Práctica 2

Metaheurísticas

Algoritmos Multiarranque para el problema del Clustering

Grado en Ingeniería Informática

Alumno: José Arcos Aneas
DNI: 74740565-H
Grupo: *Lunes 16h-18h*

Índice

Introducción

Para la realización de esta práctica se realizan tres algoritmos que son búsqueda local multiarranque básica (BMB), búsqueda local iterativa (ILC), GRASP y a demás una variación del algoritmo k-medias en versión multiarranque.
En esta introducción intento explicar como están estructurados los apartados de esta entrega.
En el punto uno se presenta una descripción del problema que abordamos con la aplicación de los algoritmos desarrollados.
El siguiente apartado pretende ser una breve explicación de los diversos algoritmos que vamos utilizar para resolver el problema que se nos presenta.
Los puntos 3,4,5 son una descripción mas profunda de los algoritmos antes mencionados.
El ultimo punto presenta una comparación entre los algoritmos BMB, ILS, GRASP con el algoritmo K-MM.
Como en los algoritmo BMB e ILS no se utiliza factorización los resultados son notablemente peores que los que podemos observar en GRASP y K-MM por lo que las comparaciones entre estos dos algoritmos se realizan entre ellos.

1. Descripción del problema.

El problema que abordamos es el del clustering. El clustering puede definirse como el proceso de agrupar un conjunto de objetos físicos o abstractos dentro de clases con objetos similares se denomina clustering. El clustering consiste en agrupar una colección dada de datos no etiquetados en un conjunto de grupos de tal manera que los objetos que pertenecen a un grupo sean homogéneos entre sí y buscando además, que la heterogeneidad entre los distintos grupos la sea lo más elevada posible . Podríamos ver un problema de clustering como un problema de optimizaron que localiza los centroides óptimos de los clusters en lugar de encontrar la partición del espacio, cayendo en óptimos locales.

En la configuración de los clúster cada objeto permanece a un único clúster, y su distancia al centroide de éste es siempre menor que a los restantes. Así, suponiendo un conjunto M formado por m objetos y configuración de k clústeres C_i, debe cumplirse:

a) $C_i \neq \varnothing$, es decir, $|C_i| > 0$.
b) $C_i \cap C_j = \varnothing$, para todo $i \neq j$.
c) $\sum_{i=1}^{k} C_i = M$, donde:
   - M = {X^1, …, X^m} es el conjunto de patrones
   - X^i = $(x_1^i, \ldots, x_n^i)$, con $x_j^i \in R$.

En el algoritmo que implementaré intentaré deducir J, según la siguiente formula:

$$Minimizar\ J = f(W, Z; X) = \sum_{i=1}^{m} \sum_{j=1}^{k} w_{ij} \| X^i - Z^j \|^2$$

Sujeto a que:

- $\sum_{j=1}^{k} w_{ij} = 1, 1 \leq i \leq m$
- $w_{ij} \in \{0,1\}, 1 \leq i \leq m, 1 \leq j \leq k$

Donde:

- X^i = $(x_1^i, \ldots, x_n^i)$, con $x_l^i \in R$ es el i-ésimo patrón de M.
- Z^j = $(z_1^j, \ldots, z_n^j)$, con $z_l^j \in R$ es el centroide del clúster C_j.

- W es una matriz de pertenencia con dimensión mxk en la que, un dato pertenece a un solo centroide, representando este pertenece con un 1, y un 0 en el caso contrario.
- K es el número de clústeres, m es el número de patrones en M, n es la dimensión del espacio y ||.|| es la distancia Euclídea al cuadrado.

2. Descripción de la aplicación de los algoritmos empleados.

La diferencia entre ellos suele ser la forma en que se generan nuevas soluciones iniciales a las que se le aplicar´ la técnica de BL. Estos métodos se consideran MHs basadas en trayectorias porque siempre realizan un paso desde una solución actual a otra, aunque ésta no tenga que pertenecer al vecindario de la solución actual utilizado por el método de BL. Algunos de los ejemplos más importantes son:

BL con Multiarranque Aleatorio (Boender y otros, 1982; Rinnoy Kan y Timmer, 1989) genera soluciones de forma totalmente aleatoria, del espacio de búsqueda, y les aplica un método de BL. Este método realiza un muestreo aleatorio de los óptimos del problema. Para solucionar este algoritmo se utilizará parte del código de la práctica uno, en donde está implementada la búsqueda local. La representación que se utiliza en la búsqueda local es un vector solución, donde se encuentra el resultado de la búsqueda, también tiene un vector de centroides que es el valor medio de todos los valores que pertenecen a un centroide determinado, y una función J que dará el resultado de la suma de todas las distancias entre el centroide que pertenece un dato, respecto al centroide.
Utiliza el código ya resuelto de la practica uno, con las mismas funciones, a las que tuve que incorporarle una función para cargar un vector solución (void setVectorSolucion (vector <int> vs)) y otra función para recuperar el vector solución (vector <int> getVectorSolucion ()). En el algoritmo ILS, existe una función mutar, que dado un vector, cambia con una probabilidad del 60% el valor de la pertenencia de ese dato a un clúster aleatoriamente.

