Clustering:
Aplicación a Ruteo de Vehículos

TALLER V

Docentes: Libertad Tansini, Omar Viera
Departamento de Investigación Operativa
IN.CO.
Facultad de Ingeniería
Universidad de la República

Febrero 2001

Roberto Cabalo ***Silvana Caetano***

Índice

Resumen

Este trabajo pertenece al Taller V de la Facultad de Ingeniería de la Universidad de la República Oriental del Uruguay, y esta planteado dentro del Departamento de Investigación Operativa del Instituto de Computación.

El mismo se divide en dos partes: Investigación y Desarrollo. Como resumen de la investigación se ofrece una reseña del Clustering actual. Para ello se estudiaron los métodos y se clasificaron los diversos algoritmos existentes.

El objetivo del Clustering es ordenar los datos en grupos o clusters a través de un criterio de similitud de tal forma que el grado de asociación sea fuerte entre los miembros del mismo cluster y débil entre los miembros de clusters distintos. Cada cluster entonces, describe en término de datos relevados, la clase a la cual sus miembros pertenecen.

Dentro de los métodos existentes para Clusterizar destacamos los de Clustering Jerárquico, de Particionamiento y Basados en Grafos.

En particular, se trabajó con el método de Particionamiento, centrando nuestro trabajo en el estudio, implementación y comparación de los siguientes algoritmos: KNN, K-Means y PAM.

Este estudio resulta particularmente importante debido a que estas técnicas de Clustering nunca fueron aplicadas como preprocesamiento de datos para un Problema de Ruteo de Vehículos con Múltiples Depósitos y restricciones de Ventanas de Tiempo.

Como resultado del proceso de desarrollo de software se presenta la implementación de los algoritmos de Clustering arriba mencionados aplicados al preprocesamiento de un Problema de Ruteo de Vehículos con Múltiples Depósitos y Restricciones de Ventanas de Tiempo, aportando un nuevo punto de vista a la resolución de este tipo de problemas. También se ofrecen conclusiones y comparaciones de los tres algoritmos implementados mostrando sus debilidades y fortalezas al aplicarlos en la resolución de este problema puntual.

1 Introducción

En la actualidad, la clasificación ha dejado de ser una actividad espontánea y se ha disciplinado hasta convertirse en un área sistematizada y cuantificada (utilizada por todas las ramas de las Ciencias) donde los datos recopilados pueden ser reagrupados obedeciendo ciertos criterios para su posterior interpretación.

Esta disciplina ha evolucionado y se ha perfeccionado en las ultimas décadas, lo cual se puede observar a través de los diversos métodos y algoritmos planteados (y su evolución en el tiempo). También es importante destacar la diversidad de ramas surgidas para clasificar información: Pattern Mining (PM), Data Mining (DM), Clustering, Knowledge Database Discoveries (KDD), etc.

La aplicación de estas técnicas de clasificación es lo que permite obtener conocimiento a partir de una bolsa inicial de datos que por si misma no brinda información relevante. Si el objetivo final es tratar de agrupar estos datos en un conjunto finito de categorías que los describa, entonces la técnica apropiada es Clustering.

Este trabajo pertenece al Taller V de la Facultad de Ingeniería de la Universidad de la Republica Oriental del Uruguay, y esta planteado dentro del Departamento de Investigación Operativa del Instituto de Computación.

El mismo colabora con la Tesis de Maestría ("Algoritmos de Asignación de Problema de Ruteo de Vehículos con Múltiples Depósitos y restricciones de Ventanas de Tiempo") de la docente Libertad Tansini. Dicha Tesis apunta a resolver un Problema de Ruteo de Vehículos con Múltiples Depósitos y restricciones de Ventanas de Tiempo (MDVRPTW, del Inglés: Multi-Depot Vehicle Routing Problem with Time Windows constraints) donde se cuenta con n depósitos, m clientes (n, m > 1) y sus horarios de servicio.

En general existen dos métodos para resolver un MDVRPTW:

1) rutear al mismo tiempo que se asignan los clientes a los depósitos (el orden de este método lo hace inviable para entradas muy grandes) o

2) realizar la asignación de clientes a depósitos formando grupos más pequeños y para cada uno de ellos generar un ruteo.

Este proyecto tiene como propósito generar -mediante la aplicación de técnicas de Clustering- los grupos de la segunda variante de resolución del MDVRPTW. Hasta el momento, no se ha utilizado la técnica de Clustering para este tipo de problemas, por lo tanto este estudio resulta novedoso e innovador.

El problema planteado es de recolección/distribución en el cual los depósitos cuentan con demandas y los clientes con producciones, recolectándose la producción de estos para satisfacer la demanda de los primeros. Es posible la existencia de clientes y depósitos incompatibles (las causas pueden ser varias: que un cliente produzca un articulo no demandado por un depósito, que sus horarios no coincidan, etc.)

Los clusters (hasta ahora llamados grupos) resultantes tendrán determinadas características. Cada uno de ellos deberá contener un único depósito y n clientes compatibles con el mismo, tales que la sumatoria de sus producciones no podrá superar la demanda de dicho depósito.

Los algoritmos utilizados para obtener los clusters -a solicitud del usuario- son *KNN, K-Means y PAM.* Estos algoritmos resultan particularmente interesantes no solo por su funcionamiento sino porque son innovadores como técnica de preprocesamiento para un problema de MDVRPTW. Por lo tanto este trabajo además de resolver el problema planteado por el usuario pretende explorar nuevas aplicaciones de los métodos de Clustering.

En particular, el criterio en el cual se basó la comparación de los algoritmos es la minimización de la distancia total recorrida luego de correr un VRP para cada cluster obtenido y sumar las distancias locales.

Las soluciones generadas con K-Means y PAM son altamente recomendables para asignar clientes a depósitos en el segundo método planteado de resolución de un MDVRPTW mientras que la solución ofrecida por KNN no es buena salvo en algunas excepciones.

El informe cuenta con cinco secciones: *Desarrollo de la Aplicación* (que incluye análisis de requerimientos, diseño, pseudocódigo de los algoritmos a implementar, implementación y testing de los mismos). En los *Resultados* se realiza un análisis de los datos de salida obtenidos para cada algoritmo y se realiza una comparación entre ellos. A continuación se presentan las *Conclusiones Generales* y sugerencias para *Trabajos Futuros*.

Finalmente, en los *Anexos* se podrá leer un estudio histórico y las perspectivas actuales de las técnicas de Clustering. Allí se podrá encontrar información sobre los distintos enfoques de Clustering, los métodos y algoritmos existentes y un cuadro comparativo con las distintas características de los algoritmos. También se presenta un relevamiento del software existente, otros casos de testeo, el manual de usuario y las referencias bibliográficas utilizadas.

2 Desarrollo de la Aplicación

2.1 Análisis de Requerimientos

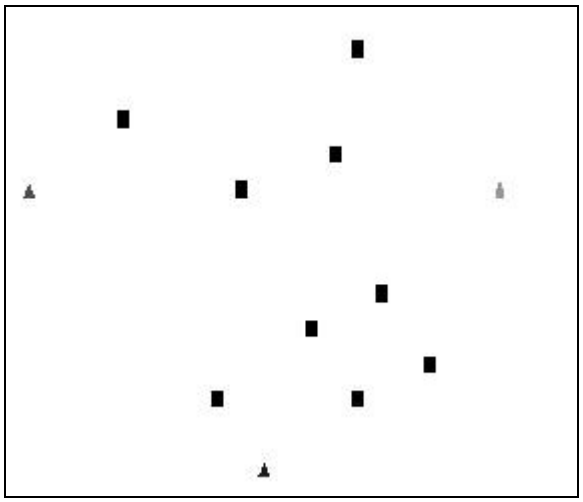
2.1.1 Descripción del Problema

El problema global planteado es la resolución de un Ruteo de Vehículos con Múltiples Depósitos y restricciones de Ventanas de Tiempo (MDVRPTW) de acuerdo a los siguientes pasos:

1) Asignar Clientes a Depósitos generando subgrupos más pequeños.
2) Realizar un ruteo sobre cada subgrupo generado.

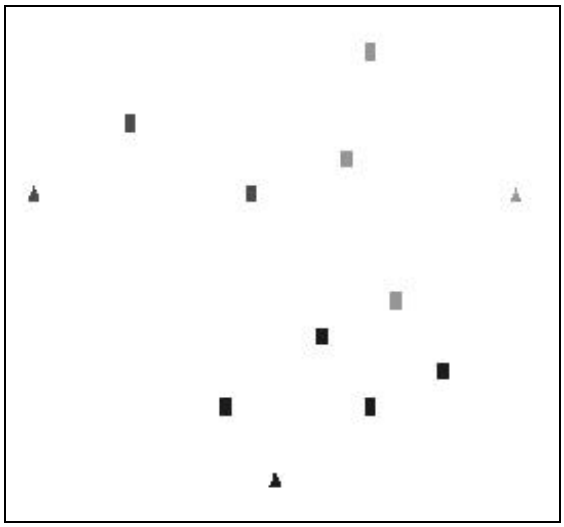
Nuestro trabajo consiste en resolver el primer punto generando la información necesaria para la segunda etapa.

A continuación se presenta un ejemplo de la asignación de clientes a depósitos. En la siguiente figura se presentan tres depósitos, representados con un triángulo de color, y nueve clientes representados con cuadrados negros (este color significa que están sin asignar).



Distribución inicial

En la próxima figura se presentan los clientes coloreados de acuerdo al depósito asignado. Este es el resultado obtenido luego de clusterizar.



Asignación final

Es a partir de la asignación final que quedan los clusters listos para aplicar el VRP para cada depósito y su conjunto de clientes. La suma de los VRP/depósito genera el resultado final.

2.1.2 Requerimientos

Los requerimientos planteados consisten en un conjunto de *datos de entrada*, la elección por parte del usuario del *método aplicable* y un conjunto de *datos de salida*.

1) Datos de Entrada

Se requiere información de depósitos y clientes.

- Depósito:
  - Identificador de deposito
  - Ubicación geográfica en coordenadas cartesianas
  - Centro de la ventana de disponibilidad de tiempo útil
  - Demanda a satisfacer en la unidad que corresponda
  - Cantidad de depósitos

- Cliente:
  - Identificador de cliente
  - Ubicación geográfica en coordenadas cartesianas
  - Centro de la ventana de disponibilidad de tiempo útil
  - Producción con la que cuenta en la unidad que corresponda
  - Lista de depósitos no compatibles con el cliente
  - Cantidad de clientes

Esta información se presenta en un archivo con formato ASCII cuya estructura se detalla a continuación:

Formato de Archivos de Datos de Entrada:

```
#Depósitos
#Clientes
IDep x y t d
...
IDep x y t d
ICli x y t p "LNC" IDep1 IDep2 ... IDepN "FinLNC"
...
ICli x y t p "LNC" IDep1 IDep2 ... IDepM "FinLNC"
```

donde:

IDep	es el identificador de deposito
ICli	es el identificador de cliente
x,y	coordenadas cartesianas
t	centro de ventana de tiempo
d	demanda
p	producción
LNC	bandera de comienzo de depósitos no compatibles
FinLNC	bandera de fin de depósitos no compatibles
Coeficiente XY	coef. de ponderación de la distancia Euclídea
Coeficiente TW	coef. de ponderación de la distancia temporal
#Depósitos	cantidad de depósitos
#Clientes	cantidad de clientes
NºClientes	cantidad de clientes asignados a un cluster

2) Método aplicable

Hay dos elecciones que el usuario debe realizar. Por un lado la ponderación de las componentes espacial y temporal de la distancia y por el otro el algoritmo a utilizar.

- Ponderación:

  El usuario podrá otorgar la ponderación deseada a las dos componentes de la distancia: la componente espacial y la temporal.

- Algoritmo:

  Los algoritmos posibles para realizar el Clustering de los datos de entrada son tres:

  - K-Nearest Neighbours (KNN)
    En este caso el usuario deberá además indicar el valor del parámetro *k*.
  - K-Means y
  - Partitioning Around Medoids (PAM)

3) Datos de Salida

La salida consiste en un archivo con formato ASCII donde se encuentran los siguientes datos:

- Ponderación: Componente espacial y temporal

- Depósitos
  - Cantidad de depósitos

- Clientes
  - Cantidad de clientes

- Lista de asignación de clientes a depósitos
- Lista de clientes no asignados y
- Tiempo de ejecución (en segundos)

El formato del archivo será el siguiente:

Formato de Archivos de Datos de Resultados:

Coeficiente XY
Coeficiente TW
#Depósitos
#Clientes
IDep 1 ClientesICli11 ... ICli1N
- -
IDep N Clientes ICliN1 ... ICliNN
NO ASIGNADOS Clientes ICli1 . . .ICliM
Tiempo Ejecución

2.1.3 Restricciones

En el marco de un MDVRPTW las restricciones al problema serán:

- Cada cluster contendrá un solo deposito.
- Cada cliente y cada deposito están asignados a un solo cluster.
- La suma de las producciones de los clientes asignados a un cluster no puede superar la demanda del deposito del mismo.
- Un cliente no podrá ser asignado a un cluster cuyo depósito esté en su lista de "no compatibles".
- Los identificadores de los Depósitos y de los Clientes deben ser todos distintos y enteros positivos.

2.1.4. Conclusiones

Luego de los relevamientos realizados con el usuario se llegó a la conclusión de que el prototipo a desarrollar debería:

✓ Implementar los siguientes algoritmos de Clustering:

- KNN (K-Nearest Neighbours),
- K-Mediodes (en particular PAM) y
- K-Means

aplicados a clusterizar un conjunto de Clientes y Depósitos respetando las restricciones impuestas.

✓ Brindar diferentes presentaciones de los resultados obtenidos (archivo de texto y representación gráfica) que permitan su estudio.

✓ Permitir la comparación de los resultados obtenidos a través de la distancia recorrida en cada clusterización al aplicar un VRP por Depósito para luego obtener la solución total a un MDVRPTW.

El propósito es generar un prototipo que implemente los algoritmos mencionados los cuales realizarán la clusterización de un conjunto de clientes y depósitos. Se presentarán los resultados en varios formatos (archivo de texto para su posterior utilización en la ejecución de los VRP y en formato gráfico para una rápida exploración de los mismos) que permitan la comparación de las soluciones obtenidas a partir de los diferentes algoritmos.

De esta forma resolvemos nuestro problema de asignación de clientes a depósitos formando subgrupos más pequeños. A partir de la solución por nosotros brindada, se podrá realizar un ruteo por cada cluster, logrando así la solución final al MDVRPTW.

2.2 Diseño

2.2.1. Definiciones

Condición de Compatibilidad:

Un elemento (Cliente o Deposito) cumple la Condición de Compatibilidad con un Cliente si el Cluster al cual esta asignado el elemento es Compatible con el Cliente (o sea no esta en la lista de incompatibles del Cliente).

Demanda Insatisfecha:

Definimos Demanda Insatisfecha para un Cluster, como la diferencia entre la Demanda de dicho Cluster con su Producción.

Demanda Insatisfecha = Demanda - Producción

Donde la demanda del Cluster es la del Depósito y su producción es la sumatoria de las producciones de los Clientes a él asignados.

Condición de Demanda:

Un elemento (Cliente o Deposito) cumple la Condición de Demanda con un Cliente si el Cluster al cual esta asignado el elemento posee Demanda Insatisfecha mayor que la Producción del Cliente.

Elemento Seleccionable:

Un elemento (Cliente o Deposito) es seleccionable para un Cliente si cumple la Condición de Compatibilidad y la Condición de Demanda con el mismo.

2.2.2 Genérico

El Software se divide en tres módulos lógicos que marcan las distintas etapas del programa:

- PRE-PROCESAMIENTO
- PROCESAMIENTO
- POST-PROCESAMIENTO

2.2.2.1 Pre-Procesamiento

Es en esta etapa donde se cargan los datos a partir del archivo de entrada y se interactúa con el usuario solicitando los diversos parámetros.

También se presenta en forma grafica la distribución de Depósitos y Clientes y su asignación inicial.

Pasos a seguir:

- ✓ Seleccionar el "Archivo de Datos"
- ✓ Cargar las estructuras de datos iniciales
- ✓ Dibujar los Depósitos y los Clientes
- ✓ Ingresar los Factores de Ponderación
- ✓ Seleccionar que algoritmo se desea ejecutar (si KNN resulta elegido, entonces solicitar K)

Como resultado de esta etapa se obtiene:

- ✓ Cantidad de Depósitos
- ✓ Cantidad de Clientes
- ✓ Lista de Depósitos
- ✓ Lista de Clientes
- ✓ Factor de Ponderación Wxy
- ✓ Factor de Ponderación Wt
- ✓ Despliegue gráfico de DEPOSITOS (en distintos colores) y CLIENTES (en negro)

2.2.2.2 Procesamiento

Se ejecuta el algoritmo de Clustering seleccionado obteniendo como resultado la formación de los Clusters, se calcula el tiempo empleado en la ejecución del algoritmo para su posterior comparación. Se despliega gráficamente cada (re) asignación de clientes. Esto permite analizar a simple vista los cambios realizados y además es una herramienta importante al momento del testeo.

Pasos a seguir:

- ✓ Iniciar (crear) un cluster a partir de cada deposito
- ✓ CLUSTERIZAR (KNN, PAM o K-Means)

Como resultado de esta etapa se obtiene:

- ✓ Clusters Formados
- ✓ Lista de Clientes No Asignados (en caso de existir)
- ✓ Tiempo de Ejecución

2.2.2.3 Post-Procesamiento

Se guardan los resultados obtenidos en un archivo y se despliegan gráficamente los Clusters (Depósitos y Clientes asociados del mismo color).

Pasos a seguir:

- ✓ Seleccionar el "Archivo de Resultados"
- ✓ Guardar en el "Archivo de Resultados" según formato definido
- ✓ Despliegue de los Clusters formados (los Clientes del mismo color que el Deposito del Cluster)

Como resultado de esta etapa se obtiene:

- ✓ Archivo de Resultados
- ✓ Despliegue Gráfico de los Clusters

2.2.2.4 Estructuras Base

Cluster:

Identificador:
Identifica que Deposito es el que esta asignado al Cluster, con esto cuando procesamos un Cliente al tratar la lista de incompatibles es indistinto si lo consideramos incompatible con el Deposito o con el Cluster. También es una manera de diferenciar un Cluster de los demás.

Clientes:
Se guarda una lista de Clientes asignados al Cluster.

Cantidad de clientes:
Se guarda la cantidad de Clientes asignados al Cluster (si bien es un dato redundante es de utilidad para muchas operaciones).

Demanda:
Se guarda la Demanda del Deposito que esta asignado al Cluster.

Producción:
Se guarda la sumatoria de la Producción de todos los Clientes asignados al Cluster. Se inicializa en CERO.

Deposito:

Identificador:
Permite la Identificación del Deposito en el conjunto, es Único.

Abscisa, Ordenada:
Guardan las coordenadas necesarias para ubicar gráficamente al Deposito.

Tiempo:
Guarda el Centro de una Ventana de Tiempo, en el cual el Deposito se encuentra abierto, ubica al Deposito temporalmente.

Demanda:
Guarda la Demanda del Deposito.

Color:
Guarda el color asignado al Deposito, sirve para la representación grafica de los resultados obtenidos.

Cliente:

Identificador:
Permite la Identificación del Cliente en el conjunto, es Único.

Abscisa, Ordenada:
Guardan las coordenadas necesarias para ubicar geográficamente al Cliente en el plano.

Tiempo:
Guarda el Centro de una Ventana de Tiempo, en el cual el Cliente se encuentra abierto, ubica al Cliente temporalmente.

Producción:
Guarda la Producción del Cliente.

Incompatibles:
Guarda una lista de Depósitos Incompatibles con el Cliente.

Cluster:

Cuando esta asignado a un Cluster tiene el Identificador del mismo sino esta asignado tiene un valor NO_VALIDO.

Color:

Inicialmente en negro (indica Cliente no asignado), luego guarda el color del Deposito del Cluster al que esta asignado, sirve para la representación grafica de los resultados obtenidos.

También se declaran Listas de Clientes, Depósitos y Clusters para manejar los conjuntos.

CLUSTER
{
identificador
clientes
cantidad de clientes
demanda
producción
}

DEPOSITO
{
identificador
abscisa
ordenada
tiempo
demanda
color
}

CLIENTE
{
identificador
abscisa
ordenada
tiempo
producción
incompatibles
cluster
color
)

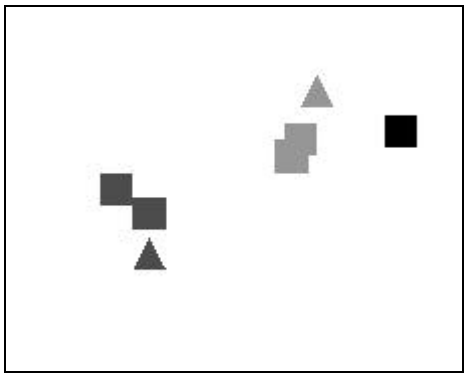
LISTAS de:
CLUSTERS
DEPOSITOS
CLIENTES

2.2.3 KNN

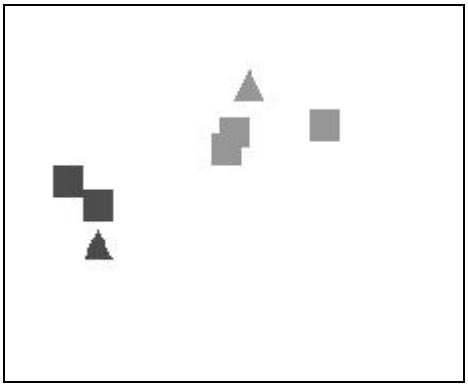
La idea básica de KNN es la siguiente:

Se considera un conjunto de datos clasificados y un nuevo elemento a clasificar. Para decidir a que cluster pertenece el nuevo elemento, el algoritmo se basa en los *k* puntos más cercanos. El cluster que tenga mayor confianza (mas elementos dentro de los *k* vecinos mas cercanos) es a quien se asigna el elemento a clasificar.

A manera de ejemplo, si se parte del conjunto de datos clasificados de la figura (un cluster en verde y el otro en rojo) y se pretende clasificar el elemento en negro, entonces se toman los *k* vecinos mas cercanos (supongamos *k*=2).



A simple vista se puede observar que los dos vecinos más cercanos pertenecen al cluster verde por lo tanto el nuevo elemento va a resultar asignado al mismo.



Seleccionar los "K Vecinos" Más Cercanos:

La selección de los *k* Vecinos más Cercanos puede "deformar" y seleccionar más (se eligieron k-i elementos sin ningún obstáculo y al intentar seleccionar el siguiente vecino más cercano se da el caso que se tienen mas de i objetos seleccionables con la misma distancia, superando así los *k*), menos de *k* (no llegan a *k* los seleccionables que cumplen las restricciones) e incluso hasta cero vecinos mas cercanos (como caso particular de menos de K).

2.2.3.1 Pseudocódigo

```
Crear un CLUSTER a partir de cada DEPOSITO

Procesar_CLIENTES
    Tomar siguiente CLIENTE
    Elegir los "K vecinos"
    Calcular la CONFIANZA de cada CLUSTER
    SI
            Existe CLUSTER de (Máximo Absoluto Ci)
    ENTONCES
            Asignar el CLIENTE a ese CLUSTER
            Marcarlo como PROCESADO
    SINO
            Crear Lista (empatados)
            SI
                    Lista (empatados) NO Es Vacía
            ENTONCES
                    Aplicar ARBITRAJES (empatados)
                    Asignar el CLIENTE a CLUSTER
                    Marcarlo como PROCESADO
            SINO
                    Asignar a Lista (sin asignar)
            FIN SI
    FIN SI
FIN Procesar CLIENTES
```

2.2.3.2 Funciones

Función de arbitraje (lista): identificador

Precondición: lista no puede ser vacía

Función de arbitraje- Recibe la lista de Clusters sobre los cuales arbitrar- en caso de que varios Clusters tengan la misma Confianza al elegir los *k* vecinos mas cercanos.

Esta función arbitrará entre todos los elementos de la lista devolviendo el de mayor confianza. Para ello contará con varios criterios de arbitraje en un orden dado y en caso de que ninguno de los criterios dados pueda definir cual es el elemento con mayor Confianza entonces se resolverá vía Sorteo.

Como criterios de Arbitrajes se proponen:

- CLUSTER más CERCANO (según Depósitos y/o Clientes)
- DEPOSITO más CERCANO
- CLIENTE más CERCANO
- CLUSTER con mas/menos CLIENTES
- CLUSTER con mayor (Demanda-Producción)
- SORTEO

2.2.3.3 Problema detectado en KNN

Si el *k* elegido es la cantidad de depósitos (o cualquier *k* muy grande) lo que va a ocurrir es que el algoritmo va a "agrupar" por orden de llegada los clientes en un mismo depósito hasta que este sature su demanda (a excepción de los clientes no compatibles) y luego seguirá con el siguiente deposito

Para el primer Cliente P1, los *k* vecinos serán todos los Depósitos (excepto los NO compatibles, y aquellos a los que NO les alcance la Demanda para cubrir la Producción, en cuyo caso tomamos menos de *k* vecinos) por lo tanto todos los Clusters tendrán a lo sumo un elemento entre los *k* vecinos de P1, se asigna entonces a C1 (a través de arbitraje, por ejemplo).

Al tomar el siguiente Cliente si C1 no es incompatible con el y entre sus *k* vecinos mas cercanos se encuentran D1 y P1 entonces se asignara a C1 y así sucesivamente hasta saturar la Demanda de C1. Luego ocurrirá lo mismo a partir del próximo Cliente (o de alguno asignado a otro Cluster C2 debido a incompatibilidad con C1) así hasta terminar de procesar los Clientes.

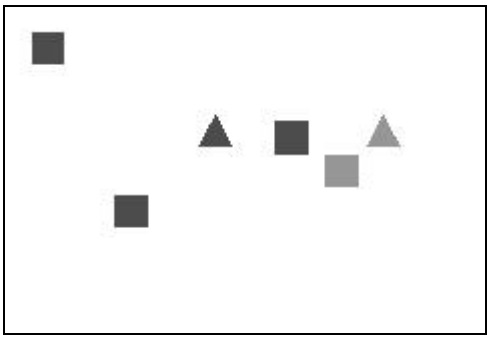
Por lo tanto debería tomarse un *k* menor a la cantidad de Depósitos.

2.2.4 PAM

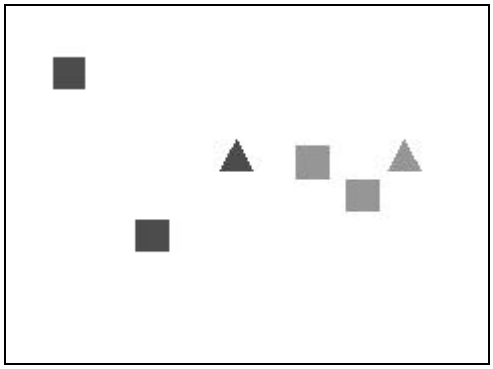
Este método tiene como objetivo encontrar objetos representativos llamados *mediodes*. De esta manera se particiona el conjunto de objetos en *k* clusters utilizando alguna medida de *similitud o distancia*. El próposito de trabajar con estos mediodes es obtener la menor distancia promedio posible entre los elementos del conjunto. La idea es que cada mediode sea el elemento ubicado mas centralmente en cada cluster y que el resto de los elementos pertenezcan al cluster cuyo mediode sea el más cercano.

Se seleccionan arbitrariamente *k* objetos como mediodes. Luego de ello, el algoritmo trata de obtener una mejor selección de mediodes analizando todos los posibles pares de objetos, tal que uno sea mediode y el otro no. Entonces en cada iteración reemplaza un mediode por otro objeto que no lo es, y se calcula la *calidad del Clustering* (es la calidad combinada de los mediodes elegidos) obtenido. Si ésta es mejor que la obtenida hasta el momento se mantiene el intercambio de objetos para la próxima iteración, en caso contrario se vuelve a los mediodes anteriores. La calidad se mide como el promedio de la distancia entre un objeto y el mediode de su cluster.

La siguiente figura muestra un paso intermedio en la ejecución de PAM. Se puede distinguir un único elemento en el cluster verde.



Dentro del cluster verde se intenta intercambiar el mediode (en este caso el triángulo), con el no mediode. Al hacer los cálculos PAM determina que este cambio es conveniente y que además se debe reasignar un elemento del cluster rojo al verde.



2.2.4.1 Pseudocódigo

```
1
    Crear un CLUSTER a partir de cada DEPOSITO
    Definir los Mediodes como los Depósitos
    Procesar los CLIENTES asignándolos al CLUSTER "más cercano"
    Definir los NO Mediodes como los Clientes

2
    Para cada Mediode
        Para cada No Mediode
            Si
                Debo_Cambiar "Mediode-No Mediode"
            Entonces
                Cambiar "Mediode-No Mediode"
                Reiniciar ambos bucles
            Fin Si
        Fin Para
    Fin Para
```

El vector Mediodes tiene *k* elementos, siendo *k* el número de Depósitos y el vector No-Mediodes tiene n-k elementos, siendo n-k el número de Clientes.

Inicialmente se consideran Mediodes los depósitos, pero luego se puede convertir en un Mediode tanto un cliente como un depósito respetando las restricciones planteadas en Análisis de Requerimientos. Análogamente, los No Mediodes iniciales son los Clientes.

Al cambiar un Mediode por un No Mediode se reinicializan ambos loops, el algoritmo se sigue ejecutando hasta que no existan mas cambios posibles.

El hecho de que no se trabaje con la media, sino con un elemento del dominio que se le aproxima (por eso llamado mediode) permite su identificación gráfica si el cluster no posee demasiados elementos.

2.2.4.2 Funciones

Int Debo_Cambiar(mediode, no mediode)

Devuelve 1 (True) en caso que sea válido en términos de costos cambiar mediode y no mediode de clusters y 0 (False) en caso contrario.

Para ello, se debe calcular el *efecto del intercambio* entre O_i *(mediode) y* O_h *(no mediode)*, se calcula el *costo* C_{jih} para todo objeto, O_j, que no es mediode. El *costo total del intercambio (*TC_{ih}*)* es:

$$TC_{ih} = \sum_j C_{jih}$$

Para calcular C_{jih} *(es la resta entre la distancia de* O_j *a su nuevo mediode y la distancia entre* O_j *y su antiguo mediode*) se pueden dar algunos de los siguientes 4 casos:

Caso 1:

Al intentar cambiar el Mediode (O_i) por el No Mediode (O_h)
Se toma en consideración un objeto (O_j) que pertenece al Cluster del Mediode (O_i)
Si el objeto (O_j) en cuestión esta mas cerca de otro Mediode O_{j2} que del No Mediode (O_h), entonces el objeto (O_j) cambia al cluster del otro objeto (O_{j2})
Quedando (costo para j en el intercambio de (i, h) como la distancia entre (j, j2) menos la distancia entre (j, i)

$$C_{jih} = d(O_j, O_{j2}) - d(O_j, O_i)$$

Caso 2:

Al intentar cambiar el Mediode (O_i) por el No Mediode (O_h)
Se toma en consideración un objeto (O_j) que pertenece al Cluster del Mediode (O_i)
Si el objeto (O_j) en cuestión esta mas cerca del No Mediode (O_h) que de otro Mediode O_{j2}, entonces el objeto (O_j) pasa al Cluster del No Mediode (O_h)
Quedando (costo para j en el intercambio de (i, h) como la distancia entre (j, h) menos la distancia entre (j, i)

$$C_{jih} = d(O_j, O_h) - d(O_j, O_i)$$

Caso 3:

Al intentar cambiar el Mediode (O_i) por el No Mediode (O_h)
Se toma en consideración un objeto (O_j) que NO pertenece al Cluster del Mediode (O_i)
Si el objeto (O_j) en cuestión esta mas cerca de SU ACTUAL MEDIODE que del No Mediode (O_h)
Entonces el objeto (O_j) se queda en el mismo Cluster
Quedando:

$$C_{jih} = 0$$

Caso 4:

Al intentar cambiar el Mediode (O_i) por el No Mediode (O_h)
Se toma en consideración un objeto (O_j) que NO pertenece al Cluster del Mediode (O_i)
Si el objeto (O_j) en cuestión esta mas cerca del No Mediode (O_h) que de SU ACTUAL MEDIODE (O_{j2})
Entonces el objeto (O_j) pasa al Cluster del No Mediode (O_h)
Quedando (costo para j en el intercambio de (i, h) como la distancia entre (j, h) menos la distancia entre (j, j2)

$$C_{ijh} = d(O_j, O_h) - d(O_j, O_{j2})$$

Si el costo total, TC_{ih}, es negativo entonces se reemplaza efectivamente a O_i por O_h. O sea, O_h pasa a ser el nuevo mediode pues ese Clustering tiene mejor calidad y se sigue iterando con el resto de los pares de los objetos.

2.2.5 K-Means

Media de un Cluster:

Se define la Media de un Cluster como:

Xm = Suma(x)/n
Ym = Suma(y)/n
Tm = Suma(t)/n

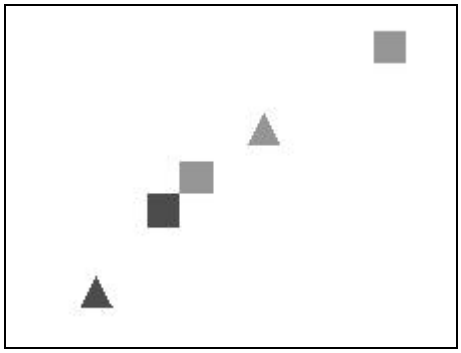
Disimilitud:

Se define como la "Distancia" de antes, distancia Euclídea al cuadrado ponderada mas la distancia "temporal" también ponderada.

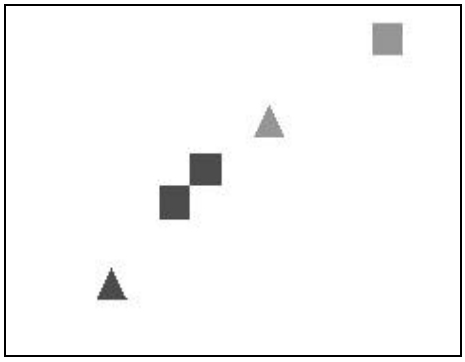
El algoritmo k-means está construido sobre cuatro operaciones básicas:

- ✓ Selección de *k* medias iniciales para los *k* clusters.
- ✓ Cálculo de la disimilitud entre un objeto y la media del cluster.
- ✓ Asignación de un objeto a un cluster cuya media es la más cercana al objeto.
- ✓ Recálculo de la media de un cluster de los objetos alocados a él como para que la disimilitud intra-cluster sea minimizada.

En la siguiente figura se observa un paso intermedio en el algoritmo de k-means y las asignaciones existentes en ese momento.



Al recalcular la media para el elemento central de la figura anterior (que pertenece al cluster verde), resulta que la media más cercana es la del cluster rojo por lo tanto se reasigna a él.



2.2.5.1 Pseudocódigo

```
1
    Crear un CLUSTER  a partir de cada DEPOSITO
    Definir las Medias Iniciales a partir de los Depósitos
    Asignación Inicial de los Clientes a los Clusters

2
    Mientras
            Cambian las medias
    Loop
            Recorrer CLIENTES
                    Calcular la Distancia a las Medias de los Clusters
                    Si
                            (Distancia al Cluster Media Mas Cercana
                            <
                            Distancia al Cluster Actual)
                    Entonces
                            Reasignar CLIENTE
                            Recalcular Medias de Ambos Clusters
                    Fin Si
            Fin Recorrer
    Fin Mientras
```

En este algoritmo se puede ver que los centros de los clusters (medias) no son necesariamente elementos y puede que no tengan sentido en el dominio de aplicación, lo que le brinda un grado extra de libertad.

2.2.6 Limitaciones

Los algoritmos de Clusterización vistos están planteados en un marco teórico. Este trabajo apunta a la resolución de un problema práctico, por lo tanto no se van a implementar los algoritmos puros, sino que en ellos se van a ver reflejadas las distintas restricciones.

La primer condicionante es que tiene que haber un depósito por cluster lo que define dos categorías de puntos a clasificar: los clientes y los depósitos. Por lo tanto el punto de partida de cada algoritmo es determinar como inicio de cada cluster a cada depósito.

Otra condicionante es cumplir que la suma de la producción de los clientes de un cluster no exceda la demanda del depósito del mismo.

Finalmente, existe una lista de incompatibilidades depósito-cliente a respetar.

Entonces si se intenta asignar algún cliente a un cluster por mas que esta asignación sea la optima, si se viola una de las dos ultimas restricciones mencionadas, entonces no es posible permitir la asignación.

Por lo tanto, es posible que se presente el caso de que existan clientes sin asignar al finalizar el algoritmo de clusterización.

2.3 Implementación

2.3.1 Elección del Lenguaje

Los algoritmos se desarrollaron en C++ debido a que este es un lenguaje de utilidad general que proporciona mecanismos flexibles y eficientes para la definición de nuevos tipos de datos. En otras palabras, permite la abstracción de datos. Se pueden definir clases, administrar dinámicamente la memoria, etc. lo que permite un alto grado de eficiencia. Además posee un amplio grupo de bibliotecas estándar que permite su transporte. Por otra parte, cuenta con una biblioteca llamada MFC para programación en Windows que permite una correcta interacción con el usuario además de una buena interfase gráfica para representar los resultados obtenidos.

Las características anteriores (abstracción de datos, disponibilidad de bibliotecas estándar para Windows e interfaces gráficas) se consideraron indispensables a la hora de elegir el lenguaje a utilizar, porque el objetivo era concentrar el esfuerzo de desarrollo en los algoritmos (y no en detalles visuales, de presentación o de programación Windows).

Lenguajes "clásicos", como Fortran, Cobol o incluso Pascal tienen defectos graves en uno o varios de esos puntos. Además, ninguno de ellos compite en eficiencia con C++.

Una alternativa posible a C++, hubiera sido algún lenguaje que cumpliera con todas las características anteriores, dentro de los mas difundidos comercialmente junto a Visual C++ -a los que se tenían acceso- son Visual Basic y JAVA.

Se descartaron Visual Basic y JAVA por diversos motivos. Visual Basic no es tan eficiente y es medianamente portable en Windows y no portable a otros SO. Está falsamente orientado a objetos y no hace un buen manejo de punteros. Por otro lado, JAVA es portable pero tiene variantes en el tema de orientación a objetos.
Ver [VIII] [IX] [X] [XIII]

2.3.2 Ambientes de Desarrollo y Testeo

El sistema se implemento en Visual C++ 6.0 de Microsoft bajo ambiente Windows (en particular Windows 95) por lo tanto es portable a los ambientes Windows superiores (Windows NT 4.0, Windows 2000, Windows 98, Windows Millenium).

El sistema se desarrollo y testeo en 3 maquinas. Sus configuraciones son:

- Pentium II, 233 MHz, 32 Mb RAM, 4.0 GB S.O. Windows 95
- Pentium MMX, 200 MHz, 64 Mb RAM, 3.0 GB y S.O. Windows 95
- Pentium II, 200 MHz, 64 Mb RAM, 3.0 GB y S.O. Windows 98

2.4 Testing

2.4.1 Políticas de Testeo

Se desea probar la correctitud de la implementación de los algoritmos diseñados y detectar en la etapa de testeo los errores que los algoritmos presenten para su corrección. Para eso se utilizarán los casos de testeo adecuados a cada instancia y se trabajará en dos etapas:

Primero con una política de testeo de caja blanca, para la depuración primaria de errores y después con una política de caja negra donde por un lado se definen conjuntos de prueba con su comportamiento esperado y luego se coteja contra el resultado obtenido. De no coincidir ambos resultados se corregirá el código y se volverá a ejecutar el caso de prueba.

Se realizará también un testeo exhaustivo que incluya entradas de gran tamaño, utilizándose los mismos conjuntos de entradas con los tres algoritmos para poder comparar los resultados obtenidos.
Ver [V]

2.4.2 Casos de Prueba

Se presenta a continuación un conjunto representativo del total de casos de prueba manejados para cada algoritmo y las funcionalidades testeadas.

- KNN:
  - Comportamiento del algoritmo en general
  - Sensibilidad respecto al orden de entrada de los datos
  - Variación de los resultados obtenidos al cambiar k
  - Saturación de la demanda
  - Manejo de las incompatibilidades
  - No poner depósitos en el conjunto de datos

- PAM:
  - Sensibilidad respecto al orden de entrada
  - Manejo de incompatibilidades
  - Comparar el resultado con el obtenido corriendo KNN

- K-MEANS:
  - Comportamiento del algoritmo en general
  - Saturación de la demanda
  - Manejo de las incompatibilidades
  - Dependencia del orden de entrada
  - Comparar el resultado con el obtenido corriendo KNN y PAM
  - No poner depósitos en el conjunto de datos

2.4.2.1 KNN

Caso De Testeo: 1

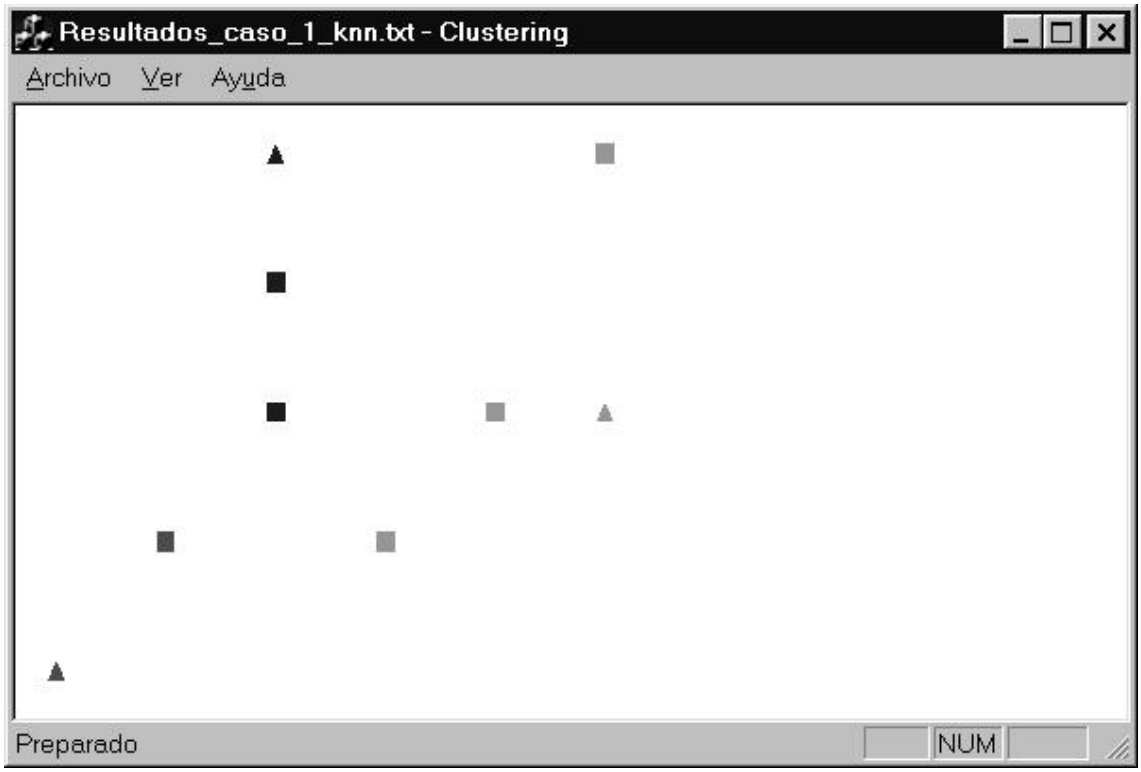
Funcionalidad Testeada: Comportamiento del algoritmo en general
Entrada:

```
3
6
1 0 0 1 1500
2 5 2 1 1500
3 2 4 1 1500
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada:

```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 1 Demanda 1500        producción 100
    Cliente 10
CLUSTER 2
    Deposito 2 #Clientes 3 Demanda 1500 producción 300
    Cliente 20              Cliente 40              Cliente 60
CLUSTER 3
    Deposito 3 #Clientes 2 Demanda 1500 producción 200
    Cliente 30              Cliente 50
CLIENTES NO ASIGNADOS: 0
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada (para k=1).



Caso De Testeo: 2

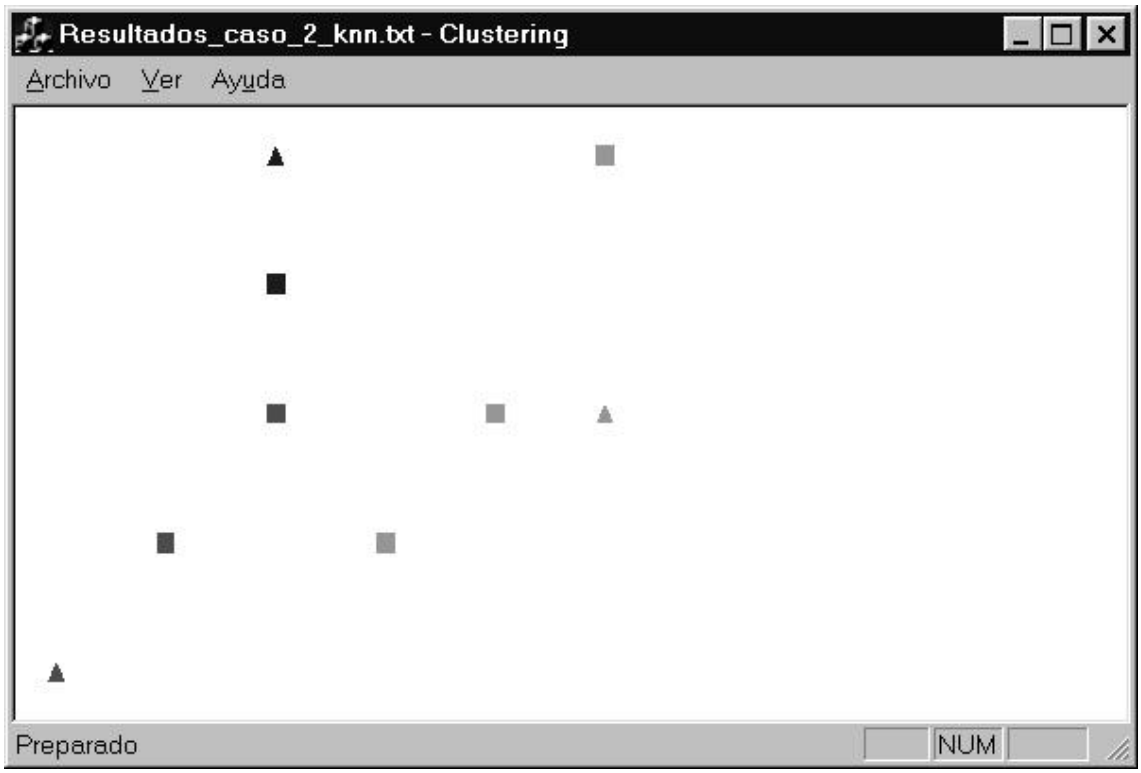
Funcionalidad Testeada: Sensibilidad respecto al orden de entrada de los datos
Entrada:

```
3
6
1 0 0 1 1500
2 5 2 1 1500
3 2 4 1 1500
10 1 1 1 100 LNC FinLNC
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC FinLNC
40 5 4 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada: Observar que el archivo de entrada es similar al del Caso 1 a menos del orden de un cliente. Esto debería ser suficiente para cambiar la conformación del Clustering resultante:

```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 6 Demanda 1500 producción 600
    Cliente 10          Cliente 30
CLUSTER 2
    Deposito 2 #Clientes 0 Demanda 1500 producción 0
    Cliente 20          Cliente 40          Cliente 60
CLUSTER 3
    Deposito 3 #Clientes 0 Demanda 1500 producción 0
    Cliente 50
CLIENTES NO ASIGNADOS: 0
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada (para k=1).



Caso De Testeo: 3

Funcionalidad Testeada: Variación de los resultados obtenidos al cambiar k.
Entrada: Idem Caso de Testeo 2
Salida Esperada: El archivo de entrada es el mismo que en el Caso 1. Esta vez, la variación es la invocación del algoritmo. Se trata de comparar los resultados con los distintos valores de *k* (1, 2 y 3). Estas corridas introducen resultados nuevos. Se muestra el comportamiento esperado para k=1 y k=2 y k=3.

