia de información es calculada de acuerdo a todos los 4 atributos. Para el atributo Viento, hay 8 ocurrencias de viento débil y 6 ocurrencias de viento fuerte. Para Viento = débil, 6 resultados de los 8 son sí y los dos restantes son no. Para Viento = fuerte, 3 resultados de los 6 son sí y el resto son no. De este modo, usando la ecuación 2.3 **[WinterSchool, 2011]**:

Ganancia(S, Viento) = Entropia(S) - (8/14) * Entropia(debil) - (6/14) * Entropia(fuerte)
= 0.940 - (8/14)(0.811) - (6/14) (1.0)
= 0.048
Entropia(debil) = -(6/8)*log_2(6/8)-(2/8)*log_2(2/8) = 0.811
Entropia(fuerte) = -(3/6)*log_2(3/6)-(3/6)*log_2(3/6) = 1.0

Similarmente:
Ganancia(S, Pronostico) = 0.246

Ganancia(S, Temperatura) = 0.029
Ganancia(S, Humedad) = 0.151

Como el Pronóstico tiene mayor ganancia, este atributo va a ser usado como el nodo raíz. Ya que el Pronóstico tiene tres posibles valores, el nodo raíz tiene tres ramificaciones (soleado, nublado, lluvioso). La siguiente pregunta ahora es, "¿cuál del resto de los atributos deberia ser probado en la rama del nodo soleado?"

En la **tabla 2.4** hay 5 muestras con valor Pronostico = soleado. De este modo usando la **ecuación 2.3 [WinterSchool, 2011]**:

Ganancia(Ssoleado, Humedad) = 0.9.70
Ganancia(Ssoleado, Temperatura) = 0.570
Ganancia(Ssoleado, Viento) = 0.0.19

Los calculos anteriores muestran que el atributo Humedad tiene la más alta ganancia, de este modo, debería de ser usado como el siguiente nodo de decisión para la rama soleado. Este proceso es repetido hasta que todos los datos son clasificados perfectamente o ningún atributo es hoja para los nodos hijo más abajo en el árbol. El árbol de decisión final obtenido usando ID3 se muestran en la **figura 2.14**.

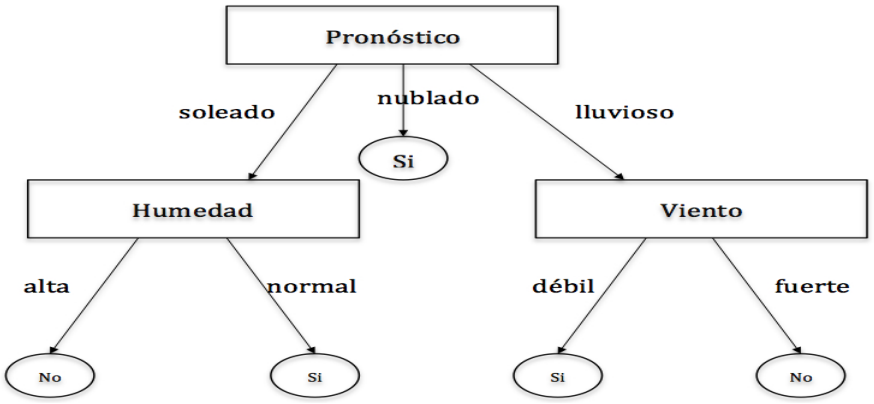


Figura 2.14: **Clasificador ID3 para el conjunto de datos del clima**. Adaptado de **Witten and Frank, 2005**.

Las correspondientes reglas de decisión son:

1. Si **pronostico** = **soleado** y **humedad** = **alto** entonces **jugar** = **no**.
2. Si **pronostico** = **soleado** y **humedad** = **normal** entonces **jugar** = **si**.
3. Si **pronostico** = **nublado** entonces **jugar** = **si**.
4. Si **pronostico** = **nublado** y **viento** = **debil** entonces **jugar** = **si**.
5. Si **pronostico** = **nublado** y **viento** = **fuerte** entonces **jugar** = **no**.

2.5.3.4. Regresión Lineal

En el método de regresión lineal se dispone de una muestra de observaciones formadas por pares de variables: $(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)$. A partir de esta muestra se desea estudiar la relación existente entre las variables **X** y **Y**. Es posible representar éstas observaciones mediante un gráfico de dispersión, en el cual se muestra el grado de asociación de los datos mediante indicadores **[Han and Kamber, 2006]**.

En el método de regresión lineal se busca encontrar relaciones entre las variables en torno a la **causa(Y)** y el **efecto(X)**. Para ello se establece un modelo:

$$Y_i = a + bX_i + e_i \tag{2.4}$$

para $i = 1, 2 \ldots, n$

en la que a y b son dos cantidades fijas (parámetros del modelo) y los e_i son cantidades aleatorias que representan las diferencias entre lo que postula el modelo $a + bx$, y lo que realmente se observa, y. Por esa razón se llama a los e "errores" o"errores aleatorios". Se asume que tienen valor esperado 0 y desviación estándar común σ.

Para mostrar como funciona el método se verá el siguiente ejemplo, en donde se usa regresión lineal simple **[UAZ, 2012]**:

En una muestra de 1,500 individuos se recogen datos sobre dos medidas antropométricas X y Y. Los resultados se muestran resumidos en los siguientes estadísticos:

$$\begin{array}{ll} \bar{x} = 14 & S_X = 2 \\ & S_{XY} = 45 \\ \bar{y} = 100 & S_Y = 2 \end{array}$$

Se desea obtener el modelo de regresión lineal que mejor aproxima Y en función de X. Utilizando este modelo, debemos calcular de modo aproximado la cantidad Y esperada cuando $X = 15$.

Solución. Lo que se busca es la recta $Y = a + b \cdot X$, que mejor aproxima los valores de Y (según el criterio de los mínimos cuadrado) en la nube de puntos que representa de representar en un plano (X, Y) las 1,5000 observaciones. Los coeficientes de esta recta son:

$$b = \frac{S_{XY}}{S_X^2} = \frac{45}{4} = 11{,}25.$$
$$a = \bar{y} - b \cdot \bar{x} = 100 - 11{,}25 \times 14 = -57{,}5.$$

As, el modelo lineal consiste en:

$$Y = -57{,}5 + 11{,}25 \cdot X$$

Por lo tanto, si $x = 15$, el modelo lineal predice un valor de Y de:

$$Y = -57{,}5 + 11{,}25 \cdot X = -57{,}5 + 11{,}25 \times 15 = 111{,}25$$

2.6. Minería de Datos Distribuida

Un enfoque tradicional de la minería de datos, es que los datos son analizados de manera central. En este enfoque centralizado los datos que se encuentran ubicados en distintos puntos geográficos son recolectados en alguna computadora central y posteriormente se les extrae conocimiento. Esta computadora central puede ser un datawarehouse en el cual los datos son depurados, unificados y analizados.

Sin embargo esta forma clásica de obtener conocimiento, se ha enfrentado a una serie de dificultades que han probado que este enfoque ha llegado a ser ineficaz ó poco factible. Entre estas dificultades se encuentran las siguientes **[Tsoumakas and Vlahavas, 2008]** :

- **Costo de almacenamiento**. Los requerimientos de un sistema de almacenamiento central son enormes. Muchas bases de datos almacenan información que escala en el orden del exabyte (10^{18} bytes) y esta se va incrementando a gran velocidad.
- **Costo de comunicación**. El transferir grandes volúmenes de datos en una red toma demasiado tiempo y requiere de un inaguantable costo financiero. Incluso un pequeño volúmen de datos crearía problemas en entornos de red con ancho de banda limitado.
- **Costo de computacional**. El costo computacional en MD centralizada es mucho más alto que analizar pequeñas partes de los datos que podrían también ser analizadas en paralelo.
- **Datos privados y sensitivos**. Existen muchas aplicaciones para minería de datos que tratan con datos sensitivos tales como personas médicas y registros financieros. El recolectar estos datos de forma central no es deseable debido a la privacidad de esta información. En ciertos casos los datos pertenecen a diferentes organizaciones que compiten, las cuales desean intercambiar información pero no sus datos.

Para enfrentar estas dificultades en los últimos años se han diseñado nuevos enfoques que buscan responder de mejor manera a la necesidad de minar una enorme cantidad de datos. Uno de estos nuevos enfoques es conocido como **Minería de**

Datos Distribuida (DDM) [Kargupta, Park, Hershberger, and Johnson, 1999]. La DDM es el resultado de la evolución de la minería de datos en donde se incorporan conceptos de Sistemas Distribuidos, tratando de forma especial datos que se encuentran remotamente localizados. El minar datos de forma distribuida ha llegado a ser una forma más efectiva y eficiente para analizar conocimiento. En está sección se describirá de manera más detallada como funciona un sistema de **Minería de Datos Distribuida**.

2.6.1. Principios básicos de la minería de datos distribuida

En DDM se busca aplicar los procedimientos clásicos de minería de datos en un entorno de computadoras distribuido tratando de mejorar la disponibilidad de recursos (redes de comunicación, unidades de cómputo y bases de datos).

En una típica arquitectura de DDM se llevan a cabo una serie de etapas básicas para extraer conocimiento. En primera instancia se hace un análisis de la base de datos local en cada sitio distribuido. Después la información (en forma de conocimiento) obtenida de cada sitio es transmitida a otro sitio, en donde los modelos locales distribuidos son integrados para formar un sólo modelo global. Una vez obtenido este modelo global, los resultado son retransmitidos a las bases de datos distribuidas para que actualicen su respectivos modelos locales. En algunos enfoques, en vez de integrar los modelos locales, cada modelo local es retransmitido a los otros sitios, para que cada sitio pueda procesar en paralelo el modelo global **[Tsoumakas and Vlahavas, 2008]**. La representación de está arquitectura es mostrada en la **figura 2.15**.

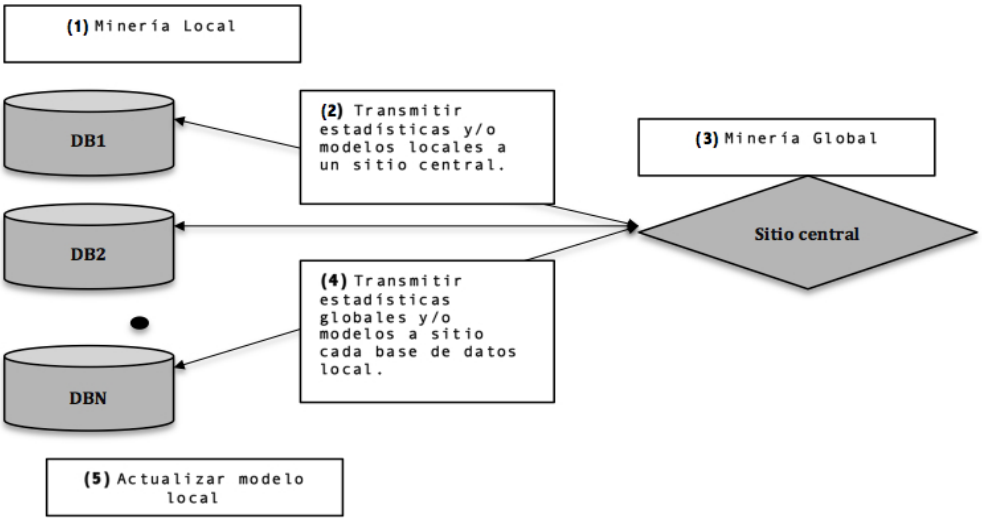


Figura 2.15: **Arquitectura típica en un enfoque de Minería de Datos Distribuida**. Adaptado de **Tsoumakas and Vlahavas, 2008**.

Las bases de datos distribuidas (DDB) pueden tener esquemas homogéneos y heterogéneos. En el primer caso, los atributos que describen a los datos son los mismos en cada DDB. Esto es frequente en base de datos que pertenecen a la misma organización (por ejemplo, almacenes locales de una cadena). En el último esquema los atributos difieren entre las bases de datos distribuidas. Por lo que las aplicaciones tienen un atributo clave presente en las bases de datos heterogéneas, la cual permitirá que las tuplas se puedan reunir **[Tsoumakas and Vlahavas, 2008]**.

La DDM ofrece una serie de estrategias que pueden ser usadas de acuerdo a la distribución de los datos, los recursos de hardware y software que este a disposición y la precisión requerida. De acuerdo a los sistemas DDM estas estratégias son las siguientes **[Jiang-Lingxia, 2009]**:

(1) **Estratégia de Datos.** En DDM se puede escoger como resultado final mover datos, mover los resultados, proveer un modelo de predicción, o mover algoritmos de minería de datos. Se puede usar un sistema DDM de *Aprendizaje Local* para establecer modelos en cada sitio distribuido, y transmitir estos modelo a una región central. Pero también se puede usar como un sistema de *Aprendizaje Centralizado* en donde los datos son recolectados en un región central y se establece un modelo. Además algunos sistemas de minería de datos pueden usar *Aprendizaje Híbrido*, esto quiere decir que un un *Aprendizaje Local* y un *Aprendizaje Centralizado* pueden ser combinados.

(2) **Estratégia de Tarea.** En un sistema DDM se puede escoger utilizar coordinadamente un tipo de algoritmo de minería de datos en varios sitios de datos, o poder usar diferentes algoritmos e minería de datos independientemente en cada sitio. En el modo de *Aprendizaje Independiente*, cada tipo de algoritmo de minería de datos es respectivamente aplicado en cada sitio de datos distribuido; en el modo de *Aprendizaje Coordinado*, un (o más) sitio(s) de dato(s) usan un algoritmo de minería de datos para coordinar tareas de minería de datos en varios sitios de datos.

(3) **Estratégia de Modelo.** Existen muchos métodos para combinar modelos de predicción establecidos en diferentes sitios. Entre estos métodos, el más simple y frecuentemente usado es de **voteo**, en el cual por mayoria de votos se elige un modelo de salida de todos los sitios.

Un sistema DDM debe tener buen desempeño, por ello debe poder responder a los siguientes puntos **[Jiang-Lingxia, 2009]**:

- **Escalabilidad**. El sistema debe de reaccionar y adaptarse sin perder calidad al flujo continuo de datos que cambia con el tiempo.
- **Eficiencia**. Hacer uso de los recursos de sistemas centralizados efectivamente y obtener los resultados de minería correctos.

- **Portabilidad**. Un sistema DDM debe poder operar de forma normal en múltiples entornos con equipamientos de software y hardware, y deben poder combinar múltiples modelos con diferentes expresiones.

- **Adaptatividad**. Un sistema DDM tiene la habilidad para evolucionar y ajustarse a los cambios del entorno.

- **Extensibilidad**. Un sistema DDM debe tener la suficiente flexibilidad para adaptarse a la tecnología de minería de datos presente y futura; sino el será obsoleto.

2.6.2. Ventajas de la Minería de Datos Distribuida

Como ya se ha mencionado la DDM se caracteriza por la integración de modelos de aprendizaje distribuidos. Esta integración puede ser hecha aprendiendo de los resultados de múltiples bases de aprendizaje procesadas. De este modo la DDM puede ser relacionada con dos importantes campos de investigación en máquinas de aprendizaje: *comité de aprendizaje*, en donde múltiples modelos de aprendizaje son aprendidos para mejorar la precisión de las predicciones evitando que existan varios sesgos de aprendizaje; y el *meta aprendizaje* en donde los modelos son re-aprendidos e integrados. La combinación de estos dos enfoques de aprendizaje dan a la DDM las siguientes ventajas **[Guo and Sutiwaraphun, 1999]**:

- **Precisión de Aprendizaje.** Usar diferentes algoritmos de aprendizaje para aprender modelos de fuentes de datos distribuidas incrementa la posibilidad de lograr alta precisión en los resultados sobre todo en dominios con gran cantidad de datos. Esto se debe a la integración de estos modelos que pueden representar diferentes aprendizajes sesgados los cuales se compensan unos a otros sus deficiencias.

- **Tiempo de Ejecución y Limitación de Memoria.** DDM proporciona una solución para una gran escala de datos en donde la complejidad de los algoritmos y las limitaciones de memoria siempre representan un obstaculo. Si existen múltiples computadoras interconectadas, cada procesador puede trabajar con una partición diferente de los datos con el fin de independientemente derivar un modelo. Algunos gastos de recursos menores pueden ocurrir en este proceso, pero en general existe un beneficio ya que una técnica de combinación puede formar un modelo limpio derivado de otros modelos distribuidos.

2.6.3. Resultados de Investigación en Minería de Datos Distribuida

La DDM ha recibido considerable atención en la literatura en años recientes. El trabajo en está area puede ser visto desde tres puntos de vista: *algoritmos de minería de datos distribuida*, *arquitecturas y sistemas para minería de datos distribuida* y *aplicaciones de minería de datos distribuida*. Estos puntos serán discutidos brevemente a continuación.

2.6.3.1. Algoritmos de Minería de Datos Distribuida

Muchos de los enfoques para DDM vienen del deseo de diseñar algoritmos que escalen bien grandes conjuntos de datos. Pero conceptualmente hay que hacer una diferencia entre los enfoques para DDM en los que los algoritmos escalan grandes cantidades de datos, distribuyendolos con el fin de mejorar la eficiencia en general, y los enfoques que asumen que los datos son inherentemente distribuidos y autónomos, por lo que ciertas restricciones necesitan ser tomadas en cuenta. En los siguientes algoritmos que a continuación se mencionan se tiene a consideración está segunda categoría **[Caragea, 2004]**:

- **Minería de Datos Paralela.** Los primeros trabajos en DDM aparecieron como una necesidad de escalar grandes cantidades de datos, pero entre otros enfoques para este problema también surgio la *minería de datos paralela* en la cual se proponen métodos para distribuir un gran conjunto de datos centralizados en múltiples procesadores que paralelamente procesarán los datos para acelerar el aprendizaje **[Srivastava, Han, and adn V. Singh, 1999]**.

- **Minería de Datos Distribuida enfocada en Ensamble.** Muchos algoritmos de DDM tienen sus raices en métodos de ensamble. Estos métodos se enfocan en conjuntar clasificadores que aprenden de fragmentos de datos horizontales distribuidos, en los cuales estan involucrados clasificadores separados de cada conjunto de datos y que son combinados típicamente usando esquemas de voteo de pesos. En general, esta combinación requiere recolectar un subconjunto de datos de cada una de las fuentes de datos en un sitio central para determinar los pesos que son asignados a las hipótesis individuales (ó alternativamente enviar los conjuntos de clasificadores y pesos asociados a las fuentes de datos donde estos pueden ser ejecutados sobre datos locales para establecer los pesos). Algunos inconvenientes de este algoritmo están en la necesidad de tener que transmitir subconjuntos de datos a un sitio central y en la falta de garantías concernientes a la precisión de las hipótesis relativas a las obtenidas en el enfoque centralizado **[Dietterich, 2000]**.

- **Minería de Datos Distribuida basada en Cooperación.** Aunque el aprendizaje con escenarios de cooperación pueden ser muy frecuentes en situaciones del mundo real, no existen muchos algoritmos que usen la cooperación en una manera activa para obtener un resultado final. En estos algoritmos lo que se busca es estimar reglas locales que puedan ser tan satisfactorias como las reglas globales, la cooperación en este sentido viene del intercambio de estos modelos (reglas) entre todos los sitios distribuidos. También existe otro tipo de colaboración en donde los datos pueden ser movidos de un sitio a otro con el fin de explotar los recursos de red **[Provost and Hennessy, 1996]**.

- **Aprendizaje de Datos Distribuidos Verticalmente.** Aunque muchos algoritmos de DDM asumen que los datos tiene fragmentación horizontal, existen algunas excepciones. Un ejemplo de esto es el sistema WoRLD ("Worldwide Relational Learning Daemon") el cual representa un enfoque colaborativo para el aprendizaje de datos frgmentados verticalmente. Este sistema trabaja calculando la distribución cardinal de los valores característicos de los conjuntos de datos individuales, y propaga está distribución através de los diferentes sitios de la red. Valores característicos con correlaciones fuertes a el concepto a ser aprendido son identificados basados en la primera aproximación estadística de primer orden a la distribución cardinal. Al estar basado en aproximaciones de primer orden, este enfoque puede llegar a ser poco práctico para problemas en donde estadísticas de alto orden son necesarias. En otros enfoque se conjuntan clasificadores locales usando técnicas basadas en un orden estadístico para combinar modelos con alta varianza generados de sitios heterogéneos **[Aronis, Kolluri, Provost, and Buchanan, 1996]**.

- **Aprendizaje Relacional.** Las tareas de aprendizaje de datos relacionales han recibido atención significativa en los últimos años. Uno de los primeros enfoques para el *aprendizaje relacional* estuvo basado en la Programación Lógica Inductiva (ILP). La ILP es un campo amplio el cual viene como resultado del desarrollo de algoritmos para la síntesis de programas lógicos de ejemplos y conocimiento previo para el desarrollo de algoritmos de clasificación, regresión, clustering, y análisis de asociación. Debido a su flexible y expresiva manera para representar conocimiento previo y ejemplos, este campo no sólo considera representaciones de tablas individuales de los datos, sino también representaciones de múltiples tablas, lo cual lo hace un buen candidato para *aprendizaje relacional*. Sin embargo las técnicas de ILP son limitadas en su capacidad para trabajar con bases de datos relacionales. Esfuerzos se han hecho para relacionar las técnicas de ILP con bases de datos relacionales, entre estos esfuerzos están el explotar el lenguaje de consulta estructurada (SQL) para recolectar la información necesaria para construir clasificadores (por ejemplo, árboles de decisión) de múltiples datos

relacionales **[Muggleton, 1992]**.

- **Minería de Datos Distribuida para preservar privacidad.** Algunos enfoques para DDM aparecieron de la necesidad de preservar privacidad de la información que es minada. En tal caso una recopilación de los datos necesarios era usada en vez de usar todo el conjunto de datos. Algunas herramientas pueden ser usadas para aprender de los datos mientras se preserva la privacidad **[Lindell and Pinkas, 2002]**.

2.6.3.2. Arquitecturas y Sistemas para Minería de Datos Distribuida

En los últimos años la ingeniería de software orientada a agentes **[Jennings and Wooldridge, 2000]** a ha ofrecido una atractiva implementación modular y extensible de sistemas de cómputo distribuido. Cada sitio tiene uno o más agentes asociados que procesan datos locales y comunican los resultados a los otros agentes o a un supervisor central que controla el comportamientos de los agentes locales. Algunos sistemas de agentes en Java implementan técnicas de *meta aprendizaje* tal como el sistema Java Agents for Meta-Learning **[Stolfo, 1997]** ó sistemas de minería de datos basada en agentes distribuidos para minería de datos colectiva como es el caso del sistema BODHI **[Kargupta, Park, Hershberger, and Johnson, 1999]**, o existen otros sistemas como PADMA **[Kargupta, Hamzaoglu, and Stafford, 1997]** el cual es una herramienta para el análisis de documentos que trabaja en un entorno distribuido basado en agentes cooperativos.

Otro enfoque para direccionar DDM escalable está basada en un cluster de estaciones de trabajo de alto rendimiento conectadas en red. Papyrus **[Grossman, Bailey, Ramu, Malhi, and Turinsky, 2000]** es un sistema para minar fuentes de datos distribuidos en un cluster local o un cluster de área amplia y en un escenario de super cluster. Este sistema es diseñado para encontrar estrategias óptimas para mover resultados, modelos o datos, sobre la red.

Un enfoque diferente a estos esta basado en un entorno de componentes distribuidos. Un ejemplo de este entorno es la arquitectura Kensington en donde los componentes son localizado en diferentes nodos en una red genérica como Intranet o Internet. Kensington **[Chattratichat, Darlington, Guo, Hedvall, Koler, and Syed, 1999]** es dividido en un cliente el cual provee la creación interactiva de datos de tareas de minería de datos; y una aplicación servidor la cual es responsable de coordinar los datos y administrar los datos. En un tercer nivel de servidores se provee servicios de minería de datos de alto rendimiento.

La minería de datos distribuida ha evolucionado hacia el aprovechamiento del paradigma de aplicaciones que proveen servicios, las cuales permiten a pequeñas organizaciones accesar a software en demanda. En una arquitectura de este tipo se trata de demostrar que la DDM puede ser integrada a una aplicación de servicios

en un entorno de comercio electrónico. Su propuesta esta basada en integrar tecnologías de agente y tecnologías cliente-servidor.

2.6.3.3. Aplicaciones del Mundo Real para Minería de Datos Distribuida

Los algoritmos de minería de datos distribuida pueden ser aplicados en varios dominios del mundo real, tales como: una red de detección de intrusos **[Kumar, 2003]**, detección de fraudes de crédito **[Chan, Fan, Prodromidis, and Stolfo, 1999]**, clasificación de textos **[Kuengkrai and Jaruskulchai, 2002]**, transacciones cortas de las bases de datos de cadenas de almacen **[Lin, Lee, Chen, and Yu, 2002]**, datos geocientíficos **[Shek, Muntz, Mesrobian, and Ng, 1996]**, dispositivos móviles **[Kargupta, Park, Pittie, LIu, Kushraj, and Sarkar, 2002]**, bases de datos distribuidas basadas en redes de sensores **[Bonnet, Gehrke, and Seshadri, 2001]**, análisis de diagnósticos cardíacos **[Wirth, Borth, and Hipp, 2001]** entre otras aplicaciones.

Capítulo 3

Análisis de cluster

Una actividad básica de los seres humanos es la de clasificar objetos que existen en su entorno y asociarlos en ciertas clases. Este proceso de clasificación se vuelve necesario por ejemplo, para poder desarrollar un lenguaje, consistente de palabras que ayude a reconocer y discutir diferentes tipos de eventos, objetos y personas que encontrar. Desde el punto de vista científico la clasificación es también fundamental para estructurar conceptos y para el desarrollo de teorías dentro de este ámbito científico.

Al proceso de clasificación se le ha dado el nombre genérico de "análisis de cluster", en el cual una serie de procedimientos son usados para formar "clusters" o grupos de alta similaridad entre entidades **[Aldenderfer and Blashfield, 1984]**. Con el desarrollo de las computadoras de alta velocidad el análisis de cluster cobró más importancia, debido a que aquellos cálculos explícitos en este proceso, que antes se tenían que hacer manualmente, ahora se podían hacer computacionalmente con un menor consumo de tiempo. Y además conforme los avances computacionales se fueron dando fue posible generar algoritmos que pudieran procesar grandes conjuntos de datos, reduciendo así la complejidad computacional existente.

En los últimos 20 años una diversidad de métodos han sido propuestos para efectuar análisis de cluster. En este capítulo se profundizará en el análisis de cluster, los requerimientos necesarios para efectuar análisis de cluster en grandes conjuntos de datos y se analizarán varios métodos para efectuar este análisis. Además se mostrarán los avances existentes en el desarrollo del análisis de cluster distribuido.

3.1. Aprendizaje supervisado vs Aprendizaje no supervisado

En la terminología de máquinas de aprendizaje la clasificación puede ser llevada de dos maneras diferentes: un *aprendizaje supervisado* y un *aprendizaje no supervisado* **[Liu, 2011]**. En el **aprendizaje supervisado** se trata de hacer una inferencia de los datos, por medio de analizar un conjunto muestra de un conjunto de datos, el cual tiene etiquetas ya definidas, y del que se busca obtener un análisis que permita clasificar otras fuentes de datos con etiquetas faltantes, de ahí, el nombre de aprendizaje, ya que trata de predecir datos futuros de datos históricos pasados. En un **aprendizaje no supervisado** las muestras de un conjunto de datos no están etiquetadas, y a diferencia de los métodos supervisados en que se intenta hacer una inferencia, en el aprendizaje no supervisado se busca establecer modelos de un conjunto de datos, que permitan obtener conclusiones sobre las relaciones existentes de los datos que conforman el conjunto. A ésta forma de clasificación, también se le conoce como *clustering* ó *análisis de cluster*, debido a que un conjunto de datos puede ser agrupado en varias categorías de acuerdo a una medida de similaridad. La **sección 3.2** de este capítulo se enfocará en detallar el concepto de **análisis de cluster**, y la **sección 3.4** de explicar la variedad de algoritmos que existen para llevar a cabo este tipo de clasificación.

3.2. ¿Qué es el análisis de cluster?

El análisis de cluster es un nombre genérico que se le da a una variedad de métodos mátemáticos, que pueden ser usados para averiguar cuales objetos en un conjunto son similares. Estos métodos ordenan objetos descritos en forma de datos, de manera que aquellos que contienen descripciones similares son matemáticamente reunidos en un mismo cluster **[Romesburg, 1984]**. Si se analizaran un conjunto de piedras físicamente, y otra vez se analizaran usando el método matemático de análisis de cluster, se deberían obtener escencialmente los mismos clusters.

3.2.1. ¿Qué es un cluster?

Algunos autores intuitivamente han tratado de definir éste término desde el punto de vista de cohesión interna (*homogeneidad*) y aislación externa (*separación*) **[Cormack, 1971]**, **[Gordon, 1999]**. En este sentido el término cluster debe entenderse como un conjunto de observaciones que intentan cumplir con los conceptos de homogeneidad (los elementos en el mismo cluster deberían parecerse) y separación matemática (los elementos de diferentes clusters deberían diferir)

[Hansen and Jaumard, 1997]. Otros autores simplemente dejan a valoración del investigador el producir un término.

Ahora bien en cuanto a su estructura, tampoco es claro como reconocer cuando un cluster es desplegado en un plano, ya que este puede tener diversas formas como se muestra en la **figura 3.1**.

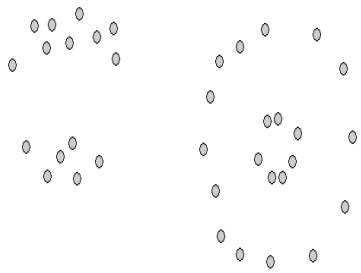


Figura 3.1: **Clusters con diversas formas**. Adaptado de **Everitt, Landau, Leese, and Stahl, 2011**.

Esto quiere decir, que no existe una estructura 'natural' de cluster, así que el criterio de evaluación más adecuado para reconocer un cluster es el de medir la distancia entre sus puntos y buscar que cumplan con cierta homogeneidad. Si no existe una estructura natural lo mejor sería considerar todo un conjunto de observaciones como una colección homogénea de puntos y entonces recurrir a métodos de análisis de clúster que divida esos datos en diferentes partes. Así que ahora la consideración sería que métodos usar para encontrar estas partes o clusters, en donde cada cluster contiene un subconjunto de todo el conjunto de observaciones.

3.2.2. Métodos numéricos de clasificación

Se han derivado numerosas técnicas de clasificación. Gran parte de éstas se deben a ciencias naturales como la biología, y la zoología en las cuales se trata de proveer una *taxonomía* en la que se provee una forma de clasificación *objetiva* y *estable* **[Sneath and Sokal, 1973]**. *Objetiva* en el sentido de que al analizar el mismo conjunto de organismos por medio de la misma secuencia de métodos numéricos estos produzcan la misma clasificación; *estable* significa que esta clasificación se mantiene para la amplia variedad de adiciones de organismos o de nuevas características que los describen.

A la varidedad de métodos numéricos existentes para clasificación se le ha dado variedad de nombres, según el campo de conocimiento que los aplica, por ejemplo, en biología se le llama *taxonomía numérica*, en psicología se emplea el término *análisis Q*. En inteligencia artificial se le conoce como *reconocimiento de patrones*

no supervisados y en investigación de mercado como *segmentación*. Hoy en día es más genérico referirlos como *análisis de cluster* en el cuál procedimientos son usados para descubrir grupos en los datos **[Everitt, Landau, Leese, and Stahl, 2011]**.

En muchas aplicaciones de análisis de cluster una *partición* de los datos es buscada, en la cual un elemento u objeto pertenece a un sólo cluster, y el conjuto de clusters contiene todos los elementos. Algunas ocasiones el traslapar clusters provee soluciones aceptables, esto es aceptable mientras una respuesta pueda ser justificada.

De los tipos de datos básicos para muchas aplicaciones de análisis de cluster está la usual matriz de datos $\mathbf{X}$, de dimensión $n \times p$, en donde cada valor contenido describe un objeto a ser agrupado. La estructura de esta matriz ésta en forma de tabla relacional (n *objetos* $\times$ p *variables*), como se representa en la **ecuación 3.1 [Everitt, Landau, Leese, and Stahl, 2011]**.

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \tag{3.1}$$

Tal matriz es frecuentemente referida como de 'dos - modos', indicando que las filas y las columnas corresponden a diferentes cosas **[Han and Kamber, 2006]**. Las variables en $\mathbf{X}$ frecuentemente son una mezcla de valores continuos, ordinales y/o categóricos, y frecuentemente algunas entradas estarán ausentes. Ésta mezcla de variables y valores ausentes en ocasiones puede complicar el clustering de los datos **[Everitt, Landau, Leese, and Stahl, 2011]**. En algunas aplicaciones, las filas de la matriz $\mathbf{X}$ pueden contener *medidas repetidas* de la misma *variable*, como por ejemplo, condiciones diferentes, o en diferentes tiempos, o en un número de posiciones espaciales, etc.

Algunas técnicas de análisis de cluster convierten la matriz $\mathbf{X}$ en una matriz de *similaridades*, *disimilaridades* o *distancias* (un término general es *proximidad*) inter-objetos. Esta es representada frecuentemente como una tabla $n \times n$, como se observa en la **ecuación 3.2 [Han and Kamber, 2006]**.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix} \tag{3.2}$$

en donde $d(i, j)$ es la medida de diferencia o disimilaridad entre objetos i y j. En general, $d(i, j)$ es un número no negativo que es muy cercano a 0 cuando los objetos i y j son altamente similares o "cercanas"la una de las otra, y llega a ser más grande si ellos difieren. Ya que $d(i, j) = d(j, i)$, y $d(i, i) = 0$, es por eso que se obtiene una matriz triángular.

Para poder medir estas disimilaridades entre objetos, éstos pueden ser descritos como variables de *intervalo escalado*, por variables *binarias*, por variables *categoricas*, *ordinales* o de *radio escalado*, o una combinación de todas éstas. Como se verá en las secciones siguientes.

3.2.2.1. Variables de intervalo escalado

Una **variable de intervalo escalado** es una medida continua de una escala más o menos lineal **[Han and Kamber, 2006]**. Ejemplo de éstas variables son valores como peso y altura, coordenadas de latitud y longitud, y la temperatura, las cuales están representadas en cierta unidad de medida. Según la unidad de medida seleccionada, ésta puede afectar el análisis de clustering. Por ejemplo, el análisis cambia si las unidades son expresadas en metros o pulgadas, o en medidas de peso como kilogramas o libras. En general, expresar una variable en una pequeña unidad de medida conducirá a afectar la variable en un amplio rango, y de este modo puede verse afectado el resultado del análisis de cluster. Para evitar esto, independientemente de la medida a que se seleccione, los datos deben ser **estandarizados**.

Para estandarizar medidas, una elección es convertir las medidas originales a variables sin unidad. Dada las medidas para una variable *f*, esto puede ser llevado como sigue **[Han and Kamber, 2006]**:

1. Calcular la **desviación media absoluta**, s_f:

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \cdots + |x_{nf} - m_f|), \tag{3.3}$$

   donde $x_{1f}, \ldots, x_{x_{nf}}$ son n medidas de f, y m_f es el valor *medio* de f, es decir: $mf = \frac{1}{n}(x_{1f} + x_{2f} + \cdots + x_{nf})$.

2. Calcular la **medida estandarizada**, o **z-score**:

$$z_{if} = \frac{x_{if} - m_f}{s_f} \tag{3.4}$$

En ocasiones para cierta aplicación la estandarización puede no ser útil. Por lo que en este caso la elección de estandarizar o no, queda a consideración del usuario.

Después de efectuar la estandarización, las disimilaridades entre los objetos son calculadas basadas en una medida de distancia para cada par de objetos. La medida de distancia más popular es la **distancia Euclidiana**, la cuál se calcula como es definido en la **ecuación 3.5 [Han and Kamber, 2006]**.

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2}, \tag{3.5}$$

donde $i = (x_{i1}, x_{i2}, \ldots, x_{in})$ y $j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ son objetos de datos n-dimensionales.

Otra métrica bien conocida es la **distancia de Manhattan**, la cual es definida en la **ecuación 3.6 [Han and Kamber, 2006]**.

$$d(i,j) = |x_{1f} - m_f| + |x_{2f} - m_f| + \cdots + |x_{nf} - m_f|, \tag{3.6}$$

Para ambas métricas los siguiente requerimiento matemáticos se tienen que cumplir:

- $d(i,j) \geq 0$: La distancia es un número no negativo.
- $d(i,j) = 0$: La distancia de un objeto a el mismo es 0.
- $d(i,j) = d(j,1)$: La distancia es una función simétrica.
- $d(i,j) \leq d(i,h) + d(h,j)$: Ir directamente de un objeto i a un objeto j en el espacio no es más que rodear sobre cualquier otro h (*desigualdad del triángulo*).

Otra medida de distancia es la de **Minkowski**, la cual es una generalización de la dos distancias anteriores. Ésta es definida en la **ecuación 3.7** como **[Han and Kamber, 2006]**:

$$d(i,j) = (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{in} - x_{jn}|^p)^{\frac{1}{p}}, \tag{3.7}$$

en donde p es un entero positivo ($p \geq 1$). Esta distancia también es conocida como L_p normal. Si $p = 1$ representa la `distancia de Manhattan` (i.e, L_1 normal) y si $p = 2$ la `distancia Euclidiana`. En caso de que p llegara a infinito, se obtendría una **distancia de Chebyshev [Bubak, van Albada, Dongarra, and Sloot, 2008]**:

$$\begin{aligned} \lim_{p\to+\infty} d(i,j) &= \lim_{p\to+\infty} \left((|x_{i1}-x_{j1}|^p + |x_{i2}-x_{j2}|^p + \cdots + |x_{in}-x_{jn}|^p)^{\frac{1}{p}} \right) \\ &= \text{máx}\left(|x_{i1}-x_{j1}| + |x_{i2}-x_{j2}| + \cdots + |x_{in}-x_{jn}|\right) \end{aligned} \tag{3.8}$$

Otra medida de interes es la **distancia del coseno**. La distancia del coseno es una medida de similaridad entre dos vectores, midiendo el `coseno` de los ángulos de ellos. Ésta distancia es definida en la **ecuación 3.9 [Manning and Schütze, 1999]** .

$$\cos[d(i,j)] = \frac{A \cdot B}{|A||B|} = \frac{(x_{i1}x_{j1}) + (x_{i2}x_{j2}) + \cdots + (x_{in}x_{jn})}{\sqrt{(x_{i1}^2) + (x_{i2}^2) + \cdots + (x_{in}^2)}\sqrt{(x_{j1}^2) + (x_{j2}^2) + \cdots + (x_{jn}^2)}} \tag{3.9}$$

Ejemplo. Calcular la distancia Euclidiana, de Manhattan y de Coseno. Sea $x_1 = (1,2)$ y $x_2 = (3,5)$ los cuales repressentan dos objetos. La *distancia euclidiana* entre los dos puntos es: $\sqrt{((1-3)^2 + (2-5)^2)} = \sqrt{(2^2+3^2)} = 3{,}61$. La *distancia de manhattan* entre los dos objetos es: $|1-3| + |2-5| = 2+3 = 5$. Para obtener la *distancia de minkowski* vamos a establecer $p = 3$. De este modo la distancia es: $\sqrt[3]{(|1-3|^3 + |2-5|^3)} = \sqrt[3]{(2^3+3^3)} = \sqrt[3]{(8+27)} = \sqrt[3]{(35)} = 3{,}27$. Finalmente la *distancia de cosenos* entre dos objetos es:

$$\frac{(1*3)+(2*5)}{\sqrt{(1^2)+(2^2)}\sqrt{(3^2)+(5^2)}} = \frac{3+10}{\sqrt{1+4}\sqrt{9+25}} = \frac{13}{\sqrt{5}\sqrt{34}} = 0{,}99$$

3.2.2.2. Variables binarias

Una **variable binaria** es aquella que sólo tiene dos estados: 0 o 1, donde el 0 representa que la variable está ausente y el 1 significa que esta está presente. Para medir la similaridad o disimilaridad entre objetos una variable binaria puede usarse para medir el grado de *simetría* o *asimetría* entre variables **[Han and Kamber, 2006]**.

Para calcular la disimilaridad entre dos variables binarias, se empezará por decir que a todas las variables binarias se les asignará un peso, el cual será el mismo para todas. Con ésto se tendrá una tabla de contingencia como se ve en la **tabla 3.1**. En donde q es el número de variables iguales a 1 para ambos objetos i y j, r es el número de variables que iguales a 1 para el objeto i pero 0 para el objeto j, s es el número de variables que son iguales a 0 para el objeto i pero iguales a 1 para el objeto j, y t es el número de variables que son iguales a 0 para ambos objetos i y j. El número total de variables es p, donde $p = q + r + s + t$.

	objeto j			
		1	0	sum
	1	q	r	q+r
objeto i	0	s	t	s
	sum	q+s	r+t	s

Tabla 3.1: **Tabla de Contingencia para valores binarios**. Adaptada de **Han and Kamber, 2006**.

Se dice que una variable binaria es **simétrica** si dos estados son igualmente valiosos y llevan el mismo peso; esto es, no hay preferencia en cual resultado debería ser codificado como 0 o 1 **[Han and Kamber, 2006]**. Un ejemplo de esto sería el atributo *genero*, en el cual hay dos estados *masculino* o *femenino*. La medida de disimilaridad basada en variables `binarias simétricas` es llamada **disimilaridad binaria simétrica** y esta puede ser usada para evaluar la disimilaridad entre objetos i y j. Para calcularla se emplea la **ecuación 3.10** **[Sokal and Michener, 1958]**.

$$d(i,j)^{1} = \frac{r+s}{q+r+s+t} \tag{3.10}$$

Por otro lado una variable binaria es **asimétrica** si los resultados de dos estados no son igual de importantes, ejemplo de ello serían los resultados *positivo* o *negativo* de una prueba de enfermedad, en donde los resultdos positivos sería considerados más significantes que los negativos **[Han and Kamber, 2006]**. La disimilaridad basada en variables `binarias asimétricas` es llamada **disimilaridad binarias asimétrica**, en donde los resultados de t son considerados poco importantes y por lo tanto se ignora en el cálculo de la disimilaridad. Ésta se calcula empleando la **ecuación 3.11** **[Kaufman and Rousseeuw, 1990]**.

$$d(i,j) = \frac{r+s}{q+r+s} \tag{3.11}$$

Complementario a esto se puede medir la distancia entre dos variables binarias basados en la noción de *similaridad* en vez de la *disimilaridad*. De este modo, para medir la **similaridad binaria asimétrica** entre dos objetos i y j, o $sim(i,j)$, se efectua el siguiente cálculo, empleando la **ecuación 3.12** **[Jaccard, 1901]**:

[1] Éste es el *coeficiente de emparejamiento simple* el cual es ambos estados llevan la misma información (simetría). Este coeficiente fue introducido en la taxonomia numérica en **[Sokal and Michener, 1958]**.

$$sim(i,j) = \frac{q}{q+r+s} = 1 - d(i,j). \tag{3.12}$$

éste coeficiente $sim(i,j)$ es llamado el **coeficiente de Jaccard**[2]

Ejemplo. Suponga que la **tabla 3.2** contiene los registros de un paciente con los atributos *nombre*, *sexo*, *fiebre*, *tos*, *examen-1*, *examen-2*, *examen-3* y *examen-4*. En donde el *nombre* es un objeto identificador, *sexo* es un atributo simétrico, y el resto de los atributos son atributos asimétricos binarios.

nombre	sexo	fiebre	tos	examen-1	examen-2	examen-3	examen-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	Y	N	N	N	N
...	...	...	...	...	...	...	...

Tabla 3.2: **Tabla relacional en donde los pacientes son descritos por atributos binarios**. Adaptado de **Han and Kamber, 2006**.

Para valores de atributo asimétricos, sea los valores **Y** (sí) y **P** (positivo) igual a 1, y el valor **N** (no o negativo) igual a 0. Suponga que la distancia entre los objetos (pacientes) es calculada basada sólo en las variables asimétricas. De acuerdo a la **ecuación 3.10**, la distancia de cada par de los tres pacientes, Jack, Mary y Jim es:

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0{,}33$$
$$d(Jack, Mary) = \frac{1+1}{1+1+1} = 0{,}67$$
$$d(Jack, Mary) = \frac{1+2}{1+1+2} = 0{,}75$$

Estas medidas sugieren que Mary y Jim probablemente no tengan una enfermedad similar porque no tienen el más alto valor de disimilaridad entre los tres pares. De los tres pacientes, Jack y Mary son los que más probablemente tengan una enfermedad similar.

