UNIVERSIDAD DE GRANADA

E.T.S.I. INFORMÁTICA Y TELECOMUNICACIÓN

Departamento de Ciencias de la Computación e Inteligencia Artificial
Metaheurísticas

Práctica 5.b: Búsquedas Híbridas para el problema del Clustering

Curso 2012-2013

Tercer Curso del Grado en Ingeniería Informática

Alumno: José Arcos Aneas
D.N.I: 74740565-H
Correo electrónico: joseaa@correo.ugr.es

ÍNDICE

0. Introducción.

En la introducción describo como se organiza esta practica.

El apartado 1 contiene la descripción del problema del clustering. Es una descripción general.
El siguiente apartado describe de forma general los algoritmos meméticos y con un poco de mayor profundidad los algoritmos en los que se centra la práctica.
El apartado 3 informa sobre los métodos mas importantes y la estructuración de los componente necesarios para aplicar los algoritmos meméticos al problema del clustering. Este apartado se describen los pseudocódigos de los métodos empleados para realizar la práctica.
El ultimo apartado es una breve comparación entre los métodos meméticos y el k-medias de la practica P1, además de una breve conclusión sobre los algoritmos.

1. Descripción del Problema Del Clustering.

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.

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) $C_i \neq \emptyset$, es decir, $|C_i| > 0$.
b) $C_i \cap C_j = \emptyset$, para todo $i \neq j$.
c) $\sum_{i=1}^{k} C_i = M$, donde:
   - M = $\{X^1, \ldots, X^m\}$ 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.

MÉTODO LOCAL SEARCH

Descripción en pseudocódigo de la búsqueda local empleada, búsqueda local al primer mejor.

En las búsquedas locales se evalúa la solución.

```
busquedaLocal(numero iteraciones, numero de evaluaciones){

        Solucion Inicial

        Aumentamos en 1 eval. ----para contabilisar bien el numero de eval que llgamos
        for interacciones < numero_iteraciones
                limpiamos el vector de usados
                for i=0 i < numero_clusters
                        inicializamos un iterador para cada clusters con el primer elemento de cada clusters.
                        recorremos con el iterador los cluster
                                Buscamos la distancia de cada patrón al cendroide
                                        si no a sido usado buscamos en otro cluster distinto para ver si alguno
mejora
                                                si la mejora
                                                recalculamos centroides
                                                movemos patrón al nuevo centroide
                                                aplicamos el operador de vecino

                                                        recalcularCentroides(i,min,it);
                                                        moverPatron(i,min,it);
                                                        calculamos nueva solucion
                                                        si solucionInicial>nueva solucion
                                                        incluirUsados(usados,it);
                                                        mejorSolucion=solVecino;

Devolver mejor solución
```

Factorización: Recalcula los centroides eliminando la contribución del elemento cambiado y añadiendo esa contribución al nuevo cluster.

```
RecalcularCentroides(clusterOrigen, clusterDestino,  posPatron){

        calcular coste de ese patrón en el cluster Origen y restarlo
        calcular coste del patrón en el cluster destino y sumarlo.
FinRecalcularCentroides
```

2. Información Algoritmos Meméticos.

El AM consistirá en hibridar el algoritmo genético generacional (AGG) de la Práctica 3.b con la BL de la Práctica 1.a. Se estudiarán las dos posibilidades de hibridación siguientes:

1. AM-(10,1.0): Cada 10 generaciones, se aplica la BL sobre todos los cromosomas de la población.

AM(10,1.0)
inicializar población
mientras no se supere el numero de evaluaciones
SeleccionAGG
OX
mutacionAG
Si el numero de generaciones%10 y numero de generaciones mayor que 0
Para toda la población realizamos BL
else
evaluamos toda la población
reemplazarAGG
Fin AM(10,0.1)

2. AM-(10,0.1): Cada 10 generaciones, se aplica la BL sobre un subconjunto de cromosomas de la población seleccionado aleatoriamente con probabilidad pLS igual a 0.1 para cada cromosoma.