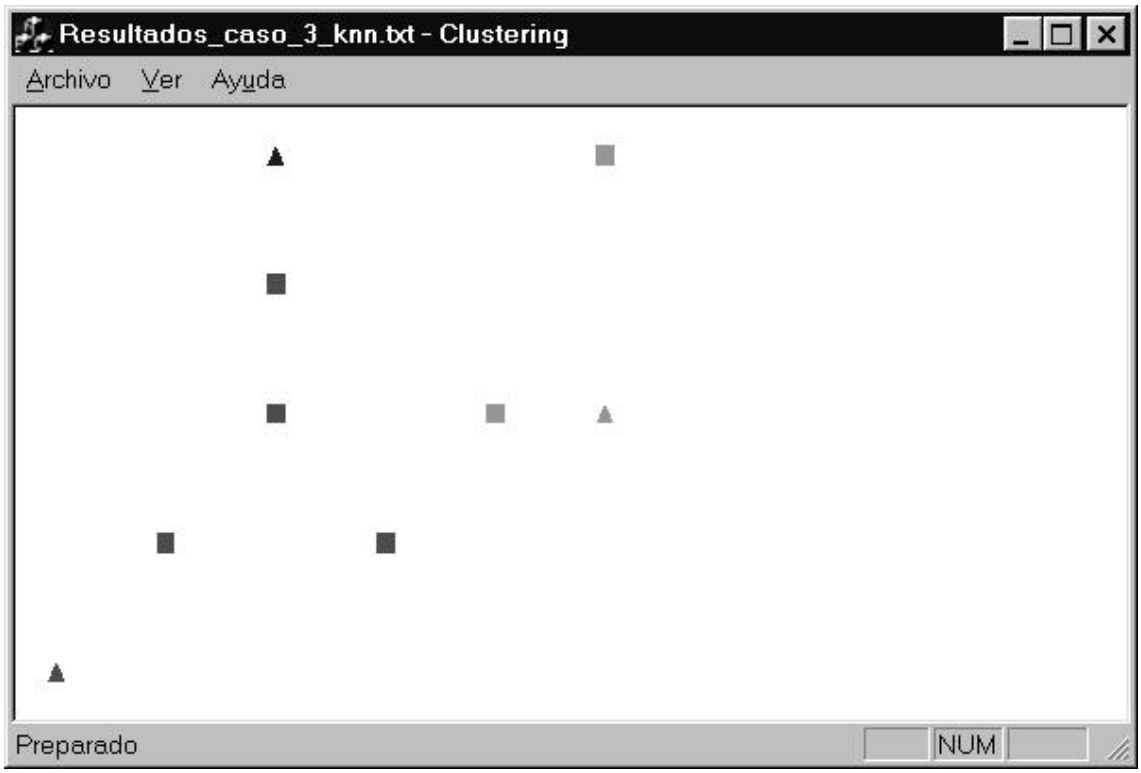
k=1

Ver Caso 1

k=2

3 CLUSTERS
CLUSTER 1
Deposito 1 #Clientes 4 Demanda 1500 producción 400
Cliente 30 Cliente 10 Cliente 50 Cliente 60
CLUSTER 2
Deposito 2 #Clientes 2 Demanda 1500 Producción 200
Cliente 20 Cliente 40
CLUSTER 3
Deposito 3 #Clientes 0 Demanda 1500 producción 0
CLIENTES NO ASIGNADOS: 0

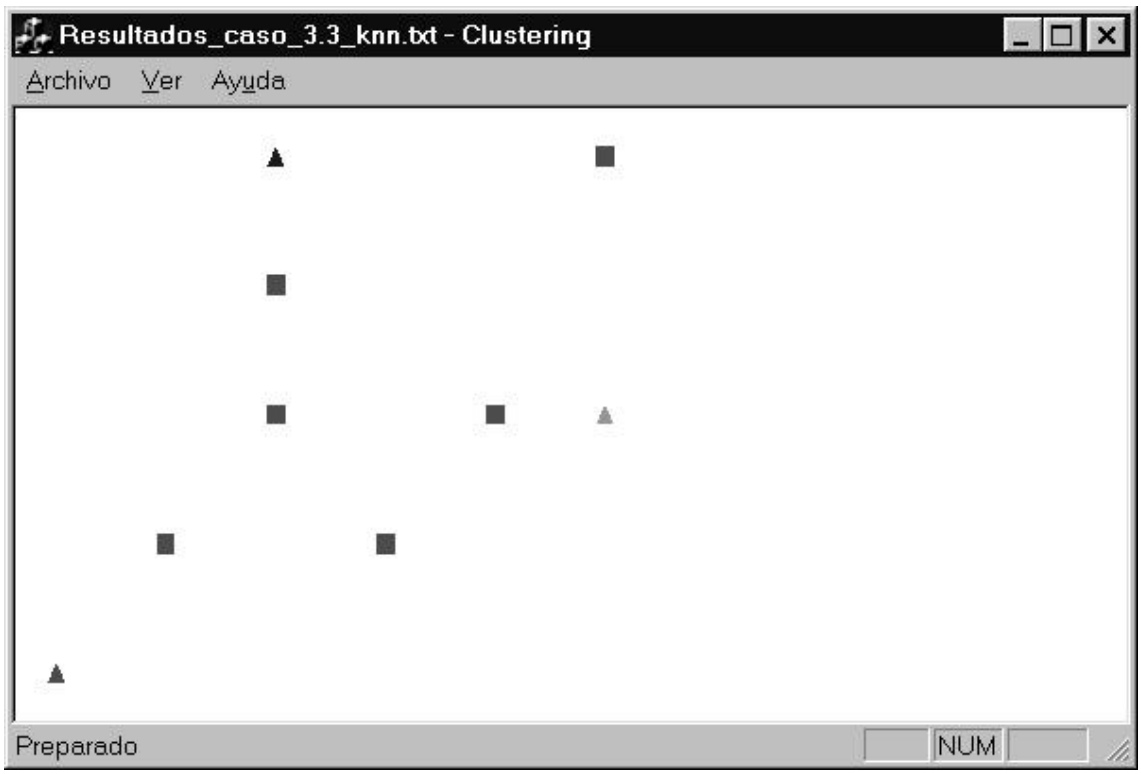
Resultado Obtenido: Se comporta de acuerdo a la salida esperada.



k=3

```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 4 Demanda 1500 Producción 400
    Cliente 20          Cliente 30 Cliente 10 Cliente 40 Cliente 50 Cliente 60
CLUSTER 2
    Deposito 2 #Clientes 2 Demanda 1500 Producción 200
CLUSTER 3
    Deposito 3 #Clientes 0 Demanda 1500 Producción 0
CLIENTES NO ASIGNADOS: 0
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada.



Caso De Testeo 4:

Funcionalidad Testeada: Saturación de la Demanda.
Entrada: Dos archivos, el primero donde se satura solamente la demanda de algunos de los depósitos y el segundo donde se saturan todos los depósitos quedando clientes sin asignar.

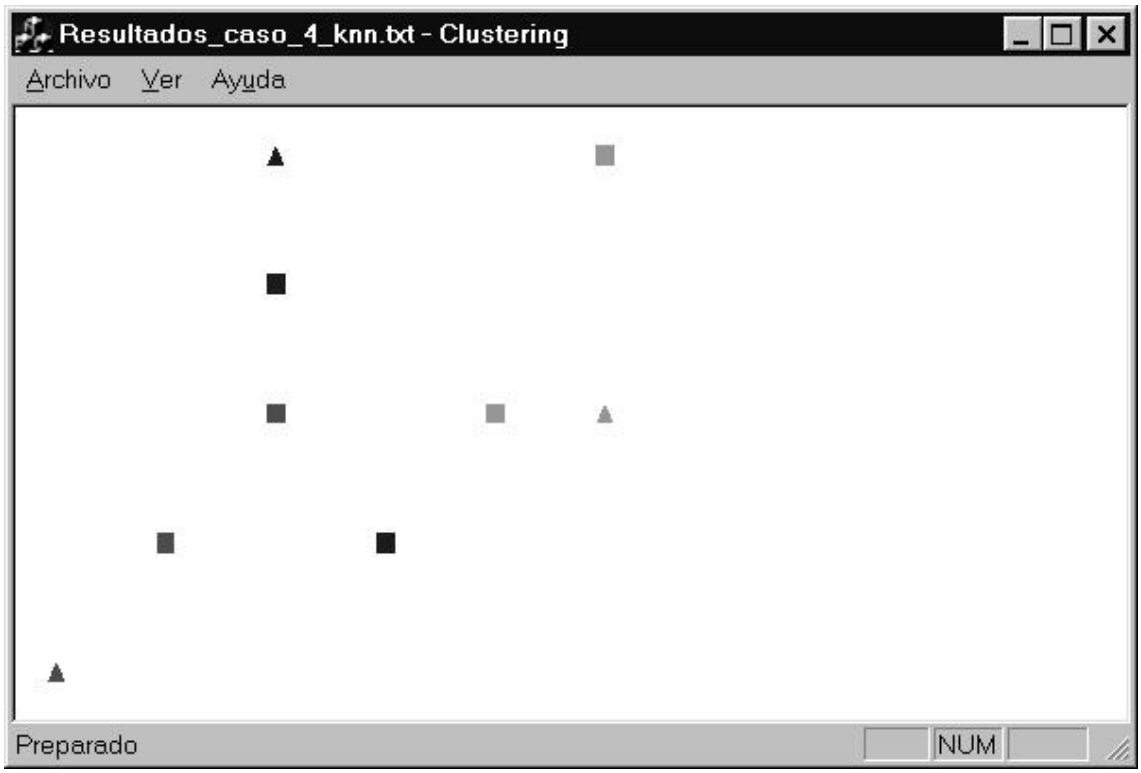
Archivo 1
3
6
1 0 0 1 200
2 5 2 1 200
3 2 4 1 1500
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC

Archivo 2
3
6
1 0 0 1 100
2 5 2 1 200
3 2 4 1 100
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC

Salida Esperada:
El comportamiento se diferenciará entre ambos archivos dado que mientras que la producción del primero no satura la demanda total, la producción del segundo si lo logra quedando entonces clientes sin asignar.

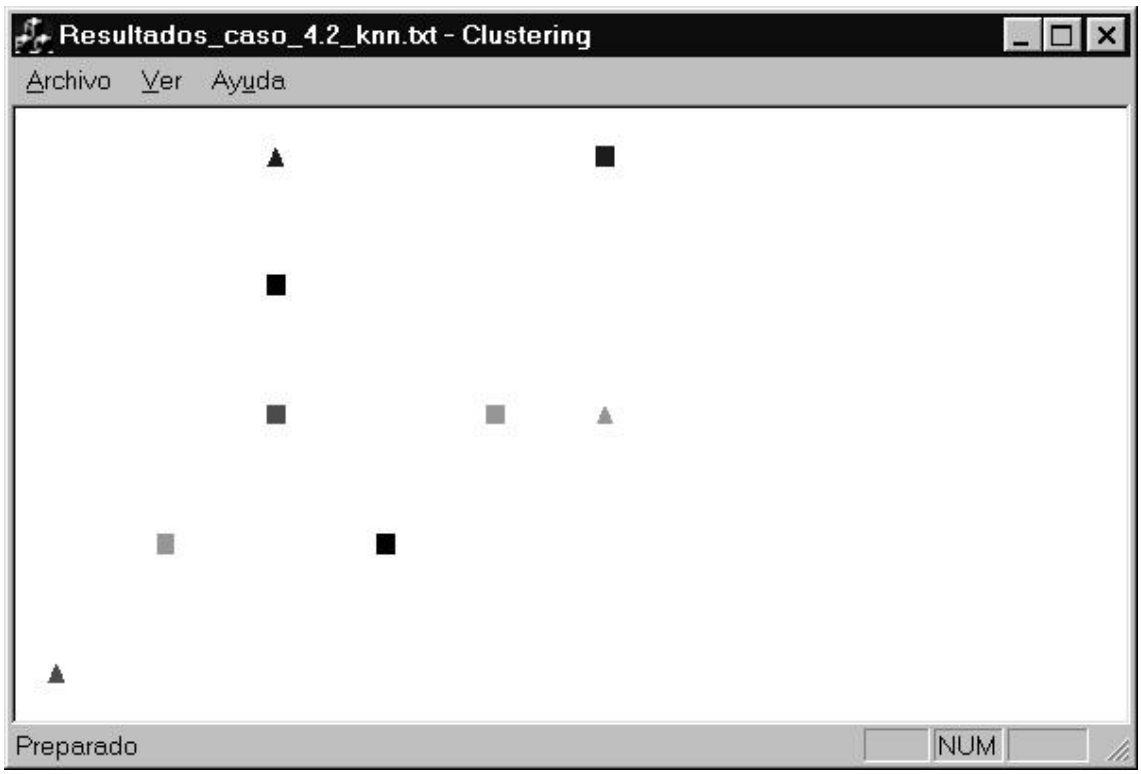
Salida 1 (k=2):

```
3 CLUSTERS
CLUSTER 1
      Deposito 1 #Clientes 2 Demanda 200 Producción 200
      Cliente 30                    Cliente 10
CLUSTER 2
      Deposito 2 #Clientes 2 Demanda 200 Producción 200
      Cliente 20                    Cliente 40
CLUSTER 3
      Deposito 3 #Clientes 2 Demanda 1500 Producción 200
      Cliente 50                    Cliente 60
CLIENTES NO ASIGNADOS: 0
```



Salida 2 (k=2):

```
3 CLUSTERS
CLUSTER 1
      Deposito 1 #Clientes 1 Demanda 100 Producción 100
      Cliente 30
CLUSTER 2
      Deposito 2 #Clientes 2 Demanda 200 Producción 200
      Cliente 20                    Cliente 10
CLUSTER 3
      Deposito 3 #Clientes 1 Demanda 100 Producción 100
      Cliente 40
CLIENTES NO ASIGNADOS: 2
      NO Asignado: Cliente 50
      NO Asignado: Cliente 60
```



Resultado Obtenido: Se comporta de acuerdo a la salida esperada

Caso De Testeo 5:

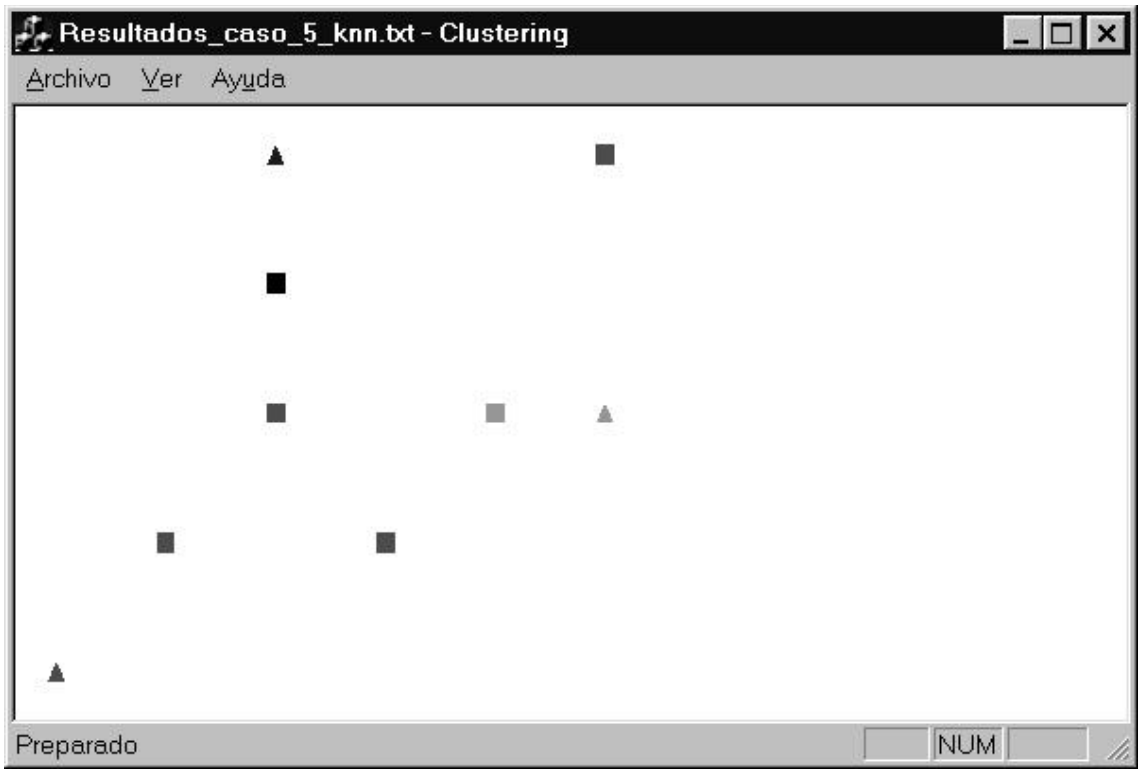
Funcionalidad Testeada: Manejo de las incompatibilidades
Entrada:

```
3
6
1 0 0 1 600
2 5 2 1 600
3 2 4 1 600
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC 3 FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC 2 FinLNC
50 2 3 1 100 LNC 1 2 3 FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada:

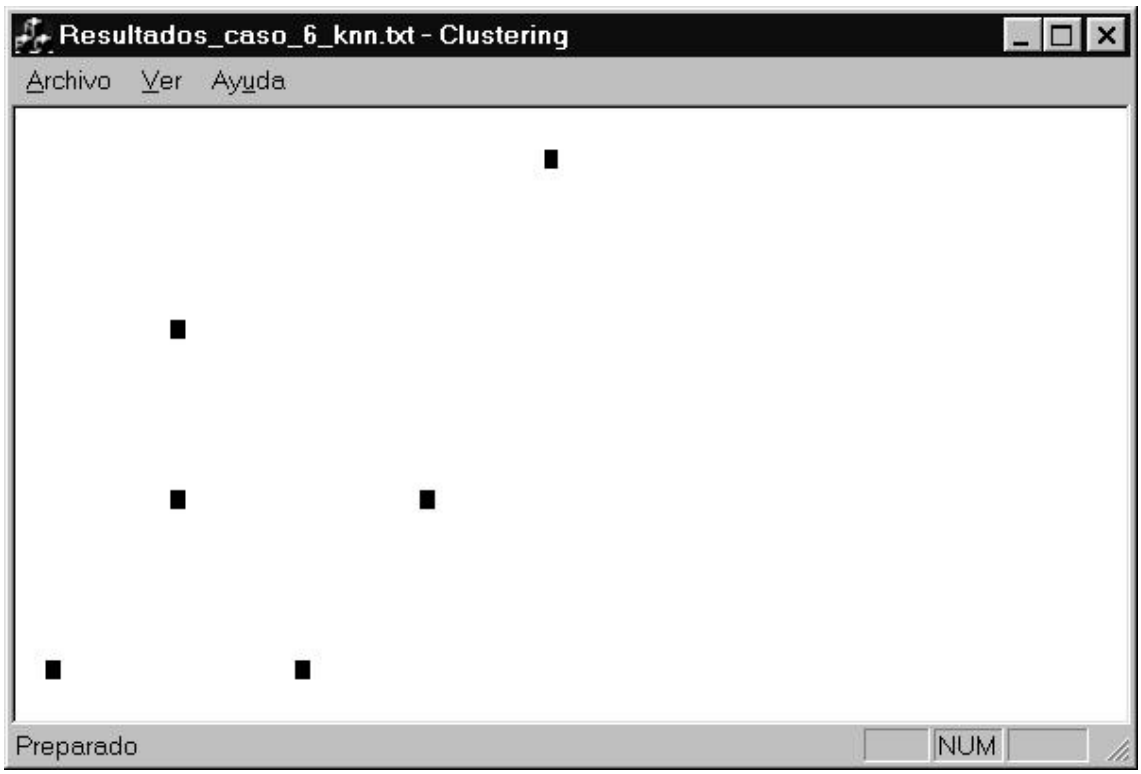
```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 5 Demanda 600 Producción 500
    Cliente 30          Cliente 10          Cliente 40          Cliente 60
CLUSTER 2
    Deposito 2 #Clientes 0 Demanda 600 Producción 0
    Cliente 20
CLUSTER 3
    Deposito 3 #Clientes 0 Demanda 600 Producción 0

CLIENTES NO ASIGNADOS: 1
    NO Asignado: Cliente 50
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada



Caso De Testeo 6:

Funcionalidad Testeada: No poner depósitos en el conjunto de datos
Entrada:
0
6
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC 3 FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC 2 FinLNC
50 2 3 1 100 LNC 1 2 3 FinLNC
60 3 1 1 100 LNC FinLNC
Salida Esperada: 0 clusters creados y reconocimiento de los 6 clientes
Resultado Obtenido: Se comporta de acuerdo a la salida esperada


Resultados_caso_6_knn.txt - Clustering
Archivo Ver Ayuda
Preparado
NUM

2.4.2.2 PAM

Caso De Testeo 1:

Funcionalidad Testeada: Sensibilidad respecto al orden de entrada
Entrada 1:

```
2
5
1  150 150 720 1400
2  225 250 660 1400
10 170 190 780 100 LNC FinLNC
20 250 260 640 100 LNC FinLNC
30 160 175 780 150 LNC FinLNC
40 198 207 575 175 LNC FinLNC
50 185 210 420 200 LNC FinLNC
```

Entrada 2:

```
2
5
1  150 150 720 1400
2  225 250 660 1400
20 250 260 640 100 LNC FinLNC
10 170 190 780 100 LNC FinLNC
50 185 210 420 200 LNC FinLNC
40 198 207 575 175 LNC FinLNC
30 160 175 780 150 LNC FinLNC
```

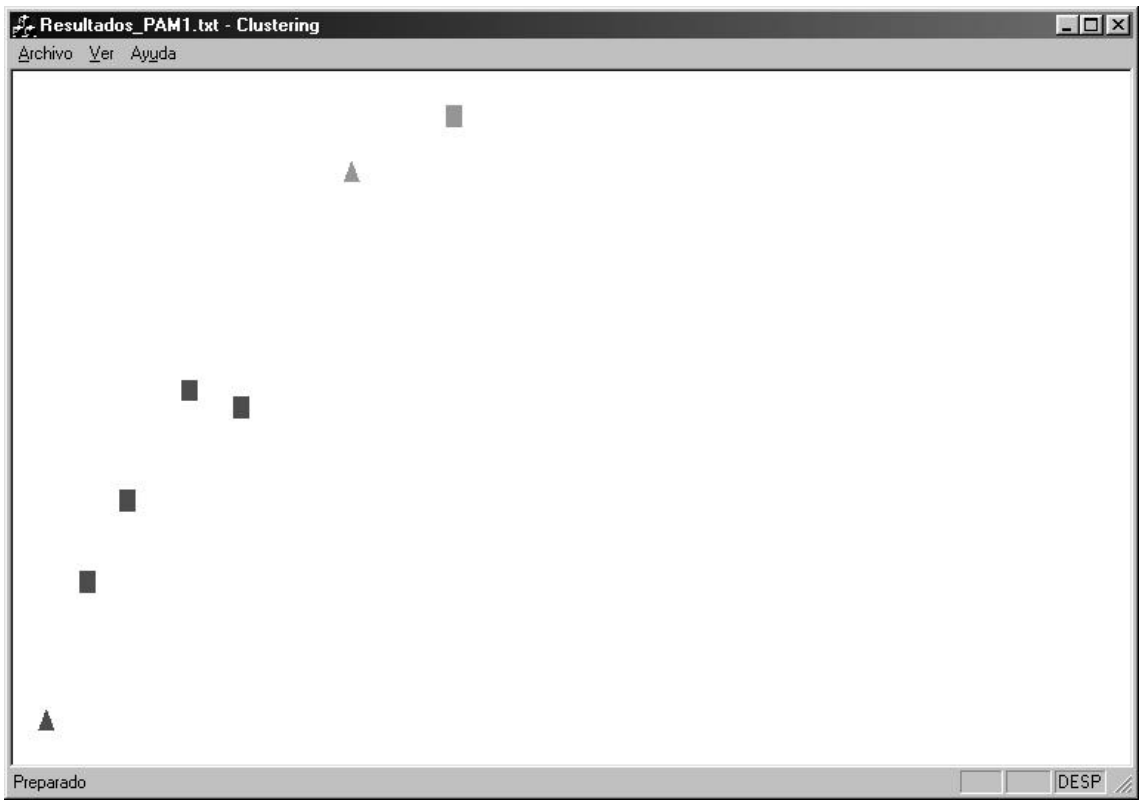
Entrada 3:

```
2
5
1  150 150 720 1400
2  225 250 660 1400
50 185 210 420 200 LNC FinLNC
30 160 175 780 150 LNC FinLNC
20 250 260 640 100 LNC FinLNC
40 198 207 575 175 LNC FinLNC
10 170 190 780 100 LNC FinLNC
```

Salida Esperada: A pesar de estar las entradas en distinto orden, el comportamiento final del algoritmo es el mismo. Por lo tanto la solución es única y es:

```
2 Clusters
CLUSTER 1
    Deposito 1 #Clientes 4 Demanda 1400 Producción 625
    Cliente 10          Cliente 30          Cliente 40          Cliente  50
CLUSTER 2
    Deposito 2 #Clientes 1 Demanda 1400 Producción 100
    Cliente 20
CLIENTES NO ASIGNADOS: 0
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada.



Caso De Testeo 2:

Funcionalidad Testeada: Manejo de Incompatibilidades Cliente-Depósito
Entrada:

```
2
5
1  49 52 720 1250
2  120 140 660 1320
10 35 40 780 100 LNC 1 FinLNC
20 100 120 640 100 LNC FinLNC
30 125 150 630 150 LNC 2 FinLNC
40 25 -15 575 175 LNC FinLNC
50 185 -210 420 200 LNC 1 2 FinLNC
```

Salida Esperada:

```
2 Clusters
CLUSTER 1
      Deposito 1 #Clientes 2 Demanda 1250 Producción 325
      Cliente 30                    Cliente 40
CLUSTER 2
      Deposito 2 #Clientes 1 Demanda 1320 Producción 200
      Cliente 10          Cliente 20
CLIENTES NO ASIGNADOS: 1
      NO Asignado: Cliente 50
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada.

Caso De Testeo 3:

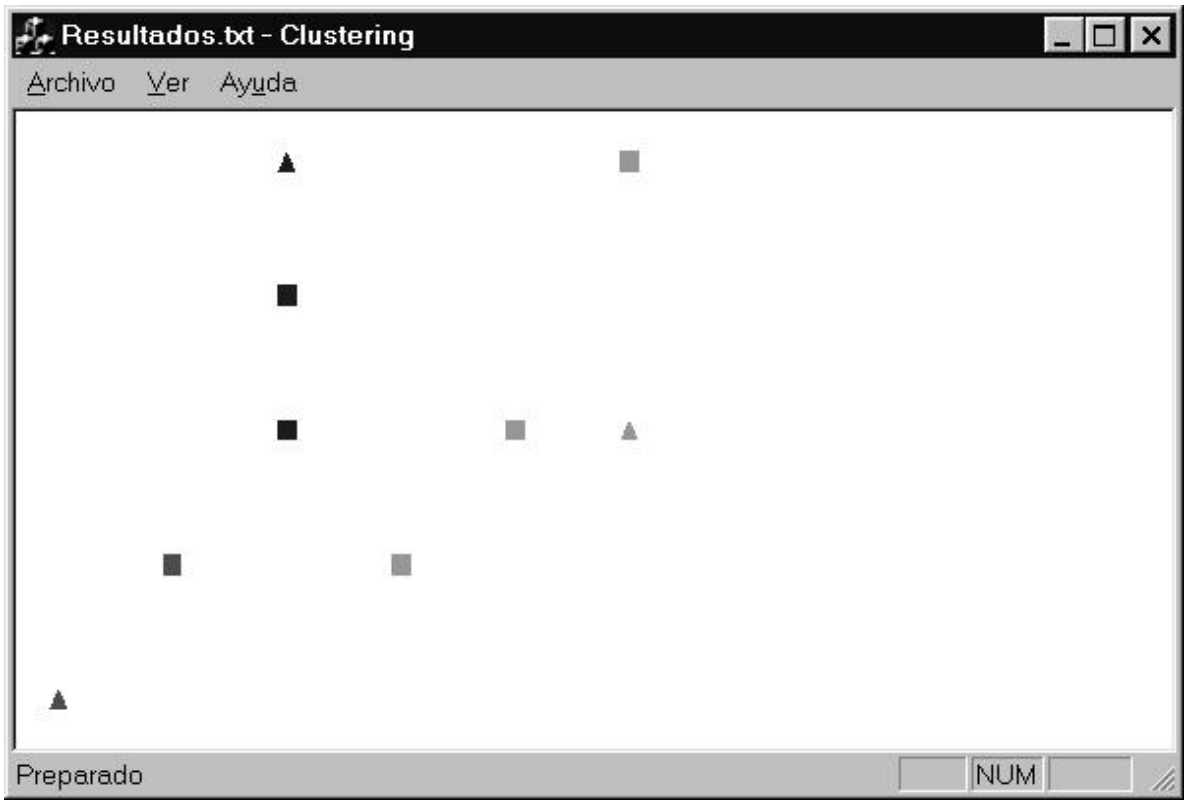
Funcionalidad Testeada: Comparar el resultado con el obtenido corriendo KNN para el *caso 1*.

Entrada:

```
3
6
1 0 0 1 1500
2 5 2 1 1500
3 2 4 1 1500
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada:

```
 3 CLUSTERS
CLUSTER 1
      Deposito 1 #Clientes 1 Demanda 1500 Producción 100
      Cliente 10
CLUSTER 2
      Deposito 2 #Clientes 3 Demanda 1500 Producción 300
      Cliente 20                Cliente 40                Cliente 60
 CLUSTER 3
      Deposito 3 #Clientes 2 Demanda 1500 Producción 200
      Cliente 30                Cliente 50
CLIENTES NO ASIGNADOS: 0
```


Resultados.txt - Clustering
Archivo Ver Ayuda
Preparado
NUM

2.4.2.3 K-MEANS

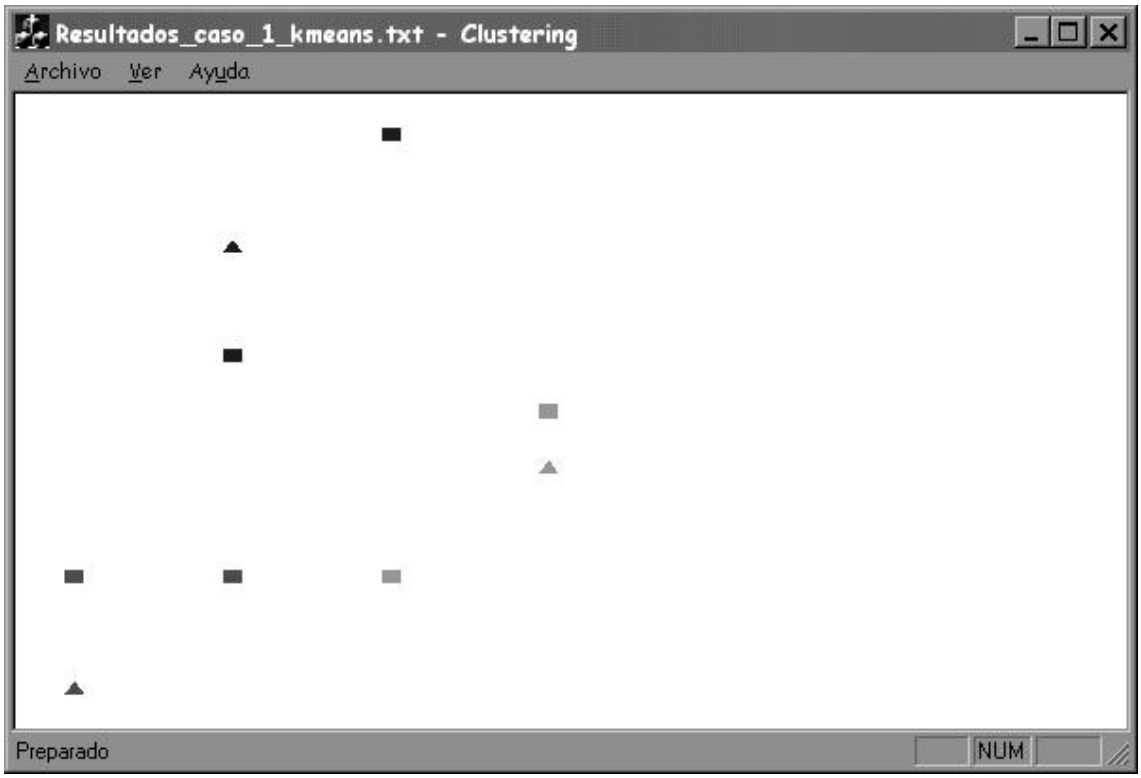
Caso De Testeo 1:

Funcionalidad Testeada: Comportamiento del algoritmo en general

Entrada:
```
3
6
1 1 0 1 500
2 4 2 1 500
3 2 4 1 500
20 4 2.5 1 100 LNC FinLNC
30 2 1 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 3 5 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada:
```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 2 Demanda 500 Producción 200
    Cliente 30          Cliente 10
CLUSTER 2
    Deposito 2 #Clientes 2 Demanda 500 Producción 200
    Cliente 20          Cliente 60
CLUSTER 3
    Deposito 3 #Clientes 2 Demanda 500 Producción 200
    Cliente 40          Cliente 50
CLIENTES NO ASIGNADOS: 0
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada



Casos De Testeo 2 y 3:

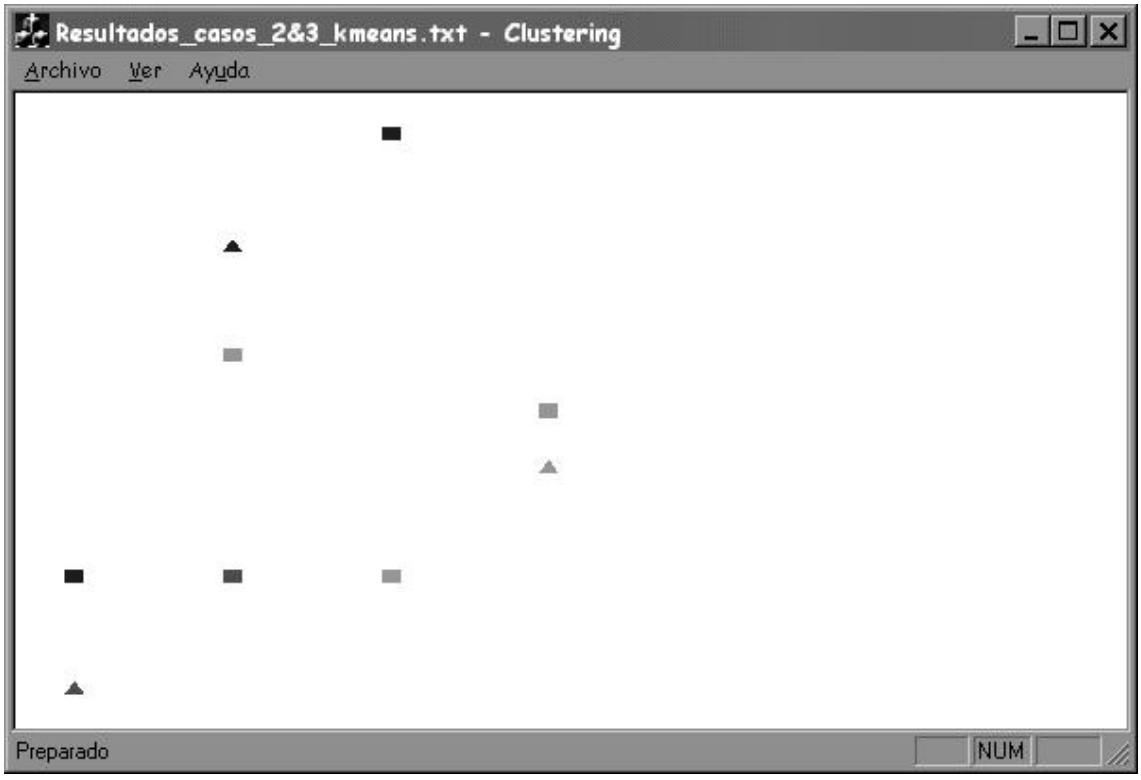
Funcionalidad Testeada: Saturación de la demanda. Se muestran dos casos, uno con la demanda justa y el segundo idéntico al primero pero con la demanda saturada.

Entrada 1:

```
3
6
1 1 0 1 100
2 4 2 1 300
3 2 4 1 200
20 4 2.5 1 100 LNC FinLNC
30 2 1 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 3 5 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada 1:

```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 1 Demanda 100 Producción 100
    Cliente 30
CLUSTER 2
    Deposito 2 #Clientes 3 Demanda 300 Producción 300
    Cliente 20          Cliente 50          Cliente 60
CLUSTER 3
    Deposito 3 #Clientes 2 Demanda 200 Producción 200
    Cliente 40          Cliente 10
CLIENTES NO ASIGNADOS: 0
```

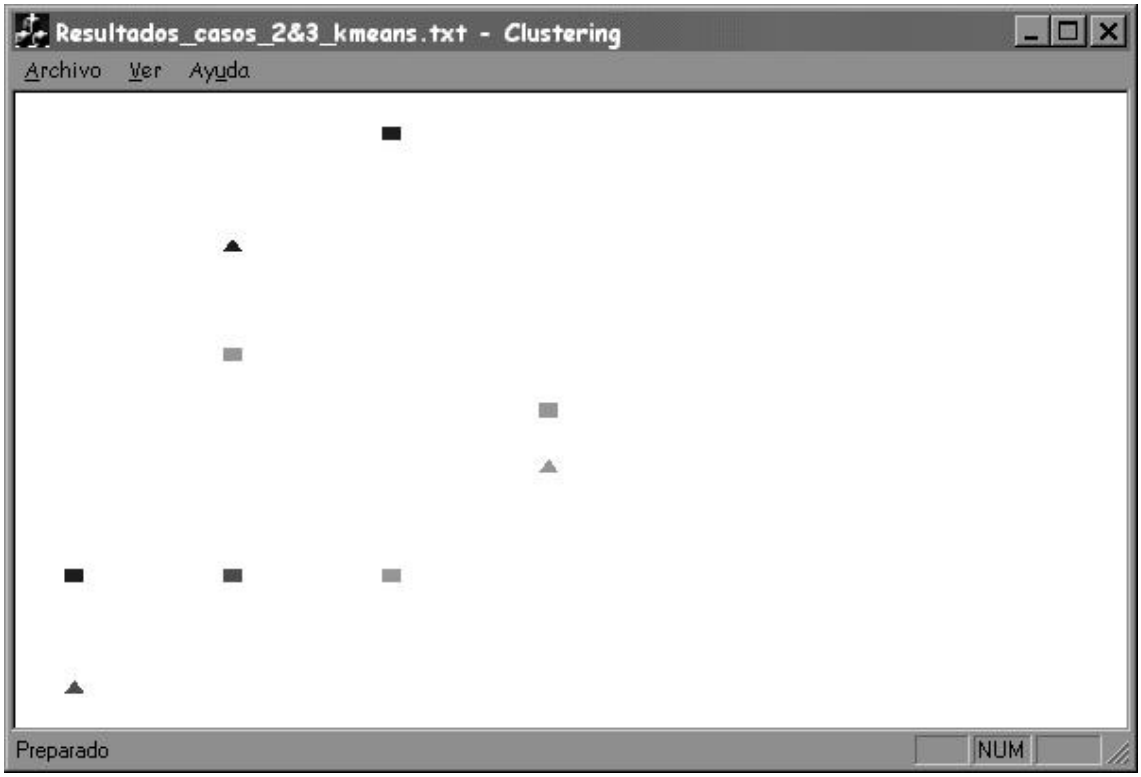


Entrada 2:

```
3
6
1 1 0 1 100
2 4 2 1 300
3 2 4 1 100
20 4 2.5 1 100 LNC FinLNC
30 2 1 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 3 5 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada2: Se pretende que quede un solo cliente sin asignar.

```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 1 Demanda 100 Producción 100
    Cliente 30
CLUSTER 2
    Deposito 2 #Clientes 3 Demanda 300 Producción 300
    Cliente 20          Cliente 40          Cliente 50
CLUSTER 3
    Deposito 3 #Clientes 1 Demanda 100 Producción 100
    Cliente 50
CLIENTES NO ASIGNADOS: 1
    NO Asignado: Cliente 60
```



Resultado Obtenido: Se comporta de acuerdo a la salida esperada

Caso De Testeo 4:

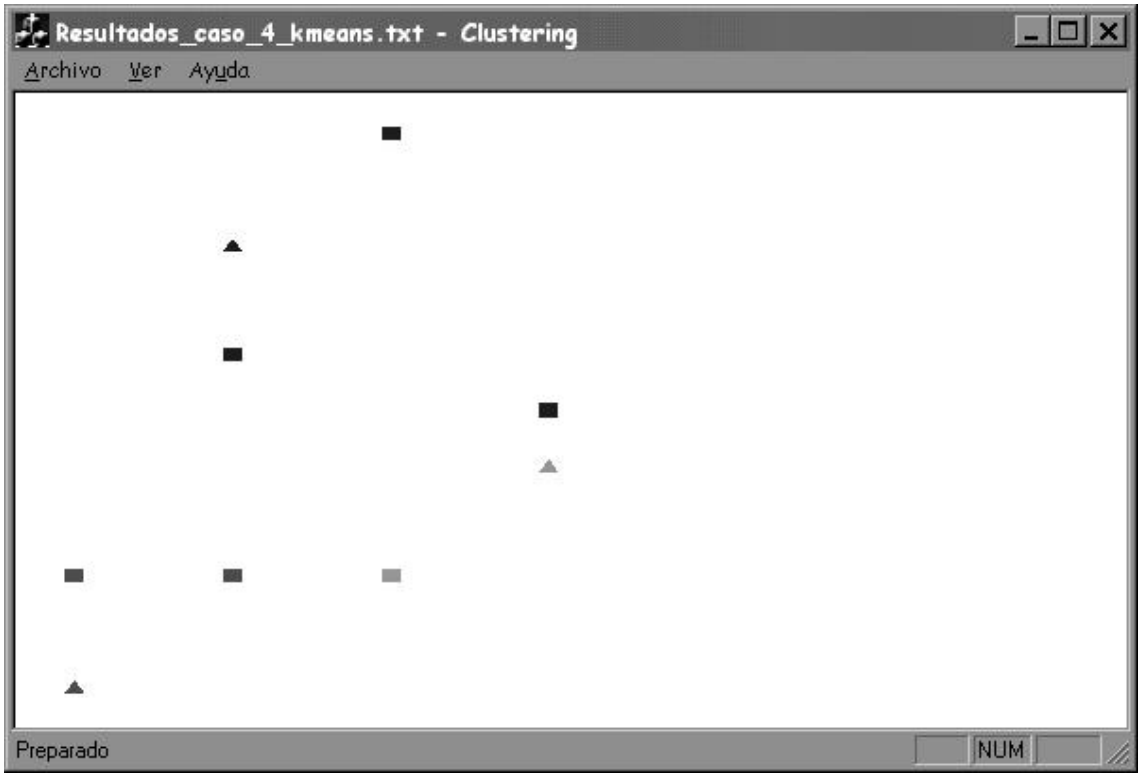
Funcionalidad Testeada: Compatibilidad Cliente-Deposito
Entrada:

```
3
6
1 1 0 1 500
2 4 2 1 500
3 2 4 1 500
20 4 2.5 1 100 LNC 2 FinLNC
30 2 1 1 100 LNC 2 FinLNC
10 1 1 1 100 LNC FinLNC
40 3 5 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC 1 FinLNC
```

Salida Esperada:

```
3 CLUSTERS
CLUSTER 1
    Deposito 1 #Clientes 2 Demanda 500 Produccion 200
    Cliente 30          Cliente 10
CLUSTER 2
    Deposito 2 #Clientes 1 Demanda 500 Producción 100
    Cliente 60
CLUSTER 3
    Deposito 3 #Clientes 3 Demanda 500 Producción 300
    Cliente 20          Cliente 40          Cliente 50
CLIENTES NO ASIGNADOS: 0
```

Resultado Obtenido: Se comporta de acuerdo a la salida esperada



Caso De Testeo 5:

Funcionalidad Testeada: Dependencia del orden de entrada
Entrada: Se corre el mismo archivo pero los clientes van ordenados distinto
Entrada 1:

```
3
8
1 0 0 1 500
2 10 0 1 500
3 6 -4 1 500
10 2 0 1 100 LNC FinLNC
20 5 0 1 100 LNC FinLNC
30 7 1 1 100 LNC FinLNC
40 6 -1 1 100 LNC FinLNC
50 6 -3 1 100 LNC FinLNC
60 5 -2 1 100 LNC FinLNC
70 6 -2.5 1 100 LNC FinLNC
80 4 -2 1 100 LNC FinLNC
```

Entrada 2:

```
3
8
1 0 0 1 500
2 10 0 1 500
3 6 -4 1 500
20 5 0 1 100 LNC FinLNC
40 6 -1 1 100 LNC FinLNC
60 5 -2 1 100 LNC FinLNC
30 7 1 1 100 LNC FinLNC
70 6 -2.5 1 100 LNC FinLNC
80 4 -2 1 100 LNC FinLNC
50 6 -3 1 100 LNC FinLNC
10 2 0 1 100 LNC FinLNC
```

Salida Esperada 1:

3 CLUSTERS
CLUSTER 1
Deposito 1 #Clientes 1 Demanda 500 Producción 100
Cliente 10
CLUSTER 2
Deposito 2 #Clientes 2 Demanda 500 Producción 200
Cliente 20 *Cliente 30*
CLUSTER 3
Deposito 3 #Clientes 5 Demanda 500 Producción 500
Cliente 40 *Cliente 50* *Cliente 60* *Cliente 70* *Cliente 80*
CLIENTES NO ASIGNADOS: 0

Salida Esperada 2:
Ídem Salida Esperada 1

Resultado Obtenido: Se comporta de acuerdo a las salidas esperadas

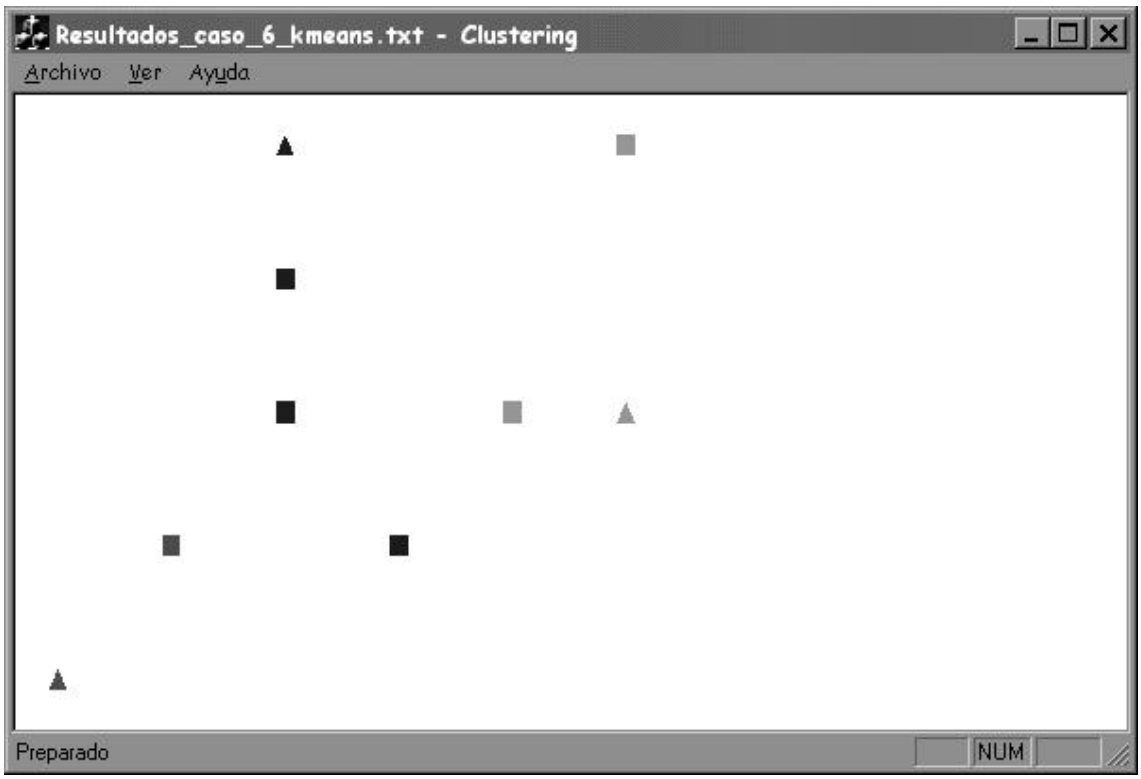
Caso De Testeo 6:

Funcionalidad Testeada: Comparar el resultado con el obtenido corriendo KNN para el *caso 1.*
Entrada:

```
3
6
1 0 0 1 1500
2 5 2 1 1500
3 2 4 1 1500
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC FinLNC
50 2 3 1 100 LNC FinLNC
60 3 1 1 100 LNC FinLNC
```

Salida Esperada:

```
 3 CLUSTERS
CLUSTER 1
      Deposito 1 #Clientes 1 Demanda 1500 Producción 100
      Cliente 10
CLUSTER 2
      Deposito 2 #Clientes 2 Demanda 1500 Producción 200
      Cliente 20          Cliente 40
CLUSTER 3
      Deposito 3 #Clientes 3 Demanda 1500 Producción 300
      Cliente 30          Cliente 50          Cliente 60
CLIENTES NO ASIGNADOS: 0
```



Resultado Obtenido: Se comporta de acuerdo a la salida esperada

Caso De Testeo 7:

Funcionalidad Testeada: No poner depositos en el conjunto de datos

Entrada:
0
6
20 4 2 1 100 LNC FinLNC
30 2 2 1 100 LNC 3 FinLNC
10 1 1 1 100 LNC FinLNC
40 5 4 1 100 LNC 2 FinLNC
50 2 3 1 100 LNC 1 2 3 FinLNC
60 3 1 1 100 LNC FinLNC

Salida Esperada: 0 clusters creados y reconocimiento de los 6 clientes

Resultado Obtenido: Se comporta de acuerdo a la salida esperada

2.4.2.4 Testeo Exhaustivo

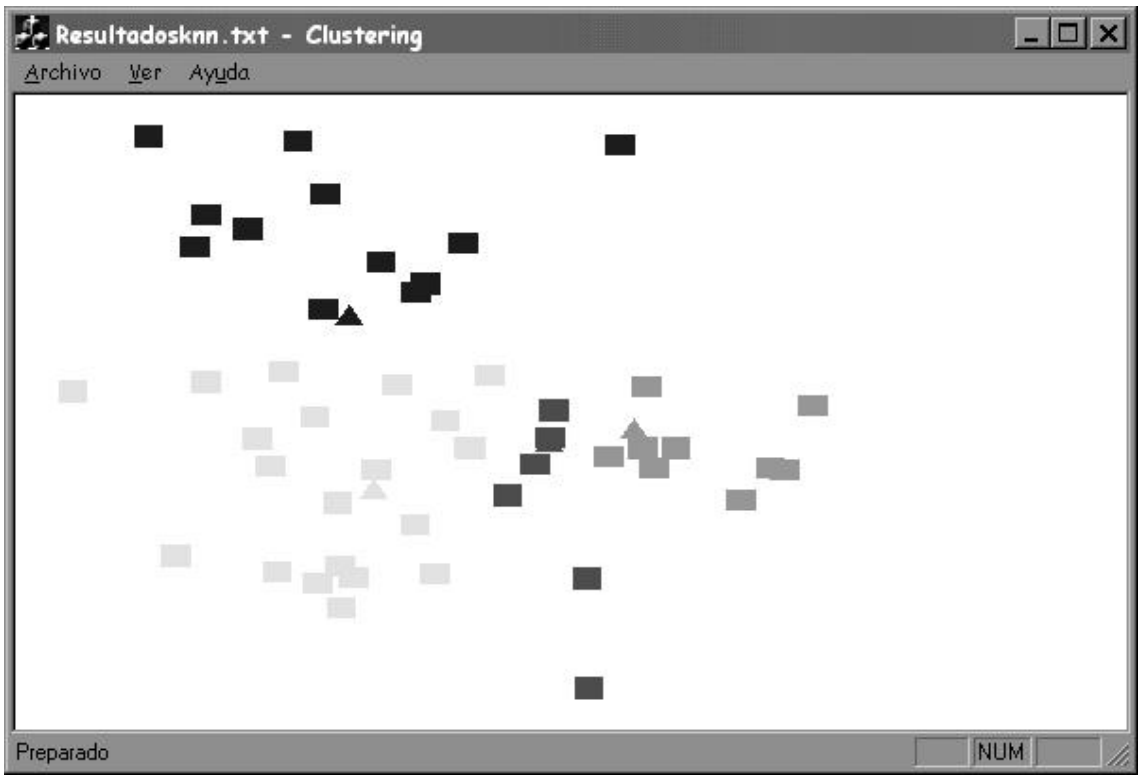
En este punto se pretende verificar el correcto funcionamiento de los algoritmos frente a entradas de gran tamaño. A diferencia del testeo de funcionalidades visto en los puntos anteriores para los diferentes algoritmos, al testear de esta manera no se tiene a priori una salida esperada.

Para lograr un análisis visual de los resultados obtenidos, se testará con ponderante de la distancia temporal en cero (o con la componente temporal de Clientes y Depósitos iguales) -técnica ya utilizada en los casos de testeo de funcionalidades- por ser este un método viable (por simplicidad y rapidez) para el correcto escrutinio visual de los resultados (recordar que estos se presentan en dos dimensiones).

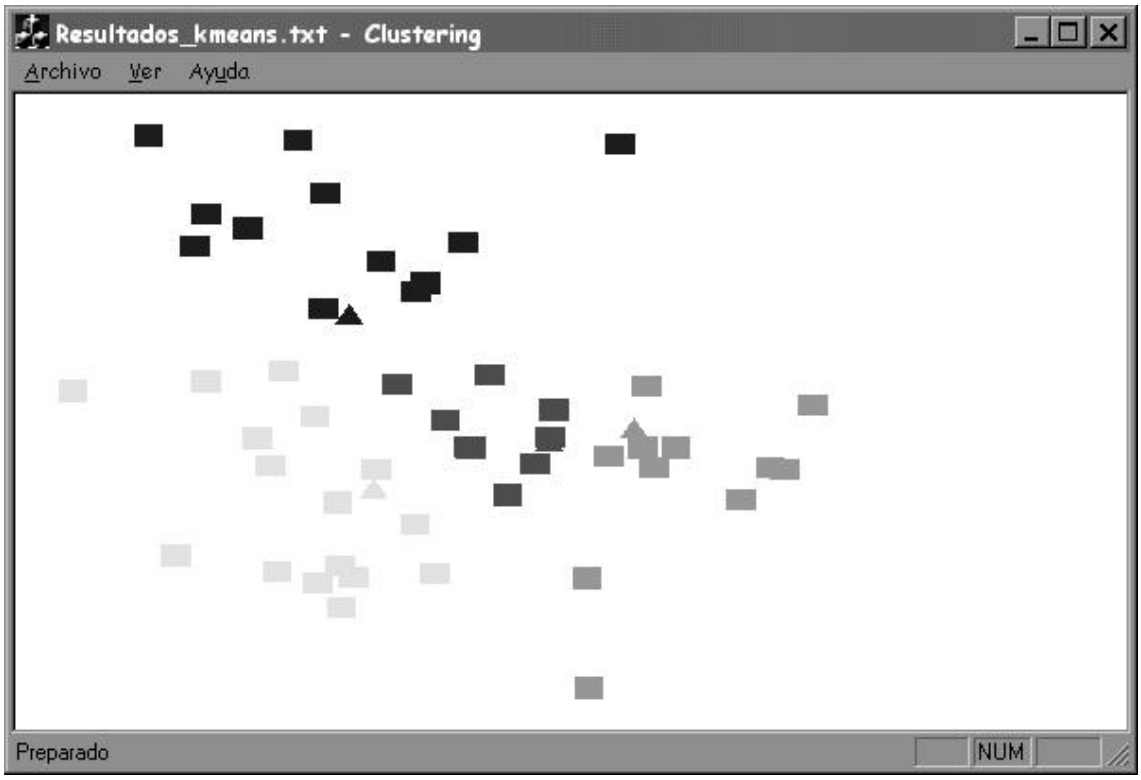
Se utilizará el mismo grupo de entradas para los tres algoritmos, obteniendo así la posibilidad de comparar los resultados obtenidos al mismo tiempo que se comprueba su buen funcionamiento.