BL Iterativa (Loureny otros, 2003) aplica un método de BL a una solución inicial hasta encontrar un óptimo local, después, perturba la solución obtenida y aplica el método de BL a la nueva solución, y vuelve a repetir este proceso. Su idea es la de realizar un recorrido por el espacio de óptimos del problema. La importancia de la perturbación es obvia: perturbaciones demasiado leves pueden ser incapaces de escapar de la base de atracción del óptimo que se acaba de encontrar. Por otro lado, perturbaciones demasiado intensas pueden hacer que BL Iterativa se comporte de forma similar a BL con Multiarranque Aleatorio. hacemos una búsqueda más extensa, ya que buscamos a partir de una solución otras que podrían ser mejores haciendo un salto en el espacio de búsqueda. El algoritmo repite este procedimiento descrito n-veces, quedándose con el resultado que minimice J.

Como el algoritmo BMB, utiliza el código ya resuelto de la practica uno, con las mismas funciones, a las que hubo que incorporarle una función para cargar un vector solución (void setVectorSolucion (vector <int> vs)) y otra función para recuperar el vector solución (vector <int> getVectorSolucion ()). En el algoritmo ILS, existe una función mutar, que dado un vector, cambia con una probabilidad del 60% el valor de la pertenencia de ese dato a un clúster aleatoriamente.

El algoritmo GRASP consta de dos partes: una construcción de soluciones greedy probabilístico y una optimización mediante el algoritmo de búsqueda local. Donde tendremos una tolerancia de umbral α igual a 0.3.
Con el algoritmo greedy tendremos una solución óptima local. Con la búsqueda local podremos buscar más soluciones en nuestro espacio, a partir de la solución dada por el algoritmo greedy. Este proceso se repetirá n veces, y se guardará la solución que minimice J.

El algoritmo K-MM es una variación de k-medias, en donde se repite la búsqueda de esta forma n veces y devuelve el mejor resultado de J, de todas las ejecuciones hechas. Para tener este resultado es necesario que el vector solución inicial en cada iteración de ejecución del algoritmo KM sea aleatorio.

3. Descripción Algoritmo Búsqueda Local Multiarranque Básica.

Para hacer este algoritmo hemos utilizado la búsqueda local de la primera práctica, en la que tenemos las siguientes funciones implementadas:
Constructor: Le pasamos como argumentos la cantidad de cluster, y una referencia a un vector de vectores de float, que serán los datos en los que queremos hacer la búsqueda.

Ejecutar: Ejecutará la búsqueda local, de tal forma que terminará cuando no haya cambios o cuando se haya evaluado tantas veces como el parámetro que se le pasa en esta función. getJ: Devuelve el valor de J.

Las funciones propias de este algoritmo son:

Constructor: Como parámetros se le pasa el número de clústeres, un puntero a un vector de vectores del tipo float (los datos), el número de iteraciones para la búsqueda local y el número de veces que se quiere ejecutar la búsqueda local.

Ejecutar: Ejecuta el programa y devuelve el valor menor de J encontrado.

El pseudocódigo utilizado en ejecutar es:

```
Ejecutar (){
        realizar n búsquedas locales
                guardar mejor resultado
        conservar mejor solución.
Devolver mejor
}
```

Cuando se ejecuta el algoritmo de búsqueda BMB, inicia una variable Jminima igual a infinito, y posteriormente se ejecuta un bucle for, que ejecuta la búsqueda local tantas veces como el parámetro numrepeticiones indica. En cada ciclo del bucle se genera una búsqueda local, y se obtiene el J de esa búsqueda, si resulta que es menor que la Jminima que hemos obtenido antes, se sustituye. Cuando termina el bucle este retorna el resultado de la Jminima encontrada.

5. Descripción GRASP

Como en los dos algoritmos anteriores se utiliza la BL, con las mismas funciones mencionadas en el punto 2. El psecódigo que sigue GRASP es la unión de una función solGreedy y la BL como antes se ha comentado:

Pseudocódigo SolGreddy

```
SolGreddy(){
        calculamos medoides.
        Buscamos los elementos mas cercanos
        Comprobamos la ganancia
        calculamos el umbral.
        Buscamos el cluster mas cercanos para cada patron Greddy.
}
```

Los algoritmos GRASP son algoritmos de tipo iterativo en los que cada iteración incluye una fase de construcción de una solución y otra de postprocesamiento en la cual se optimiza la solución generada en la primera fase.