3.2.2.3. Variables categóricas

Una **variable categórica** es una generalización de la variable binaria en la que sólo se pueden tomar dos estados, pero en el caso categórico se le puede asignar

[2]Las ecuaciones 3.11 y 3.12 representan expresiones análogas del coeffiente de Jaccard introducido por Paul Jaccard en 1901, al que llamó originalmente *coeficiente de comunidad* **[Jaccard, 1901]**.

más estados a una variable **[Han and Kamber, 2006]**. Por ejemplo, si tenemos la variable *color_ map*, a ésta se le puede asignar 5 estados: *rojo*, *amarillo*, *verde*, *rosa*, *azul*.

Sea M el número de estados de una variable categórica. Los estados pueden ser denotados por letras, símbolos, o un conjunto de enteros, tales como $1, 2, \ldots, M$. Tales enteros sólo son usados para manejar datos y no representan cualquier orden específico.

Para medir la disimilaridad entre dos objetos i y j se cálcula de acuerdo a la **ecuación 3.13** basada en la razón de los desemparejamientos **[Sokal and Michener, 1958]**:

$$d(i, j) = \frac{p - m}{p} \tag{3.13}$$

en donde m es el número de emparejamientos (es decir, el número de variables para las cuáles i y j están en el mismo estado), y p es el número total de variables. Pesos pueden ser asignados para incrementar el efecto de m o para asignar mayor peso a los emparejamientos en variables que tienen un gran número de estados.

Ejemplo. Suponga que se tiene los datos de muestra de la **tabla 3.3**, excepto que sólo el *objeto identificador* y la variable (o atributo) *examen-1* están disponibles, en donde *examen-1* es una variable categórica.

objeto identificador	prueba-1 (categorica)	prueba-2 (ordinal)	prueba-3 (razón-escalado)
1	código-A	excelente	445
2	código-B	razonable	22
3	código-C	bueno	164
4	código-A	excelente	1,210

Tabla 3.3: Una tabla de datos de muestra con una mezcla de variables de diferente tipo. Adaptado de **Han and Kamber, 2006**.

Para la variable categórica *prueba-1* se va a calcular la matriz de disimilaridad de acuerdo a la **matriz 3.2**:

Ya que sólo hay una variable categórica (*prueba-1*) de acuerdo a la **ecuación 3.13**, $p = 1$ y para $d(i, j)$ evaluamos con 0 si los objetos i y j se iguales, y 1 si los objetos son diferentes. De este modo tenemos:

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

3.2.2.4. Variables ordinales

Una **variable ordinal** es similar a una variable categórica. La diferencia entre éstas es que hay un claro ordenamiento de los estados de las variables **[Han and Kamber, 2006]**. Por ejemplo, suponga que se tiene una variable *estatus económico*, la cuál tiene tres categorías (`bajo`, `medio`, `alto`), así que para clasificar personas en éstas tres categorías, se pueden ordenar como bajo, medio o alto, estatus económico. Otro ejemplo sería la variable *profesores* la cual tiene tres categorías ordenadas como: (`asistente`, `asociado`, `tiempo completo`). Una variable ordinal puede ser **discreta** si los valores de los M estados de la variable pueden ser arreglados en algún orden o sucesión, pero las diferencias entre éstos valores no pueden ser determinados **[Han and Kamber, 2006]**, como por ejemplo la variable *estatus económico* o la variable *profesor*; y es **continua**, si los valores de los estados son formados de mediciones continuas, que no se pueden asegurar si están en una escala lineal, la única información que se tiene es el orden de los estados **[Han and Kamber, 2006]**, ejemplo de ello, sería la variable *puntuación* cuyo rango de valores está entre 0 y 20, la cual es continua, sin embargo, los valores pueden ser agrupados como: `Distinción` (15-20), `Acreditable` (9-14), `Paso` (4-8) y `Fallo` (0-3), para formar la variable ordinal.

Ahora bien, para calcular las disimilaridades entre objetos. Se establece lo siguiente **[Han and Kamber, 2006]**:

La variable ordinal f tiene M_f estados, estos estados están ordenados en un rango de $1, \ldots, M_f$. Suponga que f describe n objetos. Para calcular las disimilaridad con respecto a f se siguen los siguientes pasos **[Kaufman and Rousseeuw, 1990]**:

1. El valor de f para el i-ésimo objeto es x_{if}, y f tiene M_f estados ordenados, representando el rango $1, \ldots, M_f$. Se reemplaza cada objeto x_{if} a su correspondiente rango, $r_{if} \in \{1, \ldots, M_f\}$.

2. Como cada variable ordinal puede tener un número diferente de estados, es frecuentemente necesario mapear el rango de cada variable en [0.0,1.0] y que cada variable tenga igual peso. Esto puede ser logrado reemplazando el rango r_{if} de cada i-ésimo objeto en la f-ésima variable, por:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{3.14}$$

3. La disimilaridad puede ser calculada usando cualquier medida de distancia, usando z_{if} para representar el valor f para el i-ésimo objeto.

Ejemplo. Disimilaridad entre variables ordinales. Suponga que se tiene la muestra de datos de la **tabla 3.3**, excepto que sólo esta vez el *objeto-identificador* y la variable continua ordinal *prueba-2* están disponibles. Hay tres estados para *prueba-2*, nombrados *razonable*, *bueno* y *excelente*, por lo que $M_f = 3$. Para el paso 1, si se reemplaza cada valor para *prueba-2* por su rango, los cuatro objetos son asignados a los rangos 3,1,2, y 3, respectivamente. El paso 2 normaliza los rangos mapeando el rango 1 como 0.0, el rango 2 como 0.5, y el rango 3 como 1.0. Para el paso 3, se usará la distancia Euclidiana representada en la **ecuación 3.5**, la cual resulta en la siguiente matriz de disimilaridad:

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 0{,}5 & 0{,}5 & 0 & \\ 0 & 1{,}0 & 0{,}5 & 0 \end{bmatrix}$$

3.2.2.5. Variables de razón-escalada

Una variable de radio-escalado hace una medida positiva de una escala no lineal, aproximada a una escala exponencial Ae^{Bt} o Ae^{-Bt} en donde A y B son constantes positivas, y t típicamente representa el tiempo, ejemplos de esta variable incluye medidas del crecimiento de una población de bacterias o el decaimiento de un elemento radiactivo **[Han and Kamber, 2006]**.

Para calcular la disimilaridad entre dos objetos, se pueden emplear tres métodos **[Kaufman and Rousseeuw, 1990]**:

- Tratar las variables de razón escalado como variables de intervalo escalado. Esto no es usualmente una buena opción ya que es probable que la escala pueda ser distorsionada.
- Aplicar **transformación logaritmica** a la variable de razón escalada f teniendo el valor x_{if} para el objeto i, usando la fórmula $y_{if} = log(x_{if})$. Los valores y_{if} pueden ser tratados como intervalos-valorados. Se debe hacer notar que para algunas variables de razón-escalada, el logaritmo u otras transformaciones pueden ser aplicadas, dependiendo de la definición y a la aplicación.
- Trata x_{if} como datos ordinales continuos y tratar sus rangos como intervalos-valorados.

Ejemplo. Disimilaridad entre variables de Razón-Escalada. Esta vez, se tiene la muestra de datos de la **tabla 3.3**, excepto que sólo el *objeto-identificador* y la

variable de razón-escalada *prueba-3*, están disponibles. Aplicando una transformación logarítmica, los resultados del *log* de los valores de *prueba-3* son: 2.65, 1.34, 2.21, y 3.08, los cuales se asignan a los objetos 1 al 4, respectivamente. Usando la distancia Euclidiana representada en la **ecuación 3.5** sobre los valores transformados, se obtiene la siguiente matriz de disimilaridad:

$$\begin{bmatrix} 0 & & & \\ 1{,}31 & 0 & & \\ 0{,}44 & 0{,}87 & 0 & \\ 0{,}43 & 1{,}74 & 0{,}87 & 0 \end{bmatrix}$$

3.3. Requerimientos para el Análisis de Cluster

Han y Kamber dan una lista de los requerimientos que deben de cumplirse para poder efectuar clustering en minería de datos, algunos de estos requerimientos todavía representan desafíos para los algoritmos de clustering, los cuales se describen brevemente **[Han and Kamber, 2006]**:

- **Escalabilidad**: Los algoritmos deben trabajar bien para pequeños conjuntos de datos, pero deben de poder escalar bien para millones de objetos.
- **Habilidad para tratar con diferentes tipos de atributos**. Muchos algoritmos están diseñados para datos numéricos (basados en intervalo), pero se requiere que puedan aplicarse a otros tipos de datos, tales como, los binarios, los categóricos, y los ordinales, o una mezcla de estos tipos.
- **Descubrimiento de los clusters con forma arbitraria**. Muchos algoritmos de clustering pueden encontrar clusters con forma esférica, ya que estos algoritmos funcionan con medidas basadas en distancias. éstos algoritmos deben poder encontrar cluster con formas arbitrarias.
- **Requerimientos minimales para el dominio de conocimiento para determinar parámetros de entrada**: Los algoritmos pueden ser sensibles a los parámetros de entrada que dan los usuarios, éstos parámetros pueden ser difíciles de determinar, especialmente si los objetos de datos son de alta dimensión. Esto puede dificultar la calidad de los clusters.
- **Habilidad para tratar con datos ruidosos**: Muchas bases de datos contiene datos atípicos, faltantes, desconocidos, o erróneos. Los algoritmos pueden ser sensitivos a éstos datos, por lo que hay que saber como tratarlos, para evitar pobre calidad en los clusters.
- **Clustering incremental e insensitividad al orden de los registros**: Algunos algoritmos de clustering no pueden incorporar nuevos datos, así éstos son añadidos, un nuevo análisis debe ser empezado de cero.

- **Alta dimensionalidad**: Muchas bases de datos contienen datos con varios atributos o dimensiones, y los algoritmos que se usan son buenos para manejar datos con baja dimensionalidad, por lo que el desafío es manejar datos que contienen objetos de datos con alta dimensionalidad. Algunos algoritmos también son sensibles al orden de los datos, por lo que es importante que éstos puedan procesar bien los datos que entran.
- **Clustering basado en restricción**: Para muchos datos en el mundo real se necesita que los algoritmos que efectuan clustering, pueden trabajar con diferentes tipos de restricciones. Si los clusters satisfacen éstas restricciones se puede esperar un buen comportamiento del clustering.
- **Interpretabilidad y usabilidad**: Los resultados del clustering deben poderse interpretar, comprender y utilizar. Ésto es importante para determinar la calidad de los resultados.

3.4. Algoritmos de clustering

Es difícil poder establecer una clasificación clara para los algoritmos de clustering existentes. Ya que algunos algoritmos existentes pueden traslaparse, es decir, un algoritmo tiene características que entran en algún otro algoritmo de clustering. Pero a pesar de esto lo más práctico es tener una clasificación que permita organizar mejor los algoritmos existentes. En una clasificación básica los algoritmos están divididos en dos categorías **[Han and Kamber, 2006]**:

- **Algoritmos basados en partición**: En estos algoritmos lo que se trata es que dado un conjunto de datos con n objetos de datos, se puedan construir k particiones de los datos, en donde cada partición representa un cluster y $k \leq n$, como es representado en la **figura 3.2 [Han and Kamber, 2006]**.

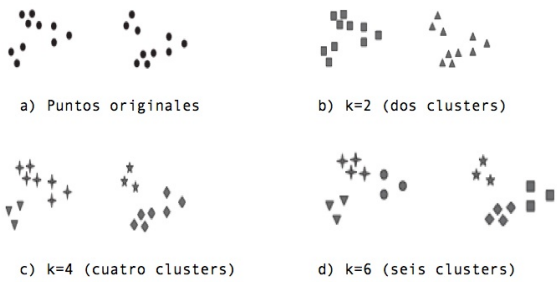


Figura 3.2: **Diferentes cluster del mismo conjunto de datos**. Adaptado de **Tang, Steinbach, and Kumar, 2006**.

Se dice que hubo un buen particionamiento si los objetos en un cluster son muy "cercanos", o están relacionados los unos con los otros, y a su vez

éstos objetos están "lejos" de los objetos que se encuentran en otros clusters. Para obtener un óptimo particionamiento se recurre a métodos como: **[1] k-medias** en donde cada cluster es representado por el valor medio de los objetos en el cluster **[MacQueen, 1967]**, y **[2]** el algoritmo **k-medoides** en donde cada cluster está representado por uno de los objetos localizados cerca del centro de los clusters **[Kaufman and Rousseeuw, 1987]**. Estos algoritmos serán más detallados en la **sección 3.4.1**.

- **Algoritmos Jerárquicos**: En éste algoritmo lo que se busca es organizar los n objetos de datos en un conjunto de clusters anidados, los cuales están organizados en forma de árbol. Cada nodo (cluster) en el árbol (excepto los nodos hoja) es la unión de sus hijos (subclusters), y la raíz de el árbol es el cluster que contiene todos los objetos, como es representado en la **figura 3.3 [Tang, Steinbach, and Kumar, 2006]**.

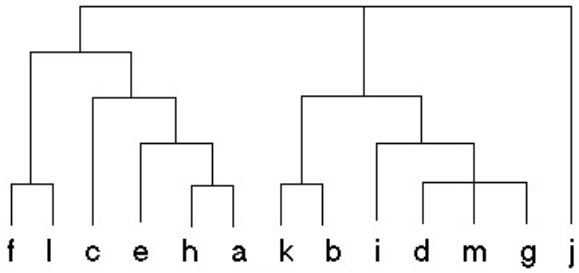


Figura 3.3: **Ejemplo de organización jerárquica**. Basado en **Hamilton, 2009**.

Para descomponer los objetos de datos hay dos formas de descomponerlos: *aglomerativamente* o *divisivamente*. En la forma **aglomerativa** se sigue un enfoque *bottom-up*, en la cada objeto representa un cluster, y sucesivamente se van mezclando los objetos o clusters que están más cercanos el uno con los otros, hasta que todos los clusters son mezclados en un solo cluster (el cual representa a la raíz del árbol en el nivel más alto de la jerarquía) **[Murtagh, 1984]**. En la forma **divisiva** se sigue un enfoque *top-down*, en donde todos los objetos están en el mismo cluster, y en cada sucesión, el cluster es dividido en clusters más pequeños, hasta que todos los objetos estén en un sólo cluster **[Kaufman and Rousseeuw, 1990]**. En ambas formas también se puede establecer una condición de terminación.

En una clasificación más avanzada de estos algoritmos, se incluyen otras dos categorias **[Han and Kamber, 2006]**:

- **Algoritmos basados en densidad**: En éste algoritmo un cluster se forma a través de una región densa de objetos, que está rodeado de una región de *densidad* (número de objetos o puntos). Un cluster puede crecer si su

densidad es amplificada, en un lenguaje más general se dice que su "vecindario" es de un radio dado, que contiene al menos un número de puntos. Éste algoritmo es representado en la **figura 3.4**.

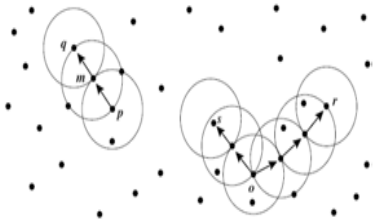


Figura 3.4: **Ejemplo de método basado en densidad**. Basado en **Han and Kamber, 2006**.

Éste método se puede usar para filtar ruido y descubrir formas arbitrarias de clusters. Un algoritmo para calcular éste tipo de método de clustering es **DBSCAN** en el cual los cluster representan regiones de baja densidad y el número de clusters es determinado por el algoritmo. Datos clasificados como ruido son omitidos **[Ester, Kriegel, Sander, and Xu, 1996]**.

- **Algoritmos basados en rejillas**: En éstos algoritmos los objetos son cuantizados (es decir, el espacio característico de los objetos es dividido en un número finito de celdas rectángulares) en un número de celdas que forma una estructura cuadriculada **[Han and Kamber, 2006]**. Todas las operaciones de clustering son llevadas bajo esa estructura. La principal ventaja con este enfoque es que el tiempo de procesamiento es más rápido, independientemente del número de objetos y depende sólo del número de celdas en cada dimensión del espacio cuatizado. Éste algoritmo es representado en la **figura 3.5**.

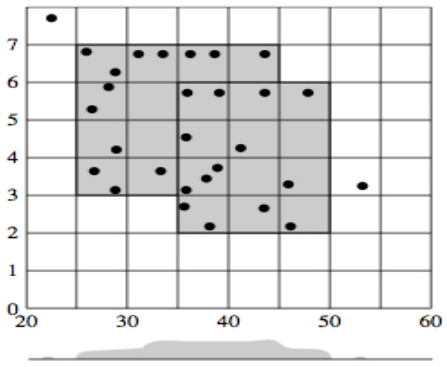


Figura 3.5: **Ejemplo de método basado en rejillas**. Basado en **Han, Kamber, and Pei, 2012**.

En las siguientes secciones se verán más a detalle estos métodos, y se analizarán los diversos algoritmos más significativos que existen dentro de cada método.

3.4.1. Algoritmos de Particionamiento

Los métodos de particionamiento trabajan construyendo una partición de k números de clusters, de un conjunto de datos D que contiene n objetos, en donde $k \leq n$. El objetivo es que dado el valor de k, se encuentre una partición de k clusters que optimice una función objetivo, este criterio puede ser alguna función de disimilaridad basada en distancia, en la que cada objeto detro de un cluster es "similar", y "disimilares" a objetos en clusters diferentes **[Han and Kamber, 2006]**.

Los algoritmos más conocidos para particionamiento son el de *k-medias* y el de *k-medoides*. Los cuales se describen a continuación.

3.4.1.1. k-Medias

El algoritmo de **k-Medias** funciona de la siguiente manera:

Dado un parámetro de entrada k y un conjunto de n objetos de datos, aleatoriamente se seleccionan k objetos del conjunto de n objetos, los cuales van a representar los puntos medios o centros de cada cluster. Después de esto, los puntos restantes van a ser asignados a cada uno de estos centros (cada punto es asignado al centro más cercano); y entonces el centro de cada cluster va a ser actualizado o recalculado en base a los puntos asignados a cada cluster. Los pasos de asignación y actualización se repetirán hasta que ninguno de los puntos en los clusters cambie, o los centros permanezcan iguales. k-Medias es formalmente descrito en el **algoritmo 3.1 [Han, Kamber, and Pei, 2012]**.

Algoritmo 3.1 Algoritmo de k-Medias. Basado en **Tang, Steinbach, and Kumar, 2006**.

1: Seleccionar k puntos como los centros iniciales.
2: **repeat**
3: Formar k clusters, asignando cada punto a su centro más cercano.
4: Recalcular los centros de cada cluster.
5: **until** Los centros no cambian.

Para asignar los puntos a los centros más cercanos, se usa una medida de proximidad para determinar que tan "cercanos" son los datos a los centros. La medida de proximidad más usada es la distancia Euclidiana, pero se pueden usar otras medidas de proximidad como lo son la distancia de Manhattan o la del Coseno, está última suele ser muy empleada para medir similaridad entre documentos.

Para garantizar que cada punto sea asignado a su centro más cercano y que la calidad del clustering sea bueno se usa una función objetivo que intentan garantizar la más mínima proximidad entre los puntos y los centros, ésta función objetivo

es la suma del error cuadrático, la cual es definida como **[Han, Kamber, and Pei, 2012]**:

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, c_i)^2, \tag{3.15}$$

en donde E representa la suma del error cuadrático para todos los objetos en el conjunto de datos; p representa un punto en el espacio el cual representa un objeto dado; c_i es centro del cluster C_i, y $dist$ es una medida estandar, generalmente la distancia Euclidiana entre dos objetos en un espacio Euclidiano. Para calcular el punto medio c_i, se usa la formula del valor medio **[Tang, Steinbach, and Kumar, 2006]**:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} p, \tag{3.16}$$

en donde m_i representa el número de objetos en el cluster i.

El algoritmo de k-Medias es relativamente escalable y eficiente ya que pueden procesar grandes conjuntos de datos, y computacionalmente solo requiere una complejidad de $O(nkt)$, en donde n es el número total de objetos de datos, k es el número de clusters, y t es el número de iteraciones. De este modo es lineal, y k es normalmente significativamente menor a n. ($k \ll n$) **[Han and Kamber, 2006]**.

3.4.1.2. k-Medoides

Una desventaja del algoritmo de k-Medias es que es sensible a los datos atípicos o con ruido, los cuales pueden distorcionar significativamente el resultado del clustering debido al uso de la función de error cuadrático. Una solución para disminuir esta sensibilidad es, que en vez de obtener los valores medios de los objetos, como puntos de referencia para cada cluster, puntos que representan objetos significativos sean seleccionados para cada cluster. Estos puntos son conocidos como *medoides* o ejemplares.

Un **medoide** es formalmente definido como el objeto de un cluster, cuyo promedio de disimilaridad para todos los objetos en el cluster es minimal, es decir, es el punto localizado más al centro **[Kaufman and Rousseeuw, 1987]**.

Basado en esto, es como funciona el algoritmo de k-**Medoides**, el cual intenta minimizar una suma de las disimilaridades entre cada objeto de datos y su correspondiente medoide.

La función objetivo usada en este algoritmo es la siguiente **[Han, Kamber, and Pei, 2012]**:

$$E = \sum_{j=1}^{k} \sum_{p \in C_j} dist(p, o_j)^2, \tag{3.17}$$

en donde E representa la suma del error absoluto para todos los objetos en el conjunto de datos; p es el punto es un punto en el espacio que representa a un objeto dado en C_j, y o_j representa el objeto representativo de C_j, k-Medoides es formalmente descrito en el **algoritmo 3.2**.

Algoritmo 3.2 Algoritmo de k-Medoides. Basado en **Shyam, 2012**.

```
1: Seleccionar aleatoriamente k puntos como los medoides.
2: repeat
3:    Formar k clusters, asignando cada punto a su medoide más cercano.
4:    for cada medoide m do
5:       for cada punto p no medoide en el conjunto de datos do
6:          Intercambiar m y p y calcular el costo total de la configuración
7:       end for
8:    end for
9:    Seleccionar la configuración con el menor costo.
10: until los medoides no cambian.
```

El algoritmo de k-Medoides o también conocido como el algoritmo **PAM** (Partitioning Around Medoids), intenta garantizar la calidad de los resultados del clustering; un objeto o_j, siempre reemplazará a aquel objeto p que causa que la función de error sea más grande, de este modo se garantiza siempre el mejor conjunto de objetos para cada cluster, al final del algoritmo.

El algoritmo de k-Medoides es eficiente para pequeños conjuntos de datos, pero no escala bien para grandes conjuntos, además a diferencia del algoritmo de k-Medias, es un proceso computacional más intensivo, y requiere un tiempo de complejidad de $O(k(n-k)^2)$, debido al número de comparaciones por iteración, en donde n representa los objetos de datos, y k el número de clusters **[Han and Kamber, 2006]**.

Para intentar reducir esto, han surgido algunas variantes del algoritmo como por ejemplo: **CLARA** (Clustering LARge Application), en el cual en vez de tomar todo el conjunto de datos, se toma una pequeña porción de ellos. Dentro de esta pequeña porción de datos, los medoides son elegidos. Si la pequeña porción de los datos fue elegida aleatoriamente, se espera que los resultados sean similares ha como si se hubieran elegido de todo el conjunto entero. La complejidad para cada iteración será $O(ks^2 + k(n-k))$, en donde s es el tamaño de la muestra,

k el número de clusters, y n el número total de objetos. Para que CLARA sea efectivo depende del tamaño de la muestra **[Han, Kamber, and Pei, 2012]**.

3.4.2. Algoritmos Jerárquicos

En los métodos jerárquicos se produce una representación gráfica de los objetos de datos. La representación gráfica usada es una estructura de árbol llamada **dendograma**. Para construir esta gráfica se siguen dos tipos de enfoque: el enfoque *bottom-up* o conocido como **aglomerativo** y el enfoque *top-down* o más conocido como **divisivo**. Estos enfoques se describen brevemente a continuación:

- **Aglomerativo**: De inicio cada objeto de datos representa un cluster, y en cada paso, del algoritmo, se mezclan los pares más cercanos para formar nuevos clusters. Para obtener esta medida de cercanía se utiliza alguna medida de distancia **[Gan, Ma, and Wu, 2007]**.
- **Divisivo**: De inicio todos los objetos de datos están en un sólo cluster, y en cada paso del algoritmo este se va cortando, hasta formar clusters individuales, donde en cada cluster hay un objeto de datos **[Gan, Ma, and Wu, 2007]**.

Como se menciono al principio los resultados del algoritmo jerárquico se representan en forma de **dendograma**, como es representado en la **figura 3.6**.

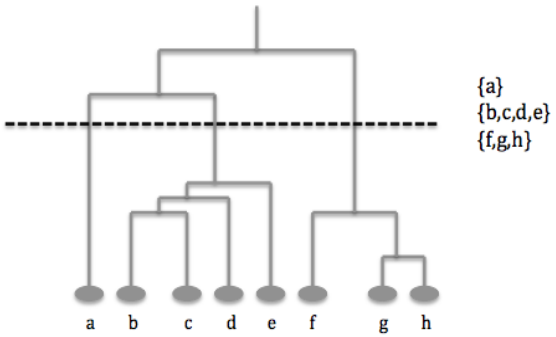


Figura 3.6: **Visualización de un dendograma**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

Un `dendograma` es un árbol binario con una raíz distinguida que tiene todos los objetos de datos en sus hojas **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**. Los dendogramas son una guía que describen el proceso de mezclar los clusters. Dependiendo del valor de distancia entre los objetos, se puede producir una secuencia de clusters anidados. Un ejemplo de esta secuencia se muestra en la **figura 3.7**.

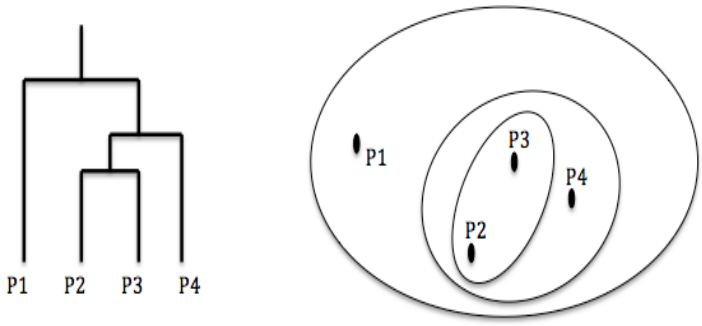


Figura 3.7: **Diagrama de clusters anidado y su respectivo dendograma**. Basado en **Tang, Steinbach, and Kumar, 2006**.

3.4.2.1. Algoritmo aglomerativo

Para n objetos de datos, los algoritmos aglomerativos empiezan con n clusters y cada cluster contiene un sólo objeto de datos. Después dos clusters se unen de acuerdo a su similaridad entre ellos, esto es, la cercanía entre ellos. En los pasos posteriores se siguen uniendo los clusters hasta llegar a uno o como lo especifico el usuario. El enfoque aglomerativo se define en el **algoritmo 3.3 [Johnson, 1967]**.

Algoritmo 3.3 Algoritmo **Aglomerativo**. Algoritmo formalmente llamado **AGNES** (AGlomerative NESting). Basado en **Tang, Steinbach, and Kumar, 2006**.

1: Empezar con n clusters, y un sólo objeto de datos indicado como un cluster.
2: Calcular la matriz de proximidad, si es necesario
3: **repeat**
4: Encontrar los clusters más similares c_i y c_j y entonces unirlos en un sólo cluster.
5: Actualizar la matriz de proximidad reflejando la proximidad entre los nuevos clusters y los clusters originales.
6: **until** Un sólo cluster permanezca.

3.4.2.2. Algoritmo divisivo

En el algoritmo divisivo se empieza por tener un sólo cluster que contiene todos los objetos de datos. Después, el único cluster se divide en dos o más clusters que tiene alta disimilaridad entre ellos hasta que el número de clusters llegan a ser el número de objetos de datos n representando cada uno un cluster o un número de clusters especificados por el usuario. Este enfoque divisivo se define en el **algoritmo 3.4 [Kaufman and Rousseeuw, 1987]**.

Algoritmo 3.4 Algoritmo **Divisivo**. Algoritmo formalmente llamado **DIANA** (DIvisive ANAlysis). Adaptado de **Kaufman and Rousseeuw, 1990**.

1: Empezar con un sólo cluster con los n objetos de datos.
2: **repeat**
3: **repeat**
4: Calcular el diámetro de cada cluster. El diámetro es la distancia máxima entre todos los objetos en el cluster. Escoger un cluster **C** que tenga el diámetro máximo de todos los clusters a cortar.
5: Encontrar el objeto x de mayor disimilaridad dentro de **C**. Sacar x del cluster original **C** para formar un nuevo cluster **N** (ahora el cluster no incluye el objeto x). Asignar todos los miembros de **C** a M_c[3].
6: Calcular las similaridades de cada miembro de M_C para el cluster **C** y **N**, y mover el miembro de M_C con más alta similaridad a su cluster similar **C** y **N**. Actualizar los miembro de los clusters **C** y **N**.
7: **until** los clusters **C** y **N** no cambian.
8: **until** el número de clusters sea igual al número de objetos de datos.

3.4.2.3. Proximidad entre clusters

Para calcular la proximidad o distancia entre los clusters, existen varios métodos para medir la distancia, los cuales se describens a continuación:

- **Método de un sólo enlace**: La distancia $d(A, B)$ es la distancia mínima entre los objetos que pertenecen a A y B. Ésta distancia es calculada como se muestra en la **ecuación 3.18** y se puede observar su representación en la **figura 3.8 [Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**.

$$d(A, B) = \min_{x \in A, y \in B} \mathbf{d(x,y)} \tag{3.18}$$

Figura 3.8: **Método de un sólo enlace**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

- **Método de enlace completo**: La distancia $d(A, B)$ es la distancia máxima entre los objetos que pertenecen a A y B. Ésta distancia es calculada como se muestra en la **ecuación 3.19** y se puede observar su representación en la **figura 3.9 [Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**.

[3] M_C representa la matriz de proximidad de los objetos en **C**

$$d(A, B) = \max_{x \in A, y \in B} \mathbf{d(x,y)} \tag{3.19}$$

Figura 3.9: Método de enlace completo. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

- **Grupo de enlace promedio**: La distancia $d(A, B)$ se calcula através del promedio entre las distancias calculadas entre cada par de objetos, con un objeto de cada cluster. Esta distancia es calculada de acuerdo a la **ecuación 3.20** y se puede observar su representación en la **figura 3.10** **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**.

$$d(A, B) = \frac{1}{card(A)card(B)} \sum_{x \in A, y \in B} \mathbf{d(x,y)} \tag{3.20}$$

Figura 3.10: **Método de grupo enlace promedio**. Adaptado de **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

Para ilustrar como se calculan estas distancias, se verán através del siguiente ejemplo, en el cual se tiene una muestra de 6 objetos de datos en dos dimensiones, representados en la **figura 3.11**. Las coordenadas x y y y las distancias entre los puntos se verán en la **tabla 3.4** y la **tabla 3.5**, respectivamente:

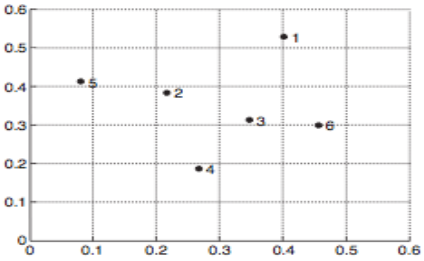


Figura 3.11: **6 objetos de datos en un conjunto en 2 dimensiones**. Basado en **Tang, Steinbach, and Kumar, 2006**.

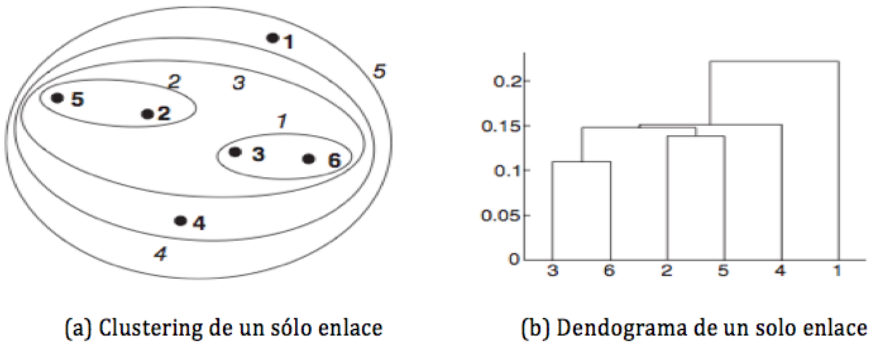
Punto	Coordenada x	Coordenada y
p_1	0.40	0.53
p_2	0.22	0.38
p_3	0.35	0.32
p_4	0.26	0.19
p_5	0.08	0.41
p_6	0.45	0.30

Tabla 3.4: **Coordenadas xy de los 6 puntos**. Basado en **Tang, Steinbach, and Kumar, 2006**.

	p_1	p_2	p_3	p_4	p_5	p_6
p_1	0.00	0.24	0.22	0.37	0.34	0.23
p_2	0.24	0.00	0.15	0.20	0.14	0.25
p_3	0.22	0.15	0.00	0.15	0.28	0.11
p_4	0.37	0.20	0.15	0.00	0.29	0.22
p_5	0.34	0.14	0.28	0.29	0.00	0.39
p_6	0.23	0.25	0.11	0.22	0.39	0.00

Tabla 3.5: **Matriz con la distancia Euclidiana para los 6 puntos**. Basado en **Tang, Steinbach, and Kumar, 2006**.

- Para el método de un sólo enlace, la distancia mínima entre dos puntos de dos clusters diferentes debe ser encontrada. En terminología de gráficas, si cada punto representa un cluster y se añaden enlaces entre los puntos uno a la vez, los primeros enlaces más corto, combinaran los puntos en grupos. El método de un sólo enlace es efectivo manejando formas no elípticas, pero es sensitivo al ruido y los datos atípicos **Tang, Steinbach, and Kumar, 2006**. Aplicando el método de un sólo enlace, los resultados son presentados en la **figura 3.12** en forma de dendograma y en una secuencia de clusters anidados.



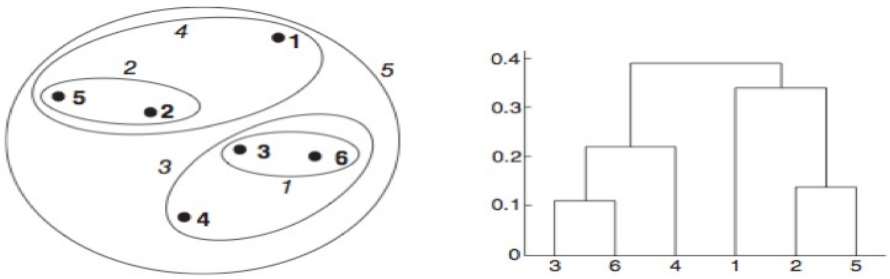
(a) Clustering de un sólo enlace (b) Dendograma de un solo enlace

Figura 3.12: Clustering de un sólo enlace para los seis puntos en la **figura 3.11**. Basado en **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**.

Si se quiere calcular las distancia mínima entre los clusters $\{3,6\}$ y $\{2,5\}$ se debe efectuar el siguiente cálculo:

$$\begin{aligned} \text{dist}(\{3,6\},\{2,5\}) &= min(\text{dist}(\{3,2\}), \text{dist}(\{6,2\}), \text{dist}(\{3,5\}), \text{dist}(\{6,5\})) \\ &= min(0{,}15, 0{,}25, 0{,}28, 0{,}39) \\ &= 0{,}15 \end{aligned}$$

- Para el método de enlace completo, la distancia máxima entre dos puntos de dos clusters diferentes se debe encontrar. En terminología de gráficas si cada punto representa un cluster y se añaden enlaces entre los puntos uno a lavez, los enlaces más cortos, agruparan puntos, pero no será un cluster hasta que todos los puntos no están completamente ligados, es decir, forma un clique[4]. El método de enlace completo es menos sensible a ruido y los datos atípicos, pero puede romper grandes clusters y favores formas globulares **[Tang, Steinbach, and Kumar, 2006]**. Aplicando el método de enlace completo, los resultados son presentados en la **figura 3.13** en forma de dendograma y en una secuencia de clusters anidados.



(a) Clustering de enlace completo (b) Dendograma de enlace completo

Figura 3.13: Clustering de enlace completo para los seis puntos en la **figura 3.11**.

A diferencia del método de sólo enlace en donde el cluster $\{3,6\}$ estaba unido con el cluster $\{2,5\}$ o $\{1\}$, esta unido ahora con cluster $\{4\}$ porque:

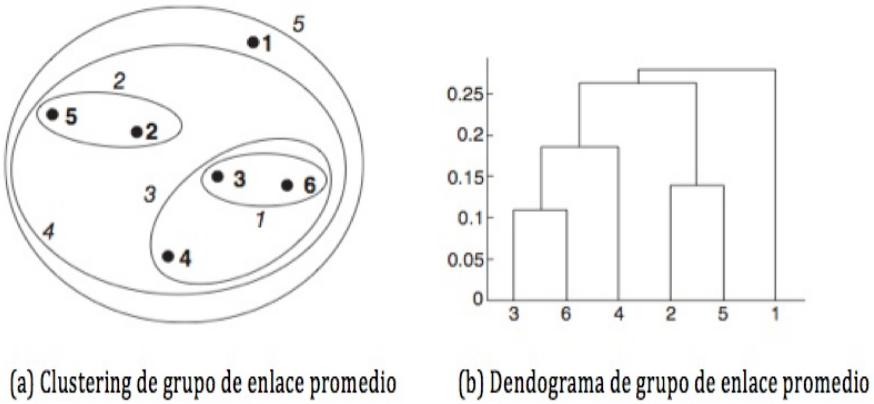
[4]Un clique en una gráfica no dirigida G es un conjunto de vértices V tal que para todo par de vértices, existe una arista que las conecta.

$$
\begin{aligned}
\text{dist}(\{3,6\},\{4\}) &= max(\text{dist}(\{3,4\}), \text{dist}(\{6,4\})) \\
&= min(0{,}15, 0{,}22) \\
&= 0{,}22. \\
\text{dist}(\{3,6\},\{2,5\}) &= max(\text{dist}(\{3,4\}), \text{dist}(\{6,2\}), \text{dist}(\{3,5\}), \text{dist}(\{6,5\})) \\
&= min(0{,}15, 0{,}25, 0{,}28, 0{,}39) \\
&= 0{,}39. \\
\text{dist}(\{3,6\},\{1\}) &= max(\text{dist}(\{3,1\}), \text{dist}(\{6,1\})) \\
&= min(0{,}22, 0{,}23) \\
&= 0{,}23.
\end{aligned}
$$

- Para el método de grupo de enlace promedio, la proximidad de dos clusters es definida como el promedio entre la proximidad entre todos los pares de los puntos en dos diferentes clusters. Este es un método intermedio entre los métodos de un sólo enlace y enlace completo. De este modo, para el grupo de enlace promedio, la proximidad del cluster $proximidad(C_i, C_j)$, es la proximidad de los cluster C_i y C_j con tamaños m_i y m_j respectivamente **[Tang, Steinbach, and Kumar, 2006]**, la cual es expresada en la **ecuación 3.21**.

$$proximidad(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} proximidad(x, y)}{m_i * m_j} \tag{3.21}$$

  Aplicando el método de enlace completo, los resultados son presentados en la **figura 3.14** forma de dendograma y en una secuencia de clusters anidados.



(a) Clustering de grupo de enlace promedio (b) Dendograma de grupo de enlace promedio

Figura 3.14: Clustering de un grupo de enlace promedio para los seis puntos en la **figura 3.11**.

Para calcular la distancia entre algunos clusters, se lleva a cabo de la siguiente manera:

$$\begin{aligned}
\mathsf{dist}(\{3,6,4\},\{1\}) &= (0{,}22 + 0{,}37 + 0{,}23)/(3 * 1) \\
&= 0{,}28 \\
\mathsf{dist}(\{2,5\},\{1\}) &= (0{,}2357 + 0{,}3421)/(2 * 1) \\
&= 0{,}2889 \\
\mathsf{dist}(\{3,6\},\{1\}) &= (0{,}15 + 0{,}28 + 0{,}25 + 0{,}39 + 0{,}20 + 0{,}29)/(6 * 2) \\
&= 0{,}26.
\end{aligned}$$

Como $\mathsf{dist}(\{3,6,4\},\{2,5\})$ es más pequeña que $\mathsf{dist}(\{3,4,6\},\{1\})$ y $\mathsf{dist}(\{2,5\},\{1\})$, los cluster $\{3, 4, 6\}$ y $\{2, 5\}$ son combinados.

3.4.3. Algoritmos basados en densidad

A diferencia de los métodos de particionamiento o jerárquicos los cuales encuentran clusters con forma esférica. Los métodos basados en densidad, encuentran clusters con formas arbitrarias a partir de modelar cada cluster como regiones densas de objetos en un espacio de datos, que están separados por regiones de baja densidad **[Swagatam, Ajith, and Amit, 2009]**. El propósito de estos métodos es encontrar regiones con alta densidad y baja densidad, en donde las regiones de alta densidad están separadas por regiones de baja densidad. El algoritmo más común que maneja este enfoque de densidad es DBSCAN.

Para poder entender bien como funciona este algoritmo, es necesario que previamente una serie de definiciones sean dadas para comprender el concepto de **densidad**, la cual es estimada a partir de un punto en un conjunto de datos D, contando el número de puntos dentro de un radio (comunmente definido como vecindad) ε, de ese punto. A partir de esto ciertas definiciones son dadas **[Tang, Steinbach, and Kumar, 2006]**:

1. **Puntos Centro**: Un punto es centro si existen un número de puntos *MinPts*, dentro de una vecindad ε alrededor de ese punto. Los valores *MinPts* y ε son valores definidos por el usuario, como parámetros de entrada.

2. **Puntos Bordes**: Un punto borde no es un punto centro pero cae en el borde de la vecindad de un punto centro. Un punto borde puede caer en la vecindad de varios puntos centro.

3. **Puntos ruido**: Un punto ruido es aquel que no es ni centro ni borde.

Otras definiciones importantes son las siguientes **[Han and Kamber, 2006]**:

1. **Densidad Directamente Alcanzable**: Un punto p es de *densidad directamente alcanzable* desde un punto q, si p está dentro del vecindario ε de q, y q es un punto centro.

2. **Densidad Alcanzable**: Un punto p es de *densidad alcanzable* desde un punto q con respecto a ε y $MinPts$ en un conjunto de puntos D, si existe una cadena de objetos $p_1, \ldots, p_n$, donde $p_1 = q$ y $p_n = p$ tal que p_{i+1} es de densidad directamente alcanzable desde p_i con respecto a ε y *MinPts*, para $1 \leq i \leq n$, $p_i \in D$.

3. **Densidad Conectada**: Un punto p es de *densidad conectada* desde un punto q con respecto ε y *MinPts* en un conjunto de puntos D, si hay un objeto $o \in D$, tal que p y q son de densidad alcanzable desde o con respecto a ε y *MinPts*.

3.4.3.1. DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise), busca clusters checando la vecindad ε de cada punto en D. Si la vecindad ε de un punto p contiene $MinPts$, un nuevo cluster con p como punto centro es creado. DBSCAN entonces iterativamente colecciona puntos de densidad directamente alcanzable. El proceso termina cuando ningún punto puede ser añadido a cualquier cluster **[Han and Kamber, 2006]**. DBSCAN es definido formalmente en el **Algoritmo 3.5**:

Algoritmo 3.5 Algoritmo **DBSCAN**. Basado en **Ester, Kriegel, Sander, and Xu, 1996**.

1: Arbitrariamente seleccionar un punto p.
2: Recuperar todos los puntos de densidad alcanzable desde p con respecto a ε y *MinPts*.
3: Si p es un punto centro, un cluster es formado.
4: Si p es un punto borde, no hay puntos de densidad alcanzable desde p y $DBSCAN$ visita el siguiente punto en D.
5: Si p es un punto ruido, este se descarta.
6: Continuar el proceso hasta que todos los puntos han sido procesados.