Caso 1 (Ver datos en D.4: Entrada 1):

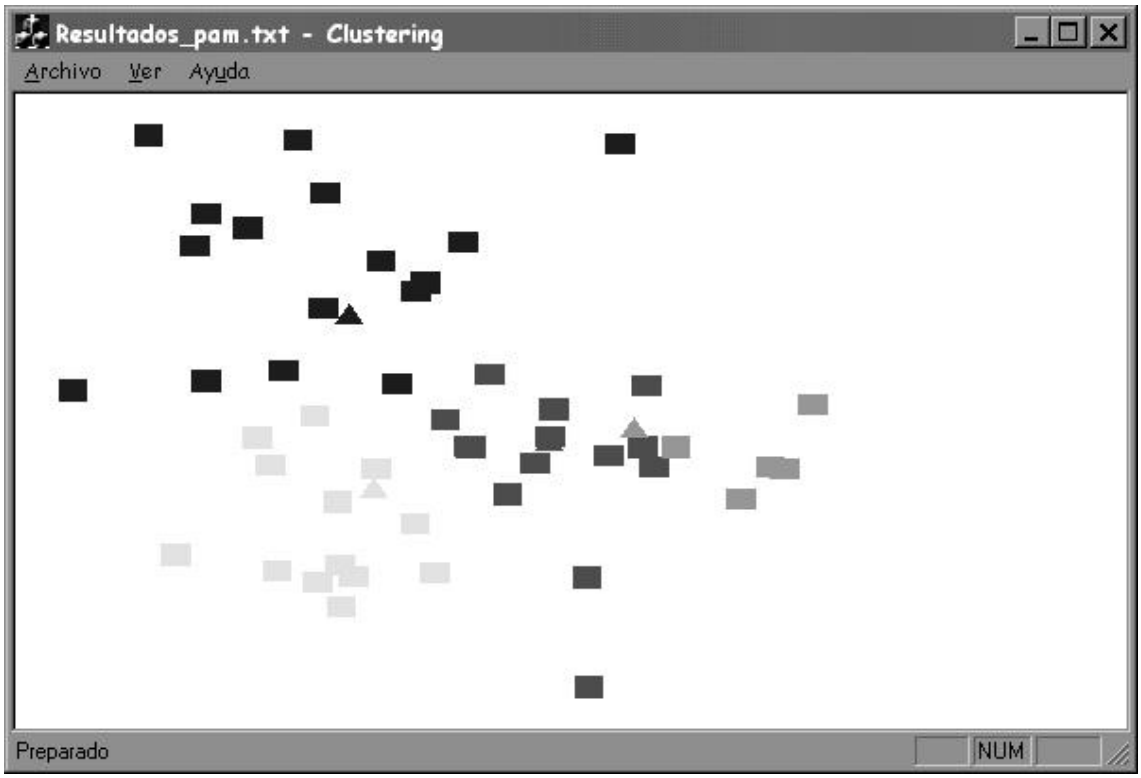
KNN con k=1



K-MEANS


Resultados_kmeans.txt - Clustering
Archivo Ver Ayuda
Preparado
NUM

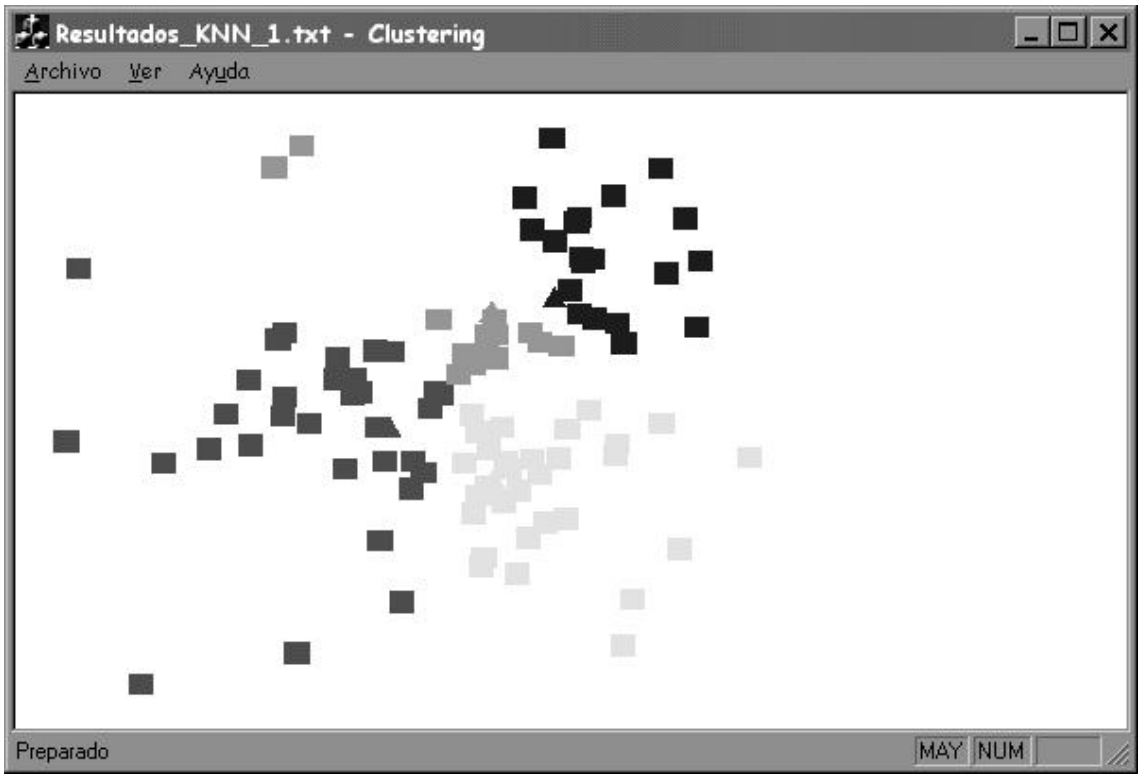
PAM


Resultados_pam.txt - Clustering
Archivo Ver Ayuda
Preparado
NUM

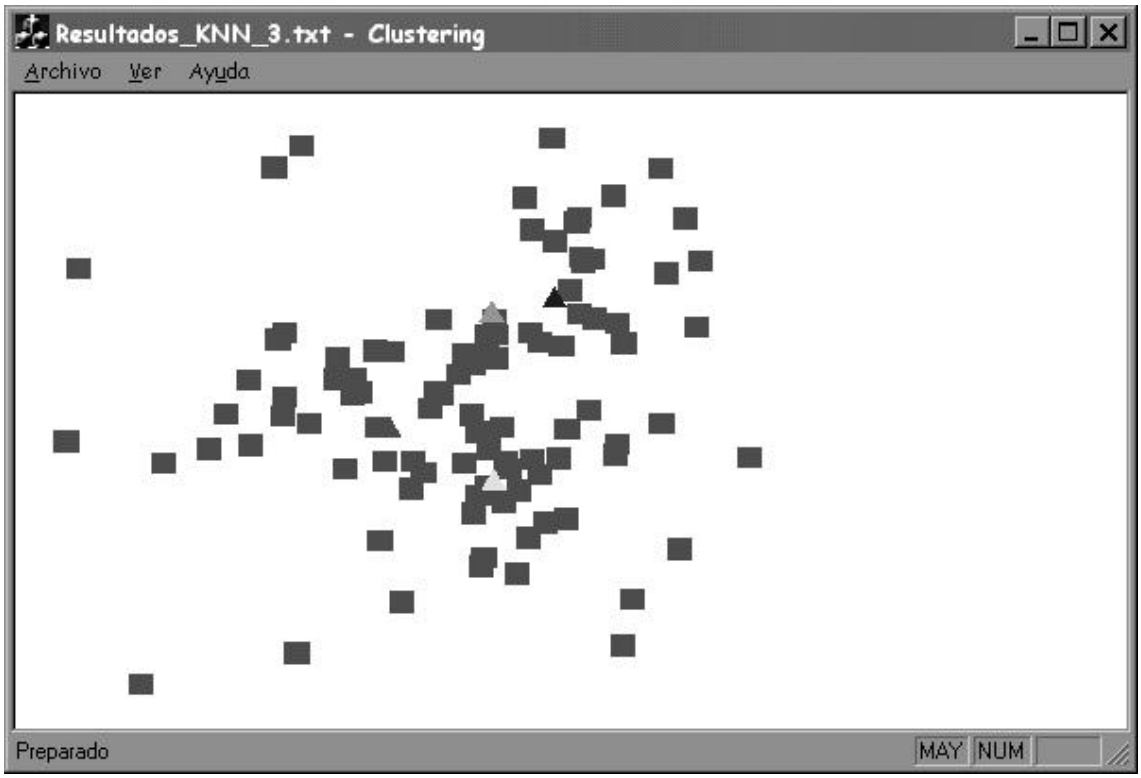
Cuando se toma k=1 para el algoritmo knn, se obtiene una buena distribución en los tres algoritmos corridos, aunque son apreciables a simple vista diferencias en los clusters obtenidos.

Caso 2 (Ver datos en D.4: Entrada 2):

KNN con k=1

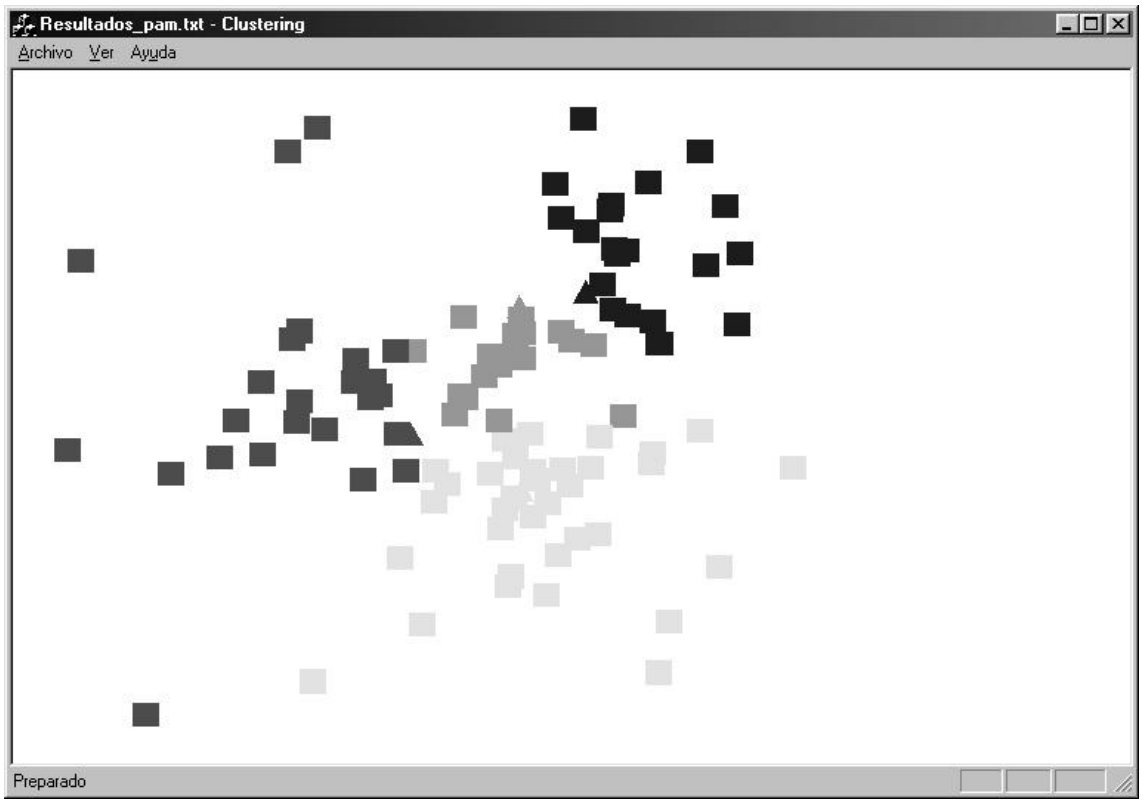


KNN con k=3



KMEANS

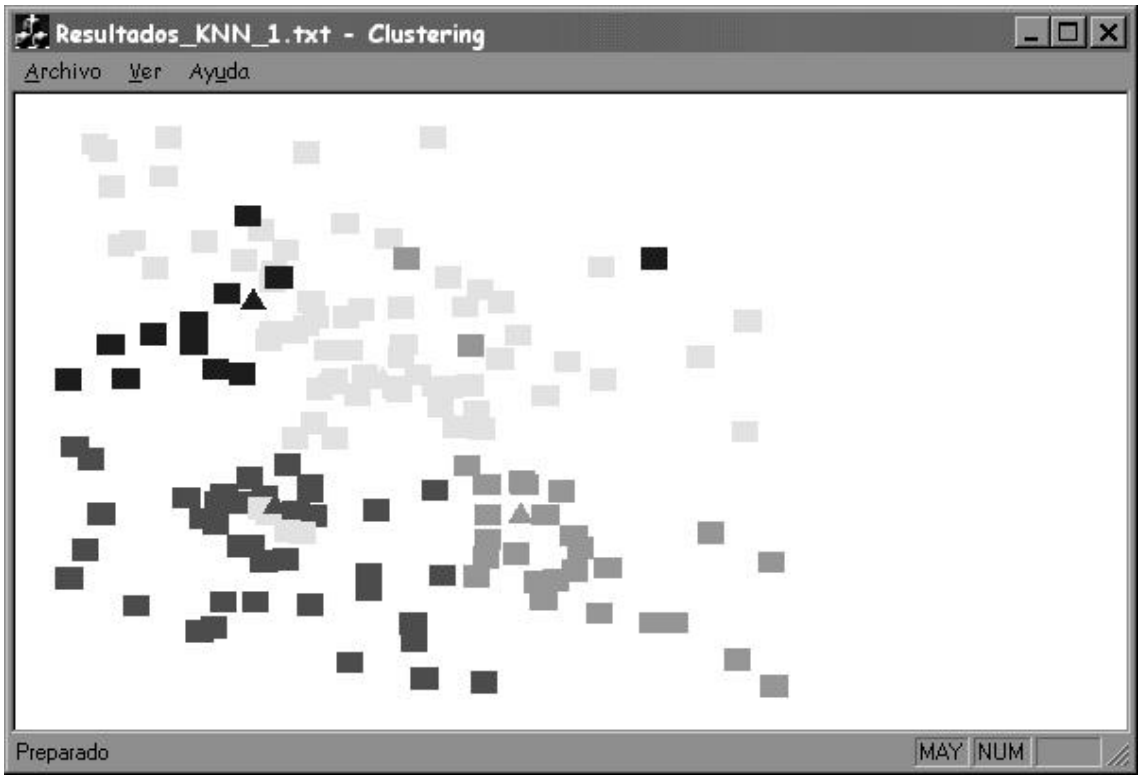
PAM



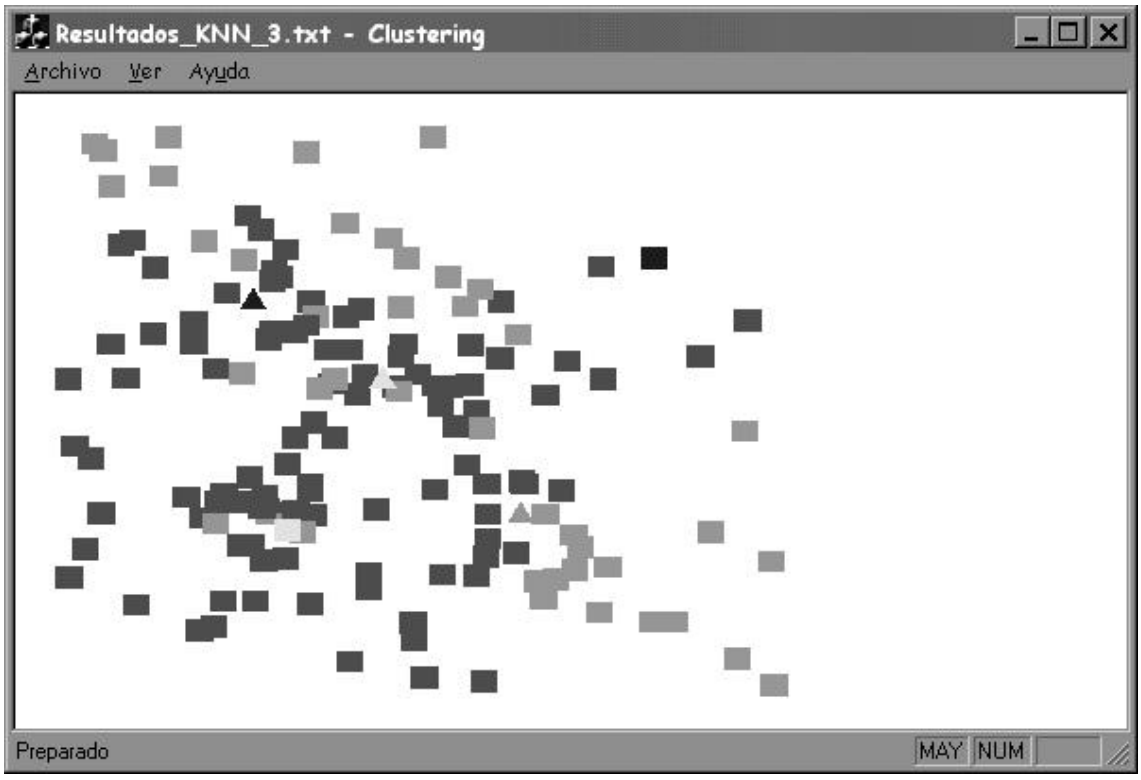
En la primer pantalla se observa la corrida para KNN con k=1, que tiene un comportamiento similar a K-Means y PAM. Sin embargo se puede apreciar la deformación de los clusters para KNN si tomamos k=3.

Caso 3 (Ver datos en D.4: Entrada 3):

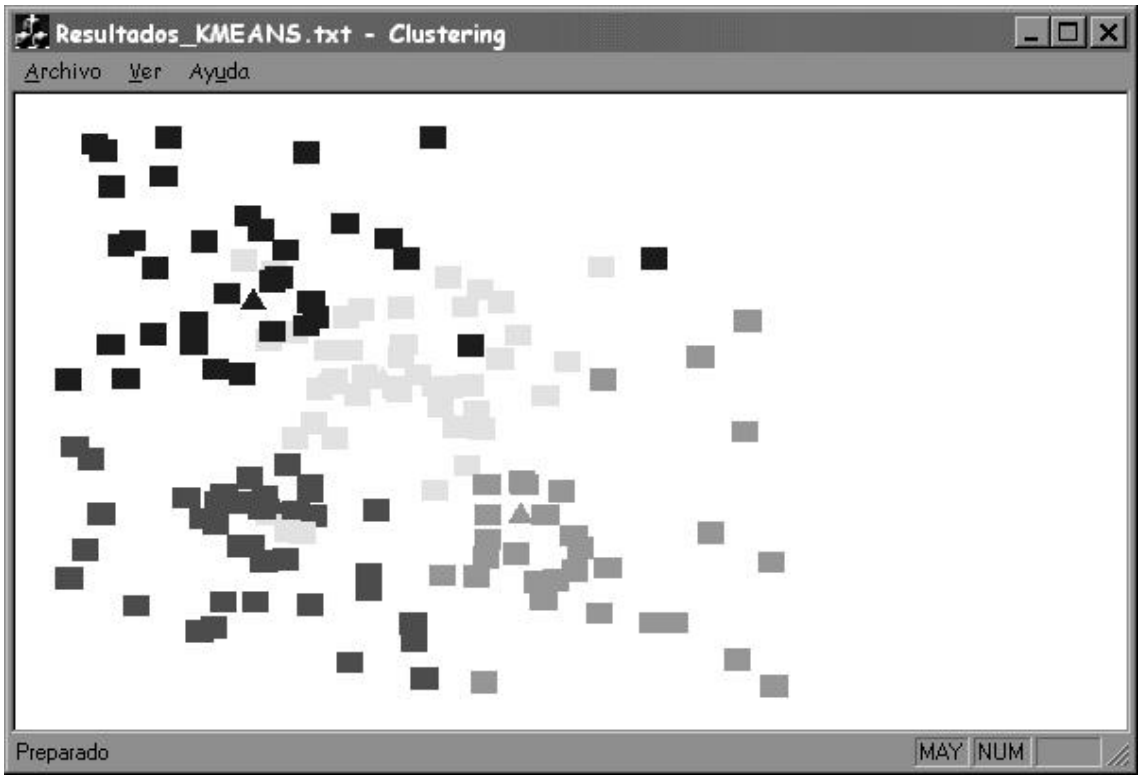
KNN con k=1



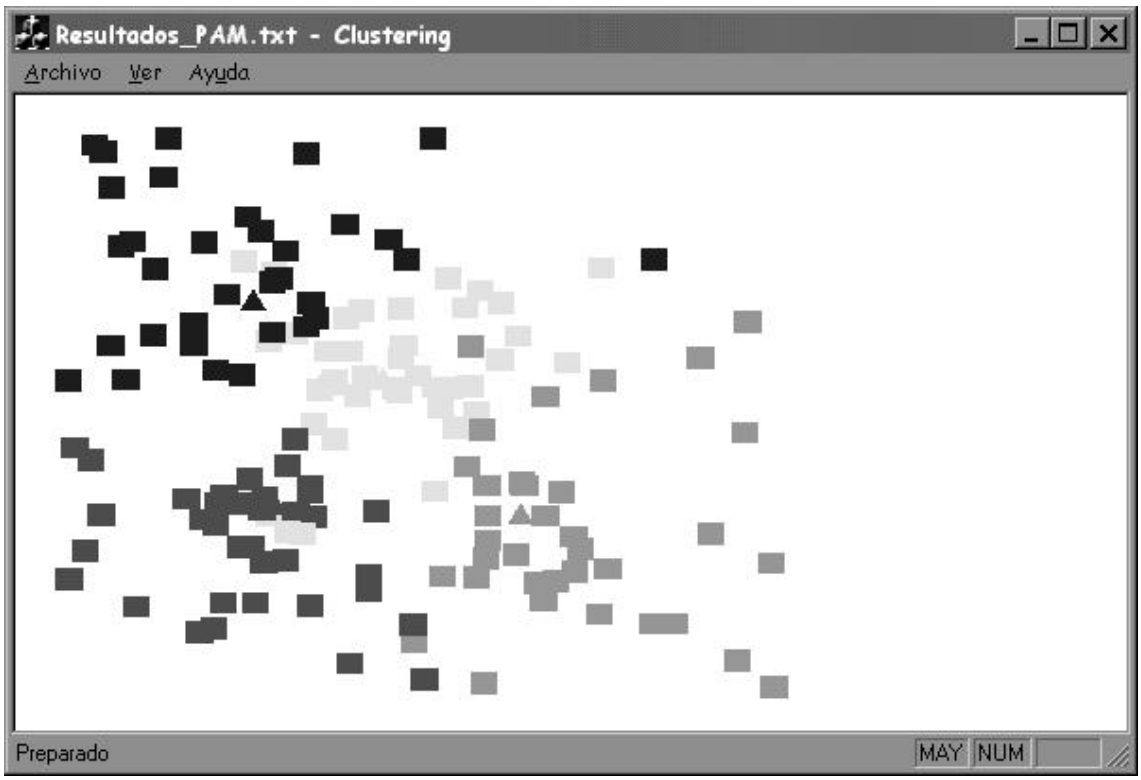
KNN con k=3



KMEANS


Resultados_KMEANS.txt - Clustering
Archivo Ver Ayuda
Preparado
MAY
NUM

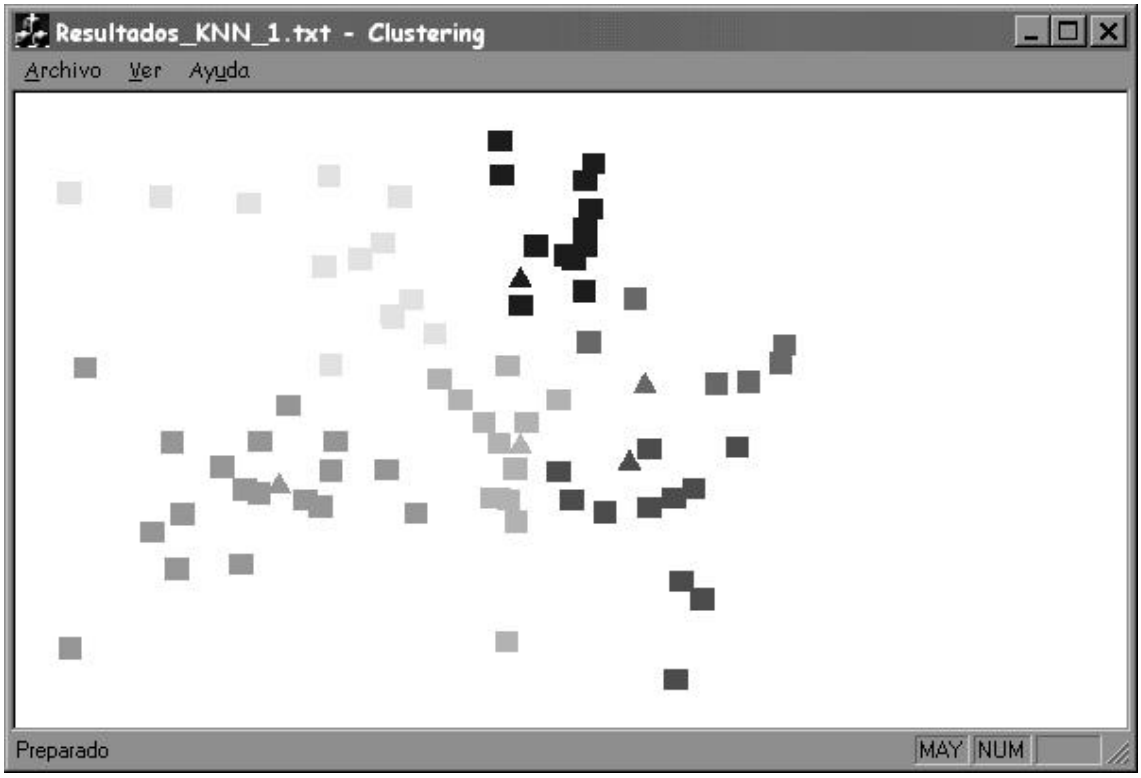
PAM



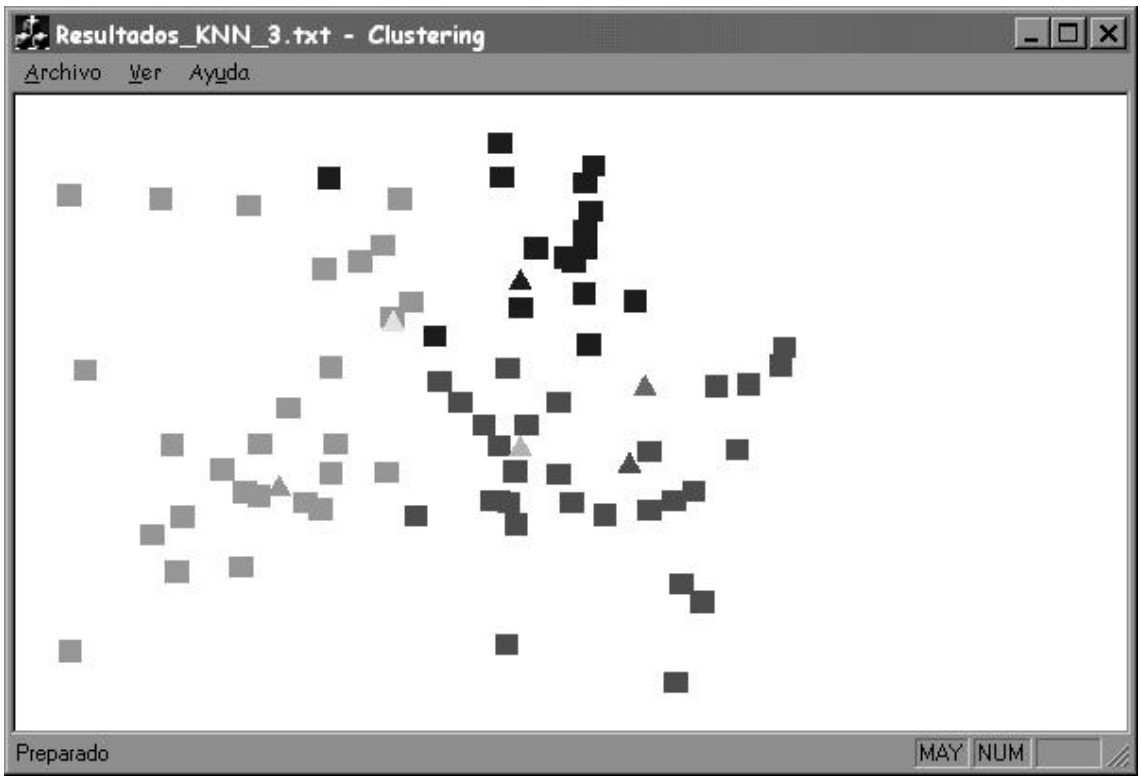
Al igual que en el Caso 2, se puede observar la diferencia de comportamiento al variar *k* para knn. En este caso, se incluyeron también incompatibilidades claramente perceptibles a simple vista.

Caso 4 (Ver datos en D.4: Entrada 4):

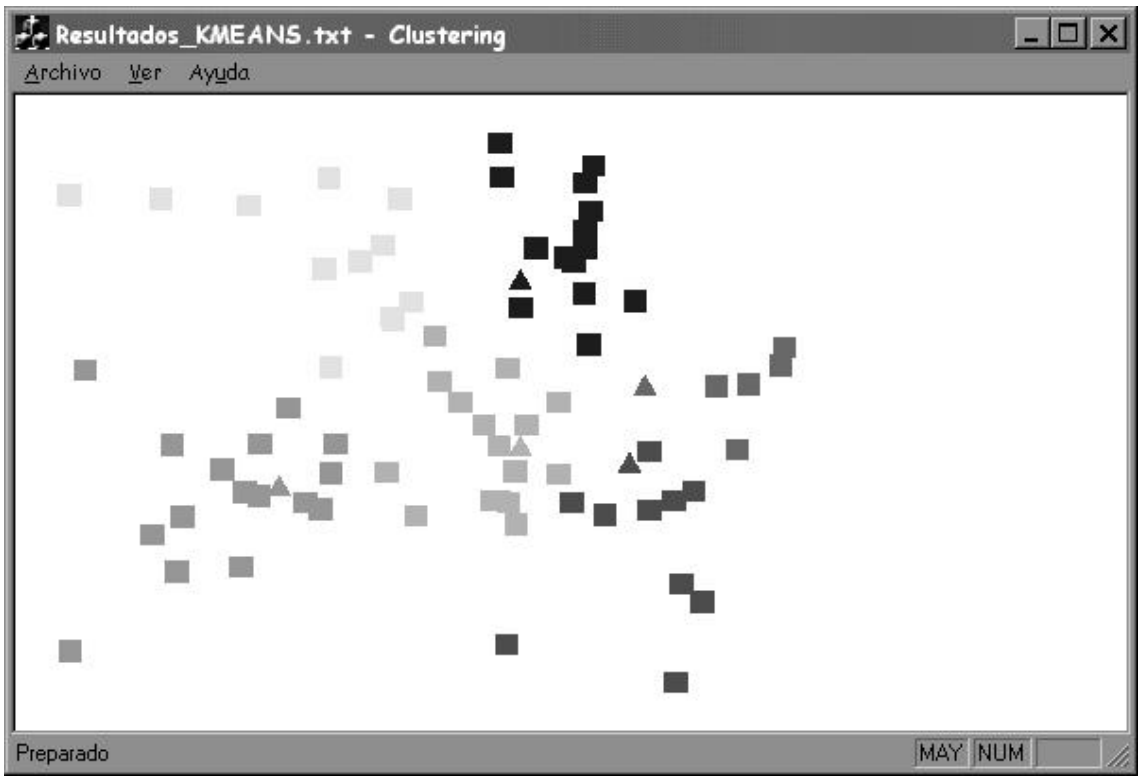
KNN con k=1



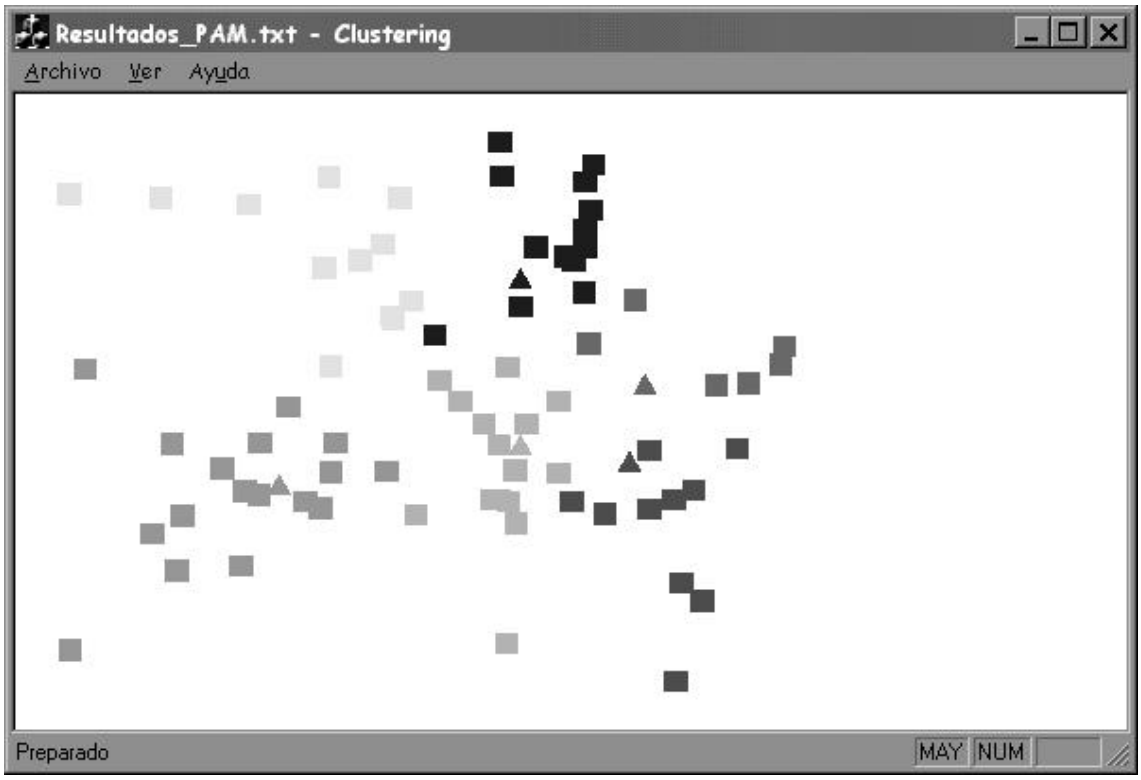
KNN con k=3



KMEANS



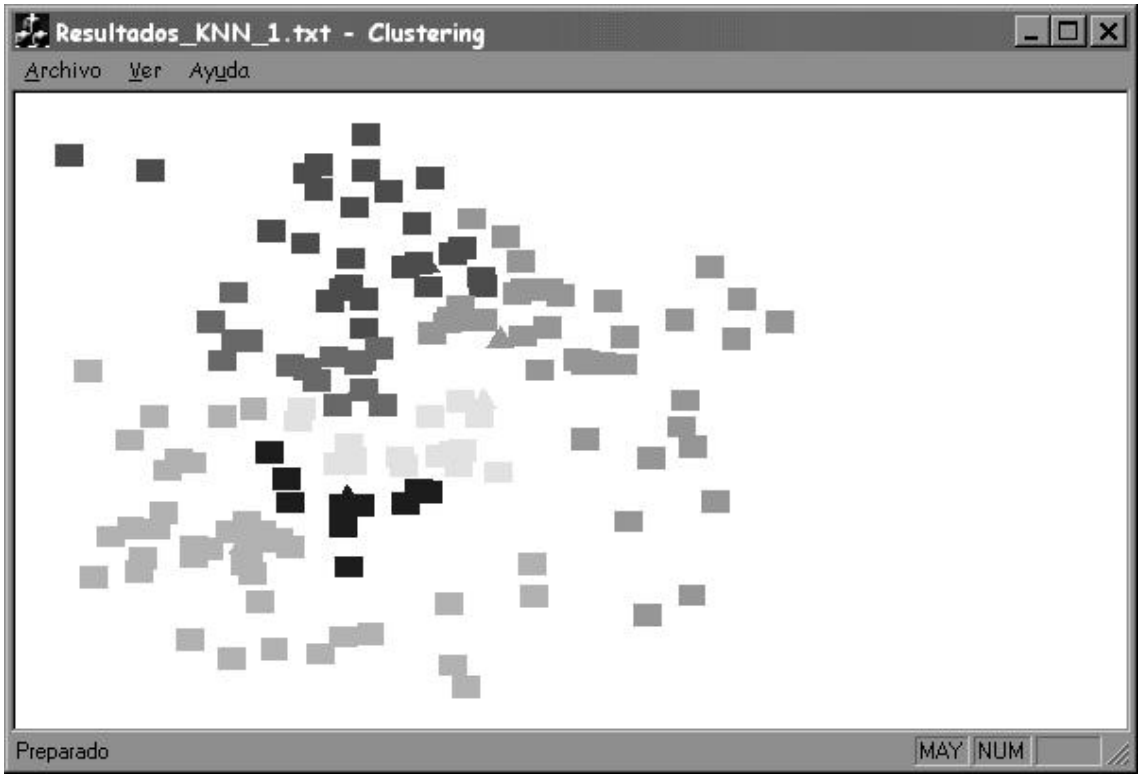
PAM



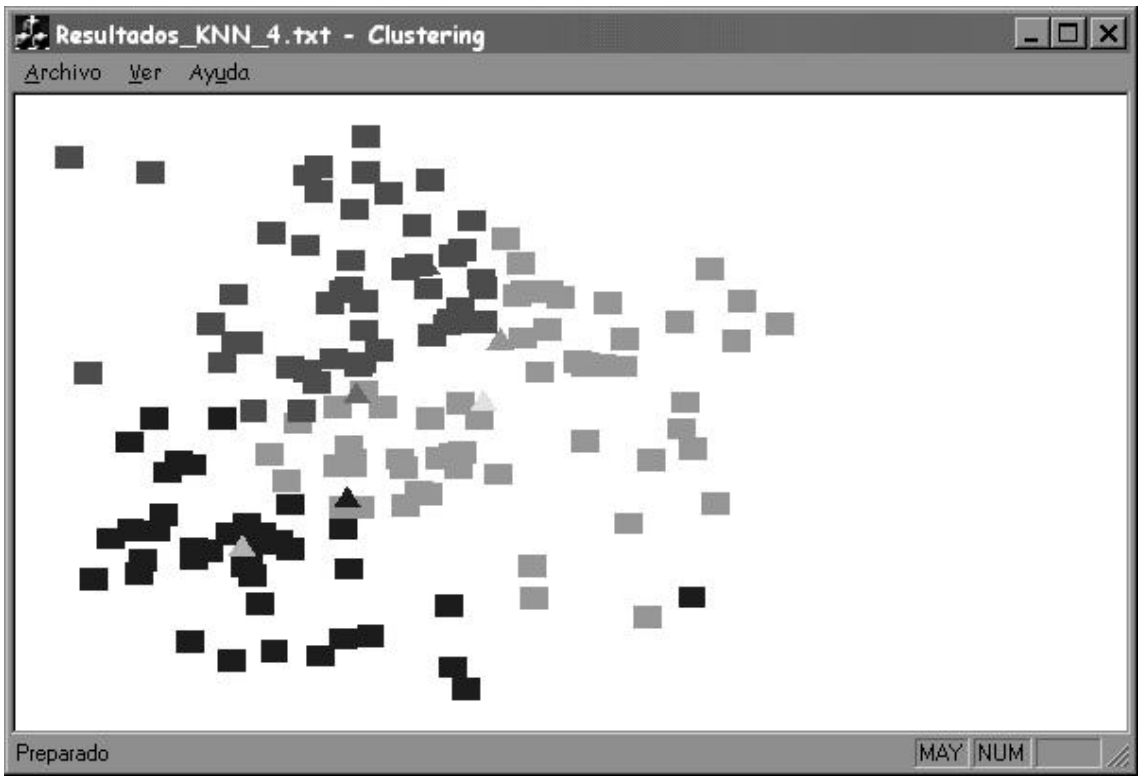
Es importante observar como cuando k=3, knn se aleja de los resultados obtenidos por kmeans y pam, disminuyendo la cantidad de clusters encontrados.

Caso 5 (Ver datos en D.4: Entrada 5):

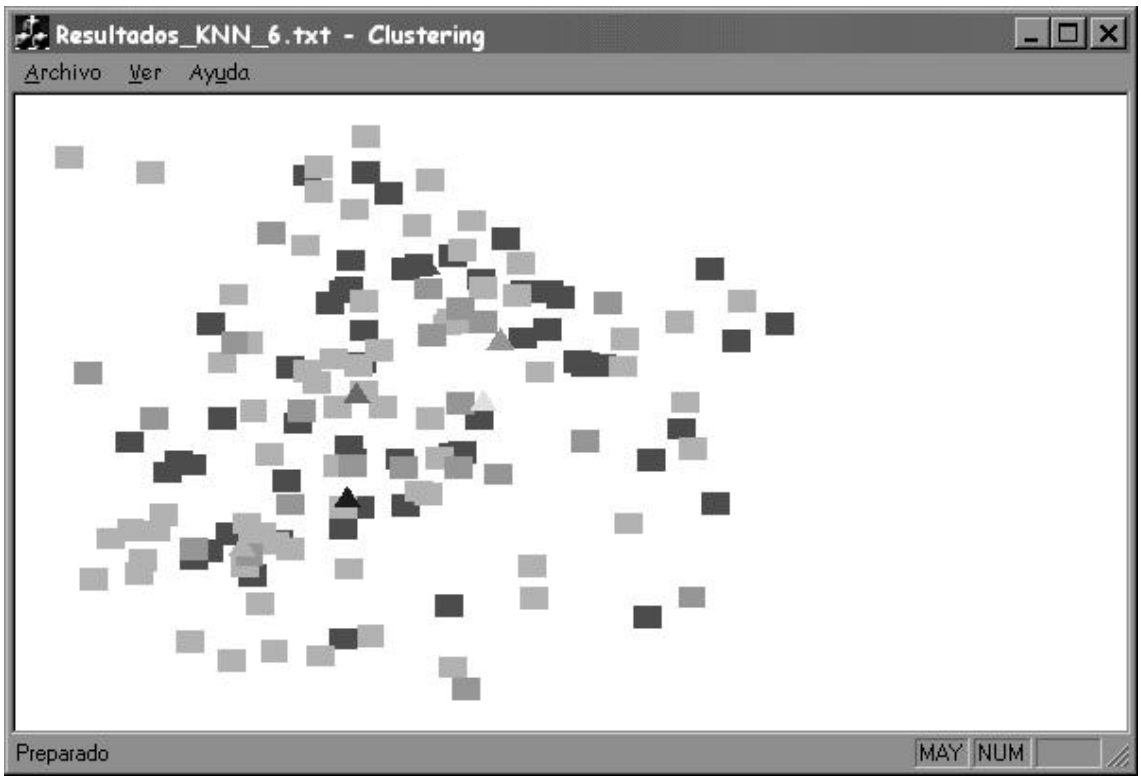
Knn k=1



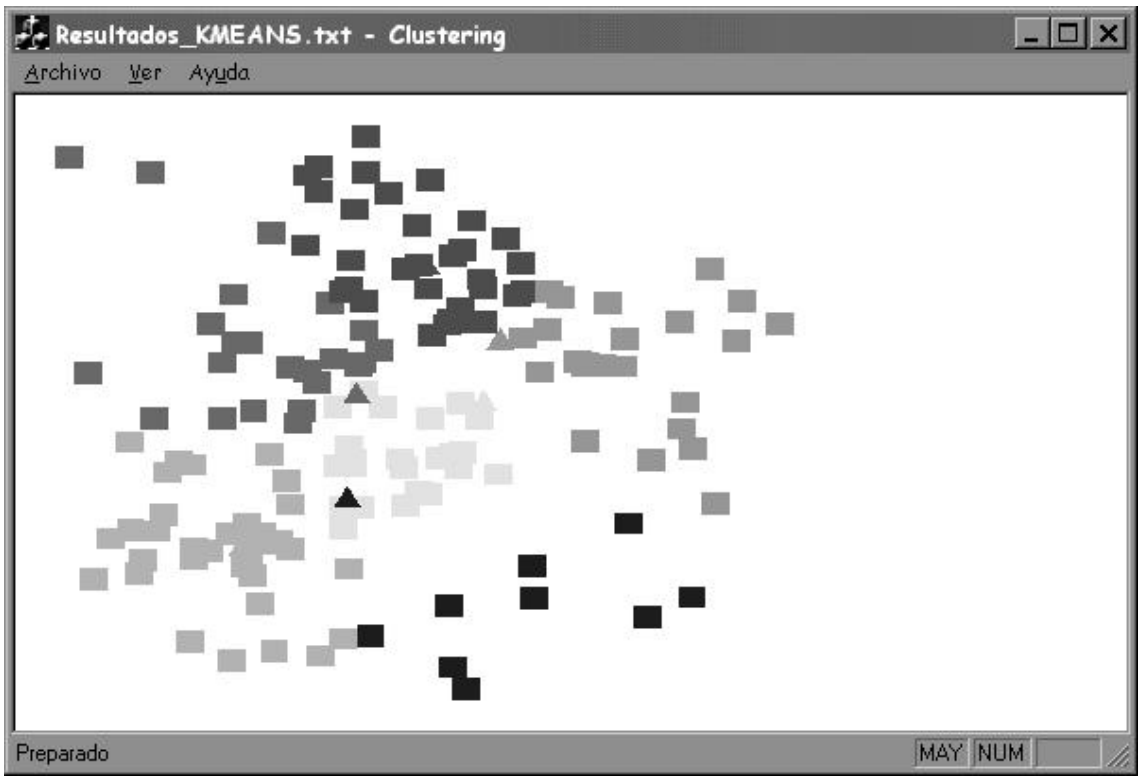
Knn k=4


Resultados_KNN_4.txt - Clustering
Archivo Ver Ayuda
Preparado
MAY NUM

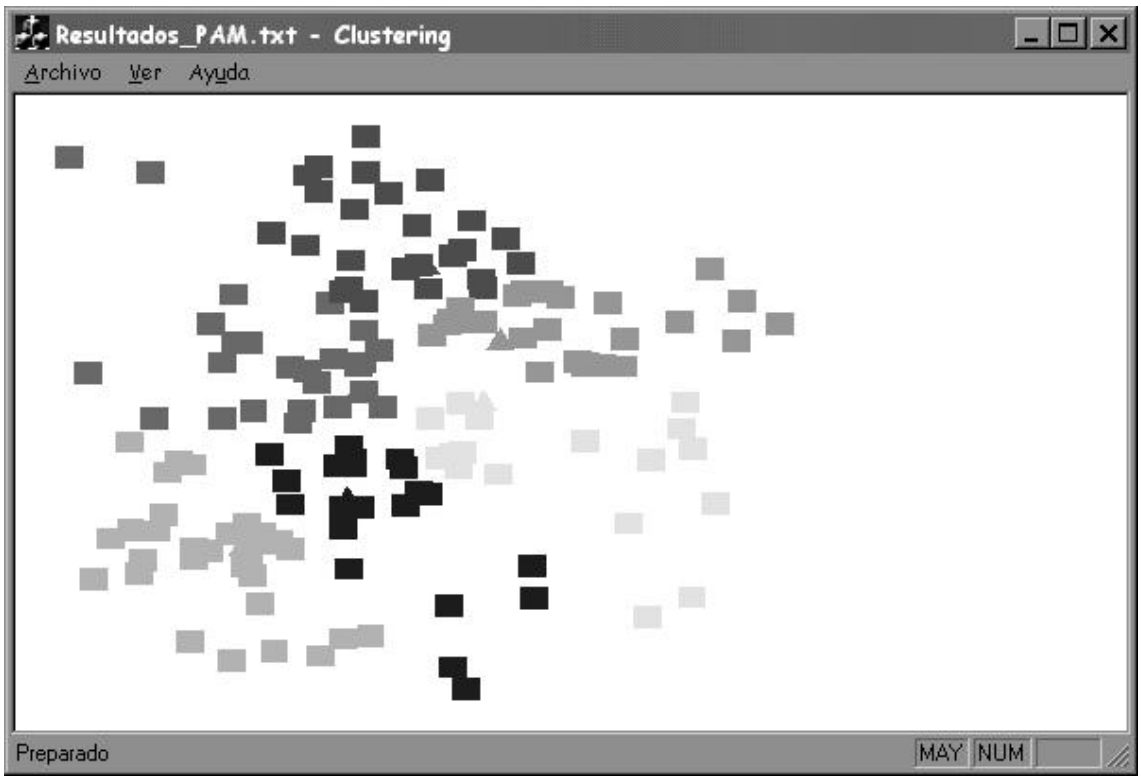
Knn k=6


Resultados_KNN_6.txt - Clustering
Archivo Ver Ayuda
Preparado
MAY NUM

KMEANS


Resultados_KMEANS.txt - Clustering
Archivo Ver Ayuda
Preparado
MAY NUM

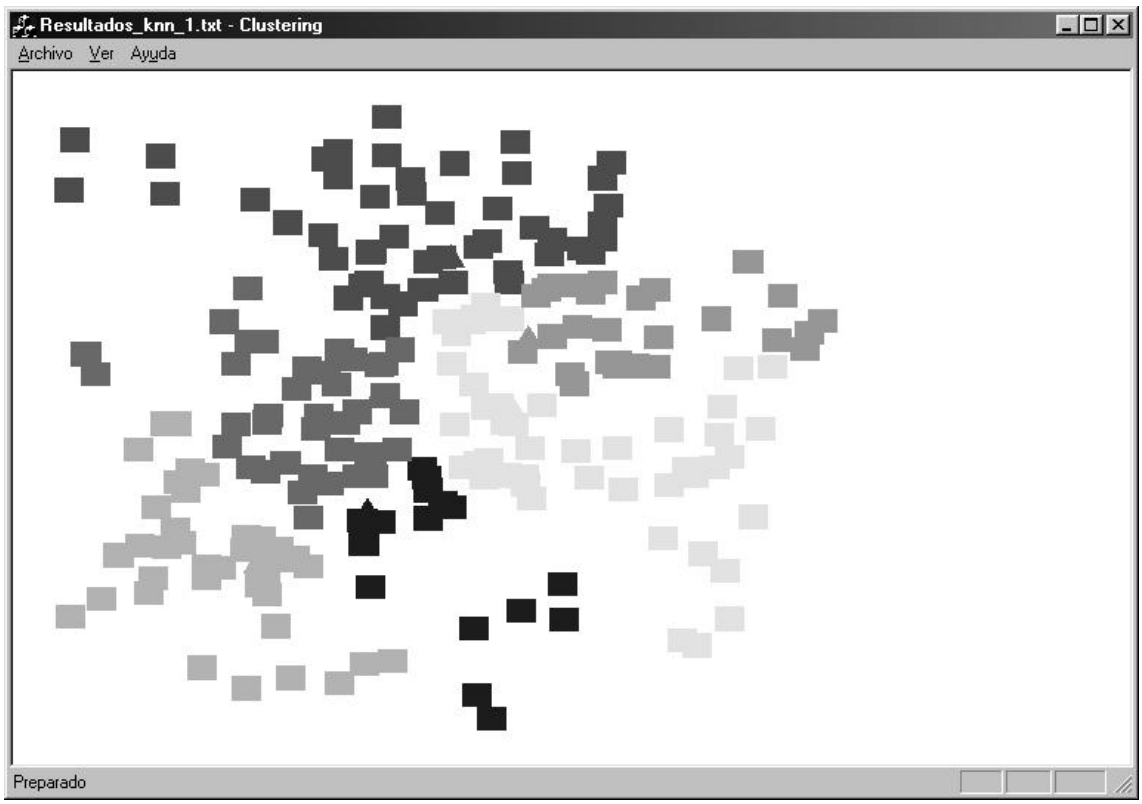
PAM


Resultados_PAM.txt - Clustering
Archivo Ver Ayuda
Preparado
MAY NUM

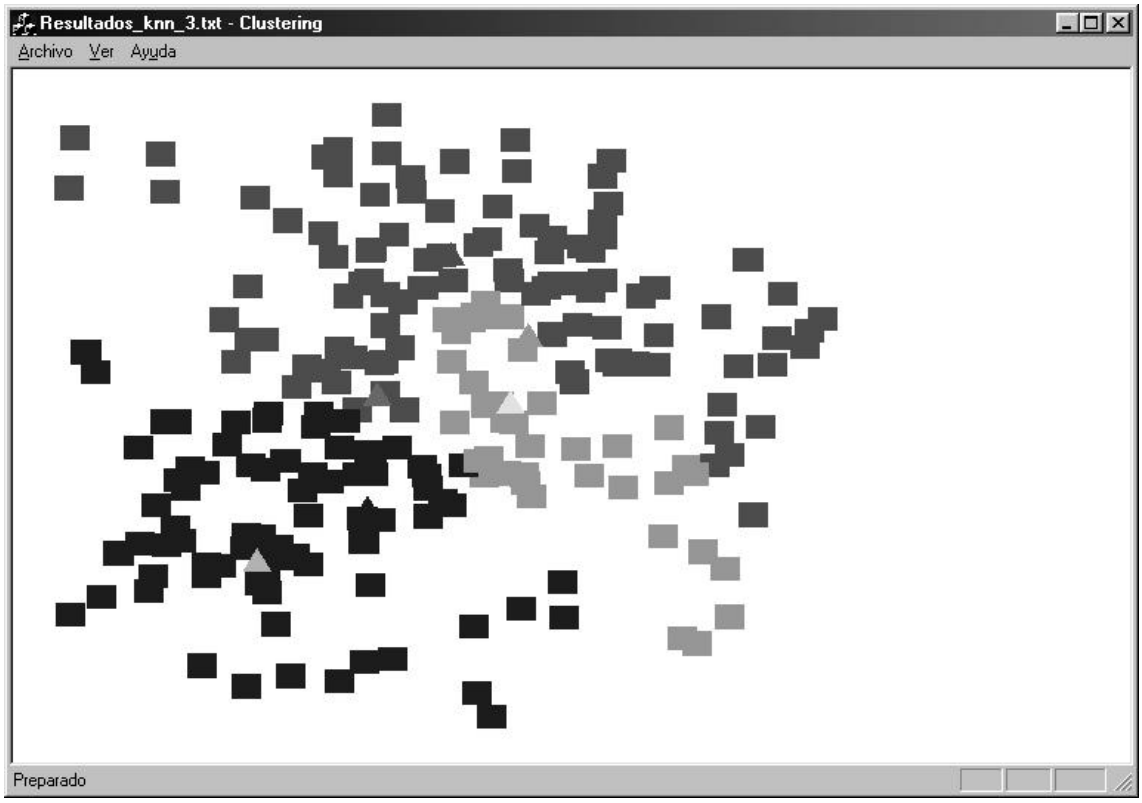
Se pueden observar nuevamente, los cambios en los resultados al variar k, aunque para k=1 se mantiene el comportamiento similar a los otros dos algoritmos.