La estructura de un algoritmo GRASP es la siguiente:

Mientras no se satisfaga el criterio de parada
1.Construir una solución greedy aleatoria
2.Aplicar una técnica de búsqueda local a la solución greedy aleatoria obtenida en el paso anterior (para mejorarla)
4.Actualizar la mejor solución encontrada

El procedimiento GRASP básico:

```
Mientras no se satisfaga el criterio de parada
 Solución = Solución greedy aleatoria
 Repetir L veces
        Solución = Búsqueda local (Solución)
        Actualizar la mejor solución (si corresponde)
        Solución = Mutación(Solución)
```

Como centros de los agrupamiento cogeremos los patrones del conjunto de entrenamiento de forma que el patrón escogido en cada momento sea el mas alejado a los centroide ya fijados, siendo la distancia a un conjunto de centroides el mínimo de las distancias a cada centroide). Obviamente, el primer centroide ha sido escogido de forma aleatoria.

El algoritmo greedy aleatorio es análogo al algoritmo greedy clásico, teniendo en cuenta que, en cada momento, se escoge aleatoriamente uno de los patrones más alejados al conjunto de centroides ya establecido.

Este algoritmo presenta mejoras frente a la BMB y a la búsqueda ILS en la mayoría de ejecuciones, esto es debido a que la solución greddy interfiere en la solución final al

generar una solución inicial que es voraz.

Por lo general los resultados de este algoritmo son tanto en tiempo como en resultados peores a los de K-MM y mejores aunque solo en tiempo a ILS y BMB.
Algo raro es que para el fichero de datos wdbc.txt el tiempo de ejecución es de escasos minutos, en comparación al archivo de datos Aggregation.txt o Yeast.txt que pueden llegar a tardar hasta varias horas para realizar todas las ejecuciones.

6- Descripción Algoritmo Búsqueda Local Iterativa

Como en BMB, en este algoritmo también hemos utilizado la BL antes comentada. Las funciones propias de ILS son:
Constructor: Se le pasa como argumento el número de veces que se ejecutara la búsqueda local, el número de clústeres que tendrá nuestra búsqueda y el puntero a los datos.
Mutar: Es una función que muta un vector solución, en otro con una probabilidad de 0.60. El parámetro de entrada será un vector solución de enteros, esta será modificada y devuelta por la misma función.
Ejecutar: Se ejecuta el algoritmo ILS según la configuración del constructor.
Tendremos una estructura auxiliar que guardará el vector solución.
El pseudocódigo de ejecutar es:

```
Ejecutar(){
        realizar búsqueda local
        ejecutar un numero_evaluaciones de BL
                guarda el mínimo J
                solucion1=guardar Vector Solución
        Repetir este proceso n veces
                solución2=búsqueda local
                mutar solución1 con solución2
                ejecutar búsqueda local
        Devolver mejor resultado.
```

El algoritmo inicia con un resultado de una búsqueda local e inicializa el Jminimo al valor de la ejecución del BL y el vectorSolucion a la solución de la BL, con un vector solución aleatorio. Luego se hace un bucle desde uno hasta el número de repeticiones, se hará una búsqueda local sobre un vector solución mutado de la variable vectorSolucion. Si el J conseguido en este es mejor que el Jminima entonces el valor de Jminima tiene el valor de J de esta última búsqueda local y el vectorSolución se actualiza. La ejecución retorna el valor de Jminima.

7. Comparación con algoritmo K- MM.

Breve descripción del algoritmo K-MM

La diferencia entre el k-m y el k-mm es que el último se repite n veces, con un valor aleatorio.

Pseudocódigo:

```
KMM(){
        solucionIni= infinito
        Repetir n veces
            sol=calcular k-M
            guarda solución si sol menor que solucionIni
        devolver menor solución.
}
```

Ejecuta n veces el algoritmo KM, y devuelve el mejor de los resultados obtenidos en todas las ejecuciones.

Comparación Grasp Vs K-mm

Los resultados para el algoritmo Grasp son peores tanto en tiempo como en el objetivo que es minimizar la J.

Aun así los valores de las ejecuciones del algoritmo Grasp dan mejores resultados en que las de BMB e ILS, aunque el tiempo de ejecución de estos algoritmos son algo menores.

Comparación BMB VS k-MM

Teniendo en cuenta las búsqueda local implementada para la ejecución de este algoritmo. Los resultados de diferentes ejecuciones de BMB dan una media y una máxima bastante superiores al ILS.

Esto es debido a que la perturbación no es demasiado débil como para escapar de la base de atracción del óptimo que se acaba de encontrar, y tampoco es lo suficientemente intenta como para dar resultados equivalentes a los de la BMB, aun así los resultados son bastante parecidos a los de BMB. La diferencia en los resultados es que ILS hace una búsqueda mas extensa, ya que a partir de una solución busca otras que podrían ser mejores haciendo un salto en el espacio de búsqueda. El algoritmo se repite un numero determinado de veces almacenando la mejor J, en nuestro caso la que mas minimice la J.