CAPÍTULO 2 Minería de datos y Conceptos generales

2.1 Minería de datos

Ya se definió brevemente lo que es la minería de datos, pero ahora conviene elaborar un poco más sobre el tema.

Se comentó anteriormente que la minería de datos es el proceso de extracción de conocimiento partiendo de un repositorio o almacén de datos. Sin embargo, hay gente que clasifica a la minería de datos más bien como un paso más en el proceso de la extracción de conocimiento. Pero como el término “minería de datos” es más corto, se ha hecho más popular entre la comunidad computacional que le término “Knowledge Discovery in Databases”.

Generalmente, se acepta que la minería de datos consta de 7 pasos esenciales:

1) Limpieza de los datos
2) Integración de los datos
3) Selección de los datos
4) Transformación de los datos
5) Minería de datos
6) Evaluación de patrones
7) Presentación del conocimiento

Se puede realizar una minería sobre una amplia gama de bases de datos, conteniendo desde luego distintos tipos de datos:

- Bases de datos relacionales
- Bases de datos transaccionales
- Bases de datos orientadas a objetos
- Bases de datos espaciales o geográficas
- Bases de datos multimedia
- Minería basada en grafos

Y muchas otras más.

2.2 Grafos y su representación

2.2.1 GRAFOS

Un grafo es un conjunto de objetos llamados vértices unidos por enlaces llamados aristas [1]. Típicamente, se representa por un conjunto de puntos (los vértices) unidos por líneas (las aristas). Los vértices de un grafo, por definición, deben de estar relacionados unos con otros. Los grafos son datos estructurales que representan relaciones entre objetos.

Dentro de un grafo, los vértices pueden representar a cualquier objeto, mientras que las aristas representan la relación existente entre esos objetos. De manera general, se dice que un grafo es dirigido cuando todas sus aristas tienen una sola dirección, mientras que es no-dirigido cuando al menos una arista toma ambas direcciones (ida y vuelta). En este proyecto, se trabaja exclusivamente con grafos dirigidos.

Un grafo ejemplo se puede utilizar para describir la relación entre los objetos geométricos mostrados en la siguiente figura:

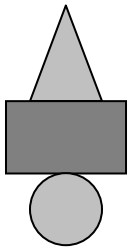


Figura 2.1

El grafo generado sería:

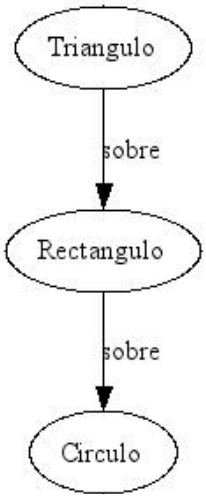


Figura 2.2

Naturalmente, pueden existir grafos mucho más complejos y que representen otro tipo de relación.

2.2.2 SUBGRAFOS

Un subgrafo es un grafo que pertenece a, o está contenido en, otro grafo original. En otras palabras, podemos decir que un grafo ***A*** contiene dentro de sí a un grafo ***B***, pero ***B*** no contiene a ***A***. Entonces se dice que ***B*** es un subgrafo de ***A***.

Al decir que un vértice puede representar a cualquier objeto, es bastante literal, ya que incluso puede representar a otro grafo. Entonces, si un grafo puede ser representado por un vértice, esto nos lleva a la conclusión de que existen dos tipos de representación de subgrafos: la simplificada y la expandida.

Tomemos como ejemplo el grafo de la figura 2.2 y asignémosle como nombre grafo ***B***. Imaginémonos que ese grafo está contenido en otro grafo superior ***A***, el cual contiene un vértice llamado “Cuadro” que tiene una relación “cerca de” con ***B***. La representación expandida de este grafo contendría a todos los vértices de ***A*** y a todos los vértices de ***B***, mientras que en la representación simplificada sustituiremos a ***B*** con un vértice llamado SUB_1. Así, los grafos correspondientes quedan de la siguiente manera:

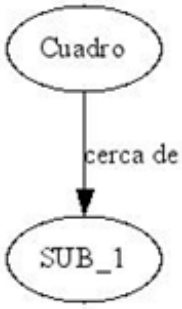


Figura 2.3 – Representación simplificada

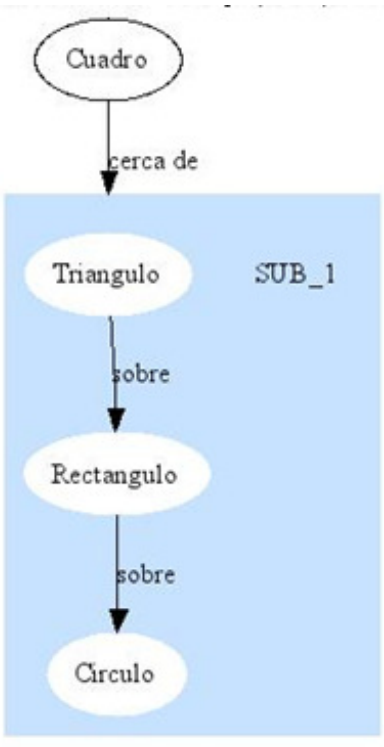


Figura 2.4 – Representación expandida

Naturalmente, pueden existir subgrafos dentro de los subgrafos, creado una serie de grafos anidados, como en el caso del siguiente ejemplo:

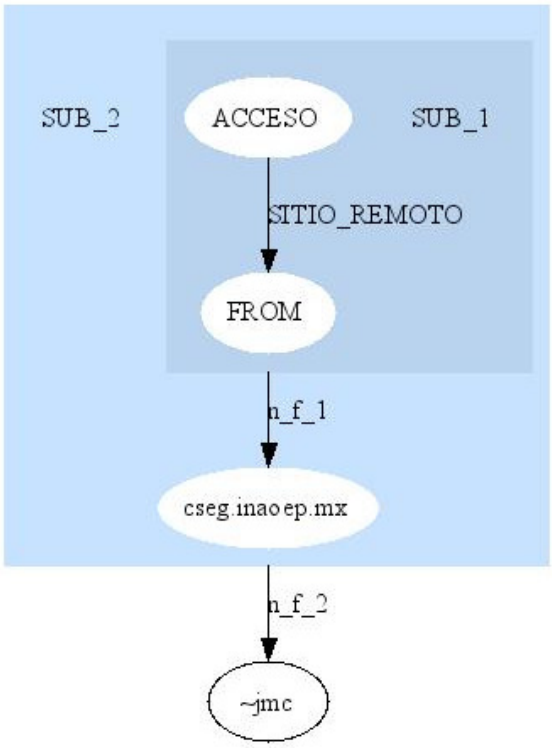


Figura 2.5

2.2.3 LENGUAJE DOT

Es un lenguaje estándar de descripción de grafos que es usado dentro de algunas aplicaciones para dibujar un grafo en la pantalla de una computadora. Este lenguaje describe grafos como texto, listo para ser leído e interpretado por alguna otra herramienta.

DOT reconoce tres tipos de objetos: nodos, aristas y subgrafos, y puede incluir un grupo de atributos para cada uno de esos elementos. Son estos atributos los que le confieren a DOT una gran flexibilidad al momento de crear el grafo, ya que los atributos pueden ser cosas tan simples como el tamaño de la fuente

de un nodo o cosas algo más complejas, como los distintos tipos de formas que puede tomar un nodo.

Un grafo muy sencillo se puede describir de la siguiente forma, siguiendo la gramática del lenguaje DOT:

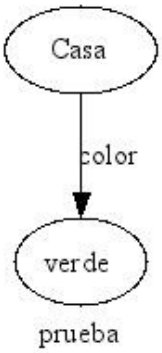
```
digraph prueba {
      graph [fontsize=12, label="prueba"];
      node [fontsize=12];
      edge [fontsize=12];
      graph [bb="0,0,305,144"];
      N1 [label=" Casa"];
      N2 [label=" verde"];
      N1 -> N2 [label="color"];
}
```

En este caso particular, se pueden observar las siguientes cosas:

- El grafo principal se llama “prueba”.
- El grafo “prueba” tiene dos atributos, “fontsize” y “label”.
- Todos los nodos de “prueba” van a tener un atributo “fontsize”, al igual que todas las aristas.
- Se declara un segundo grupo de atributos para el grafo, “bb” (Bounding Box del grafo).
- Se crean dos nodos, con nombres de “N1” y “N2”, con sus respectivas etiquetas (“label”).
- Se crea una arista (edge), que va del nodo “N1” al nodo “N2”, con una etiqueta que indica el nombre de la relación entre los nodos.