El tiempo de complejidad de DBSCAN es $O(m\times$ el tiempo para encontrar los puntos en la vecindad $\varepsilon)$, en donde m es el número de puntos. En el peor de los casos, la complejidad es $O(m^2)$. Sin embargo en espacios con baja dimesión existen estructuras de datos que permiten recuperar eficientemente estos puntos dentro de una distancia dada de un punto especificado, y el tiempo de complejidad puede ser reducido a *O(m log m)* **[Tang, Steinbach, and Kumar, 2006]**.

3.4.4. Algoritmos basados en rejillas

Los algoritmos basados en rejillas, dividen el espacio característico de los objetos de datos en un número finito de celdas rectángulares, formando una rejilla. Sobre esta estructura, todas las operaciones de clustering son llevadas acabo. La rejilla pueden estar formada de múltiples resoluciones, cambiando el tamaño de las celdas rectángulares. Una estructura de rejilla puede tener una jerarquía de varios niveles, la cual es aplicada al espacio característico de los objetos de datos en dos dimensiones. En el caso del espacio d-dimensional, hiper-rectángulos (cubos con forma rectángular) de d-dimensiones corresponden a las celdas **[Schikuta, 1996]**. En la estructura jerárquica de rejilla, el tamaño de celda en la rejilla puede decrementarse con el fin de logar más precisión en la estructura de celda **[Lepistö, Kunttu, Autio, and Visa, 2004]**. Como se ve en la **Figura 3.15** la estructura jerárquica puede ser dividida en varios niveles de resolución. Cada celda en un nivel k es particionada para formar un número de celdas en el siguiente nivel $k+1$. Las celdas en el nivel $k+1$ son pequeñas subceldas formadas por la división de la celda del nivel k.

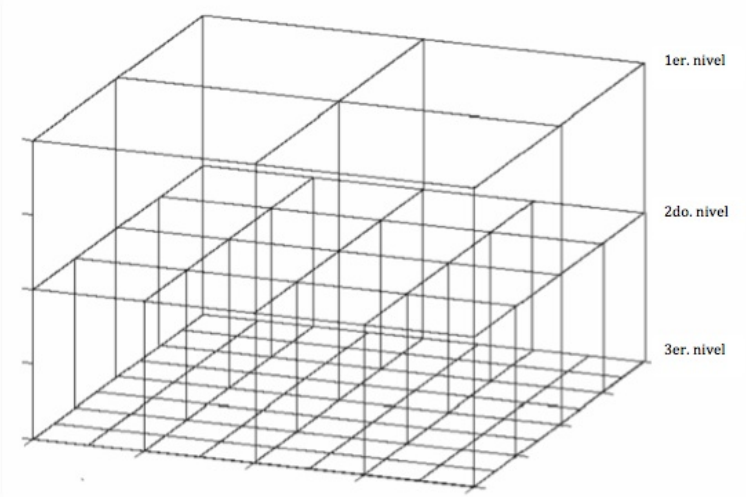


Figura 3.15: **Estructura de rejilla jerárquica en tres niveles**. Basado en **Lepistö, Kunttu, Autio, and Visa, 2004**.

La ventaja de ésta clase de algoritmos es que permiten encontrar clusters con forma arbitraria, independientemente de la distancia entre ellos. Para encontrar cluster con formas arbitarias, se mezclan las celdas en un conjuto basadas en su densidad. Además el tiempo de procesamiento es rápido, debido a que depende sólo del número de celdas en la rejilla, y no del número de objetos de datos **[Bungkomkhun, 2012]** **[Sheikholeslami, Chatterjee, and Zhang, 2000]**.

El algoritmo más común que emplea este enfoque es STING, el cual se analizará a continuación.

3.4.4.1. STING

STING (STatistical INformation Grid) es un algoritmo basaso en rejillas de multiresolución en el cual el espacio de los datos es dividido en celdas rectángulares. Usualmente hay varios niveles de celdas rectángulares que corresponden a diferentes niveles de resolución, y las celdas forman una estructura jerárquica, es decir, cada celda en el nivel más alto es particionada para formar un número de celdas en el nivel más bajo. Información estadística con respecto a los atributos de cada celda (tales como la media, la desviación estándar, los valores máximos y mínimos), pueden ser precalculados y almacenados. También es posible obtener el tipo de *distribución* de los objetos de datos (*normal*, *uniforme*, *exponencial* e incluso decir si la distribución es desconocida), de acuerdo a la información obtenida de las celdas **[Ilango and Mohan, 2010]**.

Para cada celda se tienen parámetros de *atributos dependientes* y *atributos independientes*. El parámetro de atributo independiente es **[Wang, Yang, and Muntz, 1997]**:

- n: el número de objetos (puntos) en la celda.

Para los parámetros de atributos dependientes, se asume que para cada objetos, sus atributos son valores numéricos. Para **cada** atributo numérico, se tienen los siguientes parámetros para cada celda **[Wang, Yang, and Muntz, 1997]**:

- m: la media de todos los valores de la celda.
- s: la desviación estándar de todos los valores de el atributo en la celda.
- min: el valor mínimo de el atributo de la celda.
- max: el valor máximo de el atributo de la celda.
- $distribucion$: el tipo de distribución que el valor del atributo en la celda sigue.

Para calcular estos parametros las formulas empleadas son las siguientes **[Wang, Yang, and Muntz, 1997]**:

- $n = \sum_i n_i$
- $m = \frac{\sum_i m_i n_i}{n}$
- $s = \sqrt{\frac{\sum_i (s_i^2 + m_i^2) n_i}{n} - m^2}$
- $min = \text{mín}_i(min_i)$
- $max = \text{máx}_i(max_i)$

Para poder establecer el tipo de distribución para cada celda padre se tienen que seguir ciertas reglas. Establezcamos *dist* como el tipo de distribución a seguir para muchos de los puntos en la celda. Para hacer esto se debe examinar $dist_i$ y n_i [5]. Entonces se estima el número de puntos, digamos $confl$ que causan conflicto con la distribución determinada. De acuerdo a *dist*, m y s las reglas son **[Wang, Yang, and Muntz, 1997]**:

- Si $dist_i \neq dist$, $m_i \approx m$ y $s_i \approx s$, entonces $confl$ es incrementado por una cantidad n_i.
- Si $dist_i \neq dist$, pero $m_i \approx m$ o $s_i \approx s$ no son satisfacidos, entonces establecemos $confl$ como n (Esto forza a que *dist* será establecido como ninguna distribución).
- Si $dist_i = dist$ y $m_i \approx m$, $s_i \approx s$, entonces $confl$ no es cambiado.
- Si $dist_i = dist$, pero $m_i \approx m$ y $s_i \approx s$ no se satisfacen, entonces $confl$ es establecido como n.

Finalmente, si $\frac{confl}{n}$ es más grande que un cierto umbral t (pequeña constante, establecida antes de construir la estructura), digamos 0.05, entonces establecemos *dist* como *distribución desconocida*, de otra forma, se establece la distribución originalmente establecida **[Wang, Yang, and Muntz, 1997]**.

Un ejemplo de esto sería el siguiente, a partir de la **Tabla 3.6**, la cual contiene los parámetros para los niveles más abajo de la jerárquia:

i	1	2	3	4
n_i	100	50	60	10
m_i	20.1	19.7	21.0	20.5
s_i	2.3	2.2	2.4	2.1
min_i	4.5	5.5	3.8	7
max_i	36	34	37	40
$dist_i$	NORMAL	NORMAL	NORMAL	NONE

Tabla 3.6: **Parámetros de las celdas hijo**. Basado en **Wang, Yang, and Muntz, 1997**.

[5] Parámetros de las celdas hijo, en los i-ésimos niveles

Entonces los parámetros de la celda actual son:

$$n = 220$$
$$m = 20{,}27$$
$$s = 2{,}37$$
$$min = 3{,}8$$
$$max = 40$$
$$dist = NORMAL$$

La distribución es aún **NORMAL** basado en lo siguientes: Ya que 210 puntos cuya distribución es de tipo **NORMAL**, *dist* es primero establecida como **NORMAL**. Después de examinar $dist_i$, m_i, y s_i para cada nivel de celdas más bajo, se ecnuentra que $confl = 10$. Y *dist* es establecido como **NORMAL** ($\frac{confl}{n} = 0{,}045 < 0{,}05$) **[Wang, Yang, and Muntz, 1997]**.

Habiendo establecido los parámetros anteriores, el algoritmo funciona de la siguiente manera. Empezando por el nivel más alto de la jerárquia o raíz, establecemos un intervalo de confianza [6] para esa celda. Después de estos se obtienen las celdas que son *relevantes* o *no relevantes* de acuerdo al nivel de confidencia. Una vez examinado el nivel actual, decendemos al siguiente nivel abajo, y se repite el mismo proceso. E procedimiento continua hasta que se examina el nivel más abajo en el nivel de jerarquía. STING formalmente se define en el **Algoritmo 3.6**:

Algoritmo 3.6 Algoritmo **STING**. Basado en **Wang, Yang, and Muntz, 1997**.

1: Determinar el nivel más alto con el que empezar.
2: Para cada celda en esa capa, calcular el intervalo de confianza (o rango estimado) de propabilidad para esa celda.
3: A partir del intervalo de confianza, etiquetar las celdas como *relevantes* o *no relevantes*.
4: Si este nivel es el último, ir al paso 6; si no ir al paso 5.
5: Ir al siguiente nivel abajo de la jerárquia. Ir al paso 2 para las celdas que son *relevantes* en el nivel más alto.
6: Si la especificación de solicitud es conocida, ir al paso 8; si no ir a 7.
7: Recuperar los datos que caen como celdas *relevantes* y realizar un proceso adicional. Retornar el resultado que cumple el requerimiento de la solicitud. Ir al paso 9.
8: Encontrar las regiones de *celdas relevantes*. Retornar esas regiones que cumple con la solicitud requerida. Ir al paso 9.
9: Parar.

[6] Un intervalo de confianza es un par de números entre los cuales se estima que estarán ciertos valores desconocidos con una determinada probabilidad de acierto

La ventaja de este algoritmo es que computacionalmente solo requiere $O(k)$, en donde k es el número de celdas en la rejilla de el nivel más bajo en la jerárquia. Usualmente $K \ll n$, en donde n es el número de objetos. Este algoritmo puede ser paralelizado de una manera trivial. Y cuando los datos son actualizados, no es necesario recalcular toda la información contenida en las celdas. Se puede hacer una actualización incremental **[Wang, Yang, and Muntz, 1997]**.

3.5. Evaluación de los algoritmos de clusters

La evaluación de cluster o **validación**, puede ser complicada por el hecho de que existen una variedad de algoritmos que obtienen sus clusters de diversas maneras, por ejemplo, si se aplicará *k-Medias* para obtener los clusters un buen criterio de evaluación sería en términos del *error cuadrático*, pero está medida no se podría aplicar a clusters basados en densidad, debido a que los clusters no son globulares y debido a que el error cuadrático no trabaja bien en ésta clase de algoritmos. Además en cuanto los clusters, serían muy diferentes los resultados obtenidos en cada algoritmo, por lo que se requeriría tener que comprar cada algoritmo para poder saber cual es el mejor **[Tang, Steinbach, and Kumar, 2006]**.

A diferencia de los métodos de minería de datos supervisados en donde se tiene información externa de los datos, en los métodos no supervisados, las medidas de validación se hacen de acuerdo a dos tipos de clases: **cohesión de cluster** (compacidad, estrechez), la cuál determina que tan relacionados están los objetos dentro de un cluster, y la **separación de cluster** (aislamiento), en el cual se determina que tan distintos o bien separados están los clusters **[Liu, Li, Xiong, Gao, and Wu, 2010]**. Ésto es ilustrado en la **figura 3.16**.

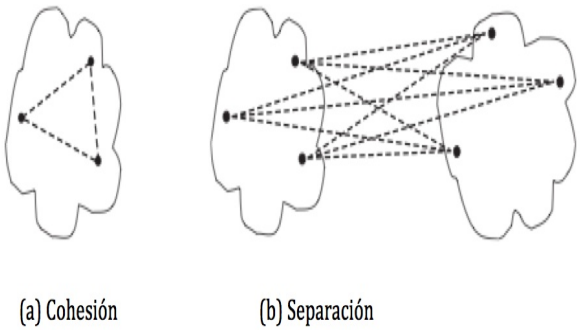


Figura 3.16: **Vista gráfica de la cohesión de cluster y separación de cluster**. Basado en **Tang, Steinbach, and Kumar, 2006**.

Matemáticamente, la cohesión y separación de cluster pueden ser expresadas através de las ecuaciones **3.22** y **3.23**:

$$cohesion(C_i) = \sum_{x \in C_i, y \in C_i} proximidad(\mathbf{x},\mathbf{y}) \tag{3.22}$$

$$separacion(C_i, C_j) = \sum_{x \in C_i, y \in C_j} proximidad(\mathbf{x},\mathbf{y}) \tag{3.23}$$

La función de *proximidad* puede ser una similaridad, una disimilaridad, o una simple función de estas cantidades. Por ejemplo, en los algoritmos basados en partición, la cohesión puede ser definida como la suma de las proximidades con respecto al centro o medoide del cluster, en donde esta proximidad podría ser el *error cuadrático medio* si la proximidad es la distancia Euclidiana cuadrática **[Tang, Steinbach, and Kumar, 2006]**.

$$cohesion(C_i) = \sum_{x \in C_i} proximidad(\mathbf{x},c_i) \tag{3.24}$$

$$separacion(C_i, C_j) = proximidad(c_i,c_j) \tag{3.25}$$

$$separacion(C_i) = proximidad(c_i,\mathbf{c}) \tag{3.26}$$

Otro enfoque para medir la calidad de los clusters es usar una matriz de similaridad. Si se tiene una matriz de similaridad para un conjunto de datos y las etiquetas de los clusters de un análisis de cluster hecho a un conjunto de datos, es posible evaluar el resultado de la matriz de similaridad contra un versión ideal de la matriz de similaridad basada en las etiquetas del cluster. Específicamente hablando, una matriz de similaridad ideal es aquella en la que todos los puntos tienen similaridad de 1 en el cluster, y similaridad de 0 para todos los puntos en otros clusters **[Tang, Steinbach, and Kumar, 2006]**. De este modo si se ordenan las filas y columnas de la matriz de similaridad de modo que todos los objetos que pertenecen a la misma clase esten juntos, entonces la matriz de similaridad ideal tiene un estructura de **bloque diagonal**. Con este enfoque es posible medir clusters de algoritmos de particionamiento y clusters basados en densidad. Un ejemplo de ésta matriz de similaridad es representada en la **figura 3.17**.

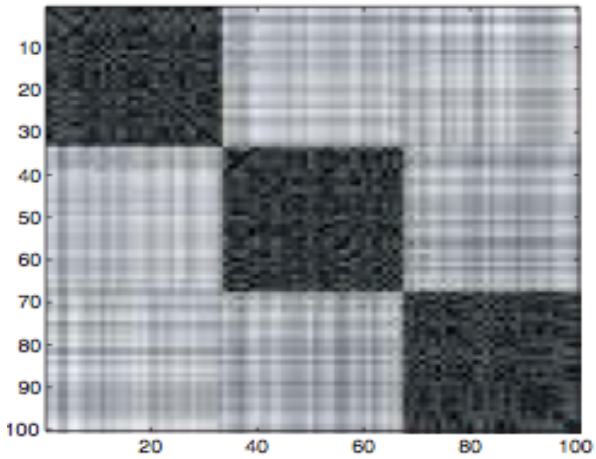


Figura 3.17: **Matriz de similaridad para clusters bien formados**. Basado en **Tang, Steinbach, and Kumar, 2006**.

Con respecto a los algoritmos jerárquicos, la medida de evaluación más popular es la técnica conocida como **distancia cofenética**, en la cual se mide la proximidad en la cual un algoritmo jerárquico aglomerativo pone los objetos por primera vez. Por ejemplo, si en algún punto el el proceso del clustering jerárquico aglomerativo, la distancia más pequeña entre dos clusters que son mezclados es 0.1, entonces todos los puntos en un cluster tienen distancia cofenética de 0.1 con respecto a los puntos en el otro cluster **[Tang, Steinbach, and Kumar, 2006]**. Si se aplica un clustering de un sólo enlace, las distancias cofenéticas para cada punto pueden ser expresadas en una matriz de distancia cofenética, como es representado en la **tabla 3.7**.

Punto	p_1	p_2	p_3	p_4	p_5	p_6
p_1	0	0.222	0.222	0.222	0.222	0.222
p_2	0.222	0	0.148	0.151	0.139	0.148
p_3	0.222	0.148	0	0.151	0.148	0.110
p_4	0.222	0.151	0.151	0	0.151	0.151
p_5	0.222	0.139	0.148	0.151	0	0.148
p_6	0.222	0.148	0.110	0.151	0.148	0

Tabla 3.7: **Matriz cofenética para datos y un clustering de simple enlace**. Basado en **Tang, Steinbach, and Kumar, 2006**.

Con esta matriz el **Coeficiente de Correlación Cofenética** puede ser calculada. Este coeficiente es una medida estándar en el cual datos que existen en esta matriz son comparados con respecto a la matriz de disimilaridad de los datos original, buscando así encontrar el algoritmos que mejor se ajuste a los datos **[Tang, Steinbach, and Kumar, 2006] [Romesburg, 1984]**.

3.6. Panorama del Clustering distribuido

Al paso de los años, los avances tecnológicos actuales han permitido que los equipos de cómputo incrementen su capacidad de procesamiento de una manera sorprendente, así como su capacidad para almacenar una cantidad muy grande de datos los cuales son generados por muchas organizaciones de negocios, manufactureras y que se dedican a procesos científicos, entre otras. Muchos de estos datos están distribuidos en diferentes sitios geográficos, por lo que se ha requerido el desarrollo de diversos métodos que puedan analizar estos datos, sin tener la necesidad de tener que transferirlos a una sóla localización para su análisis, esto se debe por razones técnicas, de seguridad e inclusive económicas.

Buscando mejorar la eficiencia de los algoritmos de clustering, en la literatura se ha generado un importante trabajo de investigación en el desarrollo de algoritmos de clustering distribuido que permitan reducir gastos de comunicación, requerimientos de almacenamiento central, y tiempo de procesamiento. Algunos de estos trabajos son presentados en esta sección.

3.6.1. Algoritmos de Clustering Distribuido

En el clustering distribuido se asume que los objetos a ser agrupados residen de diferentes sitios. En vez de transmitir estos objetos a un sitio central donde un algoritmo de clustering puede ser aplicado a los datos para su análisis, los datos son agrupados independientemente en cada sitio obteniendo modelos locales. Después un sitio central trata de establecer un clustering global basado en los modelos locales. De acuerdo a esto, los algoritmos de clustering pueden ser clasificado según dos dimensiones independientes: *clasificación basada en la distribución de los datos* y *clasificación basada en la comunicación de los datos* **[Visalakshi and Thangavel, 2009]**.

3.6.1.1. Clasificación basada en la Distribución de los Datos

En esta clasificación los datos pueden estar distribuidos homogéneamente o heterogéneamente. Homogéneamente los datos contiene el mismo conjunto de atributos através de sitios de datos distribuidos. Un ejemplo de esto incluye bases de datos locales del clima localizadas en diferentes sitios geográficos y los datos recolectados de cestas de compra en diferentes localizaciones de una cadena de tiendas. El modelo de datos heterogéneo permite diferentes sitios con diferentes esquemas. Por ejemplo, el problema de detección de enfermedades de emergencia puede requerir información colectiva de una base de datos de enfermedades, una base de datos demográfica, y bases de datos de vigilancia biológica.

3.6.1.2. Clasificación basada en la comunicación de Datos

De acuerdo al tipo de comunicación de datos, los algoritmos de clustering están clasificados en dos categorías: algoritmos de múltiples rondas de comunicación y algoritmos basados en ensamble centralizado. El primer grupo consiste de métodos los que requieren múltiples rondas de paso de mensajes, por lo que necesitan una cantidad de sincronización significativa. Algunos trabajos que manejan estos métodos en la literatura son los siguientes:

- **[Dhillon and Modha, 1999]**. Estos autores desarrollaron un implementación paralela de el algoritmo de K-Medias sobre múltiples multi-procesadores distribuidos (datos distribuidos homogéneamente). Dado un conjunto de datos de tamaño n, este es dividido en P bloques (cada uno de tamaño aproximado a n/P). Durante cada iteración de K-Medias, cada sitio procesa un actualización de los K centroides actuales basados en sus propios datos. Los sitios transmiten sus centroides, y una vez que un sitio ha recibido todos los centroides de los otros sitios, este puede formar sus centroides globales promediando todos los centroides recibidos.

- **[Kargupta, Huang, Sivakumar, and Johnson, 2003]**. Estos autores proponen el desarrolló un análisis de componentes principales colectivos basados en la técnica PCA (Principal Component Analysis[7]) para datos distribuidos heterogéneamente. Cada sitio local lleva a cabo un PCA, y proyecta los datos locales según los componentes principales, y aplica un conocido algoritmo de clustering. Habiendo obtenido estos clusters locales, cada sitio envía un pequeño conjunto de datos representativos a un sitio central. Este sitio llava a cabo PCA de los datos recolectados (procesa componentes principales globales). Los componente principales globales son enviados de vuelta a los sitios locales. Cada sitio proyecta sus datos según los componentes principales global y aplica su algoritmo de clustering.

- **[Klush, Lodi, and Moro, 2001]** Estos autores llevan a cabo un análisis de cluster basado en densidad sobre datos distribuidos homogéneamente. Su algoritmo no efectua el clustering sobre el conjunto de datos total. Sino en vez de esto cada sitio local efectua el clustering de sus datos locales basados en una función de densidad procesando *todos* sus datos y una aproximación para la función de densidad global es procesada en cada sitio usando la teoría de muestreo de procesamiento de señales[8]. Los sitios deben primero coincidir sobre un cubo o rejilla (del cubo). Cada punto en la esquina puede ser

[7] El **análisis de componentes principales (PCA)** es un procedimiento matemático que usa una transformación ortogonal para convertir un conjunto de observaciones de variables posiblemente correlacionadas en un conjunto de valores de variables no correlacionadas llamadas **componentes principales**.

[8] La teoria de muestreo trata con el proceso de tomar alguna *señal* continua que varia con uno o más parámetros, y muestrear la señal en valores discretos de estos parámetros

tomado como una muestra del espacio (no del conjunto de datos). Entonces cada sitio procesa el valor de su función de densidad local en cada esquina de la rejilla y transmiten los puntos de la esquina junto con su valor de densidad local a un sitio central. El sitio central procesa la suma de todas las muestras de cada punto de la rejilla y transmite la muestra combinada de vuelta a cada sitio. Los sitios locales pueden ahora independientemente estimar la función de densidad global sobre todos los puntos en el cubo (no sólo en los puntos de la esquina) usando técnicas de teoria de de muestreo en procesamiento de señales.

El segundo grupo consiste de métodos que construyen modelos de clustering locales y se transmiten a un sitio central (asíncronamente). El sitio central forma un modelo global combinando todos estos modelos locales. Estos métodos requieren una sóla ronda de paso de mensajes, por lo tanto sólo requieren una modesta sincronización. Algunos trabajos en la literatura que usan estos métodos son los siguientes **[Bandyopadhyay, Giannella, Maulik, Kargupta, Liu, and Datta, 2006]**:

- **[Johnson and Kargupta, 1999]** Estos autores desarrollaron un algoritmo de clustering jerárquico distribuido sobre datos heterogéneamente distribuidos. Este primero genera modelos de cluster local y los combina para formar el modelo global. En cada sitio local, el algoritmo de clustering jerárquico es aplicado para generar los dendogramas locales los cuales serán transmitidos a un sitio central. Usando límites estadísticos, un dendograma global es generado.

- **[Samatova, Ostrouchov, Geist, and Malechko, 2002]** Los autores desarrollaron un método para unir clustering jerárquicos de datos con valores reales distribuidos homogéneamente. Cada sitio produce un dendograma basado en datos locales, y los transmite a un sitio central. Para reducir los costos de comunicación, los sitios no envían una descripción completa de cada cluster en un dendograma. En vez de esto una aproximación de cada cluster es enviada consistiendo de varias estadísticas descriptivas, por ejemplo, el número de puntos en el cluster, el promedio de la distancia Euclidiana de cada punto en el cluster al centroide. El sito central combina las descripciones de dendogramas en una descripción de dendograma global.

- **[Januzaj, Kriegel, and Pfeifle, 2004]** Januzaj extiende un algoritmo de clustering centralizado basado en densidad: DBSCAN. Cada sitio lleva a cabo el algoritmo DBSCAN, transmitiendo un representación compacta de cada clustering local a un sitio central, una representación global es producida de representaciones locales, y finalmente esta representación global es enviada de vuelta a cada sitio. Un clustering es representado primero escogiendo que: (i) cada punto tenga los suficientes vecinos en su vecindario (determinado por un umbral fijo) y (ii) dos puntos no esten en el mismo ve-

cindario. Entonces el clustering de K-Medias es aplicado a todos los puntos en el cluster, usando cada una de los puntos de muestra como un centroide inicial. El centroide final junto con el punto más lejano en su cluster de K-Medias forma una representación (una colección de puntos, radios pares). El algoritmo DBSCAN es aplicado al sitio central sobre la unión de los puntos representativos locales para formar el clustering global.

3.6.2. Clustering Distribuido basado en ensamble

El ensamble de cluster consiste en combinar múltiples particiones de un conjunto de objetos en un sólo clustering consolidado sin accesar a las características ó algoritmos que determinaron estas particiones. Este ha emergido como un poderoso método de mejoramiento de la robustez así como de la estabilidad de soluciones de aprendizaje no supervisado **[Ji and Ling, 2007]**. El ensamble de cluster puede ser formado de numerosas maneras, como por ejemplo:

- ⋆ Usando diferentes algoritmos de clustering para producir particiones para combinación.
- ⋆ Cambiando la inicialización u otros parámetros de algoritmos de clustering.
- ⋆ Usando diferentes características vía extracción caracteristica para el clustering subsecuente.
- ⋆ Particionando diferentes subconjuntos de los datos originales.

Todos los mecanismos antes mencionados tratan de producir más diversidad considerando los datos desde diferentes aspectos. La dificultad más importante en el ensamble de cluster esta en las funciones de consenso y los algoritmos para combinar particiones que producen una partición final, o en otras palabras que encuentran una partición de consenso de las particiones de salida de varios algoritmos de clustering. La combinación de múltiples clustering puede también ser visto como el encontrar una partición mediana con respecto a las particiones dada, lo cual está probado que es un problema NP-Completo **[Visalakshi and Thangavel, 2009]**.

Capítulo 4

Sistemas Multi-Agentes

El propósito de éste capítulo es presentar de manera general los fundamentos de la tecnología de agentes. En la primera parte se define el término *agente inteligente*, sus principales características, así como la estructura de una arquitectura basada en agentes inteligentes. Finalmente se menciona los desafios que existen en la construcción e implementación de un sistema basado en agentes inteligentes. En la segunda parte se presenta la plataforma JADE como una herramienta de software qué permite la construcción de sistemas basados en agentes.

4.1. ¿Qué son los agentes inteligentes?

Para poder definir que es un *agente inteligente* se requiere hacer una distinción entre lo que es un *agente* y lo que es un *agente inteligente*. El término *agente* se refiere a cualquier clase de objeto o persona que lleva a cabo alguna tarea o acción con el fin de cumplir sus objetivos. Un ejemplo de este tipo de agente en la vida diaria sería una persona que actúa como intermediario entre vendedores y compradores en una transacción **[Sterling and Taveter, 2009]**.

Por otra parte el término *agente inteligente* se considera como una entidad que puede percibir su entorno por medio de **sensores** y puede actuar en cierto entorno por medio de **efectores**[1], como es representado en la **figura 4.1 [Rusell and Norvig, 2003]**. En este contexto el entorno se refiere a cualquier situación ó área donde el agente actúa.

[1]Para poder ilustrar mejor lo que son los sensores y efectores, veamos el siguiente ejemplo, en un agente humano los sensores serían sus organos, como son sus ojos, orejas, etc., y sus manos, pies, boca, y otras partes del cuerpo serían los efectores, en un agente de software las pulsaciones del teclado, contenido de los archivos, y paquetes de red serían los impulsos sensoriales y actuarían desplegandose en el monitor, archivos de escritura y envíando paquetes de red **[Rusell and Norvig, 2003]**.

Además de estas características, para que un agente pueda ser considerado "inteligente" éste debe de cumplir con una serie de atributos que no sólo le permitan actuar sino que pueda ser capáz de entender su entorno y reaccionar de acuerdo a las características que se presenten en dicho entorno. Wooldridge y Jennings presentan una definición del término *agente inteligente* desde el punto de vista computacional y listan propiedades que tiene que cumplir un agente para ser considerado inteligente **[Wooldridge and Jennings, 1995]**:

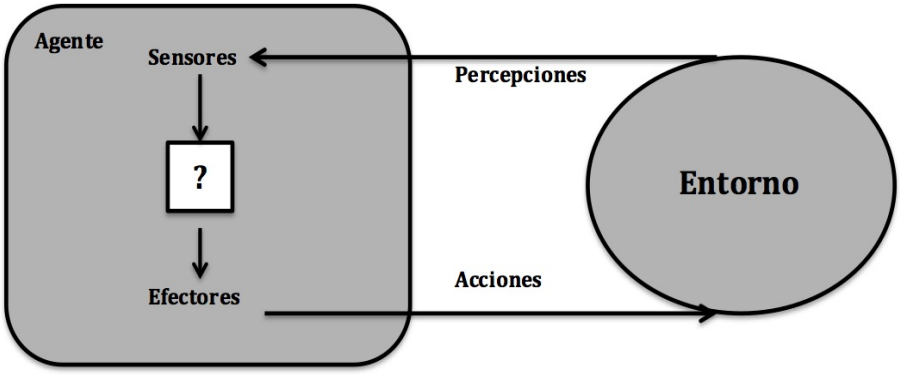


Figura 4.1: **Interacción entre agentes y el entorno através de sensores y efectores.** Adaptado de **Rusell and Norvig, 2003**.

- Un agente inteligente denota sistemas de computadoras basados en software y hardware que cumplen con las siguientes propiedades:
  - **Autonomía.** Los agentes pueden operar sin la intervención de humanos u otros agentes, y tienen algún tipo de control sobre sus acciones y estado interno.
  - **Habilidad para socializar.** Los agentes pueden interactuar con otros agentes por medio de un *lenguaje de comunicación de agentes*.
  - **Reactividad.** Los agentes tienen la capacidad de percibir su entorno y responder oportunamente a los cambios que ocurren en el.
  - **Proactividad.** Los agentes no simplemente actúan respondiendo a su entorno, ellos pueden exhibir un comportamiento dirigido a cumplir con sus objetivos tomando iniciativa propia.

A estas propiedades se le añaden otros atributos como son **[Bellifemine, Caire, and Greenwood, 2007]**:

- **Movilidad.** Es la habilidad que tienen los agentes para moverse alrededor de una red electrónica.
- **Veracidad.** Se asume que los agentes no comunicarán información falsa.

- **Benevolencia.** Cuando un agente tenga que trabajar en cooperación con otros agentes, estos no tendrán conflictos de objetivos, de este modo cada agente siempre tratará de hacer lo que se le pide.
- **Racionalidad.** Se asume que los agentes actuarán con el fin de lograr sus objetivos y nunca actuarán de manera que estos objetivos no sean logrados. Ellos podrán llevar a cabo acciones con el fin de modificar percepciones futuras obteniendo información útil (recolección de datos, exploración).
- **Aprendizaje.** Los agentes tienen la capacidad de aprender de sus experiencias pasadas.

4.2. Agentes situados en entornos

Para que un agente pueda cumplir con éxito los objetivos para el cual fué diseñado éste debe poder tomar decisiones con base en el entorno donde esta implementado. Lo que un agente haga dentro de este entorno depende de los siguientes factores **[Poole and Mackworth, 2010]**:

- el **conocimiento previo** que tiene del entorno;
- la **historia** de su interacción con el entorno, la cuál esta compuesta de:
  - **observaciones** del entorno actual,
  - **experiencias pasadas** de acciones previas y sus observaciones, u otros datos de los cuáles puede aprender;
- los **objetivos** que debe tratar de lograr o preferenciar sobre el estado del mundo;
- las **habilidades** que a su vez son acciones primitivas que el agente pueda llevar acabo.

En la **figura 4.2** se muestra como es esta interacción entre el agente y su entorno.

Ahora bien, la toma decisiones puede verse afectada por las propiedades de las que esta conformado el entorno. Russell y Norving presentan una clasificación de los distintos entornos que se pueden presentar de acuerdo a una serie de propiedades. Esta clasificación es la siguiente **[Rusell and Norvig, 2003]**:

- **Accesibles vs Inaccesibles**. Un entorno accesible es aquel donde el agente puede obtener información completa, precisa y actual del estado del entorno. La mayoría de los entornos de complejidad moderada (incluyendo, el mundo cotidiano físico e internet) son inaccesibles.

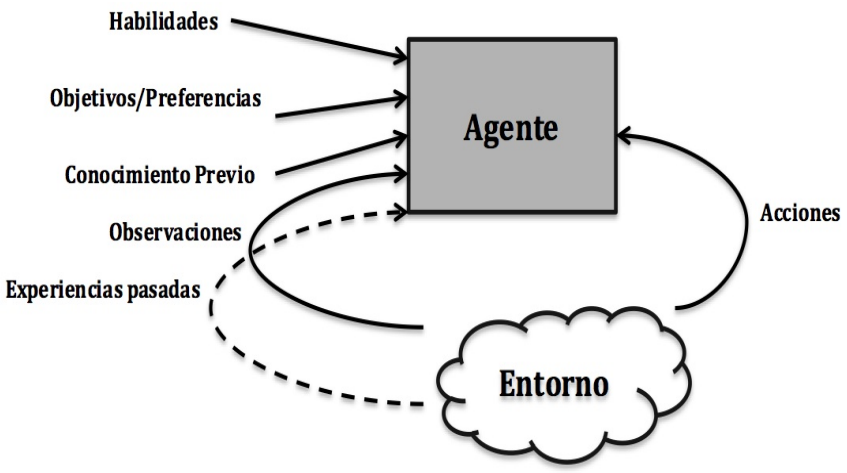


Figura 4.2: **Factores que determinan que un agente cumpla sus objetivos**. Adaptado de **Poole and Mackworth, 2010**.

- **Determinísticos vs No determinísticos**. Un entorno determinístico es aquel en donde cualquier acción que se lleve acabo sólo producirá un efecto determinado de salida, de otra manera este será no determinístico.
- **Episódicos vs Secuencial**. En un entorno episódico, la experiencia del agente es dividido en episodios atómicos. Cada episodio consiste de que el agente perciba su entorno y lleve a cabo una sóla acción. Un punto crucial es que el siguiente periódo no depende de las acciones tomadas en los episodios previos. Por ejemplo, un agente que tiene que reconocer partes defectuosas en una línea de ensamblaje basa cada decisión en la parte actual, sin tomar en cuenta las decisiones previas. Por otro lado en un entorno secuencial, la decisión actual puede afectar a las futuras decisiones que se tienen que tomar.
- **Estáticos vs Dinámicos**. Un entorno que cambia mientras un agente está deliberando, se dice que es dinámico para el agente; de otra manera se dice que es estático.
- **Discreto vs Continuo**. Un entorno se dice que es discreto si hay un número fijo y finito de acciones y percepciones en él. Si por otra parte estas acciones pueden cambiar a lo largo del tiempo entonces se dice que el entorno es continuo.

4.3. Arquitecturas para agentes inteligentes

Para que los agentes puedan llevar a cabo sus acciones, es necesario que éstos sean implementados sobre mecanismos que soporten de manera efectiva los com-

portamientos que tienen los agentes en el mundo real y en entornos dinámicos y abiertos. Estos mecanismos son conocidos como **arquitecturas agente**. La forma exacta de éstas arquitecturas dependerá de sus tareas y del entorno sobre el cuál llevarán a cabo estas tareas. Dependiendo del tipo de arquitectura sobre la cuál sean implementados los agentes estos serán más complejos e inteligentes que otros. Estas **arquitecturas agentes** pueden ser dividas en cuatro grupos fundamentales **[Bellifemine, Caire, and Greenwood, 2007]**:

- **Arquitecturas basada en lógica.** En esta clase de arquitecturas se maneja un enfoque "tradicional" para la construcción de sistemas artificiales inteligentes. Esta arquitectura sugiere que comportamiento inteligente puede ser generado en un sistema dandole una representación *simbólica* de su entorno y su comportamiento deseado. Sintácticamente esta representación puede ser manipulada. La ventaja de este enfoque es que el conocimiento humano es simbólico y puede ser codificado de manera más fácil, la desventaja es que este conocimiento es difícil de trasladar al mundo real de forma precisa por lo que esta descripción simbólica podría no ser adecuada **[Rusell and Norvig, 2003]**.

- **Arquitectura reactiva.** En este tipo de arquitectura un agente toma decisiones por medio de un conjunto de tareas que ejecutan comportamientos; cada comportamiento representa una acción individual, en donde esta acción es resultado de un mecanismo que responde a una serie de estímulos que provienen de sensores de datos. A diferencia de las arquitecturas basadas en lógica no hay un modelo simbólico central que utilice algún tipo de razonamiento simbólico, sino que más bien un comportamiento inteligente puede ser generado por medio de técnicas de inteligencia artificial que emergen ciertas propiedades de un sistema complejo. Este tipo de arquitectura está basada en la arquitectura de subsunción de Brook[2] **[Brooks, 1991]**.

- **Arquitecturas Belief, Desire, Intention.** En esta clase de arquitecturas las creencias representan la información que un agente tiene acerca de su entorno. Los deseos representan las tareas que están asignadas a los agentes y que corresponden a los objetivos y metas que deberían cumplir. Las intenciones representan los deseos a que los agentes se han comprometido a lograr. Para que este tipo de arquitectura pueda ser implementada, los agentes tienen que empezar por ejecutar un plan. Los planes son una secuencia de acciones que son llevados a cabo para lograr una o más intenciones. Éstas cuatro estructuras de datos son administradas por un agente interprete el

[2]En una arquitectura de subsunción un comportamiento inteligente complicado se puede descomponer en muchos "modulos" ó subcomportamientos más simples. Estos subcomportamientos a su vez representan un nivel de capa, en donde cada capa implementa un objetivo particular del agente, conforme cada capa es más alta esta es a su vez más abstracta. Este tipo de arquitectura esta organizada en un manera bottom-up. Esto quiere decir que comportamientos complejos son formados de una combinación de comportamientos más simples.

cual es responsable de actualizar las creencias de las observaciones hechas del entorno, generando nuevos deseos (tareas) con base en nuevas creencias, y seleccionando del conjunto de deseos activos algún subconjunto que actúe como intenciones. Agentes en esta clase de agentes también son conocidos como deliberativos **[Bratman, 1987] [Rao and Georgeff, 1995]**.

- **Arquitecturas híbridas.** En este tipo de arquitecturas los agentes aprovechan las ventajas de las arquitecturas reactiva y deliberativa. Por medio de la reactividad los agentes pueden reconocer eventos inesperados y reaccionar apropiadamente, la deliberación les permite seleccionar las acciones en base a los fines que quieren alcanzar tomando en cuenta los medios de los cuál dispone **[Ferguson, 1991] [Muller, Pischel, and Thiel, 1995]**.

4.4. Múltiples agentes

Hacer que un agente razone lo que debe hacer dentro de su entorno cuando este sólo es el único que existe puede ser bastante difícil. Si ahora este agente tiene que interactuar con otros agentes que existen dentro del mismo entorno entonces la dificultad se puede volver mayor. Ya que aparte de tener que razonar su entorno este debe razonar como estratégicamente va interactuar con los otros agentes.

En un entorno multiagente, los agentes pueden ser diseñados para cooperar y tener objetivos en común, pero el problema en esto radica en cómo los agentes tienen que coordinarse y comunicarse. Los entornos en el mundo real pueden ser tan complejos, que hacen necesario formalizar la coordinación y la comunicación entre agentes. En computación los agentes pueden representar procesos que ocurren en un mismo instante de tiempo, compartiendo recursos y comunicandose entre sí, que para poder interactuar éxitosamente, es necesario que ellos puedan coordinarse, cooperar y negociar entre sí **[Poole and Mackworth, 2010]**.

Un sistema multi-agente presenta las siguientes características **[Sycara, 1998]**:

- cada agente tiene información incompleta o ciertas capacidades limitadas para resolver un problema, y de este modo su punto de vista es limitado;
- no hay un sistema de control global;
- los datos son descentralizados; y
- la computación es asíncrona.

Estas características han motivado el creciente interes en investigar los MAS que incluyen la habilidad para poder hacer lo siguiente **[Sycara, 1998]**:

1. Resolver problemas que son demasiado grandes para un agente centralizado, dada la limitación de recursos que tiene ó el riesgo que tiene un sistema centralizado a que su desempeño se vuelva un cuello de botella o pueda fallar en momentos críticos.

2. Permitir la interconexión e interoperación de múltiples sitios de cómputo existentes.

3. Dar solución a problemas que pueden ser considerados como una sociedad de componentes de agentes que pueden interactuar autónomamente. Por ejemplo, en la programación de reuniones, un agente planificador que administra el calendario de su usuario puede ser considerado como autónomo e interactuar con otros agentes que administran calendarios de diferentes usuarios.

4. Dar soluciones a problemas usando eficientemente fuentes de información distribuidas.

5. Dar solución a situaciones donde la experiencia es distribuida.

6. Mejorar el desempeño de la **[Bradshow, 1997]**:

   - *eficiencia computacional* debido a que la concurrencia es explotada (siempre que la comunicación se mantenga al mínimo, por ejemplo, transmitiendo información de alto nivel y resultados en vez de datos de bajo nivel);

   - *confiabilidad*, es decir, la recuperación ante fallos de componentes, los agentes con capacidades redundantes ó coordinación interagente apropiada se encuentran dinámicamente (por ejemplo, tomando responsabilidades de los agentes que fallan).

   - *extensibilidad*, el número de capacidades de un agente que trabajan en un problema pueden ser alterados;

   - *robustez*, la habilidad del sistema tolera incertidumbre, debido a que información adecuada es intercambiada entre los agentes;

   - *mantenibilidad*, un sistema compuesto de múltiples componentes de agentes es más fácil de mantener debido a su modularidad;

   - *sensibilidad*, debido a que la modularidad puede manejar anomalías localmente, y no propagarlas a todo el sistema;

   - *flexibilidad*, los agentes con diferentes habilidades pueden adaptativamente organizarse para resolver el problema actual; y

- *reuso*, agentes específicos pueden ser reusados en diferentes equipos de agentes para resolver diferentes problemas.

4.4.1. Comunicación entre agentes

Un componente clave en sistemas multiagentes es la comunicación. Según el tipo de entorno donde éstos sean implementados necesitarán comunicarse con usuarios, con recursos de sistema y con otros agentes, si ellos necesitan llevar acabo acciones de coordinación, cooperación y negociación. La **coordinación** es la propiedad que tiene un sistema de agentes para llevar a cabo alguna actividad en un entorno compartido, la **cooperación** se refiere a la coordinación entre agentes no antagonístas, es decir que comparten muchos de sus objetivos y la **negociación** es un proceso en el cuál dos o más agentes con objetivos diferentes llegan a una decisión conjunta **[Weiss, 1999]**.

Para que los agentes puedan interactuar con otros agentes deben utilizar algunos lenguajes de comunicación especial, mejor conocidos como **lenguajes de comunicación de agentes**, los cuáles tratán de proporcionar un serie de **performativas** (actos comunicativos) y un contenido de lenguaje específico para comunicarse **[Bellifemine, Caire, and Greenwood, 2007]**. Algunos tipos de lenguajes de comunicación de agentes que existen son los siguientes:

- **KQML.** El *Knowledge Query and Manipulation Language* (KQML) es un lenguaje desarrollado a principios de los años 90's como parte del proyecto del gobierno de US llamado ARPA, el cuál proporciona un marco de funcionamiento entre programas y agentes para intercambiar información y conocimiento. EL KQML incide sobre los diferentes formatos de los mensajes (llamados *performativas*), operaciones y protocolos de manipulación de mensajes entre los entornos de ejecución de los agentes. No incide sobre el contenido o lenguaje usado para representar ese contenido **[Genesereth and Ketchpel, 1994]**.