Caso 6 (Ver datos en D.4: Entrada 6):

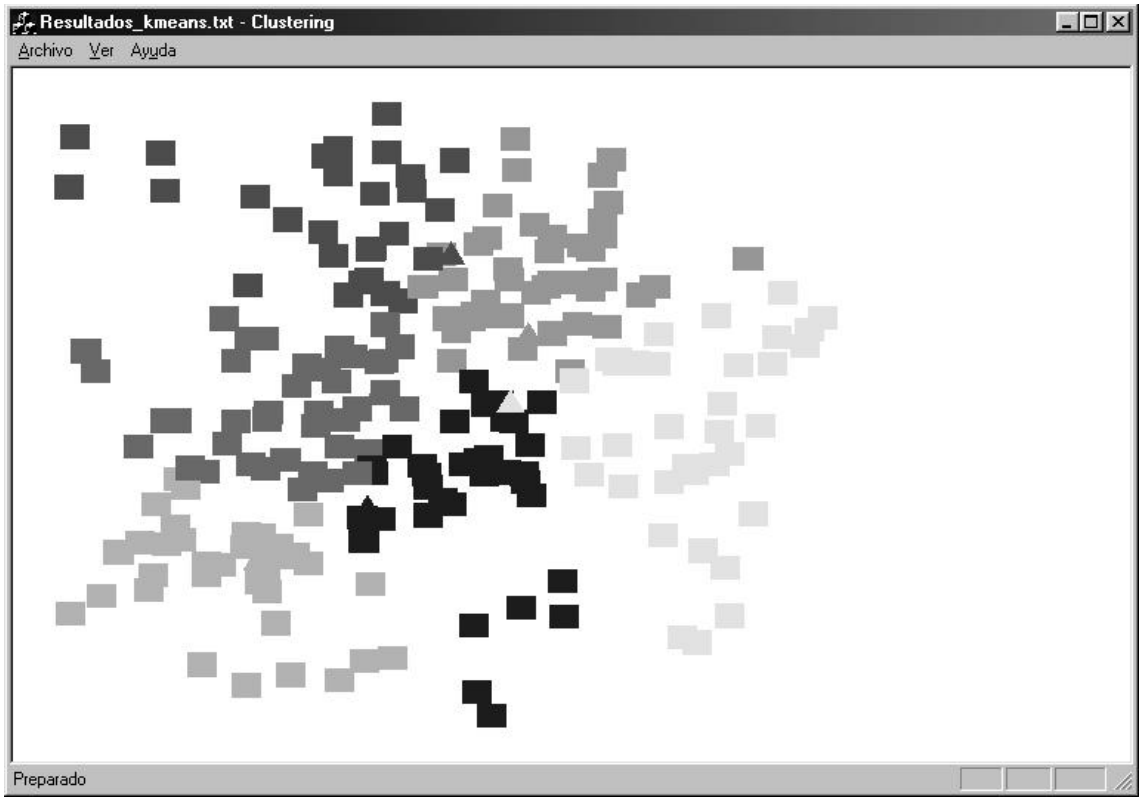
KNN con k=1



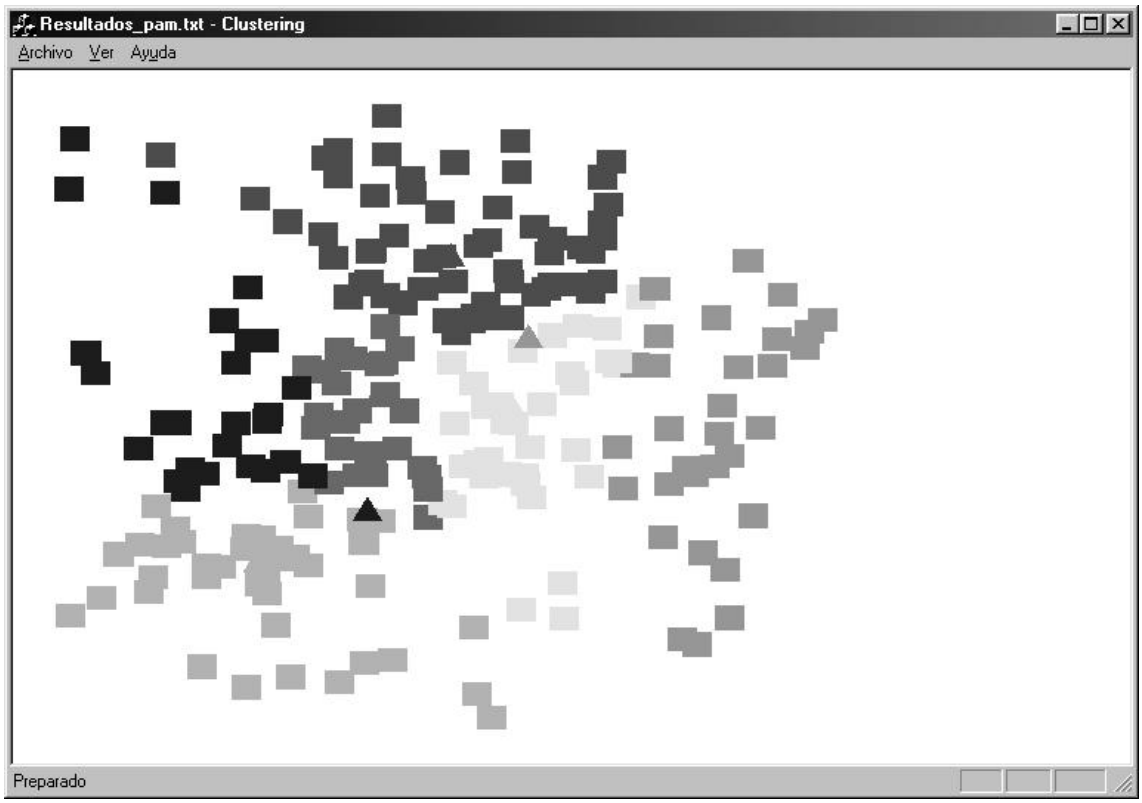
KNN con k=3



KMEANS



PAM



Aunque los tres algoritmos logren identificar seis clusters distintos, es apreciable la diferencia entre los distintos resultados.

De las distintas corridas realizadas se desprenden los siguientes resultados:

Knn en particular, logra una mejor clusterización cuando *k* es menor, deformando el resultado a medida que se incrementa su valor.

PAM y K-Means tienen un comportamiento muy similar en cuanto a los resultados obtenidos (Recordar que mientras K-Means tiene una media exacta para cada cluster aunque esta no pertenezca al dominio del problema, PAM considera un elemento del cluster como la "media" por eso denominado mediode).

3 Resultados

3.1 Consideraciones Previas

Para comparar las técnicas de Clustering y los resultados por ellas generados, es necesario establecer una medida de "bondad" del Clustering. A continuación se enumeran algunas alternativas:

- Lograr que los subconjuntos resultantes del Clustering sean homogéneos aunque el conjunto inicial no lo sea
- Que no queden elementos sin clasificar
- El método aplicado sea independiente del orden de entrada de los datos
- Que al ingresar nuevos elementos los resultados no se alteren demasiado o en otras palabras, tener cierta estabilidad en el crecimiento
- Descubrir algún tipo de "estructura" en los datos de entrada, si es que ésta existe. Este criterio esta ampliamente vinculado a la idea de descubrimiento de información
- Obtener una correcta distribución de los datos de entrada, de acuerdo a algún criterio de correctitud previamente definido
- Que la detección de pequeños errores en los datos de ingreso, lleve a pequeños cambios en el Cluster

Como posibles medidas de bondad aplicables a nuestro problema que permiten la comparación de los resultados obtenidos tenemos:

- Minimización de las distancias recorridas.
- Saturación de las demandas minimizando las distancias recorridas.
- Ídem anterior, minimizando el empleo de la flota (solo en caso de disponer de datos de flota).

En particular, como medida de bondad aplicable a nuestros algoritmos utilizamos el criterio de minimización de las distancias recorridas.

3.2 KNN

El tema clave relacionado con KNN es que este modelo incluye el seteo de la variable K. El mejor seteo de *k* es el determinado empíricamente usando técnicas de validación, como pueden ser las validaciones cruzadas sobre el modelo. K=1 es una buena base para modelar.

Para clasificar como bueno el Cluster resultante los criterios a considerar para KNN serían que:

- el resultado no se deforme por la elección del valor de k
- el Cluster no varíe como consecuencia del orden de entrada
- la distribución de los datos en los Clusters sea equilibrada y "pareja"

Como se puede observar en las pruebas realizadas en los casos de testeo, este algoritmo es altamente sensible al orden de entrada de los datos. Este problema se explicó en el punto 2.2.3.3 y se puede ver un ejemplo de ello en el punto 2.4.2.1 en los casos 1 y 2 de testeo de KNN.

Además, los resultados están altamente vinculados al *k* elegido. Si el *k*=1, entonces el resultado es compacto y con una distribución equilibrada. Sin embargo, a medida que se incrementa el valor de *k* la cantidad de clusters hallados disminuye llegando al caso extremo de agrupar todos los elementos en un único cluster.

3.3 PAM

Al evaluar un Cluster realizado con PAM se pueden evaluar las siguientes características:

- el orden de entrada no varíe la conformación del Cluster
- que la distribución de los datos en los Clusters sea compacta en torno a los mediodes

De los resultados obtenidos, se desprende que PAM es independiente del orden de entrada. Por otra parte, la distribución final en torno a los mediodes es buena y en general se logran clusters compactos y equilibrados, aunque estos se pueden ver un poco deformados bajo restricciones de incompatibilidad y demanda.

Por otra parte, el hecho de que no trabaje con la media, sino con un elemento del dominio que se le aproxima (por eso es llamado mediode) es algo importante ya que permite su identificación gráfica si el cluster no posee demasiados elementos.

3.4 K-Means

Al momento de definir la bondad de un Cluster luego de haber aplicado K-Means los criterios a considerar serían:

- la cercanía de los puntos a los centros de los Clusters a los que pertenecen
- el Cluster no varíe como consecuencia del orden de entrada
- el resultado no dependa de la elección de las medias iniciales

De los resultados de testeo se deduce que K-Means tiene como punto importante a favor la característica de ser independiente del orden de las entradas. Al iterar, los elementos van cambiando de cluster hasta lograr una distribución compacta y homogénea, existiendo la limitación en nuestro problema de posibles saturaciones e incompatibilidades lo que puede provocar la deformación de los clusters.

En este algoritmo se puede ver que los centros de los clusters (medias) no son necesariamente elementos y puede que no tengan sentido en el dominio de aplicación, lo que le brinda un grado extra de libertad.

Los resultados obtenidos respecto al primer criterio planteado son buenos dado que se obtienen clusters compactos alrededor de las medias de los clusters, minimizando las distancias de los puntos a ellas.

Por último, si bien el resultado no depende de la elección de las medias iniciales en nuestro problema en particular las medias forzosamente están conformadas por los depósitos y los clientes se agrupan en torno a estos.

3.5 Comparaciones

Al comparar los algoritmos implementados a partir de los distintos testeos realizados, obtuvimos los siguientes resultados.

Con referencia al orden de entrada quedó establecido que mientras que KNN es altamente dependiente del mismo, K-Means y PAM no lo son.

Al estudiar los tiempos de ejecución, se puede observar que PAM es el mas lento de todos, mientras que KNN es el más rápido, logrando K-Means un tiempo intermedio en su ejecución. Lo cual se debe a los ordenes de los algoritmos:

Algoritmo	*Orden*
KNN	O(n)
K_means	O(tkmn), orden menor a n*n
Pam	O(k(n-k)*(n-k)), por iteración

A K-Means y PAM se les puede incorporar nuevos datos de entrada, sin variar demasiado los resultados obtenidos. En otras palabras, el Cluster resultante no es sensiblemente modificado al incorporar nueva información a procesar, mientras que KNN si puede presentar problemas al momento de considerar la escalabilidad dependiendo del valor de *k* elegido.

Si se considera el tipo de cluster obtenido, es apreciable que tanto PAM como K-Means llevan ventaja sobre KNN al ofrecer un resultado mas compacto (se mide a través de la dispersión de los elementos alrededor del centro del cluster y se confirma gráficamente). Lamentablemente KNN vuelve a depender del *k* seleccionado, así como del orden de entrada. Una mala combinación de estos dos factores deriva en un mal cluster.

Nuevamente PAM y K-Means tienen mejoras respecto a KNN siendo los dos primeros independientes de los seteos iniciales, mientras que KNN no lo es. En otro orden de cosas, ninguno de los algoritmos trabajados dejan elementos sin clasificar a menos de restricciones impuestas.

El siguiente cuadro ofrece un resumen de lo anterior:

	KNN	PAM	K-Means
orden de entrada	Dependencia	Independencia	Independencia
tiempos de ejecución relativos	Bajo	Alto	Medio
escalabilidad	No	Si	Si
clusters compactos	Variable	Si	Si
independencia seteos iniciales	No	Si	Si

Interesa comparar los algoritmos en términos de distancia recorrida luego de realizado un ruteo para cada depósito a los resultados obtenidos por los algoritmos implementados. Para ello, contamos con un software desarrollado por la docente Libertad Tansini al cual le pasamos nuestros resultados en un formato acordado y devuelve la distancia total recorrida luego de realizado el VRP sobre cada depósito.

Se incluye una muestra de los resultados obtenidos para los mismos casos del punto 2.4.2.4 donde se muestran las distancias recorridas al aplicar cada algoritmo en cada uno de los 5 casos planteados.

	KNN			KMEANS	PAM
Caso	***k=1***	***k=4***	***k=6***		
1	12406	14990		11502	11133
2	28281	46648		24931	29208
3	42316	92913		47367	38313
4	15577		81326	16072	15675
5	28126		121186	26719	25625

Se concluye que la distancia recorrida al aplicar KNN está altamente vinculado al valor de *k* elegido, mejorando al disminuir el mismo.

PAM y K-Means ofrecen resultados mas regulares y similares, no existiendo grandes diferencias entre ellos siendo en general las distancias por ellos recorridas menores a las de KNN a excepción del caso en que *k*=1.

Otro de los aspectos a considerar de los resultados obtenidos luego de aplicado el ruteo sobre el clustering, es la duración total del mismo. El siguiente cuadro refleja la relación existente entre la duración del ruteo para los diferentes algoritmos corridos (la velocidad del vehículo fue la misma en todos los casos).

	KNN			KMEANS	PAM
Caso	***k=1***	***k=4***	***k=6***		
1	83:21:00	82:32:00		75:34:00	80:49:00
2	285:42:00	436:25:00		183:08:00	299:30:00
3	283:30:00	573:52:00		402:47:00	279:58:00
4	134:13:00		646:38:00	127:57:00	127:09:00
5	205:36:00		871:39:00	197:15:00	189:27:00

Es claro que el comportamiento de los algoritmos en la duración del ruteo es de alguna manera similar si comparamos los largos totales (Ver cuadro anterior).

Nuevamente, KNN tiene un comportamiento aproximado al de K-Means y PAM pero solamente cuando k=1. Si lo que importa es el tiempo empleado en el recorrido del vehículo, entonces hay que prestar especial atención a estos resultados dado que varían según el algoritmo que se aplique y aunque la entrada incial sea la misma.

4 Conclusiones Generales

Se realizó un estudio particularmente interesante sobre algoritmos de Clustering aplicados a la asignación de clientes a depósitos en la primera fase de un problema de MDVRPTW, dado que no se encontraron antecedentes académicos o prácticos de utilización de estas técnicas en esta clase de problemas. Por lo tanto, este trabajo además de resolver el problema planteado por el usuario explora nuevas aplicaciones de los métodos de Clustering, abriendo un camino a futuros estudios e investigaciones.

En lo que se refiere a la aplicación de estos algoritmos al problema mencionado en el párrafo anterior, hay varios puntos interesantes a destacar. Si bien los algoritmos aplican el método de particionamiento para encontrar la solución solicitada, son muy distintos entre sí, y otorgan resultados diferentes.

Tal vez la única ventaja perceptible para KNN sea el orden de su ejecución mientras que algunas de sus desventajas más notorias son la dependencia del orden de entrada y la dependencia de la elección del valor de k. El problema presentado por el valor de *k* se puede sortear utilizando técnicas de validación sobre el mismo. Mientras que el problema del orden de entrada no es fácilmente solucionable ya que se vuelve inmanejable si hablamos de grandes volúmenes de datos a procesar.

PAM carece de los problemas anteriores, y es un método elaborado y complejo que brinda resultados compactos y bien definidos. Sin embargo, tiene una gran desventaja y es que puede llegar a ser lento en problemas de gran tamaño, debido a la complejidad del algoritmo.

K-Means al igual que PAM no presenta los problemas de KNN. Es un algoritmo sencillo y potente a la vez, lo que lo convierte en una herramienta útil a la hora de Clusterizar. La calidad de sus clusters es tan buena como los obtenidos por PAM y debido a sus ordenes, obtiene los resultados en un tiempo de ejecución menor.

Una vez realizados los VRP sobre los conjuntos Depósito-Clientes resultantes de la aplicación de los algoritmos –si se compara utilizando el criterio de minimización de distancias recorridas– queda claro una vez mas que KNN es menos eficiente que
K-Means y PAM.

Por lo anteriormente expuesto, no se sugiere la utilización de KNN para este tipo de problemas mientras que son ampliamente recomendables PAM y K-Means, mas allá de la advertencia de la performance de PAM cuando trabaja con grandes volúmenes de datos.

Finalmente, queda demostrado que es viable utilizar algoritmos de Clustering en la etapa de asignación de clientes a depósitos en un problema de MDVRPTW. Con esto, se agrega (a las tantas existentes) una nueva área de aplicación de las técnicas de Clustering.

5 Trabajos Futuros

- ✓ Dado que la utilización de algoritmos de Clustering aplicados a Problemas de Ruteo de Vehículos es en mayor medida innovador, una buena propuesta seria armar una biblioteca de algoritmos paramétricos y genéricos que otorguen una mayor flexibilidad al usuario.

- ✓ En particular, el software desarrollado se basa en un diseño que básicamente tiene 3 etapas: Pre-Procesamiento, Procesamiento y Post-Procesamiento, permitiendo aprovechar la posibilidad de cambiar la primera y la última etapa. De esta forma, permite cambiar los formatos de datos de ingreso y/o salida. De igual manera, es posible variar la función distancia que es la que mide la disimilitud entre elementos en un cluster.

- ✓ En el presente trabajo los datos de entrada son procesados mediante la ejecución de algoritmos determinísticos. Sería interesante observar la variación de resultados si se agrega aleatoriedad al procesamiento, o sea generando algoritmos probabilísticos.

- ✓ Al ofrecer una nueva opción para la resolución del MDVRPTW sería positivo realizar un estudio comparativo de los resultados obtenidos al resolver el problema por el método aplicado – asignación (clusterización) y luego ruteo - y los resultados brindados por el método convencional – asignación y ruteo al mismo tiempo -.

- ✓ Otras extensiones serían aplicar los algoritmos a Sistemas de Data Warehousing, visualización de los resultados a través de la conexión de los algoritmos con un Sistema de Información Geográfica, mantenimiento de Stock (a manera de ejemplo separar en categorías de elementos zafrales y no zafrales), etc.

6 Anexos

Anexo A. Referencias

A.1 Bibliografía

[I] Classification and Clustering
Editado por J. Van Ryzin
Recopilación de Seminarios - 1976
Academic Press
ISBN 0 12 714250 9
Primera Edición - 1977

[II] Classification and Related Methods of Data Analysis
Editado por H. H. Bock
Recopilación de Conferencias 1987
North-Holland
ISBN 0 444 70404 3
Primera Edición - 1988

[III] A Probabilistic Theory of Pattern Recognition
L. Devroye, L. Gyorfi, G. Lugosi
Springer
ISBN 0 387 94618 7
Primera Edición - 1996

[IV] Solving Data Mining Problems Through Pattern Recognition
R. Kennedy, Y. Lee, B. Van Roy, C. Reed, R. Lippmann
Prentice Hall
ISBN 0 13 095083 1
Primera Edición - 1998

[V] Fundamentals of Software Engineering
Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli
Prentice Hall
ISBN 0 13 820432 2
Primera Edición - 1991

[VI] El Lenguaje de Programación C
Brian W. Kernighan, Dennis M Ritchie
Prentice Hall
ISBN 968 880 205 0
Segunda Edición - 1991

[VII] El Lenguaje de Programación C++
Bjarne Stroustrup
Addison-Wesley
ISBN 0 201 60104 4
Segunda Edición - 1997

[VIII] Como programar en C/C++
Deitel, Deitel
Prentice Hall
ISBN 968 880 471 1
Segunda Edición - 1997

[IX] Inside Visual C++
David J. Kruglinski
Microsoft Press
ISBN 1 57231 565 2
Cuarta Edición - 1997

[X] Descubre Microsoft Visual C++ 6.0
Jon Bates, Tim Tompkins
Prentice Hall
ISBN 84 8322 077 6
Primera Edición – 1999

[XI] Edición Especial Microsoft Visual C++ 6.0
Kate Gregory
Prentice Hall
ISBN 84 8322 095 4
Segunda Edición - 1999

[XII] Professional MFC with Visual C++
Mike Blaszczak
Wrox
ISBN 1 861000 14 6
Primera Edición - 1997

[XIII] Programming Windows with MFC
Jeff Prosise
Microsoft Press
ISBN 1 57231 695 0
Segunda Edición - 1999

[XIV] Como Programar en Java
Deitel y Deitel
Pearson Education
ISBN 970 17 0044 9
Primera Edición - 1999

[XV] Ayuda en línea de Visual Studio
Microsoft Developer Network
MSDN Library Visual Studio 6.0a

A.2 Sitios Explorados

[1] http://web.mit.edu/cadet/www/Clustering/
Acceso: Febrero-00

[2] http://dw-institute.com/cases.htm
Acceso: Marzo-00

[3] http://edfu.lis.uiuc.edu/~class/ifcs/
Acceso: Marzo-00

[4] http://web.mit.edu/cadet/www/Clustering/clustering.html
Acceso: Marzo-00

[5] http://web.mit.edu/course/15/15.fall1999/15.821/www/1116.DOC
Acceso: Marzo-00

[6] http://web.mit.edu/dsm/Tutorial/
Acceso: Marzo-00

[7] http://www.dc.uba.ar/people/materias/dm/intro.html
Acceso: Marzo-00

[8] http://www.it/cenit.org.ar/links/linksBDatos
Acceso: Marzo-00

[9] http://www.kdnuggets.com/
Acceso: Marzo-00

[10] http://www.media.mit.edu/~chaiwei/old/icycs.doc
Acceso: Marzo-00

[11] http://www.pitt.edu/~csna/
Acceso: Marzo-00

[12] http://w3.mor.itesm.mx/~emorales/Cursos/KDD/node101.html
Acceso: Abril-00

[13] http://web.mit.edu/
Acceso: Abril-00

[14] http://web.mit.edu/aganti/www/ppt/
Acceso: Abril-00

[15] http://www.anthro.washington.edu/facultyinfo/BOOK
Acceso: Abril-00

[16] http://www.icaen.uiowa.edu/~ankusiak/process-model.html
Acceso: Abril-00

[17] http://www.media.mit.edu/~schoner/papers/DA
Acceso: Abril-00

[18] http://www-dbv.cs.uni-bonn.de/abstracts/hofmann.scaling.nips.95.html
Acceso: Abril-00

[19] http://www.spss.com
Acceso: Abril-00

[20] http://www.clustan.com
Acceso: Abril-00

[21] http://hera.berkeley.edu/Seminars/Old.97/September/970904.kornacker.html
Acceso: Mayo-00

[22] http://www.cs.sfu.ca/people/GradStudents/koperski/personal/research/survey.html/survey/node7.html
Acceso: Mayo-00

[23] http://www.cs.wisc.edu/~zhang/sigmodpaper.ps
Acceso: Mayo-00

[24] http://www.hpl.hp.com/techreports/1999/HPL-1999-119.html
Acceso: Mayo-00

[25] http://www.hpl.hp.com/techreports/1999/HPL-1999-124.html
Acceso: Mayo-00

[26] http://www.hpl.hp.com/techreports/2000/HPL-2000-6.html
Acceso: Mayo-00

[27] http://www.cse.ucsd.edu/users/mdailey/netlab-help/index.htm
Acceso: Setiembre-00

[28] http://www.cs.toronto.edu/~delve/methods/knn-class-1/home.html
Acceso: Setiembre-00

[29] http://www.cs.toronto.edu/~delve/
Acceso: Setiembre-00

[30] http://www.lfp.blm.tu-muenchen.de/pa/Software/knn.htm
Acceso: Setiembre-00

[31] http://www.pcigeomatics.com/cgi-bin/pcihlp/KNN
Acceso: Setiembre-00

[32] http://cns-web.bu.edu/~jmbrewer/KNN/KNN.html
Acceso: Setiembre –00

[33] http://www.psychstat.smsu.edu/multibook/mlt04m.html
Acceso: Octubre - 00

[34] http://cgm.cs.mcgill.ca/~soss/cs644/projects/siourbas/cluster.html
Acceso: Octubre - 00

[35] http://www.ece.nwu.edu/~harsha/Clustering/clus.html
Acceso: Octubre – 00

[36] http://www.ulib.org/webRoot/Books/National_Academy_Press_Books/discrim_analysis/discr001.htm
Acceso: Octubre-00

[37] http://db.cs.sfu.ca/GeoMiner/survey/html/node9.html
Acceso: Octubre-00

[38] http://www.cs.berkeley.edu/~gribble/summaries/database/birch.html
Acceso: Octubre-00

[39] http://www.math.tu-dresden.de/~schuetze/linuxlist/node8.html
Acceso: Octubre-00

[40] http://www.cs.ubc.ca/spider/dxu/course_project/504.html
Acceso: Octubre-00

[41] http://www.dsslab.com/resources/DMINFO.htm
Acceso: Octubre-00

[42] http://hissa.nist.gov/dads/
Acceso: Octubre-00

[43] http://www.cs.wmich.edu/~yang/teach/cs595/slides/Clustering
Acceso: Octubre-00

[44] http://www.cs.cmu.edu/~guyb/real-world/cluster/index.html
Acceso: Octubre-00

[45] http://www.epd.unil.ch/biocomputing/array/software/clustering/sld145.html
Acceso: Octubre-00

[46] http://ei.cs.vt/edu/~cs5604/f95/cs5604cnCL/CL-alg_details.html
Acceso: Octubre-00

[47] http://www.cne.gmu.edu/modules/dau/stat/clustgalgs/clustgalgs_frm.html
Acceso: Octubre-00

[48] http://www.dbs.informatik.uni-muenchen.de/Forschung/KDD/Clustering/
Acceso: Octubre-00

[49] http://www.cs.umn.edu/classes/Spring-2000/csci5980-dm/chap5-sc99tut/sld001.htm
Acceso: Octubre-00

[50] http://www.cs.sunysb.edu/~algorith/lectures-good/node17.html
Acceso: Octubre-00

[51] http://149.170.199.144/multivar/ca_alg.htm#average
Acceso: Octubre-00

[52] http://astro.u-strasbg.fr/~fmurtagh/mda-sw/online-sw.htm
Acceso: Octubre-00

[53] http://www.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter8/sld023.htm
Acceso: Octubre-00

[54] http://www.almaden.ibm.com/cs/quest/abstracts.html
Acceso: Octubre-00

[55] http://citeseer.nj.nec.com/bradley98refining.html
Acceso: Octubre-00

Anexo B. Estado del Arte: Estudio Histórico y Perspectivas Actuales

El objetivo del Clustering es ordenar los datos en grupos o clusters a través de un criterio de similitud (que puede definirse en varias dimensiones) de tal forma que el grado de asociación sea fuerte entre los miembros del mismo cluster y débil entre los miembros de clusters distintos. Cada cluster entonces, describe en termino de datos relevados, la clase a la cual sus miembros pertenecen. Ver [10] [14]

La actividad cotidiana lleva a utilizar Clustering con distintos objetivos. Entre ellos se destacan: la clarificación mental y la comunicación, descubrir nuevos campos de investigación, planear una estructura organizativa como una Universidad, planear la estructura de una máquina, formar conceptos en la vida cotidiana y para reconocimiento, investigaciones medicas o cualquier otra disciplina.

Si bien hay una comprensión intuitiva de que los miembros de un cluster están mas relacionados entre sí que con el resto de los individuos, las especificaciones de estas relaciones pueden ser distintas. Se pueden usar distintos parámetros para definir los clusters, como ejemplo sirven: la densidad, el volumen ocupado, la conectitud entre los miembros del cluster, etc. Ver [2] [3] [33]

Hay tres tipos principales de datos usados en los clusters:

- *Datos multivariados*: da los valores de muchas variables para muchos individuos.

- *Datos de proximidad*: se refiere a la similitud de objetos del mismo tipo.

- *Datos de Clustering*: un cluster subjetivo provisto por sujetos puede llegar a ser muy útil (Ej. Un sujeto que agrupa naipes de acuerdo a su criterio).

El propósito por el cual se lleva a cabo el Clustering, puede ser *específico o vago*. Los dos son legítimos y validos. Como ejemplo de propósitos específicos sirven: la agregación y clasificación de compañías a cierta industria, Clustering de casos médicos, clasificación de información, etc. En contrapartida, los propósitos vagos no guían a medidas y como ejemplos sirven los análisis exploratorios, comprensión de datos o simplemente buscar para “ver que hay” en los datos.Ver [7]

Podemos distinguir entre clusters *naturales y arbitrarios*. Llamamos naturales a aquellos que se determinan de alguna forma natural debido a los datos y los arbitrarios son aquellos que tienen algún elemento sustancialmente arbitrario en el proceso de asignación (Ej. La división de una ciudad por parte de la policía, corte electoral, bomberos, etc. Estos clusters son arbitrarios en el sentido que no tiene mucha diferencia cual calle se usa exactamente para separar dos distritos).

Hay técnicas que buscan *clusters naturales*, y tienen que ver con el criterio por el cual definimos los clusters naturales. En algunos casos, lo compacto del cluster es el criterio. En otros lo es la clara separación existente.

Existe la posibilidad de *solapamiento* entre clusters. Una posibilidad sería no permitir solapamiento, otra sería permitir un ligero solapamiento en la frontera de los clusters o se podría establecer una forma jerárquica donde un cluster contiene completamente a otro sujetos a ciertas reglas.

B.1 Clasificación de los Distintos Enfoques de Clustering

Se podrían tener en cuenta varias *facetas* posibles al intentar dividir en grupos los distintos métodos de Clustering, por ejemplo se podrían definir las siguientes:

B.1.1 Aglomerativo Versus Divisivo

En un método aglomerativo los clusters se construyen agregando items a ellos, mientras que en un método divisivo los clusters son sistemáticamente rotos para formar sub clusters. Ver [14] [44]

B.1.2 Jerárquico Versus No Jerárquico

En una técnica jerárquica los distintos clusters forman un árbol mientras que en un método no jerárquico todos los grupos son hermanos. Ver [1] [10] [14] [44]

B.1.3 Secuencial Versus Simultáneo

Si en la etapa de cálculo cada estado de gran tamaño tiene sus cálculos realizados de forma simultanea y simétrica podemos decir que los cálculos son simultáneos. La mayoría de los métodos de clusters son secuenciales.

B.1.4 Criterio Local Versus Global

Imaginemos los distintos items embebidos en algún espacio abstracto. Si la función de distancia varía de una parte a otra del espacio, entonces se puede describir al proceso de Clustering como un proceso local, de lo contrario es global.
Ver [45]

B.1.5 Soluciones Directas Versus Iterativas

En general es intuitivo si un cálculo debe ser planteado iterativamente.Ver [14]

B.1.6 Clustering Pesado Versus No Pesado

En una técnica en la cual los items son agregados secuencialmente al cluster, los distintos clusters o las distintas direcciones pueden ser valoradas de distintas maneras.

B.1.7 Estadísticos Versus Conceptual

Los métodos de Clustering estadísticos usan medidas de similitud para dividir los objetos, mientras que los métodos de Clustering conceptual lo hacen de acuerdo con los conceptos que llevan los objetos.
Ver [I] [II]

B.2 Áreas de Aplicación

Algunas ramas de la Ingeniería como Pattern Recognition, Inteligencia Artificial y Cibernética usan el concepto de Análisis de Cluster. Ejemplos típicos de cómo ha sido aplicado el Clustering incluyen: caracteres de la escritura a mano, muestras de discursos, clasificación de huellas, etc.

En las ciencias de la vida (Biología, Botánica, Zoología, Microbiología, etc.) los objetos de análisis son las formas de vida existentes: plantas, animales, insectos, etc. El Análisis de Cluster puede ir de desarrollar taxonomías completas hasta la clasificación de especies en subespecies.

También puede ser usado en Ciencias Políticas y de la Información. Las variadas técnicas de Clustering aplicadas a documentos incluye los votos en las elecciones, encuestas de mercado, de productos o de programas de ventas.
Ver [I] [II]

B.3 Métodos y Algoritmos

Los métodos de Clustering particionan un conjunto de datos en clusters tales que los objetos dentro del mismo cluster son más similares entre sí que objetos en otros clusters de acuerdo con algún criterio predefinido.

Si $X = \{X_1, X_2,..,X_n\}$ es un conjunto de *n* objetos, $X_i = [x_{i1}, x_{i2},..x_{im}]$ un objeto representado por m atributos, el objetivo de realizar Clustering sobre X es encontrar una partición que divida los objetos de X en *k* clusters. Como para un *n* dado la cantidad posible de particiones es definida pero extremadamente grande es impracticable investigar cada partición para encontrar la mejor de todas, una solución común es elegir un criterio de Clustering que nos guíe en la búsqueda de la partición. Un criterio de Clustering es llamado función de costo y se lo puede definir de varias maneras. También es esencial poder comparar dos Clusterings para poder distinguir buenos de malos. Un buen Clustering en general es obtenido cuando los puntos inter-cluster están fuertemente relacionados y aquellos intra-clusters están débilmente relacionados.
Ver [I] [II] [III]

Presentaremos la siguiente división de métodos:

- Clustering Jerárquico
- Métodos de Particionamiento
- Métodos Basados en Grafos
- Métodos Basados en Densidad
- Técnicas Estadísticas

B.3.1 Clustering Jerárquico

Los métodos de Clustering Jerárquico toman como entrada un conjunto de items y construyen un árbol que agrupa elementos con características similares, la similitud entre items puede darse de dos maneras:

- Directamente, como una matriz de distancia entre items e
- Indirectamente, donde los items se describen de acuerdo a algunas características (atributos) y la similitud entre dos items se define de acuerdo a las similitudes de las características de los mismos.

La salida de estos métodos es un árbol donde cada hoja es un ítem y los nodos intermedios representan grupos de elementos.

El árbol referido es conocido como dendograma. Dicho árbol utiliza los ruidos (outliers) que quedan aislados para el chequeo de errores y una de sus desventajas es que los nodos del árbol están igualmente espaciados, de tal forma que la distancia entre las ramas no indica similitud.Ver [10] [14] [33]

B.3.1.1 División de Clustering Jerárquico

Existen dos tipos de métodos jerárquicos:

Clustering Jerárquico Aglomerativo

Comienza formando un cluster por cada elemento y luego une estos clusters atómicos en clusters cada vez más grandes hasta que todos los objetos pertenecen a un único cluster.

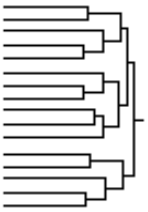
En este tipo de métodos no influye el orden en el cual se obtienen los elementos pero los algoritmos son susceptibles a la toma de decisiones locales.

La interpretación del dendograma es la siguiente:

- Cuanto antes se unan dos ramas del árbol es porque más similares son entre sí. El largo del camino de las hojas al nodo donde se realiza la unión, es la medida de *disimilitud.*

- El largo del camino de la raíz del diagrama al nodo de unión es la medida de *similitud.*

- El largo de la línea entre la unión de dos o mas caminos y la unión de estos caminos con otros, es la medida de su *coherencia* (que tan exclusivamente similares son).



dendograma

La mayoría de los métodos jerárquicos pertenecen a esta categoría, diferenciándose entre ellos por la definición de similitud entre clusters.

El algoritmo aglomerativo consiste en:

➢ Encontrar los elementos más cercanos en el conjunto de datos y construir un nodo uniendo estos dos items.
➢ Computar la medida de similitud entre este nodo y los otros elementos,
➢ Recursivamente unir items y/o nodos (conjunto de items ya unidos) hasta que un nodo incluya a todos los elementos, dicho nodo será la raíz del árbol.

Clustering Jerárquico Divisivo

Comienza poniendo todos los objetos en un único cluster y luego los subdivide en partes más pequeñas. En general, no hay algoritmos de este tipo y rara vez son aplicados ya que es muy difícil encontrar reglas efectivas que permitan dividir en casos y es costosa su implementación.

Los métodos anteriores permiten partir el conjunto de entrada en distintas cantidades de clusters dependiendo del nivel de abstracción en el cual el usuario está interesado.

La manera más común de partir un árbol construido por un algoritmo de Clustering jerárquico es cortar horizontalmente el árbol en un determinado nivel. Esta partición asegura que dos elementos de la misma clase están más cerca uno de otro que dos items de diferentes clases. Sin embargo, el usuario podría querer más detalles en algunas clases que otras, por lo tanto, en este caso, sería conveniente tomar clusters que se encuentren en distintos niveles del árbol.

Ventajas:

- ✓ La ventaja de los métodos de Clustering jerárquico, que utilizan cualquier medida arbitraria entre pares de elementos, es su simplicidad.

Desventajas:

- ✓ Cuando el tamaño de los datos es grande, el número de distancias entre pares crece enormemente, siendo su complejidad al menos $O(N^2)$.
- ✓ Son altamente dependientes de la medida de distancia elegida; diferentes medidas de distancia pueden producir diferentes configuraciones de clusters.
- ✓ No permiten deshacer lo que se hizo anteriormente, de esta manera, al cometer un error, el mismo no puede ser corregido posteriormente en la jerarquía (los errores se propagan de los niveles inferiores a los superiores).

B.3.1.2 Single, Complete Y Average Linkage

Los algoritmos jerárquicos más conocidos y usados son:

- single linkage (nearest neighbour)
- complete linkage (farthest neighbour)
- average linkage

En *single linkage* se define la distancia entre dos clusters como la mínima distancia entre un par de elementos, uno de cada cluster; la regla que se establece para las sucesivas combinaciones (Merge) dice que los clusters a unirse son aquellos que poseen la menor distancia. Ver [33] [51]

En *complete linkage*, la distancia entre clusters es la máxima distancia entre dos elementos, uno de cada cluster y luego se unen los clusters que poseen menor distancia.Ver [14] [33] [51]

El método *average linkage* opera de forma similar a los anteriores pero define la distancia entre clusters como el promedio de las distancias entre pares de items, uno de cada cluster. Ver [33] [51]

Single linkage es propenso a producir clusters largos y 'finos' que son de poco interés. Contrariamente, complete linkage tiende a producir clusters pequeños y compactos. Average linkage es un método intermedio entre los extremos de los dos anteriores.

Por otra parte, los clusters generados por single linkage tienen un número de caracterizaciones equivalentes que lo hacen atractivo para su estudio teórico. Por ejemplo, si dividimos los puntos en dos clusters tal que la mínima distancia

entre los clusters es tan grande como sea posible y luego se continúa dividiendo de la misma forma, se obtienen los clusters producidos por single linkage.

B.3.1.3 Método Ward

Este método se comporta de una manera similar a los árboles de decisión, combina en cada etapa los dos clusters que minimizan la función de error cuadrática, o la suma euclidiana de cuadrados, E:

$$E = \sum_{k} \sum_{i \varepsilon k} \sum_{j} (x_{ij} - \mu_{kj})^2$$

donde para una observación de clasificación dada, i pertenece al cluster k, x_{ij} es el valor de la variable j para la observación i y μ_{kj} es la media de la variable j en el cluster k.

Algoritmo:

1. Calcular la matriz de distancias euclidianas cuadradas para todos los pares de observaciones q y p:

$$d_{pq}^2 = \sum_{j} (x_{pj} - x_{qj})^2$$

2. Combinar los dos casos p y q cuya unión resulta en el mínimo E. Esto van a ser los dos casos iniciales para los cuales d_{pq}^2 es mínimo.

3. Transformar las distancias d_{pk}^2 a E_{pk}, el error cuadrático para la unión del cluster generado por p y q, con cualquier otro cluster k.

4. Repetir el paso 3 en cada paso combinando los dos casos de clusters cuya unión resulta en el mínimo E.

5. Finalizar cuando todos los casos estén agrupados en 1 cluster.

La matriz de distancias es simétrica. Es evidente que hay normalmente n-1 pasos 3).
Ver [51]

B.3.1.4 Optimización y Simplificación Iterativa

La idea básica de este método es partir el espacio de búsqueda en tres fases:

1- Genera una partición en clusters inicial con un bajo costo computacional.
2- Un procedimiento de optimización iterativo que continúa buscando Clusterings mejorados.
3- Una simplificación del Clustering generado de manera que se facilite su análisis.

Para la construcción del Clustering inicial, se propone:

1- Una función o medida de calidad que guía la búsqueda de clusters
2- Un método de Clustering.

Simplificación de Clusterings Jerárquicos

Un Clustering jerárquico puede crecer hasta una altura arbitraria. Idealmente, si existe una estructura en los datos los niveles superiores deben reflejarla, sin embargo, los niveles inferiores pueden no aportar ningún dato significativo. En este caso se utiliza un proceso de simplificación como fase final en la búsqueda del Clustering jerárquico de manera de facilitar la tarea.

Identificación de las variables frontera por muestreo

Dado un Clustering jerárquico y un conjunto de observaciones para validación, el conjunto de validación se usa para identificar las variables frontera para predecir cada variable.

Para cada variable A_i, los objetos del conjunto de validación se clasifican de acuerdo al Clustering jerárquico con el valor de A_i enmascarado para los propósitos de clasificación; durante la clasificación en cada cluster evaluado se compara el valor de A_i con el valor más probable de A_i en el cluster; si son iguales significa que se habría predicho correctamente la clasificación.

En cada cluster se mantiene un contador que guarda la cantidad de predicciones correctas. Una vez que se han clasificado todas las variables de todas las observaciones del conjunto de validación, se identifica la frontera de cada variable tal que maximice el número de predicciones correctas de dicha variable. Este procedimiento asegura que el número de predicciones correctas en un nodo que se encuentra en la frontera de la variable es mayor o igual que la suma de las predicciones de sus descendientes.

Una vez identificadas las fronteras de cada variable, se puede podar un cluster si el mismo cae fuera de las fronteras de todas las variables. Por lo tanto, en un Clustering simplificado una hoja es un cluster que está en la frontera de al menos una variable y ninguno de sus descendientes está en la frontera de cualquier otra variable, en el Clustering original.

Esta simplificación reduce el tamaño del Clustering y no disminuye su precisión y se podría realizar una poda más restrictiva si se focalizara la atención en pocas variables.

Una variación a esta estrategia es construir el Clustering con todos los datos disponibles, luego remover cada observación (y ajustar todos los valores de las variables desde la raíz hasta donde estaba la observación), usarla para predecir cada una de sus variables y luego restituirla a su lugar original.

Criterio de validación de los clusters

Existen dos formas de validar los Clusters:

- Interna: Evalúa la estrategia de control que realiza la búsqueda de clusters midiendo la calidad de los mismos según la función elegida.
- Externa: Determina la utilidad del cluster descubierto en forma relativa a una tarea.

B.3.1.5 BIRCH

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

Conceptos Generales:

Se define un *cluster* con *N* objetos como un conjunto de objetos: $\{\vec{X}_i\}$ con i = 1,..,*N* donde cada $\vec{X}_i$ tiene d dimensiones.

Para cada cluster se define su *centroide* $\vec{X}o$ como:

$$\vec{X}o = \frac{\sum_{t=1}^{N} \vec{X}_t}{N}$$

Para cada cluster se define su *radio* R como:

$$R = \left(\frac{\sum_{t=1}^{N} \left(\vec{X}_t - \vec{X}o\right)^2}{N} \right)^{1/2}$$

Para cada cluster se define su *diámetro* D como:

$$D = \left(\frac{\sum_{t=1}^{N} \sum_{j=1}^{N} (\vec{X}_t - \vec{X}_j)^2}{N(N-1)} \right)^{1/2}$$

Entrada:

- k: La cantidad de clusters deseados.
- Todas las medidas de similitud entre todos los objetos.
- Un conjunto de datos de cardinalidad N.

Aspectos considerados:

- La cantidad de memoria disponible es limitada.
- Se cuenta con grandes volúmenes de datos.
- Se desarrolló para atributos numéricos.
- Los ruidos son tratados en forma adecuada.

Conceptos Básicos del algoritmo:

Un Clustering *feature* (CF) es un resumen de la información que describe a un cluster. Es un vector definido de la siguiente manera: $CF = (N, \vec{LS}, SS)$, donde N es la cantidad de objetos del cluster, $\vec{LS}$ es la suma lineal de los N puntos ($\sum_{i=1}^{N} \vec{X}_i$) y SS es la suma de los cuadrados de los puntos ($\sum_{i=1}^{N} \vec{X}_i^{\,2}$).

Dados $CF_1 = (N_1, \vec{LS}_1, SS_1)$ y $CF_2 = (N_2, \vec{LS}_2, SS_2)$ los vectores CF de 2 clusters disjuntos, el cluster formado por la unión de los mismos tiene un CF de la siguiente forma: $CF_1 + CF_2 = (N_1 + N_2, \vec{LS}_1 + \vec{LS}_2, SS_1 + SS_2)$. Además teniendo el vector CF de cada cluster se pueden calcular los centroides $\vec{Xo}$, radios (R), diámetros (D), las distancias Euclidiana (D1) y de Manhattan (D2), distancias promedio entre clusters (D3) y dentro de los clusters (D4).

Árbol CF es un árbol balanceado con 2 parámetros:

B (cantidad máxima de hijos)
y
T (máximo diámetro de los subclusters en las hojas del árbol).

Cada nodo interno tiene a lo sumo B hijos y representa al cluster constituido por la unión de todos sus hijos. Cada nodo es representado por su CF y los punteros a sus hijos. Cada hoja representa al cluster formado por la suma de todos los CF asociados a esa hoja, pero el diámetro del mismo debe ser menor que T.

Algoritmo:

Primero se escanean los datos y se construye el árbol CF.

Éste se construye dinámicamente a medida que se insertan objetos, por lo tanto este método es incremental. Un punto es *insertado* en la hoja cuyo cluster es el más cercano.

Se comienza desde la raíz y desciende recursivamente sobre el árbol CF buscando el nodo hijo más cercano de acuerdo a una distancia elegida (D1, D2, D3 o D4).

Si luego de la inserción, el diámetro del cluster representado por la hoja es mayor a T, la hoja y posiblemente otros nodos son particionados. Luego de la inserción, la información del mismo es pasada hasta llegar a la raíz. Una vez construido el árbol CF, cada hoja representa un *cluster*.

Se puede cambiar el tamaño del árbol variando T. Para ello hay que reconstruir el árbol, partiendo de las hojas del viejo. Este proceso se puede hacer sin la necesidad de leer todos los puntos. Se han propuesto algunas heurísticas para tratar los ruidos y otras para mejorar la calidad del árbol, realizando escaneos adicionales de los datos. Dentro de esas técnicas incluyen la adaptación de un algoritmo jerárquico aglomerativo para reacomodar los objetos de la hojas y mejorar la calidad del Clustering, este algoritmo utiliza la distancia D3 (distancia dentro de los clusters). Ver [21]

La complejidad de BIRCH es proporcionalmente lineal al número de objetos.