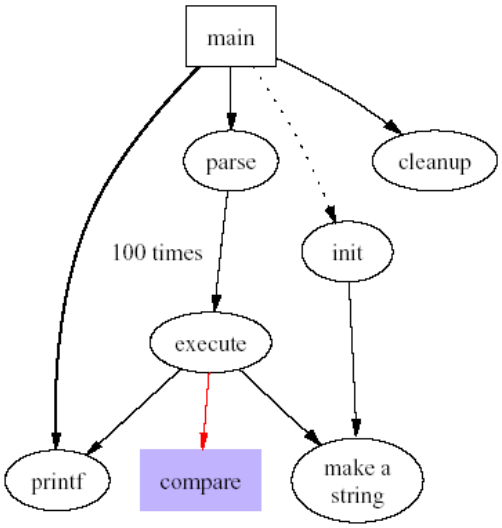
El grafo desplegado es el siguiente:

Figura 2.6



Se pueden crear grafos cada vez más complejos, como el de la siguiente imagen:

Figura 2.7



Y el código DOT necesario para crearlo fue:

```
1: digraph G {
2: size ="4,4";
3: main [shape=box]; /* this is a comment */
4: main -> parse [weight=8];
5: parse -> execute;
6: main -> init [style=dotted];
7: main -> cleanup;
8: execute -> { make_string; printf}
9: init -> make_string;
10: edge [color=red]; // so is this
11: main -> printf [style=bold,label="100 times"];
12: make_string [label="make a\nstring"];
13: node [shape=box,style=filled,color=".7 .3 1.0"];
14: execute -> compare;
15: }
```

Naturalmente, los grafos pueden ir aumentando en complejidad; tanto que, en grafos muy grandes y con muchos atributos, el tamaño del archivo DOT puede llegar a ser de más de 30 kb.

Cada objeto puede tener una cantidad indeterminada de atributos, de los cuales hay de dos tipos: los atributos de DOT; y los atributos creados por el usuario. Los atributos de DOT son aquellos que son importantes para DOT ya que afectan cómo se dibujan los grafos, como la forma de un nodo o el tipo de flecha de las aristas. Los creados por el usuario son atributos que están ahí pero que no son tomados en cuenta por DOT; es decir, un usuario puede crear un atributo con un nombre arbitrario.

Para propósitos de este proyecto, se requiere guardar en una base de datos los patrones resultantes, con todo y su grafo. Esto se podría lograr simplemente guardando las imágenes de los grafos, pero ahora podemos almacenar un grafo en forma de texto, reduciendo ampliamente el espacio requerido en disco duro. Por poner un punto de comparación, la imagen de la figura 2.7 tiene un tamaño de 41.8 kb, mientras que el archivo DOT requerido para dibujarlo, al ser sólo texto, ocupa tan solo **464 bytes** [definición de dot].

La descripción de la gramática requerida del formato DOT se encuentra en un anexo.

2.2.4 GRAPHVIZ Y HERRAMIENTA DOT

GraphViz (**Graph Viz**ualization) es un conjunto de herramientas que sirven para dibujar un grafo en la pantalla de una computadora o simplemente para

manipular su representación computacional. Estas son herramientas de acceso gratuito en Internet (Open Source, http://www.graphviz.org/).

Una de las herramientas disponibles por parte de GraphViz es la herramienta *dot*. Este programa permite al usuario transformar un archivo de salida de SUBDUE en un grafo descrito en el lenguaje de descripción de grafos DOT. Alternativamente, también hace posible la creación de archivos en formato *.jpg* con la representación gráfica completa de un grafo (o sea, una imagen del grafo). En general, *dot* permite crear grafos con formatos *.gif*, *.jpg*, *.png*, *.ps*, e incluso en SVG (*Scalable Vector Graphics*, lenguaje en texto para dibujar sobre páginas Web) para visualización en Internet.