  KQML utiliza ontologías[3] para dar especificaciones explícitas de los significados, conceptos y relaciones aplicables a algún dominio específico, para asegurarse que dos agentes que se comunican en el mismo lenguaje puedan interpretar correctamente afirmaciones es ese lenguaje. Los mensajes KQML codifican información en tres diferentes niveles arquitectónicos: contenido, mensaje y comunicación: el nivel de comunicación (emisor y receptor), nivel de mensaje (lenguaje y ontologías), y el nivel de contenido (afirmaciones específicas del lenguaje) **[Finin, Fritzson, McKay, and McEntire, 1994]**.

[3]Ontología en el término de computación representa vocabularios específicos que son usados por componentes que comparten cierto dominio de problemas y esquemas

- **FIPA ACL.** Este es introducido por la *Foundation for Intelligent Physical Agents* (FIPA), el cuál incorpora muchos aspectos de KQML. Una de las características más importantes es que posibilita el uso de diferentes contenidos de lenguajes y la administración de conversaciones através de protocolos de interacción predefinidos **[Bellifemine, Caire, and Greenwood, 2007]**.

4.4.2. Coordinación entre agentes

Existen varias razones por las que los agentes necesitan coordinarse, entre estos se encuentran **[Nwana, Lee, and Jennings, 1996]**:

1. Los objetivos de los agentes pueden causar conflictos entre las acciones de los agentes.
2. Los objetivos de los agentes pueden ser independientes.
3. Los agentes pueden tener diferentes capacidades y diferente conocimiento.
4. Los objetivos de los agentes pueden ser logrados más rápidos, si diferentes agentes trabajan en cada uno de ellos.

Para lograr tal coordinación se han manejado una variedad de enfoques entre los cuáles se encuentran: estructuras organizacionales, contratación, planificación multi-agentes y negociación. Las principales características de estos enfoques se listan a continuación **[Bellifemine, Caire, and Greenwood, 2007]**:

- **Estructura organizacional [Durfee, 1999]**. En este enfoque se provee un marco de trabajo para las actividades e interacción entre los agentes através de definir roles, caminos de comunicación y relaciones de autoridad. La mejor forma de asegurar el comportamiento coherente de los agentes y resolver conflictos entre ellos consiste en proveer a este grupo de agentes un agente el cuál tiene una amplia perpectiva de el sistema, de este modo se podría explotar una estructura organizacional o jerárquica. En este tipo de coordinación se maneja un arquitectura maestro/esclavo o cliente/servidor para asignar recursos y tareas entre los agentes esclavos por un agente maestro. El maestro controlador puede recolectar información de los agentes en el grupo, crear planes y asignar tareas a agentes individuales con el fin de asegurar coherencia global.
- **Redes de contratación [Smith and Davis, 1980]**. Esta arquitectura se basa en un conjunto de nodos, los cuáles tienen dos roles, administrador y trabajador, y que se encuentran en una estructura de mercado descentralizada en la cual la premisa básica es que si un agente no puede resolver un

problema asignado usando recursos/habilidades locales, el agente podrá descomponer el problema en sub-problemas y tratarán de encontrar otro agente dispuesto con los recursos/habilidades que resuelvan estos sub-problemas.

- **Planeación multi-agente [Georgeff, 1983]**. Con el fin de evitar inconsistencias o acciones conflictivas e interacciones, los agentes pueden construir un plan que detalla todas las acciones futuras e interacciones requeridas para lograr sus objetivos e intercalar sus ejecuciones con planeación adicional y replaneación. Una planeación multi-agente puede ser centralizada o distribuída. En la *planeación multi-agente centralizada* hay usualmente un agente coordinador que recibe todos los planes locales o parciales de los agentes, los analiza e identifica inconsistencias e interacciones conflictivas **[Georgeff, 1983]**. En la *planeación multi-agente distribuida*, la idea es que cada agente tenga un modelo de los planes de los otros agentes. Los agentes se comunicarán con el fin de construir y actualizar sus planes individuales y los modelos de otros agentes hasta que todos los conflictos sean removidos **[Georgeff, 1984]**.

- **Negociación [Bussmann and Muller, 1992]**. Es probable que sea la técnica más confiable para la coordinación de agentes. La negociación puede ser competitiva o cooperativa dependiendo del comportamiento de los agentes involucrados. En la *negociación competitiva* los agentes tiene objetivos independientes que interactuán los unos con los otros. Pero ellos no son cooperativos compartiendo información o dando marcha atrás por el bien del grupo de agentes. En la *negociación cooperativa* los agentes tienen un objetivo en común que lograr o ejecutan una sóla tarea. En este caso los sistemas multi-agentes han sido diseñados para seguir un sólo objetivo global.

4.5. Diseño de sistemas multi-agentes

Hasta esta parte se han presentado los fundamentos teóricos de los sistemas multiagentes. A partir de esta sección se estudiarán algunos métodos que existen para la construcción de sistemas multiagentes, en estos métodos generalmente se aplican mucho de los conocimientos de ingeniería de software aplicados exclusivamente para la construcción de sistemas basados en agentes, esta área de conocimiento es conocida como *Ingeniería de Software Orientada a Agentes* **[Jennings and Wooldridge, 2000]**.

Los métodos descritos en esta sección buscan proveer metodologías que permitan modelar sistemas reales complejos en el cuál los agentes tienen que trabajar **[Henderson-Sellers and Giorgini, 2005]**. Si estas metodologías son aplicadas éxitosamente entonces el sistema construído será éxitoso.

4.5.1. Gaia

Gaia es una metodología para análizar y diseñar sistemas orientados a agentes, la cuál se desarrolla llevando acabo una serie de pasos para identificar un conjunto de modelos organizacionales y como estan relacionados, que guían a los desarrolladores hacia el desarrollo de un MAS **[Sterling and Taveter, 2009]**.

La metodología Gaia comienza con una **etapa de análisis**, cuya intención es la de recolectar y organizar especificaciones o requerimientos, que son base para diseñar la organización computacional (estos requermientos implican definir un modelo de entorno, un modelo de roles y un modelo de interacción). El proceso entonces continua con la **fase de diseño detallado**, en la cuál se produce una especificación de un MAS detallado (en términos de un modelo de agente, un modelo de servicios y un modelo de conocidos) para que finalmente pueda ser implementado usando un marco de trabajo para la programación de agentes **[Zambonelli, Jennings, and Wooldridge, 2005]**. Este modelo , sus relaciones y procesos son representados en la **figura 4.3**.

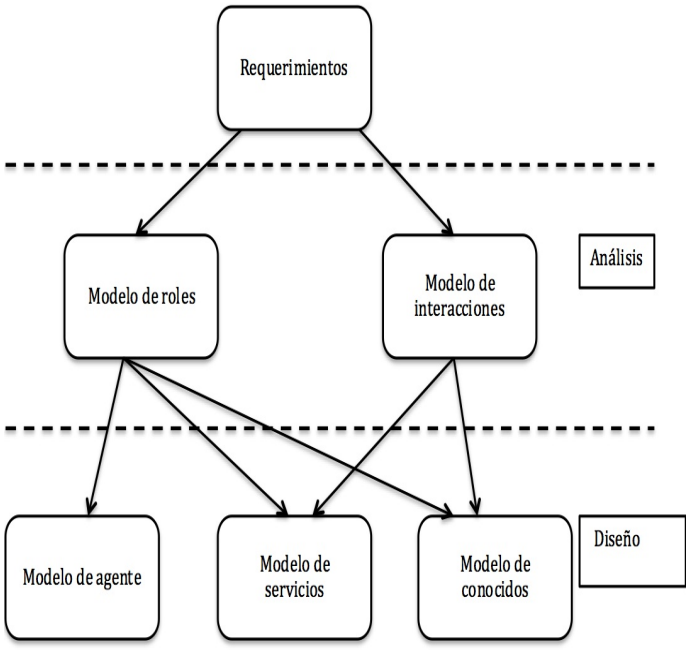


Figura 4.3: **Modelos de la metodología Gaia y sus relaciones en el proceso Gaia**. Basado en **Wooldridge, Jennings, and Kinny, 2000**:

4.5.1.1. Fase de análisis

El objetivo de la fase de análisis es organizar la recolección de especificaciones y requerimientos para modelar el entorno del sistema. En la metodología Gaia esta

es la fase donde se establecen lo modelos de roles y modelos de interacción para todo el sistema **[Zambonelli, Jennings, and Wooldridge, 2005]**.

En esta fase se especifícan los siguientes componentes **[Wooldridge, Jennings, and Kinny, 2000]**:

- **Modelo de Roles.** Se identifican características del sistema como actores y sus objetivos, así como los roles y protocolos del sistema. Para representar roles, Gaia los describe con cuatro principales clases de atributo:
  - **Permisos.** Estos tienen como meta identificar aquellos recursos que pueden ser usados legítimamente para llevar a cabo el rol; y establecer los límites que tienen estos recursos dentro del rol en el que operan.
  - **Responsabilidades.** Determinan el comportamientos que se espera de cada rol, y cómo serán los atributos claves asociados con el rol: Gaia divide las responsabilidades en dos tipos: propiedades *liveness* (intituitivamente indica "algo bueno que pasa") y propiedades *safety* (intituitivamente indica "nada malo sucede"). Las primeras describen casos de estados que un agente pueden tener, dando condiciones de certeza. Las segundas indican que un caso de estado es mantenido.
  - **Actividades.** Las *actividades* de un rol son operaciones asociadas con el rol que puede llevar a cabo un agente sin interactuar con otros agentes.
  - **Protocolos.** Definen la manera en que los roles pueden interactuar con otros roles.
- **Modelos de Interacción**. En Gaia éste modelo captura las dependencias y relaciones entre varios roles en un MAS en términos de la definición de un protocolo para cada tipo de inteacción entre roles. Un protocolo consiste de los siguientes atributos **[Zambonelli, Jennings, and Wooldridge, 2005]**:
  - *Nombre de protocolo:* descripción breve textual que captura la naturalza de la interacción. Ej. "solicitud de información","asignar tarea Y".
  - *Iniciador:* el rol o roles responsables de empezar la interacción;
  - *Socio:* el rol o roles con la respuesta con la cuál interactúan con el Iniciador;
  - *Entradas:* información usada por el rol iniciador mientras se promulga el protocolo;

- *Salidas:* información que porporciona el protocolo de respuesta durante la interacción; y
- *Descripción:* la descripción textual que explica el propósito de el protocolo y el proceso de actividades implicadas en la ejecución.

4.5.1.2. Fase de diseño detallado

Una vez que se han tomado todos los requerimientos funcionales del sistema en la fase de análisis, éstos pasan a la fase de diseño detallado donde se decide que es lo que tiene que hacer el MAS. Las actividades que llevará a cabo el sistema son diseñadas (de acuerdo a los modelos de roles e interacción) y refinadas. Al empezar el diseño de las actividades es posible identificar especificaciones faltantes o incompletas o requerimientos que son conflictivos **[Wooldridge, Jennings, and Kinny, 2000]**.

En esta fase se especifican los siguientes componentes **[Zambonelli, Jennings, and Wooldridge, 2005]**.:

- **Modelo de agente.** Desde el contexto de Gaia, un agente es una entidad de software que lleva a cabo un conjunto de roles. De este modo en el modelo de agente se identifican las clases de agente que pueden llevar a cabo roles específicos y cuántas instancias de cada clase tienen que ser instanciadas en el sistema real.
- **Modelo de servicios**. La meta en este modelo es identificar los *servicios* asociados con cada clase de agente, ó equivalentemente, con cada uno de los roles que serán llavados acabo por las clases de agente. De este modo el modelo de servicios se aplica en el caso de asignación de roles estáticos a las clases de agentes y en el caso donde los agentes pueden dinámicamente asumir roles.
- **Modelos de conocidos**. Este modelo simplemente define las ligas de comunicación que existen entre las diferentes clases de agente. Ellos **no** definen que mensajes son enviados ó cuando los mensajes son envíados, simplemente indican que caminos de comunicación existen.

4.5.2. Agent Unified Modeling Language (AUML)

Otra metodología que está teniendo mucha aceptación es AUML. AUML extiende las representaciones de UML para representar *protocolos de interacción de agentes* (AIP) y otras nociones básicas usadas con agentes **[Odell, Parunak, and Bauer, 2000]**:

AIP describe un protocolo de comunicacion de patrones como una secuencia permitida de mensajes entre los agentes y las restricciones de los contenidos de estos mensajes. FIPA ha especificado muchos protocolos tales como **[FIPA, 2003]**:

- Protocolo de Solicitud.
- Protocolos de Redes de contrato.
- Protocolos de Redes de contrato iteradas.
- etc, ...

AUML adopta una serie de capas para los protocolos **[Odell and Parunak, 2001]**:

- Nivel 1: Representa los protocolos generales (en diagramas de secuencia, paquetes y plantillas).
- Nivel 2: Representa las interacciones entre agentes (diagramas de secuencia, actividad, y diagrarnas de estado).
- Nivel 3: Representa procesamiento interno de los agentes (diagramas de estados y actividades).

En el **capítulos 5** se describirá mas a detalle ésta metodología de modelado la cual es impulsada por FIPA como un medio para modelar las actividades en el lenguaje de programacion JADE.

4.6. Herramientas para la construcción de Sistemas multiagentes

Los sistemas multi-agentes pueden ser elaborados por medio de un lenguaje programación. En particular, los lenguajes orientados a objetos son considerados como medio adecuado porque el concepto de *agente* no es tan distante del término *objeto*. De hecho comparten muchas propiedades con los objetos, tales como, encapsulación, y frecuentemente heredan y pasan mensajes. Pero éstos términos también difieren en que los objetos no son autónomos y no son capaces de tener un comportamiento flexible, y que cada agente en un sistema tiene su propio hilo de control **[Nwana and Wooldridge, 1996]**.

En esta sección se presenta de manera general algunas plataformas y lenguajes de programación para el desarrollo de sistemas multi-agentes.

4.6.1. Lenguajes de programación para el desarrollo de agentes

Un nuevo paradigma de programación que ha surgido son los lenguajes de programación orientados a agentes, que son una nueva clase de lenguajes de programación que se enfocan en tomar en cuenta las principales características de los sistemas multi-agentes **[Sterling and Taveter, 2009]**. Estos lenguajes incluyen algunas de las estructuras usadas por los agentes como agencia, creencias, objetivos, planes, roles y normas **[Shoham, 1990]**. Algunos de los lenguajes de programación que existen se mencionan a continuacIón.

4.6.1.1. JACK

JACK es un marco de trabajo en Java para la creación de sistemas multiagentes. JACK consiste de tres extensiones para el lenguaje de programación Java. La primera extensión es un conjunto de adiciones sintácticas para su servidor de lenguaje. Estas adiciones pueden ser divididas como sigue **[Howden, Rönnquist, Hodgson, and Lucas, 2001]**:

- Un número de palabras clave para identificar los principales componentes de un agente (tales como *agente*, *plan* y *evento*).
- Un conjunto de sentencias para la declaración de atributos y sus características de los componentes (por instancia, la información contenida en creencias o llevadas por eventos).
- Un conjunto de sentencias para definir interrelaciones (por instancia, cuales planes pueden ser adoptados para reaccionar a cierto evento).
- Un conjunto de sentencias para manipular el estado de los agentes (por instancia, adiciones de nuevos objetivos ó sub-objetivos para ser logrados, cambiando creencias con otros agentes).

La segunda extensión para Java es un compilador que convierte las adiciones sintácticas descritas anteriormente en clases y sentencias de Java, estas pueden ser cargadas y llamadas por otro código de Java. Además el compilador también parcialmente transforma el código de los planes con el fin de obtener las semánticas correctas de la arquitectura BDI **[Howden, Rönnquist, Hodgson, and Lucas, 2001]**.

Finalmente JACK contiene un conjunto de clases (llamadas kernel) que proveen los requerimientos para el tiempo de ejecución y la generación de código. Que incluye **[Bussete, Ronnquist, Hodgson, and Lucas, 1999]**:

- administración automática de tareas concurrentes entre agentes llevadas acabo en paralelo (*Intenciones* en la terminología BDI).
- comportamiento por defecto de los agentes en reacción a eventos, fallas de acciones y tareas, etc; y una
- infraestructura nativa de comunicación de alto rendimiento y peso ligero para aplicaciones multi-agentes.

Es importante señalar que el kernel de JACK soporta múltiples agentes dentro de un sólo proceso, múltiples agentes através de muchos procesos, y una mezcla de los dos. Esto es particularmente conveniente para el ahorro de recursos del sistema. Por instancia, los agentes que llevan a cabo unas cuantas operaciones o comparten mucho de su código o datos pueden ser agrupados juntos.

En general las entidades de las cuales está compuesto JACK son las siguientes **[Winikoff, 2005]**:

- **Agente:** Un agente es una entidad básica para un lenguaje orientado a agentes. En JACK los agentes están especificados por una definición de eventos que ellos pueden manejar y enviar, los datos (incluidos en los conjuntos de creencias (*beliefsets*)) que tienen, y los planes y capacidades que usan.
- **Conjunto de creencias (*beliefsets*):** Un conjunto de creencias (*beliefsets*) es una pequeña base de datos relacional que está almacenada en memoria, preferiblemente que en disco. JACK hace fácil definir esta y definir las consultas al conjunto de creencias. Los conjuntos de creencias pueden también publicar eventos en ciertas situaciones (p.e., cada vez que se modifica el conjunto de creencias).
- **Vista:** Las vistas son conjunto de creencias "virtuales" que son obtenidas de otros conjuntos de creencias.
- **Evento:** Un evento es una ocurrencia en el tiempo que representa algún tipo de cambio que requiere una respuesta. Los eventos son usados en JACK (y en otras arquitecturas BDI) para modelar mensajes que son recibidos, nuevos objetivos que son adoptados, e información que está siendo recibida del entorno.
- **Plan:** Un plan es una "receta" para tratar con un tipo de evento dado. Los planes incluyen una indicación de cuales eventos pueden manejar, una *condición de contexto* que describe en que situaciones el plan puede ser usado, y un cuerpo del plan. El cuerpo del plan, el cual puede ser incluido en el código Java así como en el código JACK, es lo que realmente se ejecuta en el sistema.

- **Capacidad:** Una capacidad es un módulo que contiene las habilidades que un agente puede tener. Las capacidades contienen planes, creencias y la especificación de eventos que los agentes pueden manejar y publicar. En adición a esto, las capacidades también pueden tener sub-capacidades las cuales permiten que estructuras de modulo jerárquicas que son consideradas apropiadas.

4.6.1.2. JADE

JADE es una plataforma de software que provee funcionalidades de una capa middleware[4] desarrollada por TILAB[5] para el desarrollo de aplicaciones distribuidas multi-agentes basada en una arquitectura de comunicación entre pares (P2P). En este entorno entre pares cada agente es un par que potencialmente necesita iniciar una comunicación con otros agentes, así como el de ser capaz de proveer servicios para el resto de los agentes. El modelo de comunicación empleado por esta plataforma está basado en tres características las cuales son muy importantes para cualquier sistema basado en agentes **[Bellifemine, Caire, Poggi, and Rimassa, 2003]**:

- Los ***agentes son entidades activas, que pueden decir 'no', y son libremente acopladas.*** Los agentes forman un conjunto de interrelaciones basados en un tipo de comunicación asíncrona para el paso de mensajes en vez de hacer llamadas a procedimientos remotos. Si un agente desea comunicarse sólo tiene que envíar un mensaje a cierto destino. Esta modalidad de comunicación, permite a los receptores seleccionar los mensajes que le sirven y cuales puede descartar, así como seleccionar que mensajes recibe primero y cuales puede dejar al último. Además permite a los emisores tener control de su propio hilo de ejecución y no ser bloqueados hasta que el receptor lea y atienda el mensaje. Finalmente, se remueve cualquier dependencia temporal entre el emisor y el receptor, esto es, el receptor puede no estar disponible en el momento que el emisor envía el mensaje, o incluso no existir al momento de envíar el mensaje, o incluso si este no es conocido por el emisor que, esta vez, define la intencionalidad de los receptores (i.e., todos los agentes que están interesados en 'football') o media la comunicación através de un proxy (i.e., propaga este mensaje a todos los agentes en el dominio X).

[4]El término *middleware* describe todas las librerias de alto nivel que facilitan y hacen más efectivo el desarrollo de aplicaciones, proporcionando servicios genéricos útiles no sólo para una aplicación, sino para una variedad de aplicaciones, para instancias de comunicación, accesso de datos, codificaciones, control de recursos. Estos mismos servicios son provistos por el sistema operativo, pero la idea detrás del middleware es proveerlos de mejor manera, independientemente del SO.

[5]Telecom Italia Lab.

- ⋆ Los ***agentes llevan a cabo acciones y la comunicación es sólo un tipo de acción.*** Hacer que la comunicación se establezca en el mismo nivel de acciones permite que un agente, por instancia, razone acerca de un plan que incluye acciones físicas (i.e., voltear a la izquierda) y acciones comunicativas (i.e., pedir abrir la puerta). Con el fin de hacer la comunicación planeable, los efectos y precondicionaes de cada posible comunicación necesitan ser claramente definidas.

- ⋆ La ***comunicación tiene un significado semántico.*** Cuando un agente es el objeto de una acción comunicativa (i.e., cuando recibe un mensaje), este debe poder apropiadamente entender el significado de esa acción y, en particular, porque esa acción ha sido llevada a cabo (i.e., la intención comunicativa de el emisor del mensaje).

Desde un punto funcional y de aplicación JADE provee las siguientes servicios **[Bellifemine, Caire, Poggi, and Rimassa, 2003]**:

- Cada agente es JADE puede **dinámicamente descubrir** otros agentes y comunicarse con ellos de acuerdo a una arquitectura peer-to-peer.

- Cada agente es identificado por un nombre único y provee un conjunto de servicios. Este agente puede registrar y modificar sus servicios dentro de la plataforma, y/ó buscar agentes que proveen un servicio dado. Tiene control de su ciclo de vida, y, en particular se comunica con otros agentes.

- Desde el punto de vista de **seguridad** en la comunicación, JADE provee mecanismos para autentificar y verificar **"derechos"** asignados a los agentes. Cuando sea necesario, de este modo, una aplicación puede verificar la identidad del emisor de un mensaje y prevenir acciones que no son permitidas llevar a cabo (por instancia, un agente puede permitir los mensajes recibidos del agente que representa al jefe, pero no enviarle mensajes a él).

- Los mensajes intercambiados entre agentes puede ser puestos dentro de una *envoltura* incluyendo sólo la información requerida por la capa de transporte. Esto permite, entre otras cosas, encriptar el contenido de un mensaje separadamente de la envoltura.

- La estructura de un mensaje cumple con el lenguaje ACL (*Agent Communication Language*) definido por FIPA, el cual incluye variables que indican el contexto de un mensaje (se refiere al tiempo de espera que puede ser permitido antes de recibir una respuesta), esto tiene como objetivo soportar interacciones complejas y múltiples conversaciones en paralelo. Además para soportar la implementación de **conversaciones complejas**, JADE provee un conjunto de estructuras de patrones de interacción típicas que llevan a cabo tareas de negociación, subastas y tareas de delegación.

- Para facilitar la creación y manejar el **contenido** de los mensajes, JADE

proporciona soporte para convertir automáticamente mensajes a un formato adecuado para el intercambio de contenido, incluyendo XML y RDF, o a un formato adecuado para la manipulación de contenido (i.e., objetos de Java,). Si los programadores desean tambien pueden implementar nuevos lenguajes de contenido para cubrir requerimientos específicos **[Bellifemine, Caire, and Greenwood, 2007]**. En JADE los objetos de Java son manejados por instancias especiales conocidas como *ontologías* que soportan de manera adecuada el intercambio de mensajes entre agentes.

- Para incrementar la **escalabilidad** o también para conocer las restricciones del entorno en cuanto a la limitación de recursos, JADE proporciona la oportunidad de ejecutar múltiples tareas paralelas dentro del mismo hilo de ejecución de Java. Varias tareas elementales, tales como la comunicación, pueden ser combinadas para formar tareas más complejas estructuradas como una Máquina de Estados Finitos concurrente.

- JADE proporciona soporte para la **movilidad de código y del estado de ejecución**. Esto es, un agente que está en ejecución en algún host puede pararse para migrar a un host remoto diferente (sin la necesidad de tener que codificar el agente ya instalado en ese host), y reiniciar su ejecución en el punto donde fue interrumpido. Esta funcionalidad permite por ejemplo, distribuir la carga computacional en tiempo de ejecución moviendo agentes para disminuir la carga de las máquinas sin que exista ningún impacto en la aplicación.

- La plataforma de JADE proporciona un servicio de nombres (para asegurar que cada agente tiene un nombre único) y un servicio de **páginas amarillas** que pueden ser distribuidas através de múltiples hosts. Gráficas de federación pueden ser creadas con el fin de definir dominios estructurados de servicios de agentes.

- Una característica muy importante de JADE es que proporciona herramientas gráficas para el soporte de **depuración** y fases de **administración/monitoreo** del ciclo de vida de la aplicación. Por medio de estas herramientas es posible controlar agentes remotamente, incluso si ya están desplegados y ejecutandose. Con estas herramientas aparte conversaciones entre agentes pueden ser emuladas, intercambio de mensajes pueden ser seguidos, tareas pueden ser monitoreadas y el ciclo de vida de los agentes ser controlado.

4.6.2. Otros herramientas para el desarrollo de agentes

A parte de JACK y JADE otras herramientas para la creación de agentes también han sido desarrolladas tal como son AgentBuilder, MadKit, ZEUS, AgentSpeak,

entre otras. Algunas características de estas herramientas se listan a continuación:

- **AgentBuilder** proporciona herramientas gráficas para soportar todas las fases del proceso de construcción de agentes. La programación de agentes de software se lleva a cabo especificando conceptos intuitivos tales como creencias, compromisos, reglas de comportamiento y acciones de los agentes. Para modelar la comunicación de los agentes se usan herramientas de protocolo que automáticamente construyen las reglas de comportamientos requeridas para implementar conversaciones **[Acroymics, 2004]**.

- **MadKit** es una plataforma construída de acuerdo al modelo AALAADIN. El modelo AALAADIN no es un metodología específica, sino más bien un meta-modelo que describe la organización de los agentes usando los conceptos de grupo, agente y rol. En adición a estos conceptos, esta plataforma añade otros tres principios de diseño como son **[Sohail, 2005]**:

  - **Arquitectura de micro-kernel**. Este micro-kernel es un pequeño y optimizado kernel de agente optimizado. El kernel en sí es envuelto en un agente especial *KernelAgent* el cual controla y monitorea el kernel dentro del modelo de agente.

  - **Servicios de identificación de agentes**. Estos diferentes servicios son representados en la plataforma como roles en algunos grupos específicos, definidos en una estructura organizacional.

  - **Modelo de componente gráfico**. El modelo gráfico de MADKIT esta basado en componentes gráfico independientes, que usan la especificación Java Beans en su versión estandar. En este modelo gráfico cada componente es un agente que sólamente es responsable de su propia interface gráfica, acciones y eventos.

- En **ZEUS** cada agente consiste de una capa de definición, una capa organizacional y una capa de coordinación. La *capa de definición* comprende las habilidades de razonamiento (y aprendizaje) de los agentes, habilidades, sus objetivos, recursos, experiencias, creencias, preferencias, etc. La *capa de organización* describe las relaciones de un agente con otros agentes. En la *capa de coordinación* el agente es modelado como una entidad social, i.e., en términos de la coordinación y técnicas de negociación que posee **[Nwana, Ndumu, Lee, and Collis, 1998]**.

- **AgentSpeak** es un lenguaje de programación basado en una arquitectura BDI. Un agente en AgentSpeak esta definido por un conjunto de *creencias* las cuales contituyen la *base de creencias* del agente, y un conjunto de planes los cuales forma su *plan de biblioteca*. Un plan en AgentSpeak tiene una *cabecera* la cual consiste de un evento disparador (el evento o eventos para el cual o cuales ese plan es *relevante*), y una conjunción de creencias literales

> que representan un *contexto*. Un plan también tiene un *cuerpo*, el cual es considerado una secuencia de acciones básicas (i.e., operaciones atómicas que el agente puede llevar a cabo para cambiar su entorno), objetivos que el agente tiene que lograr (o probar) cuando el plan es ejecutado, y cambios de creencias. Los planes son disparados por la adición o borrado de creencias debido a la comunicación o percepción del entorno, o debido a la adición o borrado de objetivos como resultado de la ejecución de planes disparados por eventos previos **[Bordini and Hübner, 2005]**.

Como se ha visto hasta el momento muchos de estas plataformas basan su estructura en algún tipo de arquitectura. En el caso de JACK y AgentSpeak estos basan toda su construcción en un arquitectura BDI. Algunas incoporan herramientas gráficas para el diseño de los agentes como son MadKit, AgentBuilder y la API integrada a ZEUS. Por otra parte en JADE no se pone enfásis a como esta modelado internamente el conocimiento, sino que más bien este marco de trabajo es diseñado de acuerdo a requerimientos externos de los agentes (que establece FIPA), como es la comunicación con el entorno y con otros agentes **[Laclavík, Balogh, Babík, and Hluchy, 2006]**.

Entre todas las plataformas antes mencionadas JADE en los últimos 10 años ha madurado proporcionando un conjunto de herramientas que facilitan el desarrollo de agentes. Además ha sido considerado como una excelente opción debido a que cumple con especificaciones (FIPA) que son consideradas estándares. En la siguiente sección se describirá como desarrollar sistemas multi-agentes con JADE, enfocándose en las características básicas que este marco de trabajo proporciona. Está descripción será la base para la aplicación que será implementada en el **capítulo 5**.

4.7. Programación con JADE

Como ya se ha visto JADE es una plataforma que contiene muchas funcionalidades para el desarrollo de sistemas multi-agentes de acuerdo a un conjunto de estándares establecidos por FIPA. En esta sección se describirá como está estructurada la arquitectura de JADE, como crear agentes en JADE, como incorporar comportamientos a los agentes y como éstos se pueden comunicar.

4.7.1. Arquitectura de JADE

JADE es una plataforma que está compuesta de *contenedores* que pueden ser distribuidos sobre la red. Éstos contenedores representan procesos de Java en los cuales JADE proporciona los servicios necesarios para alojar y ejecutar los

agentes. Cuando la plataforma de JADE es ejecutada el primer contenedor que debe ser inicializado dentro de la plataforma es un contenedor especial llamado *main-container*, y todos los otros contenedores deben de ser registrados por él en el momento de ser ejecutados. Para identificar estos contenedores, JADE utiliza una lógica de nombres simple, por ejemplo, el contenedor por defecto (*main-container*) es nombrado `'Main Container'` mientras que los otros contenedores son nombrados `'Container-1'`, `'Container-2'`, etc **[Grimshaw, 2010]**.

En JADE el contenedor *main-container* tiene las siguientes responsabilidades **[Bellifemine, Caire, and Greenwood, 2007]**:

- Administrar la tabla de contenedores (CT), la cual es la que registra los objetos de referencia y direcciones de transporte de todos los nodos contenedores que componen la plataforma de JADE.

- Administrar la tabla descriptora de agentes globales (GADT), la cual registra todos los agentes presentes en la plataforma, incluyendo su estatus actual y localización.

- Alojar los agentes especiales **AMS** (Agent Management System) y **DF** (Directory Facilitator). Estos son encargados de las siguientes funciones:

    1. El **Sistema Administrador de Agentes (AMS)** es el agente que se encarga de controlar toda la plataforma. Entre las funciones que tiene está el de administrar el ciclo de vida de los agentes, manteniendo una lista de todos los agentes que actualmente viven en la plataforma (páginas blancas), además es el único que puede crear y destruir agentes, así como destruir contenedores y parar la plataforma. Cada agente requiere ser registrado en el AMS (esto automáticamente es llevado a cabo por JADE cuando es inicializado) con el fin de obtener un AID (*Agent Identifier*) válido.

    2. El **Directorio Facilitador (DF)** es el agente que implementa el servicio de páginas amarillas, que son usadas por cualquier agente que desea registrar sus servicios o buscar otros servicios disponibles. El DF de JADE también acepta subscripciones de agentes que desean ser notificados de cualquier registro de servicios ó servicios modificados que coinciden con algún criterio especificado. Múltiples DFs pueden ser inicializados concurrentemente con el fin de distribuir servicios de páginas amarillas através de varios dominios. Éstos DFs pueden ser federados, si es requerido, estableciendo un cruce de registros los unos con los otros permitiendo la propagación de agentes através de la federación entera.

Para evitar que el *main-container* pueda caer en un cuello de botella, JADE provee una caché de la GATD que cada contenedor administra localmente. Las

operaciones de la plataforma, que no involucren al *main-container*, pueden ser llevadas a cabo en la caché local y en los dos contenedores en los cuales residen los agentes que son el sujeto y objeto de la operación (i.e., el emisor y receptor del mensaje). Cuando un contenedor debe descubrir donde vive el receptor del mensaje, el primero busca en su LADT (tabla descriptora de agentes local) y si la búsqueda no es exitosa, el *main-container* es contactado con el fin de obtener la apropiada referencia remota, la cual consecuentemente es guardada localmente en caché para futuros usos **[Bellifemine, Caire, and Greenwood, 2007]**.

JADE aparte de los servicios del AMS y DF, proporciona otro el cual permite que se puedan intercambiar mensajes dentro de la plataforma o entre múltiples plataformas[6], este es el **Servicio de Transporte de Mensajes** (MTS). El propósito de este servicio es el de permitir a los agentes intercambiar mensajes de acuerdo a las especificaciones de FIPA **[Bellifemine, Caire, and Greenwood, 2007]**. Para ello incorpora un conjunto de protocolos de transporte y una codificación estándar para el mensaje a enviar. Por defecto, JADE siempre se ejecuta con un MTP (Protocolo de Transmisión de Mensajes) basado en HTTP[7] el cual es inicializado con el *main-container*. Este una vez inicializado crea un servidor socket en la términal del *main-container* el cual escucha todas las conexiones con protocolo HTTP. Si la conexión entrante es válida el MTP rutea el mensaje a su destino final, el cual puede estar localizado dentro de la plataforma o en otra plataforma remota. En el caso cuando los mensajes son intercambiados por agentes que viven en distintos contenedores dentro de la misma plataforma, JADE incorpora un MTP exclusivo conocido como **Protocolo de Transporte de Mensajes Interno** (IMTP), el cual es usado sólo para comunicación interna dentro de la plataforma, permitiendo así mejorar el desempeño de la misma. Entre otras funciones del IMTP, este transporta comandos necesarios para administrar la plataforma distribuidamente así como el monitorear el estatus de los contenedores remotos que son inicializados en la plataforma.

En general como como se relacionan todos estos los elementos que forman la arquitectura de JADE se muestra en la **figura 4.4**.

[6]Una plataforma de JADE puede ser ejecutada dentro de una términal ó inclusive en diferentes términales tener una plataforma de JADE (una por términal). Es posible tener dos o más plataformas viviendo en la misma términal pero es necesario proporcionar un puerto distinto para cada plataforma que se ejecute.

[7]JADE también proporciona otros protocolos de transporte como IIOP el cual fue desarrollador por el equipo de JADE, JMS (Servicio de Mensajes de Java), Java RMI, pero aparte permite añadir otros, si es necesario.

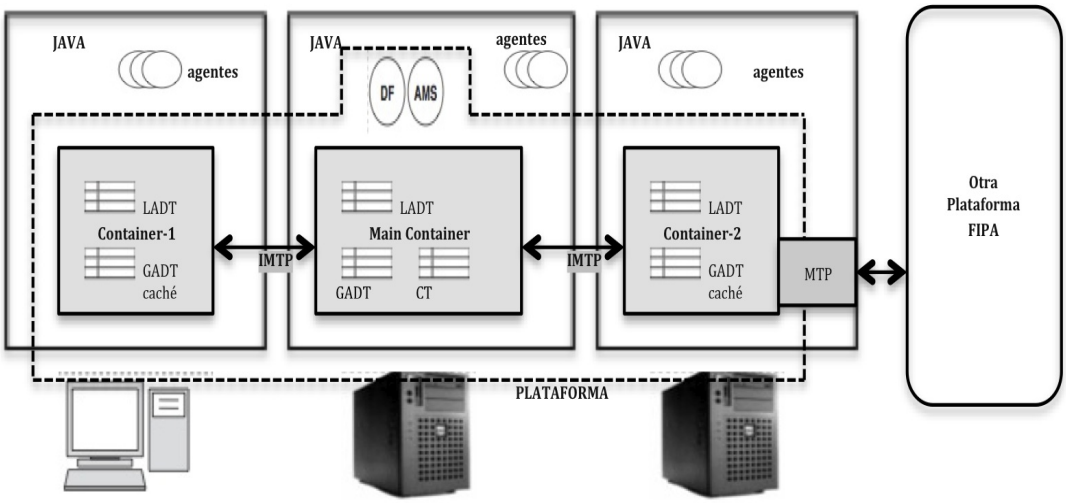


Figura 4.4: **Relación entre los elementos de la arquitectura principal de JADE**. Adaptado de **Bellifemine, Caire, and Greenwood, 2007**.

4.7.2. Inicialización de la Plataforma de JADE

Para poder inicializar JADE en alguna términal, es necesario obtener su software descargándolo de su página web[8]. Éste software es un paquete que contiene los archivos necesarios para la ejecución de la plataforma, así como un conjunto de tutoriales y ejemplos de código Java con JADE **[Caire, 2009]**.

Entre todos los archivos que contiene el software de JADE, es necesario extraer el archivo *jadeSrc.zip* el cual contiene las bibliotecas que tienen que ser incluidas en el CLASSPATH del sistema operativo. Entre estas bibliotecas esta el *jade.jar* que incluye todos los paquetes de JADE, los MTPs y sus herramientas gráficas; asi como, la biblioteca *commons-codec.jar* la cual contiene un conjunto de codecs[9] usados por JADE.

Una vez que las bibliotecas de JADE han sido incluidas en el CLASSPATH, el *main-container* junto con un GUI de JADE pueden ser inicializados (ver **figura 4.5**). Para ello se usa la siguiente instrucción **[Bellifemine, Caire, and Greenwood, 2007]**:

```
prompt> java jade.Boot -gui
```

Este GUI que es inicializado es una interfáz gráfica provista por JADE llamada **Agente de Monitoreo Remoto** (RMA) el cual permite a la plataforma de JADE administrar y monitorear a todos los agentes que corren dentro de la plataforma. Además como se mencionó previamente en el *main-container* los

[8] http://jade.tilab.com
[9] Codificador-Decoficador

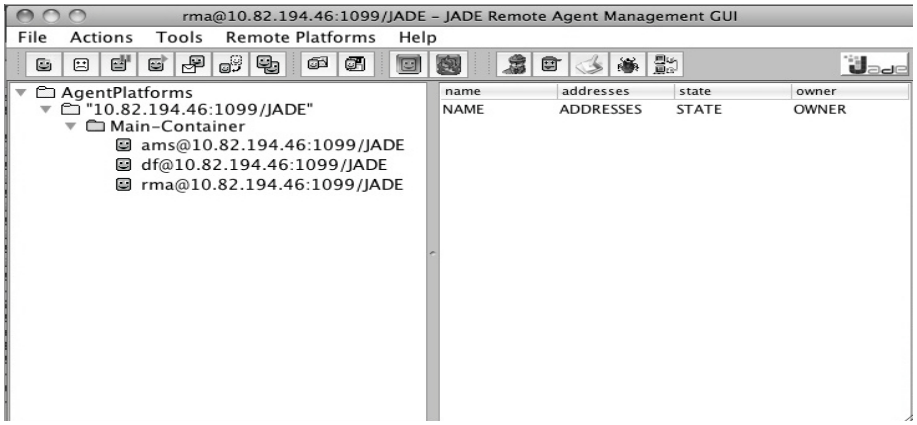


Figura 4.5: **GUI de JADE**.

agentes AMS y DF también son inicializados.

Una vez que la plataforma con el GUI y el *main-container* han sido inicializadas otros contenedores también pueden ser inicializados dentro de la plataforma. Esto se puede hacer con la siguiente instrucción **[Caire, 2009]**:

```
prompt> java jade.Boot -container <nombre_del_agente>
:<ruta_al_agente.class>
```

Además JADE permite la ejecución de agentes que pueden estar repartidos en diversas términales. Para ejecutar un agente remoto se tiene que ejecutar la siguiente instrucción **[Grimshaw, 2010]**:

```
prompt> java jade.Boot -container -host [nombre_host]
<nombre_del_agente>:<ruta_al_agente.class>
```

4.7.3. Creación de Agentes en JADE

Una vez que se sabe como la plataforma de JADE es inicializada, es necesario conocer como los agentes son creados e inicializados dentro de la plataforma. Para ello se analizará el siguiente código en Java en el cual un agente es definido **[Bellifemine, Caire, and Greenwood, 2007]**:

Código 4.1: Código de un Agente.

```
import jade.core.Agent;

public class HelloWorldAgent extends Agent
{
 protected void setup()
 {
  System.out.println("Hola Mundo. Yo soy un Agente!");
 }
}
```

Un Agente en JADE es una clase de Java que hereda instancias de la clase jade.core.Agent e implementa el método setup() como se muestra en el código 4.1. La clase jade.core.Agent permite representar un objeto de Java como un agente. Pero a diferencia de los objetos normales de Java este objeto no puede ser referenciado por ninguna otra clase de Java. Por otro lado el método setup() tiene el propósito de inicializar los agentes y es donde se incluyen las instrucciones para registrar los servicios que un agente puede proveer en el catálogo de páginas amarillas (por medio del agente **DF**) y empezar los comportamientos de un agente **[Bellifemine, Caire, and Greenwood, 2007]**.

Una vez que un agente es creado, este debe ser compilado e inicializado en la plataforma de JADE. Para compilar el agente del código 4.1, se debe de introducir la siguiente instrucción:

```
prompt> javac <ruta_al_agente>/HelloWorldAgent.java
```

Y para inicializar el agente se debe escribir la siguiente instrucción previamente vista en la **sección 4.7.2**:

```
prompt> java jade.Boot Peter:HelloWorldAgent
```

De acuerdo a las especificaciones de FIPA, cada instancia de agente dentro de la plataforma de JADE debe ser representada por un *'Identificador de Agente'* (AID). Este AID es un nombre con el cual el agente es identificado de forma **única** dentro de la plataforma y representa una instancia de la clase jade.core.AID, la cual proporciona el método getAID() con el cual este identificador puede ser obtenido. La forma de este AID puede ser representada de dos maneras **[Bellifemine, Caire, and Greenwood, 2007]**:

- Como un identificador local: $\langle nombre\_local \rangle$ ó,
- con su nombre completo en forma de identificador global único: $\langle nombre\_local \rangle$**@**$\langle nombre\_plataforma \rangle$**:**$\langle puerto \rangle$**/JADE**.

Como se acaba de observar un objeto AID incluye un identificador global único (GUID). Este GUID es usado cuando agentes que viven en diferentes plataformas necesitan comunicarse. Por otra parte, el nombre local del agente debe asignado por el creador y este debe ser único dentro de la plataforma de JADE. Cuando el agente es inicializado el nombre de la plataforma es añadido al nuevo AID creado para obtener el GUID. Con respecto al `nombre de la plataforma` este representa la dirección de la plataforma donde vive el agente. Si además este agente vive en otras plataformas, las direcciones de estas son incluidas en el `nombre de la plataforma` **[Grimshaw, 2010]**.

Para poder obtener estos identificadores y direcciones al **código 4.1** se le puedan agregar las siguientes líneas **[Bellifemine, Caire, and Greenwood, 2007]**:

Código 4.2: Identificadores de un Agente.

```
protected void setup()
{
 System.out.println("Hola Mundo. Yo soy un agente!");
 System.out.println("Mi nombre local es " + getAID().getLocalName());
 System.out.println("Mi GUID es " + getAID.getName());
 System.out.println("Mis direcciones son: ");
 Iterator it = getAID().getAllAddresses();
 While (it.hasNext())
 {
  System.out.println("- " + it.next());
 }
}
```

Al inicializar el agente, en consola se imprimirá lo siguiente:

```
Hola Mundo. Yo soy un agente!
Mi nombre local es Peter
Mi GUID es Peter@anduril:1099/JADE
Mis direcciones son:
- http://anduril:7779/acc
```

La clase jade.core.AID incorpora los métodos getLocalName() para recuperar el nombre local, getName() para obtener el GUID y getAllAddresses() para obtener las direcciones **[Bellifemine, Caire, and Greenwood, 2007]**.