Desventajas:

- ✓ El algoritmo puede llegar a trabajar mal cuando los clusters no son esféricos, porque usa el concepto de radio o diámetro para controlar los bordes del cluster.
- ✓ Sólo puede manejar datos numéricos.
- ✓ Los resultados son sensibles al orden de los registros de datos.

Ver [35] [38] [40]

B.3.2 Métodos de Particionamiento

Los algoritmos de particionamiento construyen una partición de la base de datos D de n objetos en un conjunto de *k* clusters, de forma tal que los objetos dentro de un cluster son más similares entre sí que al compararlos con los objetos de otros clusters. *k* es un parámetro de entrada para estos algoritmos, o sea, se requiere un poco de conocimiento del dominio.

El algoritmo de particionamiento típicamente comienza con una partición inicial de D y luego usa una estrategia de control iterativa para optimizar una función objetivo.

Cada cluster es representado por un centro de gravedad (en los algoritmos k-means) o por uno de los objetos del cluster ubicado cerca del centro (en los algoritmos k-medoid).

Los algoritmos de particionamiento usan un procedimiento de 2 pasos:

- Determinar *k* objetos representativos que minimizan la función objetivo.
- Asignar cada objeto al cluster cuyo elemento representativo es el más cercano.

Algunas de las técnicas de particionamiento son las siguientes:

B.3.2.1 Enumeración Exhaustiva

Examina las k^n / k! formas de particionar el conjunto de n datos en *k* clusters. No es aplicable, salvo para valores de *k* y n pequeños.

B.3.2.2 Método K_Means

El algoritmo de k-means está construido sobre cuatro operaciones básicas:

- ✓ Selección de *k* medias iniciales para los *k* clusters.
- ✓ Cálculo de la disimilitud entre un objeto y la media del cluster.
- ✓ Asignación de un objeto a un cluster cuya media es la más cercana al objeto.
- ✓ Recálculo de la media de un cluster de los objetos alocados a él como para que la disimilitud intra-cluster sea minimizada.

Excepto por la primera operación, las otras tres operaciones son realizadas repetidamente en el algoritmo hasta que el algoritmo converge.

La esencia del algoritmo es minimizar la función de costo:

$$E = \sum_{c=1}^{k} \sum_{t=1}^{n} y_{t,c} d(x_t, q_c),$$

donde n es la cantidad de objetos en un conjunto de datos X, $x_i \in X$, q_t es la media del cluster t, $y_{t,c}$ es un elemento de una matriz de partición Y_{n*c}, d es una medida de disimilitud (generalmente definida como el cuadrado de la distancia Euclidiana).

La matriz *Y* tiene las siguientes propiedades:

$$0 \leq y_{tc} \leq 1$$

$$\sum_{t=1}^{k} y_{tc} = 1$$

Si $y_{tc} \in \{0,1\}$ se dice que *Y* es una partición *dura*; en caso contrario se dice partición *difusa*.

Sea $X = \{x_1, x_2, .., x_n\}$ un conjunto de n vectores en R^d, representando los datos.

Para un entero $c \geq 2$, un Clustering "duro" de X en c clusters consiste en c subconjuntos disjuntos de X, S_1, S_2, ...S_c, cuya unión es X. El algoritmo debe seleccionar los subconjuntos S_1, S_2, ...S_c tal que minimicen para $v_i \in R^d$ la media de los vectores de datos en S^i y $|x| = \sqrt{\sum_i x_i^2}$ la norma Euclidiana de un vector

$$X: \quad \sum_i \sum_{x \in S_i} |x - v_i|^2$$

En otras palabras, elige los clusters de tal manera de minimizar las distancias de los puntos en los clusters al "centro" del cluster. La asunción de este algoritmo es que la similitud entre objetos es medida por la distancia entre sus correspondientes vectores de datos; por eso, si las distancias a la media del cluster son chicas, los puntos en el cluster son también cercanos el uno con el otro.

Una forma equivalente de definir clusters es usando funciones de pertenencia, esto es definir:

u_i: X $\rightarrow$ {0, 1} con $u_i(x)=1$ si x $\in S_i$ y $u_i(x)=0$ si x$\notin$ S_i para cada i = 1, ..., c.

Un *Clustering difuso* es una generalización de este punto de vista. Pues un Clustering difuso de X en c clusters consiste en funciones u_1, ...u_c donde u_i: X $\rightarrow$ [0, 1], $\sum_{i=1}^{c} u_i(x_k)=1$ y $\sum_{k=1}^{n} u_i(x_k)>0$ para todo x$\in$X. El valor de una función de pertenencia difusa puede ser cualquier número entre 0 y 1, y viene a ser una caracterización matemática de un conjunto que puede no estar definido precisamente. Es por eso que los "clusters" de un Clustering difuso son las funciones de pertenencia en sí mismas; pues indican la estructura de los datos en el siguiente sentido: si dos puntos tienen grados de pertenencia cercanos el uno con el otro para la misma función de pertenencia entonces pueden ser considerados similares entre sí.

La condición $\sum_i u_i(x_k)=1$ corresponde a la pertenencia de cada punto en X.

El algoritmo de Clustering difuso elige para c ≥ 2 y *m* cualquier número real mayor a 1, u_i: X $\rightarrow$ [0, 1] de tal manera que $\sum_i u_i=1$ y $v_i \in R^d$ para i = 1, .., c para minimizar la función:

$$\sum_i \sum_{x \in S_i} (u_{ik})^m \left| x - v_i \right|^2$$

Con u_{ik} el valor de la i-ésima función de pertenencia en el k-ésimo punto x_k; v_1,.., v_c los centros de los clusters, y *m* un peso elegido de manera tal de reducir el ruido (cuanto más grande es el valor de m, menos contribuyen a la función de costo aquellos puntos cuyo grado de pertenencia es bajo permitiendo así no influir en la determinación de los centros y las funciones de pertenencia).

El algoritmo está basado en el siguiente conjunto de ecuaciones, que son las condiciones necesarias para u_1, ...u_c y v_1,.., v_c para producir un mínimo local:

Cálculo de la Media:

$$v_i = \sum_k x_k \frac{(u_{ik})^m}{\sum_k (u_{ik})^m}$$

Cálculo del grado de pertenencia en un cluster difuso:

$$u_{ik} = \frac{\frac{1}{\sqrt[m-1]{|x_k - v_i|^2}}}{\sum_j \left(\frac{1}{\sqrt[m-1]{|x_k - v_j|^2}} \right)}$$

Si estas ecuaciones pudieran ser resueltas en forma cerrada, la solución proveería directamente el Clustering difuso. Pero, como no se ha hallado ninguna solución de forma cerrada, son la base para un procedimiento iterativo que converge en un mínimo local de la función de costo.

Algoritmo:

0) Se elige un valor de *c* y *m* y se ejecuta el siguiente algoritmo:

1) Elegir una aproximación inicial para los grados de pertenencia.
2) Usando los grados de pertenencia y la ecuación que define v_i calcular los centros de los clusters.
3) Usando los centros de clusters calculados y la ecuación que define u_{ik} recalcular los grados de pertenencia.
4) Repetir los pasos 2 y 3 hasta que los grados de pertenencia o los centro de clusters no difieran en sus valores en más de algún valor predeterminado.

Se puede utilizar una función de *validez* que asigne a la salida un número que mida la calidad o validez del Clustering generado por el algoritmo. Evaluando la función a la salida para varias opciones de c y m, se espera determinar los valores de estos parámetros que mejor identifican la estructura de los datos. La calidad de un Clustering está dada por cuán cerca están los puntos de los centros de los clusters a los cuales pertenecen.

Existen algunas variantes del algoritmo de k-means que difieren en la selección de las *k* medias iniciales, cálculos de disimilitudes y estrategias para calcular las medias de los clusters.

Ventajas:

- ✓ Es eficiente en el procesamiento de grandes conjuntos de datos.
- ✓ La complejidad es O(t k m n), donde m es la cantidad de atributos, *k* la cantidad de clusters, t la cantidad de iteraciones sobre todo el conjunto de datos. Usualmente k, m, t << n (por lo tanto es de orden mucho menor a N^2)

Desventajas:

- ✓ Comparado con otras técnicas de particionamiento, este método es bastante sensible a los ruidos (puntos de datos que se hallan muy alejados del resto de los datos).
- ✓ Los centros de los clusters no son necesariamente objetos de la base de datos y puede que no tengan sentido en el dominio de aplicación.
- ✓ Frecuentemente termina en un óptimo local.
- ✓ Trabaja sólo con valores numéricos ya que minimiza una función de costo calculando las medias de los clusters
- ✓ Los clusters tienen forma convexa. Por lo tanto es difícil usar k-means para encontrar clusters con formas no convexas.

Comparado con otros métodos de Clustering el k-means es muy eficiente con grandes conjuntos de datos, pero su uso está limitado a datos numéricos ya que estos algoritmos minimizan una función de costo calculando la media del cluster. Las aplicaciones frecuentemente cuentan con datos categóricos y la conversión de los datos categóricos a valores numéricos no necesariamente produce resultados significativos en el caso que los dominios categóricos no sean ordenados. El algoritmo de k-prototypes remueve esta limitación y extiende el algoritmo de k-means a dominios categóricos preservando su eficiencia. Ver [24] [25] [35] [IV]

B.3.2.3 K-Prototypes

El k-prototypes, permite hacer Clustering de conjuntos de datos con valores mixtos, numéricos y categóricos (un dominio es definido como *categórico* si es finito y sin orden, por ejemplo para cualquier a, b ya sea a = b o a ≠ b). Como el Clustering se realiza basándose en la comparación de los objetos con *k* prototipos en vez de *k* medias (*means*), se lo llama k-prototypes.

Se define una medida de disimilitud que tiene en cuenta tanto los atributos numéricos como los categóricos como $s_n + p * s_c$, donde s_n es la medida de disimilitud sobre los datos numéricos (definida como el cuadrado de la distancia Euclidiana), s_c es la medida de disimilitud sobre los atributos categóricos entre dos objetos (definida como la cantidad de atributos no coincidentes entre objetos y los prototipos de los clusters), y p es un peso para balancear las dos partes para evitar favorecer alguno de los tipos de atributos de un objeto.

El proceso del algoritmo de k-prototypes es similar al del k-means excepto por un nuevo método de actualización de los valores de los atributos categóricos de los prototipos de los clusters que permite maximizar la similitud intra-cluster.

Algoritmo:

El algoritmo consiste en los siguientes pasos:

1) Seleccionar *k* prototipos iniciales de un conjunto de datos X
2) Asignar cada objeto en X a un cluster cuyo prototipo sea el más cercano de acuerdo con E_b y actualizar el prototipo del cluster después de cada asignación.
3) Después de haber asignado todos los objetos a un cluster retestear la similitud con los prototipos actuales. Si un objeto es tal que su prototipo más cercano pertenece a otro cluster al cual está asignado, reasignarlo a ese cluster y actualizar los prototipos de ambos clusters
4) Repetir el paso 3 hasta que ningún objeto cambie de cluster después de un ciclo completo de test sobre X.

Los prototipos numéricos se actualizan al promedio de los valores numéricos en el cluster correspondiente t. Los prototipos categóricos se actualizan al c_j tal que la probabilidad de que dicho valor ocurra en el cluster t sea menor o igual a la probabilidad de que valor q_{tj} ocurra en el cluster, para todos los q_{tj} que pertenecen al conjunto de valores en el atributo categórico j. Esto se debe a que dicho valor es el que minimiza s_{nt}.

Ventajas:

- ✓ Preserva la eficiencia y escalabilidad del algoritmo de k-means.
- ✓ Permite trabajar con datos categóricos.

Desventajas:

- ✓ Generalmente termina en un óptimo local, dependiente de los prototipos iniciales y del orden de los objetos.
- ✓ Sensible al ruido.
- ✓ Sólo detecta clusters de forma convexa.
- ✓ Necesita conocimiento del dominio para especificar el peso p.
- ✓ El mayor problema de usar este algoritmo (en especial en comparación con el k-means) es el de elegir un peso apropiado. La elección de p_c es dependiente de la distribución de los atributos numéricos, pues está relacionado con σ_c (el promedio de la desviación standard de los valores numéricos en el cluster c). Pero como σ_c no es conocido antes de realizar el Clustering, entonces se puede usar σ, (el promedio de la desviación standard sobre todos los valores numéricos) para todos los σ_c.

B.3.2.4 Método K-Modes

El k-modes es una simplificación del algoritmo de k-prototypes que sólo tiene en cuenta los atributos categóricos. Por lo tanto, como desaparece s_n, el peso p no es necesario. Si existen atributos numéricos dentro de un conjunto de datos se los categoriza. La mayor ventaja de este algoritmo es que es escalable a conjuntos de datos muy grandes.

Otro método para usar el algoritmo de k-means con datos categóricos necesita convertir atributos categóricos en atributos binarios (usando 0 y 1 para representar una categoría presente o ausente) y trata los atributos binarios como atributos numéricos en el algoritmo de k-means. Este método requiere el manejo de gran cantidad de atributos binarios ya que los conjuntos de datos en general tienen atributos categóricos con miles de categorías. Esto inevitablemente incrementa los costos del algoritmo de k-means. Por otra parte la media de un cluster dado por valores reales entre 0 y 1 no indica las características de los clusters. Comparativamente el algoritmo de k-modes trabaja directo sobre los datos categóricos y produce medias que describen los clusters, por lo tanto son muy útiles para que el usuario pueda interpretar los resultados del Clustering.

El algoritmo k-modes es una versión simplificada del algoritmo k-prototypes, en él se han hecho tres modificaciones principales:

- Se usaron distintas medidas de disimilitud.
- Se reemplazaron los *k* prototipos con *k* modes.
- Se usó un método basado en frecuencias para actualizar los modes.

Medidas de disimilitud:

Si X e Y son dos objetos categóricos descriptos por m atributos categóricos entonces la medida de disimilitud entre ellos puede ser definido, al igual que en k-prototypes, como la cantidad de diferencias en los correspondientes categorías de atributos en los dos objetos. Cuanto menor es la cantidad de diferencias, más similares son los dos objetos.

Algoritmo:

El algoritmo consiste en los siguientes pasos:

1) Seleccionar *k* modes iniciales, uno para cada cluster.
2) Asignar un objeto a un cluster cuyo mode es el más cercano de acuerdo con *d*.
3) Actualizar el mode del cluster después de cada asignación
4) Después de haber alocado todos los objetos a clusters, retestear la disimilitud de los objetos con los modes actuales. Si se descubre que el mode más cercano a un objeto pertenece a otro cluster, reasignar el objeto en ese cluster y actualizar los modes de ambos clusters.
5) Repetir paso 3 hasta que ningún objeto cambie de cluster después de un ciclo completo de test sobre todo el conjunto de datos.

Al igual que el algoritmo de k-means el k-modes produce soluciones de óptimos locales que son dependientes de los modes iniciales y el orden de los objetos en el conjunto de datos.

Ventajas:

- ✓ Preserva la eficiencia y escalabilidad del algoritmo de k-means y k-prototypes.
- ✓ Permite trabajar con datos categóricos.
- ✓ No necesita conocimiento del dominio para especificar el peso p, necesario en el algoritmo k-prototypes.

Desventajas:

- ✓ Generalmente termina en un óptimo local, dependiente de los modes iniciales y del orden de los objetos
- ✓ Sensible al ruido
- ✓ Sólo detecta clusters de forma convexa

Ver [41]

B.3.2.5 Método KNN (K Nearest Neighbours)

La arquitectura de los *k* vecinos mas cercanos (KNN) es una de las arquitecturas mas simple y poderosa. La idea básica de KNN es clara:

Para un muestreo, todos los pares entrada-salida en el conjunto de muestreo son almacenados en una base de datos. Cuando se necesita una estimación o una clasificación para una nueva entrada, la respuesta se basa en los *k* puntos mas cercanos encontrados en la base de datos.

El tema clave relacionado en este modelo incluye el seteo de la variable *k* y el tipo de "distancia" a usar. El mejor seteo de *k* es el determinado empíricamente usando técnicas de validación, como pueden ser las validaciones cruzadas sobre el modelo.

La distancia métrica es lo que le da significado al "vecino mas cercano". La distancia mas popular es la medida Euclidiana, aunque también puede ser usada la distancia Manhattan u otra.

Euclidiana:

$$Dist(X,Y) = \sqrt{\sum_{i=1}^{D}(x_i - y_i)^2}$$

donde Dist (X,Y) es la distancia entre dos vectores D-dimensionales X e Y.

Algoritmo:

Muestreo:
Guardar todos los pares entrada/salida en el conjunto de muestreo.

Testeo:
Para cada patrón en el conjunto de muestreo:

1. Busco los *k* patrones mas cercanos al patrón de entrada usando la medida de distancia Euclidiana.
2. Para clasificar, computamos la confianza para cada clase. Como Ci/K, donde Ci es el número de patrones entre los *k* patrones mas cercanos pertenecientes a la clase i. La clasificación para los patrones de entrada es la clase con la confianza mas alta.
3. Para estimar, el valor de salida se basa en el promedio de los valores de salida de los *k* patrones mas cercanos.

Parámetros de muestreo

Los parámetros mas comunes de muestreo para KNN incluyen:

Numero de vecinos mas cercanos (K)

Compresión de la entrada:

Podríamos necesitar, comprimir las muestras de datos como un paso de preproceso previo a la clasificación. Obviamente, esto empeorara la performance.

Otros parámetros de muestreo incluyen:

Métrica de la distancia
Como combinar los *k* vecinos para la estimación

Seteos típicos de KNN

Algunos rangos razonables de parámetros del modelo KNN:

- ✓ #de vecinos mas cercanos

Debería estar basado en validaciones cruzadas sobre un numero de seteos sobre K. K=1 es una buena base para modelar, una buena regla es que *k* sea menor que la raíz cuadrada del número total de muestras.

- ✓ Compresión de la entrada

Aunque lo habitual es comprimir la entrada, si no hay problemas de memoria podría optarse por usar el algoritmo KNN sin ninguna compresión.

Ventajas:

- ✓ Es simple y poderoso.

Desventajas:

- ✓ Se comporta mejor en problemas con pocas dimensiones.
- ✓ Podría ser necesario comprimir las entradas para su procesamiento.

Ver [28] [31] [32] [IV]

B.3.2.6 Métodos K_Medoid

El objetivo de estos métodos es encontrar *objetos representativos* llamados *mediodes*. Para ello sólo es necesario una definición de distancia entre cualquier par de objetos.

Estos métodos particionan un conjunto de objetos en *k* clusters utilizando alguna *medida de similitud o distancia* entre los objetos.

Constan de dos partes:

- ❖ Construcción:

Se eligen *objetos representativos (o mediodes)* sucesivos, con el propósito de obtener la menor distancia promedio posible entre los objetos de la base de datos y el objeto representativo más similar. La idea es que, cada mediode sea el elemento ubicado más centralmente en cada cluster y que el resto de los objetos pertenecerán al cluster cuyo mediode sea el más cercano.

- ❖ Intercambio:

Se toman distintos objetos representativos, para quedarse con aquellos que disminuyen la distancia promedio obtenida.

La cantidad de clusters, k, es un parámetro de entrada; y se genera como salida *k* clusters con sus respectivos objetos representativos (o mediodes).

Ventajas:

- ✓ Comparado con otras técnicas de particionamiento, estos métodos son muy robusto al tratar los ruidos.
- ✓ Los clusters encontrados no dependen del orden de examinación de los objetos.
- ✓ Puede manejar grandes volúmenes de datos en forma eficiente.

Ver [IV]

PAM

PAM (Partitioning Around Medoids)

Entrada:

- K, la cantidad de clusters.
- Todas las medidas de similitud entre todos los objetos.

Algoritmo:

Se selecciona arbitrariamente *k* objetos como mediodes. Luego de ello, el algoritmo trata de obtener una mejor selección de mediodes analizando todos los posibles pares de objetos, tal que uno sea mediode y el otro no. Entonces en cada iteración reemplaza un mediode por otro objeto que no lo es, y se calcula la *calidad del Clustering* (es la calidad combinada de los mediodes elegidos) obtenido. Si ésta es mejor que la obtenida hasta el momento se mantiene el intercambio de objetos para la próxima iteración, en caso contrario se vuelve a los mediodes anteriores. La calidad se mide como el promedio de la distancia entre un objeto y el mediode de su cluster.

Para calcular el efecto del intercambio entre O_i (mediode) y O_h (no mediode), se calcula el costo C_{jih} para todo objeto, O_j, que no es mediode. El costo total del intercambio (TC_{ih}) es:

$$TC_{ih} = \sum_j C_{jih}$$

Para calcular C_{jih} (es la resta entre la distancia de O_j a su nuevo mediode y la distancia entre O_j y su antiguo mediode) se pueden dar algunos de los siguientes 4 casos:

Caso 1:

Al cambiar O_i por O_h,
O_j pertenece al cluster de O_i
O_j está más cerca de O_{j2} que de O_h.
Entonces O_j cambia al cluster de O_{j2}.

$$C_{jih} = d(O_j, O_{j2}) - d(O_j, O_i)$$

Caso 2:

Al cambiar O_i por O_h,
O_j pertenece al cluster de O_i
O_j está más cerca de O_h que de O_{j2}.
Entonces O_j pasa al cluster de O_h.

$$C_{jih} = d(O_j, O_h) - d(O_j, O_i)$$

Caso 3:

Al cambiar O_i por O_h,
O_j no pertenece al cluster de O_i
O_j está más cerca de su actual mediode que de O_h.
Entonces O_j se queda en el mismo cluster.

$$C_{jih} = 0$$

Caso 4:

Al cambiar O_i por O_h,
O_j no pertenece al cluster de O_i
O_j está más cerca de O_h que de su actual mediode(O_{j2}).
Entonces O_j pasa al cluster de O_h.

$$C_{ijh} = d(O_j, O_h) - d(O_j, O_{j2})$$

Si el costo total, TC_{ih}, es negativo entonces se reemplaza efectivamente a O_i por O_h. O sea, O_h pasa a ser el nuevo mediode pues ese Clustering tiene mejor calidad y se sigue iterando con el resto de los pares de los objetos.

Desventajas:

- ✓ No es eficiente para grandes volúmenes de datos, pues tiene orden $O(k(n-k)^2)$ en cada iteración. Donde n representa la cantidad de objetos de la base de datos.

CLARA

CLARA(Clustering Large Aplications)

Algoritmo:

Se desarrolló con la motivación de adaptar PAM a grandes volúmenes de datos. La diferencia fundamental entre ambos es que CLARA busca mediodes en una *muestra de datos*, en vez de hacerlo sobre toda la base de datos.

Se seleccionan varias muestras y se aplica PAM sobre cada una. Si la muestra está bien elegida, los mediodes de la misma se aproximarán a los del conjunto de datos completo. La idea es tomar varias muestras y quedarse con la mejor. Para ello se calcula la *calidad del Clustering* midiendo la distancia de todos los objetos del conjunto completo de datos a los mediodes obtenidos.

Cada iteración es de orden $O(k(40+k)^2+k(n-k))$, tomando 40+2k muestras, que es la cantidad recomendada con fundamentos experimentales.

Ventajas:

- ✓ CLARA puede trabajar con mayores volúmenes de datos que PAM

Desventajas:

- ✓ La eficiencia de CLARA depende del tamaño de la muestra.
- ✓ Un buen Clustering basado en muestras no necesariamente representará un buen Clustering para todo el conjunto de datos.

CLARANS

CLARANS (Clustering Large Aplications based on Randomized Search)

Este algoritmo está basado en las ideas de PAM y CLARA.

Abstracción de grafos para describir PAM vs CLARA vs CLARANS:

Sea *n* la cantidad de objetos y *k* la cantidad de clusters.
Se define el grafo $G_{n,k}=(V,X)$, donde V es el conjunto de nodos y X es el conjuntos de arcos tal que:
$V = \{ \{O_{m1}, O_{m2}, ..., O_{mk}\} / O_{m1}, O_{m2}, ..., O_{mk}$ son objetos del conjunto de datos $\}$.
$S1 \in V, S2 \in V$
$(S1, S2) \in X$ sii $|S1 \cap S2| = k-1$ (o sea, todos sus objetos salvo uno son iguales)

Cada nodo:

- Tiene k(k-n) vecinos.
- Representa los *k* objetos que son tomados como mediodes.
- Identifica un Clustering.
- Tiene un *costo* asociado, que se mide como las distancias totales entre cada objeto y el mediode de su cluster. La diferencia de costo entre un nodo S1 y su vecino S2, que difieren en los objetos O_i y O_h, equivale al cálculo que se realiza en PAM al intercambiar el mediode O_i por el no mediode O_h (TC_{ih}).

PAM consiste en la búsqueda del nodo con costo mínimo en el grafo $G_{n,k}$. En cada paso, todos los vecinos del nodo actual son examinados. El nodo actual es reemplazado por el vecino de mayor descenso de costo. La búsqueda continua hasta obtener un mínimo. Problema: cuando n y *k* son muy grandes, examinar todos los vecinos k(n-k) lleva mucho tiempo.Ver [10] [22]

CLARA examina menor cantidad de vecinos que PAM. Restringe la búsqueda a subgrafos más pequeños que $G_{n,k}$. Problema: los subgrafos examinados son definidos enteramente por los objetos de la muestra. Sea Ma el conjunto de objetos de la muestra, entonces el grafo que se obtiene es $G_{Ma,k}$. A pesar que CLARA examina $G_{Ma,k}$ usando PAM, el problema es que la búsqueda está confinada dentro de los nodos de $G_{Ma,k}$. Si Sm es el nodo mínimo de $G_{n,k}$, y Sm no está en el subgrafo $G_{Ma,k}$, entonces Sm no puede ser encontrado con la muestra Ma. Para ello se debería tomar un número considerable de muestras para tener mayor probabilidad de encontrar el nodo mínimo de $G_{n,k}$, en alguna de ellas.Ver [10] [22]

CLARANS toma en cada *iteración* un nodo de $G_{n,k}$ al azar (nodo actual) y examina una cantidad fija (*maxvecinos*) de sus vecinos elegidos al azar. Así en cada iteración, se selecciona el Clustering de mejor calidad entre el nodo y sus vecinos seleccionados; utilizando la función de costos definida por PAM. Este óptimo local es comparado con el obtenido en cada iteración, quedándose con el nodo de costo más bajo. La cantidad de iteraciones que se realizan están determinadas por un parámetro de entrada (*numlocal*); se elegirán numlocal nodos y para cada uno se estudiarán maxvecinos vecinos.Ver [10] [21] [22]

Para elegir el valor de los parámetros maxvecinos y numlocal, se usan estudios basados en experimentos de prueba.

CLARANS a diferencia de CLARA no restringe la búsqueda a un subgrafo particular, toma nodos al azar de todo $G_{n,k}$.

CLARANS a diferencia de PAM no chequea todos los vecinos de un nodo, toma una muestra de ellos (maxvecinos). Cuanto más grande es el valor de maxvecinos, CLARANS se acerca más a PAM.

B.3.2.7 LKM (Local K- Means)

El algoritmo de búsqueda local es un algoritmo general para encontrar un optimo local de una función. Para una función de performance definida en un conjunto finito, el optimo puede ser encontrado teóricamente por fuerza bruta. La única razón para no hacer eso en la practica es el costo. Si la búsqueda esta limitada a una región pequeña, el algoritmo Greedy encontrara el mejor optimo local. Repitiendo este proceso, un optimo local de la función puede ser encontrado.

Sea F: P->R una función definida en un conjunto finito P. Un vecindario de un punto $p \in P$ es un subconjunto de P que contiene a p. Cada punto de P podría tener varios vecindarios.

Para cada problema de optimización particular, nos interesa solo el tipo de vecindarios que nos ayuda a balancear el costo de la búsqueda por fuerza bruta, la velocidad de convergencia, etc.

Para encontrar un optimo local de la función de performance F, definida sobre P tenemos:

Versión genérica del Algoritmo de Local Search (G-LS):

1. Empezando de una posición inicial current_p = $p_o \in P$ y marcar todos los vecindarios no usados.
2. Elegir un vecindario no usado, N(current_p) de current_p. Si no hay mas vecindarios sin usar STOP.
3. Buscar en N(current_p) un p_l = argmin (F(p)| $p \in$ N(current_p)), p da el mínimo de F sobre N(current_p).
4. Si F(p_l) < F (current_p) , current_p= p_l , Fin Si; Marcar N(current_p) como usado. Ir al paso 2.

La versión genérica del algoritmo es muy simple y va a converger a un optimo "local" porque el valor de F(current_p) es monótono decreciente.

El óptimo local de la función de performance hallada por LKM, que incluye todos los óptimos globales es un subconjunto del óptimo local a los cuales el algoritmo K-Means original puede converger también. Es menos probable que el LKM quede atrapado por el optimo local que el K-Means.

Algoritmo LKM Un Ciclo:

1. Comenzar con una partición inicial P = (S_1 . . . S_k);
   a=1;
   Marcar todos los vecindarios (items de datos) no usados

2. Si todos los items de datos en Sa están usados entonces a++;endSi
   Si a>K entonces STOP.
   Tomar un conjunto de datos Xo en Sa.

3. 3a. Setear A= n_a * (x_0 - m_a)* (x_0 - m_a)/(n_a -1);
   3b. Para (j=1; j ≠a , j <K, j++) hacer:
   δ_j MSE(P) = MSE(P')) = n_j * (x_0 - mj)* (x_0 - mj)/(nj +1) – A;
   donde P' se deriva de P moviendo x_0 de Sa a Sj;
   3c. Sea b argmin{δ_j MSE(P)<0 todos los j≠a } // el mejor cluster para mover x_0

4. Marcar x_0 como usado.
   Si b existe entonces
   Mover x_0 de Sa a Sb;
   n_a ; ++ n_b
   $m_a = (n_a * m_{a-}x_0)/(n_a - 1)$;
   $m_b = (n_b * m_{b-}x_0)/(n_b - 1)$;
   endsi
   Ir a paso 2

Dado que S es finito, el algoritmo finalizará después de un numero finito de pasos.

Algoritmo LKM:

Correr LKM_Un_Ciclo
Si Check(Criterio_de_parada) = true entonces STOP
End.

El criterio de parada puede ser:

1. El valor de la función de performance decrementada en ε en la corrida previa de LKM_Un_Ciclo; o
2. No se mueve ningún ítem de datos en la corrida previa de LKM_Un_Ciclo, el cual es mas estricto que el anterior.

Ventajas:

- ✓ Los ciclos y el orden en los cuales los datos en cada cluster son procesados no es importante para la correctitud del algoritmo. Lo único importante es chequear cada dato para asegurarse que no se puede mover ningún dato antes de detener el algoritmo.

Si usamos el criterio de parada 2. la partición convergente tiene que ser un optimo local, lo cual significa que ningún dato puede ser movido a un valor mas bajo del total de la varianza del cluster. Es interesante puntualizar que le algoritmo K- means original no garantiza un optimo local bajo esta definición. Ver [24]

B.3.2.8 Algoritmos Basados en Paralelismo

La clase de algoritmos basados en centros, tienen una estructura paralela natural. Sea L el numero de computadoras a utilizar. Para particionar todas las unidades L para el cálculo, el conjunto es particionado en L subconjuntos, $S = D_1 \cup D_2 \cup \ldots D_L$ y cada uno reside en su unidad correspondiente.

Es importante en esta instancia, no confundir la partición con el clustering, ya que esta partición es arbitraria y no tiene nada que ver con la estructura de clustering en los datos. Por otra parte esta partición es estática y no puede moverse de una computadora a la otra. El tamaño de las particiones son proporcionales a la velocidad de las unidades involucradas.

Una de las computadoras se elige como "Integrador", el cual:

- Suma los datos de todas las particiones para tener los datos globales
- Calcula los nuevos de los datos globales
- Chequea los criterios de parada e
- Informa a las nuevas unidades cuando deben parar o envía nuevos parámetros a las computadoras para comenzar con la nueva iteración.

Algoritmo:

0. Inicialización:
   Particionar el conjunto de datos y cargarlo en la unidad correspondiente, usando el algoritmo deseado para inicializar los parámetros en el Integrador.

1. Enviar el mismo conjunto de valores de parámetros a todas las computadoras.

2. Cálculos en cada unidad:
   Calcular los datos necesarios sobre los datos locales.

3. Enviar los datos necesarios al Integrador.

4. Armar los datos globales con los datos individuales enviados por cada procesador, evaluar la performance y revisar los criterios de parada. En caso de que estos criterios den como resultado que debe pararse, informar a todas las unidades que se detengan. De lo contrario ir al paso 1) para continuar la iteración.

B.3.3 Métodos Basados en Grafos

B.3.3.1 Triangulación

Este método se basa en la idea de que los puntos vecinos que se encuentran en el mismo cluster deben ser similares. No requiere que todos los posibles pares de puntos dentro de un mismo cluster lo sean. Esto permite un aspecto de transitividad en la descripción: si un punto es similar a otros dos puntos, aunque ninguno de los dos sean particularmente parecidos el uno con el otro, entonces los tres pertenecen al mismo cluster ya que el primer y tercer puntos están indirectamente relacionados. Este método es efectivo para identificar correctamente clusters no esféricos.

Una manera alternativa para definir clusters es considerar que si dos puntos vecinos están lo suficientemente cerca entonces pertenecen al mismo cluster. En la técnica propuesta los vecinos no necesariamente son los n vecinos más cercanos; son seleccionados usando una técnica de triangulación. En la implementación que se logra con la técnica de Delaunay, la triangulación resultante es muy cercana a la óptima y existen algoritmos muy eficientes para ello (O(N log N) en dos dimensiones y $O(N^{(2M-1)/M})$ en M dimensiones, con lo cual siempre es menor a $O(N^2)$).

Algoritmo:

Triangulando un conjunto de puntos de datos, los vecinos de cualquier punto son aquellos en su conjunto de adyacencia.

Una vez que se encontraron los vecinos de un punto, el próximo paso es determinar cuáles de ellos están lo suficientemente cerca como para estar en el mismo cluster. Esto se puede lograr eligiendo un punto de corte p. Todos los ejes entre vecinos cuya longitud supere p son considerados “muy largos” y son removidos del grafo. Con la correcta elección de p, idealmente se removerán los ejes entre clusters y se conservarán aquellos dentro de un cluster. Todo lo que resta es emplear un algoritmo de partición de grafos para encontrar componentes aislados en el grafo, y considerar cada uno un cluster.

Una búsqueda profundidad-primero es fácilmente implementada con una complejidad O(E+V), donde E es la cantidad de ejes y V la cantidad de vértices. Como en este caso E ~ O(V) y V=N entonces la complejidad del particionamiento es O(N).

Ventajas:

- ✓ Permite detectar clusters de cualquier forma
- ✓ No necesita saber de antemano la cantidad de clusters en los que se debe agrupar el conjunto de datos
- ✓ La gran mayoría de los puntos de ruido se vuelven clusters trivialmente chicos (Ej.:1, 2 ó 3 puntos)
- ✓ Es eficiente en el tratamiento de grandes volúmenes de datos; pues su complejidad es del orden de O(N log N)

Desventajas:

- ✓ Necesita saber el valor del punto de corte

B.3.3.2 Minimum Spanning Tree

Otra técnica de grafos utilizada es la de *Minimum Spanning Tree*.
Esto requiere extraer el MST del grafo completo y después remover algunos ejes; pero desafortunadamente formar un MST es una operación de orden $O(N^2)$.

Este método comienza considerando cada miembro de la población como un cluster. En el paso siguiente los dos clusters con la distancia mínima se fusionan para formar un único cluster. Este proceso continua hasta que todos los componentes son agrupados en el numero total de clusters requeridos.

B.3.3.3 Otros Algoritmos Basados en Técnicas de Grafos

Otro algoritmo sugiere comparar la longitud de cada arco contra la longitud promedio de los arcos cercanos y remover aquellos con longitud mayor al doble del promedio.

Sin embargo un experimento de triangulación de Delaunay muestra que este método no puede ser utilizado directamente ya que muchos de los arcos más largos tienden a estar dentro del cluster provenientes del mismo vértice, lo cual eleva el promedio del vecindario del punto donde se preservaron todos los arcos.

Una alternativa para el caso donde todos los clusters son aproximadamente de la misma densidad, es usar la longitud promedio de todos los arcos en el grafo.

Sin embargo, se ha probado que no es efectivo considerar el punto de corte p como el doble del promedio de la longitud de los arcos cuando existe ruido significante.

El problema es que los puntos adicionales entre clusters (como es el ruido) provocan el acortamiento de las longitudes de los arcos inter-cluster, derrotando al algoritmo. Lo que se necesita es alguna forma dinámica de elegir este valor multiplicativo.

B.3.4 Métodos Basados en Densidad

Las técnicas basadas en densidad, particionan el conjunto de datos en celdas que no se superponen y construyen sus histogramas. Las celdas que tienen una frecuencia relativamente alta de puntos son potenciales centros de clusters y los límites entre clusters quedan determinados por los "valles" del histograma.

Para reconocer los clusters, se considera que dentro de los mismos se tiene una densidad de puntos considerablemente mayor que afuera de ellos. La densidad de las áreas con ruido, es menor que la densidad en cualquier otro cluster.

B.3.4.1 Dbscan

DBSCAN (Density Based Spatial Clustering of Aplications with Noise)

Características:

- Permite descubrir clusters de forma arbitraria.
- Tiene en cuenta la existencia de ruido (outliers).
- Es eficiente para grandes volúmenes de datos.
- Sólo requiere un parámetro de entrada, y da soporte para que el usuario determine su valor adecuado.
- La forma del vecindario es determinada por la elección de la función de distancia entre 2 puntos.
- Este algoritmo puede ser utilizado con cualquier función de distancia, así se puede elegir una apropiada de acuerdo al dominio de la aplicación.

Conceptos Básicos:

Dada una base de datos D con puntos de un espacio k-dimensional. La distancia entre 2 puntos, p y q, se nota como dist(p,q).
Se define:

- Eps-vecinos de un punto ($N_{Eps}(p)$) como: $N_{Eps}(p) = \{q \in D \ / \ dist\ (p,q) \leq Eps\}$.

- p es directamente densamente-alcanzable desde q con respecto a MinPts y Eps si $p \in N_{Eps}(q)$ y $| N_{Eps}(q)| \geq Minpts$.

- p es densamente-alcanzable desde q con respecto a MinPts y Eps si existe una cadena de puntos $p_1, \ldots, p_n$ con $p_1=q$, $p_n=p$ y p_{i+1} es directamente densamente-alcanzable desde p_i

- p es densamente-conectado con q con respecto a MinPts y Eps si existe un punto r tal que: p y q son densamente- alcanzables desde r con respecto a MinPts y Eps.

- Un cluster es un conjunto de puntos densamente-conectados que es maximal con respecto a la densidad-alcanzable.

- El ruido es el conjunto de puntos que no pertenecen a ningún cluster.

Algoritmo:

Dados los parámetros Eps y Minpts, un cluster se obtiene en 2 pasos:

1) Elegir un punto arbitrario q de la de la base de datos que satisfaga $|N_{Eps}(q)| \geq Minpts$

2) Construir el cluster con todos los objetos que son densamente-alcanzables desde q.

Esto se repite hasta que no quedan puntos sin clasificar.

DBSCAN puede unir 2 clusters en uno, si están demasiado cerca. La distancia entre los clusters C1 y C2 se mide como: dist(C1,C2) = min{dist(p,q) / $p \in C1$ y $q \in C2$}. Luego dos conjuntos de puntos que tienen por lo menos la densidad necesaria para ser clusters serán separados sólo si la distancia entre ellos es mayor a Eps. Consecuentemente, puede ser necesario realizar una llamada recursiva del algoritmo para detectar clusters con un valor más grande de Minpts. El Clustering recursivo de los puntos sólo es necesario bajo determinadas condiciones.

La complejidad promedio para n puntos es de O(n*logn). También se proponen técnicas para determinar los parámetros MinPts y Eps.

Desventajas:

- ✓ Su complejidad es O(n*logn). Además requiere participación humana para determinar el parámetro Eps (el otro parámetro es MinPts es fijado en un valor bajo *k*, para reducir la complejidad computacional).

- ✓ Antes de poder determinar el Eps, DBSCAN debe calcular la distancia entre un punto y su k-ésimo vecino más cercano para todos los puntos. Luego ordena todos los puntos de acuerdo, con las distancias calculadas y dibuja el grafo k-dist ordenado. Este proceso consume tiempo y además un usuario debe examinar el grafo para encontrar el primer "valle". La distancia correspondiente es elegida como Eps y la calidad del Clustering obtenido depende en gran medida de ese parámetro.
  Ver [41]

B.3.4.2 Clique

CLIQUE (Clustering In Quest)

Aspectos considerados para el desarrollo:

Da un tratamiento efectivo de dimensiones grandes de las tuplas (objetos con muchos atributos).

Interpretabilidad de los resultados: El algoritmo no es sensible al orden en que los registros son ingresados y no presume ninguna distribución de los mismos.

Tiene *escalabilidad lineal* al número de objetos de entrada.

Usabilidad: El modelo puede ser extendido para manejar datos categóricos. Para ello se introduce un orden arbitrario en el dominio categórico. El particionamiento admite un valor categórico en cada intervalo y también coloca un intervalo vacío entre 2 valores distintos. Consecuentemente, si se elige esta dimensión para hacer Clustering, los clusters tendrán el mismo valor en esta dimensión.

Conceptos Básicos:

Se trabaja sobre un *espacio*, S, d-dimensional; donde un subespacio de S tiene a lo sumo *k* dimensiones, con k<d. $S = A_1xA_2x...xA_d$, donde cada A_i es un dominio de un atributo numérico.

Se particiona a S en unidades que no se superponen, dividiendo cada dimensión en ξ (parámetro de entrada) intervalos de igual longitud. Cada *unidad* u de S, está formada por un intervalo $u_i = [l_i,h_i]$ de cada dimensión. La selectividad de una unidad es una fracción de la cantidad total de puntos que contiene. Una unidad es *densa* si su selectividad es mayor a τ (parámetro de entrada). Un objeto v, del espacio S, está *contenido* en la unidad u si $\forall u_i$ $(l_i \leq v_i < h_i)$. Una unidad de un subespacio de S, es un conjunto de intervalos de *algunos* atributos de S.

Un *cluster* es un conjunto maximal de unidades densas conectadas en k-dimensiones. Dos unidades u, w k-dimensionales están *conectadas* si tienen una cara en común o existe otra unidad z tal que u está conectada a z y w está conectada a z. Dos unidades $u = \{u_{t1},u_{t2},...,u_{tk}\}$, $w = \{w_{t1},w_{t2},...,w_{tk}\}$ k-dimensionales *tienen una cara en común* si para k-1 dimensiones vale $u_{tj} = w_{tj}$, y para la k-ésima vale $h(u_{tk}) = l(w_{tk})$ o $l(u_{tk}) = h(w_{tk})$.

El problema que hay que resolver es encontrar clusters en todos los subespacios. Es necesario tener en cuenta que: Para una unidad densa de *k* dimensiones, sus proyecciones en cualquiera de sus k-1 dimensiones son densas.

Parámetros de entrada:

$V = \{v_1, v_2, \ldots, v_m\}$, con $v_i \in S$.
ξ, cada dimensión se divide en ξ intervalos de igual longitud.
τ, para que una unidad sea densa su selectividad debe ser mayor a τ.

Algoritmo:

Identificación de subespacios que contienen clusters.
La idea es encontrar las unidades densas en los distintos subespacios.

1) Determinar las unidades densas de una dimensión recorriendo todos los datos.

2) Habiendo determinado las unidades densas de k-1 dimensiones (D_{k-1}), las unidades de *k* dimensiones se calculan de la siguiente forma:

C_k se obtiene del join entre D_{k-1} y D_{k-1}, donde la condición del mismo iguala a las primeras k-2 dimensiones.

3) Descartar las unidades en C_k que tienen una proyección en k-1 dimensiones que no esté incluida en C_{k-1}.

Para que este algoritmo sea más eficiente se utiliza la siguiente técnica: MDL-based pruning (minimal description length) que trata de usar sólo unidades densas que están en subespacios "interesantes".

La idea es para cada subespacio contar la cantidad de puntos que se hallan en todas las unidades densas del mismo; eligiendo los subespacios de una alta cantidad de puntos. Esto elimina subespacios y hace que el algoritmo sea más rápido, pero se pueden perder algunos clusters.

1) Encontrar los clusters.

El problema es equivalente a encontrar componentes conexas en un grafo. Los *nodos* son las unidades densas. Hay un *arco* entre 2 nodos sii las unidades densas tienen una cara en común. Para encontrar las componentes conexas o clusters se puede usar el algoritmo primero en profundidad (depth-first).

2) Generar una descripción minimal para cada cluster.

Cada cluster puede ser descrito por un conjunto de desigualdades, la idea es obtener una descripción compacta cubriendo el cluster con un número minimal de rectángulos maximales y describir al mismo como su unión.

Esto se realiza en 2 pasos:

a. Cubrir al cluster con un algoritmo Greedy con un número de rectángulos maximales.

b. Descartar los rectángulos redundantes para generar un cubrimiento minimal.

Evaluaciones empíricas muestran que el orden del algoritmo es lineal a la cantidad de registros de datos.

Desventajas:

La adecuación de los clusters obtenidos puede ser degradada por la simplicidad del método.
Ver [41] [35] [37]

B.3.5 Métodos Basados en Técnicas Estadísticas

Estas técnicas hacen la suposición de que distribuciones probabilísticas sobre atributos separados son estadísticamente independientes una de la otra. Esto se aleja un poco de la realidad, donde puede existir correlación entre atributos, y tal vez esa correlación es la que estamos tratando de descubrir. Las representaciones probabilísticas de los clusters tienen un alto costo de actualización y almacenamiento, especialmente si los atributos tienen un gran número de valores posibles, ya que su complejidad depende de la cantidad de atributos y de la cantidad de valores que puede tomar cada uno.

La idea de estos métodos es dado un conjunto de ejemplos, recuperar la especificación del problema que resuelven. En general estos métodos se basan en aspectos probabilísticos y el representante más conocido es COBWEB.

B.3.5.1 Cobweb

COBWEB es un programa de formación de conceptos para la creación de árboles de clasificación jerárquicos.

Una clasificación es ' buena' si la descripción de un ejemplo puede ser predicha con una alta precisión, dado que el mismo pertenece a una clase específica. COBWEB evalúa la clasificación de un conjunto de ejemplos en clases mutuamente excluyentes C_1, C_2, ... , C_n por una función estadística llamada *Utilidad de Categoría(CU)*:

$$CU(C_k) = \frac{\sum_{k=1}^{n} P(C_k) \sum_i \sum_j [P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2]}{n}$$

donde C_k es una clase, $A_i = V_{ij}$ es un par propiedad-valor, P(x) es la probabilidad de x, y n es el número de clases. El primer término en el numerador mide el número esperado de pares propiedad-valor que pueden ser predichos correctamente usando la clasificación. El segundo término mide la misma cantidad pero sin usar las clases. Por lo tanto, la función mide el aumento en la predicción de dichos pares con respecto al solo uso de su frecuencia. Finalmente, la medida se normaliza respecto al número de clases.

Cuando se introduce un nuevo ejemplo, COBWEB trata de acomodarlo en la jerarquía existente comenzando por la raíz.

El sistema realiza una de las siguientes operaciones:

1) Expandir la raíz, si no tiene ninguna sub-clase, creando una nueva clase y uniendo la raíz y el nuevo ejemplo como su subclase;
2) Agregar el nuevo ejemplo como una nueva subclase de la raíz;
3) Agregar el nuevo ejemplo a una de las subclases de la raíz;
4) Combinar las dos mejores subclases y poner el nuevo ejemplo en la subclase combinada; o
5) dividir la mejor subclase y nuevamente considerar todas las alternativas.

Si el ejemplo ha sido agregado a una clase existente, continúa el proceso recursivamente con esa clase como el primer nivel de la nueva jerarquía. COBWEB utiliza nuevamente la función CU para determinar cual es el siguiente operador que debe aplicar.

Las clases se describen usando sólo sus propiedades características, considerándose que las características de la misma son valores de las propiedades que satisfacen $P(A_i = V_{ij} \mid C_k) \geq$ *umbral* y $P(C_k \mid A_i = V_{ij}) \geq$ *umbral* donde *umbral* es un valor fijo predeterminado.

Ventajas:

- ✓ Permite producir un número de clases pequeño y útil sin tener que especificarse a priori un límite superior para el número de clases. Por otra parte, la jerarquía crece linealmente con el número de ejemplos.
- ✓ Tiene la habilidad de adaptarse a dominios cambiantes ya que es sensible al orden en la presentación de los ejemplos.

Desventajas:

- ✓ Sólo maneja propiedades nominales.
- ✓ Asume que la distribución de la probabilidad de los diferentes atributos son independientes entre sí, siendo esta una suposición muy fuerte ya que suele existir correlación entre ellos.
- ✓ Tiene un esquema de diseño y predicción muy rígido. Ya que continúa con el proceso de diseño hasta que logra un diseño existente completo y no permite la generación de un nuevo diseño o el agregado de conocimiento subjetivo. Además, el esquema de predicción no es adecuado, porque produce solo un posible candidato para un elemento dado y es susceptible a ruido, que provoca tasas de error altas.
- ✓ Usa sólo la función CU para guiar el aprendizaje y la predicción. No usa conocimiento del dominio del problema, a pesar que esté disponible y puede mejorar sustancialmente el aprendizaje.

Ver [41]

B.3.5.2 Ecobweb

Enhanced COBWEB

Para superar algunas de las limitaciones de COBWEB, se creó otro método llamado ECOBWEB, el cual fundamentalmente permite el tratamiento de propiedades continuas.

Para ello debe adaptar CU para el uso de propiedades continuas (ya que la probabilidad en un solo punto es cero), esto lo realiza calculando la media de los pares propiedad-valor dentro de una clase transformando la fórmula de la siguiente forma:

$$\sum_j P(A_i = V_{ij} \mid C_k)^2 \Rightarrow P(A_i = \overline{V_i} \mid C_k)^2 \Rightarrow \left(\int_{-\frac{d_i}{\sigma_i}}^{+\frac{d_i}{\sigma_i}} p_i(x)dx \right)^2$$

donde:

$$2d_i = \frac{RangoEsperadoDeLosValoresDeA_i}{NúmeroEsperadoDeLosIntervalosDist\,int\,osDeA_i}$$

$\overline{V_i}$ es la media de los valores de A_i en C_k, y σ_i es la desviación estándar de los valores de A_i en C_k.

El segundo término de CU, se calcula de forma similar calculando σ_i como la desviación estándar de los valores A_i en la raíz de la clasificación.

La sensibilidad de la performance de ECOBWEB a la elección de $2d_i$ es baja.

La distribución de la probabilidad $p_i(x)$ debe ser seleccionada antes de comenzar el proceso de aprendizaje. Idealmente, la selección debe estar basada en el dominio de aplicación, aunque alternativamente se pueden asumir distribuciones comunes. La elección por defecto es la distribución normal:

$$p_i(x) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-(x-\overline{V_i})^2 / 2\sigma_i^2}$$

Una distribución que resulta ser deficiente puede ser reemplazada sin que sea necesario reprocesar los ejemplos anteriores.

Otra mejora al método es el agregado de un mecanismo para el manejo del ruido.

B.3.5.3 Sting

Sting (Statistical Information Grid Approach)

Aspectos considerados para el desarrollo:

Cuando los datos sobre los que se hace Clustering son el resultado de una consulta, los algoritmos CLARANS, BIRCH y DBSCAN, entre otros, requieren de tres fases:

Fase I: Encontrar respuesta a la consulta.
Fase II: Construir estructura auxiliar.
Fase III: Hacer el Clustering.