Dentro del proyecto, se usa el programa dot para convertir los grafos dados por la salida de la aplicación de minería de datos a lenguaje DOT, para poder ser pintados en la pantalla de la computadora.

2.2.5 GRAPPA

Creado por John Mocenigo en los laboratorios de AT&T en Estados Unidos, Grappa (**Grap**h **Pa**ckage) es un paquete de dibujo de grafos hecho en Java que realiza cualquier función de manipulación o de presentación para cualquier aplicación Java [def Grappa]. Grappa es una de las herramientas disponibles en GraphViz.

Grappa contiene un conjunto de clases capaces de administrar toda la información relativa a un grafo. Es más, no se conforma con sólo dibujar un grafo en su panel especializado, sino que también crea objetos Java para almacenar ese grafo, como lo son las clases de *Node*, *Edge*, *Subgraph* y *Graph*.

Otras de las características de Grappa son:

- Contiene métodos para la creación, manipulación y presentación de grafos, nodos, aristas y subgrafos.
- Cada elemento del grafo puede tener asociado cualquier cantidad de atributos.
- Incluye métodos para leer grafos o escribirlos en archivos, usando obviamente el formato DOT.
- Provee funciones para manipular de manera interactiva los grafos, agregando o eliminando elementos.
- Provee también una función de ToolTip.
- Utiliza la herramienta *dot* para obtener las instrucciones finales de cómo dibujar un grafo en pantalla.

Es importante hacer notar que Grappa está implementado usando la librería *javax.swing*, y que el hacer uso de esta librería es uno de los requerimientos de este proyecto.

Esta librería también contiene un tipo especializado de **JPanel** (**GrappaPanel**) para poder mostrar los grafos y poder interactuar con ellos. Este proyecto hace uso de ese panel para mostrar los grafos de los patrones almacenados.

El diagrama completo de clases de Grappa se mostrará en un anexo.

2.3 Minería de datos basada en grafos

El enfoque principal de la minería de datos es buscar asociaciones o patrones dentro de los atributos de una entidad. Sin embargo, otra fuente muy rica en información y conocimiento es ***la relación*** existente entre dichas entidades [“Graph-based Data Mining”].

Muchas de las aplicaciones de minería de datos funcionan bien cuando se trabaja con datos numéricos o con una representación de atributo-valor o con texto. Sin embargo, cada vez es más común que los datos sean representados en forma estructural que pone de manifiesto relaciones existentes entre objetos. Hay varias formas de representar datos en forma estructurada: texto en HTML; árboles de cualquier tipo; relaciones lógicas; etc. Pero la manera más común para representar datos estructurales es mediante un grafo, que es también una de las topologías fundamentales de las matemáticas.

La minería de datos basada en grafos se enfoca en encontrar subestructuras repetitivas y comunes dentro de los datos de entrada (grafos). Pero, obviamente, no es sólo el encontrar subestructuras que se repiten una y otra vez, sino que también es el proceso de identificar conceptos que describen a las subestructuras más importantes para una mejor interpretación de los datos. Una vez descubierta, una subestructura puede usarse para simplificar el grafo original mediante el reemplazo de la subestructura por un vértice que represente a la recién descubierta subestructura.

Este tipo de minería puede ser especialmente útil, ya que muchos campos generan información que toma la forma de un grafo, como la biología, química, telecomunicaciones, etc [“State of the art of graph based data mining”].

2.4 Subdue

SUBDUE es un sistema de minería de datos basada en grafos que descubre subestructuras interesantes en datos estructurales [2]. Su nombre es un acrónimo de la palabra inglesa *Substructure Discovery* (descubrimiento de subestructuras), que es el proceso de identificar conceptos que describen subestructuras interesantes y repetitivas en datos estructurales [3].

Desarrollado en la Universidad de Texas en Arlington, este sistema basado en C representa datos estructurales en forma de grafos dirigidos. Una subestructura es un subgrafo conectado dentro de la representación gráfica [3]. Esta representación gráfica sirve de entrada para el sistema SUBDUE.