Si el nombre local del agente es conocido, su AID puede ser obtenido de la siguiente manera:

```
String localname = "Peter";
AID id = new AID(localname, AID. ISLOCALNAME);
```

Similarmente si el GUID del agente es conocido, su AID puede ser obtenido como sigue:

```
String guid = "Peter@plataforma";
AID id = new AID(guid, AID. ISGUID);
```

Para resumir, cuando un agente es inicializado dentro de la plataforma una serie de tareas múltiples son realizadas de manera automática. Las cuales se listan a continuación **[Tangient, 2010]**:

1. Se llama al constructor del agente.
2. Se crea un identificador del agente (AID).
3. Se registra el agente en el AMS.
4. Se ejecuta el método setup(), que debe contener el código relativo a las tareas de inicialización.

4.7.4. Terminación de un Agente

Para terminar la ejecución de un agente se puede hacer de las siguientes maneras **[Tangient, 2010]**:

- **Llamando al método doDelete()**. Este método se utiliza para finalizar agentes cuya ejecución es continua. Para ello invoca al método takeDown().
- **Por invocación del método takeDown()**. Este método se invoca antes de que el agente termine su ejecución. Este generalmente es usado para realizar tareas de "limpieza" como por ejemplo, desregistrar el agente en el DF. Si se requiere que este método provea un comportamiento especial este debe ser proporcionado por el programador.

En general un agente puede verse de la siguiente manera **[Tangient, 2010]**:

Código 4.3: Representación general de un Agente.

```
import jade.core.Agent;

public class MiAgente extends Agent
{
 protected void setup(){// inicialización de MiAgente }
 protected void takeDown(){// liberación de recursos del agente }
}
```

4.7.5. Paso de argumentos en JADE

Cuando un agente es inicializado un conjunto de argumentos pueden ser pasados a este. Para poder recuperarlos el agente los toma como un arreglo de objetos (tipo `Object`), los cuales estan contenidos en el método getArguments() de la clase jade.core.AID **[Caire, 2009]**. Para especificar estos argumentos en la instrucción con la cual se inicializa en la gente, estos deben ir entre parentesis y espacios[10] como se muestra a continuación:

```
prompt> java jade.Boot Peter:HelloWorldAgent(arg1 arg2
arg3)
```

Para recuperarlos desde el código, estos deben ser tomados como objetos genéricos. Modificando el **código 4.2** el método setup() se verá de la siguiente manera **[Bellifemine, Caire, and Greenwood, 2007]**:

[10]Dependiendo de la versión de SO, los argumentos pueden ir separados por comas ó puntos y comas.

Código 4.4: Paso de argumentos a un Agente.

```
protected void setup()
{
 System.out.println("Hola Mundo. Yo soy un agente!");
 System.out.println("Mi nombre local es " + getAID().getLocalName());
 System.out.println("Mi GUID es " + getAID.getName());
 System.out.println("Mis direcciones son: ");
 Iterator it = getAID().getAllAddresses();
 While (it.hasNext())
 {
  System.out.println("- " + it.next());
 }
 System.out.println("Mis argumentos son:");
 Object[] args = getArguments();
 if(args != null)
 {
  for(int i=0; i<args.length; i++)
  {
   System.out.println("-" + args[i]);
  }
 }
}
```

Y en consola se mostrará lo siguiente:

```
Hola Mundo. Yo soy un agente!
Mi nombre local es Peter
Mi GUID es Peter@anduril:1099/JADE
Mis direcciones son:
- http://anduril:7779/acc
Mis argumentos son:
- arg1
- arg2
- arg3
```

4.7.6. Comportamientos de Agentes

En JADE para que un agente pueda llevar a cabo tareas, se incorpora una funcionalidad conocida como *comportamiento* ó *behaviour*. Este comportamiento es implementado como un objeto de una clase que hereda instancias de la clase jade.core.behaviours.Behaviour **[Caire, 2009]**.

Un agente puede ejecutar uno ó varios comportamientos. Para ello cada agente tiene un *planificador* o *scheduler* cuya función es la de gestionar la ejecución de todos los comportamientos que estan activos en el agente. Desde un punto de vista de programación los comportamientos son como los hilos de ejecución de Java, esto quiere decir, que en un agente pueden estar activos tantos comportamientos como sea necesario. Pero en JADE sólo puede correr un sólo hilo de

ejecución por agente[11], por lo que los comportamientos son ejecutados cooperativamente dentro del hilo de ejecución del agente. El que corran cooperativamente significa que un comportamiento cuando es ejecutado no puede ser interrumpido por otro comportamiento hasta que su función principal finalice. De este modo, si es requerido que un agente cambie de la ejecución de un comportamiento a otros es responsabilidad de los programdores definir la ejecución de estos en cada momento. El resultado de todo esto implica un mejoramiento del rendimiento del sistema debido a que sólo se permite un hilo de ejecución por agente ahorrando tiempo de CPU y espacio de memoria **[Bellifemine, Caire, and Greenwood, 2007]**.

4.7.6.1. Añadir y Remover Comportamientos

Para añadir y eliminar comportamientos la clase jade.core.Agent incorpora dos métodos: addBehaviour(Behaviour) y removeBehaviour(Behaviour). Estos permiten la entrada y salida de los objetos *Behaviour* en la cola del planificador.

Cuando un comportamiento va a ser añadido al agente este tiene que inicializado dentro del método setup() del agente o dentro de otro comportamiento, como se muestra a continuación **[Tangient, 2010]**:

Código 4.5: Adición de un comportamiento a un Agente.

```
import jade.core.Agent;
import jade.core.behaviours.*;

public class MiAgente extends Agent
{
 protected void setup()
 {
  //Aquí es donde se añade el comportamiento.
  addBehaviour(new MiComportamiento1());
 }
 //Este es el comportamiento.
 private class MiComportamiento1 extends Behaviour
 {
  public void action()
  {
   System.out.println("Mi nombre es: "+getName() );
   System.out.println("Soy el comportamiento del agente");
  }
  public boolean done()
  {
   return true;
  }
 }
}
```

En JADE crear un comportamiento representa crear una clase privada dentro del agente y asociarlo al agente usando el método addBehaviour(Behaviour) **[Tan-**

[11]Cada agente es un entidad autónoma.

gient, 2010]. Esta clase que hereda de jade.core.behaviours.Behaviour a su vez tiene que implementar dos métodos abstractos: el método action() con el cual se definen las operaciones que son llevadas a cabo cuando el comportamiento está en jecución; y el método done() el cual retorna un valor booleano que indica si un comportamiento ha acompletado su tarea.

También es posible añadir un comportamiento desde otro comportamiento, para ello se hace uso de una variable de la clase jade.core.behaviours.Behaviour llamada **myAgent** que funciona de referencia al agente que está ejecutado el comportamiento, es decir al agente al que pertenece el comportamiento **[Tangient, 2010]**. Esto se ejemplifica a continuación:

Código 4.6: Adición de un comportamiento dentro de otro comportamiento.

```
import jade.core.Agent;
import jade.core.behaviours.*;

public class MiAgente extends Agent
{
 protected void setup()
 {
  //Aquí es donde se añade el comportamiento.
  addBehaviour(new MiComportamiento1());
 }
 //Este es el comportamiento.
 private class MiComportamiento1 extends Behaviour
 {
  public void action()
  {
   System.out.println("Mi nombre es: "+getName() );
   System.out.println("Soy el primer comportamiento");

   myAgent.addBehaviour(new MiComportamiento2());
  }

  public boolean done()
  {
   return true;
  }
 }
 //Éste es el otro comportamiento
 private class MiComportamiento2 extends Behaviour
 {
  public void action()
  {
   System.out.println("Soy el segundo comportamiento");
  }
  public boolean done()
  {
   return true;
  }
 }
}
```

Si es requerido eliminar un comportamientos esto se puede hacer con el método removeBehaviour(Behaviour). En el código ejemplo dos comportamientos son definidos, pero el segundo comportamiento borra el primer comportamiento. Este se muestra a continuación **[Tangient, 2010]**:

Código 4.7: Eliminación de un comportamiento en un Agente.

```
package examples.practica2;

import jade.core.Agent;
import jade.core.behaviours.*;

public class Ejemplo extends Agent
{
 private Behaviour comp;

 // Inicialización del agente
 protected void setup()
 {
  //Creamos un comportamiento: un objeto de la clase MiComportamiento1
  comp = new MiComportamiento1();
  //Aquí es donde se añade el comportamiento.
  addBehaviour(comp);
 }

 //Definición de un comportamiento
 private class MiComportamiento1 extends Behaviour
 {
  // define la acción a ser ejecutada cuando se ejecute el comportamiento.
  public void action()
  {
   System.out.println("Mi nombre es: "+getName() );
   System.out.println("Soy el primer comportamiento");
   // Añade un comportamiento desde otro comportamiento.
   myAgent.addBehaviour(new MiComportamiento2());
  }

  // Determina si el comportamiento ha sido completado o no.
  // Si el comportamiento ha finalizado, este se elimina de la cola de
      comportamientos activos.
  public boolean done()
  {
   return true;
  }
 }

 //Definición de un segundo comportamiento
 private class MiComportamiento2 extends Behaviour
 {
  public void action()
  {
   System.out.println("Soy el segundo comportamiento");
   myAgent.removeBehaviour(comp); //Borramos el primer comportamiento;
  }

  public boolean done()
  {
   return true;
  }
 }
}
```

4.7.6.2. Métodos de un comportamiento

En un comportamiento además de los métodos action() y done(), otros métodos pueden ser implementados dentro de este. Entre estos métodos se encuentran los

siguientes **[Bellifemine, Caire, and Greenwood, 2007]**:

- ⋆ Método block(). Este método permite bloquear un comportamiento hasta que algún acontecimiento ocurra (típicamente, hasta que llegue un mensaje).
- ⋆ Método onStart(). Este método ejecuta una acción justo antes de la ejecución del método action().
- ⋆ Método onEnd(). Se ejecuta al finalizar el comportamiento (después de que el método done() devuelve *true*) y devuelve un entero que representa un valor de terminación del comportamiento.

Para ejemplificar como estos métodos son implementados en el agente, veamos el siguiente ejemplo **[Tangient, 2010]**:

Código 4.8: Eliminación de un comportamiento en un Agente.

```
package examples.practica2;

import jade.core.Agent;
import jade.core.behaviours.*;

public class Ejemplo2 extends Agent
{
 // Inicialización del agente
 protected void setup()
 {
  // Añadir un comportamiento.
  addBehaviour(new MiComportamiento());
 }

 // Definición de un comportamiento
 private class MiComportamiento extends Behaviour
 {
  // Este método se ejecuta justo antes de la ejecución del método action()
  public void onStart()
  {
   System.out.println("Esto se hace cada vez que se inicia el comportamiento"
       );
  }

  // Función a realizar por el comportamiento
  public void action()
  {
   System.out.println("Hola a todos.");

   //lo bloqueamos durante un segundo
   block(1000);
   System.out.println("Despues de 1 segundo");
  }

  // Comprueba si el comportamiento ha finalizado
  public boolean done()
  {
   return true;
  }

  // Se ejecuta antes de finalizar el comportamiento
  public int onEnd()
  {
```

```
      // Hace que el comportamiento se reinicie al finalizar.
      reset();
      myAgent.addBehaviour(this);

      return 0;
    }
  }
}
```

4.7.7. Comportamientos Genéricos

JADE provee otras útilidades que pueden ser heredadas a los comportamientos para llevar a cabo tareas más complejas. Pero en general los comportamientos se dividen en dos grupos: **Comportamientos Primitivos** y **Comportamientos Complejos**. Estos será analizados a continuación.

4.7.7.1. Comportamientos Primitivos

Un comportamiento primitivo es de tipo atómico, esto quiere decir que tareas simples pueden ser realizadas por el. Los tipos de comportamiento primitivos que proporciona JADE son los siguientes **[Bellifemine, Caire, and Greenwood, 2007]**:

- **SimpleBehaviour**. Como su nombre lo indica, este es un comportamiento simple el cual ejecuta un método action().
- **CyclicBehaviour**. Este comportamiento se mantiene activo mientras el agente vive y su método action() se repetirá cíclicamente. Un riesgo que puede existir con este comportamiento es que puede consumir toda la CPU. Dentro de este tipo de comportamientos cíclicos existe el **TickerBehaviour** el cual ejecuta periódicamente una tarea.
- **OneShotBehaviour**. Este comportamiento se ejecuta una sóla vez y muera. Dentro de este tipo de comportamiento existen dos casos especiales: el **WakerBehaviour** el cual se ejecuta una vez hasta que un tiempo especifico haya transcurrido; y el **ReceiverBehaviour** el cual termina cuando recive mensaje.

4.7.7.2. Comportamientos Complejos

Este tipo de comportamiento es una clase abstracta la cual esta compuesta de subcomportamientos, que se pueden ejecutar siguiendo diferentes políticas de planificación. Los tipos de comportamientos complejos que proporciona JADE son los siguientes **[Tangient, 2010]**:

- **ParallelBehaviour**. Como su nombre lo indíca en esta clase los subcomportamientos son ejecutados en paralelo y termina cuando todos son ejecutados o cuando un subcomportamiento especificado termina.
- **SequentialBehaviour**. En esta clase los subcomportamientos son ejecutados uno después del otro y se termina una vez que todos los subcomportamientos han terminado la ejecución de su método action().
- **FMSBehaviour**. Esta clase permite definir una Máquina de estados finita mediante subcomportamientos. Cada subcomportamiento representa un estado de la máquina y las transiciones se van produciendo según la salida de dichos estados.

4.7.8. Comunicación entre agentes

En JADE la comunicación es un paradigma basado en el *pase de mensajes asíncrono* (ver **figura 4.6**). De este modo cada agente tiene un 'buzón' (la cola de mensajes del agente) donde en tiempo de ejecución los mensajes enviados por otros agentes son depositados. En particular el formato de un mensaje en JADE cumple con las especificaciones de FIPA definiendo una estructura de mensaje FIPA-ACL. En esta cada mensaje incluye **[Bellifemine, Caire, and Greenwood, 2007]**:

- El *emisor* del mensaje.
- La lista de *receptores*.
- Los actos comunicativos (también llamados '*performativas*') indicando la intención del emisor del mensaje. Por instancia, si una performativa es de tipo REQUEST, el emisor desea que el receptor efectue una acción, si es de tipo INFORM desea que el emisor este informado de un hecho, si es de tipo PROPOSE o CFP (Call for Proposals), el emisor desea entrar en una negociación.
- El *contenido* con la información real a ser intercambiada por el mensaje (i.e., la acción que va a ser llevada a cabo en un mensaje REQUEST, o el hecho que el emisor del mensaje desea revelar en un mensaje INFORM.
- El *lenguaje* de contenido indicando la sintáxis usada para expresar el contenido. Tanto el emisor como el receptor deben poder codificar y analizar expresiones que cumplan con la sintáxis para que la comunciación se a efectiva.
- La *ontología* que indica el vocabulario de símbolos usados en el contenido. Tanto el emisor como el receptor deben dar el mismo significado a estos

símbolos para que la comunicación sea efectiva.

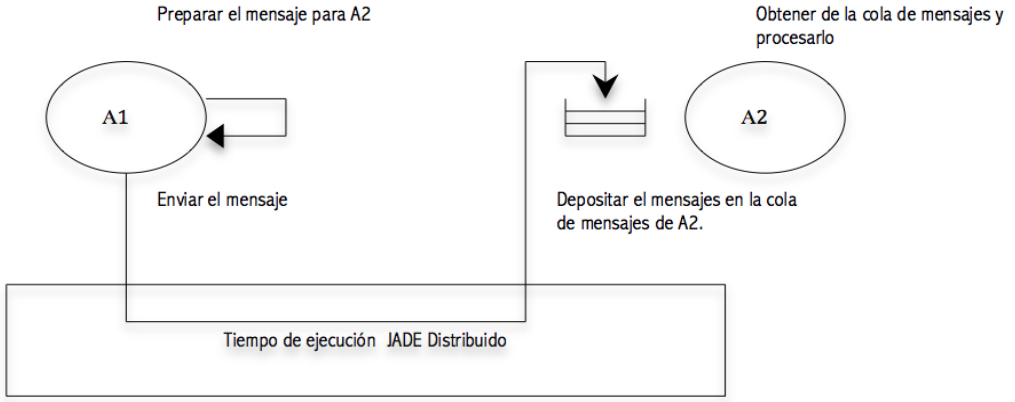


Figura 4.6: Paradigma de Paso de mensajes asíncrono de JADE. Adaptado de **Bellifemine, Caire, and Greenwood, 2007**

JADE implementa un mensaje como un objeto de la clase jade.lang.acl.ACLMessage la cual proporciona una serie de métodos para accesar a todos los campos especificados por el formato ACL. Entre los métodos más importantes que incorpora están los siguientes **[Tangient, 2010]**:

- **setPerformative(int)**: toma como parámetro una constante representativa de un tipo de acción performativa[12] y la establece como performativa del mensaje. Por ejemplo, para hacer que el mensaje *msg* sea de tipo *inform* bastará con escribir: msg.setPerformative(ACLMessage.INFORM).
- **getPerformative()**: Devuelve un entero equivalente a la constante que representa a la performativa del mensaje,
- **createReply()**: Crea un mensaje de respuesta para el mensaje sobre el que es aplicado, poniendo los valores oportunos en campos como receiver, conversation-id, etc.
- **addReceiver(AID)**: Toma como parámetro un AID y lo añade a la lista de receptores,
- **getAllReceiver()**: Devuelve un iterador sobre la lista de receptores.
- **setContent(String)**: Recibe como parámetro una cadena y la pone como contenido del mensaje,
- **getContent()**: Devuelve una cadena con el contenido del mensaje,

[12]Todas las performativas definidas en la especificacion FIPA son mapeadas como constantes en la clase ACLMessage.

4.7.8.1. Envío de Mensajes

Para poder enviar un mensaje en JADE los siguientes pasos deben ser llevados a cabo **[Bellifemine, Caire, and Greenwood, 2007]**:

- Crear un Objeto ACLMessage.
- Usar los métodos de ACLMessage para llenar los campos necesarios.
- Llamar al método send() de la clase jade.core.Agent. El método send() recibe como parámetro un objeto ACLMessage, añadiendo el valor del campo sender (emisor) y envía el mensaje a los destinatarios.

En el siguiente código se observa la creación de un mensaje que informa al agente que se llama *Peter* que el *día de hoy está lloviendo*:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setContent("El día de hoy está lloviendo');
send(msg);
```

El mensaje enviado será el siguiente:

(INFORM
:sender (agent-identifier :name emisor@plataforma:1099/JADE :addresses (sequence http://plataforma:7778/acc))
:receiver (set (agent-identifier :name Peter@plataforma:1099/JADE))
:content "El día de hoy está lloviendo"
:language English)

4.7.8.2. Recepción de Mensajes

Como se menciono previamente, los mensajes en JADE son depositados en una cola de mensajes tan pronto ellos llegan. Para que un agente pueda recoger estos mensajes de la cola, debe hacerlo usando el método receive() que provee la clase jade.core.Agent. Este método retorna el primer mensaje que se encuentra en la cola, o un valor `nulo` si es que la cola está vacía. Una implementación de este método es la siguiente **[Caire, 2009]**:

```
ACLMessage msg = receive()
if(msg != null)
{
// Proceso del mensaje
}
```

Para dar más detalle de como es el intercambio de mensajes entre dos agentes, se mostrarán dos códigos que representan a dos agentes: Emisor y Receptor. En estos, se muestra como los mensajes son implementados en los comportamientos y como el agente Receptor recibe los mensajes. En el caso del emisor este va a mandar un mensaje al receptor que dice: *"Hola que tal receptor?"*. El código emisor es representado como sigue **[Tangient, 2010]**:

Código 4.9: Emisor del mensaje.

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.*;

public class Emisor extends Agent
{
 private class EmisorComportamiento extends SimpleBehaviour
 {
  boolean fin = false;
  public void action()
  {
   System.out.println(getLocalName() +": Preparandose para enviar un mensaje
       a receptor");
   AID id = new AID();
   id.setLocalName("receptor");

   // Creación del objeto ACLMessage
   ACLMessage mensaje = new ACLMessage(ACLMessage.REQUEST);

   //Rellenar los campos necesarios del mensaje
   mensaje.setSender(getAID());
   mensaje.setLanguage("Español");
   mensaje.addReceiver(id);
   mensaje.setContent("Hola, que tal receptor ?");

   //Envía el mensaje a los destinatarios
   send(mensaje);

   System.out.println(getLocalName() +": Enviando hola a receptor");
   System.out.println(mensaje.toString());
   fin = true;
  }

  public boolean done()
  {
   return fin;
  }
 }

 protected void setup()
 {
  addBehaviour(new EmisorComportamiento());
 }
}
```

Y el código del agente Receptor es el que sigue **[Tangient, 2010]**:

Código 4.10: Receptor del mensaje.

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;

public class Receptor extends Agent
{
 private class ReceptorComportamiento extends SimpleBehaviour
 {
  private boolean fin = false;
  public void action()
  {
   System.out.println(" Preparandose para recibir");

   //Obtiene el primer mensaje de la cola de mensajes
   ACLMessage mensaje = receive();

   if (mensaje!= null)
   {
    System.out.println(getLocalName() + ": acaba de recibir el siguiente
        mensaje: ");
    System.out.println(mensaje.toString());
    fin = true;
   }
  }

  public boolean done()
  {
   return fin;
  }
 }

 protected void setup()
 {
  addBehaviour(new ReceptorComportamiento());
 }
}
```

Para inicializar a los agentes en la plataforma de JADE se lleva a cabo de la siguiente manera **[Tangient, 2010]**:

1. Un agente de la clase Receptor llamándolo con el nombre "receptor" (java jade.Boot -container receptor:Receptor).

2. Un agente de la clase Emisor llamándolo con el nombre "emisor" (java jade.Boot -container emisor:Emisor).

El resultado de ejecutar se muestra en pantalla de la siguiente manera:

emisor: Preparandose para enviar un mensaje a receptor
emisor: Enviando hola a receptor
receptor: acaba de recibir el siguiente mensaje:

```
(REQUEST
:sender ( agent-identifier :name emisor1@luna:1099/JADE
:addresses (sequence http://luna:7778/acc ))
:receiver (set ( agent-identifier:name receptor@luna:
1099/JADE ))
```

```
:content "Hola, que tal receptor ?"
:language Español )
```

Nota Importante: Si se hubiera lanzado primero el emisor no se habría obtenido ninguna salida porque al no estar registrado el receptor del mensaje en el AMS, este enviaría un mensaje FAILURE al agente emisor.

Si se da el caso de que el Receptor no recibe ningún mensaje, éste de todos modos se mantendrá ejecutandose consumiendo tiempo de CPU. Para ello JADE incorpora el método block() el cual bloquea un comportamiento del agente hasta que llegue un mensaje ó puede implementar el método blockingReceive() el cual bloquea instantaneamente todos los comportamientos del agente hasta que llegue un mensaje **[Bellifemine, Caire, and Greenwood, 2007]**.

4.7.8.3. Respondiendo Mensajes

Para que un agente pueda responder a un mensaje, esto se puede hacer de dos maneras **[Tangient, 2010]**:

1. **Indicando como receptor al emisor del mensaje anterior**. Modificando el **código 4.9**, que representa al agente Emisor este se verá de la siguiente manera **[Tangient, 2010]**:

Código 4.11: Emisor como receptor de un mensaje.

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.*;

public class Emisor extends Agent
{
 private class EmisorComportamiento extends SimpleBehaviour
 {
  boolean fin = false;
  public void action()
  {
   System.out.println(getLocalName() +": Preparandose para enviar un mensaje
       a receptor");
   AID id = new AID();
   id.setLocalName("receptor");

   // Creación del objeto ACLMessage
   ACLMessage mensaje = new ACLMessage(ACLMessage.REQUEST);

   //Rellenar los campos necesarios del mensaje
   mensaje.setSender(getAID());
   mensaje.setLanguage("Español");
   mensaje.addReceiver(id);
   mensaje.setContent("Hola, que tal receptor ?");

   //Envía el mensaje a los destinatarios
   send(mensaje);
   System.out.println(getLocalName() +": Enviando hola a receptor");
   System.out.println(mensaje.toString());
```

```
    //Espera la respuesta
    ACLMessage mensaje2 = blockingReceive(); // Bloqueando los comportamientos
    if (mensaje2!= null) // hasta que llegue un mensaje.
    {
     System.out.println(getLocalName() + ": acaba de recibir el siguiente
         mensaje: ");
     System.out.println(mensaje2.toString());
     fin = true;
    }
   }

   public boolean done()
   {
    return fin;
   }
  }

  protected void setup()
  {
   addBehaviour(new EmisorComportamiento());
  }
 }
```

Y el agente receptor se modificará de la siguiente manera [**Tangient, 2010**]:

Código 4.12: Receptor contestando mensaje.

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;

public class Receptor extends Agent
{
 private class ReceptorComportamiento extends SimpleBehaviour
 {
  private boolean fin = false;
  public void action()
  {
   System.out.println(" Preparandose para recibir");

   //Obtiene el primer mensaje de la cola de mensajes
   ACLMessage mensaje = receive();
   if (mensaje!= null)
   {
    System.out.println(getLocalName() + ": acaba de recibir el siguiente
        mensaje: ");
    System.out.println(mensaje.toString());

    // Envia contestación
    System.out.println(getLocalName() +": Enviando contestación");
    ACLMessage respuesta = new ACLMessage(ACLMessage.INFORM);
    respuesta.setContent("Bien");
    respuesta.addReceiver(mensaje.getSender());
    send(respuesta);
    System.out.println(getLocalName() +": Envíando Bien a receptor");
    System.out.println(respuesta.toString());
    fin = true;
   }
  }

  public boolean done()
  {
   return fin;
  }
 }
```

```
 protected void setup()
 {
  addBehaviour(new ReceptorComportamiento());
 }
}
```

2. **Haciendo uso del método createReply()**. Este método crea un nuevo mensaje con los atributos emisor y receptor intercambiados y todos los otros atributos establecidos correctamente. Para los códigos que se han visto, sólo será necesario modificar el código del receptor y el código del emisor quedará de la misma manera como el **código 4.11**. Por lo tanto el **código 4.10** modificandolo se verá de la siguiente manera **[Tangient, 2010]**:

Código 4.13: Receptor usando el método createReply().

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;

public class Receptor extends Agent
{
 private class ReceptorComportamiento extends SimpleBehaviour
 {
  private boolean fin = false;
  public void action()
  {
   System.out.println(" Preparandose para recibir");

   // Obtiene el primer mensaje de la cola de mensajes
   ACLMessage mensaje = receive();
   if (mensaje!= null)
   {
    System.out.println(getLocalName() + ": acaba de recibir el siguiente
        mensaje: ");
    System.out.println(mensaje.toString());

    // Envia contestación
    System.out.println(getLocalName() +": Enviando contestación");
    ACLMessage respuesta = mensaje.createReply();
    respuesta.setPerformative(ACLMessage.INFORM);
    respuesta.setContent("Bien");
    send(respuesta);
    System.out.println(getLocalName() +": Enviando Bien a receptor");
    System.out.println(respuesta.toString());
    fin = true;
   }
  }

  public boolean done()
  {
   return fin;
  }
 }

 protected void setup()
 {
  addBehaviour(new ReceptorComportamiento());
 }
}
```

4.7.8.4. Selección de Mensajes

En JADE es posible que un agente pueda seleccionar que tipo de mensaje desea recibir, para ello hace uso de "plantillas" (templates) las cuales son implementadas como instancias de la clase jade.lang.acl.MessageTemplate la cual proporciona una serie de métodos para filtrar los mensajes ACLMesage. Este se utiliza como parámetro en los métodos receive() y blockingReceive(). Algunos métodos que incorpora el tipo de objeto MessageTemplate son los siguientes **[Bellifemine, Caire, and Greenwood, 2007]**:

- **MatchPerformative(*performativa*)**. Donde la performativa puede ser:
  - ACLMessage.INFORM
  - ACLMessage.REQUEST
  - ACLMessage.QUERY_IF
  - ...
- MatchSender(AID). Selecciona al emisor del mensaje. Recibe un atributo de tipo AID.
- MatchProtocol(String). Devuelve el nombre del protocolo que se haya pasado como parámetro.

En el siguiente código se va a filtrar el agente Receptor para que reciba mensajes de tipo REQUEST, en español y procedentes exclusivamente del agente "emisor" **[Tangient, 2010]**:

Código 4.14: Filtrando Mensajes

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class FiltroReceptor extends Agent
{
 private class Comportamiento extends SimpleBehaviour
 {
  boolean fin = false;
  MessageTemplate plantilla = null;
  public Comportamiento ()
  {
   AID emisor = new AID();
   emisor.setLocalName("emisor");

   //Devuelve una plantilla de mensaje que coincida con algún mensaje con un slot :sender dado.
   MessageTemplate filtroEmisor = MessageTemplate.MatchSender(emisor);

   //Devuelve una plantilla de mensaje que coincida con algún mensaje con una perfomativa dada.
   MessageTemplate filtroInform = MessageTemplate.MatchPerformative(
       ACLMessage.REQUEST);

   //Devuelve una plantilla de mensaje que conicida con algún mensaje con una slot :language dado
```

```
      MessageTemplate filtroIdioma = MessageTemplate.MatchLanguage("Español");

      //Operación lógica AND entre dos objetos de esta clase.
      plantilla = MessageTemplate.and(filtroInform,filtroEmisor);
      plantilla = MessageTemplate.and(plantilla,filtroIdioma);
     }

     public void action()
     {
      ACLMessage mensaje = receive(plantilla);

      if (mensaje!= null)
      {
       System.out.println(getLocalName() + ": ha recibido el siguiente mensaje:
           ");
       System.out.println(mensaje.toString());
       fin = true;
      }
      else
      {
       System.out.println(getLocalName() + ":Esperando mensajes...");
       block();
      }
     }