AM(10,1.0)
inicializar población
mientras no se supere el numero de evaluaciones
SeleccionAGG
OX
mutacionAG
Si el numero de generaciones%10 y numero de generaciones mayor que 0
calcular a que cromosoma aplicar BL según pLS que era 0.1.
para ese cromosoma hacemos BL.
else
evaluamos toda la población
reemplazarAGG
Fin AM(10,0.1)

Este problema será representado por un conjunto de parámetros (genes), éstos pueden ser unidos para formar una cadena de valores (cromosoma), a este proceso se le llama codificación.

Algoritmos genéticos generacionales con búsqueda local.

En tal modelo se realizan cruces en una población de individuos, los descendientes son colocados en otra. Al final de la fase reproductiva se elimina la generación anterior y se utiliza la nueva, ahora si el numero de generaciones de la población es múltiplo de 10 y mayor que 0 se procede a realizar una serie de búsquedas locales a los individuos que forman nuestra población con el objetivo de intensificar la explotación del entorno de soluciones.

Cruce
El cruce se lleva a cabo sobre el conjunto intermedio generado por la reproducción. Primero se selecciona aleatoriamente una pareja de individuos para ser cruzados. Después, con el uso de la teoría de las probabilidades se determina si habrá cruce entre los dos individuos seleccionados o no. No todas las parejas de cromosomas se cruzarán, sino que habrá algunas que pasarán intactas a la siguiente generación. Usaremos una técnica llamada elitismo, en la que el individuo más apto a lo largo de las distintas generaciones no se cruza con nadie y se mantiene intacto hasta que surge otro mejor que lo desplazará.

Mutación
La mutación es aplicada a cada descendiente individualmente después de cada cruce. Esta altera cada uno de los genes del cromosoma al azar, con una probabilidad pequeña. . Este operador permite la introducción del nuevo material cromosómico en la población, tal y como sucede con sus equivalentes biológicos. Al igual que el cruce, la mutación se maneja como un porcentaje que indica con qué frecuencia se efectuará, aunque se distingue de la primera por ocurrir mucho más esporádicamente. Este bajo por ciento en la probabilidad de mutación evita oscilaciones en el promedio de los valores objetivos de la población. Implementamos la mutación probabilidad fija. La mutación busca mejorar aún más las soluciones creadas con los dos operadores anteriores. Se puede decir entonces que los algoritmos genéticos utilizan soluciones que por su evaluación han demostrado ser buenas para combinarlas y producir otras todavía mejores.

La hibridación se produce porque cada 10 generaciones se ejecuta el algoritmo de búsqueda local para diferentes porciones de la población según el caso.
Esto hace que varia la explotación de las solución y que el tiempo de ejecución de los algoritmos varié ya que uno tendrá que realizar mas búsqueda locales que el otro. Esto hace que el algoritmo que ejecute mas búsqueda locales sea el que explote mejor el entorno de solución y que al contar como evaluaciones las realizadas por las búsquedas locales se reduzca el numero de regeneraciones de la población que al ser un algoritmo genético generacional realizara con toda la población. Por lo tanto el algoritmo memético que realice mas búsquedas locales probablemente consiga mejores resultado y seguro mejora los tiempos del que realice menos búsquedas locales.

El objetivo de esta hibridación es la de producir una explotación del entorno de búsquedas de soluciones, el empleo de la técnica de búsqueda local proporciona una intensificación de la exploración de entornos a los que pertenecen las soluciones.

3. Información Algoritmos Meméticos para el problema del Clustering.

Se adjunta la documentación correspondiente al algoritmo genético generacional elitista. Como ya se comento los cromosomas son las posibles soluciones o configuraciones de clusters a las que cada 10 generaciones se les aplicara una búsqueda local. La búsqueda local se efectuara para la toda la población de cromosomas en uno de los casos y para un porcentaje que vendrá dado por una probabilidad de 0.1 para cada cromosoma de la población, en el otro caso.

En el primero el objetivo es intensificar la explotación del espacio de búsqueda de soluciones de nuestra población.