A STING, en cambio, sólo le toma un paso responder a la consulta.
Fue diseñado para ser aplicado en bases de datos espaciales y para procesar en forma eficiente consultas "orientadas a regiones" sobre un conjunto de datos. Estas, preguntan por la selección de regiones que satisfacen ciertas condiciones de densidad, área total, etc.

Conceptos Básicos:

El *espacio* es dividido en celdas rectangulares y se arma una estructura jerárquica con niveles. Una *celda* en el nivel i corresponde a la unión de las áreas de sus hijos en el nivel i + 1. Cada celda tiene 4 hijos, exceptuando a las hojas. Cada hijo corresponde a un cuadrante del padre. La *raíz* representa el espacio completo. Se define para un espacio de 2 dimensiones pero se puede generalizar a más dimensiones. El tamaño de las hojas se define en función de la densidad de objetos.

Atributos de las celdas:

- n, es la cantidad de objetos que tiene la celda.
- m, es la media de los objetos que tiene la celda.
- s, es la desviación estándar de los objetos que tiene la celda.
- min, es el valor mínimo que toma un objeto de la celda.
- max, es el valor máximo que toma un objeto de la celda.
- distribución, es el tipo de distribución que tienen los puntos de una celda (normal, uniforme, exponencial, etc).

Estos atributos son calculados para las hojas de la jerarquía usando los datos de entrada, luego los niveles superiores se calculan usando los valores de sus hijos. El único parámetro complicado de determinar es el de la distribución. Para la misma se puede realizar un test χ^2 o se puede ver como parámetro de entrada.

Algoritmo:

Con la jerarquía de celdas ya construida, se utiliza un técnica top-down para responder a consultas espaciales de data mining. Para cada consulta se comienza examinando las celdas de los niveles más altos.

1) Determinar el nivel de la jerarquía dónde comenzar.

2) Para cada celda de ese nivel, calcular el intervalo de confianza de la probabilidad de que esta celda sea relevante para la consulta.

3) Con el intervalo, se rotula a cada celda como relevante o no.

4) Si la celda es del nivel de las hojas, Ir al paso 6) Si no Ir al paso 5)

5) Bajar un nivel de la jerarquía, volver al paso 2) con las celdas relevantes del nivel anterior.

6) Si se satisfizo la consulta, ir al paso 8). En este caso la información estadística asociada es suficiente para responder a la consulta. Si no ir al paso 7). En este caso la información estadística asociada no es suficiente para responder a la consulta, entonces en el paso 7) se obtendrán más datos de las celdas relevantes que permitan responder la consulta.

7) Obtener los datos de las celdas relevantes y procesarlos. Devolver los resultados que satisfacen la consulta. Ir al paso 9).

8) Encontrar la *región* de celdas relevantes.

9) Devolver las regiones que satisfacen los requerimientos de la consulta.

La idea es buscar todas las regiones que satisfacen la densidad especificada y se usa un *algoritmo de bredth- first*. Para cada celda relevante, se examinan las celdas dentro de cierta distancia al centro de la celda, para ver si la densidad promedio dentro de este área pequeña es mayor que la densidad especificada. Si es así, entonces el área es marcada y todas las celdas relevantes, que fueron examinadas, se encolan. En cada paso, se toma una celda de la cola y se repite el mismo procedimiento salvo que las celdas encoladas son aquellas que no fueron examinadas antes. Cuando la cola se vacía, entonces se encontró la región.

Ir al paso 9).

Fin.

Ventajas:

- ✓ Es una técnica independiente de las consultas. La información que guarda de cada celda permite contestar una gran variedad de consultas.
- ✓ La complejidad computacional es de orden O(K), donde *k* es la cantidad de celdas en el nivel más bajo (hojas). Generalmente K<<n, donde n es la cantidad de objetos.

- ✓ Cuando se actualizan datos, no se necesita recalcular toda la información de la jerarquía de celdas. Se puede realizar una actualización incremental.

Ver [41]

B.3.6 Definiciones de Distancias

Algunas de las *Distancias* para medir cercanía entre clusters son:

- ✓ Distancia Euclidiana con centroides:

$$D1 = ((\vec{X}o_1 - \vec{X}o_2)^2)^{1/2}$$

- ✓ Distancia de Manhattan con centroides:

$$D2 = \left|\vec{X}o_1 - \vec{X}o_2\right| = \sum_{t=1}^{d} \left|\vec{X}o_1{}^{(t)} - \vec{X}o^{(t)}{}_2\right|$$

- ✓ Distancia promedio entre clusters:

$$D3 = \left(\frac{\sum_{t=1}^{N1} \sum_{j=N1+1}^{N1+N2} (\vec{X}_t - \vec{X}_j)^2}{N1N2} \right)^{1/2}$$

- ✓ Distancia promedio dentro de los clusters:

$$D4 = \left(\frac{\sum_{t=1}^{N1+N2} \sum_{j=1}^{N1+N2} (\vec{X}_t - \vec{X}_j)^2}{(N1+N2)(N1+N2-1)} \right)^{1/2}$$

Ver [45] [51]

B.4 Cuadros Comparativos

A continuación se presenta un compendio del orden de algunos de los algoritmos vistos.

Algoritmo	***Orden***
Single Linkage	n*n
Complete Linkage	n*n
Average Linkage	n*n
Ward	n*n
BIRCH	O(n)
KNN	O(n)
K_means	O(tkmn), orden menor a n*n
Pam	O(k(n-k)*(n-k)), por iteración
Clara	O(kS*S + k(n-k)) por iteración
Clarans	Complejidad cuadrática
Trianqulacion	O(n*log n)
MST	O(n*n)
DBScan	O(n*log n)
Sting	O(K), K<<n, n #objetos

Ahora se detallan algunas de las características de los algoritmos relacionadas con los *ruidos*, los tipos de *datos* que estos manejan (categóricos o numéricos), la *escalabilidad*, *dependencia* del orden de entrada de los objetos y *forma* del Cluster generado.

Algoritmo	Ruidos	Datos	Escalabilidad	Dependencia	Forma del Cluster
		Categ. y/o num.		Orden entrada	
Single Linkage	Sensible				Alargados y finos
Complete Linkage	Sensible				Pequeños y compactos
Average Linkage	Sensible		Usa la conectividad promedio para escalar		
Ward	Sensible				Pequeños y homogeneos
BIRCH	Robusto	Númericos	La provee a traves de decisiones locales	Independiente	
K_means	Sensible	Númericos	Escalable con compresión de datos	Dependiente	Solo Clusters convexos
K-Prototypes	Sensible	Ambos			Clusters convexos
K_modes	Sensible	Ambos	Escalable a conjuntos de datos muy grandes	Dependiente	Clusters convexos
KNN	Robusto				
Pam			No escala bien		
Clara			Escalable		
Clarans	Robusto		Escalable		
Trianqulacion	Robusto				De forma arbitraria
MST		Ambos			
DBScan	Robusto		Escalable		De forma arbitraria
CLIQUE	Robusto		Lineal	Independiente	De forma arbitraria

Finalmente, un resumen de las ventajas y desventajas de los algoritmos:

Algoritmo	Ventajas	Desventajas
Single Linkage	Eficiente, no es necesario un centroide del cluster Puede manejar formas no-elipticas	No efectivo con clusters compactos y de igual tamaño
Complete Linkage	Efectivos en clusters compactos y de igual tamaño	No efectivo con clusters alargados
Ward	Eficiente	Pobre al recuperar clusters elongados En instancias grandes, su complejidad es un problema
BIRCH	Metodo incremental, rapido y produce buenos clusters Mas eficiente que Clarans	Trabaja mal cuando los datos no son esfericos
K_means	Eficiente en gdes. # de datos, no en gdes. # de clusters Termina en un optimo local	Es necesario especificar la cantidad de clusters
K-Prototypes	Eficiencia del K-means, trabaja con datos categoricos	Solo detecta clusters de forma convexa
K_modes	Algoritmo muy rapido Versión simplificada del algoritmo K-Prototypes	Solo detecta clusters de forma convexa
KNN	Simple y poderoso	Se comporta mejor con clusters de pocas dimensiones
K-medoid	Grandes volumenes de datos en forma eficiente Para hallar medoides solo necesita una def. de distancia	
Pam		Ineficiente para grandes volumenes de datos
Clara	Mejora de PAM, para grandes volumenes de datos Trabaja con mayores volumenes de datos que PAM	Eficiencia depende del tamaño de la muestra
Clarans	Mezcla de PAM y CLARA, mas eficiente que ambos	
Trianqulacion	Eficiente en grandes volumenes de datos Eficiente y sensible para el analisis de clusters	
DBScan	Eficiente en grandes volumenes de datos Independiente de la funcion de distancia	
CLIQUE	Halla clusters en subespacios aun si no existen en el espacio original	
Cobweb	Produce un numero de clases pequeño y util	Esquema de diseño y prediccion muy rigidos
Sting	Permite actualización incremental	

Ref. para los cuadros: Ver [1], [22], [27], [37], [41], [42]

B.5 Reseña

En los últimos 40 años, se han desarrollado las técnicas de Clustering, las cuales han evolucionado considerablemente y mas allá de que continúan evolucionando, hoy por hoy ya existe una amplia gama de algoritmos que sirven a múltiples propósitos.

La evolución de la tecnología informática ha incidido positivamente en esta disciplina. Como una consecuencia directa de la creación y propagación de redes se están desarrollando algoritmos de Clustering de procesamiento en paralelo (Clase de algoritmos de Clustering de gran utilidad para conjuntos de datos muy grandes).

En la actualidad se pretende perfeccionar los algoritmos a nivel de eficiencia, escalabilidad, interpretabilidad y usabilidad de los mismos. Por ejemplo, se ha trabajado en algoritmos sensibles a los ruidos que incluso los utilizan como prueba de errores.

Algunos algoritmos atacan puntos específicos, o surgen a raíz de un problema planteado. Muchos de ellos por ejemplo, están limitados a datos numéricos ya que minimizan una función de costo, los cuales no consideran aquellas aplicaciones que cuentan con datos categóricos, ya que los mismos no son trivialmente convertidos a valores numéricos.

Existen también algoritmos muy eficientes pero que no son adecuados para grandes volúmenes de datos y otros están limitados a detectar clusters de determinada forma. Si bien estas técnicas son perfectibles cada una ataca algunos aspectos particulares, y las soluciones "mágicas" que cubran todos los requerimientos adecuadamente, no existen. Es por eso que la técnica a utilizar depende mucho del contexto de uso, de cual sea la información con la cual se cuenta y cual sea el algoritmo que mejor se adecua.

Hay un esfuerzo en la actualidad por testear el significado de las clasificaciones. Tal vez, una forma de lograr testeos significativos sea desarrollar criterios de optimalidad para las clasificaciones, otra forma podría ser la optimización del criterio en uso.

Otro desafío presente en Clustering al día de hoy es testear la homogeneidad dentro y entre los clusters en términos de las variables que los definen.

Es a través de estos testeos que se demuestra que “tan bien” fueron particionados los datos a cada nivel del Clustering; se trata entonces no solo de demostrar la correctitud y consistencia (que ya están implícitas en el método elegido), sino también de alcanzar el máximo “grado de calidad” en los resultados obtenidos para los parámetros dados en el planteo inicial del problema. Ver [3]

Anexo C. Software Existente

Hasta el momento se han diseñado variadas herramientas para Clusterizar. En general, son con propósitos específicos (medicina, bioingeniería, etc.) aunque también hay genéricas.
A continuación se presenta una lista del software encontrado en Internet.

- Clustan

Se especializan en diseño de software para análisis de clusters. Incluye un paquete de gráficos que ofrece además algunas herramientas analíticas, realiza análisis jerárquico de clusters usando 11 métodos distintos. Encuentra los miembros de cada cluster, reordena árboles, busca clusters en áreas de gran densidad y trata de hallar clusters naturales, etc.

Dir.:http://www.clustan.com

- CognoSuite

Producto para utilizar en minería de datos.

Dir.: http://www.cognos.com

- SPSS

Proveen soluciones analíticas como forma de ayudar en la toma de decisiones de las compañías y ayudarlas a hacer predicciones. Entre otras utilidades, su software provee una variada gama de soluciones analíticas, entre ellas la clusterización de datos.

Dir.: http://www.spss.com

- Applied Maths

Grupo de biocientificos que interactúan con desarrolladores para hacer "software a medida" entre estos cuentan con varios programas dedicados a distintas áreas: pattern matching, clustering, data mining y métodos de identificación para conjuntos de datos masivos como los microarrays y los genechips. Desde hace nueve años se dedican al análisis de huellas dactilares para el cual desarrollaron el paquete de software GelCompar.

Dir.: http://www.applied-maths.com

- P&M research technologies

Se especializan en la *visualización*, cluster automático, clasificación, etc.

Dir.: http://www.pmrt.com/

- ASA Corp. (Varias herramientas)

Diferentes herramientas de clustering y data mining encarado mas que nada al usuario analista de negocios. Utiliza modelos automatizados

Dir.: http://www.asacorp.com/HTML/Products/products.htm

- Fuzzy logic Box

La fuzzy logic box ayuda a resolver rápidamente problemas matemáticos. Identifica clusters. Se usa con Matlab

Dir.: http://www.mathworks.com/products/fuzzylogic

- Kdnuggets

Sitio donde hay una recopilación de distinto software de Clustering, tanto comercial como de distribución gratuita.
Los paquetes de software comercial que difieren de los anteriores son:

- Darwin 3.6 Suite
- Cviz Cluster Visualization
- IBM Intelligent Miner for Data
- SOMine

Dentro de los de distribución gratuita se encuentran:

- Autoclass C
- Ecoweb
- MCLUST/EMCLUST
- FCMeans Clustering Package para Matlab
- Snob

Dir.:http://www.kdnuggets.com/software/clustering.html

- Online Software for Clustering

Sitio con una breve reseña de programas (solo los ejecutables) y paquetes disponibles al público, realizado por estudiantes u otras personas dedicadas al estudio de las técnicas de Clustering. Tiene la forma de cut & paste de e-mails y otros lugares.

Dir.: http://astro.u-strasbg.fr/~fmurtagh/mda-sw/online-sw.html

Anexo D. Otros Casos de Testeo

En este anexo se incluyen corridas paso a paso de los diferentes algoritmos, como forma de ver la evolución de los mismos a lo largo de toda su ejecución.

D.1 KNN paso a paso

El archivo de Datos de Entrada contendrá la siguiente información:

```
2
7
1 -5 -2 600 1500
2 5 -2 600 1500
10 -2 0 600 100 LNC FinLNC
20 2 0 600 100 LNC FinLNC
30 1 -1 600 100 LNC FinLNC
40 -1 -1 600 100 LNC FinLNC
50  6.5 1.5 600 100 LNC FinLNC
60  -7 -4 600 100 LNC FinLNC
70 0 1.5 600 100 LNC 1 2 FinLNC
```

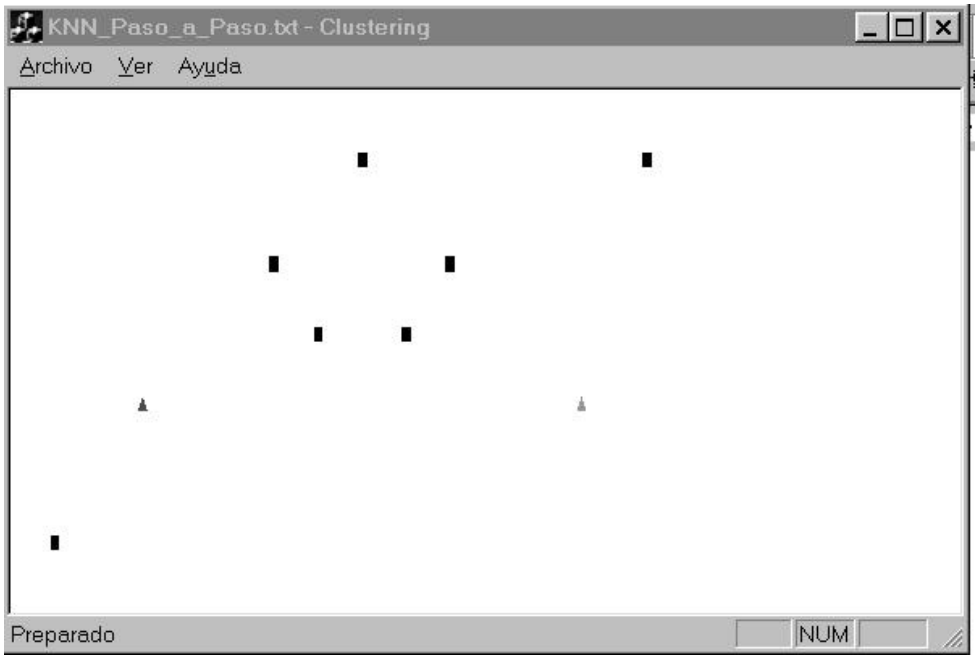
Dado que KNN no itera, la asignación inicial es la definitiva. También se muestra como un cliente sin asignar (por incompatibilidad con los depósitos) queda de color negro.

Ejecución:

Factores de Ponderación: espacial=1, temporal =1

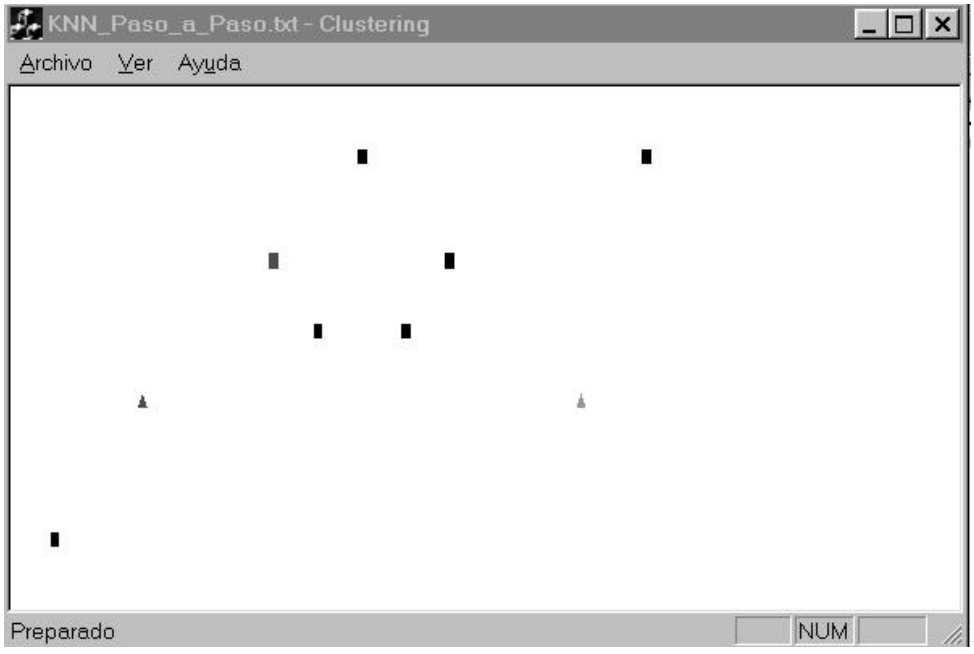
Distribución Inicial:

A continuación se muestra la distribución espacial de Clientes y Depósitos. En KNN se define inicialmente un cluster a partir de cada Depósito(coincidiendo identificadores) y se almacenan los Depósitos en una lista de asignados. Luego, se procesan los Clientes en el orden de entrada asignándolos al Cluster que tenga mayoría en los *k* elementos más cercanos al Cliente.



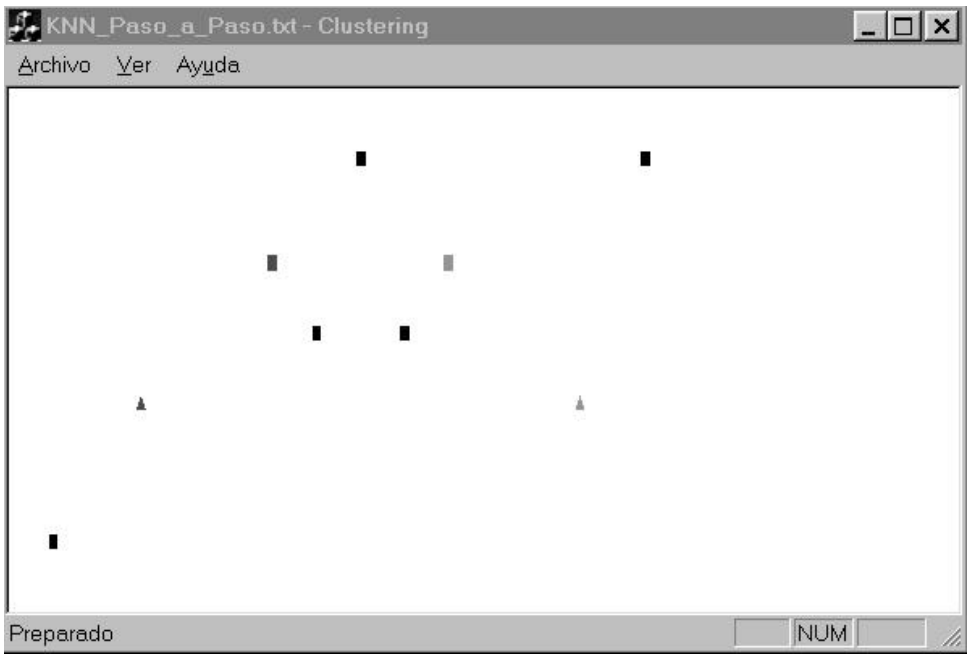
Asignación - Cliente 10:

Dado que el Depósito 1 es el más cercano al Cliente 10, este se asigna al Cluster del Depósito 1.



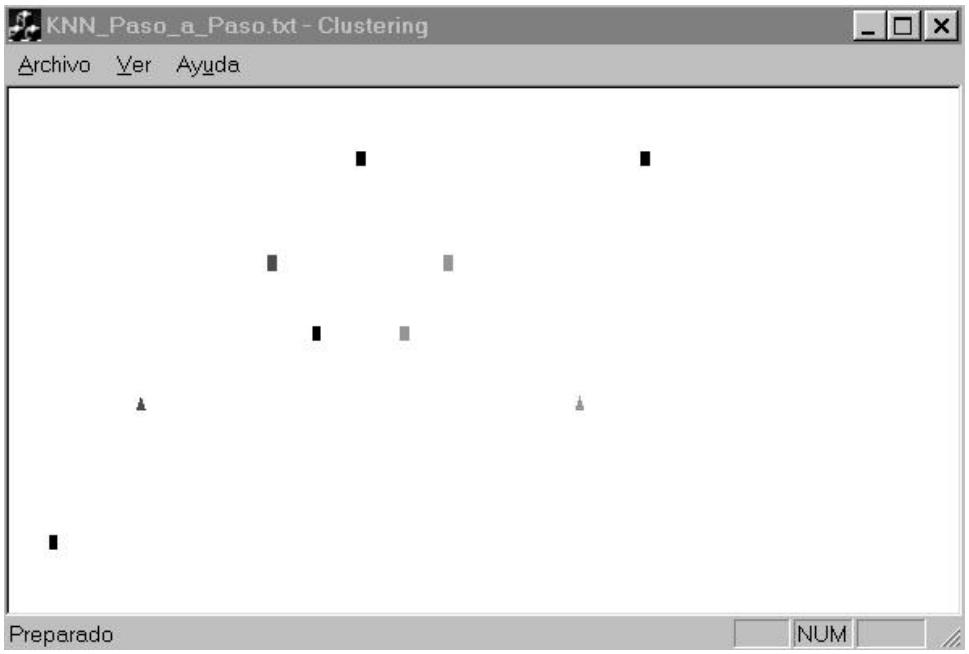
Asignación - Cliente 20:

De manera similar al paso anterior, el Cliente 20 se asigna al cluster del Depósito 2.



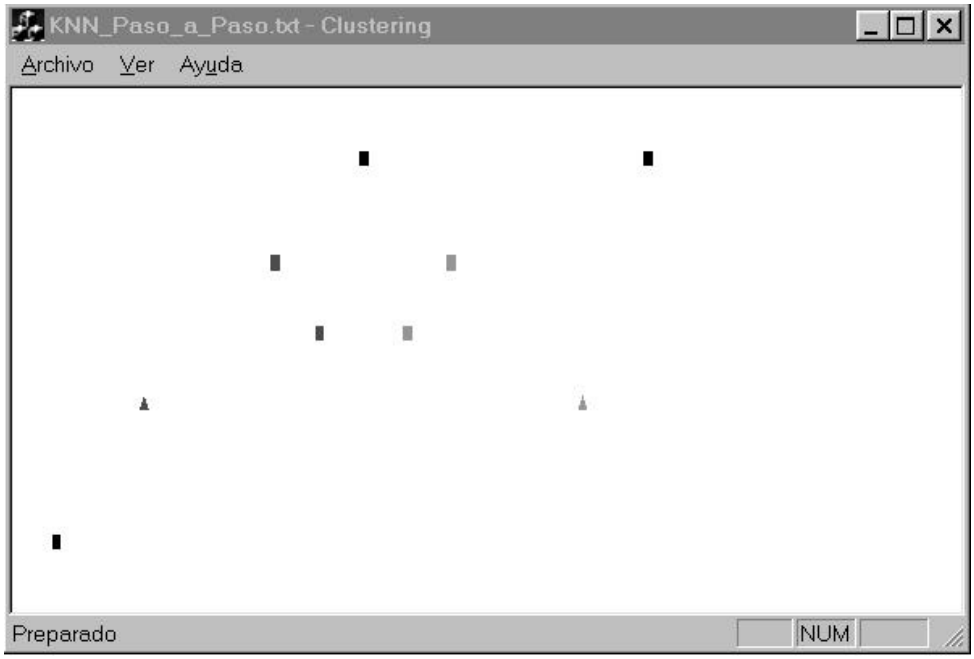
Asignación - Cliente 30:

El Cliente 30 se asigna al cluster del Depósito 2, debido a que el elemento procesado mas cercano es el Cliente 20 que pertenece a ese cluster.



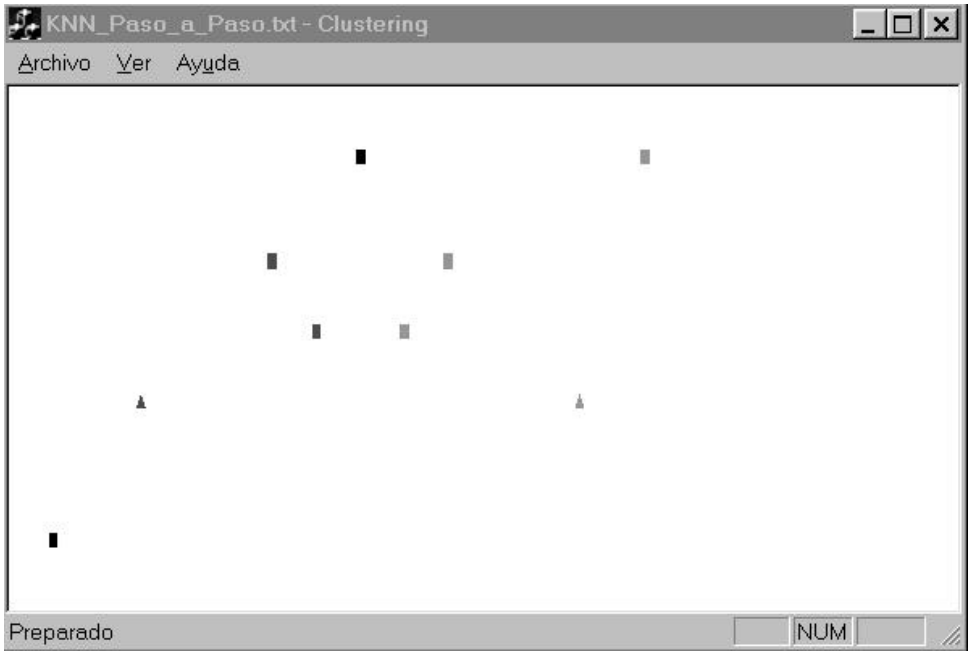
Asignación - Cliente 40:

Como en el paso anterior, el Cliente 40 está mas cercano al Cliente 10 que a los demás elementos procesados, por lo tanto el algoritmo lo asigna al Cluster 1



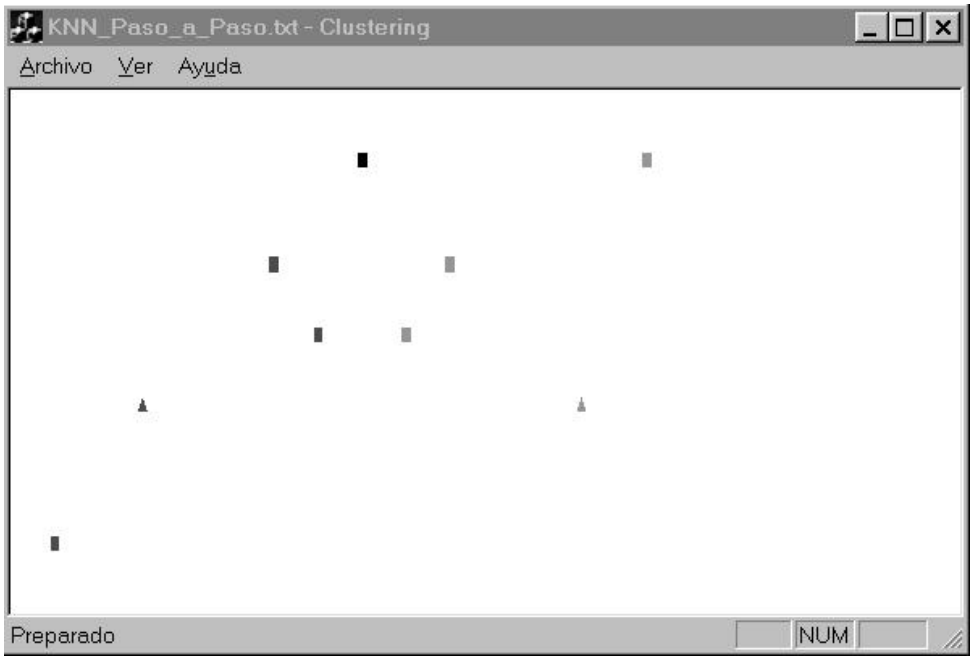
Asignación - Cliente 50:

Este Cliente se encuentra mas cercano al Cliente 20 que al resto, por lo tanto se asigna al Cluster 2.



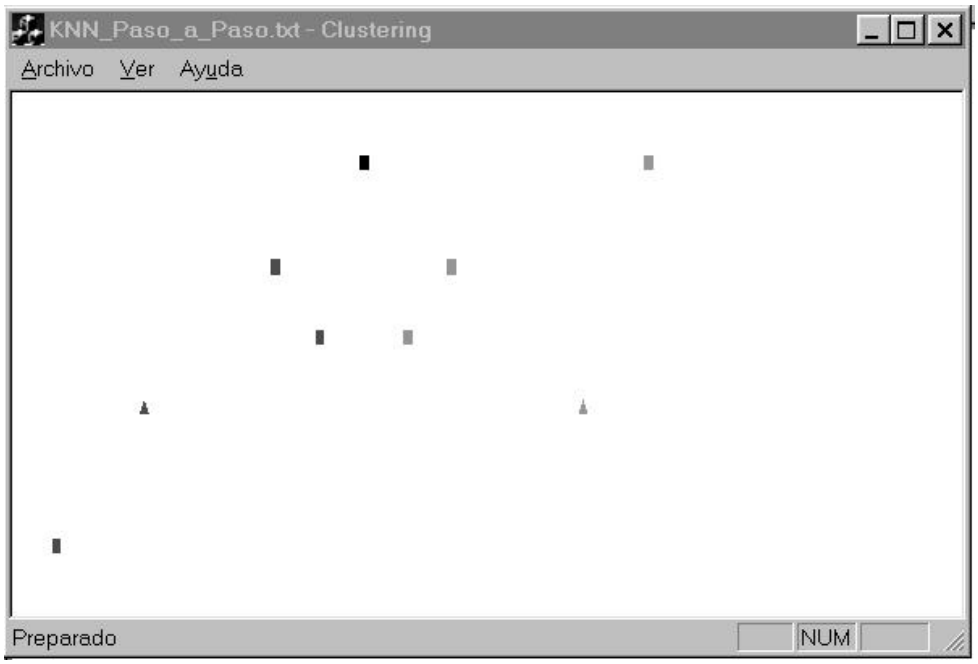
Asignación - Cliente 60:

El Cliente 60 está mas cerca del Depósito 1 que de cualquier otro elemento, por lo tanto es asignado al Cluster 1.



Asignación - Cliente 70 Resultado Final:

Este Cliente no se asigna debido a que es incompatible con todos los Depósitos de los Clusters. Dado que este es el último Cliente en procesar, la siguiente es la configuración final.



D.2 PAM paso a paso

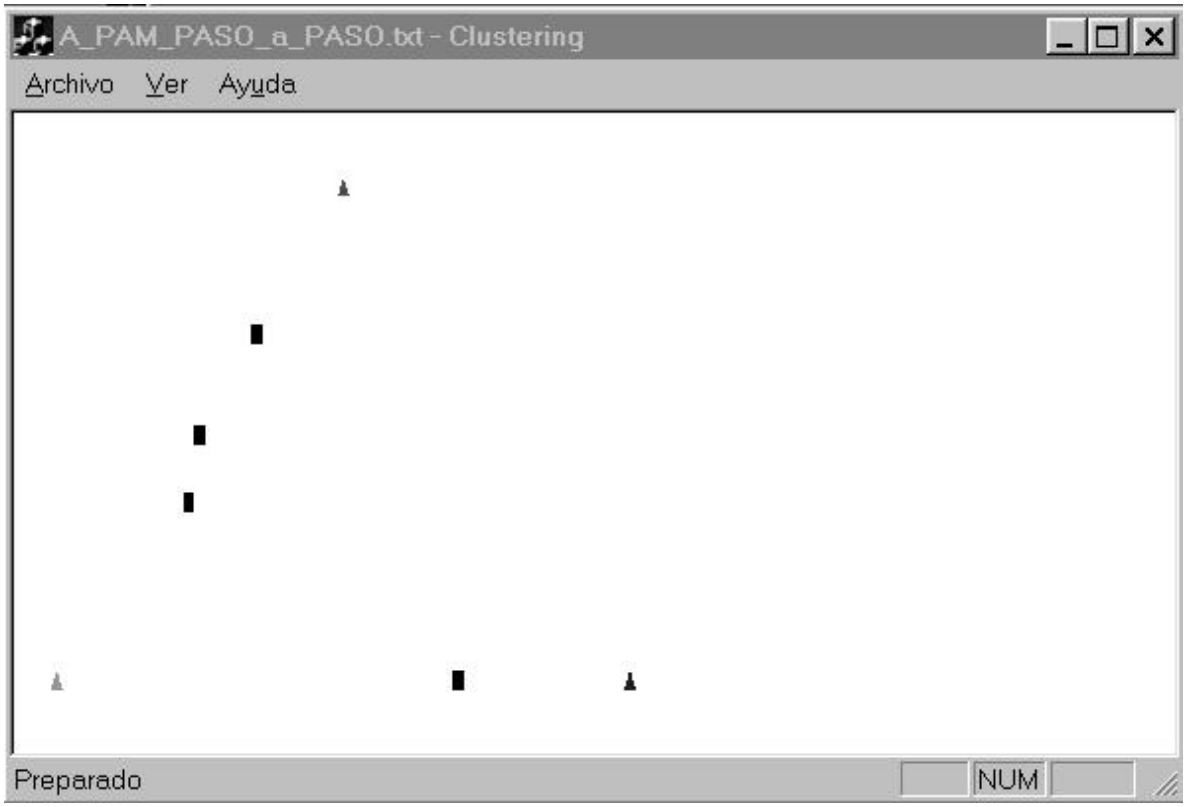
El archivo de Datos de Entrada contendrá la siguiente información:

```
3
4
1 5 5 600 1500
2 0 0 600 1500
3 10 0 600 1500
10 3.5 3.5 600 100 LNC FinLNC
20 2.48 2.48 600 100 LNC FinLNC
40 7 0 600 100 LNC FinLNC
30 2.3 1.8 600 100 LNC 1 FinLNC
```

Distribución Inicial:

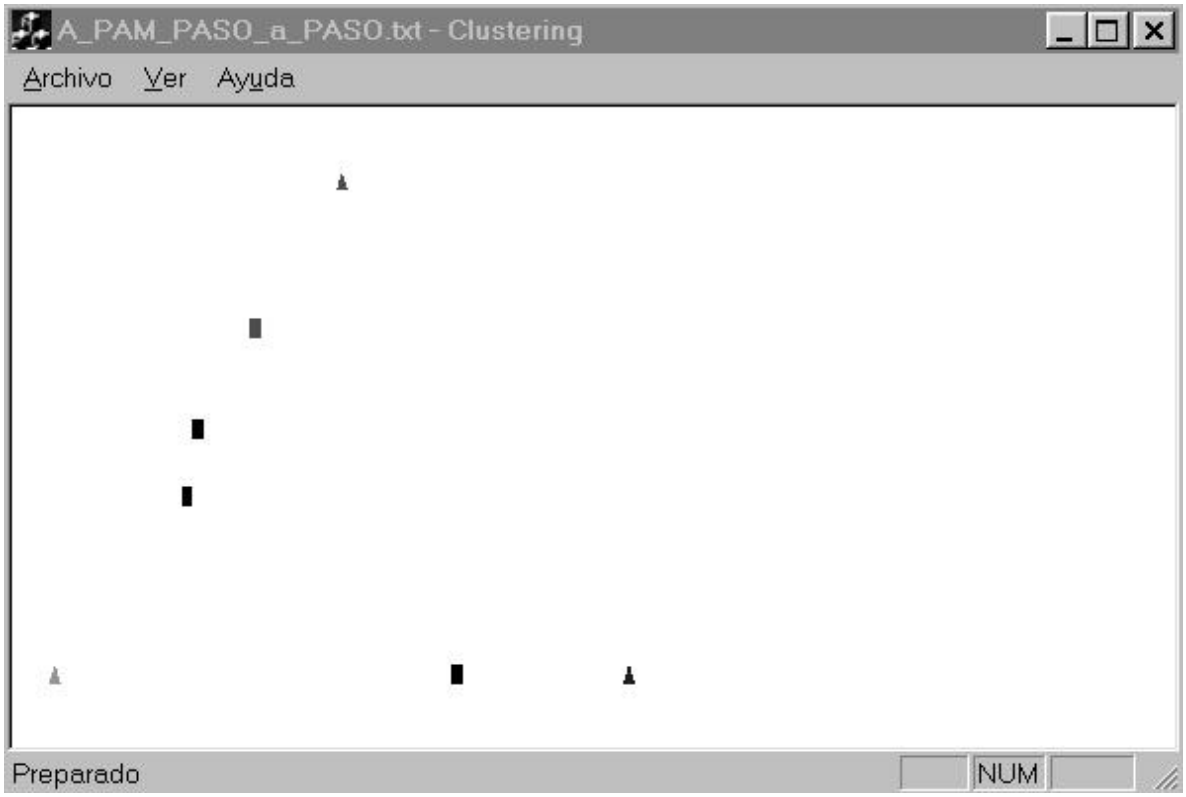
La configuración inicial se presenta a continuación. Es en esta etapa donde el algoritmo define los mediodes iniciales. Por ser el primer paso, estos coinciden con los Depósitos.

A grandes rasgos, PAM funciona de la siguiente manera: realiza una recorrida sobre los Depósitos, creando un cluster a partir de cada uno al mismo tiempo que lo define como mediode del cluster. A continuación, a través de una recorrida sobre los Clientes los asigna al cluster del mediode más cercano, simultáneamente los define como los no mediodes del cluster. Luego itera sobre los mediodes, dentro de esta iteración recorre los no mediodes intentando cambiar el par mediode-no mediode cambiándolos si un costo calculado es adecuado. Al lograr un intercambio recomienza ambas iteraciones desde el principio.



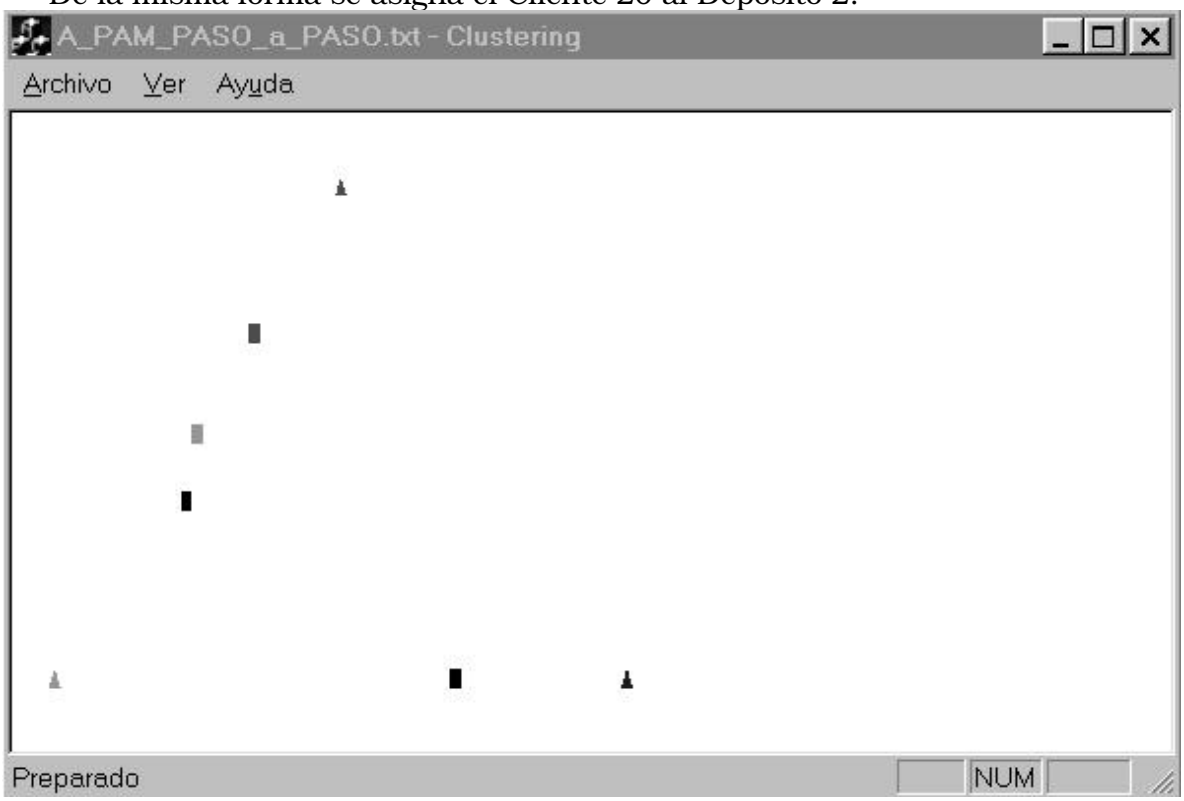
Asignación - Cliente 10:

El primer Cliente se asigna al Depósito más cercano, en este caso el Depósito 1.



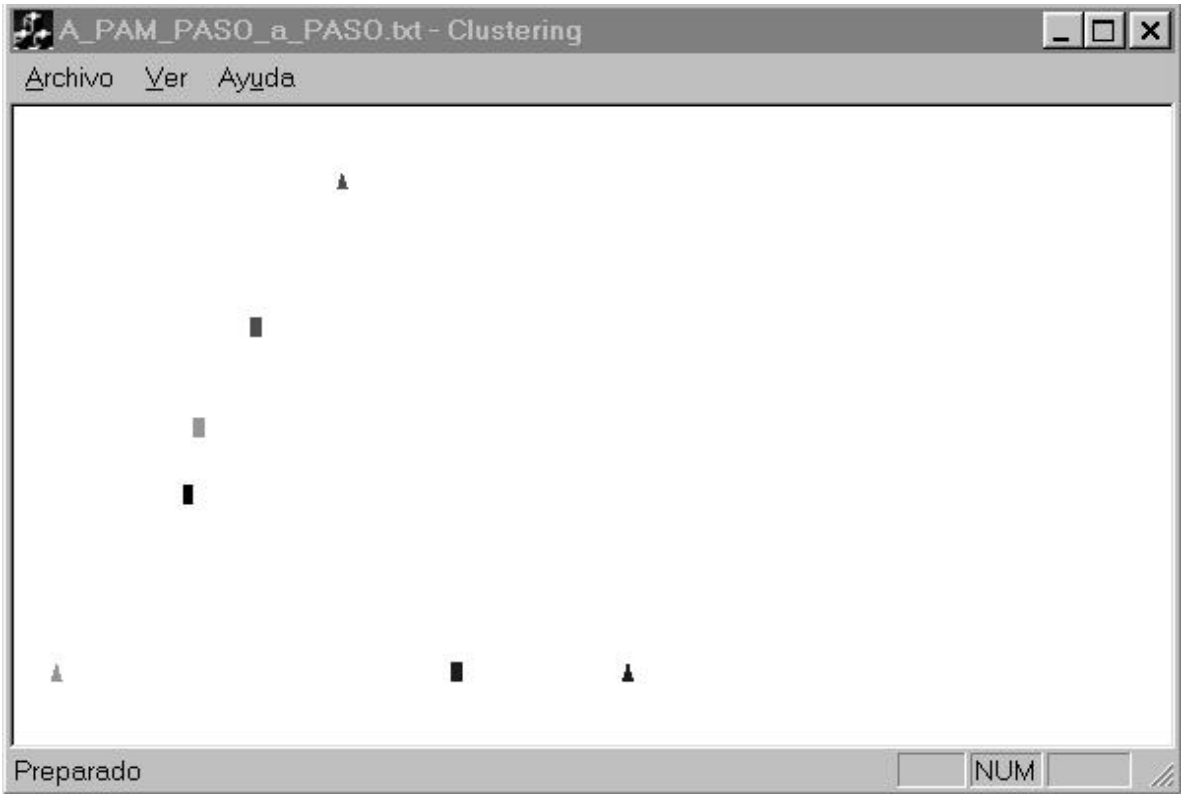
Asignación - Cliente 20:

De la misma forma se asigna el Cliente 20 al Depósito 2.



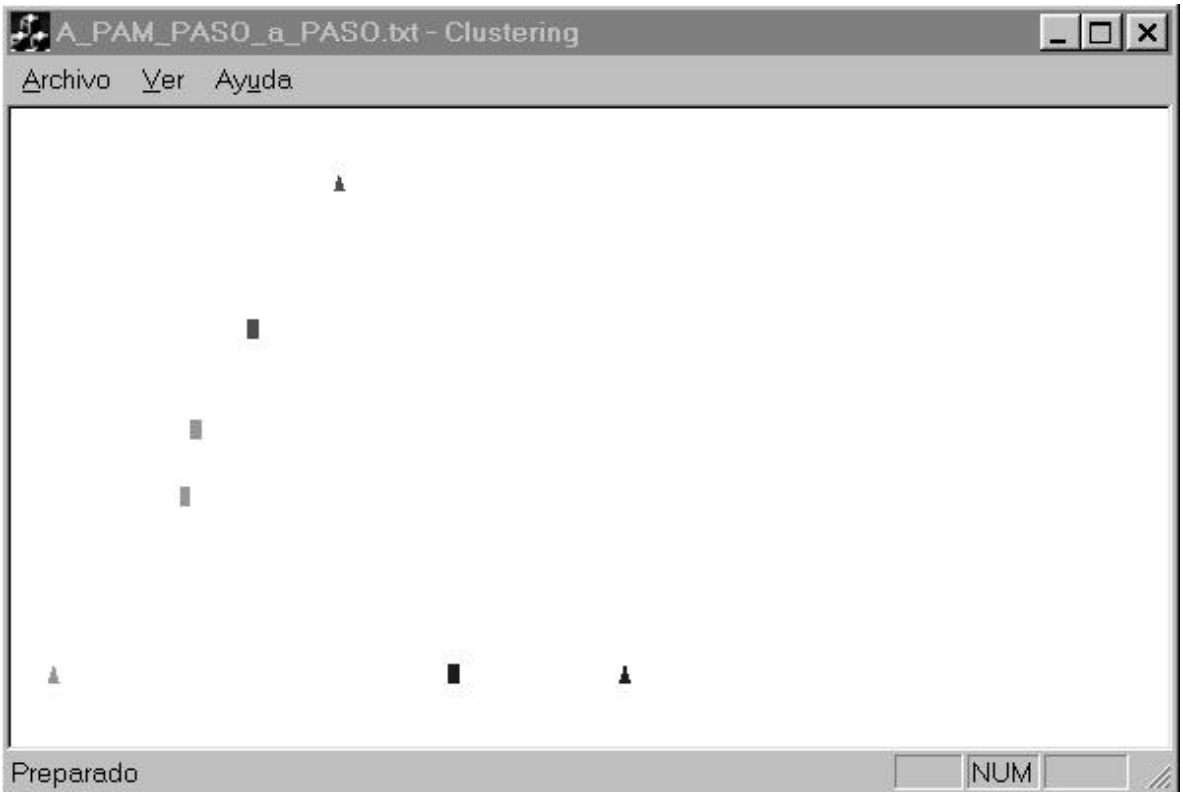
Asignación - Cliente 40:

Se asigna al Cluster del Depósito 3.



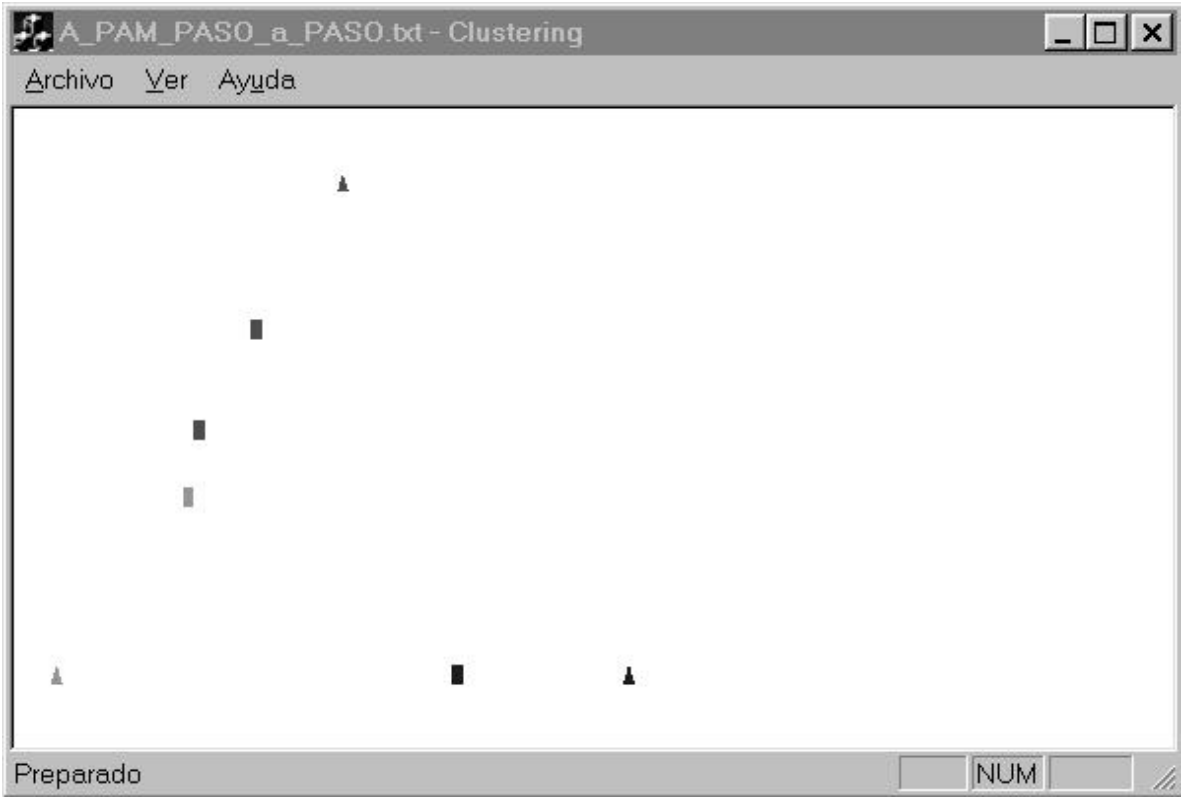
Asignación - Cliente 30:

Como último paso de la asignación inicial de Clientes, este se asigna al Cluster del Depósito 2.