     public boolean done()
     {
      return fin;
     }
    }

    protected void setup()
    {
     addBehaviour(new Comportamiento());
    }
   }
```

Capítulo 5

Análisis e Implementación

Los Sistemas Multi-Agentes están creciendo en popularidad dentro del ambiente de desarrollo debibo a que estos permiten la construcción de sistemas computacionales que pueden trabajar de forma autónoma e inteligente. Sin embargo, estos sistemas pueden presentar dificultades a la hora de su construcción por lo que en años recientes se han incorporado prácticas de ingenieria de software que tienen la intención de automatizar el proceso de desarrollo. Para dar soporte en especial al desarrollo de MAS han surgido varias metodologías de **Ingeniería de Software Orientada a Agentes** (AOSE) las cuales modelan a los agentes como una estructura social en la que se pueden identificar roles que los agentes deben llevar a cabo. Entre estas metodologías las más representativas son: (*i*)Tropos la cual establece una serie de conceptos que son usados por el paradigma de agentes junto con un proceso de desarrollo de software. Nociones como agente, objetivo, tarea y dependencia (social) son usadas para modelar y analizar los requerimientos de software, la arquitectura y la (posible) implementación final del sistema **[Bresciani, Perini, Giorgini, Giunchiglia, and Mylopoulos, 2004]**. (*ii*) Gaia la cual distingue diferentes modelos que estan asociados con las fases de análisis y diseño. Gaia especialmente se enfoca en términos como roles, interacciones, y conocidos para establecer una estructura organizacional **[Moraitis, Petraki, and Spanoudakis, 2003]**. (*iii*) SODA la cual se concentra en métodos de análisis y diseño de aspecto social (inter-agente) y que emplea el concepto de coordinación de modelos **[Omicini, 2000]**. Y (*iv*) MaSE la cual basa su análisis y diseño en términos de objetivos, casos de uso, refinación de roles y construcción de conversaciones **[of Scientific Research]**. También se debe mencionar la metodología (*v*) **AUML** (*Agent UML*) la cual extiende las propiedades de el estándar UML para representar protocolos de interacción entre agentes, los cuales describen modelos de comunicación que permiten una pequeña secuencia de mensajes entre los agentes **[Odell, Parunak, and Bauer, 2000]**.

Por otra parte, han surgido *marcos de trabajo* genéricos que simplifican la cons-

trucción de MAS. Estos *marcos de trabajo* pueden ser descritos como una capa `middleware` sobre la cual operan los agentes. En general estos, están construidos de acuerdo a las especificaciones de FIPA (*Foundation for Intelligent Physical Agents*), la cual es una entidad responsable de normalizar los MAS buscando la interoperabilidad entre ellos **[FIPA, 2005]**. Entre los *marcos de trabajo* que estan construidos de acuerdo a las especificaciones de FIPA están: (*i*) **FIPA-OS** la cual es la primera implementación de código libre de normas de FIPA, y que es totalmente implementada en lenguaje de programación Java que ofrece un conjunto de herramientas basadas en componentes para la construcción de Agentes **[Poslad, Buckle, and Hadingham, 2000]**; (*ii*) **JACK Intelligent Agents** el cual es un ambiente de desarrollo orientado a agentes que extiende Java para implementar comportamientos de agentes. Estos comportamientos son modelados de acuerdo a la arquitectura BDI **[Bussete, Ronnquist, Hodgson, and Lucas, 1999]**. Y (*iii*) **JADE** (*Java Agent DEvelopment Framework*) el cual es un `middleware` que facilita el desarrollo de MAS inteligentes conforme a las normas de FIPA y que es usado también para administrar agentes **[Bellifemine, Caire, and Greenwood, 2007]**.

En este capítulo se propone la implementación de un marco de trabajo para Minería de Datos Distribuida basada en Sistemas Multi-Agentes. Para el proceso de análisis y diseño de el *marco de trabajo* que se propone se han seleccionado las metodologías Gaia y AUML. Una de las motivaciones para usar Gaia es que permite representar de manera sencilla a los agentes como una organización, de manera que sea posible identificar roles, interacciones entre roles, la estructura organizacional de los agentes y las reglas de la organización. Por otra parte AUML proporciona un conjunto de diagramas que son adaptados a la tecnología de agentes que permiten especificar como los agentes interaccionan, a través de protocolos de comunicación y el intercambio de mensajes. Para la etapa de implementación se ha seleccionado el ambiente de desarrollo JADE el cual ofrece un entorno totalmente compatible con FIPA para el desarrollo de agentes, así como un conjuntos de herramientas que nos permiten generar varios agentes en conjunto, inclusive remotamente, además de dar soporte a la ejecución de múltiples comportamientos de forma secuencial o paralela.

5.1. Requerimientos y Especificaciones

Para el *marco de trabajo* propuesto en este capítulo, se han definido los siguientes requerimientos y especificaciones:

- El sistema implementado trabaja en un entorno distribuido, en el cual varios sitios están conectados en red y en donde en cada sitio hay una fuente de datos. Una fuente de datos esta compuesta por un conjunto de archivos planos y bases de datos.

- En cada sitio del sistema residen un número de agentes los cuales están encargados de administrar una fuente de datos, ejecutar tareas y generar modelos de clustering y obtener un conjunto de mediciones.
- El sistema permite efectuar dos tipos de tarea de clustering: K-Medias y Clustering Jerárquico. Además permite que estas tareas se realicen en dos tipos de escenario: centralizado o distribuido. En un escenario centralizado se han definido las siguientes especificaciones:
  - Un usuario puede solicitar llevar a cabo una tarea de clustering. Para esto, inicialmente debe de seleccionar una fuente de datos y la tarea de clustering que desea que se lleve a cabo. Si el usuario selecciona K-Medias, junto con la fuente de datos seleccionada, debe seleccionar el número de clusters que desea que se generen y una métrica la cual establece una medida de proximidad entre dos puntos de datos. Si el usuario selecciona Clustering Jerárquico junto con la fuente de datos seleccionada debe seleccionar un criterio de enlace, el cual determina una medida de distancia entre un conjunto de observaciones y una métrica la cual establece una medida de proximidad entre dos puntos datos.
  - El usuario también puede elegir si desea que se lleve a cabo una transferencia de datos o no. Si esta es llevada a cabo la fuente de datos seleccionada es transferida a un sitio central para ser procesada. En caso contrario, esta fuente de datos es procesada en su respectivo sitio local.
  - El MAS recibe esta solicitud y debe generar un resultado. Para obtener este resultado debe llevar a cabo las siguientes actividades:
    - Determinar costos de rendimiento generales para la tarea a ejecutar y establecer un estatus.
    - A partir de este estatus se tiene que establecer una estrategia para la ejecución de esa tarea. Esta estrategia establece una cantidad de agentes que de forma particular la procesarán.
    - Transferir la fuente de datos seleccionada al sitio central (si es que el usuario seleccionó efectuar esta transferencia).
    - Ejecutar la tarea de clustering y obtener un modelo.
    - Evaluar el modelo de clustering generado. Y,
    - enviar el modelo de clustering final al usuario y las medidas de evaluación generadas.

  Y para el escenario distribuido se han definido estas especificaciones:

- Un usuario puede solicitar llevar a cabo una tarea de clustering. Para esto, inicialmente debe de seleccionar una fuente de datos y la tarea de clustering que desea que se lleve a cabo. Si el usuario selecciona K-Medias, junto con la fuente de datos seleccionada, debe seleccionar el número de clusters que desea que se generen y una métrica, la cual establece una medida de proximidad entre dos puntos de datos. Si el usuario selecciona Clustering Jerárquico junto con la fuente de datos seleccionada debe seleccionar un criterio de enlace, el cual determina una medida de distancia entre un conjunto de observaciones y una métrica, la cual establece una medida de proximidad entre dos puntos datos.

- El MAS recibe esta solicitud y debe generar un resultado. Para obtener este resultado debe llevar a cabo las siguientes actividades:

  - Determinar costos de rendimiento generales para la tarea a ejecutar y establecer un estatus.

  - A partir de este estatus se tiene que establecer una estratégia para la ejecución de esa tarea. Esta estratégia establece la cantidad de agentes que procesarán dicha tarea.

  - Enviar la solicitud de procesamiento y el estatus a cada sitio del sistema.

  - Cada sitio procesa su partición de la fuente de datos distribuida y genera un modelo local.

  - Los modelos locales son transferidos al sitio central y en éste se unen para formar un sólo modelo global.

  - Evaluar el modelo de clustering generado. Y,

  - enviar el modelo de clustering final al usuario y las medidas de evaluación generadas.

- Para evaluar la calidad de los resultados obtenidos en el clustering un agente debe de considerar valores de cohesión y separación para el caso de k-medias. Y en el caso del clustering jerárquico, debe tomar como medida la distancia cofenética para determinar la precisión del clustering.

- El MAS debe de capturar otras medidas de recursos del sistema operativo con el fin de obtener el rendimiento total de los algoritmos procesados en términos de transmisión de datos, acceso a datos y procesos de datos. Estas medidas serán usadas para establecer la estrategia que se usará para ejecutar una tarea de clustering.

5.2. Análisis y Diseño con Gaia

Con el fin de entender como un sistema multi-agentes (MAS) es concebido e implementado para el *marco de trabajo* de minería de datos distribuida propuesto, se empleará la metodología Gaia la cual divide el desarrollo en dos fases: de análisis y diseño **[Moraitis, Petraki, and Spanoudakis, 2003]**.

5.2.1. Fase de Análisis

El objetivo de la fase de análisis, es organizar la recolección de especificaciones y requerimientos para modelar el entorno del sistema. En la metodología Gaia esta es la fase donde se establecen los modelos de roles y modelos de interacción para todo el sistema **[Sánchez-Pi, Carbó, and Molina, 2010]**.

5.2.1.1. Modelo de Roles

Un rol es una descripción de la funcionalidad de un agente, y que incluye cuatro atributos: permisos (recursos que puede usar para llevar a cabo su rol), responsabilidades (comportamientos esperados de el rol), actividades (acciones sin interactuar con otros roles) y protocolos (acciones que involucran la interacción con otros roles) **[Sánchez-Pi, Carbó, and Molina, 2010]**. De acuerdo a los requerimiento y especificaciones previas para el MAS se han identificado los siguientes roles:

- InterfazUsuario. Éste rol permite a los usuarios finales interactuar con el MAS. Es responsable de recibir las solicitudes de los usuarios finales, transmitir las solicitudes al MAS, recibir una respuesta del MAS y enviar una respuesta a los usuarios finales.
- Bitácora. Éste rol se encarga de almacenar la información de una solicitud y la información de rendimiento obtenida por el MAS al procesar la solicitud en la bitácora.
- Coordinador. Éste rol es responsable de coordinar los mensajes que son transmitidos por los agentes. Y las actividades que el sistema multi-agentes llevará a cabo.
- AdministradorAlgoritmos. Éste rol es responsable de coordinar las tareas de clustering y de administrar el ciclo de vida de los agentes que procesan estas tareas.
- RolClustering. Éste rol es el encargado de procesar las tareas de clustering que son solicitadas al MAS.

- AdministradorDatos. Este rol es responsable de accesar a los datos que se encuentran almacendos en diferentes fuentes de datos.

- RolValidación. Este rol evalua la calidad de alguna tarea procesada, en términos de alguna medida de precisión.

- RolRendimiento. Éste rol es responsable de establecer una estrategia para la ejecución de una tarea de clustering. Para establecer esta estrategia evalua alguna mediciones previas almacenadas por el rol de bitácora y determina la mejor estrategia para efectuar esa tarea. Esta estratégia es expresada como un estatus.

El modelo de roles para el sistema multi-agente es presentado en la **tablas 5.1, 5.2, 5.3**.

Rol: `InterfazUsuario` **(IU)**
Descripción: Éste rol permite a los usuarios finales interactuar con el MAS. Es responsable de recibir las solicitudes de los usuarios finales, transmitir las solicitudes al MAS, recibir una respuesta del MAS y enviar una respuesta a los usuarios finales.
Protocolos y Actividades: RecibirSolicitud. EnviarSolicitud. ObtenerResultados. ResponderSolicitud.
Permisos: Leer solicitudes.
Responsabilidades: Viveza: **IU** = (RecibirSolicitud. EnviarSolicitud. ObtenerResultado. ResponderSolicitud) Seguridad: exitosa.

Rol: `Bitácora` **(B)**
Descripción: Éste rol se encarga de almacenar la información de una solicitud y la información de rendimiento obtenida por el MAS al procesar la solicitud en la bitácora.
Protocolos y Actividades: ObtenerSolicitud. ObtenerResultados. ExtraerInformaciónRendimiento. ExtraerInformacionSolicitud. AlmacenarInformaciones.
Permisos: Escribir Bitácora.
Responsabilidades: Viveza: **B** = (ObtenerSolicitud. ObtenerResultados. ExtraerInformaciónRendimiento. ExtraerInformacionSolicitud. AlmacenarInformaciones) Seguridad: Una conexión exitosa debe establecerse con la Bitácora.

Tabla 5.1: **Modelos de roles de Gaia. Adaptado de Moraitis, Petraki, and Spanoudakis, 2003**.

Rol: `Coordinador` **(C)**
Descripción: Éste rol es responsable de coordinar los mensajes que son transmitidos por los agentes. Y las actividades que el sistema multi-agentes llevará a cabo.
Protocolos y Actividades: RegistrarDF. RecibirSolicitud. SolicitarEstrategia. RecibirEstrategia. EstablecerTipoActividad. EnviarActividad. ObtenerResultados. EnviarResultados.
Permisos: Leer Solicitud.
Responsabilidades: Viveza: **C** = (RegistrarDF. RecibirSolicitud. SolicitarEstrategia. RecibirEstrategia. EstablecerTipoActividad. EnviarActividad. ObtenerResultados. EnviarResultados.) **EstablecerTipoActividad** = (Centralizada | Distribuida)^w Seguridad: exitosa.

Esquema de Rol: `CoordinadorAlgoritmos` **(CA)**
Descripción: Éste rol es responsable de coordinar las tareas de clustering y de administrar el ciclo de vida de los agentes que procesan estas tareas.
Protocolos y Actividades: RegistrarDF. RecibirActividad. RealizarActividad. RecibirResultados. EnviarResultados.
Permisos: Leer Actividad, Leer Fuente de Datos.
Responsabilidades: Viveza: **CA** = (RegistrarDF. RecibirActividad. RealizarActividad. EnviarResultados) **RealizarActividad** = (Centralizada | Distribuida)^w **Centralizada** = (CrearAgentes. SolicitarFuenteDatos. ObtenerFuenteDatos. EnviarFuenteDatos. RecibirResultados. EliminarAgentes) **Distribuida** = (CrearAgentes. SolicitarFuenteDatos. ObtenerFuenteDatos. ParticionarFuenteDatos. EnviarParticiones. RecibirResultadosLocales. GenerarResultadoGlobal. EliminarAgentes.) Seguridad: Una conexión exitosa debe establecerse con la Fuente de Datos.

Esquema de Rol: `RolClustering` **(RC)**
Descripción: Éste rol es el encargado de procesar las tareas de clustering que son solicitadas al MAS.
Protocolos y Actividades: RegistrarDF. RecibirFuenteDatos. ProcesarFuenteDatos. ObtenerResultados. EnviarResultados. DesRegistrarDF.
Permisos: Leer Resultados.
Responsabilidades: Viveza: **RC** = (RegistrarDF. RecibirFuenteDatos. ProcesarFuenteDatos. ObtenerResultados. ObtenerMediciones. EnviarResultados. DesRegistrarDF.) **ProcesarFuenteDatos** = (KMedias | Jerárquico)^w **KMedias** = (ProcesarParticion1 $\|$ ... $\|$ $ProcesarParticionN$) **Jerárquico** = (ProcesarParticion1 $\|$ ... $\|$ $ProcesarParticionN$) Seguridad: exitosa.

Tabla 5.2: **Modelos de roles de Gaia. Adaptado de Moraitis, Petraki, and Spanoudakis, 2003**.

Esquema de Rol: `AdministradorDatos` **(AD)**
Descripción: Este rol es responsable de accesar a los datos que se encuentran almacenados en diferentes fuentes de datos.
Protocolos y Actividades: RecibirSolicitud, ExtraerDatos, EnviarDatos.
Permisos: Leer datos.
Responsabilidades Viveza: **AD** = (RecibirSolicitud. ExtraerDatos. EnviarDatos.)^w. Seguridad: Una conexión exitosa debe establecerse con la Fuente de Datos.

Esquema de Rol: `RolValidación` **(RV)**
Descripción: Este rol evalua la calidad de alguna tarea procesada, en términos de alguna medida de precisión.
Protocolos y Actividades: ObtenerResultados, ObtenerMedida, EvaluarResultados, EnviarEvaluacion.
Permisos: Obtener medidas, Leer Resultados.
Responsabilidades: Viveza: **RV** = (ObtenerResultados. ObtenerMedida. EvaluarResultados. EnviarEvaluacion.)^w. Seguridad: exitosa.

Esquema de Rol: `RolRendimiento` **(RR)**
Descripción: Éste rol es responsable de establecer una estrategia para la ejecución de una tarea de clustering. Para establecer esta estrategia evalua alguna mediciones previas almacenadas por el rol de bitácora y determina la mejor estrategia para efectuar esa tarea. Esta estratégia es expresada como un estatus.
Protocolos y Actividades: RecibirSolicitud. AccesarBitacora. ObtenerMediciones. EvaluaMediciones. EstablecerEstrategia. EnviarEstrategia.
Permisos: Leer bitácora.
Responsabilidades: Viveza: **RR** = (RecibirSolicitud. AccesarBitacora. ObtenerMediciones. EvaluaMediciones. EstablecerEstrategia. EnviarEstrategia.)^w. Seguridad: Una conexión exitosa debe establecerse con la Bitácora.

Tabla 5.3: **Modelos de roles de Gaia. Adaptado de Moraitis, Petraki, and Spanoudakis, 2003**.

Para el caso del MAS propuesto en la **sección 1.7**, las actividades RegistrarDF y DesRegistrarDF son representas en el modelo de roles, aunque estos servicios son proporcionados directamente por JADE, y no existe una representación en Gaia.

5.2.2. Modelo de Interacción

El modelo de interacción es usado para representar dependencias y relaciones entre los roles en el MAS, de acuerdo a la definición de protocolos. Para los roles previamente definidos varios protocolos han sido definidos **[Sánchez-Pi, Carbó, and Molina, 2010]**. Estos son muestran en la **tabla 5.4** y **5.5**.

Protocolo	EnviarSolicitud	SolicitarEstrategia
Iniciador(es)	InterfazUsuario	Coordinador
Receptor(es)	Coordinador	RolRendimiento
Propósitos/Parámetros	Envía las solicitudes de los usuarios finales.	Envía una solicitud de estrategia para el procesamiento de una tarea de clustering.

Protocolo	EnviarActividad	SolicitarFuenteDatos
Iniciador(es)	Coordinador	AdministradorCoordinador
Receptor(es)	CoordinadorAlgoritmos	AdministradorDatos
Propósitos/Parámetros	Envia el tipo de actividad que debe ser llevada a cabo para procesar una tarea de clustering.	Envía una solicitud para obtener la fuente de datos que se usará para procesar una tarea de clustering.

Protocolo	EnviarFuenteDatos	EnviarResultados
Iniciador(es)	CoordinadorAlgoritmos	RolClustering
Receptor(es)	RolClustering	CoodinadorAlgoritmos
Propósitos/Parámetros	Envia la fuente de datos que será procesada por alguna tarea de clustering.	Envía los resultados obtenidos por la tarea de clustering procesada.

Protocolo	EnviarResultados	EnviarEvaluacion
Iniciador(es)	CoordinadorAlgoritmos	RolValidación
Receptor(es)	RolValidación	CoodinadorAlgoritmos
Propósitos/Parámetros	Envia los resultados de la tarea de clustering procesada para obtener una medida de evaluación de clustering.	Envía los resultados de evaluación de la tarea de clustering procesada.

Tabla 5.4: **Modelos de interacción de Gaia. Representación propia de acuerdo al modelo de Moraitis, Petraki, and Spanoudakis, 2003.**

Protocolo	EnviarResultados	EnviarResultados
Iniciador(es)	CoordinadorAlgoritmos	Coordinador
Receptor(es)	Coordinador	InterfazUsuario
Propósitos/Parámetros	Envia los resultados de la tarea de clustering procesada junto con las medidas de evaluación obtenidas y medidas de rendimiento del procesamiento.	Envía los resultados de la tarea de clustering procesada junto con las medidas de evaluación obtenidas y medidas de rendimiento del procesamiento. Estos resultados serán mostrados al usuario.

Protocolo	EnviarResultados	SolicitarDB
Iniciador(es)	InterfazUsuario	Coordinador
Receptor(es)	Bitácora	AdministradorDatos
Propósitos/Parámetros	Envia los resultados finales para extraer la información de solicitud y rendimiento, las cuales son almacenadas en una bitácora.	Envía una salicitud para obtener la lista de bases de datos y relaciones existentes para el sistema.

Tabla 5.5: **Modelos de interacción de Gaia. Representación propia de acuerdo al modelo de Moraitis, Petraki, and Spanoudakis, 2003.**

5.2.3. Fase de Diseño

Gaia transforma los modelo del análisis en un nivel de abstracción lo suficientemente bajo en el que se diseñan técnicas que pueden ser aplicadas con el fin de implementar los agentes. Para esta fase un modelo de agentes es obtenido junto con los modelos de servicios y conocidos.

5.2.3.1. Modelo de Agentes

La definición de un modelo de agentes consiste en identificar qué roles específicos juegan los agentes y cuantas instancias de agente tienen que ser instanciadas en el sistema actual. Para el sistema multi-agentes propuesto se han identificado siete tipos de agentes: el tipo de agente Usuario el cual cumple con el rol de InterfazUsuario, el tipo de agente Coordinador el cual cumple con el rol de Coordinador, el tipo de agente CoordinadorAlgoritmos el cual cumple con el rol de CoordinadorAlgoritmos, el tipo de agente Clustering el cual cumple con el rol de RolClustering, el tipo de agente Datos el cual cumple con los roles de AdministradorDatos y Bitácora, el tipo de agente Validación el cual cumple con el rol de RolValidación y el tipo de agente Rendimiento el cual cumple con el rol de **RolRendimiento**. El modelo de agentes es mostrado gráficamente en la **figura 5.1**.

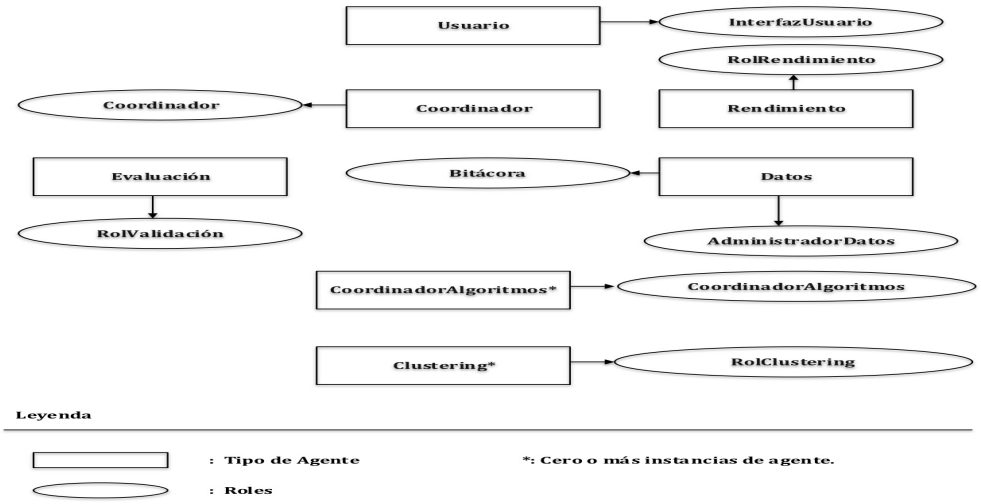


Figura 5.1: **Modelo de Agentes de Gaia. Basado en Moraitis, Petraki, and Spanoudakis, 2003**.

5.2.4. Modelo de Servicios

El modelo de servicios en la metodología Gaia representa todos los protocolos, actividades, responsabilidades y vivezas, asociadas a los roles que los agentes juegan en el sistema. El modelo de servicios utilizado en el sistema multi-agente es mostrado en la **tabla 5.6**.

Servicio	Administrador de Tareas de Clustering	Hiatorial de Tareas de Clustering
Entradas	Información de Clustering: clusters, métricas, criterio de enlace.	Los resultados de la tarea de clustering obtenidos.
Salidas	Modelo de Clustering, información de proceso.	La información de los resultados de la tarea de clustering.
Pre-Condiciones	Un agente debe tomar estas especificaciones e interactuar con el sistema multi-agente.	Los resultados de la tarea de clustering son almacenadas en una bitácora la cual es accesada por el sistema con el fin de obtener una estrategia para ejecutar una tarea de clustering.
Post-Condiciones	Un usuario debe proporcionar la información previa para la tarea de clustering.	Una tarea de clustering fue solicitada.

Tabla 5.6: **Modelos de servicios de Gaia. Basado en Moraitis, Petraki, and Spanoudakis, 2003**.

5.2.5. Modelo de Conocidos

Este modelo simplemente define las ligas de comunicación que existen entre los tipos de agentes **[Wooldridge, Jennings, and Kinny, 2000]**. No se define que mensajes son enviados o cuando son enviados, simplemente se indica los caminos de comunicación que existen. Para mostrar este modelo se emplea la propuesta empleada por **[Moraitis, Petraki, and Spanoudakis, 2003]** la cual varia ligeramente a la propuesta original de **[Wooldridge, Jennings, and Kinny, 2000]**. En esta propuesta se toma en cuenta la idea de que un agente puede interactuar con otros agentes (es decir, respondiendo una solicitud) sin tener la necesidad de tener conocimiento (información) sobre ellos. El modelo de conocidos para el MAS propuesto se muestra en la **tabla 5.7**.

	Usuario	Coordinador	Coordinador Algoritmos	Datos	Rendimiento	Evaluacion	Clustering
Usuario		I					
Coordinador	I		I	I	I		
Coordinador Algoritmos		I		I		I	I
Datos		I					
Rendimiento		I					
Evaluacion			I				
Clustering			I, A				

Leyenda:
I: Interacciones (leer "I" ocurrencias en fila, es decir el agente Coordinador interactua con los tipos CoordinadorAlgoritmos, Rendimiento, Usuario.
A: este tipo, es conocido en su estrucutura de datos (leer "A" ocurrencias en fila, es decir el tipo de agente CoordinadorAlgoritmos sabe de los tipos de agente Clustering.

Tabla 5.7: **Modelos de Conocidos de Gaia. Basado en Moraitis, Petraki, and Spanoudakis, 2003**.

5.3. Desarrollando Agentes en JADE desde el modelo Gaia

Una vez terminado el análisis y diseño de los agentes usando Gaia, es necesario definir los *comportamientos* que estos tendrán dentro del sistema. Para definir estos comportamientos, se debe tomar en consideración los protocolos y actividades que fueron definidos en la parte de *vivezas* de cada rol. A partir de estos, será posible establecer comportamientos y actos comunicativos que los agentes deberán llevar a cabo una vez que sean implementados en JADE **[Moraitis, Petraki, and Spanoudakis, 2003]**.

5.3.1. Diseño Detallado

El definir comportamientos de agentes en un nivel de desarrollo debe ir acompañado de otros conceptos al momento de que estos sean implementados. Entre estos están: los Mensajes ACL, las estructuras de datos y los algoritmos y componentes de software que emplearán los agentes.

5.3.1.1. Estructuras de Datos, Algoritmos y Componentes de Software

De acuerdo a las tareas de clustering que el MAS propuesto debe llevar a cabo, se emplearán las siguientes estructuras de datos[1], algoritmos y componentes de software:

- **Estructura de Actividades**. Esta estructura mantiene información sobre las tareas de clustering que se llevan a cabo. En particular maneja información sobre el **nombre del algoritmo**, **número de clusters** (para el caso del algoritmos K-Medias), **el tipo de criterio de enlace** (para el caso del algoritmo de clustering jerárquico), **métrica**, etc. Esta estructura es empleada por el `rol de clustering`.

- **Objetos Serializables**. Estos son responsables de transferir las solicitudes de los usuarios al MAS. Estos en particular manejan información sobre los **sitios** del sistema (bases de datos y archivos que contienen) e información sobre las **actividades de clustering** (tipos de algoritmos con su información respectiva).

- **Algoritmo de K-Medias**. Este algoritmo es empleado por el `rol de clustering`, el cual lleva como parámetros el número de clusters y la métrica.

- **Algoritmo Jerárquico**. Este algoritmo es empleado por el `rol de clustering`, el cual lleva como parámetros el tipo de criterio de enlace y la métrica.

- **Componentes de Software.** Para que el agente de Datos pueda extraer información de fuentes de datos externas se emplea el SMBD PostgreSQL, el cual administra información que los algoritmos de clustering pueden procesar. Además para acceder al sistema de archivos se emplea la clase FTPClient[2] de Java.

[1] Estas estructuras representan métodos internos pre-definidos que los agentes implementan

[2] Esta clase viene dentro de la biblioteca org.apache.commons.net.ftp.FTPClient, la cual encapsula funcionalidades de un servidor FTP.

5.3.1.2. Mensajes ACL

Un mensaje ACL permite transmitir una serie de conocimiento que vendrá expresado en un **lenguaje de contenido**. Los términos del lenguaje de contenido que representan conocimiento pertenecerán a un vocabulario común a los distintos agentes que se llama **ontología**. De acuerdo a la especificación de estructura de un mensaje ACL, para el sistema multi-agente propuesto se han definido los siguientes mensajes, los cuales se muestran en las **tablas 5.8** y **5.9**

Mensaje ACL: EnviarSolicitud Emisor: Agente Usuario Receptor: Agente Coordinador Performativa FIPA: `REQUEST` Protocolo: EnviarSolicitud Lenguaje: SL Ontología: OntologiaAlgoritmo Contenido: Acción de Ontología: Algoritmos	Mensaje ACL: EnviarResultados Emisor: Agente Coordinador Receptor: Agente Usuario Performativa FIPA: `INFORM` Performativa FIPA: EnviarSolicitud Lenguaje: SL Ontología: OntologiaAlgoritmo Contenido: Concepto de Ontología: Algoritmos
Mensaje ACL: SolicitarEstrategia Emisor: Agente Coordinador Receptor: Agente Rendimiento Performativa FIPA: `REQUEST` Protocolo: SolicitarEstrategia Lenguaje: SL Ontología: OntologiaEstrategia Contenido: Acción de Ontología: SolicitarEstrategia	Mensaje ACL: EnviarEstrategia Emisor: Agente Rendimiento Receptor: Agente Coordinador Performativa FIPA: `INFORM` Performativa FIPA: SolicitarEstrategia Lenguaje: SL Ontología: OntologiaEstrategia Contenido: Concepto de Ontología: Estrategia
Mensaje ACL: AccesarBitacora Emisor: Agente Rendimiento Receptor: Agente Datos Performativa FIPA: `QUERY_IF` Protocolo: AccesarBitacora Lenguaje: SL Ontología: OntologiaMediciones Contenido: Acción de Ontología: SolicitarMediciones	Mensaje ACL: EnviarMediciones Emisor: Agente Datos Receptor: Agente Rendimiento Performativa FIPA: `INFORM` Performativa FIPA: AccesarBitacora Lenguaje: SL Ontología: OntologiaMediciones Contenido: Concepto de Ontología: Mediciones
Mensaje ACL: SolicitarDatos Emisor: Agente Coordinador Receptor: Agente Datos Performativa FIPA: `REQUEST` Protocolo: SolicitarDatos Lenguaje: SL Ontología: OntologiaDatos Contenido: Acción de Ontología: SolicitarDataSet	Mensaje ACL: EnviarFuenteDatos Emisor: Agente Datos Receptor: Agente Coordinador Performativa FIPA: `INFORM` Performativa FIPA: EnviarFuenteDatos Lenguaje: SL Ontología: OntologiaDatos Contenido: Concepto de Ontología: DataSet

Tabla 5.8: **Definición de Mensajes ACL** . Basado en **Moraitis, Petraki, and Spanoudakis, 2003**.

Mensaje ACL: EnviarActividad Emisor: Agente Coordinador Receptor: Agente CoodinadorAlgoritmos Performativa FIPA: `REQUEST` Protocolo: EnviarActividad Lenguaje: SL Ontología: OntologiaActividad Contenido: Acción de Ontología: EnviarActividad	Mensaje ACL: EnviarResultados Emisor: Agente CoordinadorAlgoritmos Receptor: Agente Coordinador Performativa FIPA: `INFORM` Performativa FIPA: EnviarActividad Lenguaje: SL Ontología: OntologiaActividad Contenido: Concepto de Ontología: Actividad
Mensaje ACL: SolicitarFuenteDatos Emisor: Agente CoordinadorAlgoritmos Receptor: Agente Datos Performativa FIPA: `REQUEST` Protocolo: SolicitarFuenteDatos Lenguaje: SL Ontología: OntologiaFuenteDatos Contenido: Acción de Ontología: SolicitarFuenteDatos	Mensaje ACL: EnviarDatos Emisor: Agente Datos Receptor: Agente CoordinadorAlgoritmos Performativa FIPA: `INFORM` Performativa FIPA: SolicitarFuenteDatos Lenguaje: SL Ontología: OntologiaFuenteDatos Contenido: Concepto de Ontología: FuenteDatos
Mensaje ACL: EnviarFuenteDatos Emisor: Agente CoordinadorAlgoritmos Receptor: Agente Clustering Performativa FIPA: `REQUEST` Protocolo: EnviarFuenteDatos Lenguaje: SL Ontología: OntologiaClustering Contenido: Acción de Ontología: SolicitarClsutering	Mensaje ACL: EnviarResultados Emisor: Agente Clustering Receptor: Agente CoordinadorAlgoritmos Performativa FIPA: `AGREE` Performativa FIPA: EnviarFuenteDatos Lenguaje: SL Ontología: OntologiaClustering Contenido: Concepto de Ontología: Clustering
Mensaje ACL: EnviarResultados Emisor: Agente CoordinadorAlgoritmos Receptor: Agente Validación Performativa FIPA: `QUERY_IF` Protocolo: EnviarResultados Lenguaje: SL Ontología: OntologiaValidacion Contenido: Acción de Ontología: SolicitarValidación	Mensaje ACL: ObtenerResultados Emisor: Agente Validación Receptor: Agente CoordinadorAlgoritmos Performativa FIPA: `INFORM` Performativa FIPA: EnviarResultados Lenguaje: SL Ontología: OntologiaValidación Contenido: Concepto de Ontología: Validación

Tabla 5.9: **Definición de Mensajes ACL** . Basado en **Moraitis, Petraki, and Spanoudakis, 2003**.

5.3.2. Implementación en JADE

Para cumplir con las características de los mensajes y los roles establecidos previamente para propósitos de implementación en JADE se han definido los siguientes tipos de comportamientos y protocolos de comunicación que se van a usar para el MAS propuesto:

- **SimpleBehaviour.** Este clase representa a comportamientos atómicos, los cuales realizan tareas simples.
- **SequentialBehaviour.** Subclase de *CompositeBehaviour* la cual ejecuta subcomportamientos de manera secuencial y que termina cuando ellos han terminado.
- **ParallelBehaviour.** Subclase de *CompositeBehaviour* la cual ejecuta subcomportamientos de manera concurrente y que termina cuando se cumple una determinada condición sobre la terminación de los subcomportamientos o todos terminan su ejecución.
- **AchieveREInitiaror.** Protocolo de las clases AchieveRE la cual solicita a otro agente la realización de una acción. Este protocolo envía una solicitud de tipo `REQUEST`.
- **AchieveREResponder.** Protocolo de las clases AchieveRE la cual responde a la solicitud de otro agente que solicita la la realización de una acción. Este protocolo puede enviar diferentes respuestas tales como: `INFORM` (informa un resultado), `REFUSE` (rechaza el mensaje), `NOT UNDERSTOOD` (no entiende el mensaje) ó `FAILURE` (fallo el mensaje).

Estos comportamientos serán empleados por cada agente, como se muestra en las **tablas 5.11, 5.12, 5.13, 5.14**.

Agente: `Usuario`		
Nombre	**Comportamiento** ó **Protocolo**	**Características**
UserCoordinatorInitiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `InterfazUsuario`, el cual tiene el propósito de recibir las solicitudes de los usuarios y enviarlas al MAS.

Tabla 5.10: **Comportamientos implementados en JADE** . Adaptado de **Moraitis, Petraki, and Spanoudakis, 2003**.

Agente: `Coordinador`		
CoordinatorUserResponder	AchieveREResponder	Este protocolo cubre las actividades del rol `Coordinador`, en especial la actividad de `EnviarResultados` el cual contesta a la solicitud del rol `InterfazUsuario`.
CoordinatorPerformanceInitiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `Coordinador`, en especial la actividad de `SolicitarEstrategia` la cual envía una solicitud de estrategia al rol `RolRendimiento`.
CentralizedInitiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `Coordinador`, en especial la actividad `EstablecerActividad` la cual envía una solicitud de tarea centralizada, al rol `CoordinadorAlgoritmos`.
DistributedInitiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `Coordinador`, en especial la actividad `EstablecerActividad` la cual envía una solicitud de tarea distribuida al rol `CoordinadorAlgoritmos`.
ClusteringDistributed	SequentialBehaviour	Este comportamiento cubre las actividades del rol `Coordinador`, en especial la actividad `EnviarActividad` la cual envía una solicitud de tarea distribuida al rol `CoordinadorAlgoritmos`.

Tabla 5.11: **Comportamientos implementados en JADE** . Adaptado de **Moraitis, Petraki, and Spanoudakis, 2003**.

Agente: `CoordinadorAlgoritmos`		
CentralizedResponder	AchieveREResponder	Este protocolo cubre las actividades del rol `CoordinadorAlgoritmos`, en especial la actividad de `EnviarResultados` la cual responde al rol `Coordinador` enviando los resultados de una tarea de clustering centralizado.
DistributedResponder	AchieveREResponder	Este protocolo cubre las actividades del rol `CoordinadorAlgoritmos`, en especial la actividad de `EnviarResultados` la cual responde al rol `Coordinador` enviando los resultados de una tarea de clustering distribuido.
AlgorithmCoordinatorDataInitiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `CoordinadorAlgoritmos`, en especial la actividad de `SolicitarFuenteDatos` la cual envía una solicitud de fuente de datos al rol `AdministradorDatos`.
AlgorithmCoordinatorValidation Initiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `CoordinadorAlgoritmos`, en especial la actividad de `SolicitarEvaluacion` la cual envía una solicitud de evaluación de resultados de una tarea de clustering al rol `RolValidación`.

Tabla 5.12: **Comportamientos implementados en JADE** . Adaptado de **Moraitis, Petraki, and Spanoudakis, 2003**.

Nombre	**Comportamiento** ó **Protocolo**	**Características**
AlgorithmCoordinatorValidation Initiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `CoordinadorAlgoritmos`, en especial la actividad de `SolicitarEvaluacion` la cual envía una solicitud de evaluación de resultados de una tarea de clustering al rol `RolValidación`.
AlgorithmCoordinatorClustering Initiator	ParallelBehaviour[3]	Este comportamiento cubre las actividades del rol `CoordinadorAlgoritmos`, en especial la actividad de `RealizarActividad` la cual envía una solicitud para llevar a cabo una tarea de clustering al rol `RolClsutering`. Esta solicitud viene acompañada de particiones de datos de acuerdo a la estrategia establecida por el rol `RolRendimiento`.
Agente: `Rendimiento`		
CoordinatorPerformanceResponder	AchieveREResponder	Este protocolo cubre las actividades del rol `RolRendimiento`, en especial la actividad de `EnviarEstrategia` la cual responde al rol `Coordinador` enviando una estrategia para llevar a cabo una tarea de clustering.
LogInitiator	AchieveREInitiator	Este protocolo cubre las actividades del rol `RolRendimiento`, en especial la actividad de `AccesarBitacora` la cual envia una solicitud de mediciones al rol de `Bitacora` para poder establecer una estrategia de tarea de clustering.

Tabla 5.13: **Comportamientos implementados en JADE** . Adaptado de **Moraitis, Petraki, and Spanoudakis, 2003**.

[3]Este comportamiento internamente hace la solicitud de protocolo AchieveREInitiator al `RolClustering`. El cual pertenece a varias instancias de Agentes de Clustering.

Agente: `Datos`		
Nombre	**Comportamiento** ó **Protocolo**	**Características**
AlgorithmCoordinatorDataResponder	AchieveREResponder	Este protocolo cubre las actividades del rol `AdministradorDatos`, en especial la actividad de `EnviarDatos` la cual responde a la solicitud del rol `CoordinadorAlgoritmos` enviando una fuente de datos.
LogResponder	AchieveREResponder	Este protocolo cubre las actividades del rol `Bitacora`, en especial la actividad de `EnviarMediciones` la cual envia una serie de mediciones al rol `RolRendimiento` para poder establecer una estrategia de tarea de clustering.
Agente: `Clustering`		
AlgorithmCoordinatoClustering	AchieveREResponder	Este protocolo cubre las actividades del `RolClustering`, en especial la actividad de `ProcesarFuenteDatos` la cual procesa los datos enviados por el rol `CoordinadorAlgoritmos`. Una vez procesados los datos este envía una respuesta al rol `CoordinadorAlgoritmos` [4].
KMeansProcess	SimpleBehaviour	Este comportamiento procesa los datos que recibe, empleando el algoritmo de K-Medias.
HierarchicalProcess	SimpleBehaviour	Este comportamiento procesa los datos que recibe, empleando el algoritmo de Clustering Jerárquico.
Agente: `Validacion`		
AlgorithmCoordinatorValidation Initiator	AchieveREInitiator	Este protocolo cubre las actividades del `RolValidacion`, en especial la actividad de `EvaluarResultos` la cual evalua los resultados de una tarea de clustering llevada a cabo. Este envia los resultados de la evaluacion al rol `CoordinadorAlgoritmos`.

Tabla 5.14: **Comportamientos implementados en JADE** . Adaptado de **Moraitis, Petraki, and Spanoudakis, 2003**.

[4]Una respuesta es enviada a cada instancia del Agente **Coordinador de Algoritmos**.

5.4. Interacción entre Agentes

Una vez establecidos los agentes que serán implementados en JADE, es necesario modelar como los protocolos y comportamientos definidos previamente van a interacturar en el sistema multi-agente cuando esta en ejecución. Como se mencionó en la **sección 4.7.8** del **capítulo 4** enviar un mensaje en JADE implica cumplir con las especificaciones que FIPA establece para la estructura de un mensaje, entre estas estan el establecer: un *emisor*, un receptor, un *lenguaje de contenido*, una *ontología* y un conjunto de *performativas* (actos comunicativos). Para poder modelar esta interacción se empleará la metodología **AUML** la cual es una extensión del estándar **UML** para especificar un *Protocolo de interacción de agentes* (AIP), el cual describe un patrón de comunicación como una secuencia permitida de mensajes entre agentes y las restricciones en el contenido de estos mensajes **[Odell, Parunak, and Bauer, 2000]**. Para el MAS propuesto este modelo es mostrado en las **figuras 5.2** y **5.3** por medio de diagramas[5] de colaboración y secuencia:

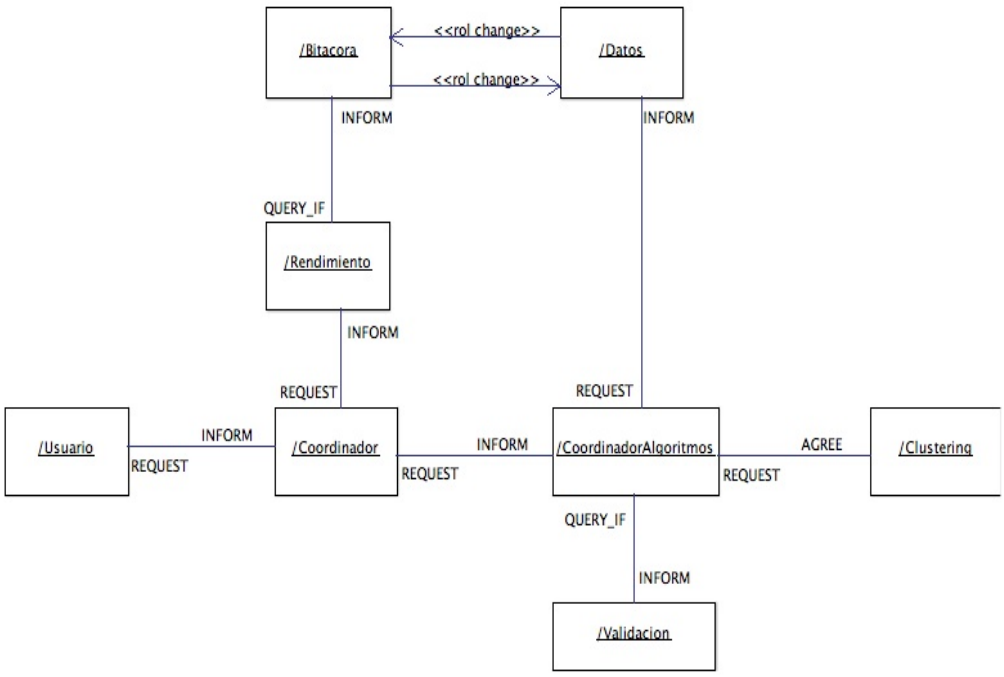


Figura 5.2: **Ejemplo de un diagrama de colaboración que muestra la interacción entre los roles de los agentes**. Basado en **Odell, Parunak, and Bauer, 2000**.

[5]Para cuestiones de modelado simple solo se tomará en consideración algunos diagramas de la metodología AUML y no se describirá todo el paquete de estos.

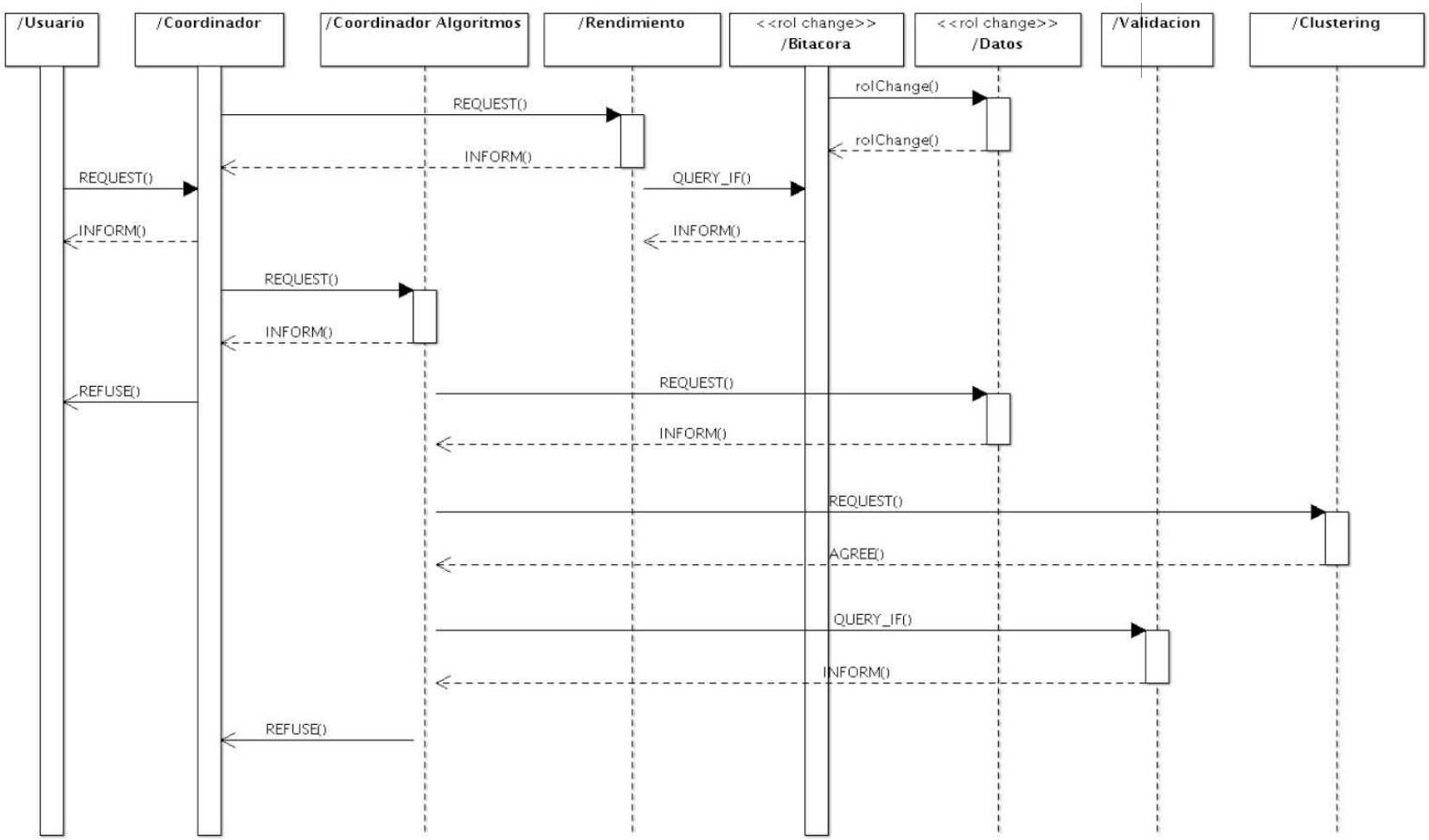


Figura 5.3: **Versión de la figura 5.2 en un Diagrama de Secuencia**. Basado en **Odell, Parunak, and Bauer, 2000**.

En los diagramas vistos previamente se han identificado los siguientes performativas las cuales se describen a continuación **Bellifemine, Caire, and Greenwood, 2007**:

- **REQUEST**. Solicita a un agente receptor que realice una acción.
- **QUERY_IF**. Pregunta a otro agente si una determinada proposición es cierta.
- **INFORM**. Informa a un agente receptor que una proposición es cierta.
- **REFUSE**. Rechazar realizar una acción (en el caso de un error en el mensaje).
- **AGREE**. Estar de acuerdo en realizar una acción.

Además se identifica a los agentes que juega varios roles como es el *AgenteDatos*, el cual puede ser un Administrador de Datos o una Bitacora. Esto es indicado por el estereotipo «`rolchange`».

5.4.1. Diseño de Ontologías

En JADE existen tres formas distintas de llevar a cabo la comunicación entre agentes **[Vaucher and Ncho, 2004]**:

1. A través de cadenas de caracteres para representar el contenido de los mensajes. Esta forma es conveniente cuando el contenido de los mensajes son datos atómicos y no abstractos.
2. A partir de objetos serializables de Java, los cuales transmiten el contenido de los mensajes directamente. Este método es conveniente para aplicaciones que son implementadas totalmente en Java.
3. A través de definir objetos (ontologías) que van a ser transferidos como extensión de clases predefinidas de JADE que pueden codificar/decodificar los mensajes en un formato FIPA estándar. Esto permite a los agentes de JADE interoperar con otros sistemas de agentes.

A fin de cumplir con los requerimientos de FIPA para el *marco de trabajo* propuesto se establecerán un conjunto de ontologías que permitan la comunicación entre los agentes del sistema. Este diseño de ontologías exige la identificación de los siguientes esquemas **[Tangient, 2010]**:

- **Conceptos**: representan las entidades que forma parte de la ontología. Estas entidades son representadas como atributos de objeto con sus respectivos métodos *get* y *set*.

- **Predicados**: son expresiones que relacionan a los conceptos para decir algo.
- **Acciones**: son acciones que pueden llevar a cabo los agentes.

De acuerdo a las características del sistema multi-agente a implementar, se han definido los siguientes esquemas de conceptos, acciones y predicados de ontologías, los cuales son representados, en las **figuras 5.4**, **5.5**, **5.6**.

I. **Conceptos:**

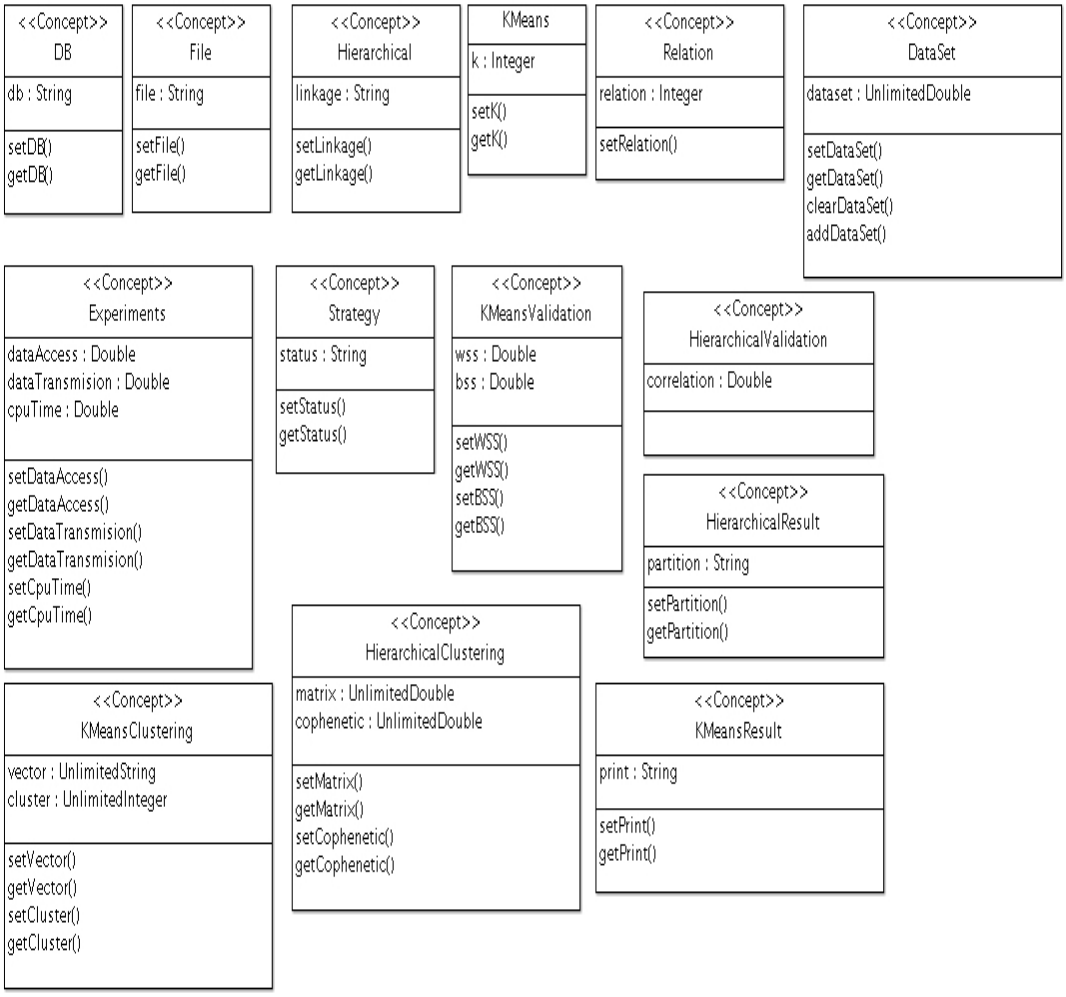


Figura 5.4: **Vista parcial del diagrama de ontologías para las clases de conceptos**. Basado en **Mylopoulos, Kolp, and Castro, 2001**.

II. **Acciones de Agentes:**

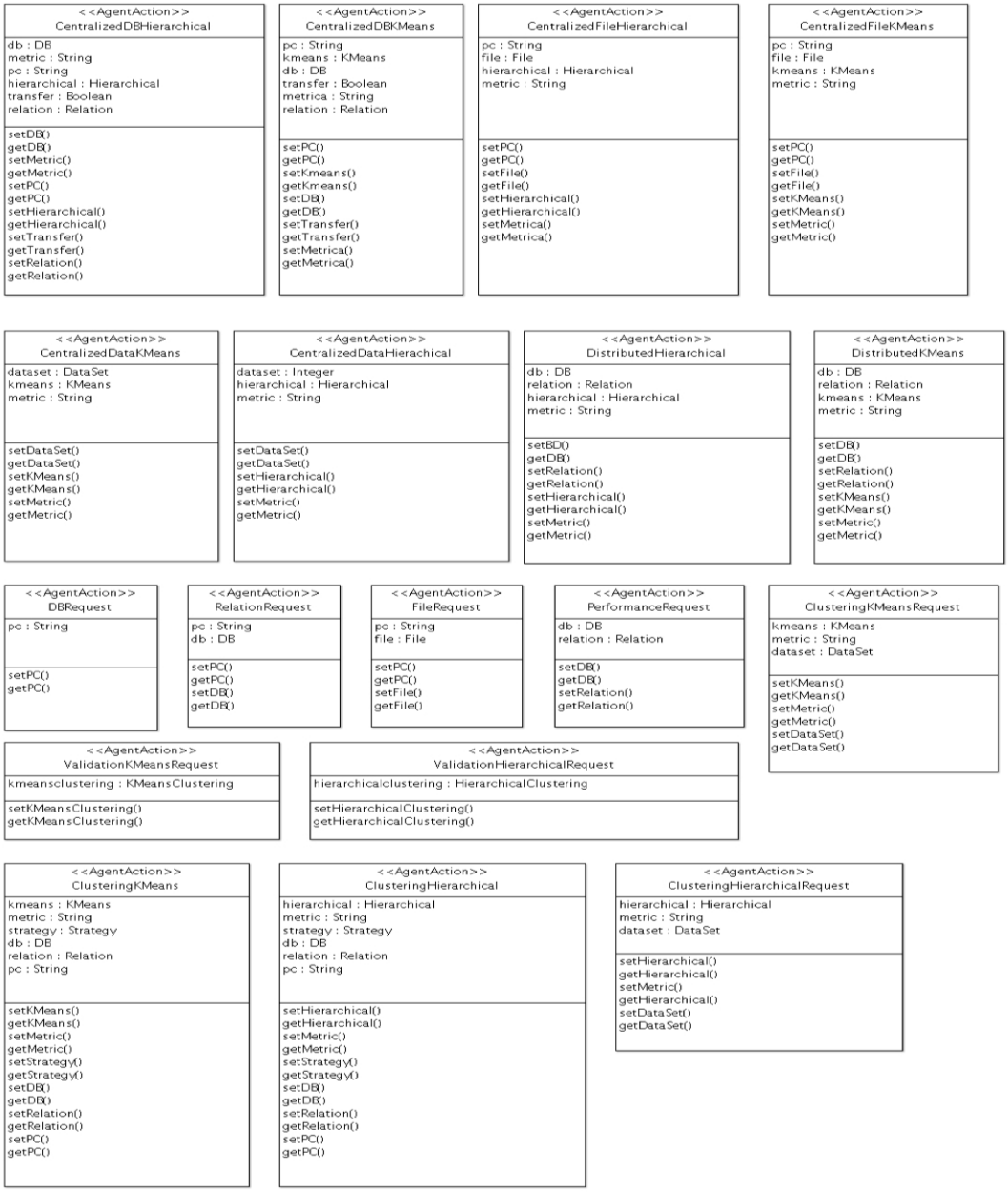


Figura 5.5: **Vista parcial del diagrama de ontologías para las clases de acciones de agente**. Basado en **Mylopoulos, Kolp, and Castro, 2001**.

III. **Predicados:**

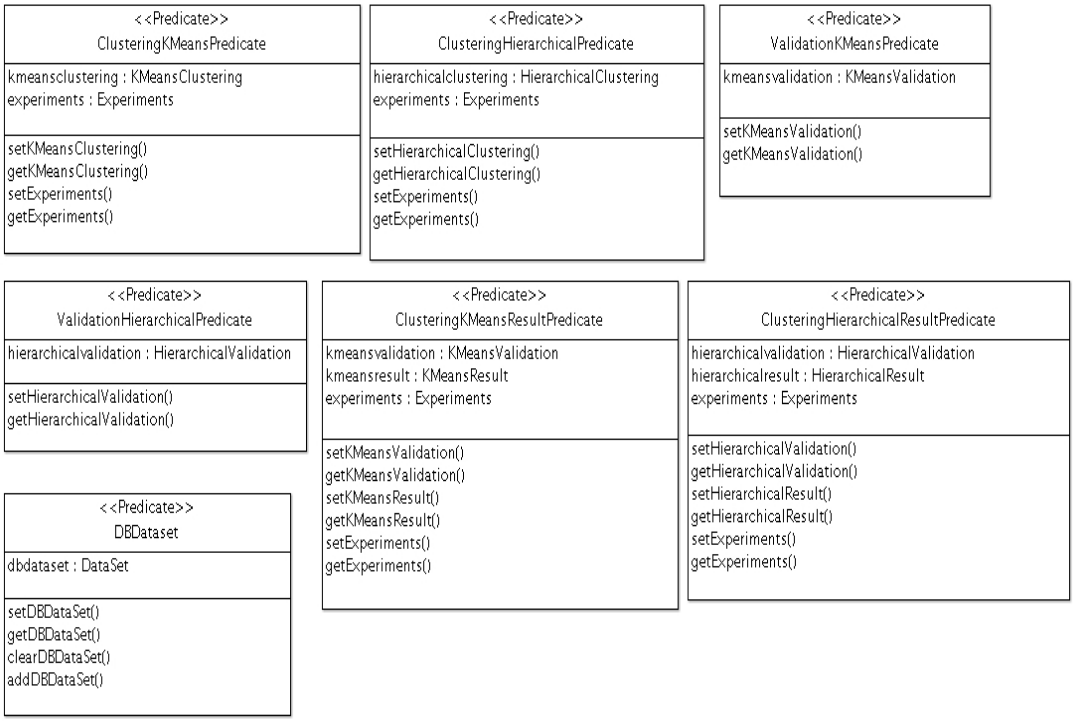


Figura 5.6: **Vista parcial del diagrama de ontologías para las clases de predicados**. Basado en **Mylopoulos, Kolp, and Castro, 2001**.

5.4.2. Codificación

Cada agente identificado en el proyecto fue representado como una clase de Java la cual hereda propiedades de la clase **jade.core.Agent**. Estas clases fueron declaradas en un paquete **Agents.** y los comportamientos y protocolos fueron implementados en un paquete separado **Behaviours.**, para que estos puedan ser reutilizados por los agentes que tengan comportamientos similares.

Para representar la estructura general de los agentes, se verá un ejemplo de codificación del agente Coordinador, el cual es representado en el **código 5.1**

Código 5.1: Agente Coordinador