Para este problema se utilizan un operador de mutación y otro de cruce. El operador de cruce y mutación se ejecutaran con cierta probabilidad que sera fija durante toda la ejecución.

Las búsqueda explotan las zonas que según los algoritmos genéticos son prometedoras.

Representación de orden: Se seguirá la representación de orden basada en una permutación $\pi = [\pi(1), ..., \pi(m)]$ de los patrones del conjunto M. Los k primeros patrones del cromosoma se consideran los centroides (medoides) iniciales de los k clusters. Los m-k patrones restantes se asignan a cada cluster según la regla del prototipo más cercano, actualizando los centroides después de cada asignación.
Cada miembro de la poblacion tendrá esta forma

```
typedef struct {
        TipoCromosoma Gene; //permutación (representación de orden)
        double objetivo; // evaluación
        int cambio; // bool , si ha sido evaluado 1, si no 0
} ESTRUCTURA;
```

Operador de mutación: Se considerará el operador de intercambio para representación de orden, explicado en las transparencias del Seminario 3.a. Para aplicarlo, se escogerá aleatoriamente otro gen con el que intercambiar el patrón existente en el gen a mutar.
El tamaño de la población será de 10 cromosomas. La probabilidad de cruce será 0,7 en el AGG. La probabilidad de mutación (por gen) será de 0,01 en ambos casos. El criterio de parada en las dos versiones del AG consistirá en realizar 20000 evaluaciones de la función objetivo. Se realizarán cinco ejecuciones sobre cada caso del problema

```
MutacionAGG()
        Generamos población a mutar
                elegimos un cromosoma
                elegimos dos genes
                sustituimos uno por otro
                marcamos cromosoma como cambiado.
fin MutacionAGG()
```

cruce :

Esquema de evolución: se seleccionará una población de padres del mismo tamaño que la población genética.

SeleccionAGG()
for(0,tamaño de la poblacion,1)
elegir dos
torneo binario
fin del for
fin seleccionAGG

Además aquí se aplica el torneo binario tantas veces como el tamaño de la población, incluyendo en este método el operador de selección.

Esquema de reemplazamiento: En el esquema generacional, la población de hijos sustituye automáticamente a la actual. Para conservar el elitismo, si la mejor solución de la generación anterior no sobrevive, sustituye directamente la peor solución de la nueva población.

ReemplazarAGG()
Buscamos la mejor solución de nuestra población.
Comprobamos si la población a cambiado.
si a cambiado y la peor solución es menor
sustituimos el peor de la nueva por el mejor de nueva.
fin ReemplazarAGG

Operador de cruce: El operador de cruce empleado el el OX. Este operador creados dos hijos a partide de dos cromosomas padre. Se eligen dos padres con el operador de selección, se producen dos hijos. Para el primer hijo se eligen se coge parte de la representaron del cromosoma por ejemplo la parte intermedia, esto formara parte del hijo tal cual, el resto de valores que haya el padre 1 y padre 2 que no se hayan introducido se colocan en el orden en que aparecen y se introducen en el hijo 1 a continuación del ultimo valor introducido, cuando se llegue al final se terminan introduciendo los que falten por el principio. Con el hijo 2 pasa igual pero el padre al que pertenece parte de su representación sera el padre 2.

4. Valoración de los algoritmos.

Valores obtenidos por k-medias de la practica 1.

jose-ia@joseia-Satellite-A300:~/Escritorio/sofware/k-medias$ ./k-medias wdbc.cfg
Resumen:
Media: 200971263017607392.000000. Maximo: 200971263017607392.000000. Minimo: 200971263017607392.000000

jose-ia@joseia-Satellite-A300:~/Escritorio/sofware/k-medias$ ./k-medias yeast.cfg

Resumen:
Media: 37.785089. Maximo: 37.785089. Minimo: 37.785089

jose-ia@joseia-Satellite-A300:~/Escritorio/sofware/k-medias$ ./k-medias yeast.cfg

AM(-10,0.1)-k-medias

Este explora mejor el entorno aunque no consigue explotarlo lo suficiente.