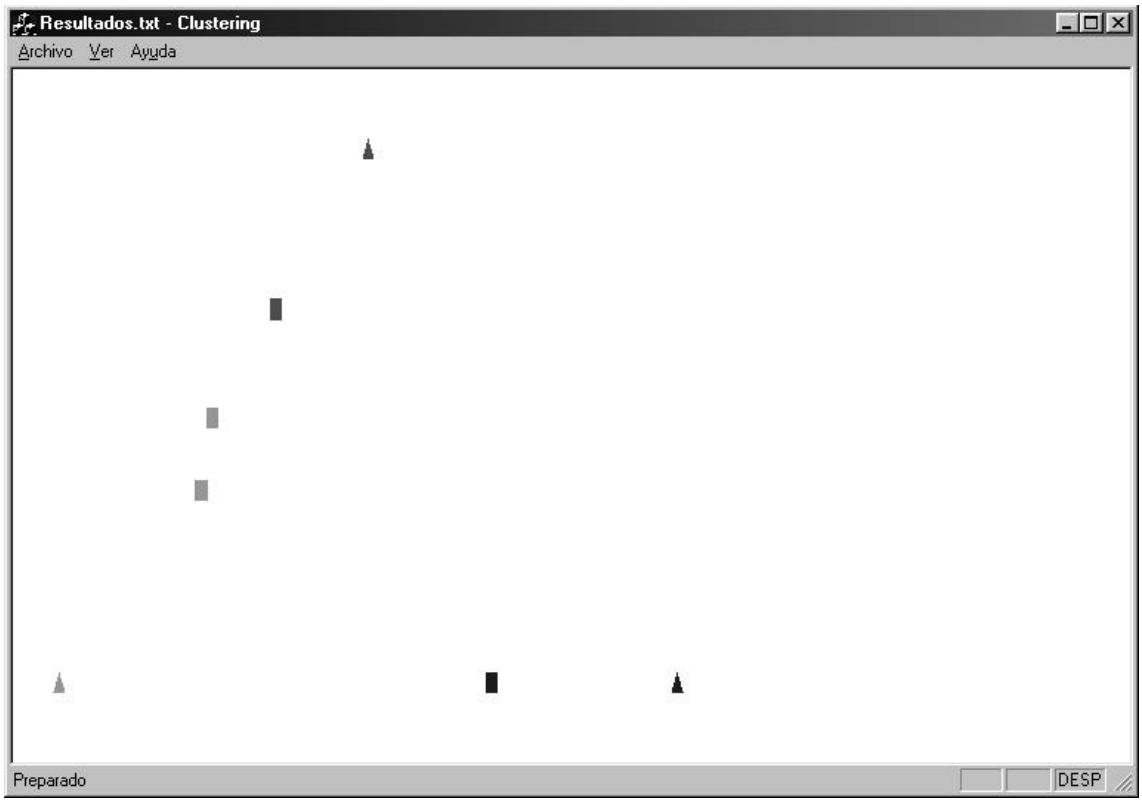
Reasignación:

Como se mencionó anteriormente, luego de la asignación inicial se intentan intercambios mediode- no mediode. Al intentar intercambiar el mediode 1 con el no mediode 10 el costo asociado resultó favorable para el cambio, efectivizándose el mismo



Resultado Final:

Por último y al igual que en el paso anterior, se intercambian el mediode 2 por el no mediode 30.



D.3 K-Means paso a paso

El archivo de Datos de Entrada contendrá la siguiente información:

```
3
9
1 0 0 600 1500
2 10 0 600 1500
3 5 -4 600 1500
10 2 1 600 100 LNC FinLNC
20 4.5 0 600 100 LNC FinLNC
30 7 2 600 100 LNC FinLNC
40 6.5 0.5 600 100 LNC FinLNC
50 7.5 -1.5 600 100 LNC FinLNC
60 7 -3 600 100 LNC FinLNC
70 6 -2 600 100 LNC FinLNC
80 8.5 -2.5 600 100 LNC FinLNC
90 4 -3 600 100 LNC FinLNC
```

La idea es mostrar como puede variar la asignación de un cliente en los distintos "pasos".

Cabe recordar que K-Means se inicia recorriendo los Depósitos creando un cluster a partir de cada uno de ellos e inicializando las medias a partir de estos. Se asignan los Clientes a los clusters con la media mas cercana a ellos (al asignarse un Cliente se recalcula la media del cluster modificado). Mientras existan cambios en las medias se itera sobre los Clientes llevando a cabo los siguientes pasos:

- ✓ se quita el Cliente del cluster al que pertenece
- ✓ se recalcula la media de este cluster
- ✓ se reinserta el Cliente en el cluster con la media mas cercana y
- ✓ se recalcula la media del cluster que recibió al Cliente

Hay que prestar principal atención al Cliente 20, dado que es el que va a cambiar de clusters hasta encontrar su posición definitiva.

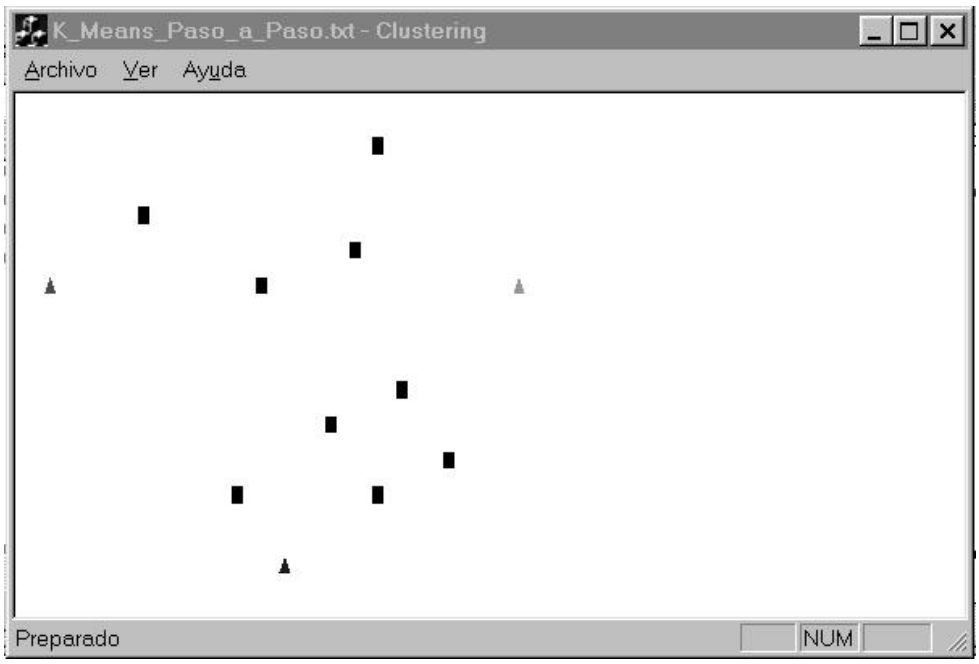
En las nueve pantallas iniciales (tituladas "Asignación – Cliente ...") se muestra la asignación inicial de los Clientes a los distintos Depósitos. Luego, se intenta reasignar cada Cliente de acuerdo a las medias, mostrándose los cambios efectivos. Es posible observar que el Cliente 20 comienza en el Depósito 1 (rojo), pasa al Depósito 2 (verde) y en ese momento el Cliente 50 pasa del Depósito 2 al 3 (azul).

Con esta movida del Cliente 50 se cambia la media del Cluster 3 y en la próxima pasada sobre el Cliente 20, este queda mas cerca de la media del Cluster 3 que de la media de su Cluster actual, generando un nuevo movimiento.

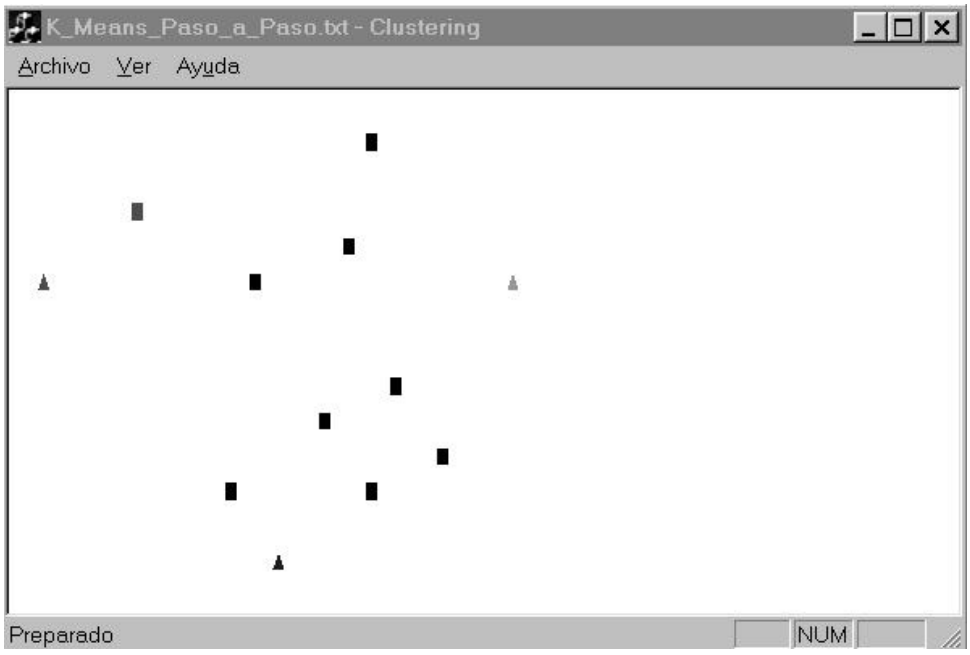
Ejecución:

Factores de Ponderación: espacial=1, temporal =1

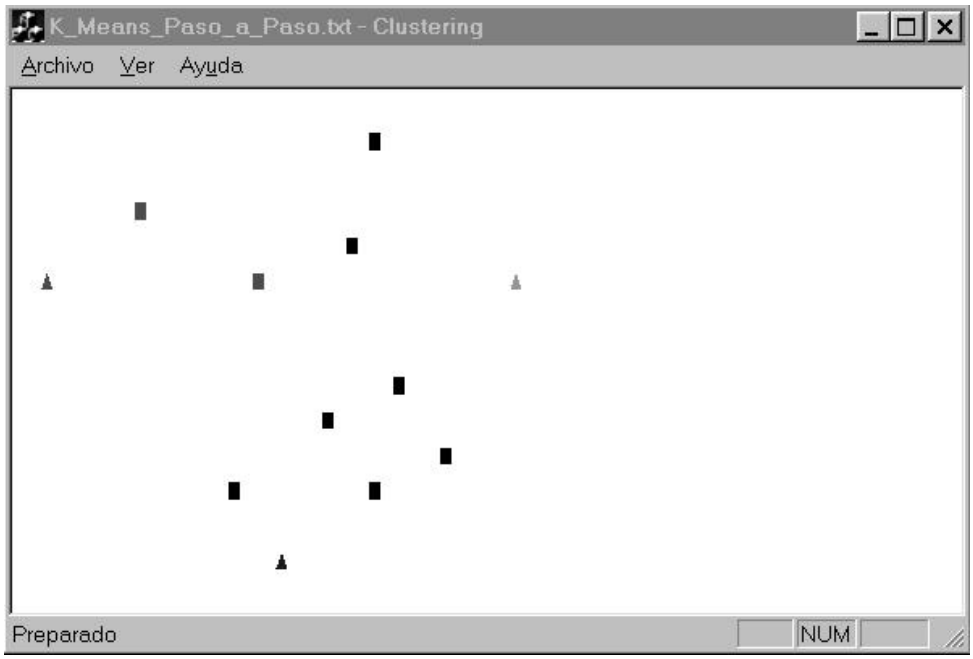
Distribución Clientes y Depósitos:



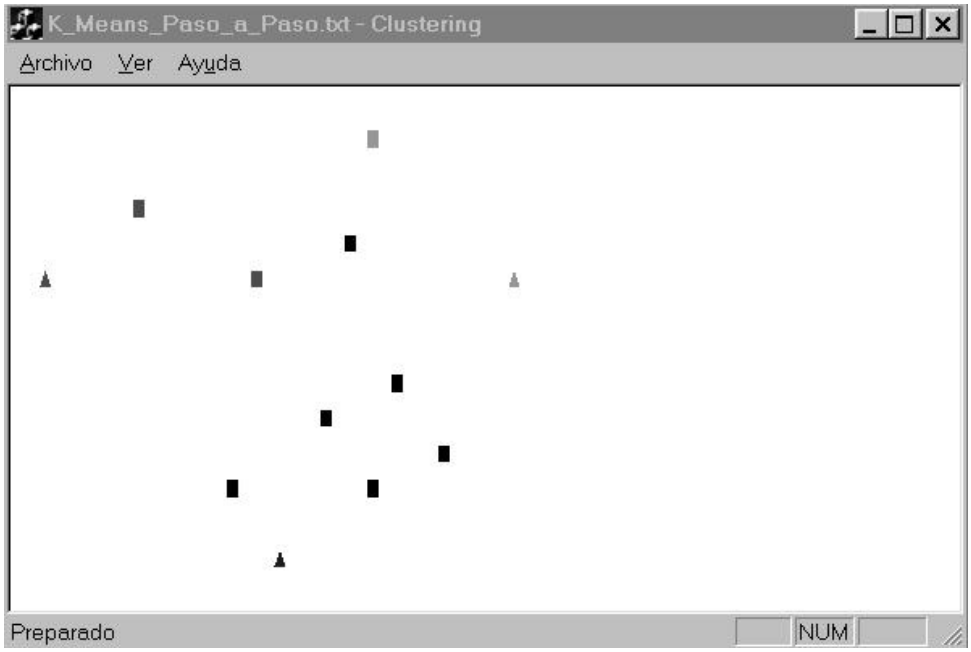
Asignación - Cliente 10



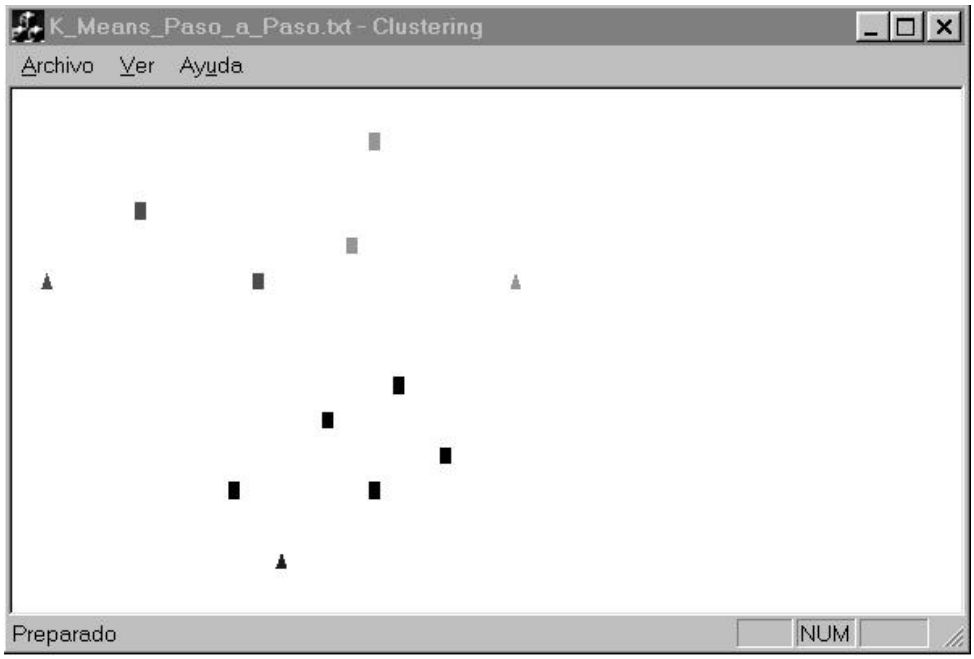
Asignación - Cliente 20



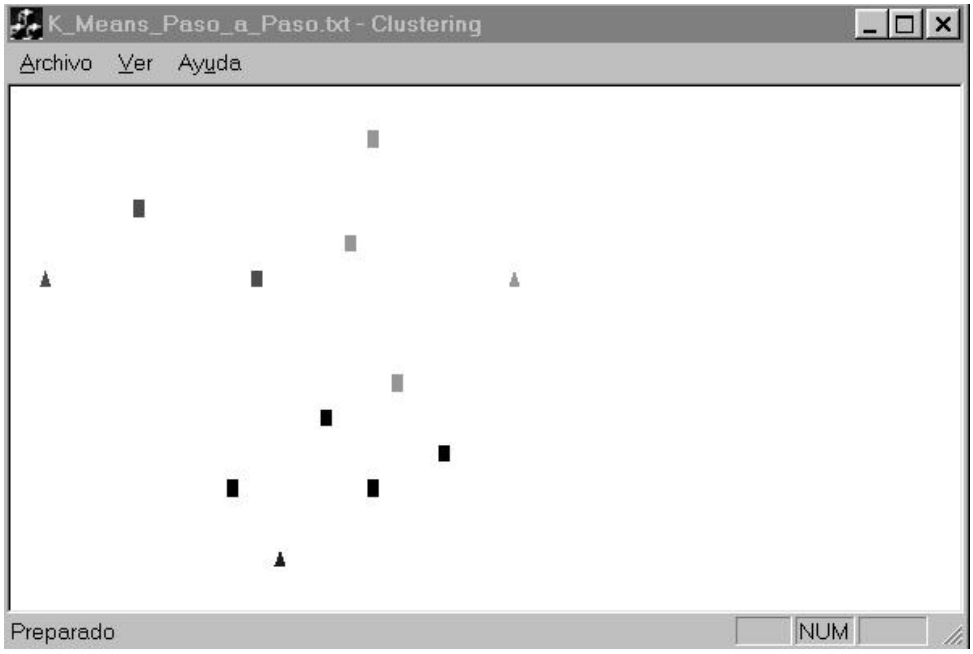
Asignación - Cliente 30



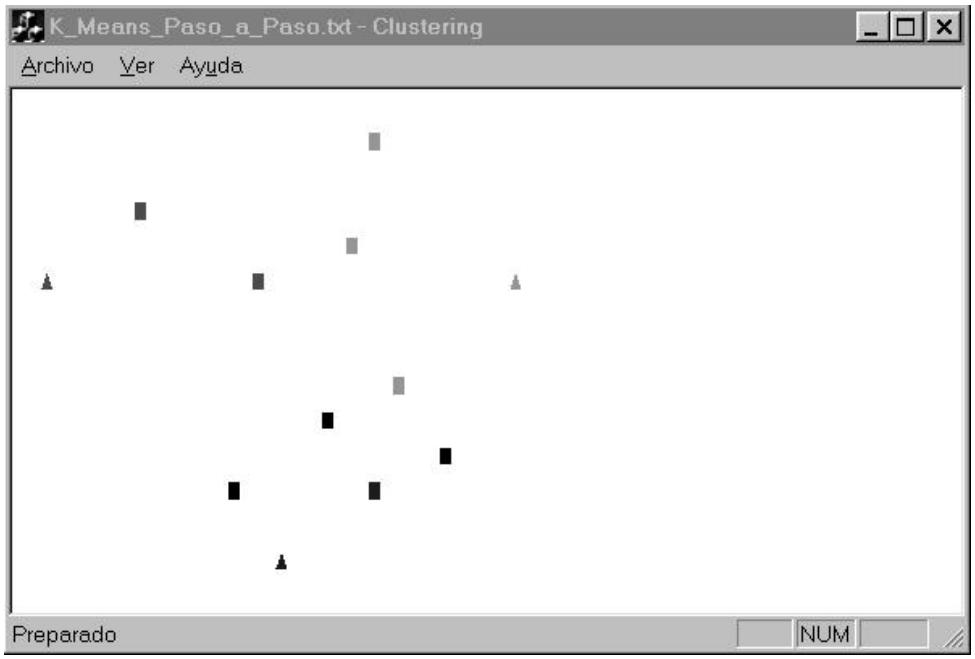
Asignación - Cliente 40



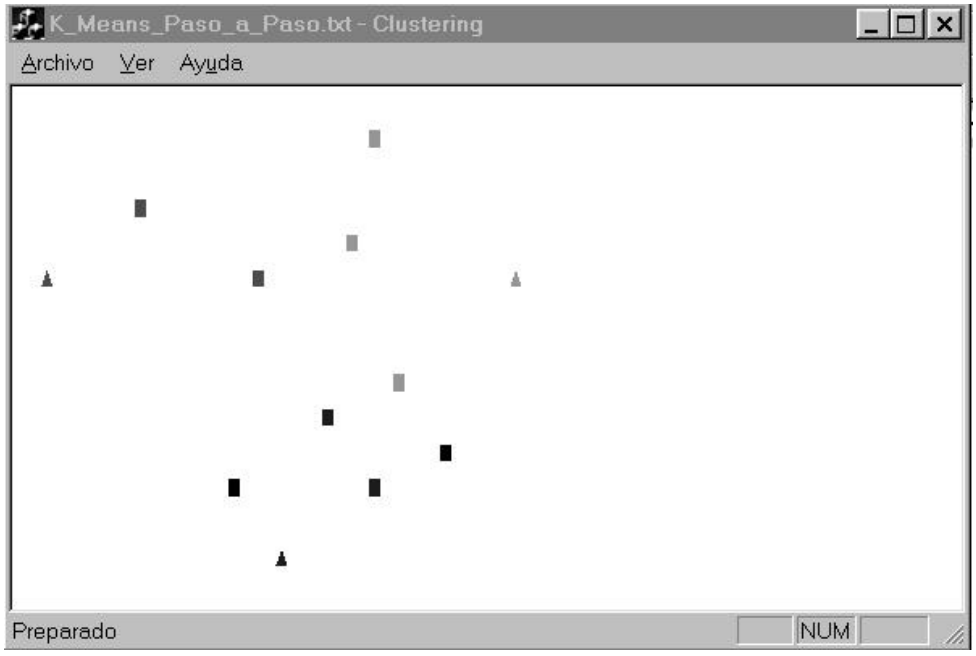
Asignación - Cliente 50



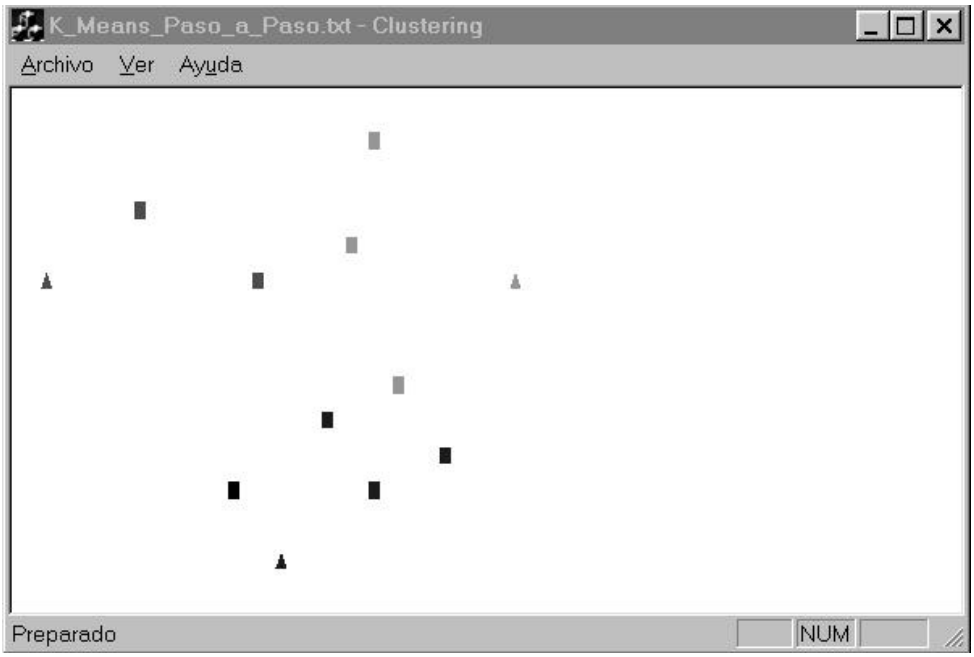
Asignación - Cliente 60



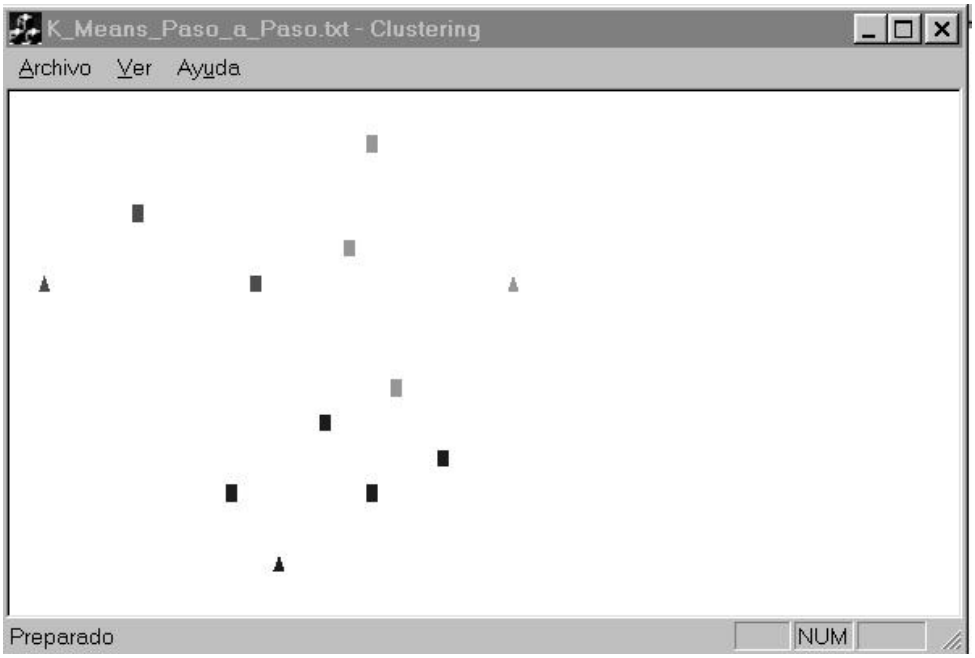
Asignación - Cliente 70



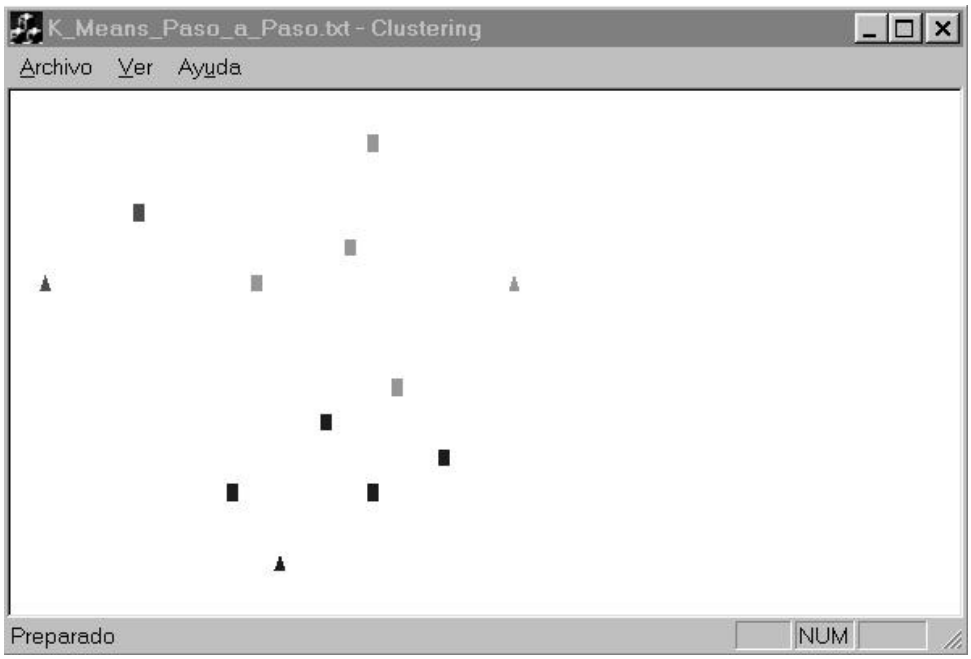
Asignación - Cliente 80



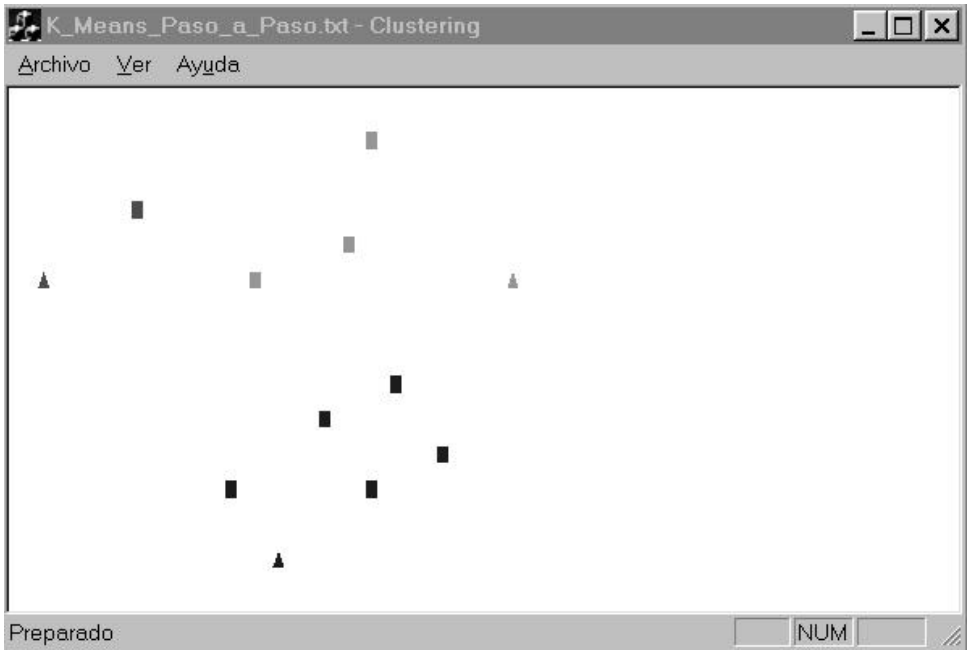
Asignación - Cliente 90



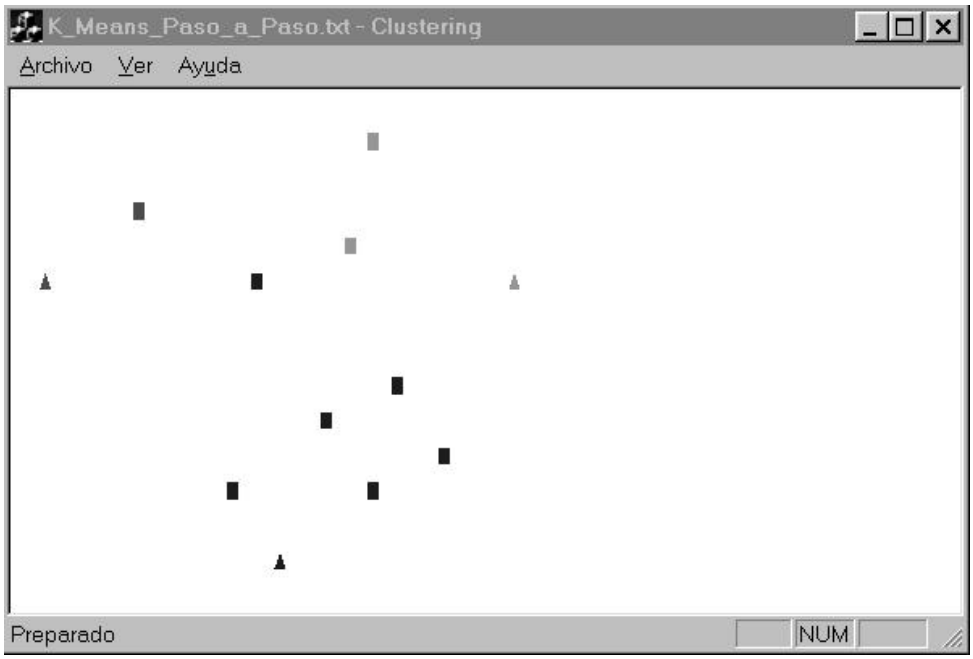
Reasignación - Cliente 20



Reasignación - Cliente 50



Reasignación - Cliente 20 Resultado Final



D.4 Archivos de testeo exhaustivo

Este conjunto de entradas fue aportado por Libertad Tansini, son entradas obtenidas de Internet para testear un MDVRPTW (en nuestro caso fue necesario adecuar el formato para contemplar los requerimientos).

Entrada 1

```
4
48
49   4.163  13.559  600  10000
50  21.387  17.105  600  10000
51 -36.118  49.097  600  10000
52 -31.201   0.235  600  10000
 1 -29.730  64.136  600 12 LNC FinLNC
 2 -30.664   5.463  600  8 LNC FinLNC
 3  51.642   5.469  600 16 LNC FinLNC
 4 -13.171  69.336  600  5 LNC FinLNC
 5 -67.413  68.323  600 12 LNC FinLNC
 6  48.907   6.274  600  5 LNC FinLNC
 7   5.243  22.260  600 13 LNC FinLNC
 8 -65.002  77.234  600 20 LNC FinLNC
 9  -4.175  -1.569  600 13 LNC FinLNC
10  23.029  11.639  600 18 LNC FinLNC
11  25.482   6.287  600  7 LNC FinLNC
12 -42.615 -26.392  600  6 LNC FinLNC
13 -76.672  99.341  600  9 LNC FinLNC
14 -20.673  57.892  600  9 LNC FinLNC
15 -52.039   6.567  600  4 LNC FinLNC
16 -41.376  50.824  600 25 LNC FinLNC
17 -91.943  27.588  600  5 LNC FinLNC
18 -65.118  30.212  600 17 LNC FinLNC
19  18.597  96.716  600  3 LNC FinLNC
20 -40.942  83.209  600 16 LNC FinLNC
21 -37.756 -33.325  600 25 LNC FinLNC
22  23.767  29.083  600 21 LNC FinLNC
23 -43.030  20.453  600 14 LNC FinLNC
24 -35.297 -24.896  600 19 LNC FinLNC
25 -54.755  14.368  600 14 LNC FinLNC
26 -49.329  33.374  600  6 LNC FinLNC
27  57.404  23.822  600 16 LNC FinLNC
28 -22.754  55.408  600  9 LNC FinLNC
29 -56.622  73.340  600 20 LNC FinLNC
30 -38.562  -3.705  600 13 LNC FinLNC
31 -16.779  19.537  600 10 LNC FinLNC
32 -11.560  11.615  600 16 LNC FinLNC
33 -46.545  97.974  600 19 LNC FinLNC
34  16.229   9.320  600 22 LNC FinLNC
35   1.294   7.349  600 14 LNC FinLNC
36 -26.404  29.529  600 10 LNC FinLNC
37   4.352  14.685  600 11 LNC FinLNC
38 -50.665 -23.126  600 15 LNC FinLNC
39 -22.833  -9.814  600 13 LNC FinLNC
40 -71.100 -18.616  600 15 LNC FinLNC
41  -7.849  32.074  600  8 LNC FinLNC
42  11.877 -24.933  600 22 LNC FinLNC
43 -18.927 -23.730  600 24 LNC FinLNC
44 -11.920  11.755  600  3 LNC FinLNC
45  29.840  11.633  600 25 LNC FinLNC
46  12.268 -55.811  600 19 LNC FinLNC
47 -37.933 -21.613  600 21 LNC FinLNC
48  42.883  -2.966  600 10 LNC FinLNC
```

Entrada 2

```
4
96
 97   6.229  10.590 600   1420
 98  32.663  44.730600   1200
 99  48.807  48.792600   1180
100  33.179  -4.968600   1070
  1  33.588  30.750 600     4       LNC FinLNC
  2  48.828  65.314 600    12       LNC FinLNC
  3  86.176  59.344 600     3       LNC FinLNC
  4  39.270 -33.057 600    15       LNC FinLNC
  5 -23.370  86.853 600    13       LNC FinLNC
  6  48.132  95.593 600    20       LNC FinLNC
  7 -16.357  93.311 600    21       LNC FinLNC
  8 -57.703 -65.601 600     4       LNC FinLNC
  9   7.147  32.684  600    25       LNC FinLNC
 10  42.950  68.701600    21       LNC FinLNC
 11  37.085  -2.112 600     7       LNC FinLNC
 12  77.759  55.817600     3       LNC FinLNC
 13 -17.462 -56.567        600     9       LNC FinLNC
 14  58.575  59.888600     5       LNC FinLNC
 15  57.776  15.344600    20       LNC FinLNC
 16 -22.327  36.072600    21       LNC FinLNC
 17  -7.080  30.493 600    19       LNC FinLNC
 18  55.658  60.425600     2       LNC FinLNC
 19 -14.307  11.456600    16       LNC FinLNC
 20 -29.724  24.268600    18       LNC FinLNC
 21  43.219   0.739 600    24       LNC FinLNC
 22  45.184  35.474600    24       LNC FinLNC
 23  64.484   2.240 600     8       LNC FinLNC
 24  55.078  72.241600    16       LNC FinLNC
 25  16.925  15.741600    19       LNC FinLNC
 26  45.038  -3.723 600    12       LNC FinLNC
 27 -76.782   5.939 600     9       LNC FinLNC
 28  36.169   0.256 600     3       LNC FinLNC
 29  29.218   8.936 600    15       LNC FinLNC
 30  65.057   5.225 600     4       LNC FinLNC
 31  42.175 -22.284600    24       LNC FinLNC
 32  25.574  31.726600    22       LNC FinLNC
 33  31.561  37.262600    14       LNC FinLNC
 34  66.498 -54.169600    11       LNC FinLNC
 35  46.576 -17.938600    20       LNC FinLNC
 36  65.063  40.875600     5       LNC FinLNC
 37  -2.716  24.768 600    11       LNC FinLNC
 38 -40.002   3.870 600    12       LNC FinLNC
 39 -73.505  57.043600     7       LNC FinLNC
 40  81.146 -25.714600     1       LNC FinLNC
 41  12.006  -7.965 600    23       LNC FinLNC
 42  42.761  38.092600    13       LNC FinLNC
 43   3.857 -23.181 600     4       LNC FinLNC
 44  -7.367  24.390 600    23       LNC FinLNC
 45  35.944 -11.835600    17       LNC FinLNC
 46  52.075   9.692 600    13       LNC FinLNC
 47  30.725  30.701600     9       LNC FinLNC
 48  41.223  77.924600    10       LNC FinLNC
 49  68.884 -40.546600    22       LNC FinLNC
 50  76.312  86.670600    21       LNC FinLNC
 51  63.934  78.540600    13       LNC FinLNC
 52  29.150  -9.961 600    25       LNC FinLNC
 53  85.522  39.954600     8       LNC FinLNC
 54  31.775   3.870 600    20       LNC FinLNC
 55 -20.544  19.086600     3       LNC FinLNC
```

56	55.353	43.817	600	4	LNC FinLNC
57	35.406	10.278	600	13	LNC FinLNC
58	25.464	-0.287	600	18	LNC FinLNC
59	12.396	0.244	600	1	LNC FinLNC
60	18.359	20.917	600	10	LNC FinLNC
61	27.960	-15.039	600	16	LNC FinLNC
62	-3.192	19.879	600	9	LNC FinLNC
63	9.332	-41.351	600	13	LNC FinLNC
64	-20.581	38.177	600	23	LNC FinLNC
65	27.820	28.729	600	14	LNC FinLNC
66	59.027	42.310	600	5	LNC FinLNC
67	56.311	58.734	600	1	LNC FinLNC
68	-51.654	-0.342	600	6	LNC FinLNC
69	2.576	32.721	600	20	LNC FinLNC
70	15.131	-3.046	600	10	LNC FinLNC
71	66.595	34.937	600	19	LNC FinLNC
72	19.476	20.142	600	24	LNC FinLNC
73	-1.172	20.648	600	3	LNC FinLNC
74	-21.204	13.660	600	5	LNC FinLNC
75	98.846	1.257	600	13	LNC FinLNC
76	82.593	72.003	600	5	LNC FinLNC
77	30.017	-30.896	600	4	LNC FinLNC
78	33.228	41.663	600	12	LNC FinLNC
79	39.490	-8.539	600	11	LNC FinLNC
80	54.498	70.813	600	20	LNC FinLNC
81	3.058	10.248	600	13	LNC FinLNC
82	51.831	-16.870	600	1	LNC FinLNC
83	76.416	11.346	600	3	LNC FinLNC
84	5.243	0.238	600	2	LNC FinLNC
85	50.592	34.247	600	23	LNC FinLNC
86	-5.231	-2.081	600	8	LNC FinLNC
87	24.194	25.836	600	25	LNC FinLNC
88	33.630	37.585	600	18	LNC FinLNC
89	31.445	-7.001	600	4	LNC FinLNC
90	30.707	-28.168	600	5	LNC FinLNC
91	49.860	1.038	600	21	LNC FinLNC
92	-35.767	14.142	600	23	LNC FinLNC
93	-29.309	4.889	600	19	LNC FinLNC
94	19.049	41.974	600	2	LNC FinLNC
95	27.600	13.934	600	21	LNC FinLNC
96	52.832	50.684	600	10	LNC FinLNC

Entrada 3

```
4
144
145 -40.082 -24.780  540 1250
146  24.292 -27.704  540 1120
147 -45.877  41.092  540 1470
148 -11.896  15.875  540 999
  1 -55.280 -24.371    540  9   LNC 144 146 FinLNC
  2 -48.297  53.314    540 22   LNC 145 147 FinLNC
  3 -49.072 -38.489    540 10   LNC 146 FinLNC
  4  25.311 -18.561    540 24   LNC FinLNC
  5 -24.469  -3.815    540 25   LNC FinLNC
  6  24.591 -17.896    540  6   LNC FinLNC
  7 -10.419  60.364    540 22   LNC FinLNC
  8  38.177 -35.175    540  1   LNC 145 FinLNC
  9 -68.280  93.073    540 19   LNC FinLNC
 10 -30.792 -57.336    540 19   LNC FinLNC
 11 -37.061 -12.122    540  5   LNC 147 FinLNC
 12 -15.741 -47.638    540  9   LNC FinLNC
 13 -63.379 -22.919    540  8   LNC FinLNC
 14 -43.109 -43.439    540 14   LNC 148 FinLNC
 15 -25.623  13.599    540  4   LNC FinLNC
 16  -2.625  16.632    540  7   LNC FinLNC
 17 -41.577 -28.497    540 10   LNC 145 FinLNC
 18   2.081  12.885    540 22   LNC FinLNC
 19   3.925 -47.845    540 17   LNC FinLNC
 20 -83.295  26.324    540 10   LNC 146 FinLNC
 21  -6.458  26.355    540  8   LNC FinLNC
 22   3.290   6.732    540 14   LNC FinLNC
 23 -34.869  30.426    540 12   LNC 147 FinLNC
 24  15.546 -36.273    540 17   LNC FinLNC
 25  12.842   5.127    540 19   LNC FinLNC
 26 -48.450 -24.426    540 25   LNC 148 FinLNC
 27 -88.538 -10.461    540 25   LNC FinLNC
 28 -29.773   0.995    540  6   LNC FinLNC
 29   9.827  38.416    540 21   LNC 145 FinLNC
 30   1.410  92.938    540  3   LNC FinLNC
 31 -79.303  15.381    540 19   LNC FinLNC
 32 -30.652  40.063    540 16   LNC 146 FinLNC
 33  18.927  21.637    540 21   LNC FinLNC
 34  45.087 -59.906    540 17   LNC FinLNC
 35 -34.949  -3.815    540 19   LNC 147 FinLNC
 36  81.201 -74.744    540  6   LNC FinLNC
 37  15.594 -28.455    540 10   LNC FinLNC
 38 -72.192  29.547    540 22   LNC 148 FinLNC
 39 -42.914 -22.675    540 17   LNC FinLNC
 40 -76.392 -57.489    540 16   LNC FinLNC
 41 -28.540  12.073    540  5   LNC 145 FinLNC
 42 -61.389  26.526    540  1   LNC FinLNC
 43  -8.472  12.616    540 21   LNC FinLNC
 44 -61.475  33.392    540 20   LNC FinLNC
 45 -34.674 -27.588    540  6   LNC 146 FinLNC
 46  10.315 -12.518    540  1   LNC FinLNC
 47 -83.014  77.002    540 11   LNC FinLNC
 48 -40.417  49.988    540 22   LNC 147 FinLNC
 49  83.832  33.905    540 25   LNC FinLNC
 50 -20.563 -75.830    540 18   LNC FinLNC
 51  15.442 -18.719    540  9   LNC FinLNC
 52 -59.937 -65.802    540 23   LNC 148 FinLNC
 53   5.151  47.815    540 17   LNC FinLNC
 54  46.008  14.990    540  4   LNC FinLNC
 55  82.977  -1.660    540  1   LNC 145 FinLNC
 56  39.893 -38.916    540 14   LNC FinLNC
 57  90.839 -83.539    540 11   LNC FinLNC
 58   7.068   0.067    540  7   LNC FinLNC
 59  18.958  40.088    540 21   LNC 146 FinLNC
 60 -80.487  58.209    540  3   LNC FinLNC
 61  64.459 -62.946    540 18   LNC FinLNC
 62 -43.616 -26.117    540  5   LNC 147 FinLNC
```

```
 63 -18.408  10.303   540  6   LNC FinLNC
 64 -43.921  63.251   540 22   LNC FinLNC
 65  11.353  26.221   540 13   LNC 148 FinLNC
 66  -6.995  38.239   540 17   LNC FinLNC
 67  74.084 -34.216   540  5   LNC FinLNC
 68  14.301  -0.800   540  7   LNC 145 FinLNC
 69 -94.141 -48.779   540  1   LNC FinLNC
 70  15.332 -42.169   540  3   LNC FinLNC
 71 -46.637 -16.461   540 10   LNC FinLNC
 72  -0.958 -81.000   540  5   LNC 146 FinLNC
 73 -37.445 -42.505   540  1   LNC FinLNC
 74  -3.680 -69.073   540 17   LNC FinLNC
 75  34.894 -20.898   540 14   LNC 147 FinLNC
 76  45.544  51.410   540  4   LNC FinLNC
 77 -31.927  88.239   540 10   LNC FinLNC
 78 -13.580 -26.959   540 25   LNC 148 FinLNC
 79  12.653 -48.340   540 11   LNC FinLNC
 80 -46.179 -38.269   540 16   LNC FinLNC
 81 -29.620  35.083   540 25   LNC 145 FinLNC
 82  13.696  43.988   540 20   LNC FinLNC
 83  38.342 -46.338   540  3   LNC FinLNC
 84 -26.227  24.506   540  7   LNC 146 FinLNC
 85  90.009 -43.640   540  2   LNC FinLNC
 86  -6.995  22.522   540  6   LNC FinLNC
 87 -29.993 -28.851   540 14   LNC 147 FinLNC
 88 -56.268 -64.740   540 14   LNC FinLNC
 89 -87.415  90.796   540  2   LNC FinLNC
 90  -5.762  54.034   540 15   LNC 148 FinLNC
 91 -34.845 -27.185   540 22   LNC FinLNC
 92 -16.479  16.180   540  4   LNC FinLNC
 93  23.712  29.492   540 22   LNC 145 FinLNC
 94 -40.991  46.594   540 19   LNC FinLNC
 95 -53.516 -21.619   540  5   LNC FinLNC
 96 -17.560  37.494   540 10   LNC 146 FinLNC
 97 -37.311  56.836   540  4   LNC FinLNC
 98  71.515  22.510   540  7   LNC FinLNC
 99 -41.864  27.710   540 11   LNC 147 FinLNC
100  14.819 -82.117   540 14   LNC FinLNC
101  30.316 -55.322   540  2   LNC FinLNC
102 -39.172  47.998   540 12   LNC 148 FinLNC
103 -94.415  15.192   540 12   LNC FinLNC
104  -7.806  11.273   540 24   LNC 145 FinLNC
105  28.571 -49.908   540  8   LNC FinLNC
106 -20.654  24.554   540  4   LNC FinLNC
107 -30.963 -18.903   540 25   LNC FinLNC
108   1.721 -20.447   540  6   LNC 146 FinLNC
109 -31.012 -21.106   540  9   LNC FinLNC
110 -85.718 -28.015   540 23   LNC FinLNC
111  58.826 -63.043   540 15   LNC FinLNC
112 -15.753 -52.686   540  7   LNC 147 FinLNC
113 -71.661  51.184   540  6   LNC FinLNC
114 -92.633  -6.598   540  5   LNC FinLNC
115  33.508 -49.255   540 12   LNC FinLNC
116  59.314  54.095   540  3   LNC 148 145 146 FinLNC
117  30.737 -28.436   540 22   LNC FinLNC
118 -55.896  18.457   540 18   LNC FinLNC
119 -89.960 -39.532   540 16   LNC FinLNC
120  36.438  20.819   540  5   LNC 147 FinLNC
121 -53.674 -56.427   540  1   LNC FinLNC
122 -31.866  32.538   540  8   LNC FinLNC
123   5.658  12.756   540 20   LNC FinLNC
124 -52.905  42.804   540 17   LNC 146 FinLNC
125  47.131 -45.465   540 11   LNC FinLNC
126  30.792   9.869   540  8   LNC FinLNC
127 -45.209 -56.293   540 15   LNC FinLNC
128  10.919  13.306   540 16   LNC FinLNC
129 -32.990 -34.003   540  6   LNC 145 FinLNC
130 -40.869  30.579   540 24   LNC FinLNC
131 -69.348  80.408   540 10   LNC FinLNC
132 -58.966 -29.541   540  4   LNC FinLNC
133 -36.847 -33.710   540 10   LNC 145 146 147 FinLNC
134 -21.820  65.369   540 17   LNC FinLNC
135  22.980 -41.010   540  5   LNC FinLNC
```

136 -77.557 59.814 540 14 LNC FinLNC
137 -47.205 67.725 540 16 LNC 148 FinLNC
138 -21.606 35.168 540 11 LNC FinLNC
139 -55.676 -31.384 540 25 LNC FinLNC
140 -85.229 88.788 540 4 LNC FinLNC
141 -3.918 -63.525 540 12 LNC 146 FinLNC
142 -58.557 59.375 540 19 LNC FinLNC
143 -24.542 14.935 540 6 LNC FinLNC
144 -48.779 16.968 540 23 LNC 145 FinLNC