```
package Agents;

import jade.content.lang.Codec;
import jade.content.lang.sl.SLCodec;
import jade.content.onto.Ontology;
import jade.content.onto.basic.Action;
import jade.core.AID;
import jade.core.Agent;
```

```
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.FailureException;
import jade.domain.FIPAAgentManagement.NotUnderstoodException;
import jade.domain.FIPAAgentManagement.RefuseException;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAException;
import jade.domain.FIPANames;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;

public class AgenteCoordinador extends Agent
{
 public String AGENT_TYPE = "Agente Coordinador";
 public String OWNER =  "Jonathan";

 private Codec codec = new SLCodec();
 private Ontology ontology = AlgorithmOntology.getInstance();

 public AgenteCoordinador()
 {
  super();
 }

 @Override
 protected void setup()
 {
  // Registrar Agente en el Directorio Facilitador (DF).
  ServiceDescription sd = new ServiceDescription();
  sd.setType(AGENT_TYPE);
  sd.setName(getLocalName());
  sd.setOwnwership(OWNER);
  DFAgenteDescription dfd = new DFAgenteDescription();
  dfd.addServices(sd);
  dfd.setName(getAID());
  try
  {
   DFService.register(this, dfd);
   System.out.println("Agente Coordinador fue registrado en el DF.");
  }
  catch(FIPAException e)
  {
   System.err.println("Registro del Agente " + getLocalName() +
   "no fue exitoso. Razón: " + e.getMessage());
   doDelete();
  }
  // Registro de lenguaje de agentes y ontologia
  getContentManager().registerLanguage(codec);
  getContentManager().registerOntology(ontology);

  // Filtrando mensaje que recibirá el protocolo AchieveREResponder del
      Agente Coordinador
  MessageTemplate protocol = MessageTemplate.MatchProtocol(FIPANames.
      InteractionProtocol.FIPA_REQUEST);
  MessageTemplate performative = MessageTemplate.MatchPerformative(ACLMessage
      .REQUEST);
  MessageTemplate contentLanguage = MessageTemplate.MatchLanguage(codec.
      getName());
  MessageTemplate ontoTemplate = MessageTemplate.MatchOntology(ontology.
      getName());

 MessageTemplate template = MessageTemplate.and(MessageTemplate.and(protocol,
      performative), MessageTemplate.and(contentLanguage,ontoTemplate));

 this.addBehaviour(new UsuarioCoordinadorResponder(this, template));
```

```
    super.setup();
   }
  }
```

Y la estructura del protocolo **UserCoordinatorResponder** se muestra a continuación en el fragmento de **código 5.2**.

Código 5.2: Protocolo UsuarioCoordinadorResponder

```
package Behaviours;

import Ontology.CentralizedDBKMeans;
import Ontology.DistributedKMeans;
import Ontology.DBRequest;
import Ontology.ClusteringKMeans;
import jade.content.onto.Ontology;
import jade.content.onto.basic.Action;
import jade.core.AID;
import jade.core.Agent;
import jade.core.behaviours.SequentialBehaviour;
import jade.proto.AchieveREInitiator;
import jade.proto.AchieveREResponder;

private class UserCoordinatorResponder extends AchieveREResponder
 {
  public UsuarioCoordinatorResponder(Agent a, MessageTemplate mt)
  {
   super(a,mt);
  }

  @Override
  protected ACLMessage prepareResponse(ACLMessage request)
  throws NotUnderstoodException, RefuseException
  {
   return null;
  }

  @Override
  protected ACLMessage prepareResultNotification(ACLMessage request,
      ACLMessage response)
  throws FailureException
  {
   System.out.println(" Agente Coordinador esta recibiendo datos ....");

   ACLMessage inform = request.createReply();
   inform.setPerformative(ACLMessage.INFORM);

   try
   {
    Action action = (Action) myAgent.getContentManager().extractContent(
        request);
    String actionType = action.getAction().toString();
    CentralizedDBKMeans cdbkm = new CentralizedDBKMeans(); // AgentAction (
        Ontologia)
    DistributedKMeans dkm = new DistributedKMeans();
    String agente = "";


    if(actionType.indexOf("CentralizedDBKMeans") > 0)
    {
     if(cdbkm.getPC().equals("PC_1"))
     {
      agente = "db@192.168.1.65:1099/JADE";
     }
```

```
        else
        {
         agente = "db2@192.168.1.65:1099/JADE";
        }

       // Agente Coordinador establec omunicación con el Agente Datos
        ACLMessage msg = new ACLMessage(ACLMessage.REQUEST);
        msg.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
        AID a = new AID(agente,AID.ISGUID);
        a.addAddresses();
        msg.addReceiver(a);
        msg.setLanguage(codec.getName());
        msg.setOntology(ontology.getName());

        DBRequest dbr = new DBRequest();  // Accion mandando
        dbr.setPC(cdbkm.getPC());

        action = new Action(getAID(), dbr);  // Enviando accion al Agente Datos

        getContentManager().fillContent(msg, action);

        addBehaviour(new CoordinadorDatosInitiator); // Comportamiento que envia
             un REQUEST al Agente Datos
        // El ComportamientoDatosResponder deberá enviar una lista con las bases
             de datos contenidos en una PC.
        .
        .
        .
       }
       .
       .
       SequentialBehaviour sb = new SequentialBehaviour(); // Comportamiento que
            envia

       if(actionType.indexOf("DistributedKMeans") > 0)
       {
         ClusteringKMeans ckm = new ClusteringKMeans();

         // Enviando un mensaje a los Agentes CoordinadorAlgoritmos en todas las
              PC
        ACLMessage message = new ACLMessage(ACLMessage.REQUEST);
        message.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST);
        for(int i=1; i<=2; i++)
        {
         message.addReceiver(new AID("coordinadoralgoritmo"+i+"@192
              .168.1.65:1099/JADE", AID.ISGUID))
        }

        ckm.setKMeans(dkm.getKMeans());
        ckm.setMetric(dkm.getMetric());
        ckm.setDB(dkm.getDB());
        ckm.setRelation(dkm.getRelation());
        ckm.setPC(dkm.getPC());
        ckm.setStrategy("High"); // Esta Estrategia viene del Agente Rendimiento

        action = new Action(getAID(), ckm);  // Enviando accion a los Agentes
             CoordinadorAlgoritmos

        getContentManager().fillContent(message, action);

        sb.addSubBehaviour(new CoordinadorCoordinadorAlgoritmoInitiator(this.
             myAgent, message));
        addBehaviour(sb);
       }
       .
       .
```

```
        .
      }
      catch(Exception e)
      {
       System.out.println(e.getMessage());
      }

      return inform;
     }
  }
```

5.5. Detalles adicionales de implementación

En las secciones anteriores se han tratado aspectos que tienen que ver con el diseño de sistemas multi-agentes sin tratar detalles de arquitectura de sistema e interfaz, los cuales serán tratados a partir de esta sección.

5.5.1. Arquitectura general del sistema

La propuesta de arquitectura actual incluye la implementación de agentes que llevan a cabo una función dentro del *marco de trabajo* usando la plataforma JADE y el diseño de una aplicación Web. La arquitectura actual provee los siguientes elementos:

- **Interface Web.** Ésta interface ofrece el medio por el cual los usuarios pueden ejecutar sus solicitudes para las tareas de clustering.
- **Repositorios de Datos.** Éstos consisten de carpetas ó sistemas de bases de datos que pueden almacenar datos en un sistema de DDM. En la parte de base de datos se incorpora un SMBD: PostgreSQL.
- **Repositorio de Clustering.** En este bloque se encuentran todos los algoritmos que son implementados para tareas de clustering. Y los algoritmos para validar los resultados de los algoritmos de clustering. En particular, dos tipos de tareas de clustering son llevadas a cabo: K-Medias y Clustering Jerárquico.
- **Motor del sistema.** Éste administra a los agentes diseñados, llevando a cabo las acciones de procesamiento de los datos y accediendo al DBMS. Para el sistema MAS se incoporan objetos de Java los cuales son responsables de definir acciones que son llevadas a cabo entre los agentes.

La arquitectura del **marco de trabajo** propuesto es representada en la **figura 5.7**.

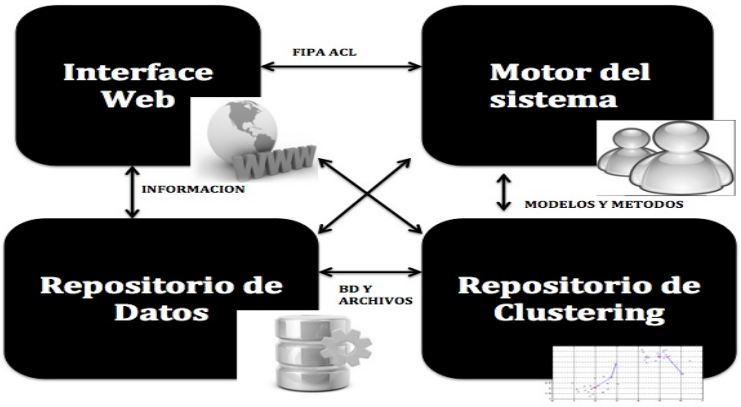


Figura 5.7: **Arquitectura del *marco de trabajo* para DDM basada en MAS**.

En particular, el motor del sistema, contiene una estructura de Sistemas Multi-Agentes la cual es representada en la **figura 5.8**.

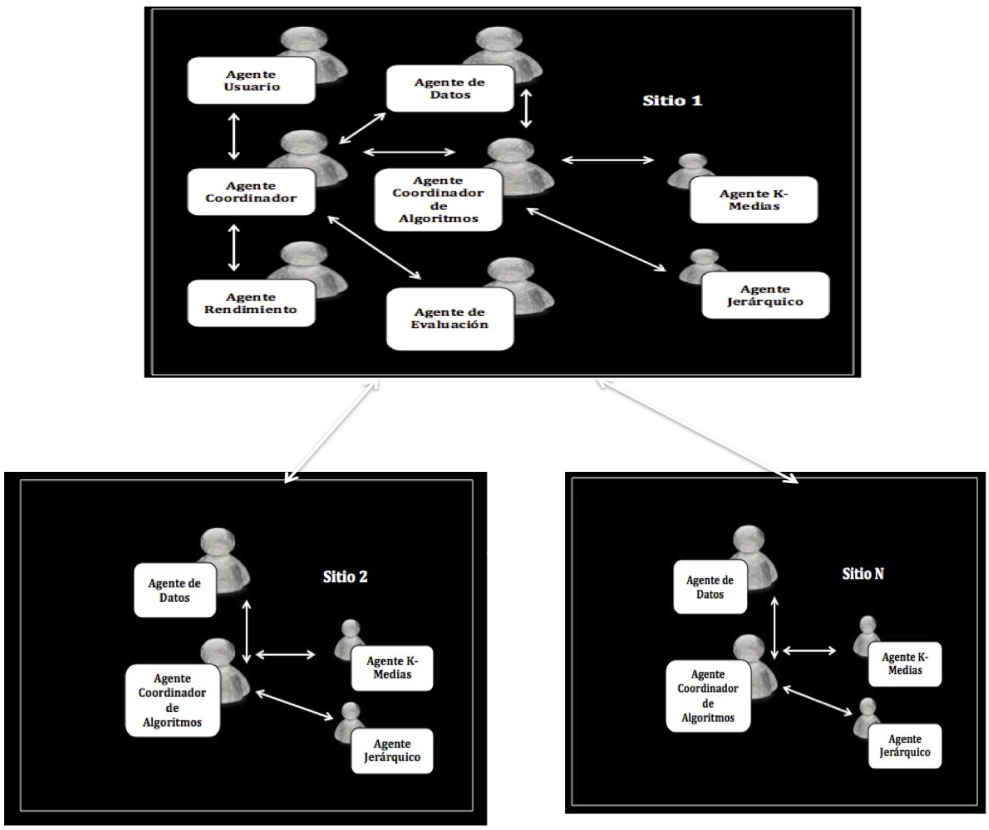


Figura 5.8: **Estructura del Sistema Multi-Agentes para el *marco de trabajo***.

5.5.2. Interfaz del sistema

El diseño de la interfaz esta motivado en la idea del proyecto DAME en el cual diversos sitios ponen a disposición una colección de tablas relacionales y archivos, los cuales pueden ser minados para propósito de investigación **[Board, 2010]**. La página principal del sistema es presentada en la **figura 5.9**.

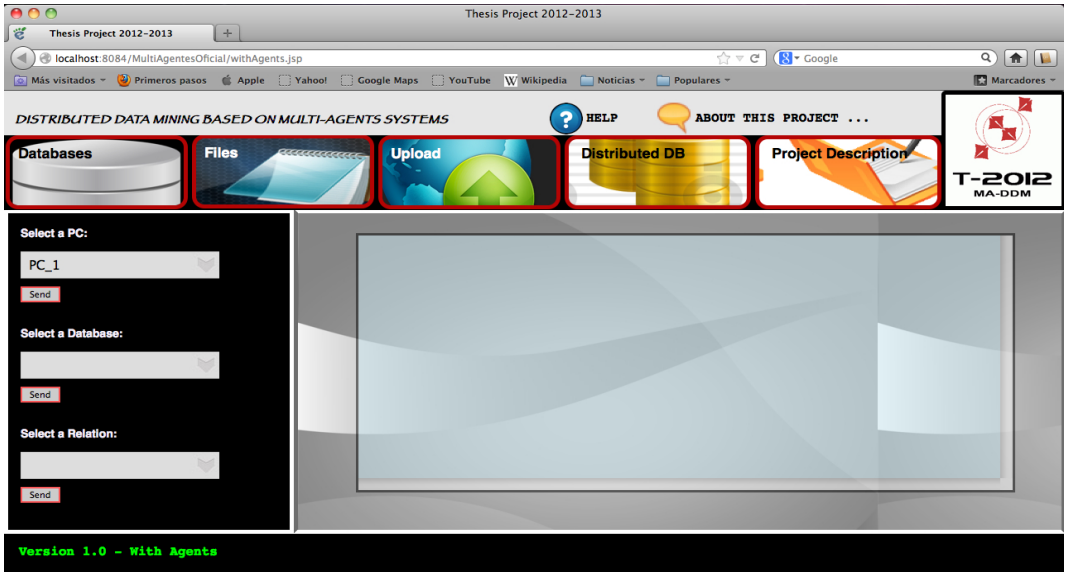


Figura 5.9: **Interfaz web para el sistema multi-agente**.

Esta interfaz consta básicamente de 4 módulos:

- **Módulo de Bases de Datos**. Este módulo permite a los usuarios seleccionar una tabla relacional de cualquier sitio del sistema para analizar sus datos. Esta petición implica solicitar, el nombre del sitio, de la base de datos y la tabla relacional a consultar. Este módulo es mostrado en la **figura 5.10**.

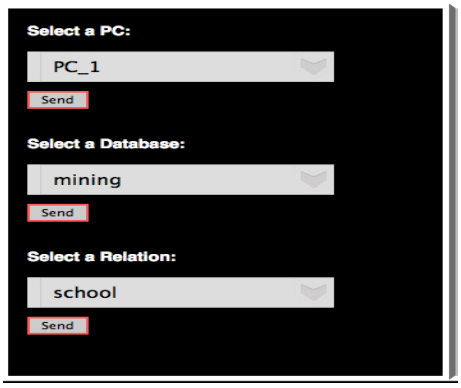


Figura 5.10: **Modulo de Bases de Datos del Sistema.**

Una vez llevada a cabo esta solicitud una ventana modal es desplegada solicitando el algoritmo que procesará la información de la tabla, la métrica, el número de clusters (en el caso del algoritmo K-Medias) y un criterio de enlace (en el caso del algoritmo jerarquico). Esta ventana es mostrada en la **figura 5.11**.

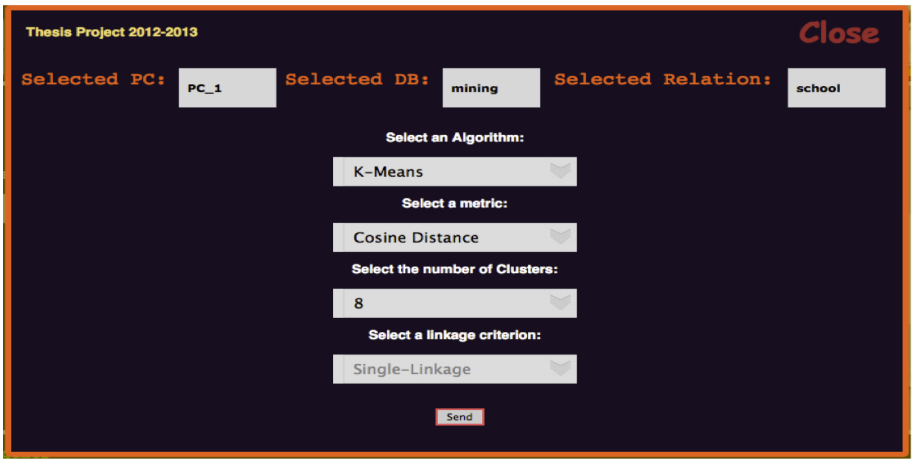


Figura 5.11: Ventana Modal del Modulo de Bases de Datos.

- **Módulo de Archivos**. Este módulo permite a los usuarios seleccionar un archivo plano de cualquier sitio del sistema para analizar sus datos. Esta petición implica solicitar, el nombre del sitio y archivo a consultar. Este módulo es mostrado en la **figura 5.12**.

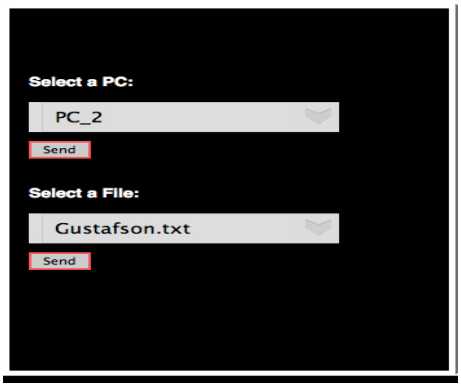


Figura 5.12: Módulo de Archivos del Sistema.

Una vez llevada a cabo esta solicitud una ventana modal es desplegada solicitando el algoritmo que procesará la información de la tabla, la métrica, el número de clusters (en el caso del algoritmo K-Medias) y un criterio de enlace (en el caso del algoritmo jerarquico). Esta ventana es mostrada en la **figura 5.13**.

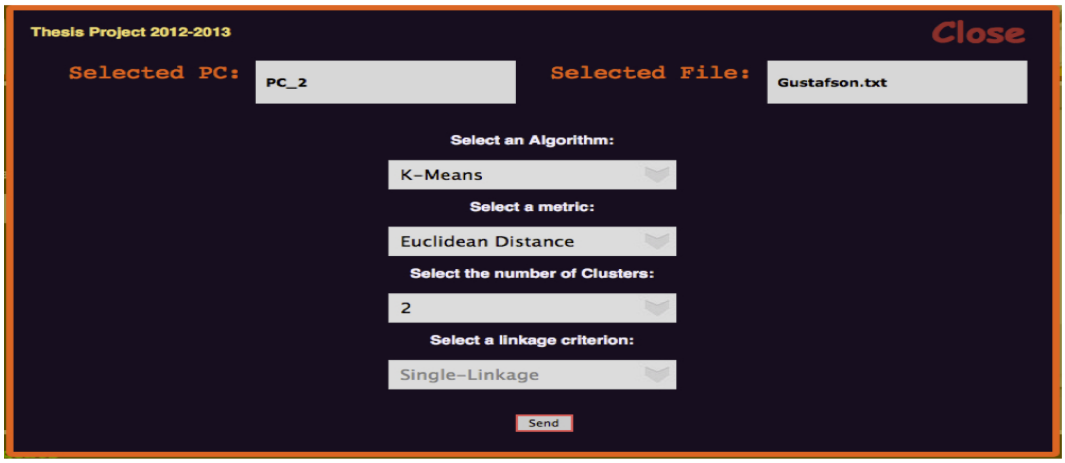


Figura 5.13: **Ventana Modal del Modulo de Archivos**.

- **Módulo para Subir Archivos**. Este módulo permite a los usuarios subir un archivo al sistema, para extraer y analizar su información. Este modulo es mostrado en la **figura 5.14**.

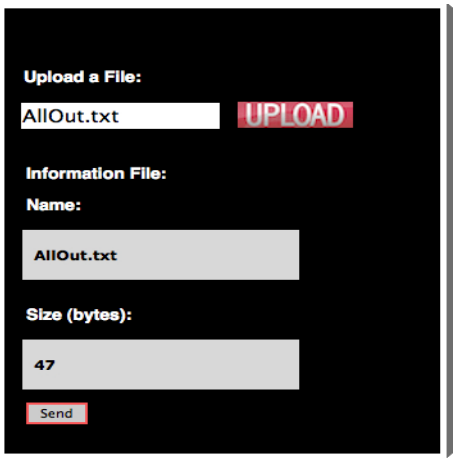


Figura 5.14: **Módulo para Subir Achivos del Sistema**.

Una vez subido el archivo una ventana modal es desplegada solicitando el algoritmo que procesará la información de la tabla, la métrica, el número de clusters (en el caso del algoritmo K-Medias) y un criterio de enlace (en el caso del algoritmo jerarquico). Esta ventana es mostrada en la **figura 5.15**.

Figura 5.15: **Ventana Modal del Modulo para Subir Archivos.**

- **Módulo para Base de Datos Distribuida**. Este módulo permite a los usuarios seleccionar una base de datos distribuida del sistema para su análisis. Este módulo es mostrado en la **figura 5.16**.

Figura 5.16: **Módulo de Bases de Datos Distribuida del sistema.**

Una vez llevada a cabo esta solicitud una ventana modal es desplegada solicitando el algoritmo que procesará la información de la tabla, la métrica, el n'umero de clusters (en el caso del algoritmo K-Medias) y un criterio de enlace (en el caso del algoritmo jerarquico). Esta ventana es mostrada en la **figura 5.17**.

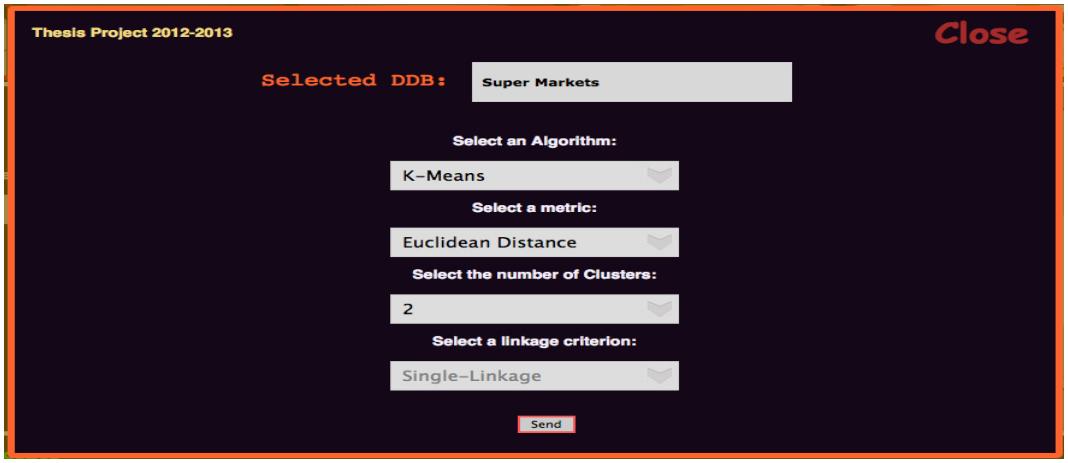
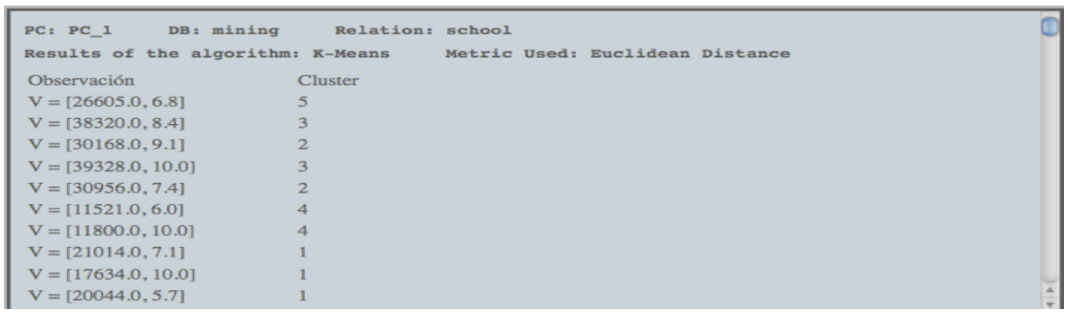


Figura 5.17: **Ventana Modal del Modulo de Bases de Datos Distribuida**.

Los resultados finales de las tareas de clustering procesadas son desplegadas en una ventana de resultados, la cual acompaña a cada módulo de la interfaz. Esta ventana es mostrada en la figura **figura 5.18**.


PC: PC_1 DB: mining Relation: school
Results of the algorithm: K-Means Metric Used: Euclidean Distance

Observación	Cluster
V = [26605.0, 6.8]	5
V = [38320.0, 8.4]	3
V = [30168.0, 9.1]	2
V = [39328.0, 10.0]	3
V = [30956.0, 7.4]	2
V = [11521.0, 6.0]	4
V = [11800.0, 10.0]	4
V = [21014.0, 7.1]	1
V = [17634.0, 10.0]	1
V = [20044.0, 5.7]	1

Figura 5.18: **Ventana de Resultados en cada módulo del sistema**.

Juntos con los resultados de clustering, se despliega cierta información sobre la tarea procesada, esta información es mostrada en la figura **figura 5.19**.

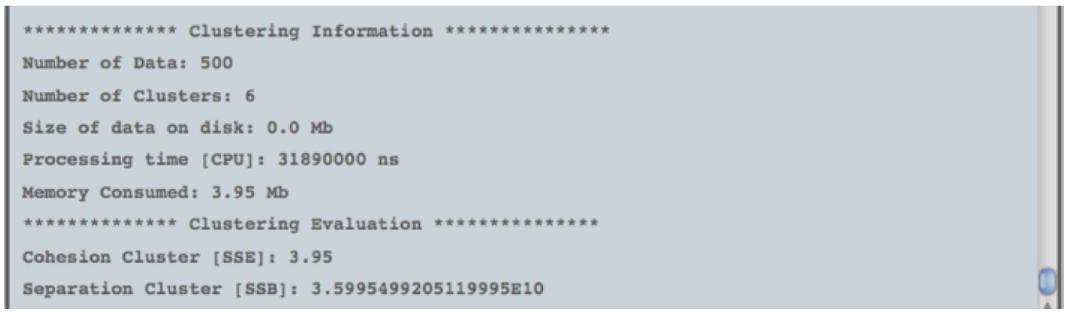