AM-(10,0.1)	Aggregation		Yeast	
	J	Tiempo	*J*	Tiempo
Ejecución 1	9433,41	62,67	37,47	116,91
Ejecución 2	9437,82	65.74	37,71	138,85
Ejecución 3	9435,86	70,36	37,79	167,95
Ejecución 4	9434,74	71,08	37,68	186,40
Ejecución 5	9434,84	57,33	37,62	199,62
Mejor	9437,82	---	37,79	---
Peor	9433,41	---	37,47	---
Media	9435,33	65,36	37,66	161,95
Desv. típica	1,64	6,57	0,12	33,97

En los archivos de prueba yeast y aggregation los algoritmos meméticos son capaces de mejor en todos los casos de ejecución a los valores obtenidos por k-medias. Esto es debido a que los meméticos son capaces de explorar mejor el entorno de búsqueda de la solución ya que a parte del operador de mutación y de cruce, incorporan búsquedas locales de las soluciones que forman nuestra población de cromosomas, de esta forma explotan mejor el entorno, pudiendo llegar como en los caso de yeast y aggregation a mejorar las ejecuciones de k-medias. En cambio el coste de tiempo es mucho mayor en los algoritmos meméticos, ya que el orden de el k-medias es de 1 segundo y los meméticos pueden llegar casi a 200 segundos.

Para el archivo wdbc la convergencia es rápida y se cae en el mismo óptimo que lo hace el k-medias no pudiendo mejor la solución a pesar del coste de tiempo que supone su ejecución.

AM(-10,1.0) Vs k-medias

Este algoritmo explota los entornos de las soluciones que unido a la diversificación(explotación) de los algoritmos genéticos consigue mejorar los resultados del anterior.

AM-(10,1.0)	Aggregation		Yeast	
	J	Tiempo	*J*	Tiempo
Ejecución 1	9432.28	17,54	37,62	40,72
Ejecución 2	9435,68	17,05	36,41	37,38
Ejecución 3	9432,28	18,82	37,82	30,34
Ejecución 4	9435,73	16,54	37,29	27,06
Ejecución 5	9432,28	17,04	37,59	45,65
Mejor	9435,73	---	37,82	---
Peor	9432,28	---	36,41	---
Media	9433,99	17,40	37,35	36,23
Desv. típica	1,98	0,87	0,56	7,57

En este caso el coste de tiempo se ve disminuido al reducirse el numero de regeneraciones de la población Las búsquedas locales permiten que se explore mejor el entorno. Ya que el numero de evaluaciones considera también las búsquedas locales ejecutar, mas búsquedas locales supoonen dedicar menor tiempo a regenerar la población que es lo que causa que se produzca un coste de tiempo tan alto.

Conclusión algoritmos meméticos.

Estos algoritmos presentan mejoras en problemas con variables muy dispersas. Son capaces de equilibrar de forma sencilla exploración y explotación del entorno de soluciones. Los algoritmos genéticos de por si son capaces de explorar bien el entorno, gracias al operador de mutación, que permite que se realicen, aunque con poca frecuencia, mutaciones que producen grandes cambios en el domino de soluciones (población). En particular este es un algoritmo genético generacional elitista por lo que todos los individuos se sustituyen por los nuevos, excepto el mejor el de la población que prevalece si mejora a alguno de la nueva generación, con lo que mantenemos en nuestra población (espacio de soluciones) la mas prometedora de la ultima generación. La regeneración de toda la población hace que se explore mejor todo el dominio del problema que unido a búsquedas locales de esas soluciones es capas de mejor los resultados de los genéticos de forma individual.

Las búsquedas locales son mas rápidas que la regeneración de la población por lo que el algoritmo que realice mas realizara menor regeneraciones. Esto supondrá que el que realice mas búsquedas locales pueda tardar menos y además explorar de forma profunda el espacio de solución de nuestro problema mejorando en casi todos los casos de los dos ficheros al que realiza menos búsquedas locales. En cambio el que realice mas regeneraciones explorará mejor el entorno de soluciones.