Entrada 4

```
6
72
 73  42.395  -8.344  680         720
 74 -42.175 -14.554  680         650
 75  16.034  40.726  680         480
 76 -14.639  29.633  680         520
 77  16.049  -3.934  680         560
 78  46.112  12.430  680         610
  1 -92.700 -59.180    680  20  LNC FinLNC
  2  71.179  12.543    680   6  LNC FinLNC
  3  31.537  66.638    680  19  LNC FinLNC
  4  -4.694  25.537    680  10  LNC FinLNC
  5 -30.194  67.773    680  18  LNC FinLNC
  6  12.677 -57.471    680   1  LNC FinLNC
  7 -32.355 -20.966    680  15  LNC FinLNC
  8  19.910  48.975    680  23  LNC FinLNC
  9  13.202 -19.135    680  13  LNC FinLNC
 10  54.877 -41.168    680  12  LNC FinLNC
 11  15.063 -25.171    680   7  LNC FinLNC
 12 -50.598 -16.418    680  25  LNC FinLNC
 13 -29.730  17.078    680   5  LNC FinLNC
 14  17.542   1.575    680   1  LNC FinLNC
 15  11.127  77.216    680  25  LNC FinLNC
 16  33.752  71.259    680  10  LNC FinLNC
 17 -56.012 -10.394    680   2  LNC FinLNC
 18  57.874 -16.290    680   7  LNC FinLNC
 19  10.718 -18.787    680  11  LNC FinLNC
 20  53.088 -18.750    680   4  LNC FinLNC
 21   1.569   7.532    680   6  LNC FinLNC
 22  31.531  48.944    680  15  LNC FinLNC
 23 -66.833 -37.854    680   7  LNC FinLNC
 24 -70.740  62.244    680   8  LNC FinLNC
 25  32.538  23.096    680  10  LNC FinLNC
 26 -51.453 -36.444    680   9  LNC FinLNC
 27  36.456 -22.638    680   4  LNC FinLNC
 28 -31.207  43.494    680   6  LNC FinLNC
 29 -10.388  34.491    680  22  LNC FinLNC
 30  14.722 -10.834    680  17  LNC FinLNC
 31  47.095 -21.387    680   9  LNC FinLNC
 32  43.781  34.766    680  25  LNC FinLNC
 33  53.546 -67.487    680  10  LNC FinLNC
 34  26.801  46.515    680  18  LNC FinLNC
 35  63.385  11.981    680  21  LNC FinLNC
 36  47.192  -5.475    680  10  LNC FinLNC
 37 -16.315 -11.267    680  23  LNC FinLNC
 38  78.900  17.651    680  23  LNC FinLNC
 39  79.822  22.272    680  11  LNC FinLNC
 40  12.878  16.919    680   1  LNC FinLNC
 41 -67.981  -3.754    680  23  LNC FinLNC
 42   9.198 -18.597    680  16  LNC FinLNC
 43 -35.950 -19.141    680  10  LNC FinLNC
 44  28.766  45.276    680  12  LNC FinLNC
 45  11.469  68.231    680  12  LNC FinLNC
 46 -22.760  45.496    680   3  LNC FinLNC
 47 -65.674 -23.120    680  22  LNC FinLNC
 48   7.239   1.599    680  21  LNC FinLNC
 49 -29.785 -11.285    680  13  LNC FinLNC
 50 -89.050  16.211    680  15  LNC FinLNC
 51 -46.887  -3.363    680  13  LNC FinLNC
 52 -14.972  30.621    680  20  LNC FinLNC
 53 -17.035  49.774    680  10  LNC FinLNC
 54  31.635  53.619    680  25  LNC FinLNC
 55  -3.577  13.342    680   7  LNC FinLNC
 56  33.008  58.960    680  15  LNC FinLNC
 57 -92.950  63.263    680   2  LNC FinLNC
 58  -9.137 -22.931    680  23  LNC FinLNC
 59 -39.960   6.195    680   5  LNC FinLNC
 60  28.430 -19.214    680  25  LNC FinLNC
```

61	-28.540	-3.485	680	9	LNC	FinLNC
62	31.415	36.859	680	2	LNC	FinLNC
63	-49.426	60.602	680	24	LNC	FinLNC
64	-72.827	-27.765	680	13	LNC	FinLNC
65	60.083	-45.905	680	20	LNC	FinLNC
66	10.870	-3.900	680	13	LNC	FinLNC
67	25.122	7.672	680	15	LNC	FinLNC
68	-46.997	-17.474	680	4	LNC	FinLNC
69	16.058	33.020	680	20	LNC	FinLNC
70	25.409	-11.700	680	8	LNC	FinLNC
71	68.323	-5.145	680	19	LNC	FinLNC
72	-13.104	62.158	680	20	LNC	FinLNC

Entrada 5

```
6
144
145  -3.799  44.290  575        722
146  14.233  21.173  575        728
147 -23.334 -28.397  575        780
148  10.065   1.822  575        870
149 -49.115 -43.549  575        978
150 -21.054   4.144  575       1024
  1 -40.289 -42.303    575  20  LNC FinLNC
  2 -64.709 -17.389    575  10  LNC FinLNC
  3   5.060 -14.349    575   8  LNC FinLNC
  4  72.095  20.233    575   6  LNC FinLNC
  5   2.594 -15.002    575   1  LNC FinLNC
  6 -24.176 -72.894    575  19  LNC FinLNC
  7 -13.190  66.498    575  13  LNC FinLNC
  8  33.191  13.690    575  24  LNC FinLNC
  9  66.827 -30.554    575  15  LNC FinLNC
 10  19.934  35.883    575  13  LNC FinLNC
 11 -27.771  32.269    575  10  LNC FinLNC
 12  51.154 -16.827    575   4  LNC FinLNC
 13 -19.214  23.749    575  12  LNC FinLNC
 14 -46.643 -52.954    575  21  LNC FinLNC
 15 -10.590 -16.626    575   2  LNC FinLNC
 16  35.150  12.622    575  10  LNC FinLNC
 17 -37.378  12.256    575  20  LNC FinLNC
 18  82.794  25.836    575   9  LNC FinLNC
 19   8.936  -3.723    575  24  LNC FinLNC
 20 -56.659  25.555    575  19  LNC FinLNC
 21 -52.020 -40.137    575  13  LNC FinLNC
 22  39.270  12.787    575  14  LNC FinLNC
 23  19.757  21.228    575  17  LNC FinLNC
 24 -23.071 -12.659    575  25  LNC FinLNC
 25  15.686  52.374    575   1 LNC FinLNC
 26  58.771  -7.056    575   9  LNC FinLNC
 27 -33.087  72.266    575  12  LNC FinLNC
 28   2.551  46.851    575  22  LNC FinLNC
 29 -18.719  73.126    575   2  LNC FinLNC
 30 -67.627 -20.679    575  21  LNC FinLNC
 31 -76.862 -11.365    575   8  LNC FinLNC
 32 -24.194 -38.263    575  19  LNC FinLNC
 33 -20.374 -31.903    575  24  LNC FinLNC
 34   9.290  39.618    575  24  LNC FinLNC
 35  26.044  35.974    575   2  LNC FinLNC
 36 -57.385 -45.264    575  20  LNC FinLNC
 37   1.489 -62.592    575  20  LNC FinLNC
 38  34.937  13.672    575   5  LNC FinLNC
 39 -22.290  45.648    575  17  LNC FinLNC
 40  -5.920  44.061    575   4  LNC FinLNC
 41 -38.446 -23.541    575  21  LNC FinLNC
 42  28.894  33.936    575   3  LNC FinLNC
 43 -35.388  -5.341    575  16  LNC FinLNC
 44 -19.501  13.519    575  14  LNC FinLNC
 45  -9.216 -30.988    575   6  LNC FinLNC
 46 -22.131 -17.065    575   5  LNC FinLNC
 47 -22.827  36.810    575  12  LNC FinLNC
 48 -61.365 -18.463    575  23  LNC FinLNC
 49  50.128 -66.046    575  12  LNC FinLNC
 50  25.647  24.139    575  18  LNC FinLNC
 51  65.521  42.810    575  24  LNC FinLNC
 52 -60.956 -47.925    575   3  LNC FinLNC
 53  -9.070  42.798    575   3  LNC FinLNC
 54 -19.061   4.480    575  25  LNC FinLNC
 55 -54.071  -3.870    575  22  LNC FinLNC
 56 -25.598 -18.866    575  25  LNC FinLNC
 57 -18.707  84.296    575  15  LNC FinLNC
 58 -70.184 -39.166    575  18  LNC FinLNC
 59 -14.636  -0.446    575   8  LNC FinLNC
 60 -76.465 -38.922    575  20  LNC FinLNC
 61  44.092  12.433    575  18  LNC FinLNC
```

62	-22.772	-50.867	575	5	LNC	FinLNC
63	19.104	44.672	575	25	LNC	FinLNC
64	-44.019	-39.935	575	16	LNC	FinLNC
65	-2.972	-4.059	575	14	LNC	FinLNC
66	-24.261	35.907	575	1	LNC	FinLNC
67	-61.877	-73.901	575	15	LNC	FinLNC
68	-5.640	-26.721	575	22	LNC	FinLNC
69	45.844	-36.835	575	6	LNC	FinLNC
70	-51.746	-79.840	575	24	LNC	FinLNC
71	-33.008	10.992	575	19	LNC	FinLNC
72	-25.653	-0.372	575	12	LNC	FinLNC
73	-21.497	61.487	575	13	LNC	FinLNC
74	2.032	26.727	575	10	LNC	FinLNC
75	-26.758	14.807	575	12	LNC	FinLNC
76	-37.256	-44.861	575	13	LNC	FinLNC
77	-0.861	-16.418	575	25	LNC	FinLNC
78	-51.160	34.967	575	12	LNC	FinLNC
79	-41.266	-76.886	575	2	LNC	FinLNC
80	-30.255	74.701	575	11	LNC	FinLNC
81	-46.368	-1.514	575	10	LNC	FinLNC
82	-71.619	73.077	575	9	LNC	FinLNC
83	-46.606	-39.948	575	21	LNC	FinLNC
84	-30.243	67.102	575	14	LNC	FinLNC
85	-40.997	-43.036	575	15	LNC	FinLNC
86	-74.286	-52.307	575	6	LNC	FinLNC
87	-17.609	-71.851	575	25	LNC	FinLNC
88	2.350	-81.805	575	14	LNC	FinLNC
89	-85.486	-54.059	575	17	LNC	FinLNC
90	23.822	10.571	575	8	LNC	FinLNC
91	-24.530	-31.726	575	4	LNC	FinLNC
92	-42.474	-15.009	575	17	LNC	FinLNC
93	-33.624	50.330	575	23	LNC	FinLNC
94	-81.458	-41.565	575	4	LNC	FinLNC
95	-73.505	-48.279	575	4	LNC	FinLNC
96	-3.052	70.898	575	9	LNC	FinLNC
97	0.946	25.458	575	9	LNC	FinLNC
98	-54.144	13.525	575	19	LNC	FinLNC
99	-30.634	7.526	575	12	LNC	FinLNC
100	-68.335	-34.277	575	25	LNC	FinLNC
101	7.178	57.935	575	8	LNC	FinLNC
102	4.712	48.669	575	13	LNC	FinLNC
103	58.087	26.379	575	3	LNC	FinLNC
104	18.347	34.784	575	6	LNC	FinLNC
105	-44.629	-61.975	575	11	LNC	FinLNC
106	5.231	26.068	575	24	LNC	FinLNC
107	-48.065	-36.755	575	16	LNC	FinLNC
108	-3.589	-27.734	575	21	LNC	FinLNC
109	9.808	36.896	575	25	LNC	FinLNC
110	59.460	1.233	575	10	LNC	FinLNC
111	-47.522	19.873	575	2	LNC	FinLNC
112	-48.431	-50.317	575	13	LNC	FinLNC
113	44.666	20.953	575	14	LNC	FinLNC
114	-19.226	32.770	575	19	LNC	FinLNC
115	-15.662	17.462	575	16	LNC	FinLNC
116	-91.669	77.722	575	23	LNC	FinLNC
117	-30.035	-78.284	575	13	LNC	FinLNC
118	73.663	32.941	575	24	LNC	FinLNC
119	22.400	-60.083	575	17	LNC	FinLNC
120	61.206	-13.501	575	8	LNC	FinLNC
121	22.131	-50.012	575	12	LNC	FinLNC
122	-6.317	56.665	575	6	LNC	FinLNC
123	-20.471	12.885	575	20	LNC	FinLNC
124	-61.084	-44.769	575	20	LNC	FinLNC
125	-9.576	-19.366	575	24	LNC	FinLNC
126	5.627	-88.501	575	10	LNC	FinLNC
127	-2.618	22.131	575	18	LNC	FinLNC
128	40.497	32.343	575	10	LNC	FinLNC
129	-3.296	36.774	575	15	LNC	FinLNC
130	-50.995	19.781	575	12	LNC	FinLNC
131	-70.520	-3.809	575	22	LNC	FinLNC
132	4.205	0.812	575	11	LNC	FinLNC
133	9.955	26.337	575	14	LNC	FinLNC
134	13.538	-21.313	575	9	LNC	FinLNC

135	-21.869	-18.768	575	6	LNC	FinLNC
136	-37.158	-30.768	575	23	LNC	FinLNC
137	-86.975	10.345	575	8	LNC	FinLNC
138	-47.272	-46.326	575	22	LNC	FinLNC
139	4.095	-19.354	575	7	LNC	FinLNC
140	4.205	30.292	575	15	LNC	FinLNC
141	-42.017	53.986	575	22	LNC	FinLNC
142	35.095	-10.895	575	7	LNC	FinLNC
143	61.188	-59.875	575	14	LNC	FinLNC
144	-34.674	-1.337	575	21	LNC	FinLNC

Entrada 6

```
6
216
145  -3.799  44.290  575        10000
146  14.233  21.173  575        10000
147 -23.334 -28.397  575        10000
148  10.065   1.822  575        10000
149 -49.115 -43.549  575        10000
150 -21.054   4.144  575        10000
 19948   7.239   1.599680  21  LNC FinLNC
 19949 -29.785 -11.285        680  13  LNC FinLNC
 19950 -89.050  16.211        680  15  LNC FinLNC
 19951 -46.887  -3.363        680  13  LNC FinLNC
 19952 -14.972  30.621        680  20  LNC FinLNC
 19953 -17.035  49.774        680  10  LNC FinLNC
 19954  31.635  53.619        680  25  LNC FinLNC
 19955  -3.577  13.342        680   7  LNC FinLNC
 19956  33.008  58.960        680  15  LNC FinLNC
 10  19.934  35.883    575  13  LNC FinLNC
 11 -27.771  32.269    575  10  LNC FinLNC
 12  51.154 -16.827    575   4  LNC FinLNC
 13 -19.214  23.749    575  12  LNC FinLNC
 14 -46.643 -52.954    575  21  LNC FinLNC
 15 -10.590 -16.626    575   2  LNC FinLNC
 16  35.150  12.622    575  10  LNC FinLNC
 17 -37.378  12.256    575  20  LNC FinLNC
 18  82.794  25.836    575   9  LNC FinLNC
 19   8.936  -3.723    575  24  LNC FinLNC
 20 -56.659  25.555    575  19  LNC FinLNC
 21 -52.020 -40.137    575  13  LNC FinLNC
 22  39.270  12.787    575  14  LNC FinLNC
 23  19.757  21.228    575  17  LNC FinLNC
 24 -23.071 -12.659    575  25  LNC FinLNC
 25  15.686  52.374    575   1 LNC FinLNC
 26  58.771  -7.056    575   9  LNC FinLNC
 27 -33.087  72.266    575  12  LNC FinLNC
 28   2.551  46.851    575  22  LNC FinLNC
 29 -18.719  73.126    575   2  LNC FinLNC
 30 -67.627 -20.679    575  21  LNC FinLNC
 31 -76.862 -11.365    575   8  LNC FinLNC
 32 -24.194 -38.263    575  19  LNC FinLNC
 33 -20.374 -31.903    575  24  LNC FinLNC
 34   9.290  39.618    575  24  LNC FinLNC
 35  26.044  35.974    575   2  LNC FinLNC
 36 -57.385 -45.264    575  20  LNC FinLNC
 37   1.489 -62.592    575  20  LNC FinLNC
 38  34.937  13.672    575   5  LNC FinLNC
 19924 -70.740  62.244        680   8  LNC FinLNC
 19925  32.538  23.096        680  10  LNC FinLNC
 19926 -51.453 -36.444        680   9  LNC FinLNC
 19927  36.456 -22.638        680   4  LNC FinLNC
 19928 -31.207  43.494        680   6  LNC FinLNC
 19929 -10.388  34.491        680  22  LNC FinLNC
 19930  14.722 -10.834        680  17  LNC FinLNC
 19931  47.095 -21.387        680   9  LNC FinLNC
 39 -22.290  45.648    575  17  LNC FinLNC
 40  -5.920  44.061    575   4  LNC FinLNC
 41 -38.446 -23.541    575  21  LNC FinLNC
 42  28.894  33.936    575   3  LNC FinLNC
 43 -35.388  -5.341    575  16  LNC FinLNC
 44 -19.501  13.519    575  14  LNC FinLNC
 45  -9.216 -30.988    575   6  LNC FinLNC
 46 -22.131 -17.065    575   5  LNC FinLNC
 47 -22.827  36.810    575  12  LNC FinLNC
 48 -61.365 -18.463    575  23  LNC FinLNC
 49  50.128 -66.046    575  12  LNC FinLNC
 50  25.647  24.139    575  18  LNC FinLNC
 1995 -30.194  67.773        680  18  LNC FinLNC
 1996  12.677 -57.471        680   1  LNC FinLNC
 1997 -32.355 -20.966        680  15  LNC FinLNC
```

1998	19.910	48.975	680	23	LNC	FinLNC
1999	13.202	-19.135	680	13	LNC	FinLNC
19910	54.877	-41.168	680	12	LNC	FinLNC
51	65.521	42.810	575	24	LNC	FinLNC
58	-70.184	-39.166	575	18	LNC	FinLNC
59	-14.636	-0.446	575	8	LNC	FinLNC
60	-76.465	-38.922	575	20	LNC	FinLNC
61	44.092	12.433	575	18	LNC	FinLNC
62	-22.772	-50.867	575	5	LNC	FinLNC
63	19.104	44.672	575	25	LNC	FinLNC
64	-44.019	-39.935	575	16	LNC	FinLNC
65	-2.972	-4.059	575	14	LNC	FinLNC
66	-24.261	35.907	575	1	LNC	FinLNC
109	9.808	36.896	575	25	LNC	FinLNC
110	59.460	1.233	575	10	LNC	FinLNC
111	-47.522	19.873	575	2	LNC	FinLNC
112	-48.431	-50.317	575	13	LNC	FinLNC
113	44.666	20.953	575	14	LNC	FinLNC
114	-19.226	32.770	575	19	LNC	FinLNC
115	-15.662	17.462	575	16	LNC	FinLNC
116	-91.669	77.722	575	23	LNC	FinLNC
67	-61.877	-73.901	575	15	LNC	FinLNC
68	-5.640	-26.721	575	22	LNC	FinLNC
69	45.844	-36.835	575	6	LNC	FinLNC
80	-30.255	74.701	575	11	LNC	FinLNC
92	-42.474	-15.009	575	17	LNC	FinLNC
93	-33.624	50.330	575	23	LNC	FinLNC
94	-81.458	-41.565	575	4	LNC	FinLNC
95	-73.505	-48.279	575	4	LNC	FinLNC
96	-3.052	70.898	575	9	LNC	FinLNC
97	0.946	25.458	575	9	LNC	FinLNC
98	-54.144	13.525	575	19	LNC	FinLNC
99	-30.634	7.526	575	12	LNC	FinLNC
100	-68.335	-34.277	575	25	LNC	FinLNC
101	7.178	57.935	575	8	LNC	FinLNC
102	4.712	48.669	575	13	LNC	FinLNC
103	58.087	26.379	575	3	LNC	FinLNC
104	18.347	34.784	575	6	LNC	FinLNC
105	-44.629	-61.975	575	11	LNC	FinLNC
106	5.231	26.068	575	24	LNC	FinLNC
107	-48.065	-36.755	575	16	LNC	FinLNC
108	-3.589	-27.734	575	21	LNC	FinLNC
117	-30.035	-78.284	575	13	LNC	FinLNC
118	73.663	32.941	575	24	LNC	FinLNC
119	22.400	-60.083	575	17	LNC	FinLNC
120	61.206	-13.501	575	8	LNC	FinLNC
121	22.131	-50.012	575	12	LNC	FinLNC
122	-6.317	56.665	575	6	LNC	FinLNC
81	-46.368	-1.514	575	10	LNC	FinLNC
82	-71.619	73.077	575	9	LNC	FinLNC
83	-46.606	-39.948	575	21	LNC	FinLNC
84	-30.243	67.102	575	14	LNC	FinLNC
85	-40.997	-43.036	575	15	LNC	FinLNC
86	-74.286	-52.307	575	6	LNC	FinLNC
87	-17.609	-71.851	575	25	LNC	FinLNC
88	2.350	-81.805	575	14	LNC	FinLNC
89	-85.486	-54.059	575	17	LNC	FinLNC
90	23.822	10.571	575	8	LNC	FinLNC
91	-24.530	-31.726	575	4	LNC	FinLNC
123	-20.471	12.885	575	20	LNC	FinLNC
124	-61.084	-44.769	575	20	LNC	FinLNC
125	-9.576	-19.366	575	24	LNC	FinLNC
126	5.627	-88.501	575	10	LNC	FinLNC
127	-2.618	22.131	575	18	LNC	FinLNC
128	40.497	32.343	575	10	LNC	FinLNC
129	-3.296	36.774	575	15	LNC	FinLNC
130	-50.995	19.781	575	12	LNC	FinLNC
131	-70.520	-3.809	575	22	LNC	FinLNC
132	4.205	0.812	575	11	LNC	FinLNC
133	9.955	26.337	575	14	LNC	FinLNC
134	13.538	-21.313	575	9	LNC	FinLNC
135	-21.869	-18.768	575	6	LNC	FinLNC
136	-37.158	-30.768	575	23	LNC	FinLNC

```
137 -86.975  10.345   575   8  LNC FinLNC
138 -47.272 -46.326   575  22  LNC FinLNC
139   4.095 -19.354   575   7  LNC FinLNC
140   4.205  30.292   575  15  LNC FinLNC
141 -42.017  53.986   575  22  LNC FinLNC
142  35.095 -10.895   575   7  LNC FinLNC
143  61.188 -59.875   575  14  LNC FinLNC
144 -34.674  -1.337   575  21  LNC FinLNC
 1991 -92.700 -59.180         680  20  LNC FinLNC
 1992  71.179  12.543         680   6  LNC FinLNC
 1993  31.537  66.638         680  19  LNC FinLNC
 1994  -4.694  25.537680  10  LNC FinLNC
19911  15.063 -25.171         680   7  LNC FinLNC
19912 -50.598 -16.418         680  25  LNC FinLNC
19913 -29.730  17.078         680   5  LNC FinLNC
19914  17.542   1.575         680   1  LNC FinLNC
19915  11.127  77.216         680  25  LNC FinLNC
19916  33.752  71.259         680  10  LNC FinLNC
19917 -56.012 -10.394         680   2  LNC FinLNC
19918  57.874 -16.290         680   7  LNC FinLNC
19919  10.718 -18.787         680  11  LNC FinLNC
19920  53.088 -18.750         680   4  LNC FinLNC
52 -60.956 -47.925   575   3  LNC FinLNC
53  -9.070  42.798   575   3  LNC FinLNC
54 -19.061   4.480   575  25  LNC FinLNC
55 -54.071  -3.870   575  22  LNC FinLNC
56 -25.598 -18.866   575  25  LNC FinLNC
57 -18.707  84.296   575  15  LNC FinLNC
19921   1.569   7.532680   6  LNC FinLNC
19922  31.531  48.944         680  15  LNC FinLNC
19923 -66.833 -37.854         680   7  LNC FinLNC
19932  43.781  34.766         680  25  LNC FinLNC
19933  53.546 -67.487         680  10  LNC FinLNC
19934  26.801  46.515         680  18  LNC FinLNC
19935  63.385  11.981         680  21  LNC FinLNC
19936  47.192  -5.475         680  10  LNC FinLNC
19937 -16.315 -11.267         680  23  LNC FinLNC
19938  78.900  17.651         680  23  LNC FinLNC
19939  79.822  22.272         680  11  LNC FinLNC
19940  12.878  16.919         680   1  LNC FinLNC
70 -51.746 -79.840   575  24  LNC FinLNC
71 -33.008  10.992   575  19  LNC FinLNC
72 -25.653  -0.372   575  12  LNC FinLNC
73 -21.497  61.487   575  13  LNC FinLNC
74   2.032  26.727   575  10  LNC FinLNC
75 -26.758  14.807   575  12  LNC FinLNC
76 -37.256 -44.861   575  13  LNC FinLNC
77  -0.861 -16.418   575  25  LNC FinLNC
78 -51.160  34.967   575  12  LNC FinLNC
79 -41.266 -76.886   575   2  LNC FinLNC
19941 -67.981  -3.754         680  23  LNC FinLNC
19942   9.198 -18.597         680  16  LNC FinLNC
19943 -35.950 -19.141         680  10  LNC FinLNC
19944  28.766  45.276         680  12  LNC FinLNC
19945  11.469  68.231         680  12  LNC FinLNC
19946 -22.760  45.496         680   3  LNC FinLNC
19947 -65.674 -23.120         680  22  LNC FinLNC
19957 -92.950  63.263         680   2  LNC FinLNC
19958  -9.137 -22.931         680  23  LNC FinLNC
19959 -39.960   6.195         680   5  LNC FinLNC
19960  28.430 -19.214         680  25  LNC FinLNC
 1 -40.289 -42.303    575  20  LNC FinLNC
 2 -64.709 -17.389    575  10  LNC FinLNC
 3   5.060 -14.349    575   8  LNC FinLNC
 4  72.095  20.233    575   6  LNC FinLNC
 5   2.594 -15.002    575   1  LNC FinLNC
 6 -24.176 -72.894    575  19  LNC FinLNC
 7 -13.190  66.498    575  13  LNC FinLNC
 8  33.191  13.690    575  24  LNC FinLNC
 9  66.827 -30.554    575  15  LNC FinLNC
19961 -28.540  -3.485         680   9  LNC FinLNC
19962  31.415  36.859         680   2  LNC FinLNC
19963 -49.426  60.602         680  24  LNC FinLNC
```

19964	-72.827	-27.765	680	13	LNC	FinLNC
19965	60.083	-45.905	680	20	LNC	FinLNC
19966	10.870	-3.900	680	13	LNC	FinLNC
19967	25.122	7.672	680	15	LNC	FinLNC
19968	-46.997	-17.474	680	4	LNC	FinLNC
19969	16.058	33.020	680	20	LNC	FinLNC
19970	25.409	-11.700	680	8	LNC	FinLNC
19971	68.323	-5.145	680	19	LNC	FinLNC
19972	-13.104	62.158	680	20	LNC	FinLNC

Anexo E. Manual de Usuario

Versión 1.00
Febrero 2001

E.1 Recorrida por el programa:

Se ejecuta el programa con un doble click sobre el acceso, el .exe e inmediatamente, aparecerá la ventana de la aplicación:

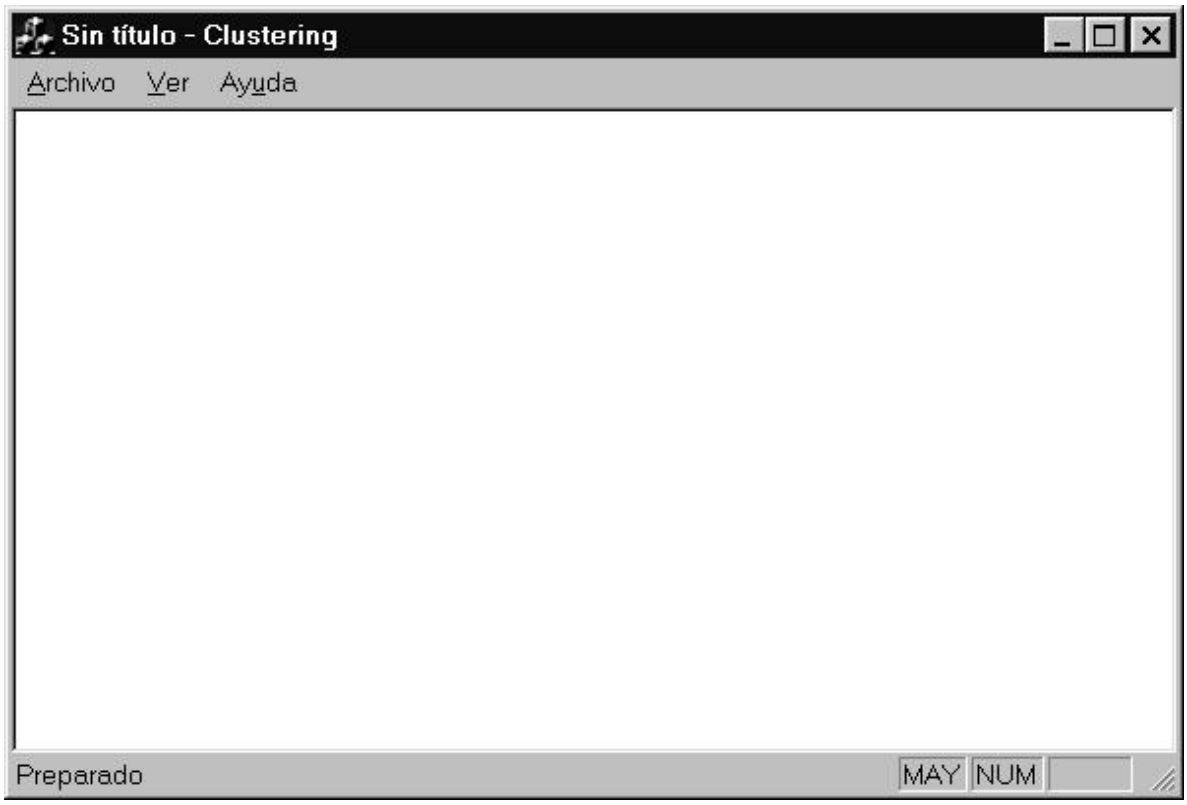


Fig. 1

Esta pantalla es la de inicio del programa. Para comenzar a trabajar se clickea sobre la misma y entonces aparece el siguiente cuadro de dialogo:

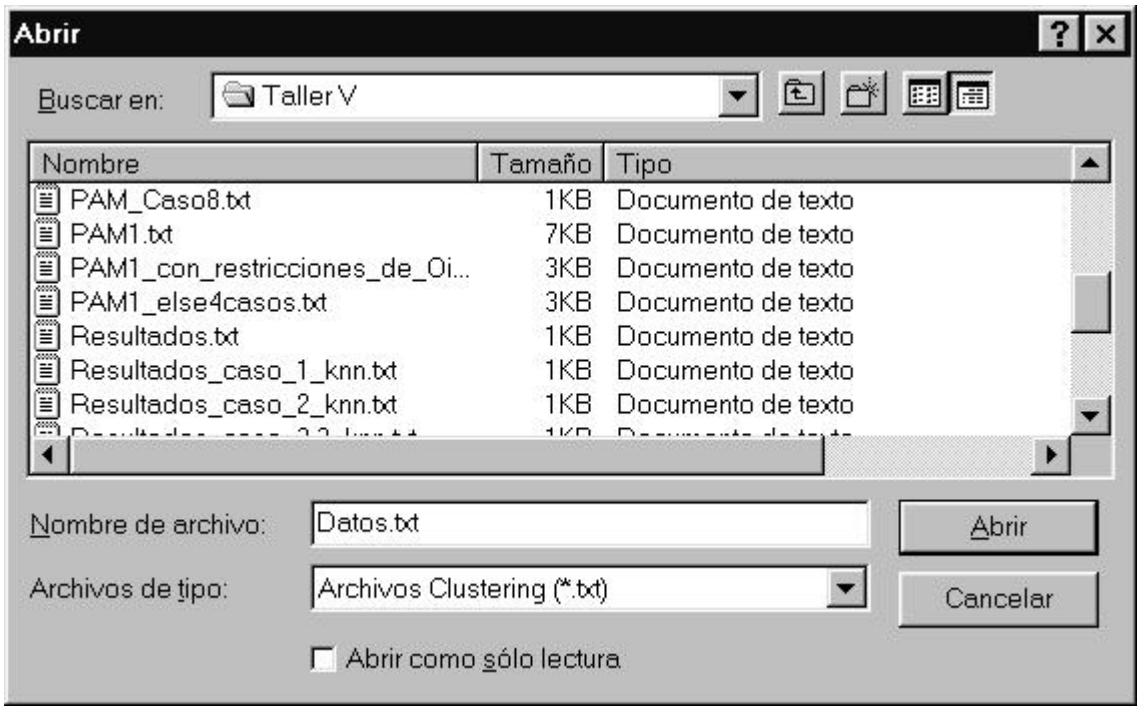


Fig 2.

que tiene como finalidad permitir la elección del Archivo de Entrada (conteniendo los datos a ser procesados en el formato adecuado)

Se selecciona tal archivo (mediante el mouse o desplazándose con las flechas del teclado) y luego se dibujan sobre la pantalla de la Fig. 1 dibujados los triángulos de colores que representan a los Depósitos y los cuadrados negros que representan a los Clientes.(Fig. 3)

Importante: Si en este momento (elección del archivo de datos) se oprime la tecla Cancelar simplemente se sale del programa.

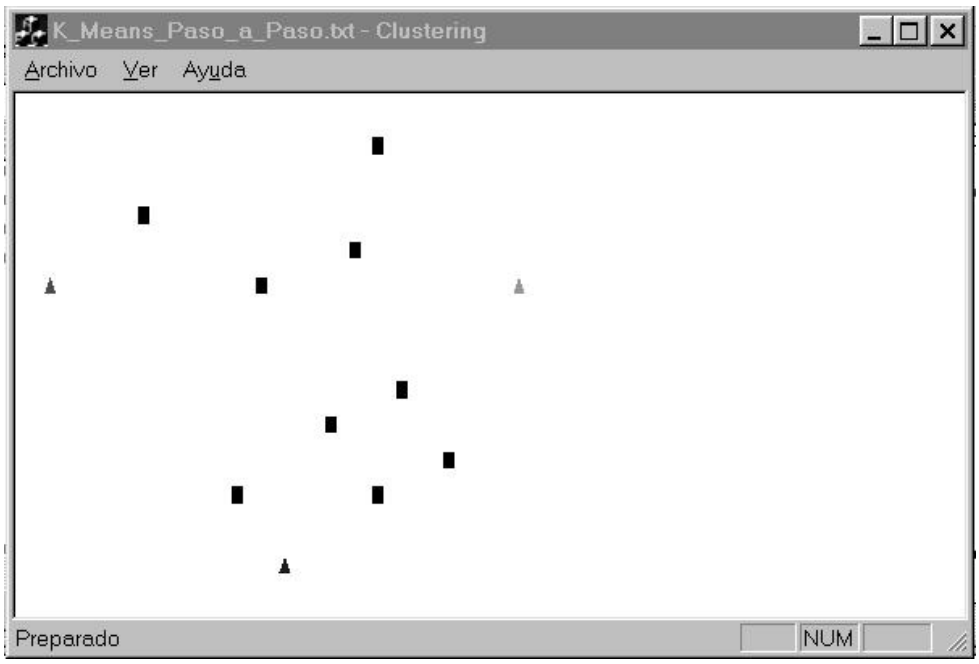


Fig. 3

Con esto queda finalizada la distribución inicial de Depósitos y Clientes.

A continuación el siguiente cuadro de dialogo

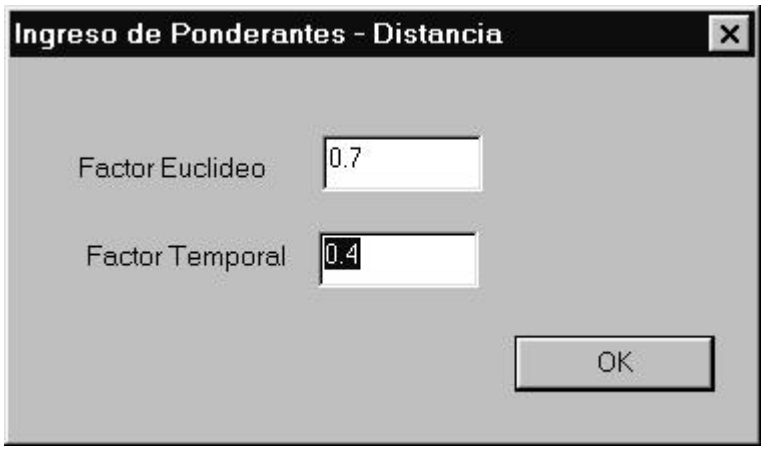


Fig. 4

permite ingresan los Ponderantes a ser usados por la Función Distancia. Si dichos ponderantes no presentan valores en el rango 0 a 1 el programa desplegara el siguiente mensaje de error:

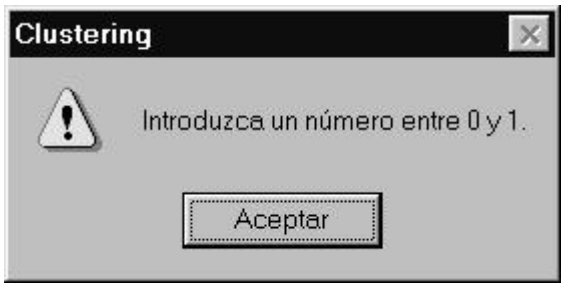


Fig. 5

Y reiterará la solicitud hasta que los Ponderantes entren dentro del rango correcto.

Luego de ingresar los ponderantes, aparece el siguiente cuadro de dialogo que permite seleccionar el algoritmo a utilizar:

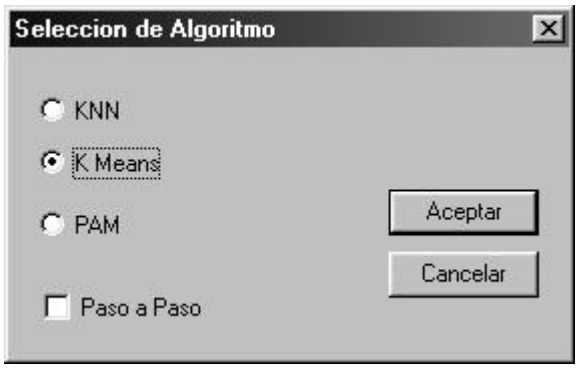


Fig. 6

Uno de los algoritmos aparece seleccionado por defecto (y será el que se ejecutara en caso de elegir el botón Cancelar), si se selecciona la opción “Paso a Paso” lo que se obtendrá será la ejecución del algoritmo mostrando en cada momento las asignaciones que realiza permitiendo al usuario no solo contar con los resultados finales, sino que también podrá realizar un seguimiento del algoritmo para observar su evolución

Importante: SI al momento de la elección del algoritmo se oprime el botón Cancelar, entonces se ejecuta el algoritmo que estaba seleccionado por defecto.

Durante la ejecución del modo"Paso a Paso" el programa solicitara del usuario el OK para continuar. Eso se hace mediante los diálogos que se muestran a continuación. Cuando el usuario desee puede pasar a la siguiente pantalla eligiendo el botón Aceptar o presionando la tecla Enter.

Fig. 7

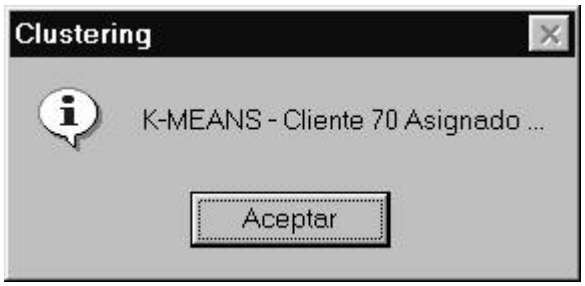


Fig. 8

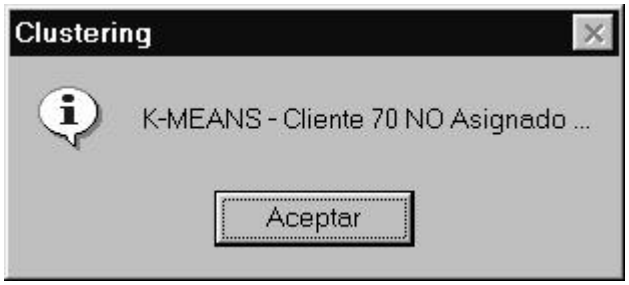


Fig. 9

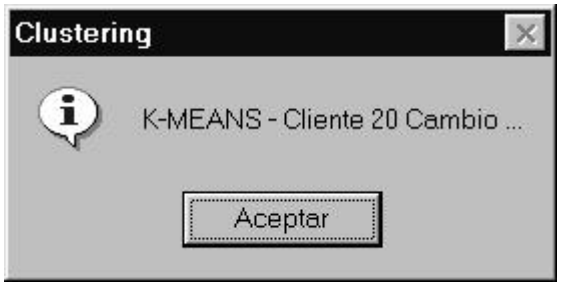


Fig. 10

Fig. 11

Se podrá de esta manera, apreciar los cambios de color de los Clientes al ir ser asignados a los distintos Depósitos.

Recordar: Gráficamente un Cluster se identifica por los Clientes que están del mismo color que su Deposito, y los Clientes NO Asignados quedan en negro.

Si al elegir el algoritmo, se opta por KNN, entonces aparecerá la siguiente ventana donde se elige el valor de *k* a ser utilizado por el algoritmo.

Recordar: el VALOR MAXIMO DE k es el NUMERO DE DEPOSITOS.

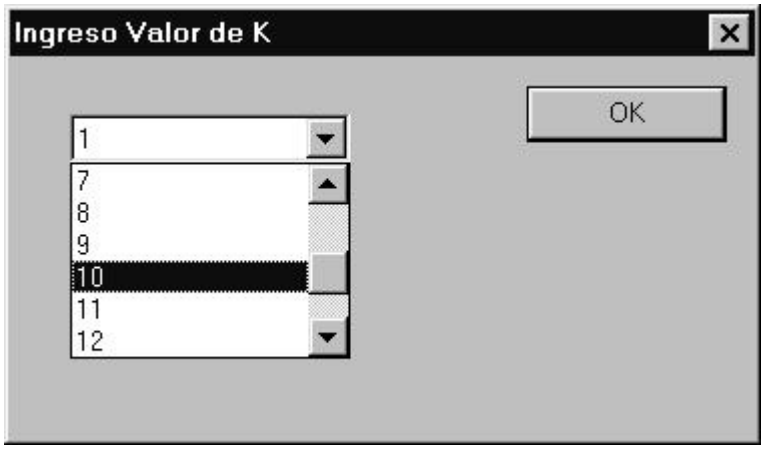


Fig. 12

Al finalizar la ejecución del algoritmo, aparece por encima de la ventana de ejecución (donde ya están pintados los Clientes de los colores que corresponden de acuerdo a las asignaciones de los algoritmos), el siguiente cuadro de dialogo:

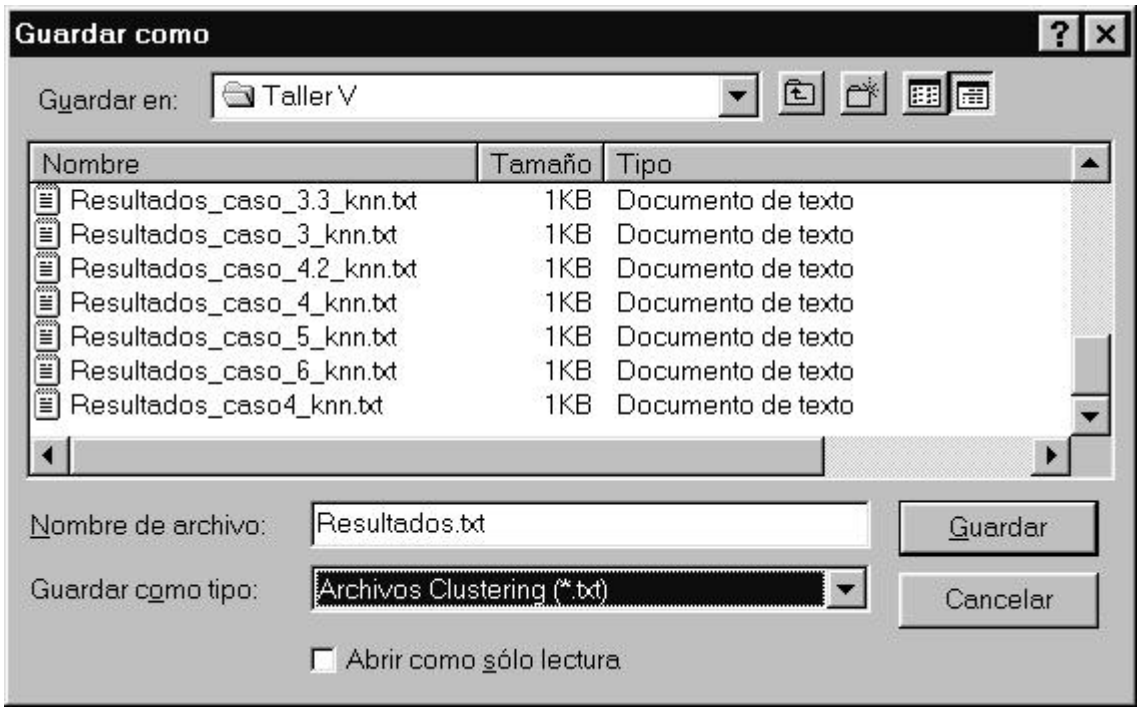


Fig. 13

Este cuadro permite guardar en un archivo los resultados obtenidos por el algoritmo ejecutado. No hay restricciones respecto al nombre y/o la Carpeta donde se almacenará dicha información.

Como sucede en las aplicaciones Windows, si el archivo destino no existe, el programa presentara el siguiente mensaje de advertencia:

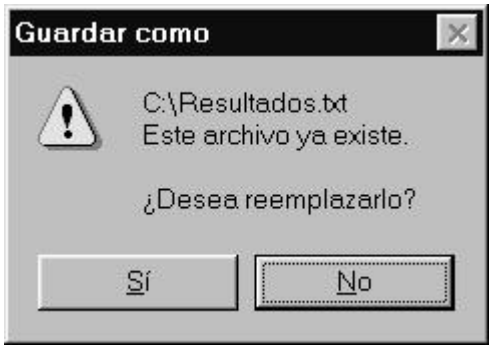


Fig. 14

De esta forma, se evita la sobre escritura no deseada.

Importante: Si se oprime el botón Cancelar al elegir el archivo de Resultados (Salida) entonces los resultados se almacenan por defecto en:

C:_RESULTADOS_DDMMYYYY_HHMISS.TXT

Es entonces, (después de guardar los resultados) que se puede observar gráficamente la distribución de los Clientes en los distintos Depósitos conformando los Clusters.

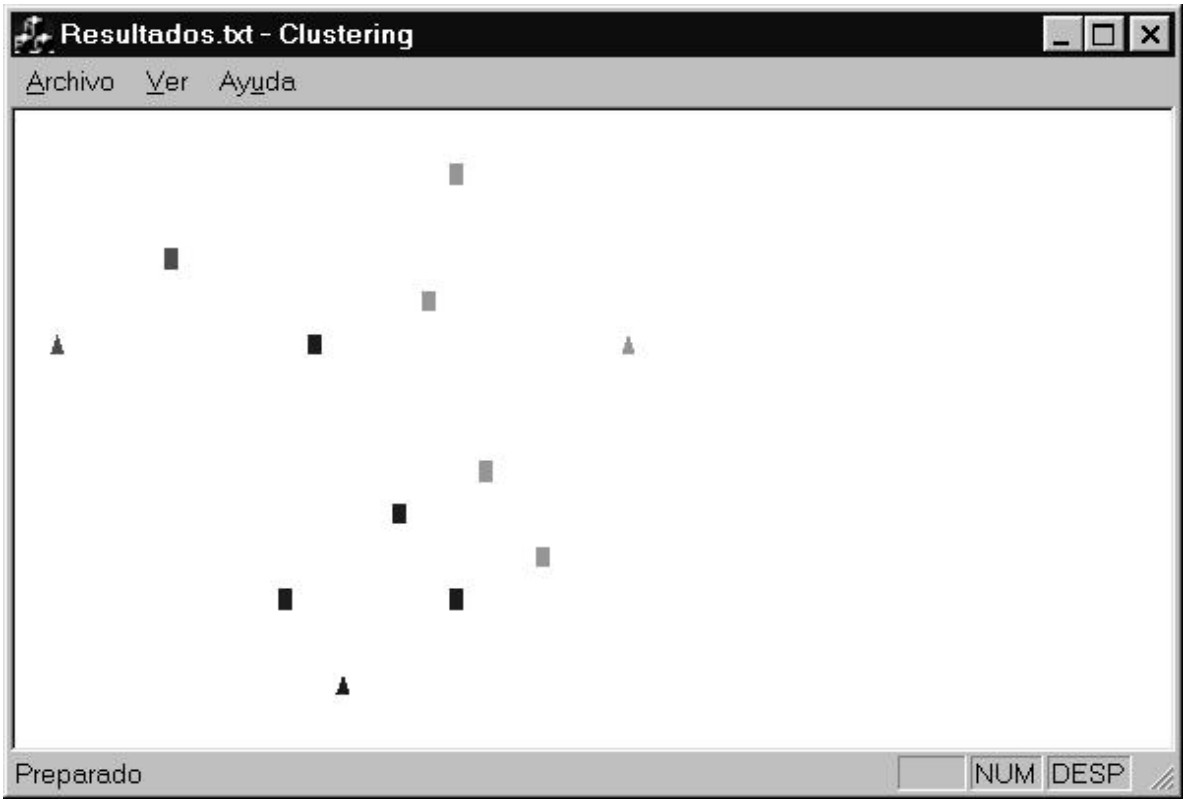


Fig. 15

Finalmente, para volver a ejecutar el programa simplemente se clickea sobre la pantalla resultado y vuelven a ejecutarse los pasos arriba mencionados. Por el contrario, si se desea salir del programa bastará con ir al Menú Archivo-> Salir

E.2 Formato de Archivos

Los formatos de los archivos de Entrada (Datos) y Salida (Resultados) son los siguientes:

Formato de Archivos de Datos:

```
#Depósitos
#Clientes
IDep x y t d
...
IDep x y t d
ICli x y t p "LNC" IDep1 IDep2 ... IDepN "FinLNC"
...
ICli x y t p "LNC" IDep1 IDep2 ... IDepM "FinLNC"
```

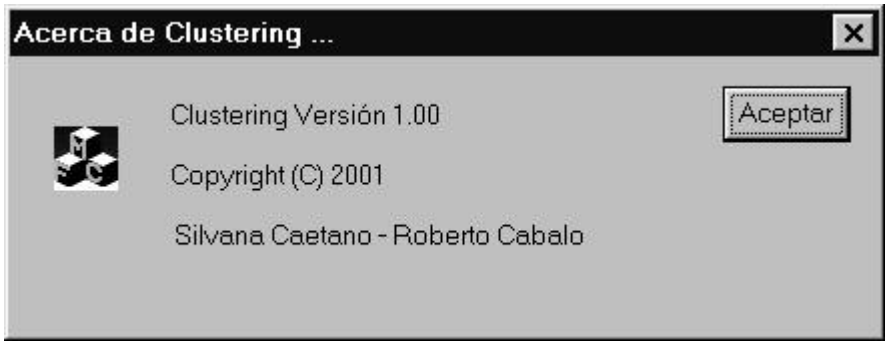
Formato de Archivos de Resultados:

```
Coeficiente XY
Coeficiente TW
#Depósitos
#Clientes
IDep 1 ClientesICli11 ... ICli1N
- - - - - - - - - - - - - - - - - - - - - - - - -
IDep N Clientes ICliN1 ... ICliNN
NO ASIGNADOS Clientes ICli1 . . .ICliM
Tiempo Ejecución
```

donde:

IDep	es el identificador de deposito
ICli	es el identificador de cliente
x,y	coordenadas cartesianas
t	centro de ventana de tiempo
d	demanda
p	producción
LNC	bandera de comienzo de depósitos no compatibles
FinLNC	bandera de fin de depósitos no compatibles
Coeficiente XY	coef. de ponderación de la distancia Euclídea
Coeficiente TW	coef. de ponderación de la distancia temporal
#Depósitos	cantidad de depósitos
#Clientes	cantidad de clientes
NºClientes	cantidad de clientes asignados a un cluster

E.3 Acerca de . . .


Acerca de Clustering ...
Clustering Versión 1.00
Copyright (C) 2001
Silvana Caetano - Roberto Cabalo
Aceptar