```
************** Clustering Information ***************
Number of Data: 500
Number of Clusters: 6
Size of data on disk: 0.0 Mb
Processing time [CPU]: 31890000 ns
Memory Consumed: 3.95 Mb
************** Clustering Evaluation ***************
Cohesion Cluster [SSE]: 3.95
Separation Cluster [SSB]: 3.5995499205119995E10
```

Figura 5.19: **Información desplegada con los resultados.**

Capítulo 6

Experimentación

En este capítulo se presentará todo el proceso de experimentación para evaluar el *marco de trabajo* que se propuso en el **capítulo 5**. Este proceso de experimentación consiste en dos etapas. En la primera etapa se validarán las tareas de clustering que fueron implementadas en el sistema tomando como variables de medición una serie de resultados tomados de técnicas de cohesión, separación y correlación las cuales verifican la correcta aplicación de los algoritmos de clustering. En la segunda parte se evaluará el rendimiento general del *marco de trabajo propuesto* en cuatro tipos de escenarios: [1] centralizado sin agentes, [2] centralizado con agentes, [3] distribuido sin agentes, [4] distribuido con agentes. Tomando como variables de medición costos de acceso a datos, transmisión de datos y procesamiento.

6.1. Validación de Clustering

El objetivo de la validación de clustering es medir de forma objetiva la bondad o calidad del resultado obtenido por un algoritmo de clustering. Esta calidad implica determinar el número de correcto de clusters en un conjunto de datos determinado o si una jerarquía de clusters es correcta. Para medir esta calidad es necesario emplear medidas numéricas que ayuden a determinar la correctez del clustering y comparar diferentes conjuntos de clusters, para determinar cual es mejor **[Goikoetxea, 2010]**. En el *marco de trabajo* propuesto en este documento, se han empleado dos tipos de tarea de clustering: K-Medias y Clustering Jerárquico; y se han empleado las siguientes técnicas de medición de la calidad de clustering:

- Cohesión. Esta medida establece que tan cercanos están los objetos en un cluster. Para medir esta cercanía se empleará una suma de cuadrados dentro de grupos (WSS) **[Tang, Steinbach, and Kumar, 2006]**:

$$\text{WSS}^{1} = \sum_{i} \sum_{x \in C_i} (x - m_i)^2 \tag{6.1}$$

Esta medida es buena si WSS es un valor pequeño. Se emplea para validar el algoritmo de K-Medias.

- Separación. Esta medida establece que tan separados estan los clusters. Para medir esta separación se empleará una suma de cuadrados entre grupos (BSS) **[Tang, Steinbach, and Kumar, 2006]**:

$$\text{BSS}^{2} = \sum_{i} |C_i|(m - m_i)^2 \tag{6.2}$$

En donde $|C_i|$ es el tamaño del cluster i.

Esta medida es buena si el BSS es un valor lo suficientemente grande. Se emplea para validar el algoritmo de K-Medias.

- Correlación Cofenética. Esta medida compara una jerarquía de clusters con la matriz de distancias de los datos. Para ello se calcula una matriz a partir de la jerarquía y se realiza una comparación entre las dos matrices utilizando el coeficiente de correlación momento-producto de Pearson **[Romesburg, 1984]**:

$$r_{x,y} = \frac{\sum xy - (1/n)(\sum x)(\sum y)}{\{[\sum x^2 - (1/n)(\sum x)^2][\sum y^2 - (1/n)(\sum y^2)]\}^{1/2}} \tag{6.3}$$

Si este coeficientes es bastante cercano a 100 %, se puede decir que la jerarquía de clustering es correcta.

Esta medida es empleada para validar el algoritmo de Clustering Jerárquico.

6.1.1. Métricas y Criterios de Enlace

Con el fin de definir el número de clusters que mejor se ajustan a un conjunto de datos, es necesario establecer un serie de *métricas* y *criterios de enlace* que permitan medir la similaridad entre un conjunto de puntos de datos. Para el *marco de trabajo* propuesto se emplearán las siguientes mediciones **[Han, Kamber, and Pei, 2012]** **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**:

[1] **Sum of squares within groups**
[2] **Sum of squares between groups**

Métricas:	
Nombre	**Formula**
Distancia Euclidiana[3]	$d(i,j) = ((x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2)^{\frac{1}{2}}$
Distancia de Manhattan[4]	$d(i,j) = \lvert x_{1f} - m_f \rvert + \lvert x_{2f} - m_f \rvert + \cdots + \lvert x_{nf} - m_f \rvert$
Distancia de Minkowski[5]	$d(i,j) = (\lvert x_{i1} - x_{j1} \rvert^p + \lvert x_{i2} - x_{j2} \rvert^p + \cdots + \lvert x_{in} - x_{jn} \rvert^p)^{\frac{1}{p}}$
Distancia del Coseno[6]	$\cos[d(i,j)] = \frac{A \cdot B}{\lvert A \rvert \lvert B \rvert} = \frac{(x_{i1}x_{j1})+(x_{i2}x_{j2})+\cdots+(x_{in}x_{jn})}{\sqrt{(x_{i1}^2)+(x_{i2}^2)+\cdots+(x_{in}^2)}\sqrt{(x_{j1}^2)+(x_{j2}^2)+\cdots+(x_{jn}^2)}}$
Criterios de Enlace:	
Nombre	**Formula**
Enlace Simple[7]	$d(A,B) = \text{mín}_{x \in A, y \in B}\, \mathbf{d(x,y)}$
Enlace Completo[8]	$d(A,B) = \text{máx}_{x \in A, y \in B}\, \mathbf{d(x,y)}$
Enlace Promedio[9]	$d(A,B) = \frac{1}{card(A)card(B)} \sum_{x \in A, y \in B} \mathbf{d(x,y)}$

Tabla 6.1: **Métricas y Criterios de Enlace** . Tomado de **Han, Kamber, and Pei, 2012** y **Cios, Pedrycz, Swiniarski, and Kurgan, 2007**.

6.1.2. Experimentación de validación de clustering

El proceso de experimentación en esta etapa consistirá en procesar 2 archivos planos. Uno utilizando el algoritmo de K-Medias y otro con el algoritmo de Clustering Jerárquico con diferentes métricas y criterios de enlace, con el fin de poder validar los resultados de cohesión, separación y correlación cofenética. Es necesario señalar que estos archivos son pequeños en cantidad de datos, pero que pueden aportar valiosa información y cumplen con el propósito de validación de datos. El objetivo final de esta validación será encontrar un cierto número de clusters ó cierta estructura jerárquica que mejor se ajuste a los datos. El primer archivo que se procesará es el `AllOut.txt` el cual solo contiene 9 datos. Los resultados obtenidos para este archivo se muestran en las **tablas 6.2** y **6.3**.

Archivo: `AllOut.txt`				
Algoritmo: `K-Means` **Métrica:** `Distancia Euclidana`				
	c = 2	c = 4	c = 6	c = 8
WSS	125.5	32	19.333	4.0
BSS	118.05	211	224.22	239.55

Tabla 6.2: **Mediciones de Validación de Clustering para el archivo `AllOut.txt`**.

[3]Detalles de esta distancia en la **ecuación 3.5**.
[4]Detalles de esta distancia en la **ecuación 3.6**.
[5]Detalles de esta distancia en la **ecuación 3.7**.
[6]Detalles de esta distancia en la **ecuación 3.9**.
[7]Detalles de este criterio de enlace en la **ecuación 3.18**
[8]Detalles de esta criterio de enlace en la **ecuación 3.19**
[9]Detalles de esta criterio de enlace en la **ecuación 3.20**

Archivo: `AllOut.txt`				
Algoritmo: `K-Means` **Métrica:** `Distancia Manhattan`				
	c = 2	c = 4	c = 6	c = 8
WSS	139.95	x	x	x
BSS	103.606	x	x	x
Algoritmo: `K-Means` **Métrica:** `Distancia Minkowski`				
	c = 2	c = 4	c = 6	c = 8
WSS	161.55	30.0	20.5	4.0
BSS	82.006	213.55	223.05	239.557
Algoritmo: `K-Means` **Métrica:** `Distancia de Coseno`				
	k = 2	k = 4	k = 6	k = 8
WSS	161.55	92.25	x	x
BSS	82.006	151.305	x	x

Tabla 6.3: **Mediciones de Validación de Clustering para el archivo `AllOut.txt`**.

Estos resultados fueron procesados utilizando el algoritmo de K-Medias con un valor de $k = 2$, $k = 4$, $k = 6$ Y $k = 8$.

El segundo archivo que se procesará, será el `4Distinct.txt`, el cual contiene 101 datos. Este será procesado con el algoritmo de Clustering Jerarquíco utilizando los criterios de enlace *único*, *completo* y *promedio*. Los resultados obtenidos para este archivo se muestran en la **tabla 6.4**.

Archivo: `4Distinct.txt`				
Algoritmo: `Clustering Jerárquico`				
		Enlace Unico	Enlace Completo	Enlace Promedio
Correlación Cofenética		0.89	0. 72	0.65

Tabla 6.4: **Mediciones de Validación de Clustering para el archivo `4Distinct.txt`**.

6.2. Rendimiento del Sistema

Con el fin de medir el rendimiento del *marco de trabajo* propuesto, en el **capítulo 5**, se han identificado un conjunto de experimentos de acuerdo a los siguientes escenarios:

a) **Datos Centralizado**: Un típico sistema de minería de datos, compuesto por un proceso centralizado de minería de datos, sin la intervención de agentes.

b) **Datos Centralizados con Multi-Agentes**: Un sistema de minería de datos centralizado basado en un sistema multi-agentes.

c) **Distribuido**: Un escenario de minería de datos distribuida sin la intervención de agentes.

d) **Minería de Datos Distribuido con Multi-Agentes**: Un sistema de DDM basado en un sistema multi-agentes.

Y las siguientes variables independientes: [1] métodos de clustering, [2] métricas, [3] número de clusters; y [4] fuentes de datos. Además de las siguientes variables dependientes: el [1] tiempo de acceso a datos, el [2] tiempo de transmisión de datos; y el [3] tiempo de procesamiento de alguna tarea de clustering.

Para cada uno de los escenarios, se procesarán un conjunto de 9 fuentes de datos, cuyos resultados serán presentados en las siguientes secciones.

6.2.1. Escenario de Datos Centralizado

Los resultados presentados de este escenario se obtuvieron de procesar las 9 fuentes de datos para el algoritmo de K-Medias y Clustering Jerarquíco, considerando para el algoritmo de K-Medias 10 clusters, y para el algoritmo de clustering jerarquíco un criterio de enlace simple. Los resultados para el algoritmo de K-Medias son presentados en la **tabla 6.5**.

Algoritmo: `K-Medias`				
Nombre de la Tabla	**Número de Tuplas**	**Tiempo de Acceso a Datos (ns)**	**Tiempo de Transmisión de Datos (ns)**	**Tiempo de procesamiento (ns)**
agency	35000	2427900	3.13E+08	1197083000
school	500	48200	6.08E+07	39682000
supermarket	150	29700	1.56E+07	1070000
weights	70	8000	7.44E+06	318000
substance	800	82700	5.22E+07	9810000
articles	500	28800	7.44E+06	2965000
survey	300	19000	5.22E+07	2560000
population	300	63000	3.52E+07	1257000
school_age	1200	82700	1.38E+08	31101000

Tabla 6.5: **Mediciones para el Algoritmo K-Medias en un Escenario Centralizado**.

Y a continuacion en la **tabla 6.6** se presentan los resultados para el algoritmo de Clustering Jerárquico.

Algoritmo: `Clustering Jerárquico`				
Nombre de la Tabla	**Número de Tuplas**	**Tiempo de Acceso a Datos (ns)**	**Tiempo de Transmisión de Datos (ns)**	**Tiempo de procesamiento (ns)**
agency	35000[10]	2417000	3.13E+08	1198003000
school	500	43400	6.08E+07	4602000
supermarket	150	14000	1.56E+07	1890000
weights	70	7800	7.44E+06	10830000
substance	800	91700	5.22E+07	2090000
articles	500	27900	7.44E+06	26695000
survey	300	18300	5.22E+07	3580000
population	300	44700	3.52E+07	1867000
school_age	1200	91700	1.38E+08	31711000

Tabla 6.6: **Mediciones para el Algoritmo de Clustering Jerárquico en un Escenario Centralizado**.

6.2.2. Escenario de Datos Centralizado con Sistemas Multi-Agentes

Los datos presentados en la **tabla 6.7** son el resultado de procesar 9 fuentes de datos empleando el algoritmo de K-Medias, se ha considerado tomar 10 clusters. En el caso de la tabla agency, por cuestiones de memoria de la máquina virtual de Java (JVM), solo se procesaron 1600 tuplas.

Algoritmo: `K-Medias`				
Nombre de la Tabla	**Número de Tuplas**	**Tiempo de Acceso a Datos (ns)**	**Tiempo de Transmisión de Datos (ns)**	**Tiempo de procesamiento (ns)**
agency	35000	2362800	3.13E+08	1197083000
school	500	41300	6.08E+07	39682000
supermarket	150	15500	1.56E+07	1070000
weights	70	9700	7.44E+06	318000
substance	800	62000	5.22E+07	9810000
articles	500	20800	7.44E+06	2565000
survey	300	21100	5.22E+07	2560000
population	300	26500	3.52E+07	847000
school_age	1200	62000	1.38E+08	30691000

Tabla 6.7: **Mediciones para el Algoritmo K-Medias en un Escenario Centralizado con agentes**.

[10]Por cuestiones de consumo de recursos sólo se procesaron 1600 datos de los 35000

Y a continuación se muestra la **tabla 6.8** donde se presentan los resultados de procesar las 9 fuentes de datos con el algoritmo de Clustering Jerárquico, tomando un criterio de enlace simple.

Algoritmo: `Clustering Jerárquico`				
Nombre de la Tabla	**Número de Tuplas**	**Tiempo de Acceso a Datos (ns)**	**Tiempo de Transmisión de Datos (ns)**	**Tiempo de procesamiento (ns)**
agency	35000[11]	2286200	3.13E+08	1197593000
school	500	63100	6.08E+07	4192000
supermarket	150	29300	1.56E+07	1070000
weights	70	8900	7.44E+06	1580000
substance	800	92300	5.22E+07	10320000
articles	500	27700	7.44E+06	26175000
survey	300	18800	5.22E+07	3070000
population	300	23700	3.52E+07	1357000
school_age	1200	92300	1.38E+08	31201000

Tabla 6.8: **Mediciones para el Algoritmo de Clustering Jerárquico en un Escenario Centralizado con agentes**.

6.2.3. Escenario de Datos Distribuido

Para los resultados del escenario distribuido se particionó la tabla *agency* en dos nodos A y B. Esta tabla fue procesada por los algorimos de K-Medias y Clustering Jerárquico a a partir de la invocación remota de métodos (Java RMI). En el caso del algoritmo de Clustering Jerárquico solo se procesaron 800 tuplas en cada nodo. Los resultados son presentados en la **tabla 6.9**.

Clustering Distribuido sin Agentes			
Número de Tuplas en Nodo A	**Número de Tuplas en Nodo B**	**Algoritmo**	**Tiempo de procesamiento (ns)**
18000	18000	K-Means	7.76E+08
800	800	Clustering Jerárquico	1.11E+10

Tabla 6.9: **Mediciones en un Escenario Distribuido sin agentes**.

[11] Por cuestiones de consumo de recursos sólo se procesaron 1600 datos de los 35000

6.2.4. Escenario de Datos Distribuido con Sistemas Multi-Agentes

En este escenario los resultados fueron obtenidos procesando la tabla distribuida *agency* la cual se particionó en dos nodos A y B. Estab tabla fue procesada utilizando los algoritmos de K-Medias y Clustering Jerárquico utilizando los sistemas multi-agentes. En la **tabla 6.10** se presentan los resultados del procesamiento.

Clustering Distribuido sin Agentes			
Número de Tuplas en Nodo A	**Número de Tuplas en Nodo B**	**Algoritmo**	**Tiempo de procesamiento (ns)**
18000	18000	K-Means	7.48E+08
800	800	Clustering Jerárquico	1.01E+10

Tabla 6.10: **Mediciones en un Escenario Distribuido sin agentes**.

Capítulo 7

Conclusiones Experimentales

En esta sección se presentará el análisis de resultados de los experimentos realizados en el **capítulo 6**. Estos experimentos fueron divididos en dos etapas. Una etapa en donde se validarón las tareas de clustering empleadas en el sistema; y una segunda etapa en donde se midió el rendimiento del sistema en diferentes tipos de escenario.

7.1. Análisis de Resultados de Validación de Clustering

El objetivo de la etapa de validación es determinar el número correcto de grupos para un conjunto de datos, a través de uso de indicadores que permitan obtener información sobre los datos. En este trabajo se establecieron los siguientes indicadores para encontrar el número de clusters que mejor se ajustan a determinados datos o representar la mejor jerarquía: métricas, criterios de enlace, cohesión, separación y correlación.

Estos indicadores fueron aplicados a 2 diferentes archivos, procesando el primero con el algoritmo de K-Medias y el segundo con el algoritmo de Clustering Jerárquico. Para el archivo `AllOut.txt` de acuerdo a los resultados obtenidos en las **tablas 6.2** y **6.3** se puede observar que la mejor métrica que se ajusto a los datos es una distancia Euclidiana con un valor de $k = 8$. Como se puede observar el valor de cohesión (WSS) es igual a 4.0 siendo este el valor el más pequeño que se calculo, estableciendo que en cada cluster la distancia entre puntos es mínima; y además se obtuvo un valor de separación (BSS) igual a 239.55 lo cual nos índica que cada cluster mantiene una separación considerable el uno del otro. Una de las ventajas de la distancia Euclidiana es que esta puede calcular las distancia más corta entre puntos por lo que rápidamente puede determinar una correlación en los datos.

Tambien se puede observar que de acuerdo a las características de los datos del archivo `AllOut.txt`, estos tienen un grado de disimilaridad alto por lo que $K = 8$ se ajusto mejor a los datos. A parte de la distancia Euclidiana hay otras medidas que es necesario considerar como es el caso de la distancia de Manhattan la cual se puede comparar con la distancia Euclidiana, pero a diferencia de esta, no calcula directamente la distancia entre dos puntos, por lo que no enfatiza las distancias más grandes a consideración de las más pequeñas por lo que deja menos en claro la solución de clustering. Se puede analizar también que muy de la mano con el resultado de la distancia Euclidana se encuentra la distancia de Minkowski la cual es una generalización de las dos distancia anteriores permitiendo medidas más exactas, esta distancia se puede adaptar perfectamente a datos en $n-$dimensiones **[Romesburg, 1984]**. Y finalmente se tiene el caso de la distancia de Cosenos que no satisface por completo los requerimientos de una métrica de distancia matemática pero puede ajustarse mejor a la "naturaleza de los datos", sin necesidad de determinar el número de clusters, aunque requiere de una mayor simetría de similaridad de los datos. la unión se establece en un nivel no local a través de encontrar la distancia máxima entre los datos **[StackExchange, 2013]** **[SIAM, 2003]**.

Los resultados de clustering para el archivo `AllOut.txt` es representado en la **figura 7.1**.

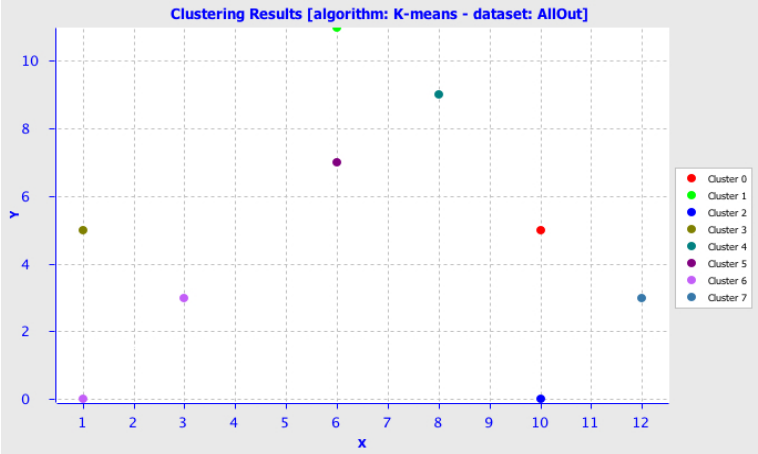


Figura 7.1: **Resultados de Clustering para el archivo `AllOut.txt` aplicando el algoritmo de K-Medias.**

Para el caso de los criterios de enlace para el archivo `4Distinct.txt` se puede observar que el mejor criterio para establecer una similaridad entre clusters fue el de un sólo enlace (`SINGLE LINKAGE`) debido a que la unión de los clusters se hace a nivel local, poniendo atención al área donde los clusters estan más cercanos. En el caso de enlace completo (`COMPLETE LINKAGE`) la unión de clusters se hace identificando a los miembros más disimilares. Este enlace a diferencia del sólo enlace no es local, debido a que la estructura entera del clustering puede in-

fluenciar en la decisión de las uniones. Para el caso de enlace promedio (AVERAGE LINKAGE) el criterio de unión de clusters radica en obtener una distancia promedio entre un par de observaciones, en un clusters **[Cios, Pedrycz, Swiniarski, and Kurgan, 2007]**. Este enlace promedio tiende a unir clusters con pequeñas varianzas. Podemos observar que la jerarquía obtenida en el sólo enlace es mejor observando que su correlación fue de 0.89, que es muy cercana al 100 %, esto indica que existen una correlación entre la matriz de distancias y la matriz cofenética, por lo que podemos decir que el clustering se ajusta bien a los datos.

Los resultados de clustering para el archivo `4Distinct.txt` es representado en la **figura 7.2** por medio de un *dendograma*.

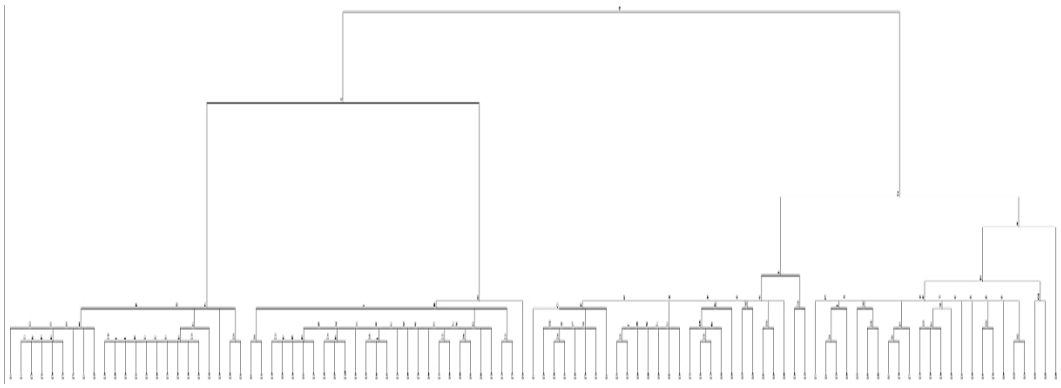


Figura 7.2: **Resultados de Clustering para el archivo `4Distinct.txt` aplicando el algoritmo de Clustering. Representación en forma de dendograma.**

7.1.1. Conclusiones de Validación de Clustering

En ésta etapa de validación se tomaron una serie de indicadores para determinar la mejor configuración de clustering para un grupo de datos determinado, estos indicadores fueron comparados indicando el porque cierta métrica se ajustaba mejor a los datos, y por lo tanto se podía determinar la mejor configuración de clustering para un archivo. Cabe señalar que en este trabajo se han manejado algoritmos de clustering los cuales manejan clusters con estructuras circulares, pero en un trabajo futuro se puede validar otros algoritmos como lo son los de densidad y basados en rejillas, los cuales manejan cluster con diferentes estructuras geométricas. Los resultados obtenidos al final contribuyeron en la validación de los resultados de clustering y en la predicción de particiones de clustering óptimas.

7.2. Análisis de Resultados de Rendimiento del Sistema

Los resultados de la etapa de experimentación de rendimiento en los diferentes tipos de escenarios planteados, son presentado a continuación. En las **figuras 7.3** y **7.4** se puede observar una gráfica de comparación del tiempo de procesamiento del sistema en un escenario centralizado con agentes y sin agentes aplicando los algoritmo de K-Medias y Clustering Jerarquico; y en las **figuras 7.5** y **7.6** se puede observar una comparación del tiempo de acceso a datos del sistema en un escenario centralizado con agentes y sin agentes aplicando loa algoritmos de K-Medias y Clustering Jerárquico.

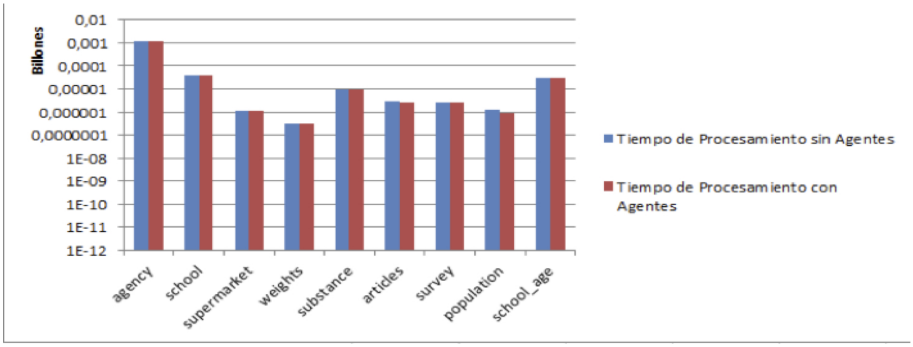


Figura 7.3: **Tiempo de Procesamiento en un escenario centralizado con agentes y sin agentes, aplicando el algoritmo de K-Medias.**

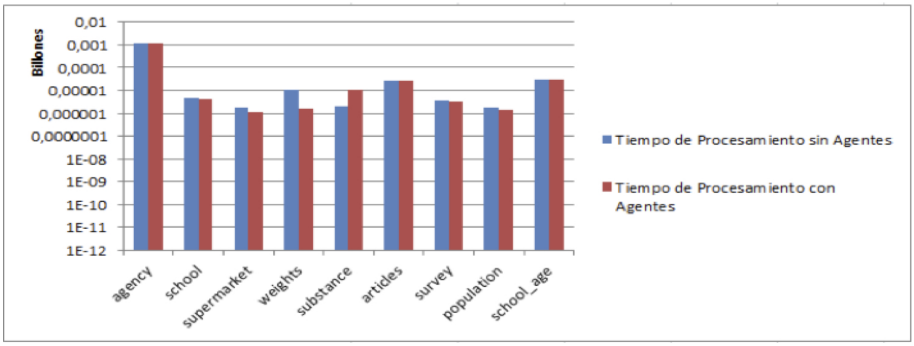


Figura 7.4: **Tiempo de Procesamiento en un escenario centralizado con agentes y sin agentes, aplicando el algoritmo de Clustering Jerárquico.**

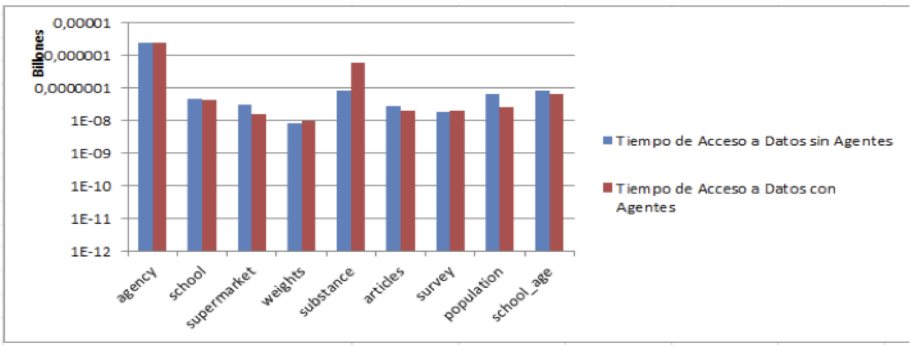


Figura 7.5: **Tiempo de Accesos a Datos en un escenario centralizado con agentes y sin agentes, aplicando el algoritmo de K-Medias.**

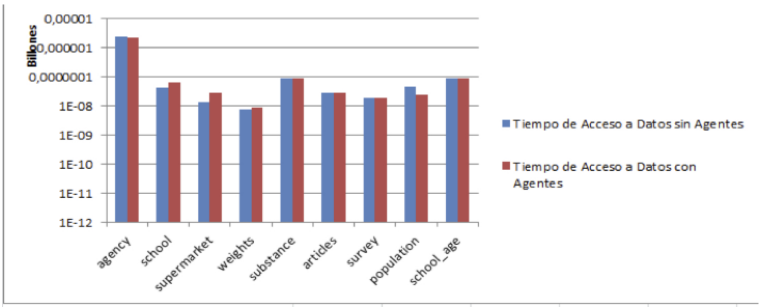


Figura 7.6: **Tiempo de Acceso a Datos en un escenario centralizado con agentes y sin agentes, aplicando el algoritmo de Clustering Jerárquico.**

En las **tablas 7.3** y **7.4** se puede identificar una ligera ventaja en el procesamiento de datos utilizando el sistema multi-agente. Procesar particiones de datos con MAS, y unir los resultados, permite de manera más rápida procesar los datos. Si la cantidad de datos es significativamente grande, los datos pueden ser compartidos entre n agentes, reduciendo el tiempo de procesamiento. Una desventaja de esto es que al particionarse los datos, se pierde precisión en la calidad de los clusters.

En las **tablas 7.5** y **7.6** se puede observar que el tiempo de acceso puede ser un poco variable en un escenario con agentes y sin agentes, esto es debido al SMBD el cual administra las fuentes de datos, pero en general podemos concluir que los agentes responden bien al accesar a los datos, obteniendo una rápida respuesta de la fuente de datos.

Finalmente en un escenario distribuido, se puede observar de acuerdo a las **tablas 6.9** y **6.10**, que el tiempo de procesamiento de los agentes fue ligeramente mejor que el tiempo de procesamiento de un método RMI **[Caballé and Xhafa, 2008]**, debido a que en el escenario distribuido con agentes los datos fueron particionados y procesados por varios agentes, mientras que el escenario sin agentes el método RMI, tuvo que transferir los datos, procesarlos todos en ambos nodos, y enviar una respuesta, lo cual le exigio un tiempo de respuesta mayor. Los resultados son mostrados en la **figura 7.7**.

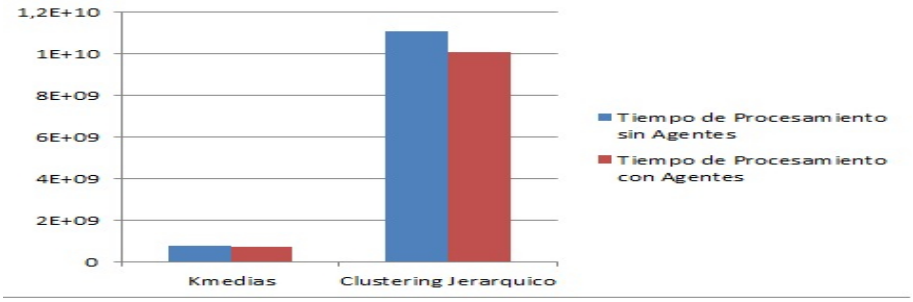


Figura 7.7: **Tiempo de Acceso a Datos en un escenario centralizado con agentes y sin agentes, aplicando el algoritmo de Clustering Jerárquico.**

7.2.1. Conclusiones de Rendimiento del Sistema

En la etapa de rendimiento del sistema de minería de datos distribuida se puede observar que los sistemas múlti-agentes respondieron de manera óptima en cada uno de los escenarios en donde fueron implementados, por lo que se puede concluir que cumplieron con su objetivo de mejorar el rendimiento del sistema de minería de datos distribuida. Éste mejoramiento quedo demostrado en el tiempo en que los agentes respondieron a una solicitud de tarea de clustering, en el tiempo en el que procesaron ésta solicitud y en el tiempo en que los datos fueron transmitidos en el sistema. En cada uno de los escenarios se estableció una estrategia de particionamiento de los datos con la cual se logro reducir el tiempo de respuesta y aligerar la carga de trabajo del sistema. Si la cantidad de memoria del sistema es limitada los datos podían ser particionados entre varios agentes de manera que cada agente procesara una partición reduciento gastos generales. De acuerdo a la estrategia de particionamiento seleccionada cada agente concurrentemente podía procesar su partición reduciendo el tiempo de respuesta total del sistema. Con respecto a los escenarios centralizados y distribuidos, el uso de sistemas múlti-agente presentó una ventaja significativa debido a que los agentes fueron diseñados para trabajar como un sistema distribuido. En el caso del escenario distribuido con múlti-agentes, el cada sitio procesara sus datos localmente y enviara sus resultados permitio que el tiempo de procesamiento de los datos fuera optimizado, con respecto al caso no distribuido en el cual se implementó RMI (Métodos de Invocación Remota Java) para invocar métodos de manera remota, ofreciendo la ventaja de exportar objetos de Java entre los sitios, pero que no es lo suficientemente rápido para procesar tareas distribuidas, comparado con una herramienta totalmente distribuida como JADE.

Capítulo 8

Trabajos Futuros y Conclusiones

La tecnología de agentes ha estado sujeta a una extensa discusión e investigación dentro de la comunidad científica por varios años, pero es en años recientes que ésta ha adquirido un valor significativo debido a su grado de explotación en diversidad de aplicaciones científicas y comerciales. En particular, debido a su naturaleza distribuida los sistemas multi-agentes estan siendo empleados por aplicaciones de minería de datos las cuales requieren analizar datos que se encuentran geográficamente distribuidos. De acuerdo con esto, este documento de tesis ha sido dirigido principalmente a los siguientes aspectos de investigación:

1. Aplicar la tecnología de agentes para construir un *marco de trabajo* genérico que soporte sistemas de minería de datos centralizada y distribuida.

2. Comparar el desempeño del marco de trabajo construido con respecto a otros sistemas de minería de datos centralizada y distribuida que no emplean agentes, analizando las siguientes características:

   - **Costos de tiempo de acceso a los datos**. Tiempo requerido para que el sistema accese a los datos en disco.

   - **Costo de transmisión de los datos**. Tiempo que tarda para que la información sea transmitida a otro sitio.

   - **Costo de procesamiento (CPU).** Periodo de tiempo utilizado en el procesamiento de una tarea de minería de datos.

La efectividad de este *marco de trabajo* quedo demostrado en un escenario de clustering de datos el cual obtuvo un mejor rendimiento desde una perpectiva de minería de datos. Ahora bien, es necesario señalar que el *marco de trabajo* propuesto tiene como meta mejorar el rendimiento de un sistema de minería de datos distribuido, contruyendo agentes que cumplen con roles de procesamiento, administración y evaluación, pero un trabajo futuro incluye las siguientes apor-

taciones:

- Optimizar las estrategias para el procesamiento de tareas de clustering distribuido. Estas estrategias involucran aspectos de organización de la información, administración de recursos y análisis de datos.
- El desarrollo de agentes que elaboren tareas de preprocesamiento de datos, estas tareas involucran aspectos de limpieza de datos, integración de datos, selección y transformación de datos.
- El desarrollo de agentes que lleven a cabo otras tareas de clustering, como por ejemplo, clustering basado en densidad y rejillas.
- El desarrollo de agentes que controlen actividades de concurrencia y distribución. Este desarrollo involucra la creación de agentes móviles.
- Construir una estructura de MAS que ponga atención a la privacidad de los datos y tolerancia a fallas.
- La integración de diversas comunidades de agentes que puedan intercambiar información.
- La contrucción de una infraestructura de sistema que maneje diversos casos de uso práctico. Como por ejemplo, los trabajos expuestos en **[Helbing and Balietti, 2011]**, **[Oates, Schmill, and Cohen, 1996]**, **[Zhang and Segall, 2010]**, **[Yang, Tao, Xu, and Zhang, 2009]**, **[Pérez-Ortega, Martínez, Iturbide, and Zavala-Díaz, 2013]**.

Finalmente es necesario comentar que el *marco de trabajo* propuesto fue implementado usando la herramienta de desarrollo JADE. Ésta experiencia reforzó la necesidad de emplear metodologías de ingeniería de software orientada a agentes (OASE) para modelar la estructura de los agentes y como interactúan éstos en el sistema.

Bibliografía

Inc Acroymics. Agentbuilder, 2004. URL http://www.agentbuilder.com/Documentation/Pro/.

P. Adriaans and D. Zantinge. *Data Mining*. Addison-Wesley, 1996.

R. Agrawal, T. Imielinski, and A. Swami. *Mining association rules between sets of items in large databases*. ACM, 1993.

K. A. Albashiri, F. Coenen, and P. Leng. *An investigation into the issues of Multi-Agent Data MIning*. PhD thesis, The University of Liverpool, 2010.

M. Aldenderfer and R. K. Blashfield. *Cluster Analysis*. Sage Publications, 1984.

J. Aronis, V. Kolluri, F. Provost, and B. Buchanan. *The WoRLD: knowledge discovery from multiple distributed databases*. Department of Computer Science, Pittsburgh, university of pisttburgh edition, 1996.

S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta. *Clustering distributed data streams in peer-to-peer environments*. Elsevier, 2006.

F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. *JADE: A White Paper*, volume 3, pages 6–19. 2003.

F. Bellifemine, G. Caire, and D. Greenwood. *Developing multi-agents systems with JADE*. John Wiley and Sons, Inc., 2007.

DAME Board. Data mining and exploration, 2010. URL http://dame.dsf.unina.it/.

P. Bonnet, J. Gehrke, and P. Seshadri. *Towards sensor database systems.*, pages 3–14. Springer-Verlag, 2nd. international conference on mobile data management edition, 2001.

R. H. Bordini and J. F. Hübner. *BDI Agent Programming in AgentSpeak Using Jason*, volume 3900, pages 143–164. 6th. International Workshop on Computational Logic in MAS., 2005.

J. M. Bradshow. *An Introduction to Software Agents. Software Agents.* MIT Press, 1997.

M. E. Bratman. *Intention, Plans, and Practical Reason.* Harvard University Press, Cambridge, MA, 1987.

P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodlogy. *Autonomoues Agents and Multi-Agent Systems*, 2004.

R. A. Brooks. *Intelligence Wothout Representation*, pages 139–159. Artificial Intelligence, 1991.

M. Bubak, G. D. van Albada, J. Dongarra, and P. M. A. Sloot. *Computational Science - ICCS 2008.* 8th International Conference. Springer, Kraków Poland, iii edition, 2008.

P. Bungkomkhun. *Grid-Based Supervised Clustering Algorithm Using Greedy and Gradient Descent Methods to Build Clusters.* PhD thesis, National Institute of Development Administration, 2012.

P. Bussete, R. Ronnquist, A. Hodgson, and A. Lucas. Jack intelligent agents - components for intelligent agents in java. 1999.

S. Bussmann and J. A. Muller. *A Negotiation Framework for Co-operating Agents*, pages 1–17. CKBS-SIG, Keele, UK, 1992.

S. Caballé and F. Xhafa. *Aplicaciones Distribuidad en Java Con Tecnología RMI.* Delta Publicaciones, 2008.

G. Caire. *JADE Tutorial. JADE Programming for Begginers.* TILAB, 2009.

Doina Caragea. *Learning classifiers from distributed, semantically heterogeneous, autonomous data sources.* PhD thesis, Iowa State University, 2004.

S. Chaimontree, K. Atkinson, and F. Coenen. *Multi-Agent Based Clustering: Towards Generic Multi-Agent Data Mining*, volume 6171, pages 115–127. Lecture Notes in Computer Science, 2010.

P. Chan, W. Fan, A. Prodromidis, and S. Stolfo. *Distributed data mining in credit fraud detection*, pages 67–74. IEEE Intelligent Systems, 1999.

J. Chattratichat, J. Darlington, Y. Guo, S. Hedvall, M. Koler, and J. Syed. *An architecture for distributed enterprise data mining*, pages 573–582. High Performance Caomputing Networking, Amsterdam, Netherlands, 1999.

K.J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. *Data Mining A Knowledge Discovery Approach.* Springer Science, 2007.

R. M. Cormack. *A review of classification.* Journal of the Royal Statistical Society A, 1971.

W. H. E. Day and H. Edelsbrunner. *Efficient algorithms for agglomerative hierarchical clustering methods*, pages 1:7–24. J. Classification, 1984.

I. Dhillon and D. Modha. *A data-clustering algorithm on distributed memory multiprocessors.*, pages 245–260. KDD'99 Workshop on High Performance Knowledge Discovery, 1999.

T. G. Dietterich. *Ensemble methods in machine learning*, pages 1–15. Lecture Notes in Computer Science, 2000.

M. J. Donahoo and G. D. Speegle. *SQL. Practical Guide for Developers.* Morgan Kaufmann, 2005.

E. Durfee. *Distributed Problem Solving and Planning*, pages 121–164. MIT Press, Cambridge, 1999.

M. Ester, H-P. Kriegel, J. Sander, and X. Xu. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, pages 226–231. AAAI Press, 1996.

B. S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis.* Wiley Series in Probability and Statistics, 2011.

I. A. Ferguson. *Towards an Architecture for Adaptive, Rational, Mobile Agents.* Third European Workshop on Modelling Autonomous Agents and Multi-Agent World, 1991.

T. Finin, R. Fritzson, D. McKay, and R. McEntire. *KQML as an Agent Comunication Language.* Third International Conference on Information and Knowledge Management, 1994.

FIPA. *FIPA Modeling: Interaction Diagrams.* Foundation For Intelligent Physical Agents, 2003.

IEEE FIPA. The foundation for intelligent physical agents, June 2005. URL http://www.fipa.org/.

G. Gan, C. Ma, and J. Wu. *Data Clustering. Theory, Algorithms, and Applications.* Society for Industrial and Applied Mathematics, 2007.

M. R. Genesereth and S. P. Ketchpel. *Software Agents.* CIFE Working Paper, 1994.

M. Georgeff. *Communication and Interaction in Multi-Agent Planning.* 3th National Conference on Artificial Intelligence, 1983.

M. Georgeff. *Theory of Action for Multi Agent Planning*, pages 121–125. 4th National Conference on Artificial Intelligence, 1984.

I. G. Goikoetxea. *Aportaciones a la clasificación no supervisada y a su validación. Aplicación a la seguridad informatica.* PhD thesis, Universidad del País Vasco, Donostia, 2010.

A.D. Gordon. *Classification.* Chapman and Hall/CRC, Boca Raton, FL, 2 edition, 1999.

D. Grimshaw. Jade administration tutorial, 2010. URL http://jade.tilab.com/doc/tutorials/JADEAdmin/jadeArchitecture.html.

R. L. Grossman, S. Bailey, A. Ramu, B. Malhi, and A. Turinsky. *The preliminary design of Papyrus: a system for high performance, distributed data mining over clusters.* MIT Press, 2000.

Y. Guo and J. Sutiwaraphun. *Probing Knowledge in Distributed Data Mining.* Springer-Verlag, 1999.

M. Halkidi, Y. Batistakis, and M. Vazirgiannis. *On Clustering Validation Techniques.* Kluwer Academic, 2001.

H. Hamilton. Computer science 831: Knowledge discovery in databases, 2009. URL http://www2.cs.uregina.ca/~dbd/cs831/notes/clustering/clustering.html.

J. Han and M. Kamber. *Data Mining, Concepts and Techniques.* Morgan Kaufmann, 2006.

J. Han, M. Kamber, and J. Pei. *Data Mining, Concepts and Techniques.* Morgan Kaufmann, 3 edition, 2012.

D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

P. Hansen and B. Jaumard. *Cluster Analysis and Mathematical Programming*, volume 79, pages 191–215. Mathematical Programming: Series A and B, 1997.

D. Helbing and S. Balietti. From social data mining to forecasting socio-economic crises. *The Europeans Physical Journal*, 2011.

B. Henderson-Sellers and P. Giorgini. *Agent-Oriented Methodologies.* Idea Grupo Inc, 2005.

N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. Jack intelligent agents - summary of an agent infrastructure. *5th ACM International Conference on Autonomous Agents*, 2001.

Ilango and V. Mohan. *A survey of Grid Based Clustering Algorithms*, volume 8, pages 3441–3446. International Journal of Engineering Science and Technology, 2010.

P. Jaccard. *Étude comparative de la distribution florale dans une portion des Alpes et des Jura.* Bulletin de la Société Vaudoise des Sciences Naturelles, 1901.

R. Jain. *Rules Generation using Decision Trees.* Indian Agricultural Statistics Research Institute, 2001.

E. Januzaj, H. P. Kriegel, and M. Pfeifle. *DBDC: density based distributed clustering.* Computer Science, 2004.

N.R. Jennings and M. Wooldridge. *Agent-Oriented Software Engineering*, volume 117, pages 277–296. Artificial Intelligence, 2000.

G. Ji and X. Ling. *Ensemble Learning Based Distributed Clustering*, volume 4819 of *Lecture Notes in Computer Science*, pages 312–321. Emerging Technologies in Knowledge Discovery and Data Mining, 2007.

Jiang-Lingxia. Research on distributed data mining system and algorithm based on multi-agent. Master's thesis, Université du Québec à Chicoutimi, 2009.

E. Johnson and H. Kargupta. *Collective, hierarchical clustering from distributed heterogeneous data*, pages 221–244. Springer-Verlag-V, 1999.

S. C. Johnson. *Hierarchical Clustering Schemes.* Hierarchical Systems. Computer Journal, 1967.

H. Kargupta, I. Hamzaoglu, and B. Stafford. *Sacalable, distributed data mining using an agent based architecture*, pages 211–214. Knowledge Discovery And Data Mining, Melo Park, CA, d. heckerman, h. mannilla, d. pregibon, r. uthurusamy edition, 1997.

H. Kargupta, B. H. Park, D. Hershberger, and E. Johnson. *Collective Data Mining: A New Perspective Toward Distributed Data Mining*, pages 133–184. Advances in Distributed and Parallel Knowledge Discovery, 1999.

H. Kargupta, B. H. Park, S. Pittie, L. LIu, D. Kushraj, and K. Sarkar. *Mobimine: monitoring the stock market from a PDA*, pages 37–46. SIGKDD Explorations Newletter, 2002.

H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. *Distributed clustering using collective principal component analysis.* Knowledge Inform., 2003.

L. Kaufman and P. J. Rousseeuw. *Clustering by means of Medoids*, pages 405–416. Statistical Data Analysis Based on the L1–Norm and Related Methods., 1987.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons, Inc., 1990.

M. Klush, S. Lodi, and G. Moro. *Distributed clustering based on sompling local density estimates.* Knowledge Inform., 2001.

F. Kovács, Csaba Legány, and Attila Babos. *Cluster Validity Measurement Techniques*, pages 388–393. 5th. WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, 2006.

C. Kuengkrai and C. Jaruskulchai. *A parallel learning algorithm for text classification.* 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 2002.

V. Kumar. *Network intrusion detection using distributed data mining.* Workshop on Data Mining and Exploration Middleware for Distributed and Grid Computing, Minneapolis, MN, 2003.

M. Laclavík, Z. Balogh, M. Babík, and L. Hluchy. *Agentowl: Semantic Knowledge Model and Agent Architecture*, volume 25. Computig and Informatics, 2006.

D.T. Larose. *Discovering Knowledge in Data.* John Wiley and Sons, Inc., 2005.

L. Lepistö, I. Kunttu, J. Autio, and A. Visa. *Combining Classifiers in Rock Image Classification – Supervised and Unsupervised Approach.* Advanced Concepts for Intelligent Vision Systems, Brussels, Belgium, 2004.

C. R. Lin, C. H. Lee, M. S. Chen, and P. S. Yu. *Distributed data mining in a chain store database of short transactions*, pages 576–581. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Canada, 2002.

Y. Lindell and B. Pinkas. *Privacy preserving data mining*, pages 177–206. Journal of Cryptology, crypto 2000 conference edition, 2002.

G.S. Linoff and M. J. A. Berry. *Data Mining Techniques, For Marketing, Sales, and Customer Relationship Management.* Wiley Publishing, Inc., 2011.

B. Liu. *Web Data Mining.* Springer, 2011.

Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu. *Understanding of Internal Clustering Validation Measures.* IEEE International Conference on Data Mining, 2010.

S. P. Lloyd. *On computing, storing and querying frequent patterns*, pages 607–612. Conf. Knowledge Discovery and Data Mining. ACM SIGKDD, Washington, 2003.

J. B. MacQueen. *Some methods for classification and analysis of multivariate observations.* Fifth Berkeley Symposium on Mathematical Statistics and Probability., 1967.

O. Maimon and L. Rokach. *The Data Mining and Knowledge Discovery Handbook*, chapter 9, pages 321–352. Springer, 2005.

C. D. Manning and H. Schütze. *Foundations of Statical Natural Langage Processing.* Massachusetts Institute of Technology, 1999.

P. Moraitis, E. Petraki, and N. I. Spanoudakis. *Engineering JADE Agents with the Gaia Methodology.* Agent Technologies, Infrastructures, Tools, and Applications for e-Services., 2003.

S. Muggleton. *Inductive Logic Programming.* Academic Press Ltd, 1992.

J.P. Muller, M. Pischel, and M. Thiel. *Modelling Reactive Behaviour in Vertically Layered Agent Architectures.* Springer-Verlag, Heidelberg, 1995.

F. Murtagh. *A survey of recent advances in hierarchical clustering algorithms.*, pages 354–359. Computer Journal, 26 edition, 1984.

J. Mylopoulos, M. Kolp, and J. Castro. *UML for Agent-Oriented Software Development: The Tropos Proposal*, pages 422–441. 4th. International Conference on the Unified Modeling Language, Modeling Langauges, Concepts, and Tools, 2001.

H. S. Nwana and M. Wooldridge. *Software Agent Technologies.* BT Technology Journal, 1996.

H. S. Nwana, D. T. Ndumu, L. C. Lee, and J. C. Collis. Zeus: A collaborative agents tool-kit. 1998.

H.S. Nwana, L. Lee, and N.R. Jennings. *Coordination in Software Agent Systems*, pages 79–88. BT Technology Journal, 1996.

T. Oates, M. D. Schmill, and P. R. Cohen. Parallel and distributed search for structure in multivariate time series. *In Proceedings of the Ninth European Conference on Machine Learning*, 1996.

J. Odell, H. V. D. Parunak, and B. Bauer. Extending uml for agents. *In Proc. of the Agent-Oriented Information Systems, Workshop at the 17th National conference on Artificial Intelligence.*, 2000.

J. J. Odell and H. V. D. Parunak. *Representing Agent Interaction Protocols in UML.* Springer-Verlag, 2001.

AFOSR. Air Force Office of Scientific Research. Multiagent and cooperative reasoning laboratory. URL http://macr.cis.ksu.edu/mase.

A. Omicini. *SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems.* Springer-Verlag, 2000.

S. Paul. Parallel and distributed data mining. Technical report, Karunya University, Coimbatore, India, Junuary 2011.

J. Pérez-Ortega, A. Martínez, E. Iturbide, and C. Zavala-Díaz. An epidemiological data mining application based on census databases. *DBKDA 2013 : The Fifth International Conference on Advances in Databases, Knowledge, and Data Applications*, 2013.

D. L. Poole and A. K. Mackworth. *Artificial Intelligence. Foundations of Computational Agents.* Cambridge University Press, 2010.

S. Poslad, P. Buckle, and R. Hadingham. The fipaos agent platform: Open source for open standards. 2000.

A. L. Prodromidis, P. K. Chan, and S. J. Stolfo. *Meta-learning in distributed data mining systems: Issues and approaches.* AAAI Press. Advances of Distributed Data Mining, 2000.

F. Provost and D. Hennessy. *Scaling up: distributed machine learning with cooperation.* 13th. National Conference on Artificial Intelligence, 1996.

J. R. Quinlan. *Simplifying Decision Trees.* International Journal of Man-Machine Studies, 1987.

A. S. Rao and M. Georgeff. *BDI Agents: from Theory to Practice*, pages 312–319. Firts International Conference on Multi-Agents Systems, San Francisco, 1995.

V. S. Rao. *Multi Agent-Based Distributed Data Mining: An Over View.* International Journal of Reviews in Computing, 2010.

H.C. Romesburg. *Cluster Analysis for Researchers.* Lifetime Learning Publications, 1984.

S. Rusell and P. Norvig. *Artificial Intelligence. A Modern Approach.* Person Education, Inc., 2 edition, 2003.

N. Samatova, G. Ostrouchov, A. Geist, and A. Malechko. *RACHET: An efficient cover-based merging of clustering hierarchies from distributed datasets.* Parallel Databases, 2002.

N. Sánchez-Pi, J. Carbó, and J.M. Molina. Analysis and design of a multi-agent system using gaia methodology in an airport. case of use. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 14(45):9–17, 2010.

E. Schikuta. *Grid-Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets*, volume 2, pages 101–105. 13th. International Conference on Pattern Recognition, 1996.

G. Sheikholeslami, S. Chatterjee, and A. Zhang. *WaveCluster: a wavelet-based clustering approach for spatial data in very large databases*. Springer-Verlag, 2000.

E. C. Shek, R. R. Muntz, E. Mesrobian, and K. W. Ng. *Scalable exploratory data mining of distributed geoscientific data*, pages 32–37. 2nd. International Conference on Knowledge Discovery and Data Mining, Portland, OR, 1996.

Y. Shoham. *Agent-Oriented Programming*. Number 42. Stanford University, 1990.

C. Shyam, 2012. URL https://www.classle.net/projects/k-medoids.

SIAM. *Proceedings of the 3rd SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 2003.

A. Silberschatz, H. F. Korth, and S. Sudarshan. *Fundamentos de Bases de Datos*. Mac Graw Hill, 4 edition, 2002.

J. C. Da Silva, C. Giannella, R. Bhargava, H. Kargupta, and M. Klush. *Distributed Data Mining and Agents*, pages 791–807. Engineering Applications of Artificial Intelligence, 2005.

F. Siraj and M. A. Abdoulha. *Mining Enrolment Data Using and Deescriptive Approaches*. Knowledge-Oriented Applications in Data Mining, 2011.

R. Smith and R. Davis. *The Contract Net protocol: High Level Communication and Control in a Distributed Problem Solver*, pages 1104–1113. IEEE Transactions on Computers, 1980.

P.H. Sneath and R.R. Sokal. *Numerical Taxonomy*. W. H. Freeman, San Francisco, 1973.

M. Sohail. Madkit (multi-agent development kit) : A generic multi-agent platform, 2005. URL http://perso.limsi.fr/jps/enseignement/examsma/2005/1.plateformes_2/SOHAIL/SOHAIL.htm.

R.R. Sokal and C.D. Michener. *A Statistical Method for Evaluating Systematic Relationships*, pages 1409–1438. he University of Kansas Scientific Bulletin 38, 1958.

A. Srivastava, E. Han, and V. Kumar adn V. Singh. *Parallel formulations of decision-tree classification algorithms*, pages 237–261. Data Mining and Knowledge Discovery, 1999.

StackExchange, 2013. URL http://stackoverflow.com/questions/11150523/clustering-with-cosine-similarity.

L. S. Sterling and Kuldar Taveter. *The Art of Agent-Oriented Modeling*. The MIT Press, 2009.

S. Stolfo. *Java agents for meta-learning over distributed databases*. AAAI Press, 1997.

S. Sumathi and S.N. Sivanandam. *Introduction to Data Mining and its Applications*. Springer, 2006.

D. Swagatam, A. Ajith, and K. Amit. *Metaheuristic Clustering*. Springer-Verlag, 2009.

K. P. Sycara. *Multiagent Systems*, volume 19 of *Intelligent Agents Summer*. AI Magazine, 1998.

P. N. Tang, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

LLC Tangient. Programación en jade, 2010. URL http://programacionjade.wikispaces.com/home.

K. Teknomo. Ejemplo numérico de agrupamiento k-medias, 2007. URL http://people.revoledu.com/kardi/tutorial/kMean/EjemploNumerico.htm.

G. Tsoumakas and I. Vlahavas. Distributed data mining. *Encyclopedia of Data Warehousing and Mining*, 2:709–715, 2008.

UAZ. Unidad académica de matemáticas, 2012. URL http://matematicas.reduaz.mx/home/Docentes/ltrueba/arqueo/arq7.htm.

J. Vaucher and A. Ncho. Jade tutorial and primer, 2004. URL http://www.iro.umontreal.ca/~vaucher/Agents/Jade/Ontologies.htm#7.2.

N. K. Visalakshi and K. Thangavel. *Distributed Data Clustering: A Comparative Analysis*, volume 6, pages 371–397. Springer, 2009.

L. Wang and X. Fu. *Data Mining with Computational Intelligence*. Springer-Verlag, 2005.

W. Wang, J. Yang, and R. Muntz. *STING: A Statistical Information Grid Approach to Spatial Data Mining*, pages 186–195. Morgan Kaufmann Publishers, San Francisco, 23th. international conference on very large data bases edition, 1997.

G. Weiss. *Multiagent Systems. A Modern Approach to Distributed Modern Approach to Artificial Intelligence.* MIT Press, 1999.

M. Winikoff. Jack intelligent agents: An industrial strength platform. *Multi-Agent Programming*, pages 175–193, 2005.

WinterSchool. *Classification Using Decision Trees.* Data Mining Techniques and Tools for Knowledge Discovery in Agricultural Datasets, 2011.

R. Wirth, M. Borth, and J. Hipp. *When distribution is part of the semantics: a new problem class for distributed knowledge discovery*, pages 56–64. PKDD-2001, workshop on ubiquitous data mining for mobile and distributed enviroments edition, September 2001.

I. H. Witten and E. Frank. *Data Mining, Practical Machine Learning and Techniques.* Morgan Kaufmann, 2005.

M. Wooldridge. *An Introduction to Multiagent Systems.* John Wiley and Sons, 2002.

M. Wooldridge and N.R. Jennings. *Intelligent agents: theory and practice.*, pages 115–152. Springer-Verlag, 1995.

M. Wooldridge, N.R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:285–312, 2000.

P. Yang, L. Tao, L. Xu, and Z. Zhang. Multiagent framework for bio-data mining. *RSKT '09 Proceedings of the 4th International Conference on Rough Sets and Knowledge Technology*, 2009.

F. Zambonelli, N.R. Jennings, and M. Wooldridge. *Multi-Agent Systems as Computational Organizations: The Gaia Methodology*, chapter 6, pages 136–171. Idea Grupo Inc, 2005.

Q. Zhang and R. S. Segall. Commercial data mining software. *Data Mining and Knowledge Discovery Handbook*, 2010.