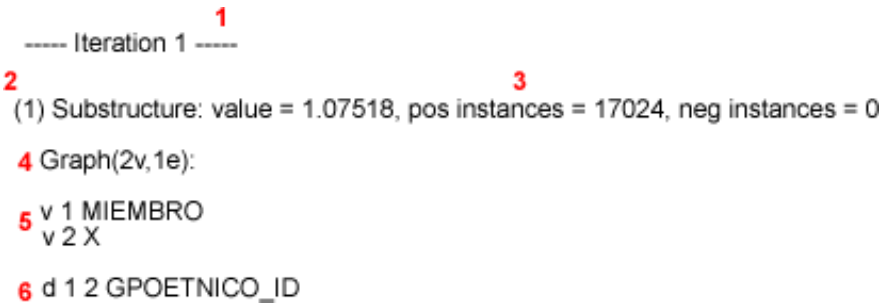
Un archivo de salida de SUBDUE contiene básicamente los mejores patrones encontrados en el grafo de entrada. Para cada subestructura de salida obtenida, se muestra la iteración en la que se obtuvo, el número de subestructura dentro de esa iteración, el número de instancias encontradas de esa subestructura en el grafo original y finalmente la descripción de la subestructura en sí. Esta descripción contiene una lista de vértices con el formato:

v <ID del vértice> <nombre del vértice>

y de aristas con el formato:

d <ID del vértice origen> <ID del vértices destino> <nombre de la arista>

Debido a que las aristas muestran la relación entre los vértices que conectan, el nombre de la arista es también el nombre de la relación. Un ejemplo de uno de los registros del archivo de salida de SUBDUE es el siguiente:



1) Número de iteración
2) Número de la subestructura encontrada dentro de la iteración
3) Instancias encontradas en el grafo original
4) Cantidad de vértices y aristas (en este caso, 2 vértices y 1 arista)
5) Lista de los vértices
6) Lista de las aristas

Los grafos con los que se trabaja en este proyecto se obtienen por medio de SUBDUE. Un ejemplo completo de un archivo de salida de SUBDUE se encuentra como anexo.

Crearemos un ejemplo para darnos una idea de los patrones que encuentra SUBDUE. La siguiente figura:

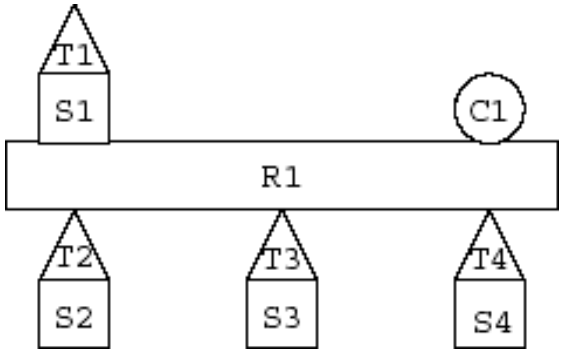


Figura 2.8

es representada por el siguiente grafo:

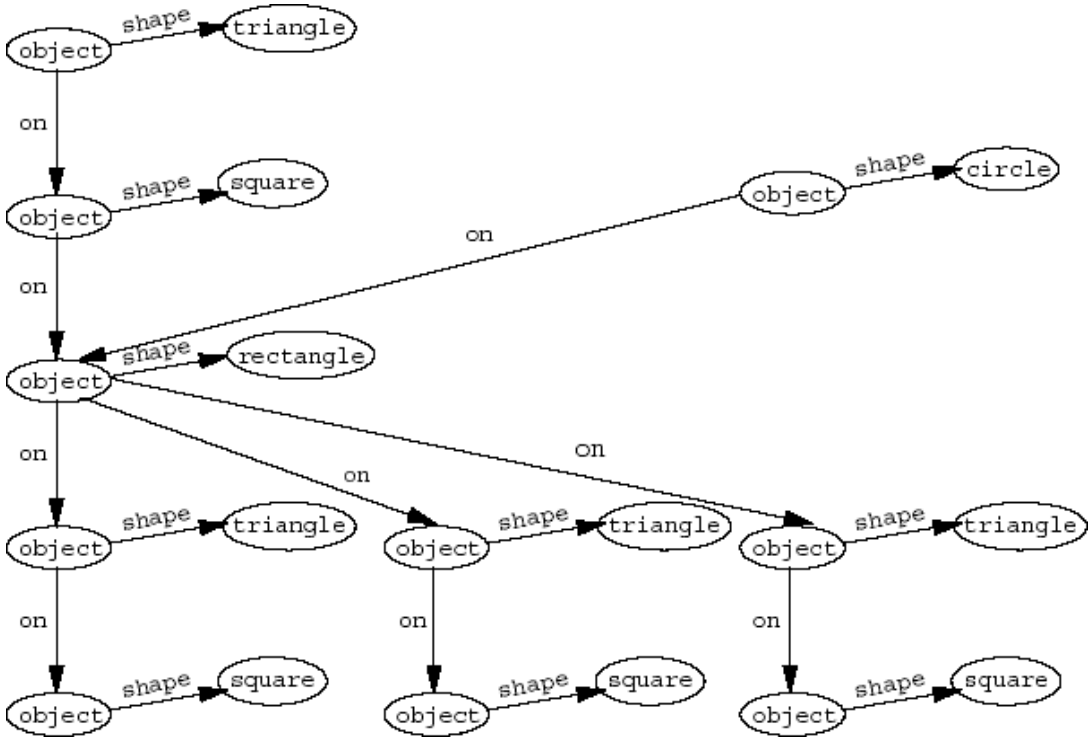


Figura 2.9

Este grafo se le da como entrada a SUBDUE para realizar una minería sobre él y encontrar el patrón más repetido, y SUBDUE nos ofrece el siguiente resultado:

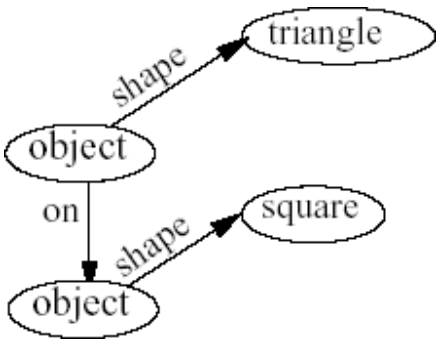


Figura 2.10

Esta subestructura se encuentra cuatro veces dentro del grafo original [3].

2.5 Minería de datos espaciales

2.5.1 DATOS GEOGRÁFICOS

Un dato geográfico, también llamado dato espacial, tiene la característica de ocupar un lugar (real o virtual) en el espacio. Este lugar en el espacio puede ser un río, una montaña, otro planeta o incluso un órgano humano dentro de un mapa de imágenes médicas. Básicamente, un dato geográfico representa información topológica de algún tipo.

Existen varios tipos de datos geográficos, desde los más simples que podrían ser un punto y una línea, hasta tipos complejos que representen toda la información de una región: su forma; regiones vecinas; temperatura promedio; número de habitantes; relieve topográfico; etc.

2.5.2 BASES DE DATOS GEOGRÁFICAS

Una base de datos geográfica o espacial contiene una gran cantidad de (obviamente) datos geográficos, como mapas, imágenes médicas, etc. Básicamente, una base de datos geográfica es una base de datos que contiene información geográfica acerca de un área y materia en particular [Geo Inf Sys & Sci].

Una base de datos geográfica se organiza en *capas* (o clases de objetos), que es una colección de datos sobre un tema en específico. Por ejemplo, puede existir una capa de las tuberías de agua, otra que contenga los polígonos del relieve del lecho de un río, otra con valores de altitud, etc [ ]. Cada una de estas capas está almacenada en una tabla de la base de datos. Obviamente, una base de datos geográfica es usada por un Sistema de Información Geográfica(GIS, por sus siglas en inglés).

2.5.3 MINERÍA DE DATOS ESPACIALES

Este término se refiere a la extracción de conocimiento, relaciones espaciales, u otros patrones interesantes que no se ven de forma explícita en una base de datos geográfica [Data Mining]. Este tipo de minería puede usarse para para un mejor entendimiento de datos espaciales, para descubrir relaciones entre datos espaciales o entre datos espaciales con no espaciales, etc. Estos datos pueden ser usados en cualquier campo que requiera información geográfica, como navegación, astronomía, control de tráfico, expansión de ciudades, etc.